Primeiro Trabalho de Inteligência Artificial

Matheus Gomes Arante de Souza

Abstract

Trabalho de Inteligência Artificial com o objetivo de comparar experimentalmente metaheurísticas aplicadas ao Problema da Mochila.
Keywords: Inteligência Artificial, Metaheurísticas, Hill Climbing, Beam Search, Simulated Annealing, GRASP, Genetic Algorithm

1. Introdução

Este trabalho consiste em analisar experimentalmente as metaheurísticas Hill Climbing, Beam Search, Simulated Annealing, Genetic Algorithm e GRASP quando submetidas ao Problema da Mochila. Para isso, foram definidas 20 instâncias distintas de problemas, sendo que metade foi utilizada na etapa de treino, da qual consiste em selecionar o melhor hiperparâmetro para cada metaheurística, e a outra metade dos problemas para a etapa de teste, onde é feita a comparação entre as metaheurísticas, considerando os hiperparâmetros previamente selecionados.

2. Descrição do Problema da Mochila

O Problema da Mochila é um problema NP-Completo e ilustra uma situação onde se quer preencher uma mochila com diferentes itens, que tem associados a si valor e peso. O objetivo consiste em maximizar o valor dos itens que estão na mochila, sem ultrapassar sua capacidade de peso.

Neste trabalho, foi utilizada a versão do Problema da Mochila em que não há restrição da quantidade máxima que se pode carregar de um determinado item.

3. Descrição dos Métodos Utilizados

Nesta seção serão apresentadas as descrições das metaheurísticas utilizadas.

3.1. Hill Climbing

O Hill Climbing é uma metaheurística que utiliza estratégia gulosa para encontrar a solução dos problemas, ou seja, não há retrocesso na busca. Para o problema da mochila, a cada passo do algoritmo, ele selecionará o item que maximizará o valor da mochila até que se obtenha uma solução completa.

3.2. Beam Search

O algoritmo Beam Search realiza uma busca em amplitude. Seu funcionamento ocorre da seguinte forma: existe uma fila de estados, e a cada iteração é realizada a expansão de todos os estados desta fila. A partir disso, são selecionados os m melhores estados (os que maximizam o valor da mochila). O procedimento é repetido até que se obtenha uma solução completa.

3.3. GRASP

A ideia do algoritmo GRASP (Greedy Randomized Adaptive Search) gira em torno de iterar uma quantidade de vezes realizando duas instruções: construir uma solução s e depois submeter s a um algoritmo de busca local para obter melhorias na solução e armazenar a melhor solução encontrada.

A solução s, que posteriormente é aplicada uma busca local, vem de um método chamado GreedyRandomConstruct que pode ser comparado a um "Hill Climbing aleatório", ao passo que a cada iteração, enquanto o Hill Climbing seleciona o melhor elemento avaliado, GreedyRandomConstruct seleciona aleatoriamente um dos m melhores elementos avaliados.

3.4. Simulated Annealing

O algoritmo Simulated Annealing, que é uma analogia com processo físico de resfriamento de sólido superaquecido, funciona da seguinte maneira: a partir uma solução inicial s, e até que um critério de parada seja atingido, no caso desta

implementação $temperatura < 1$, é gerada a vizinhança de s, e até que se atinja um limite de iterações, seleciona-se um estado aleatório s' da vizinhança. Se s' for melhor do que s, $s \leftarrow s'$ e a vizinhança é atualizada. Caso contrário, ainda assim há uma probabilidade de que isso ocorra, possibilitando o movimento para uma solução pior. Entretanto, quão pior a solução, menor a chance dela ser aceita. Durante todo o procedimento o melhor estado encontrado é armazenado.

3.5. Genetic Algorithm

De maneira geral, o Genetic Algorithm funciona da seguinte maneira: é gerada uma população inicial P (um conjunto de soluções para o problema da mochila), são selecionados os indivíduos mais aptos de P e a partir destes são gerados novos indivíduos através de crossover e mutação, formando uma população P'. Os k elementos inválidos de P' são substituídos pelos k melhores elementos de P, formando uma nova população. Este procedimento é repetido até que um critério de parada seja atingido. Nesta implementação, por exemplo, foi utilizado um limite de 30 gerações. A seleção dos indivíduos mais aptos foi feita por meio do método da roleta.

4. Descrição dos Experimentos Realizados

O procedimento experimental foi realizado em duas partes: Treino e Teste. De maneira geral, o treino foi o procedimento utilizado para selecionar o melhor hiperparâmetro, dentre as diversas combinações de hiperparâmetros que existiam inicialmente para cada metaheurística. O Teste, por sua vez, foi o procedimento utilizado para comparar o comportamento das metaheurísticas com os hiperparâmetros selecionados no treino.

As metaheurísticas foram submetidas as etapas de treino e teste a 20 diferentes instâncias de problemas (10 para cada etapa), de modo que cada problema estava associado à capacidade da mochila e uma coleção de itens, cada um deles descrito por uma tupla com valor e tamanho.

4.1. Treino

Nesta seção serão apresentadas as descrições das etapas do treino e os resultados obtidos para cada metaheurística. Como a metaheurística Hill Climbing não possui hiperparâmetros, não participa desta etapa.

4.1.1. Descrição do Algoritmo de Treino

O objetivo da etapa de treino é selecionar o melhor hiperparâmetro para cada metaheurística, e para isso foi utilizado um algoritmo, cuja ideia gira em torno de para cada metaheurística, gerar todas as suas possíveis combinações de hiperparâmetros e submeter cada um deles ao conjunto de problemas de treino. Após isso, normalizar os resultados obtidos para cada problema e, por fim, calcular para cada hiperparâmetro a média de seus respectivos resultados normalizados em cada problema. O hiperparâmetro com a melhor média de resultados normalizados é selecionado para ser utilizado na etapa de Teste. A seguir é apresentado o pseudo-código do algoritmo descrito:

Algorithm 1: Algoritmo de Treino

```
for cada metaheurística que necessita de ajuste de hiperparâmetros do
    for cada comb. de valores de hiperparâmetros da busca em grade do
        for cada problema do conjunto de treino do
            Executar metaheurística no problema até condição de parada
            Armazenar resultado alcançado e tempo de execução
    for cada problema do conjunto de treino do
        Normalizar resultados alcançados pela metaheurística
    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
    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
```

4.1.2. Descrição dos Problemas de Treino

Para o treino foram selecionadas dez instâncias distintas para o problema da mochila. Cada instância está relacionada a um tamanho de mochila (t) e um conjunto de itens (vt) que possuem valor e tamanho.

A seguir são apresentados os problemas do conjunto de treino.

P1: t = 19 vt = [(1,3),(4,6),(5,7)]

P3: t = 58 vt = [(1,3),(4,6),(5,7),(3,4)]

P4: t = 58 vt = [(1,3),(4,6),(5,7),(3,4),(8,10),(4,8),(3,5),(6,9)]

P6: t = 58 vt = [(1,3),(4,6),(5,7),(3,4),(8,10),(4,8),(3,5),(6,9),(2,1)]

P8: t = 120 vt = [(1,2),(2,3),(4,5),(5,10),(14,15),(15,20),(24,25),(29,30),(50,50)]

P9: t = 120 vt = [(1,2),(2,3),(3,5),(7,10),(10,15),(13,20),(24,25),(29,30),(50,50)]

P11:t = 120 vt = [(24,25),(29,30),(50,50)]

P14: t = 138 vt = [(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: t = 13890000 vt = [(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: t = 45678901 vt = [(1,3),(4,6),(5,7),(3,4),(2,6),(1,2),(3,5),(7,10), (10,15), (13,20),(15,20)]

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

A cada metaheurística estavam associados diferentes hiperparâmetros, dos quais foram combinados de todas as maneiras possíveis para realizar a busca em grade em cada metaheurística.

A seguir são apresentados os hiperparâmetros de cada metaheurística e a lista de valores que cada um deles pode assumir:

- Beam Search
  - Número de melhores elementos: [10, 25, 50, 100]
- Simulated Annealing
  - Temperatura: [500, 100, 50]
  - Alpha: [0.95, 0.85, 0.7]
  - Número de iterações: [350, 500]
- GRASP
  - Número de iterações: [50, 100, 200, 350, 500]
  - Número de melhores elementos: [2, 5, 10, 15]
- Genetic Algorithm
  - Tamanho da população: [10, 20, 30]
  - Taxa de crossover: [0.75, 0.85, 0.95]
  - Taxa de mutação: [0.10, 0.20, 0.30]

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

Nesta seção serão apresentados os boxplots de valores normalizados e tempos dos 10 melhores hiperparâmetros de cada metaheurística. A escolha do melhor hiperparâmetro foi feita baseada naquele que obteve maior média de resultados normalizados.

Nos boxplots, os triângulos verdes representam a média, os losangos pretos representam outliers e a faixa colorida representa a variação de valores dos problemas para determinada combinação de hiperparâmetros.

**As imagens estão em formato vetorial, portanto, é possível aplicar zoom para uma visualização mais detalhada dos boxplots.*

- Beam Search

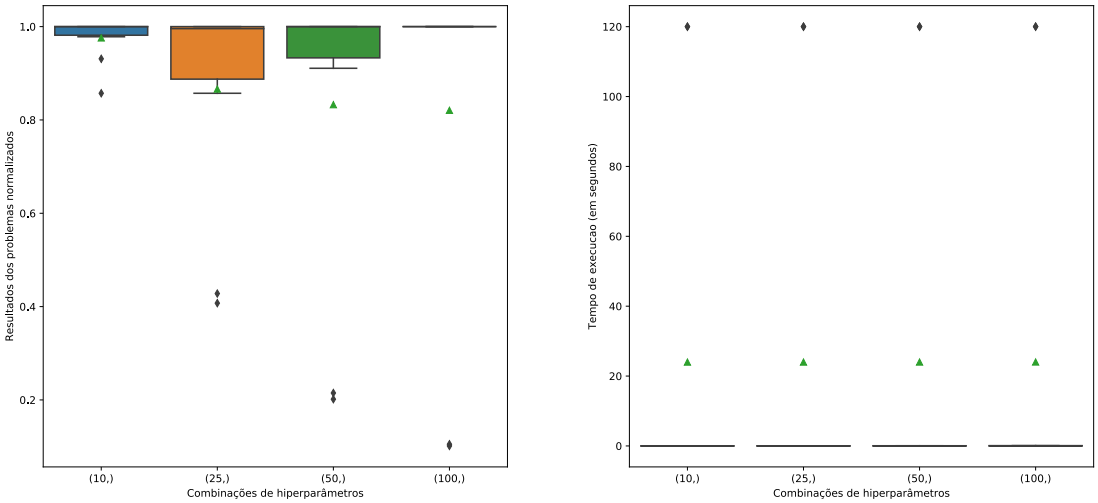

Figura 1: Resultados dos problemas normalizados e Tempos de execução (em segundos), respectivamente, para o Beam Search

(NUM. MELHORES)	MÉDIA RESULTADOS NORMALIZADOS
(10)	0.975782
(25)	0.866211
(50)	0.832713
(100)	0.820645

Tabela 1: Dez melhores hiperparâmetros e suas respectivas médias.

Hiperparâmetro escolhido para o teste: (10)

- Simulated Annealing

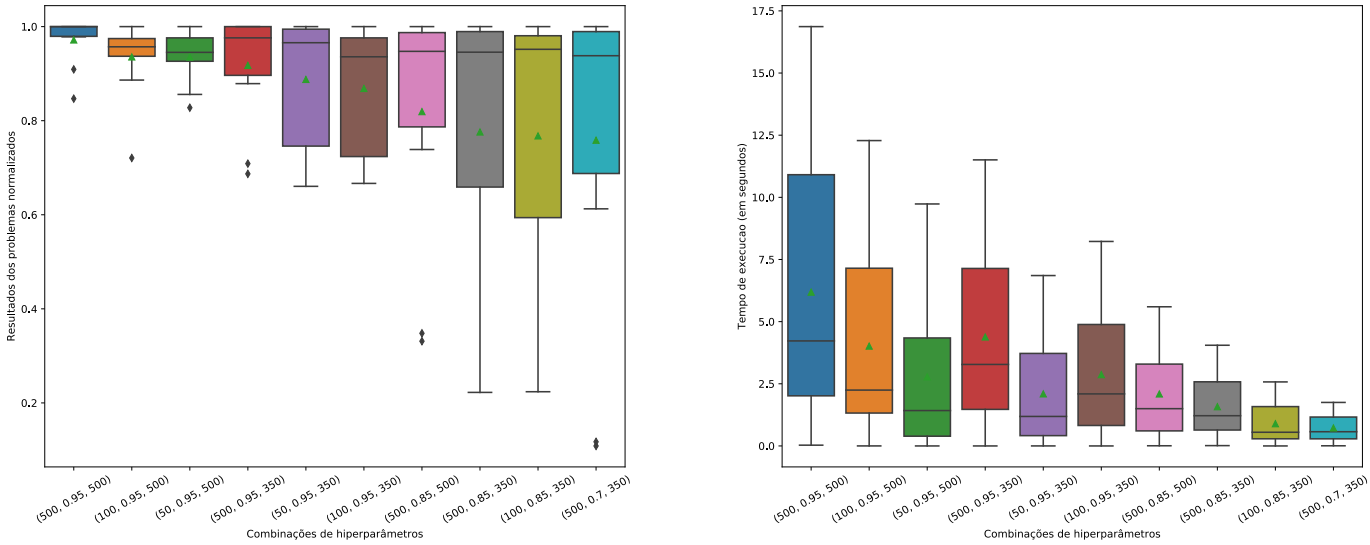

Figura 2: Resultados dos problemas normalizados e Tempos de execução (em segundos), respectivamente, para o Simulated Annealing

(TEMP., ALPHA, NUM. ITER.)	MÉDIA RESULTADOS NORMALIZADOS
(500, 0.95, 500)	0.971681
(100, 0.95, 500)	0.935574
(50, 0.95, 500)	0.935249
(500, 0.95, 350)	0.917579
(50, 0.95, 350)	0.88775
(100, 0.95, 350)	0.868531
(500, 0.85, 500)	0.819293
(500, 0.85, 350)	0.776037
(100, 0.85, 350)	0.767725
(500, 0.7, 350)	0.758539

Tabela 2: Dez melhores hiperparâmetros e suas respectivas médias.

Hiperparâmetro escolhido para o teste: (500, 0.95, 500)

- GRASP

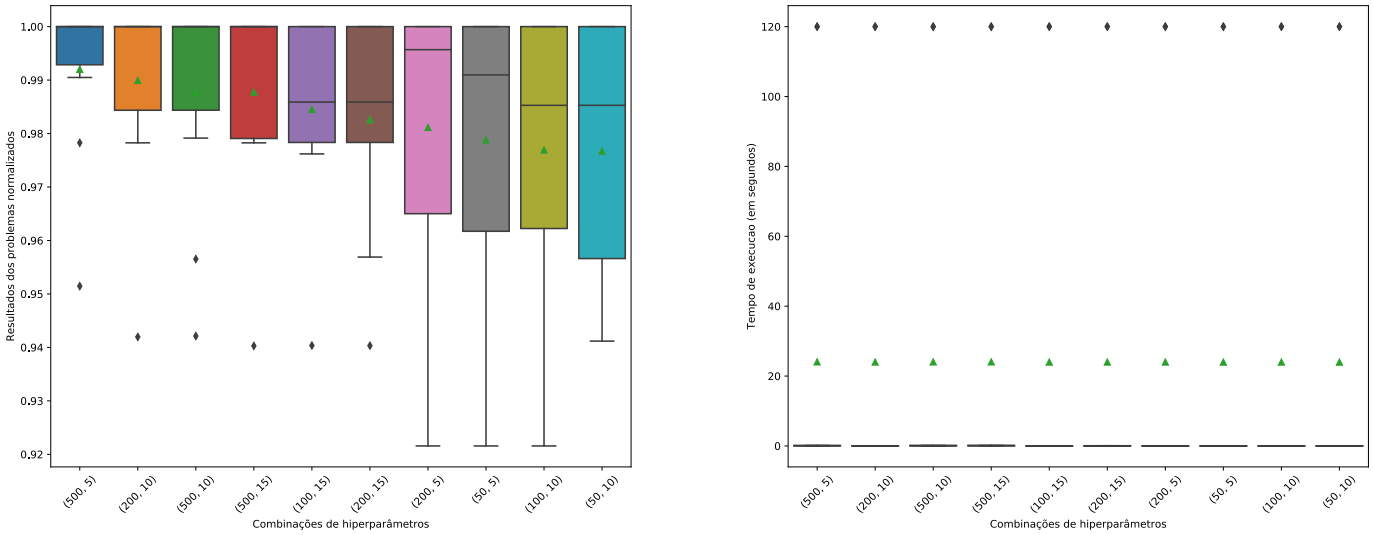

Figura 3: Resultados dos problemas normalizados e Tempos de execução (em segundos), respectivamente, para o GRASP

(NUM. ITER., NUM. MELHORES)	MÉDIA RESULTADOS NORMALIZADOS
(500, 5)	0.99202
(200, 10)	0.989936
(500, 10)	0.987779
(500, 15)	0.987756
(100, 15)	0.984513
(200, 15)	0.982581
(200, 5)	0.981135
(50, 5)	0.978808
(100, 10)	0.976935
(50, 10)	0.976728

Tabela 3: Dez melhores hiperparâmetros e suas respectivas médias.

Hiperparâmetro escolhido para o teste: (500, 5)

- Genetic Algorithm

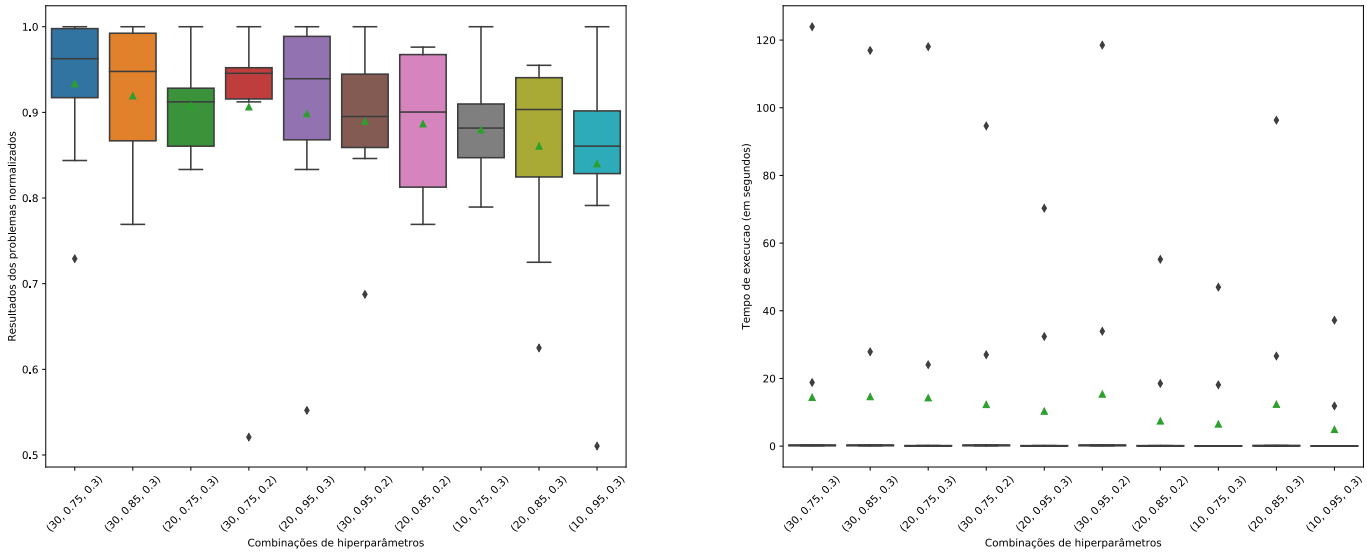

Figura 4: Resultados dos problemas normalizados e Tempos de execução (em segundos), respectivamente, para o Genetic Algorithm

(TAM. POP, TX. CROSS., TX. MUT.)	MÉDIA RESULTADOS NORMALIZADOS
(30, 0.75, 0.3)	0.933481
(30, 0.85, 0.3)	0.919336
(20, 0.75, 0.3)	0.907692
(30, 0.75, 0.2)	0.906475
(20, 0.95, 0.3)	0.898493
(30, 0.95, 0.2)	0.889797
(20, 0.85, 0.2)	0.886643
(10, 0.75, 0.3)	0.879679
(20, 0.85, 0.3)	0.860735
(10, 0.95, 0.3)	0.839972

Tabela 4: Dez melhores hiperparâmetros e suas respectivas médias.

Hiperparâmetro escolhido para o teste: (30, 0.75, 0.3)

4.1.5. Análise dos resultados alcançados

- Beam Search

  Os outliers que aparecem no boxplot são referentes aos problemas P17 e P20, as maiores instâncias do conjunto de treino. É interessante observar que embora o hiperparâmetro (100) tenha um boxplot bem achatado, indicando que na maioria dos problemas ele obteve bom resultado, seu desempenho para os problemas maiores foi inferior aos demais, tornando sua média a pior de todas. O hiperparâmetro (10) se mostra uma boa escolha para o teste, pois além de possuir a maior média, demonstra pouca variação nos valores da média normalizada dos problemas e que nos maiores problemas (P17 e P20), obteve um resultado superior aos demais.

  Com relação ao tempo gasto, as combinações apresentaram tempos de execução semelhantes, e acabaram parando a execução pelo limite de dois minutos apenas nos problemas P17 e P20.

- Simulated Annealing

  O hiperparâmetro que apresentou a melhor média foi o (500, 0.95, 500), e de acordo com o boxplot podemos verificar que, de fato, quando comparado aos outros, apresentou resultados mais concentrados.

  Entretanto, é possível destacar que utilizar parâmetro alpha = 0.95 pode ser uma boa escolha, pois várias combinações que utilizavam este valor, apareceram dentre as melhores. Além disso, os três melhores hiperparâmetros utilizaram alpha = 0.95 e 500 iterações, com temperaturas iniciais diferentes. Isso pode ser um indicativo de que para os problemas do conjunto de treino, independente da temperatura inicial, associar um fator de decaimento alpha de tal modo que a temperatura caia lentamente a um número grande de iterações pode gerar bons resultados.

  Nenhuma das combinações atingiu o tempo limite. O melhor hiperparâmetro foi o que utilizou em média maior tempo de execução, porém, no problema em que demorou mais, gastou apenas cerca de 17 segundos.

- GRASP

  De maneira geral, os hiperparâmetros obtiveram médias bem altas, sendo que a diferença entre as médias do melhor e do 10º melhor foi pequena: o melhor hiperparâmetro (500, 5) apresentou aproximadamente 0.99 enquanto 10º melhor (50, 10) aproximadamente 0.97.

  Apesar da média das combinações ser bem próxima, pelo boxplot é possível verificar que o melhor hiperparâmetro demonstrou menor variação do que os outros, indicando que um número de iterações grande e "número de melhores"pequeno, de fato, é a melhor escolha.

  Além disso, os tempos de execução foram muito parecidos, atingindo o limite de tempo apenas nos maiores problemas, P17 e P20.

- Genetic Algorithm

  A partir do boxplot, é possível ver a menor variação de resultados foi no hiperparâmetro (30, 0.75, 0.2). Apesar disso, o que obteve maior média, e portanto é o melhor hiperparâmetro, foi o (30, 0.75, 0.2).

  É interessante observar que hiperparâmetros com a maior taxa de mutação (0.30) e a menor taxa de crossover (0.75) figuraram dentre as melhores médias. Vale ressaltar que os hiperparâmetros que utilizavam populações muito pequenas (de tamanho 10) não se destacaram tanto.

  Dentre as diferentes combinações, o tempo de execução dos problemas foi bem parecido, com a maioria dos tempos concentrados e P17 e P20 demandando um pouco mais de tempo. No melhor hiperparâmetro, P20 chegou a bater o tempo limite.

4.2. Teste

Nesta seção serão apresentadas as descrições das etapas do teste e os resultados obtidos.

4.2.1. Descrição do Algoritmo de Teste

O objetivo da etapa de teste é avaliar as metaheurísticas quando submetidas ao conjunto de problemas de teste com os valores de hiperparâmetros selecionados na etapa de treino, e para isso foi utilizado um algoritmo que funciona da seguinte forma: Executar as metaheurísticas para todos os problemas do conjunto de teste e armazenar os resultados. Com base nisso, para cada problema, normalizar os resultados alcançados pelas metaheurísticas. E por fim, com os resultados normalizados em mãos, calcular diversas métricas e criar ranqueamentos a partir deste conjunto de dados para posteriormente avaliar o desempenho das metaheurísticas.

O pseudo-código do algoritmo de treino é descrito a seguir.

Algorithm 2: Algoritmo de Teste

```
1  for cada metaheurística do
2      for cada problema do conjunto de teste do
3          Executar metaheurística no problema até condição de parada
4          Armazenar resultado alcançado e tempo de execução
5  for cada problema do conjunto de teste do
6      Normalizar resultados alcançados pelas metaheurísticas
7  Obter média e desvio padrão dos resultados normalizados de cada
     metaheurística
8  Gerar tabela contendo média e desvio padrão absolutos e normalizados,
     e média e desvio padrão dos tempos de execução de todas as
     metaheurísticas
9  for cada problema do conjunto de teste do
10     Fazer ranqueamento das metaheurísticas segundo resultado absoluto
11 Obter média dos ranqueamentos das metaheurísticas segundo resultado
     absoluto
12 Apresentar as metaheurísticas em ordem crescente de média de
     ranqueamento
13 Obter média dos resultados normalizados de cada metaheurística
14 Apresentar as metaheurísticas em ordem crescente de média dos
     resultados normalizados
15 Gerar boxplot dos resultados normalizados alcançados pelas
     metaheurísticas
16 Gerar boxplot dos tempos alcançados pelas metaheurísticas
```

4.2.2. Descrição dos Problemas de Teste

Para o teste foram selecionadas dez instâncias distintas para o problema da mochila. Cada instância está relacionada a um tamanho de mochila (t) e um conjunto de itens (vt) que possuem valor e tamanho. Todos os problemas do conjunto de teste são diferentes dos problemas do conjunto de treino.

A seguir são apresentados os problemas do conjunto de teste.

P2: t = 192 vt = [(1,3),(4,6),(5,7)]

P5: t = 287 vt = [(1,2),(2,3),(3,4),(4,5),(5,6),(6,7),(7,8),(8,9),(9,10)]

P7: t = 120 vt = [(1,2),(2,3),(4,5),(5,10),(14,15),(13,20),(24,25),(29,30),(50,50)]

P10: t = 1240 vt = [(1,2),(2,3),(3,5),(7,10),(10,15),(13,20),(24,25),(29,30),(50,50)]

P12: t = 104 vt = [(25,26),(29,30),(49,50)]

P13: t = 138 vt = [(1,3),(4,6),(5,7),(3,4), (2,6), (2,3), (6,8)]

P15: t = 13890 vt = [(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: t = 13890 vt = [(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: t = 190000 vt = [(1,3),(4,6),(5,7)]

P19: t = 4567 vt = [(1,3),(4,6),(5,7),(3,4),(2,6),(1,2),(3,5),(7,10),(10,15),(13,20),(15,20)]

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

	METAHEURÍSTICAS				
	Beam Search	Hill Climbing	GRASP	Simulated Annealing	Genetic
MÉDIA ABS.	16587	16585.8	15879.3	6623.4	15597.9
DESVIO PADRÃO ABS.	42153	42153.3	40010.4	11463	40044.1
MÉDIA NORM.	0.991818	0.989762	0.97699	0.904791	0.891419
DESVIO PADRÃO NORM.	0.0225793	0.0229178	0.0209245	0.225842	0.0472912
MÉDIA TEMPO (seg)	0.711999	0.147622	16.0509	6.29606	0.274411
DESVIO PADRÃO TEMPO (seg)	1.61097	0.33462	46.7419	5.28078	0.164846

Tabela 5: Média e desvio padrão absolutos e normalizados, e média e desvio padrão dos tempos de execução de todas as 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

PROBLEMA	Hill Climbing	Beam Search	Simulated Annealing	GRASP	Genetic
P2	1	1	1	4	5
P5	1	1	4	3	5
P7	1	1	4	1	5
P10	1	1	4	3	5
P12	4	3	1	1	5
P13	2	1	4	2	5
P15	1	1	4	3	5
P16	4	3	1	2	5
P18	1	1	5	3	4
P19	1	1	3	4	5

Tabela 6: Colocações de cada metaheurística nos ranqueamentos dos problemas.

POSIÇÃO	METAHEURÍSTICA	MÉDIA DOS RANQUEAMENTOS
1	Beam Search	1.85
2	Hill Climbing	2.2
3	GRASP	2.8
4	Simulated Annealing	3.25
5	Genetic Algorithm	4.9

Tabela 7: Média dos ranqueamentos de cada metaheurística.

4.2.5. Apresentação de tabela contendo os resultados normalizados para cada problema de teste e metaheurísticas em ordem crescente de média dos resultados normalizados

PROBLEMA	Hill Climbing	Beam Search	Simulated Annealing	GRASP	Genetic
P2	1.0	1.0	1.0	0.9852	0.8970
P5	1.0	1.0	0.9651	0.9844	0.9496
P7	1.0	1.0	0.9745	1.0	0.8644
P10	1.0	1.0	0.9328	0.9878	0.9012
P12	0.98	0.99	1.0	1.0	0.79
P13	0.9902	1.0	0.9805	0.9902	0.9126
P15	1.0	1.0	0.9647	0.9657	0.8554
P16	0.9273	0.9281	1.0	0.9684	0.8826
P18	1.0	1.0	0.2648	0.9495	0.9495
P19	1.0	1.0	0.9652	0.9381	0.9115

Tabela 8: Resultados normalizados para cada problema de teste.

METAHEURÍSTICA	MÉDIA NORMALIZADA
Genetic Algorithm	0.891419
Simulated Annealing	0.904791
GRASP	0.97699
Hill Climbing	0.989762
Beam Search	0.991818

Tabela 9: Metaheurísticas em ordem crescente de média dos resultados normalizados.

4.2.6. Apresentação dos boxplots dos resultados normalizados obtidos por cada metaheurística

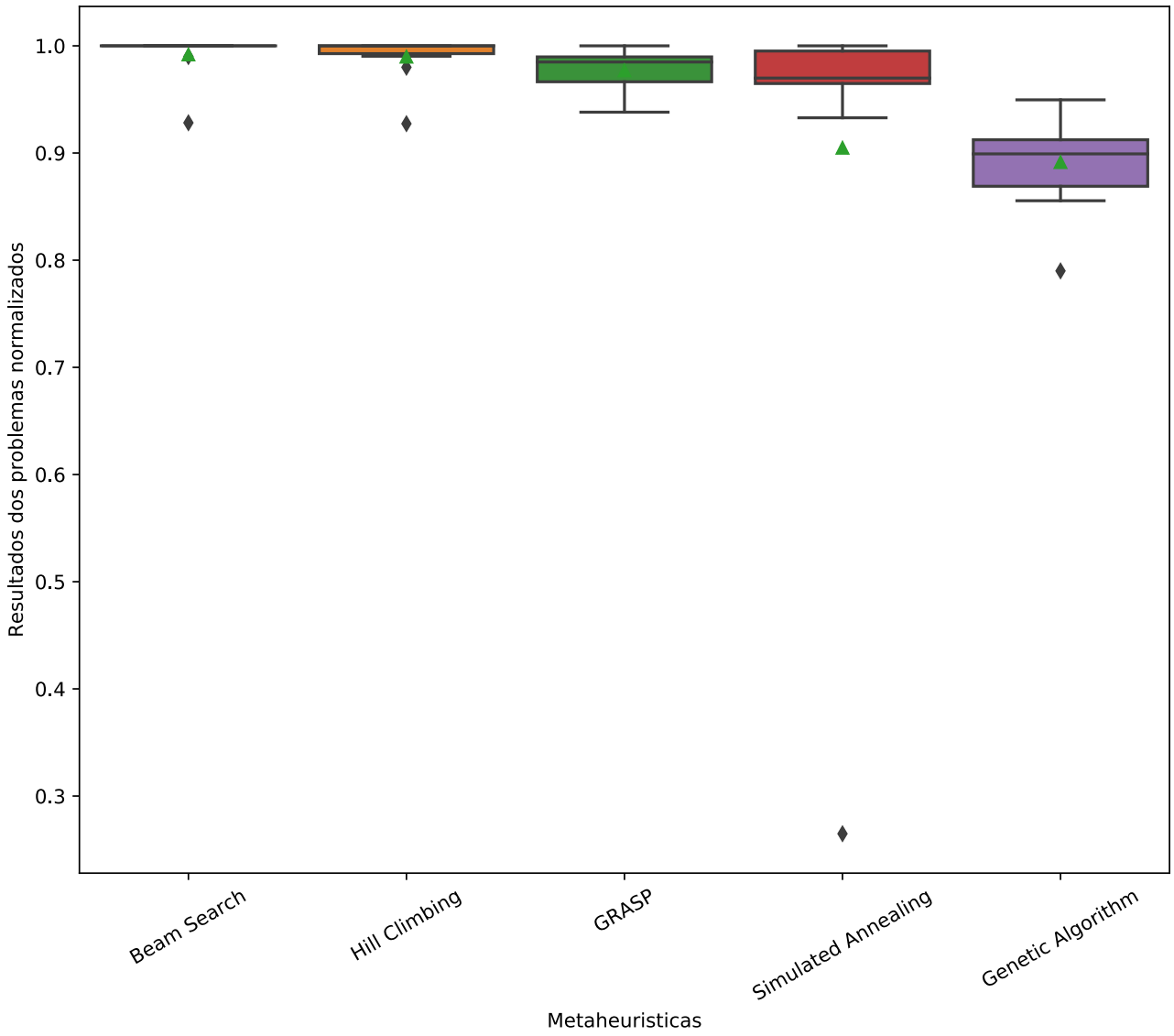

Figura 5: Resultados normalizados obtidos por cada metaheurística

4.2.7. Apresentação dos boxplots dos tempos de execução obtidos por cada metaheurística

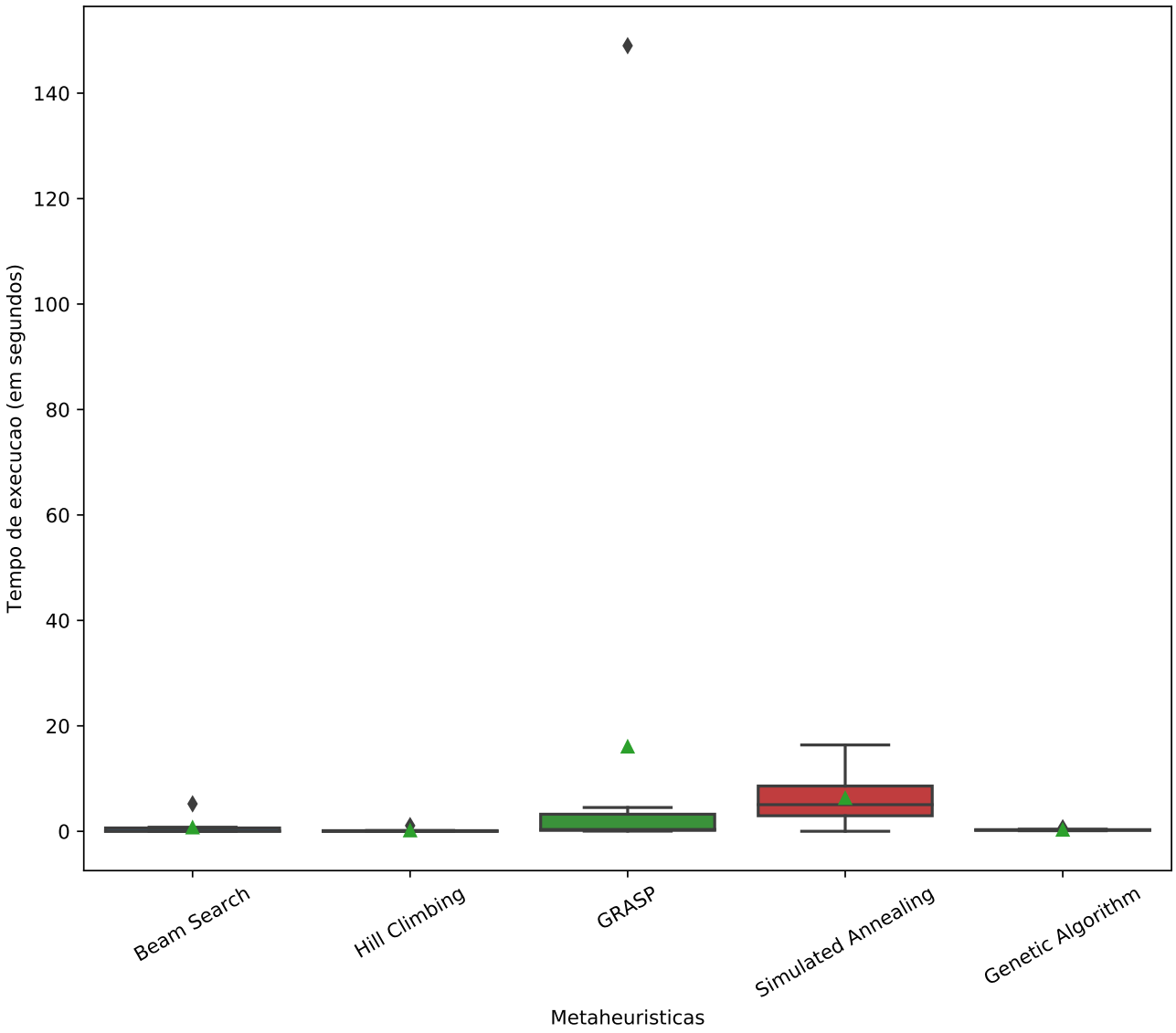

Figura 6: Tempos de execução (em segundos) obtidos por cada metaheurística

4.2.8. Análise dos resultados alcançados

Inicialmente, avaliando o tempo de execução gasto para as metaheurísticas a partir da tabela 5 e do boxplot dos tempos, nota-se que para o conjunto de problemas de treino, nenhuma delas atingiu o tempo limite de cinco minutos, ou seja, terminaram a execução dos problemas ao atingir algum critério de parada definido por cada metaheurística, tornando a avaliação dos resultados obtidos mais interessante.

Analisando a média absoluta dos resultados, o algoritmo Simulated Annealing apresenta uma média inferior aos demais, mas isso se dá devido ao desempenho ruim que obteve no problema P18 (Tabela 6), em que a mochila era muito grande. Entretanto, através do boxplot é possível ver que trata-se de um outlier, pois o gráfico está mais concentrado nos outros problemas, de maneira que em alguns deles onde a capacidade da mochila era menor, chegou a ocupar a primeira colocação no ranking.

O algoritmo Genético, por sua vez, obteve média e desvio padrão absolutos relativamente próximos aos demais, mas ao verificar a média normalizada (Tabela 9), percebe-se que esta foi a menor de todas. Isso se comprova ao verificar o boxplot dos resultados normalizados, que quando comparado aos outros, foi a de maior variação e também apresentou a pior média dos ranqueamentos. Ou seja, apesar de possuir uma média absoluta semelhante as demais, não apresentou resultados que se destacassem.

O GRASP foi aquele que apresentou maior média e desvio padrão de tempo de execução. Embora este fato tenha ocorrido, para todos os problemas manteve-se dentro do tempo limite e apresentou bons resultados de média absoluta e normalizada.

Por fim, Hill Climbing e Beam Search entregaram os melhores resultados em diversos quesitos. O boxplot dos resultados normalizados mostra com clareza que ambos tiveram baixa variação e média alta. Analisando os rankings de cada problema é possível notar que estes diversas vezes empataram na primeira colocação, mas no final o Beam Search atingiu maior média de ranqueamento.

5. Conclusões

Ao analisar os resultados obtidos nas etapas de treino e teste é possível perceber a importância de realizar a separação do conjunto de problemas de treino e de teste para evitar o superajuste dos hiperparâmetros.

Vale salientar que variedade de instâncias com tamanhos de mochila pequenos, médios ou grandes associados à listas com poucos ou muitos itens auxilia bastante na parte de treino, pois torna visível o comportamento das metaheurísticas e suas diversas combinações de hiperparâmetros quando submetidas a problemas de diferentes magnitudes.

Neste experimento, a metaheurística que obteve maior destaque foi o Beam Search. Entretanto, algoritmos como Simulated Annealing e Genetic Algorithm, que apresentaram os piores resultados do teste, possuem passos cuja implementação pode variar dependendo da interpretação, por exemplo a forma de gerar a vizinhança de um estado, realizar crossover e mutação. Portanto, pode ser que se esses passos fossem implementados de outra maneira, talvez estas metaheurísticas apresentassem resultados mais satisfatórios.

É interessante notar que algoritmos como Beam Search, Hill Climbing e GRASP que tem uma implementação menos sofisticada, se comparados aos algoritmos Simulated Annealing e Genético, obtiveram as maiores médias de resultados normalizados no treino, indicando que possuem um bom "custo/benefício", dependendo da qualidade que se espera dos resultados. Porém, se soluções com resultados muito bons sejam necessárias, como dito anteriormente, vale o esforço de se verificar e testar diversas formas para encontrar a melhor maneira de, por exemplo, gerar a vizinhança, realizar crossover e mutação nos algoritmos Simulated Annealing e Genético e maximizar seus resultados.

Referências

Slides sobre "Busca", disponibilizados pelo professor Flávio Miguel Varejão.