LOGÍSTICA URBANA PARA ENTREGA DE MERCADORIAS

Desenho de Algoritmos | Grupo 122 | Entrega 1 | 22/04/2022

André Gabriel Correia Vieira - up202004159@edu.fe.up.pt
Francisca Horta Guimarães - up202004229@edu.fe.up.pt
Pedro Joaquim Alves Oliveira - up202004324@edu.fc.up.pt

PROBLEMA

Este projeto tem como objetivo especificar e implementar a plataforma de gestão de uma empresa de logística urbana, com a finalidade de tornar a sua operação o mais eficiente possível.

Para isso, as soluções algorítmicas a desenvolver devem ser o mais eficiente possível, utilizando para esse propósito, os algoritmos abordados em contexto de sala de aula.

Para este projeto, pretende-se particularmente explorar os seguintes cenários:

Cenário 1 – minimizar o número de estafetas.
Cenário 2 - maximizar o lucro da empresa.
Cenário 3 - minimizar o tempo médio previsto das entregas expresso.

CENÁRIO 1 - FORMALIZAÇÃO

O objetivo principal para este cenário é minimizar o número de estafetas para a entrega de todos os pedidos ou do maior número de pedidos, num dia.

Dados:

$e1, \ldots, en$ – Conjunto de estafetas, E, com capacidade de volume ve , peso we , e custo ce (valores dos estafetas $1, \ldots, n$).

$p1, \ldots, pn$ – Conjunto de pedidos, P, com volume vp , peso wp , e recompensa rp (valores dos pedidos $1, \ldots, n$).

Variáveis de Decisão:

ne - Número de Estafetas. np - Número de Pedidos.

Minimizar a função objetivo:

$\frac{\max(\sum_{i=1}^{n} wp_i + \sum_{i=1}^{n} vp_i)}{n}$ – sendo n o número de estafetas.

Sujeito à restrição:

$\frac{\max(\sum_{i=1}^{n} wp_i + \sum_{i=1}^{n} vp_i)}{n} <$ capacidade máxima do estafeta.

CENÁRIO 1 – ALGORITMOS RELEVANTES

- Para a realização deste cenário, considerámos um algoritmo ganancioso (greedy).

- Num ponto inicial de planeamento, consideramos algoritmos como:

    - Brute Force
    - Backtracking

No entanto, imediatamente nos apercebemos da enorme complexidade temporal, tendo portanto, optado pelo algoritmo definido anteriormente.

- Na implementação do algoritmo greedy abordamos a questão como uma espécie de mistura dos clássicos problemas:

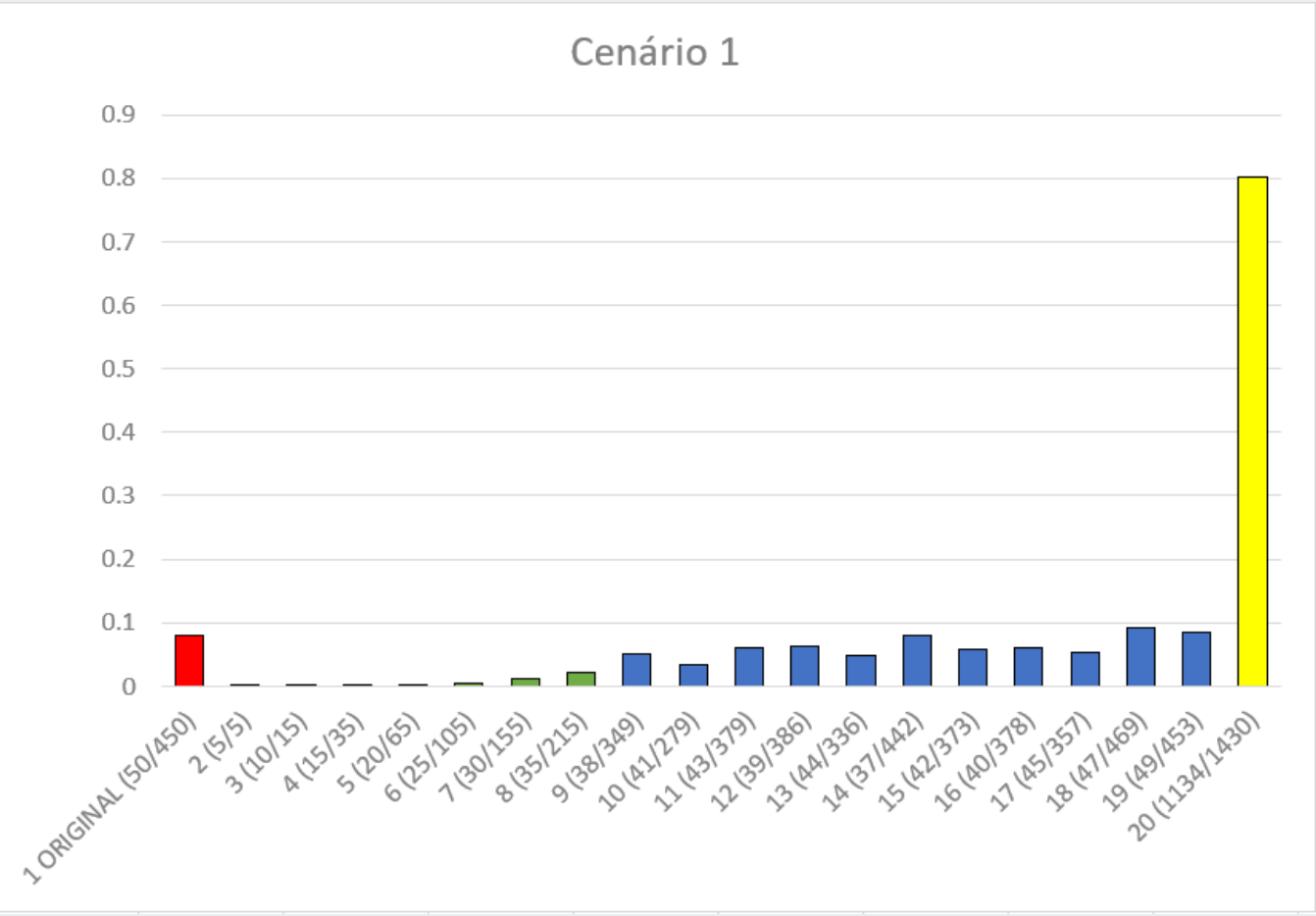
    - Knapsack
    - Bin Packing

CENÁRIO 1 – ANÁLISE DE COMPLEXIDADE

- Complexidade Temporal:
  - $T(N)= N + N*(N + N*N + N) = N + N*N*N = N*N*N = O(N^3)$;
- Complexidade Espacial:
  - $S(N) = O(N)$;

CENÁRIO 1 – AVALIAÇÃO EMPÍRICA

Dias (carrinhas/encomendas)	segundos	nanossegundos
1 ORIGINAL (50/450)	0.0796341	79634100
2 (5/5)	2.06E-05	20600
3 (10/15)	0.00012	120000
4 (15/35)	0.0006351	635100
5 (20/65)	0.0021333	2133300
6 (25/105)	0.0055691	5569100
7 (30/155)	0.0113741	11374100
8 (35/215)	0.0204321	20432100
9 (38/349)	0.05187	51870000
10 (41/279)	0.0333798	33379800
11 (43/379)	0.0598267	59826700
12 (39/386)	0.0627288	62728800
13 (44/336)	0.0475491	47549100
14 (37/442)	0.0803149	80314900
15 (42/373)	0.0588285	58828500
16 (40/378)	0.0602894	60289400
17 (45/357)	0.0542931	54293100
18 (47/469)	0.0922618	92261800
19 (49/453)	0.0842588	84258800
20 (1134/1430)	0.8023508	802350800



CENÁRIO 2 - FORMALIZAÇÃO

O objetivo principal para este cenário é maximizar o lucro da empresa.

Dados:

$e1, \dots, en$ – Conjunto de estafetas, E, com capacidade de volume ve , peso we , e custo ce (valores dos estafetas $1, \dots, n$).

$p1, \dots, pn$ – Conjunto de pedidos, P, com volume vp , peso wp , e recompensa rp (valores dos pedidos $1, \dots, n$).

Variáveis de Decisão:

pr – O valor da recompensa dos pedidos em função do custo de entrega.

Maximizar a função objetivo:

$\sum_{i=1}^{n} rp_i - \sum_{i=1}^{n} ce_i$ – $Diferença\ entre\ a\ recompensa\ do\ pedido\ e\ o\ custo\ da\ entrega$

Sujeito à restrição:

$\sum_{i=1}^{n} rp_i - \sum_{i=1}^{n} ce_i$ < capacidade máxima do estafeta && lucro positivo em cada estafeta

CENÁRIO 2 – ALGORITMOS RELEVANTES

Para este cenário, decidimos utilizar o mesmo algoritmo utilizado no cenário 1, o algoritmo ganancioso (greedy), diferenciando-se apenas na forma como lidamos com o prejuízo das entregas.

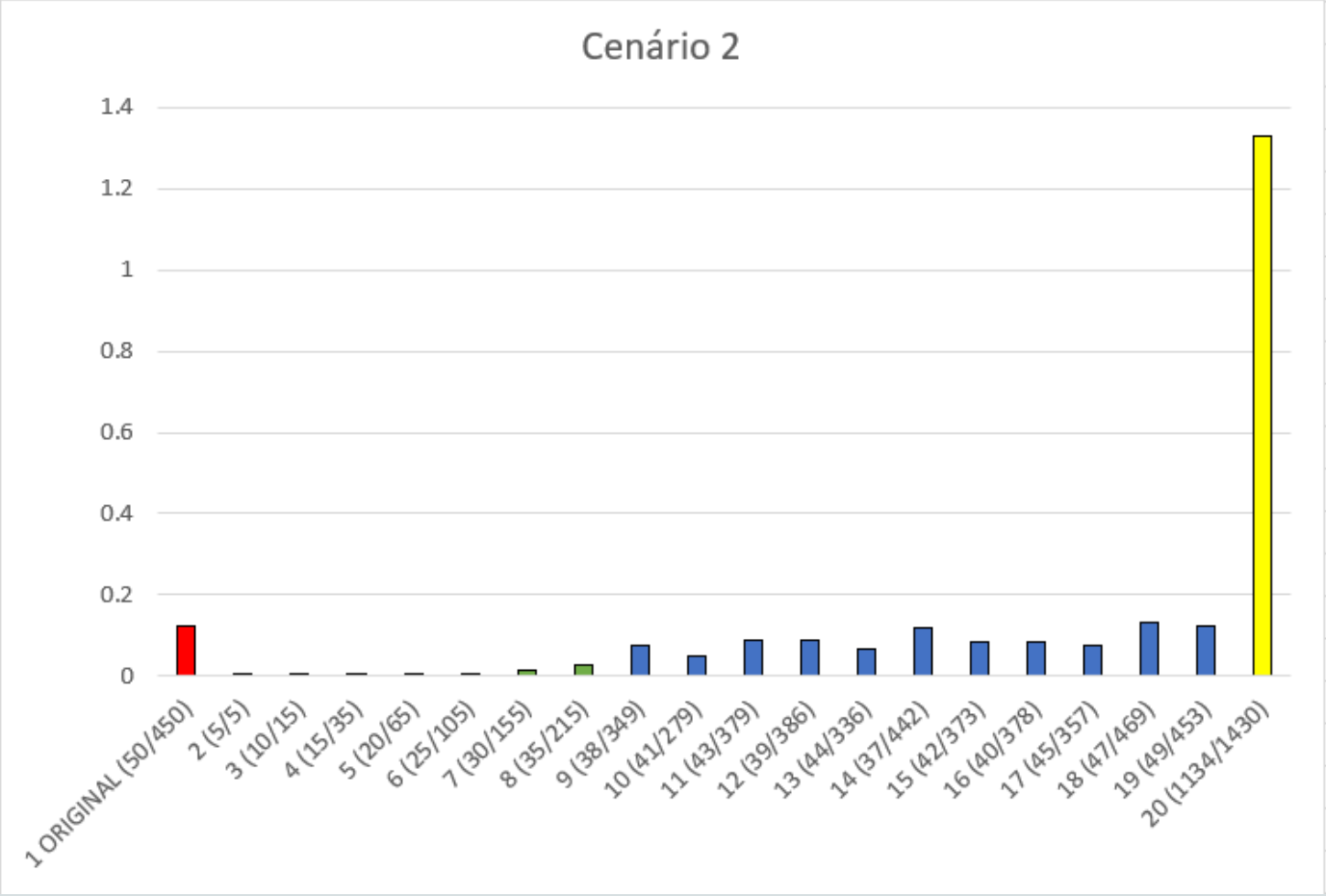
Assim, neste algoritmo descartamos as entregas que davam prejuízo obtendo desta forma o maior lucro possível.

CENÁRIO 2 – ANÁLISE DE COMPLEXIDADE

- Complexidade Temporal:
  - $T(N)= N + N*(N + N*N + N) = N + N*N*N = N*N*N = O(N^3)$;
- Complexidade Espacial:
  - $S(N) = O(N)$;

CENÁRIO 2 – AVALIAÇÃO EMPÍRICA

Dias (carrinhas/encomendas)	segundos	nanossegundos
1 ORIGINAL (50/450)	0.1248874	124887400
2 (5/5)	0.0000289	28900
3 (10/15)	0.0001536	153600
4 (15/35)	0.0007603	760300
5 (20/65)	0.0027528	2752800
6 (25/105)	0.0068107	6810700
7 (30/155)	0.014703	14703000
8 (35/215)	0.0279947	27994700
9 (38/349)	0.0738763	73876300
10 (41/279)	0.0490117	49011700
11 (43/379)	0.0873492	87349200
12 (39/386)	0.0895912	89591200
13 (44/336)	0.0681592	68159200
14 (37/442)	0.118183	118183000
15 (42/373)	0.0840012	84001200
16 (40/378)	0.0852412	85241200
17 (45/357)	0.0773944	77394400
18 (47/469)	0.1320632	132063200
19 (49/453)	0.1241524	124152400
20 (1134/1430)	1.3303273	1330327300



CENÁRIO 3 – FORMALIZAÇÃO

O objetivo principal para este cenário é minimizar o tempo médio previsto das entregas expresso a serem realizadas num dia:

Dados:
h - Horário de trabalho correspondente a 8 horas (h = $8 \times 3600s$).
$v1, \dots, vn$ - Conjunto de pedidos, P, com volume vp , peso wp , e tempo de entrega tp. (valores dos pedidos $1, \dots, n$).

Variáveis de Decisão:
$p1, \dots, pn$ - Conjunto de pedidos expresso, PE, com volume vp , peso wp e tempo de entrega tp. (valores dos pedidos $1, \dots, n$).

Minimizar a função objetivo:
$\sum_{i=1}^{n} tp_i \ - Soma\ do\ tempo\ de\ entrega\ dos\ pedidos.$

Sujeito à restrição:
$\sum_{i=1}^{n} tp_i \leq h \ - A\ soma\ do\ tempo\ de\ entrega\ dos\ pedidos\ está\ dentro\ do\ horário\ de\ trabalho$

CENÁRIO 3 – ALGORITMOS RELEVANTES

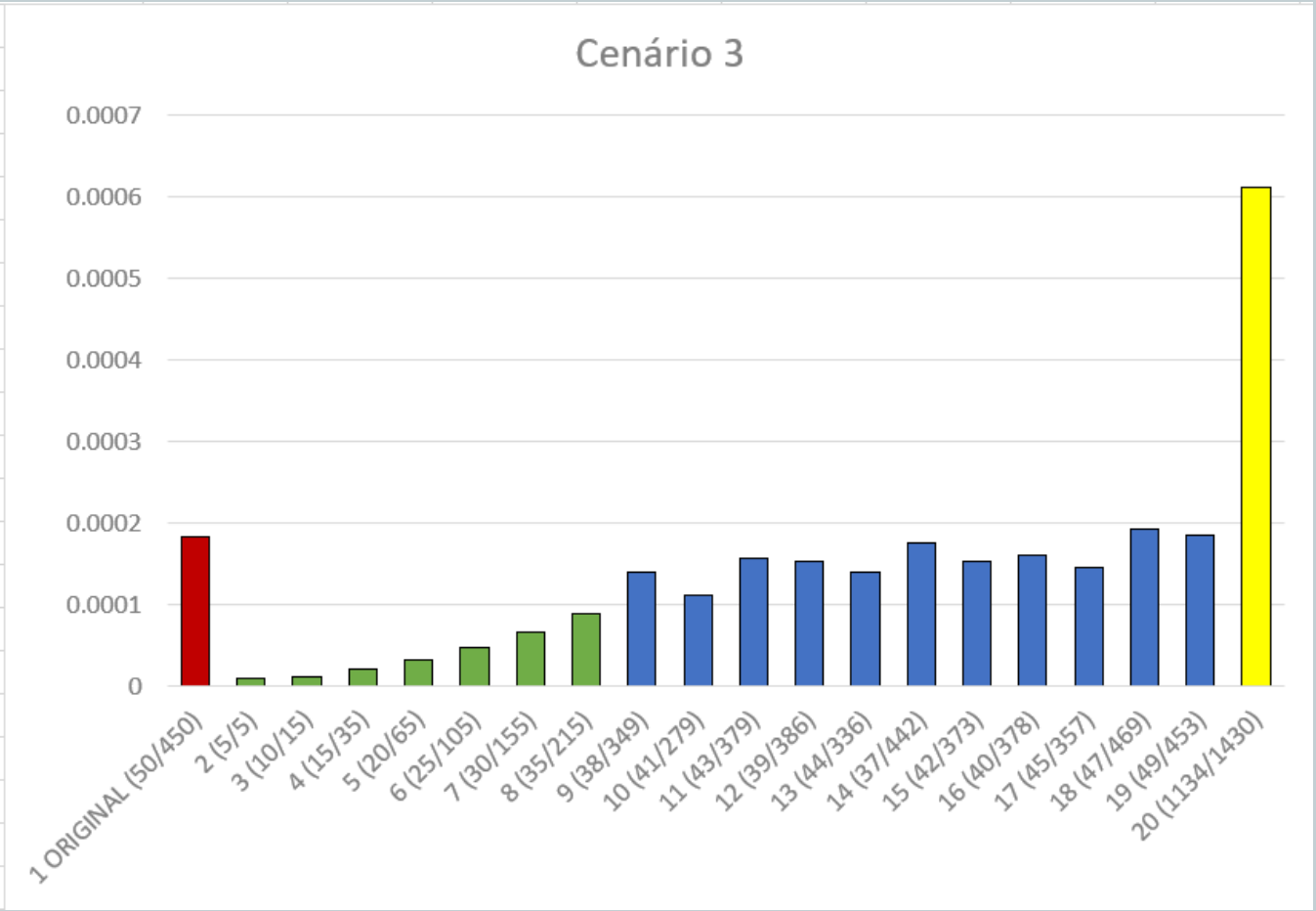
- Para a resolução deste cenário considerámos um algoritmo ganancioso (greedy).
- Para algoritmo de ordenação considerámos alguns algoritmos já referidos no curso, como:
  - Bubble Sort (O(N2))
  - Selection Sort (O(N2))
  - Insertion Sort (O(N2))
- No entanto, acabámos por implementar este algoritmo recorrendo ao std::sort() da biblioteca de C++.

CENÁRIO 3 – ANÁLISE DE COMPLEXIDADE

- Complexidade Temporal:
  - $T(N) = N*1 + N* \log(N) = O(N*\log(N))$
- Complexidade Espacial:
  - $S(N) = O(\log(N))$

CENÁRIO 3 – AVALIAÇÃO EMPÍRICA

Dias (carrinhas/encomendas)	segundos	nanossegundos
1 ORIGINAL (50/450)	0.0001821	182100
2 (5/5)	0.0000095	9500
3 (10/15)	0.0000102	10200
4 (15/35)	0.00002	20000
5 (20/65)	0.0000319	31900
6 (25/105)	0.000047	47000
7 (30/155)	0.0000665	66500
8 (35/215)	0.0000879	87900
9 (38/349)	0.0001396	139600
10 (41/279)	0.0001117	111700
11 (43/379)	0.000157	157000
12 (39/386)	0.0001534	153400
13 (44/336)	0.000139	139000
14 (37/442)	0.0001749	174900
15 (42/373)	0.0001521	152100
16 (40/378)	0.0001594	159400
17 (45/357)	0.0001454	145400
18 (47/469)	0.0001915	191500
19 (49/453)	0.0001844	184400
20 (1134/1430)	0.0006127	612700



SOLUÇÃO ALGORÍTMICA A DESTACAR

- A solução algorítmica a destacar é o algoritmo ganancioso (greedy) utilizado ao longo de quase todo o projeto.
- Destacamos, em particular, a implementação deste algoritmo através das funções BestFit() e bestPossibleOrder().

CONCLUSÃO

- **Principais Dificuldades Encontradas**

  - Pouca prática com linguagem de programação utilizada;
  - Otimização do cenário 2, apesar de utilizarmos o melhor algoritmo, na nossa opinião. Sabíamos o que queríamos fazer mas não como o fazer por falta de conhecimento e tempo.

- **Esforço de Cada Elemento do Grupo**

  - André Vieira: 33%
  - Francisca Guimarães: 33%
  - Pedro Oliveira: 33%