Table of Contents

[1. Descrição do tema 1](#_Toc68624371)

[2. Formalização do problema 1](#_Toc68624372)

[2.1. Dados de entrada 1](#_Toc68624373)

[2.2. Dados de Saída 2](#_Toc68624374)

# Descrição do tema

A **PapaRica** é uma empresa de confeção e distribuição de refeições prontas a consumir. Para tal, existe uma frota de veículos encarregue das entregas ao cliente, sendo que as recolhe diariamente em Vila do Conde para as distribuir na área metropolitana do Porto.

As encomendas são distribuídas em cabazes, sendo identificadas com a informação relevante acerca do destino e conteúdo.

O trabalho consiste em implementar um sistema que calcule os trajetos ótimos para a realização das entregas.

Numa primeira fase vai-se considerar a existência de um único veículo de entrega com capacidade ilimitada, sendo que se expande numa segunda fase a uma frota de veículos de diferentes capacidades e tipos.

Um exemplo de um trajeto de um veículo pode ser:

Sede -> Cliente 1 -> … -> Cliente n -> Sede

Para uma entrega poder ser realizada tem de existir pelo menos um trajeto possível que permita sair e retornar à Sede passando por todos os Clientes, tendo em consideração a existência de obras públicas que podem, em certas situações, impossibilitar a entrega de certas encomendas.

# Formalização do problema

## Dados de entrada

* Vi – Sequência de veículos disponíveis para utilização, sendo Vi[n] o n-ésimo elemento e caracterizado por:
* type – Tipo de veículo (numa fase inicial = ‘heavy’);
* cap – Capacidade (numa fase inicial = ∞).
* Ci – Sequência de cabazes para distribuição, sendo Ci[n] o n-ésimo elemento e caracterizado por:
* clientName – Nome do destinatário;
* packageNumber – Número de embalagens contidas no cabaz;
* invoiceNumber – Número da fatura;
* destAddress – Vértice de destino.
* Gi = (Ni, Ei) – Grafo dirigido pesado inicial, composto por:
* N – Vértices do grafo, que representam pontos da cidade, caracterizados por:
* type – Tipo de vértice;
* address – Endereço;
* adj ⊆ E – Conjunto de arestas que se iniciam do vértice.
* E – Arestas do grafo, que representam o caminho entre 2 vértices, caracterizados por:
* weight – Peso da aresta, que representa a distância;
* dest Ni – Vértice de destino;
* ID – Identificador de aresta.
* S Ni – Vértice que representa a Sede, de onde o veículo sai e retorna.

## Dados de Saída

* Gf = (Nf, Ef) – Grafo dirigido pesado final, sendo que Nf = Ni e Ef = Ei.
* Vf – Sequência de veículos utilizados em entregas, sendo Vf[n] o n-ésimo elemento. Cada um é caracterizado por:
* T – Sequência de arestas ordenadas por ordem de passagem (com possibilidade de repetidas);
* B – Sequência de cabazes para entrega pelo veículo ordenada pela ordem de entrega;
* cap – Capacidade ocupada do veículo.

## Restrições

### Restrições sobre os dados de entrada

* n [0, |Vi| - 1]:
  + type(Vi[n]) = ‘light’ ‘heavy’ ‘motorcycles’ – tipos de veículos têm de ser carros ligeiros (‘light’), carros pesados (‘heavy’) ou motociclos (‘motorcycles’);
  + cap(Vi[n]) >= 0 – capacidade têm de ser maior ou igual a zero, visto que se trata de uma quantidade de embalagens.
* n [0, |Ci| - 1]:
  + packageNumber(Ci[n]) >= 1 – número de embalagens tem de ser positivo, visto que um cabaz tem de ter pelo menos uma embalagem;
  + invoiceNumber(Ci[n]) >= 1 – número de fatura tem de ser maior ou igual a um;
  + destAddress(Ci[n]) Ni – vértice de destino tem de existir no conjunto de vértices do grafo.
* n Ni, type(n) = ‘HQ’ ‘destiny’ ‘intermediate’.
* e Ei:
  + weight(e) >= 0 – arestas têm de ter um peso igual ou maior que zero visto que este corresponde à distância;
  + ID(e) >= 0 – identificador de uma aresta tem de ser maior ou igual a zero e único para cada uma;
  + Deve ser utilizável por veículos.
* type(S) = ‘HQ’ – tipo de sede deve ser ‘HQ’.

### Restrições sobre os dados de saída

* – os vértices iniciais e finais vão ser iguais.
* – as arestas iniciais e finais vão ser iguais.
* Vf Vi – os veículos utilizados têm de ser parte do conjunto de veículos disponíveis.
* v Vf:
  + T(v) Ei – as arestas de passagem têm de fazer parte do conjunto de arestas iniciais;
  + B(v) Ci – o conjunto de cabazes para entrega tem de fazer parte do conjunto de cabazes inicial;
  + cap(v) = – a capacidade utilizada de um veículo tem de ser igual ao número de embalagens totais em cabazes;
  + Capacidade utilizada do veículo tem ser menor ou igual à capacidade disponível do veículo.
* T[0] adj(S) dest(T[|T| - 1]) = S – o trajeto de todos os veículos começa e acaba na sede.

## Funções objetivo

O objetivo do trabalho é encontrar o mínimo número de veículos para entregar as encomendas e o menor trajeto para cada veículo possível. Para tal é necessária a minimização de duas funções, sendo que uma se refere ao menor número de veículos e a outra ao menor trajeto possível:

* f = |Vf|
* g =

Numa fase inicial devido à consideração de uma capacidade infinita de veículos, a minimização da função f é desnecessária, sendo que na fase seguinte se prioriza a sua minimização à função g.

# Perspetiva de Solução

## Primeira fase

Inicialmente vai ser considerada a existência de apenas um veículo com capacidade infinita para a distribuição das encomendas, sendo então apenas necessário encontrar o trajeto ótimo para o mesmo. Para tal, seguem-se as próximas etapas:

1. Remoção das arestas indesejáveis, que vão ser as inutilizáveis pelos veículos.
2. Verificar se os pontos de interesse (Sede e clientes) pertencem a uma componente fortemente conexa.
3. Ordenação dos pontos de interesse.
4. Calcular as arestas a percorrer no trajeto.

## Segunda fase

Numa segunda fase, já vão ser considerados vários veículos de diferentes tipos e capacidades, pelo que é necessária uma distribuição ótima das encomendas pelos veículos disponíveis antes do cálculo do trajeto ótimo. Para tal, são necessárias as seguintes etapas:

1. Remoção das arestas indesejáveis.
2. Distribuir as encomendas pelos veículos disponíveis.
3. Verificar se os pontos de interesse (Sede e clientes) pertencem a uma componente fortemente conexa.
4. Ordenação de pontos de interesse para cada veículo, considerando os pontos de entrega específicos do trajeto.
5. Calcular as arestas a percorrer no trajeto de cada veículo.

## Remoção das arestas indesejáveis

A remoção vai consistir em encontrar as arestas inutilizáveis por veículos, atribuindo ao seu peso o valor de infinito.

## Distribuição das encomendas

Esta etapa vai ser realizada utilizando uma estratégia de programação dinâmica segundo o seguinte algoritmo, baseado no algoritmo da mochila lecionado:

* Calcular a melhor combinação para todos os veículos de capacidade 1 até C (capacidade pretendida);
* Começar por considerar que só se pode usar o primeiro cabaz, depois os cabazes um e dois, etc., e finalmente todos os itens de 1 até N (número total de cabazes).

PSEUDO-CÓDIGO

## Verificar se os pontos pertencem a um CFC

Este passo vai ser realizado utilizando o método lecionado na cadeira:

* Pesquisa em profundidade no grafo G determina floresta de expansão, numerando vértices em pós-ordem;
* Inverter todas as arestas de G;
* Segunda pesquisa em profundidade, em Gr, começando sempre pelo vértice de numeração mais alta ainda não visitado;
* Cada árvore obtida é um componente fortemente conexo.
* Percorrer cada CFC e verificar se têm todos os vértices do trajeto (Sede e clientes), sendo que se nenhum CFC for encontrado não vai existir um trajeto possível.

?PSEUDO-CÓDIGO?

## Ordenação dos pontos de interesse do trajeto

Nesta etapa utiliza-se um cálculo da densidade do grafo:

Considerando que uma densidade igual ou inferior a ½ representa um grafo esparso e superior um grafo denso, utilizam-se nas seguintes situações os algoritmos:

* Grafo esparso – Djistrka
* Grafo denso – Floyd-Warlshall

Depois -> Maybe Backtracking / Maybe Dynamic programming -> Git Educorreia932