TC1 - PROBLEMA DA MOCHILA MÚLTIPLA

Adriano Rodrigues Alves
Julio Huang
Luís Fernando Leite França
Patrick Escorsi Silva

Novembro 2021

1 Introdução

O objetivo deste trabalho foi avaliar a qualidade das soluções geradas por heurísticas ingênuas + melhoria para o problema da mochila múltipla. Para isso, nos foi fornecido um algoritmo exato do tipo *branch-and-bound* e, ao decorrer do trabalho, foram implementadas duas heurísticas, uma denominada heurística gulosa e uma outra heurística randômica, e uma heurística de melhoria, denominada *Destroy and Repair*.

Descreveremos neste documento os detalhes sobre o projeto e como foi feita a implementação das partes fundamentais do trabalho, como as heurísticas e a melhoria, os testes e quais foram os resultados obtidos.

2 Heurísticas

Nesta seção nós descreveremos as duas heurísticas ingênuas implementadas, assim como a heurística de melhoria, para o desenvolvimento deste trabalho.

2.1 Heurística Gulosa

Para a heurística gulosa, havia algumas opções, mas a ideia, no fundo, era fazer uma ordenação seguindo um critério que parecia ser mais vantajoso na hora de preencher as mochilas. Dito isso, realizamos a ordenação por meio da função *qsort*, usando uma função de comparação auxiliar chamada *Comparador*. Para o critério de comparação, nós utilizamos três opções: ordenando de forma crescente pelo **valor** de cada item, ordenando de forma decrescente pelo **peso** de cada item e ordenando de forma crescente pela **razão** entre o valor e o peso de cada item. Depois disso, a heurística percorre toda a lista de itens e verifica se o item cabe ou não dentro de alguma mochila. Ou a heurística coloca o item na primeira mochila que couber ou o item será descartado. Por descartado, nos referimos ao item não ser mais verificado. Ao final, após passar por toda a lista de itens, a heurística entrega o subconjuntos dos itens selecionados, além do valor total da mochila. Após alguns testes e verificações, optamos por deixar apenas a ordenação dos itens pelo valor, por produzir um valor final da mochila melhor do que as outras ordenações propostas.

2.2 Heurística Aleatória

A heurística aleatória baseia-se em sortear aleatoriamente um item qualquer e ir preenchendo as mochilas até as suas respectivas capacidades. Caso o item sorteado tenha um peso maior que o suportado pela mochila selecionada, o algoritmo tentará encontrar uma outra mochila que o suporte, mas caso não encontre, este item é descartado, fazendo com que não seja escolhido novamente pelo função *RandomInteger*. Ao final, a heurística entrega o subconjunto dos itens selecionados, além do valor total da mochila.

2.3 Heurística de Melhoria

Na heurística de melhoria foi feito uma implementação baseada em uma combinação entre a heurística + métodos exatos. Para isso, implementamos a ideia do *Destroy and Repair*. Para a parte do *destroy*, implementamos uma heurística de propósito geral chamada RINS (*Relaxation Induced Neighborhood Search*), onde, dada a nossa solução inteira gerada por uma das heurísticas ingênuas e a solução gerada pelo PL,

é feito a retirada dos itens não pegos integralmente pela solução relaxada e fixado os itens que foram levados em comum entre as duas soluções de forma inteira. Em alguns casos, pode acontecer de o *destroy* retirar todos os itens das mochilas. Nesse caso, é feito outro *destroy*, mas retirando somente os itens que tiverem o valor de decisão maior que 0,1.

Para a parte do *repair*, é necessário fazer uma verificação antes. Quando o *destroy* retira todos os itens das mochilas, utilizamos a função *repair_rins* como auxiliar. A função seleciona outro subconjunto de itens para preencher as mochilas, de forma a não ultrapassar a capacidade de cada mochila. Em seguida, a função *destroy* é chamada novamente. Esse procedimento é necessário pois o passo seguinte é chamar o PLI, assim não podemos mandar as mochilas vazias porque a solução final não seria produzida por uma heurística.

Ao final, na reparação criamos um subconjunto do problema e armazenamos na variável *I_PLI* todos os itens que ainda não foram levados e o remanescente das capacidades de cada mochila, e o PLI é chamado para resolver esse subproblema.

3 Testes e resultados

Nesta seção nós apresentaremos quais testes foram realizados para o desenvolvimento do trabalho e quais resultados obtivemos a partir destes testes.

3.1 T1

A primeira verificação feita foi uma comparação entre o limitante superior (UB) gerado pelo PLI e a heurística gulosa, com e sem melhoria. Os gráficos abaixo mostram essa comparação para o conjunto de instâncias **t25**:

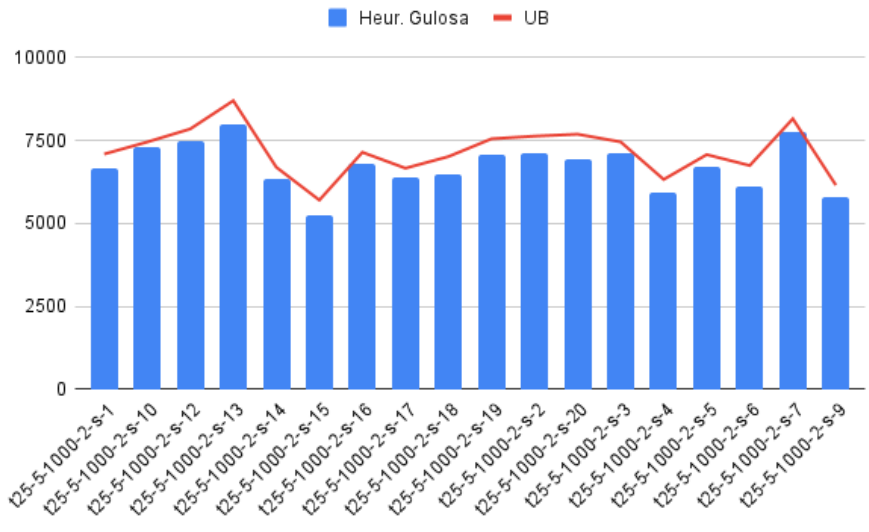

Figure 1: Comparação entre a solução PLI e Heurística Gulosa

O *gap* médio da heurística gulosa sem melhoria foi de 5,19%, enquanto com a melhoria, o *gap* médio fio de 1,73%.

Foram feitos testes com um total de 94 instâncias e, dentre elas, a heurística com melhoria foi melhor em 91 instâncias, enquanto a outra foi melhor apenas em 3, mas com uma diferença de menos de 1%.

Como um dos passos do *Destroy and Repair* é chamar o PLI para reparar a solução, o tempo médio da heurística melhorado foi de 1 segundo, enquanto a sem melhoria foi de 0,000035 segundos.

3.2 T2

A segunda verificação foi feita comparando os resultados gerados pelo heurística gulosa com e sem melhoria. O gráfico abaixo mostra essa comparação para o conjunto de instâncias **t25**:

Em geral, as melhorias proporcionadas pela heurística de melhoria *Destroy and Repair* foram boas, não somente para esse grupo de instância, mas para todas como um todo.

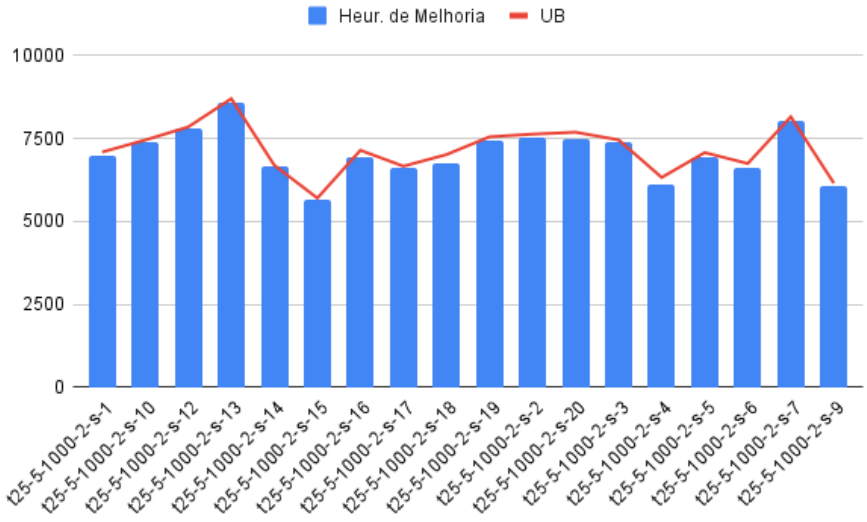

Figure 2: Comparação entre a solução PLI e Heuristica Gulosa Melhorada

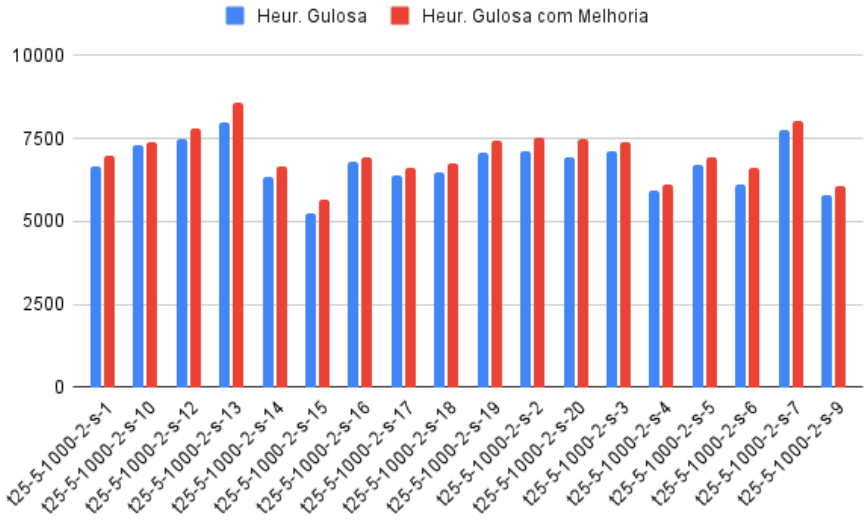

Figure 3: Comparação entre a Heuristica Gulosa com e sem Melhoria

4 Tabela Comparativa

Seguem o link para a tabela que compara o desempenho das heurísticas: `https://docs.google.com/spreadsheets/d/1lmo71rNT8P9VXZdwyBnz2ctcvwPoYskjZV57gABTXn8/edit?usp=sharing`. Obtamos por disponibilizar o link, pois a tabela em si ficou muito grande e, para adicioná-la, seria necessário ultrapassar o limite máximo de páginas.

5 Conclusões

O desenvolvimento desse trabalho auxiliou na absorção e no aprofundamento do conhecimento referente aos problemas NP-difícil e a todo o processo para encontrar uma possível solução razoável para este tipo de problema e aperfeiçoa-la.

Nele foi possível observar o quanto a heurística de melhoria aperfeiçoou a solução inicial encontrada pelas heurísticas ingênuas. Além disso, também pudemos observar o quanto o resultado da heurística melhorada chegou perto da solução ótima encontrada pelo PLI, principalmente quando se leva em condideração o tempo que o PLI levou para encontrar essa solução, que foi de 5 minutos, enquanto o tempo médio da gulosa melhorada foi de: 1,01 segundo. No fim, a implementação da melhoria obteve resultados satifatórios, sendo possível observa-los nas análises gráficas.