U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Logística Urbana para Entrega de Mercadorias

José Carvalho

Eduardo Correia

Carlos Madaleno

Descrição do problema

O objetivo principal será explorar diferentes formas de distribuir os vários pedidos pelos estafetas, assim como a sequência de entregas dos pedidos expresso.
Implementando assim uma plataforma de gestão da empresa tendo como objetivo principal a eficiência.

Objetivos principais por cenário

1. Otimização de estafetas
2. Otimização de lucro
3. Otimização de tempo

Cenário 1 - Otimização do número de estafetas

Dados de Entrada & Domínios

$estafetas$: "$conjunto\ de\ estafetas$"
$pedidos$: "$conjunto\ de\ pedidos$"
nEs = "$numero\ de\ estafetas$"
$nPed$ = "$numero\ de\ pedidos$"
$\forall\ Es\ \epsilon\ estafetas$:
$VEs_i \epsilon \mathbb{R}^+$: "$Volume\ maximo\ do\ estafeta\ i$"
$PEs_i \epsilon \mathbb{R}^+$: "$Peso\ maximo\ do\ estafeta\ i$"
$\forall\ Ped\ \epsilon\ pedidos$:
$VPed_i \epsilon \mathbb{R}^+$: "$Volume\ do\ pedido\ i$"
$PPed_i \epsilon \mathbb{R}^+$: "$Peso\ do\ pedido\ i$"
$DPed_i \epsilon \mathbb{N}$: "$Tempo\ do\ pedido\ i$"

Variáveis de decisão

EsI : "$conjunto\ de\ estafetas\ escolhidos$"
$PedI$: "$conjunto\ de\ pedidos\ escolhidos$"
$\forall\ PedI_i\ \epsilon\ PedI$:
$isPedI_i \epsilon [1, nEsI] \cap \mathbb{N}$: "$Pedido\ foi\ inserido\ no\ estafeta\ resultado$"
$PedI\_EsI_i$: "$Conjunto\ de\ pedidos\ escolhidos\ que\ pertecem\ ao\ estafeta\ i$"
$nPedI\_EsI_i$ = "$Numero\ de\ pedidos\ escolhidos\ que\ pertecem\ ao\ estafeta\ i$"
$nPedI$ = "$Numero\ de\ pedidos\ escolhidos$"
$nEsI$ = "$Numero\ de\ estafetas\ escolhidos$"

Cenário 1 - Otimização do número de estafetas

Função Objetivo

$$max(nPedI) \wedge min(nEsI)$$

Escolher o número máximo de pedidos, com o menor número de estafetas possíveis.

Restrições

$$\forall PedI_i \epsilon PedI, VPedI_i \leq VEsI_{isPedI_i}$$

$$\forall PedI_i \epsilon PedI, PPedI_i \leq PEsI_{isPedI_i}$$

$$\forall\ EsI_i\ \epsilon\ EsI:$$

$$\sum_{j=1}^{nPedI\_EsI_j} VPedI\_EsI_j < VEsI_i$$

$$\forall\ EsI_i\ \epsilon\ EsI:$$

$$\sum_{j=1}^{nPedI\_EsI_j} PPedI\_EsI_j < PEsI_i$$

$$\forall\ EsI_i\ \epsilon\ EsI:$$

$$\sum_{j=1}^{nPedI\_EsI_j} DPedI\_EsI_j < 24h$$

Descrição do Algoritmo

1. Escolha de um critério de ordenação
    a. Peso.
    b. Volume.
    c. Tempo.
    d. Comparação entre as 3 opções.
2. Ordenar estafetas pelo critério escolhido
3. Ordenar pedidos pelo critério respetivo
4. A variação best-fit do algoritmo bin packing organiza os pedidos pelos estafetas, com objectivo de minimizar o número dos mesmos
5. A opção de comparação calcula os três critérios de ordenação e retorna o melhor resultado

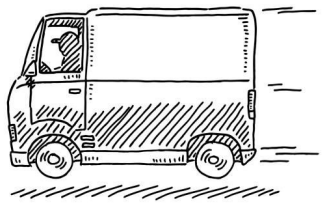
Análise da Complexidade

Complexidade Temporal -> O(n^2)

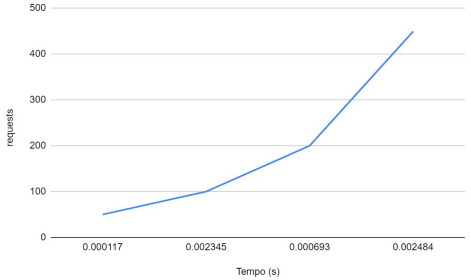
1. Copiar estafetas da classe dados para um vetor -> O(n)
2. Copiar pedidos da classe dados para um vetor -> O(n)
3. Ordenar estafetas -> O(nlogn)
4. Ordenar pedidos -> O(nlogn)
5. Inserir pedidos em estafetas -> O(n^2)
6. Escolher o melhor resultado -> O(1)

Complexidade Espacial -> O(n^2)

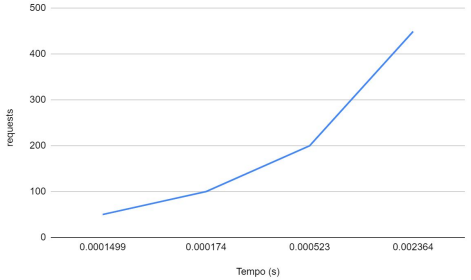
1. Guardar pedidos -> O(n)
2. Guardar estafetas -> O(n^2)
3. Ordenar estafetas -> O(n^2)
4. Ordenar pedidos -> O(n)
5. Inserir pedidos em estafetas -> O(n)



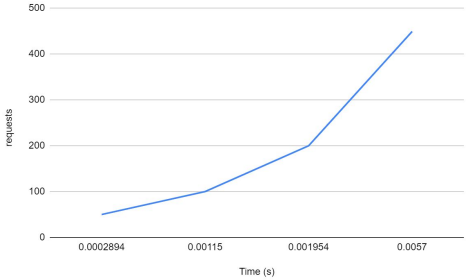
Avaliação Empírica



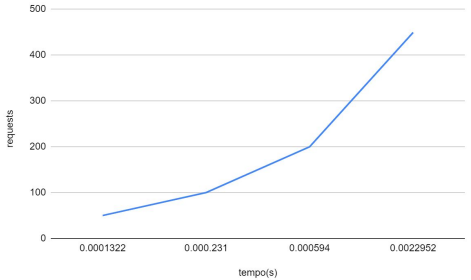
Peso



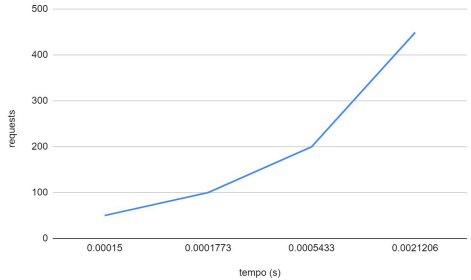
Sem Sorting



Melhor Caso



Volume



Duração

Cenário 2 – Otimização do lucro da empresa

Dados de Entrada & Domínios

$estafetas$: "$conjunto\ de\ estafetas$"
$pedidos$: "$conjunto\ de\ pedidos$"
nEs = "$numero\ de\ estafetas$"
$nPed$ = "$numero\ de\ pedidos$"
$\forall\ Es\ \epsilon\ estafetas$:
$VEs_i \epsilon \mathbb{R}^+$: "$Volume\ maximo\ do\ estafeta\ i$"
$PEs_i \epsilon \mathbb{R}^+$: "$Peso\ maximo\ do\ estafeta\ i$"
$CEs_i \epsilon \mathbb{R}^+$: "$Custo\ do\ estafeta\ i$"
$\forall\ Ped\ \epsilon\ pedidos$:
$VPed_i \epsilon \mathbb{R}^+$: "$Volume\ do\ pedido\ i$"
$PPed_i \epsilon \mathbb{R}^+$: "$Peso\ do\ pedido\ i$"
$RPed_i \epsilon \mathbb{R}_0^+$: "$Recompensa\ do\ pedido\ i$"

Variáveis de decisão

EsI : "$conjunto\ de\ estafetas\ escolhidos$"
$PedI$: "$conjunto\ de\ pedidos\ escolhidos$"
$\forall\ PedI_i\ \epsilon\ PedI$:
$isPedI_i \epsilon [1, nEsI] \cap \mathbb{N}$: "$Pedido\ foi\ inserido\ no\ estafeta\ resultado$"
$PedI\_EsI_i$: "$Conjunto\ de\ pedidos\ escolhidos\ que\ pertecem\ ao\ estafeta\ i$"
$nPedI\_EsI_i$ = "$Numero\ de\ pedidos\ escolhidos\ que\ pertecem\ ao\ estafeta\ i$"

Cenário 2 – Otimização do lucro da empresa

Função Objetivo

$$max(receitaTotal - custoTotal)$$

$$receitaTotal = \sum_{i=1}^{nPedI} RPedI_i$$

$$receitaTotal = \sum_{i=1}^{nEsI} CEsI_i$$

Restrições

$$\forall PedI_i \epsilon PedI, VPedI_i \leq VEsI_{isPedI_i}$$

$$\forall PedI_i \epsilon PedI, PPedI_i \leq PEsI_{isPedI_i}$$

$$\forall\ EsI_i\ \epsilon\ EsI:\ \sum_{j=1}^{nPedI\_EsI_j} VPedI\_EsI_j < VEsI_i$$

$$\forall\ EsI_i\ \epsilon\ EsI:\ \sum_{j=1}^{nPedI\_EsI_j} PPedI\_EsI_j < PEsI_i$$

$$\forall\ EsI_i\ \epsilon\ EsI:\ \sum_{j=1}^{nPedI\_EsI_j} RPedI\_EsI_j > CEsI_i$$

Descrição do Algoritmo

1. Escolher um critério de ordenação -> Peso ou Volume ou Peso*Volume / Custo ou Recompensa (dependendo se é um estafeta ou pedido).
2. Ordenar estafetas pelo critério escolhido.
3. Ordenar pedidos pelo critério respetivo e inserir numa fila.
4. Inserir pedido no topo da fila no primeiro estafeta disponível, enquanto houver pedidos.
5. Calcular lucro total, removendo camiões com lucro <= 0.
6. Repetir passos 1-5 com os restantes critérios.
7. Escolher resultado com valor máximo.

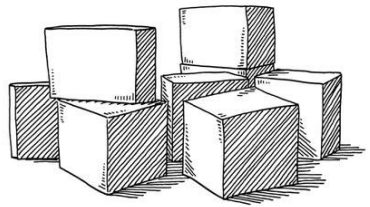
Análise da Complexidade

Complexidade Temporal -> O(n^2)

1. Copiar estafetas da classe dados para um vetor -> O(n)
2. Copiar pedidos da classe dados para um vetor -> O(n)
3. Ordenar estafetas -> O(nlogn)
4. Ordenar pedidos -> O(nlogn)
5. Inserir pedidos em estafetas -> O(n)
6. Calcular lucro e remover estafetas com lucro_local < 0 -> O(n^2)

Complexidade Espacial -> O(n^2)

1. Guardar pedidos -> O(n)
2. Guardar estafetas -> O(n^2)
3. Ordenar estafetas -> O(n^2)
4. Ordenar pedidos -> O(n)
5. Inserir pedidos em estafetas -> O(n)



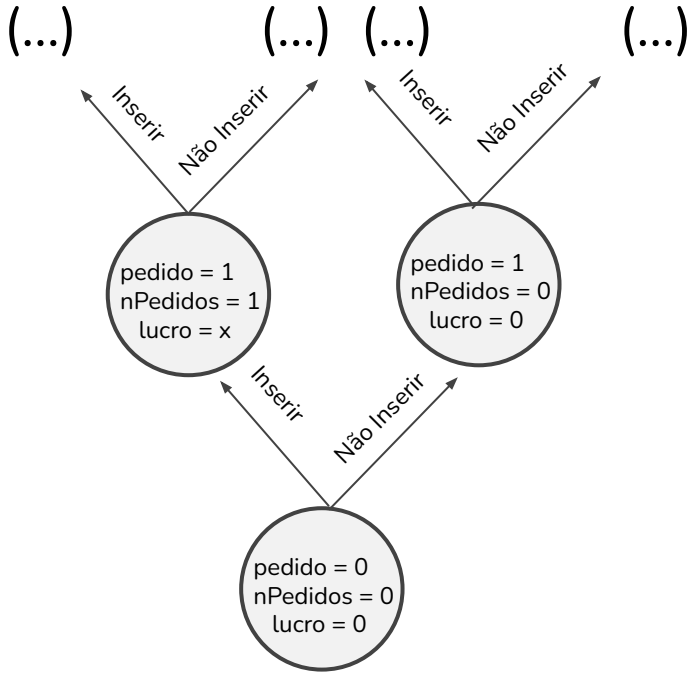
Algoritmo de Backtracking Alternativo (Adicional)

Descrição :

Algoritmo de backtracking, com limite de um camião, que itera todos os pedidos e segue os seguintes dois ramos :

1. Tentar inserir pedido atual.
2. Não inserir pedido atual.

O Algoritmo acaba quando chega ao final dos pedidos e devolve o lucro máximo.



Notas :

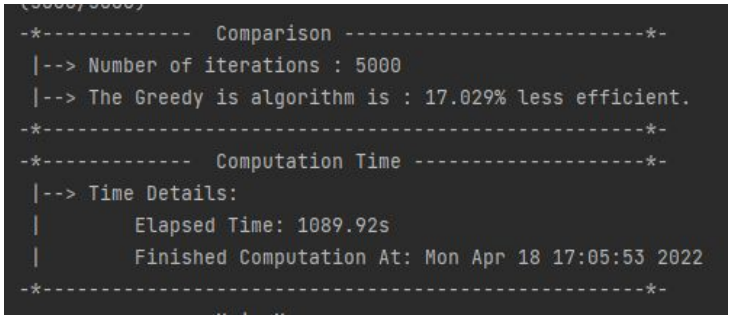
- Tem uma complexidade temporal e espacial de O(2^n).
- Garante a melhor solução.

Comparação estatística (Funcionalidade Extra) -> O(n^3)

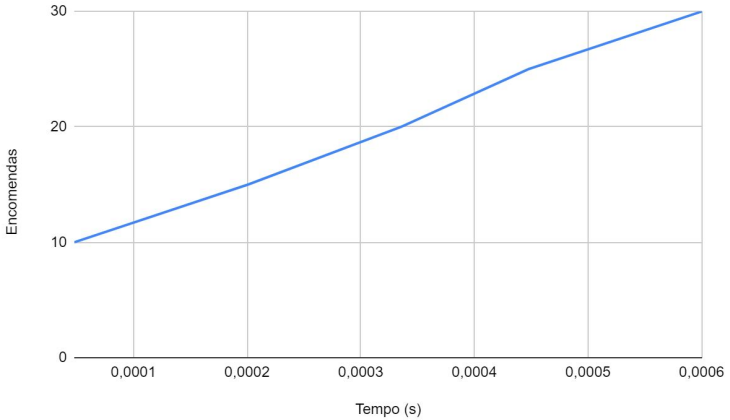
Com a criação de dois algoritmos, sendo que um garante a melhor distribuição, foi possível fazer uma funcionalidade que permite calcular a margem de erro para o algoritmo greedy.

O programa vai buscar aos datasets, 1 estafeta e [15,25] pedidos aleatórios. Depois o programa corre ambos os algoritmos e calcula a diferença (em percentagem). O programa faz este cálculo n vezes, com dados aleatórios diferentes (n é dado pelo utilizador), e depois faz a média do resultado obtido (casos onde não é possível ter lucro não são considerados)

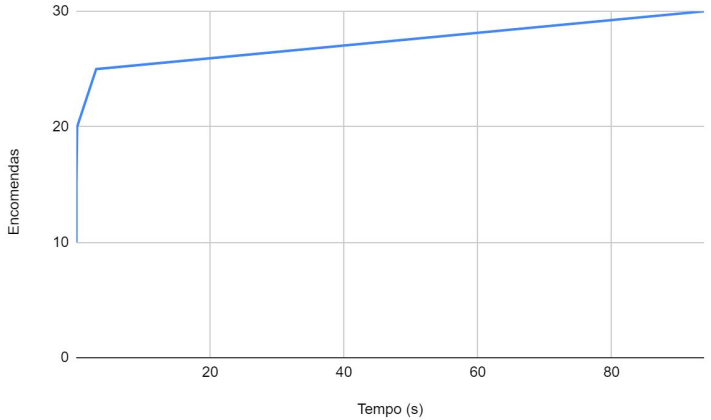
$$percentageDif = \frac{\sum_{i=1}^{nIterations} \frac{optimal - greedy}{optimal} * 100}{nIterations}$$



Avaliação Empírica



Greedy



Backtracking

Cenário 3 – Otimização das entregas expresso

Dados de Entrada & Domínios

$estafetas : "conjunto\ de\ estafetas"$
$pedidos : "conjunto\ de\ pedidos"$
$nEs = "numero\ de\ estafetas"$
$nPed = "numero\ de\ pedidos"$
$\forall\ Es\ \epsilon\ estafetas :$
$VEs_i \epsilon \mathbb{R}^+ : "Volume\ maximo\ do\ estafeta\ i"$
$PEs_i \epsilon \mathbb{R}^+ : "Peso\ maximo\ do\ estafeta\ i"$
$\forall\ Ped\ \epsilon\ pedidos :$
$VPed_i \epsilon \mathbb{R}^+ : "Volume\ do\ pedido\ i"$
$PPed_i \epsilon \mathbb{R}^+ : "Peso\ do\ pedido\ i"$
$DPed_i \epsilon \mathbb{N} : "Tempo\ do\ pedido\ i"$

Variáveis de decisão

$PedI : "conjunto\ de\ pedidos\ escolhidos"$
$nPedI = "Numero\ de\ pedidos\ escolhidos"$
$EsI : "Estafeta\ escolhido"$
$tempoUsado = \sum_{i=1}^{nPedI} DPedI_i$

Cenário 1 - Otimização do número de estafetas

Função Objetivo

$$min(tempoUsado/nPedI)$$

Minimizar a média do tempo necessário para fazer uma entrega.

Restrições

$$\forall PedI_i \epsilon PedI : VPedI_i \leq VEsI$$

$$\forall PedI_i \epsilon PedI : PPedI_i \leq PEsI$$

$$\sum_{i=1}^{nPedI} VPedI_i \leq VEsI$$

$$\sum_{i=1}^{nPedI} PPedI_i \leq PEsI$$

$$\sum_{i=1}^{nPedI} DPedI_i \leq 8h$$

Descrição do Algoritmo

1. Ordena as encomendas por tempo de entrega
2. Seleciona um estafeta
3. Considerando as características do estafeta verifica quais encomendas podem ser entregues enquanto existir tempo livre no dia.

Análise da Complexidade

Temporal:

Devido a necessidade de ordenar as encomendas. Para isso foi usado o std::sort de c++, cuja complexidade temporal e O(n log (n)).

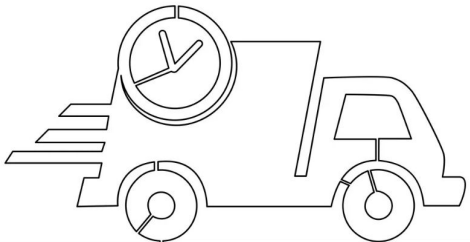
Para fazer a seleção e a gestão das encomendas na *timeframe* e usado um ciclo for tendo como complexidade temporal O(n)

Espacial:

Para guardar estafetas e pedidos: O(n).

Ordenar Pedidos: O(n^2).

Colocar pedidos em estafetas: O(n)



Algoritmo de Backtracking Alternativo

Descrição:

Este algoritmo tem como objetivo garantir **sempre** a melhor solução descartando o tempo necessário para concluir essa tarefa.

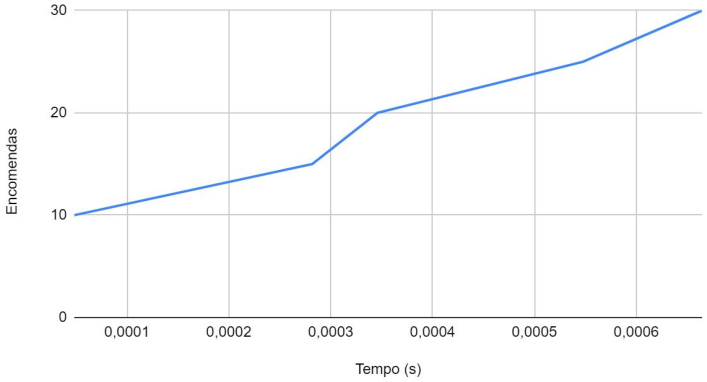
Análise da complexidade:

Tanto a complexidade espacial e temporal são O(2^n).

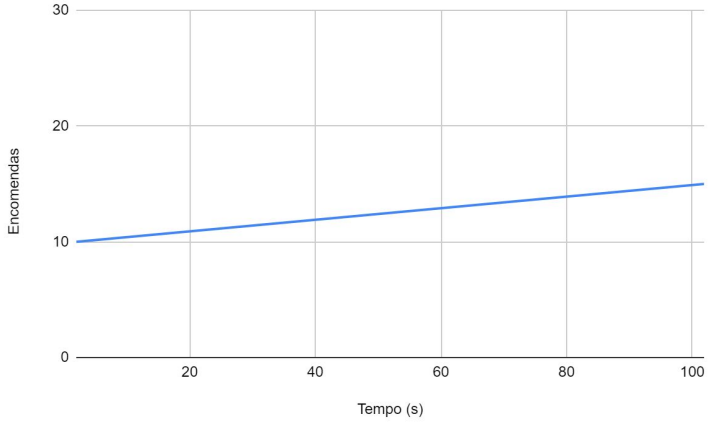
Funcionamento:

1. Seleciona um estafeta
2. Dos pedidos para o dia separa os possíveis de transportar pelo estafeta.
3. Gera **todas** as combinações de entregas possíveis
4. Seleciona a que tem o tempo de entrega mais próximo da *timeframe* e ordena os pedidos ascendentemente.

Avaliação Empírica



Greedy



Backtracking

Exemplo de Execução

Solução algorítmica de destaque.

Um algoritmo greedy faz a melhor escolha em cada passo com o objetivo gerar uma solução globalmente ótima. Com estas escolhas, garante que uma solução é alcançada com uma complexidade relativamente pequena.

Principais dificuldades encontradas

Criar um algoritmo baseado nos problemas clássicos.
Utilização de Dynamic Programming para a resolução dos problemas.

Conclusões

Por vezes, temos de ficar satisfeitos por uma solução boa, e não ótima.
Problemas O(2^n) não são viáveis para grandes conjuntos de dados.

Esforço de cada elemento do grupo

Cada elemento trabalhou de forma igual, sendo o esforço dividido equitativamente.