Logística Urbana para Entrega de Mercadorias

DA_T1 2021/22

- João Ricardo Alves - up202007614
- Marco André Rocha - up202004891
- Ricardo de Matos - up202007962

Descrição do problema

Este trabalho tem como objetivo especificar e implementar uma plataforma de **gestão de logística urbana**. Nesta logística destacam-se as carrinhas (estafetas) que entregam encomendas diárias e uma carrinha unitária que entrega encomendas expresso.

Os algoritmos desenvolvidos devem ser o mais eficientes possíveis temporal e espacialmente, adotando, para isso, vários dos métodos abordados em aula.

O trabalho divide-se em **3 cenários** a serem explorados:

- **Cenário 1** - Minimização do número de estafetas
- **Cenário 2** - Maximização do lucro da empresa
- **Cenário 3** - Minimização do tempo médio de entregas expresso

Formalização - Cen. 1

Seja T, um conjunto de carrinhas.
Seja O, um conjunto de encomendas.

Maximizar: número de encomendas $\sum_{i=1}^{n} o_i$

Minimizar: número de carrinhas $\sum_{i=1}^{n} t_i$

Sujeito a:

$t.\text{pesoMax} >= \sum_{i=1}^{n} o_{i.\text{peso}} \quad \Lambda \quad t.\text{volMax} >= \sum_{i=1}^{n} o_{i.\text{vol}}$ para qualquer $t \in T$

Os 4 Algoritmos criados:

- **Rapid Knapsack:** Escolhe a carrinha com a maior quantidade de encomendas usando *knapsack* e continua a escolher a melhor carrinha com as restantes encomendas (*greedy*).

- **Rapid Greedy:** Escolhe a carrinha com a maior quantidade de encomendas usando um algoritmo *greedy* e continua a escolher a melhor carrinha com as restantes encomendas (*greedy*).

- **Slow Brute Force:** Escolhe a carrinha com a maior quantidade de encomendas calculando todas as combinações usando *brute force,* e continua a escolher a melhor carrinha com as restantes encomendas (*greedy*).

- **Slow Backtracking:** Calcula todas as possibilidades que existem e obtém a solução ideal usando backtracking. Algoritmo muito pesado.

Complexidade - Cen. 1

Sejam:
T -> número de carrinhas E -> número de encomendas

Complexidade Temporal:

- **Rapid Knapsack:** $O(N) = O(T * E * (volMax * pesoMax + T))$
- **Rapid Greedy:** $O(N) = O(T*(E*\log(E) + T*E) = O(T*E*(\log(E)+T))$
- **Slow Brute Force:** $O(N) = O(T*(2^E + T * 2^E) = O(T^2 * 2^E)$
- **Slow Backtracking:** $O(N) = O(T^E)$

Complexidade Espacial:

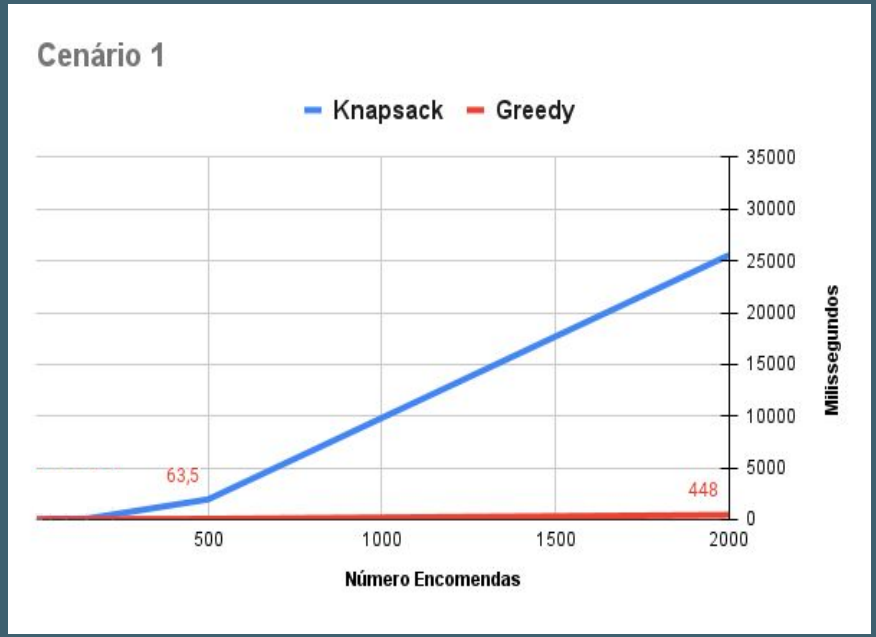
- **Rapid Knapsack:** $S(N) = S(E * volMax * pesoMax)$
- **Rapid Greedy:** $S(N) = S(1)$
- **Slow Brute Force:** $S(N) = S(2^E)$
- **Slow Backtracking:** $S(N) = S(T^E)$

Resultados - Cen. 1

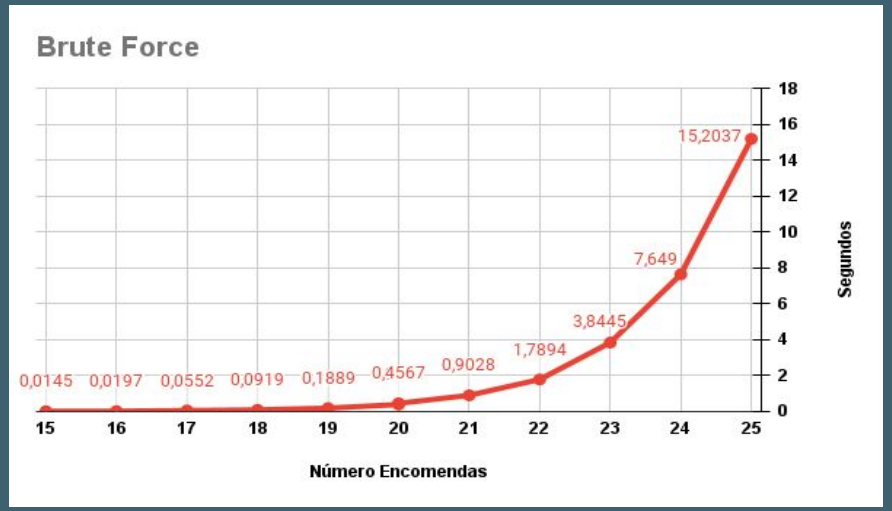
Como expectável, a abordagem com o algoritmo greedy é muito mais rápida para grandes números de encomendas.

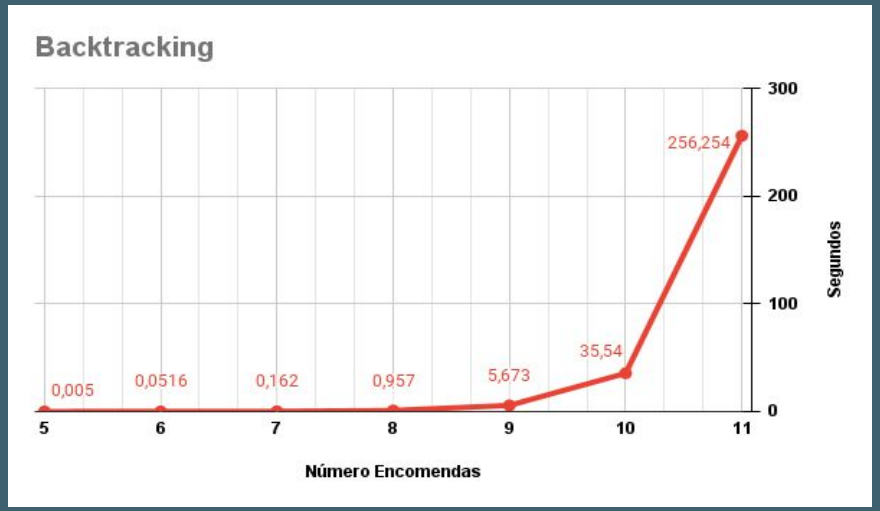
No entanto, apesar do algoritmo knapsack ser consideravelmente mais lento, o seu crescimento é linear, provando ser, ainda assim, uma alternativa viável.

Como o algoritmo knapsack apresenta geralmente melhores resultados, o utilizador pode optar por equilíbrio entre tempo de execução e optimização do resultado pretendido.



Resultados - Cen. 1





Ambos estes algoritmos são muito mais lentos comparativamente aos anteriores. O algoritmo de Brute Force, apesar de apresentar um grande tempo de execução, apresenta a mesma qualidade de resultados que o knapsack, não tendo por isso aplicação prática (utilizado apenas a título comparativo).

No que concerne o algoritmo de backtracking, este apresenta uma solução ideal, no entanto, é também a solução mais lenta de todas. O seu crescimento é exponencial com o número de encomendas, inviabilizando o seu uso.

Formalização - Cen. 2

Seja T, um conjunto de carrinhas.
Seja O, um conjunto de encomendas.

Maximizar: número lucro $\sum_{i=1}^{n} o_i.reward - \sum_{j=1}^{z} t_j.cost$, sendo n o nr de encomendas usadas e z carrinhas.

Sujeito a:

$t.pesoMax >= \sum_{i=1}^{n} o_{i}.peso \quad \Lambda \quad t.volMax >= \sum_{i=1}^{n} o_{i}.vol$ para qualquer $t \in T$, com Oi atribuído à carrinha

Os 3 Algoritmos criados:

- **Rapid pseudo-Fractorial:** Escolhe a carrinha com o maior profit usando um algoritmo semelhante ao **Fractorial Knapsack** mas em que as encomendas não são divididos.

- **Spacious linear knapsack :** Escolhe a carrinha com o maior profit usando *knapsack 1-0 com duas restrições* e continua a escolher a melhor carrinha com as restantes encomendas (*greedy*). Sem otimização de Espaço.

- **Optimize linear knapsack:** Escolhe a carrinha com o maior profit usando o algoritmo de hirschberg *(com divide and conquer, de forma a conseguirmos "recuperar" as encomendas colocadas nas carrinhas, ver) com duas restrições* e continua a escolher a melhor carrinha com as restantes encomendas (*greedy*). Com otimização de Espaço e utilização de uma abordagem diferente da anterior mas com mesmos resultados.

Complexidade - Cen. 2

Sejam:
T -> número de carrinhas
E -> número de encomendas

- **pseudo-Fractorial:** $O(N) = O(e*\log(e) * T*Z) \longrightarrow Z$ = n.carrinhas com lucro

  $S(N) = O(N) \longrightarrow$ encomendas usadas em cada iteração

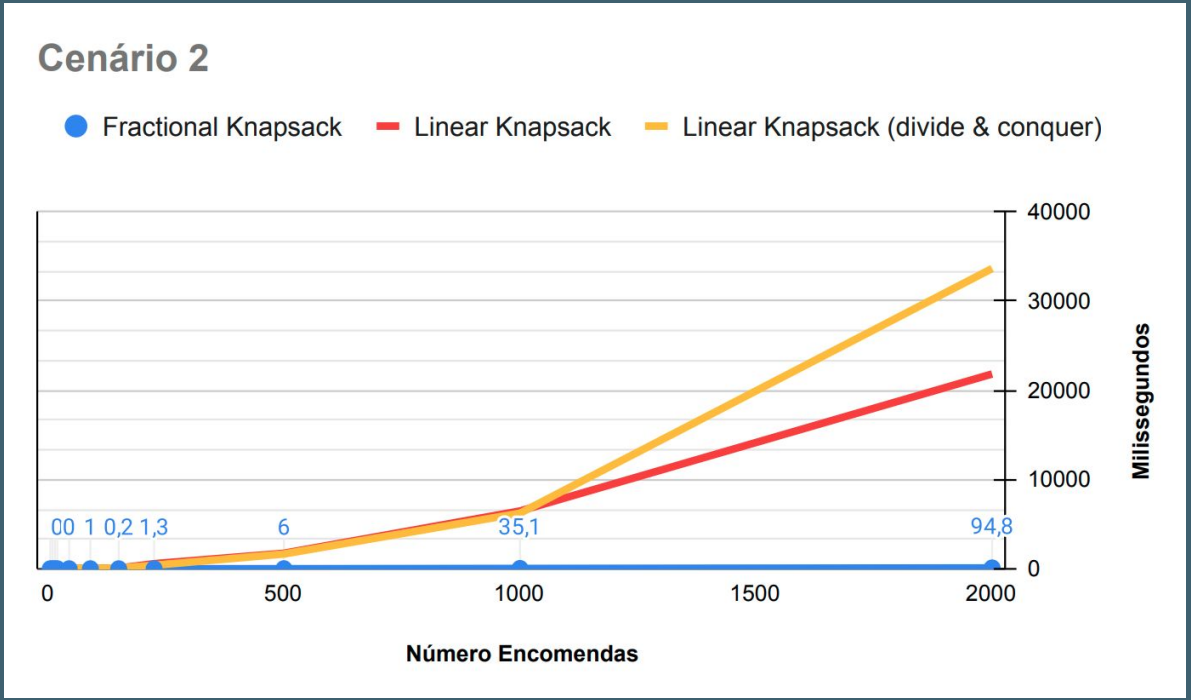
- **Spacious linear knapsack :** $O(N) = O(E * volMax * pesoMax * T*Z)$

  $S(N) = O(E * volMax * pesoMax)$

- **Optimize linear knapsack :** $O(N) = O(E * volMax * pesoMax * T*Z)$

  $S(N) = O(volMax * pesoMax)$

Resultados - Cen. 2



Nota: O número de carrinhas também varia. Para mais detalhes, conferir:

https://docs.google.com/spreadsheets/d/1grAwCQUWOP5RiLvV50HRBSo_Yy3XrwvnA9YKelCbnxw/edit?usp=sharing

O Fractional Knapsack revela-se ordens de grandeza mais rápido a computar os resultados enquanto que os outros dois algoritmos são semelhantes para valores abaixo das 1000 encomendas. Ambos os knapsack atingem os resultados ideais.

Formalização - Cen. 3

Input: Conjunto V de *N* encomendas express V1,...,Vn.

Di ⟶ duração de entrega de uma encomenda i (1 <= i <= n).

Objetivo: minimizar tempo médio de entregas expresso:

$$\sum_{i=1}^{n} \frac{(n - i + 1) * Di}{n}$$

Por intuição, queremos entregar as encomendas com menor duração (abordagem Greedy). Assim, o tempo pode ser minimizado se as encomendas forem ordenadas: D1 <= D2 <= ... <= Dn

Solução: Ordenar crescentemente por duração o vetor de entrada e iterar os elementos até o tempo limite ser atingido (8 horas)

Algoritmos - Cen. 3

Neste cenário foram implementados 2 algoritmos:

1. Fazer sort do vetor de encomendas de entrada (abordagem greedy)

2. Brute force de todas as permutações do vetor de entrada

Para sort do vetor foram utilizados dois algoritmos: STL Sort e Bubblesort.

No Brute force, testa-se todas as permutações do vetor de entrada de forma a descobrir qual a melhor ordem (embora já se saiba por prova de otimalidade que ordenação crescente seja a melhor abordagem. Serve apenas a título de exemplo e comparação).

Perante um vetor ordenado, itera-se os elementos, até a soma das suas durações atingir as 8 horas. Todas as encomendas até esse último índice serão entregues.

Complexidade - Cen. 3

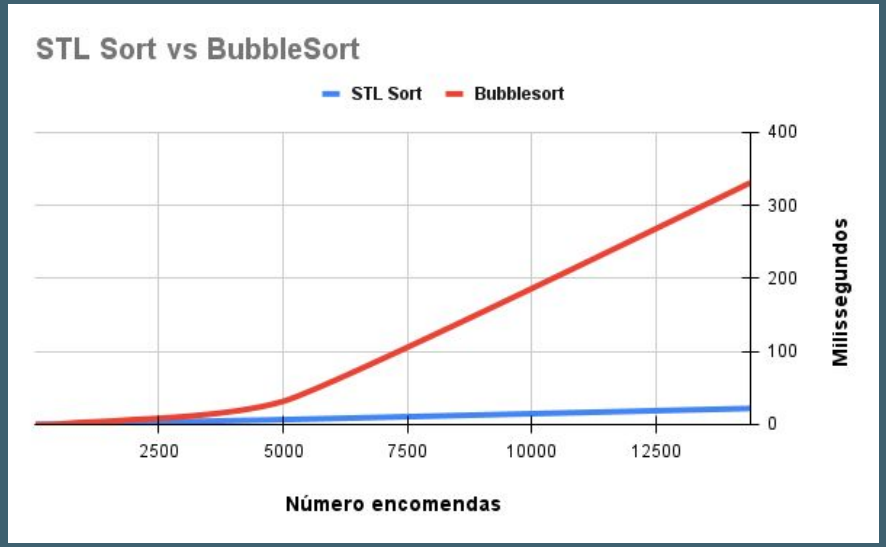
Tendo em conta os 3 algoritmos implementados, abaixo seguem-se as suas complexidades:

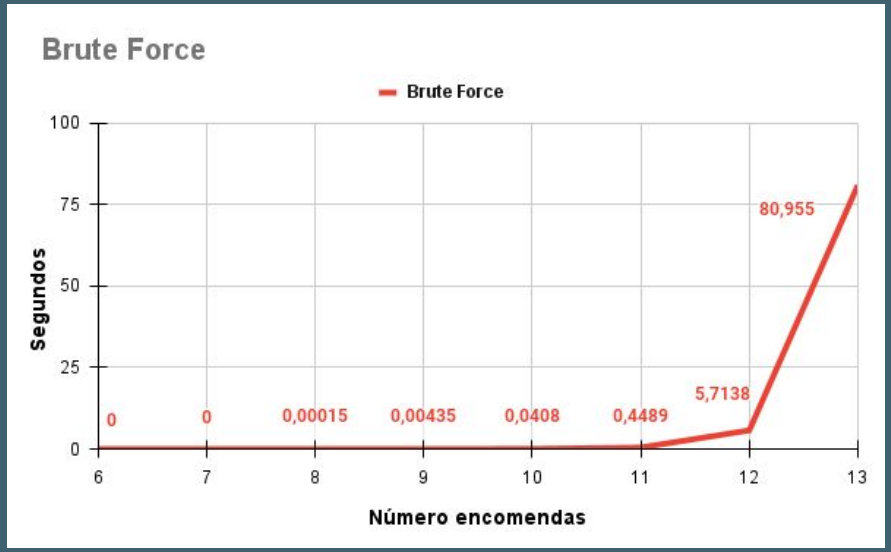
E ⟶ número de encomendas

- **STL Sort:** **O(N) = O(N * Log N)**
- **BubbleSort:** **O(N) = O(N ^ 2)**
- **Brute Force:** **O(N) = O(E !)**

Espacialmente, o primeiro é S(N) = log N e os restantes são S(N) = 1.

Resultados - Cen. 3





Como seria previsível, a diferente complexidade dos sort utilizados verifica-se empíricamente perante o aumento de dados (N*LOG N vs N^2).

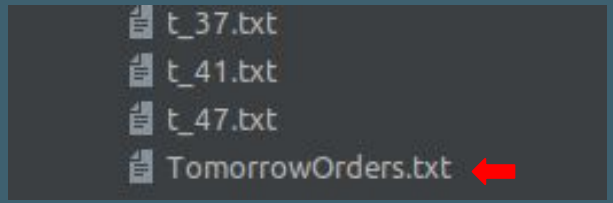
Um outro dado interessante referir é a inviabilidade de utilizar brute force e testar todas as permutações do vetor de encomendas de entrada visto serem necessários 80s para calcular apenas 13 encomendas.

Funcionalidades Extra

Foram implementadas 3 funcionalidades extra:

1. **Percentagem de entregas feitas num dia**
2. **Guardar encomendas não entregues e usar esses dados no dia seguinte**
3. **Gerar ficheiros report de cada carrinha para o dia**

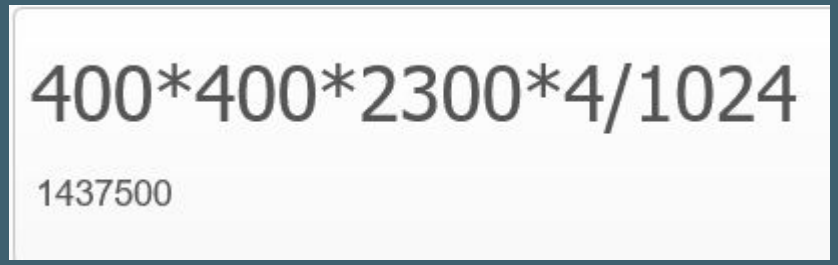
```
Profit: 11257€
Deliveries: 10 / 10 (100%)
Time taken: 0.081s
```



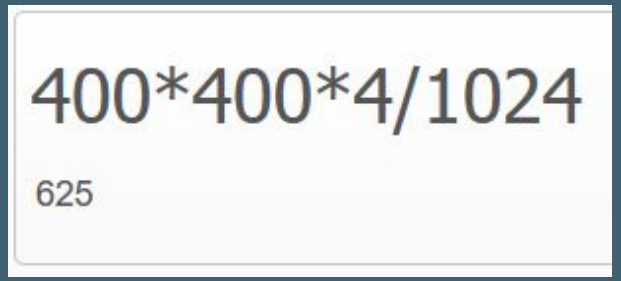

```
Truck_id: 23
Total Reward: 890€
==================================
  OrderId            Reward
==================================
   310               1083€
   307                834€
   346                666€
   336                522€
   342                427€
   377               1232€
    44               1160€
   413                851€
    31               1051€
   417               1166€
   202                671€
    45                873€
   314                823€
    83                816€
   137                654€
   265               1170€
    86               1233€
==================================
N_Orders: 17    T_Reward: 15232€
```

Algoritmo Destaque

O destaque do grupo vai para o algoritmo de knapsack linear optimizado.
A solução utilizada poupa espaço, impedindo o programa de falhar em oposição ao que acontece com o knapsack não otimizado para ficheiros com números de encomendas e carrinhas nas ordens das dezenas de milhares por atingir o limite do espaço de endereçamento. Abaixo encontram-se os espaços ocupados por ambos os algoritmos em KB:



1. Espaço utilizado ao calcular o reward máximo com limites de 400 de peso e volume e 2300 encomendas, com 4 bytes em kB. (esta informação é reutilizada para cada Truck, sendo calculada só uma vez por iteração)



2. Algoritmo Optimizado com recurso a divide and conquer

Dificuldades do Grupo

A principal dificuldade do grupo foi encontrar algoritmos eficientes que fossem de encontro aos nossos objetivos. Tal se mostrou mais complicado no cenário 2 onde, após várias abordagens, se alcançou resultados satisfatórios.

Várias abordagens Greedy e Knapsack também foram usadas, sendo algumas descartadas por não serem viáveis do ponto de vista da eficiência e ou da maximização da função objetivo.

Apesar disso, procuramos abordar o projeto de várias formas, usando diferentes algoritmos que julgamos enriquecerem o trabalho.

Relativamente à participação, todos os membros desempenharam um papel ativo no desenvolvimento do trabalho.

Exemplos de Execução

```
    Order: 265      weight: 26      volume: 26
    Order: 192      weight: 28      volume: 25
Truck 7: 11 deliveries
    Order: 382      weight: 29      volume: 22
    Order: 377      weight: 29      volume: 23
    Order: 353      weight: 29      volume: 23
    Order: 301      weight: 29      volume: 22
    Order: 269      weight: 29      volume: 26
    Order: 196      weight: 29      volume: 29
    Order: 158      weight: 26      volume: 29
    Order: 116      weight: 26      volume: 29
    Order: 86       weight: 29      volume: 26
    Order: 79       weight: 29      volume: 29
    Order: 64       weight: 29      volume: 29
Truck 3: 4 deliveries
    Order: 433      weight: 29      volume: 19
    Order: 418      weight: 29      volume: 26
    Order: 408      weight: 28      volume: 22
    Order: 329      weight: 26      volume: 29

Total deliveries: 450
Number of Trucks: 21
Time taken: 1.416s
Deliveries: 450 / 450 (100%)
```

Cenário 1 (Rapid Knapsack)

```
Truck 7: 11 deliveries
    Order: 337      weight: 26      volume: 26
    Order: 265      weight: 26      volume: 26
    Order: 76       weight: 26      volume: 26
    Order: 55       weight: 26      volume: 26
    Order: 3        weight: 28      volume: 25
    Order: 27       weight: 28      volume: 25
    Order: 84       weight: 28      volume: 25
    Order: 81       weight: 28      volume: 25
    Order: 75       weight: 28      volume: 25
    Order: 192      weight: 28      volume: 25
    Order: 329      weight: 26      volume: 29
Truck 3: 8 deliveries
    Order: 116      weight: 26      volume: 29
    Order: 269      weight: 29      volume: 26
    Order: 418      weight: 29      volume: 26
    Order: 158      weight: 26      volume: 29
    Order: 86       weight: 29      volume: 26
    Order: 196      weight: 29      volume: 29
    Order: 79       weight: 29      volume: 29
    Order: 64       weight: 29      volume: 29

Total deliveries: 443
Number of Trucks: 21
Time taken: 0.011s
Deliveries: 443 / 450 (98.4444%)
```

Cenário 1 (Rapid Greedy)

Exemplos de Execução

```
Truck_id: 12
Items left:    213  Max profit:  8126
Truck_id: 19
Items left:    196  Max profit:  7790
Truck_id: 7
Items left:    178  Max profit:  4551
Truck_id: 35
Items left:    158  Max profit:  4684
Truck_id: 27
Items left:    139  Max profit:  2775
Truck_id: 23
Items left:    122  Max profit:  890

Total Profit = 222025€
Time Taken: 0.009s
Deliveries: 122 / 450 (27.1111%)
```

Cenário 2 (Fractional Knapsack)

```
Truck_id: 22
Items left:    206  Max profit: 8626
Truck_id: 34
Items left:    188  Max profit: 7077
Truck_id: 27
Items left:    170  Max profit: 5160
Truck_id: 38
Items left:    150  Max profit: 3910
Truck_id: 15
Items left:    132  Max profit: 2500
Truck_id: 2
Items left:    115  Max profit: 1188

Total Profit = 231694€
Time Taken: 1.349s
Deliveries: 115 / 450 (25.5556%)
```

Cenário 2 (Fractional Knapsack)

Exemplos de Execução

```
Orders delivered:
201 47  220 185 328 96  358 257 49
308 335 234 31  294 160 63  434 247
101 329 62  19  41  403 395 157 295
324 282 210 192 134 359 267 198 341
79  347 173 332 214 216 446 40  230

Delivery time: 28652s
Avg Time: 231s
Profit: 147842€
Deliveries: 124 / 450 (27%)
Time taken: 0.001s
```

Cenário 3 (STL Sort)

```
Starting brute force...

Orders delivered:
628     4769    5302    5709    5826
6658    7221    8167    10263   10385

Delivery time: 4860s
Avg Time: 486s
Profit: 11257€
Deliveries: 10 / 10 (100%)
Time taken: 0.081s
```

Cenário 3 (Brute Force)

Fim

- João Ricardo Alves - up202007614
- Marco André Rocha - up202004891
- Ricardo de Matos - up202007962