

**Mestrado Integrado em Engenharia Informática**

**e Computação**

2.º Ano

Relatório do projeto da Unidade Curricular

**Concepção e Análise de Algoritmos**

Trabalho realizado por **Diogo Maia up201904974@fe.up.pt,**

**Guilherme Garrido up201905407@gmail.com**

**e Luís Lucas up201904624@fe.up.pt**

Turma 2 – Grupo 2

**2020/2021**

Índice

[Descrição do problema 4](#_Toc68515427)

[Formalização do problema 5](#_Toc68515428)

[Descrição da solução 6](#_Toc68515429)

[Casos de utilização 7](#_Toc68515430)

[Conclusão 8](#_Toc68515431)

[Participação de cada membro 9](#_Toc68515432)

[Bibliografia 10](#_Toc68515433)

Enunciado (a remover no final)

Tema: **PapaRica: distribuição de refeições prontas**

Os veículos da PapaRica, uma empresa de confecção e distribuição de refeições prontas a consumir do distrito de Vila do Conde, recolhem, diariamente, as encomendas dos clientes para o almoço e entregam-nos nas casas ou empresas dos clientes. A empresa faz entregas em toda a área metropolitana do Porto. Depois de realizar as entregas, os veículos de distribuição recolhem à garagem, localizada na sede da empresa em Vila do Conde. Este tipo de transporte de refeições prontas a consumir, de elevada qualidade, é cada vez mais frequente em zonas urbanas, permitindo o acesso a refeições de qualidade, saudáveis e confeccionadas de forma caseira, algo cada vez mais valorizado pelos consumidores modernos.

Neste trabalho, pretende-se implementar um sistema que permita à empresa gerir a sua frota e a sua carteira de entregas. As refeições são recolhidas diariamente na sede da empresa onde se encontram já distribuídas por cabazes, identificadas com o nome do destinatário, número de embalagens que contém o cabaz, destino, número de factura, entre outras informações que julgar necessárias. Inicialmente, considere que a empresa tem um único veículo, que é capaz de realizar a entrega de todos os cabazes a aguardar transporte. Numa segunda fase, considere que a empresa conta com uma frota de veículos, de diferentes tipos e capacidades; tente, então, agrupar itens de forma a não exceder a capacidade de carga do veículo, e estejam próximos de forma a minimizar o caminho gerado, desde a empresa até ao regresso à mesma, passando pelos pontos de entrega.

Avalie a conectividade do grafo, de forma a evitar que sejam aceites entregas em zonas inacessíveis da rede. Algumas vezes, obras nas vias públicas podem fazer com que certas zonas tornem-se inacessíveis, tornando as entregas inviáveis.

Considere a possibilidade de utilizar mapas reais, extraídos do OpenStreetMaps (www.openstreetmap.org) e coordenadas geográficas do depósito, da garagem, e dos endereços dos destinatários para as respetivas entregas.

# Descrição do problema

Uma empresa de distribuição de comida pretende implementar um sistema de gestão da sua frota, onde as rotas são determinadas de forma a minimizar o número de veículos utilizados e a distância percorrida por cada um.

Com sede em Vila do Conde, a PapaRica faz recolhas e posteriores entregas de refeições confecionadas em toda a Área Metropolitana do Porto. Os veículos têm como pontos de partida e de chegada a sede da empresa.

**1ª iteração: um único veículo com capacidade ilimitada**

Nesta fase, despreza-se o limite de capacidade do veículo, que irá iniciar uma única viagem com toda a carga necessária de transportar. O objetivo desta iteração é encontrar o caminho mais curto, começando na sede, passando por todos os pontos de entrega e, por fim, regressando novamente ao local de partida em Vila do Conde.

Neste caso, é importante notar que zonas inacessíveis devem ser descartadas de forma a evitar o bloqueio da viagem.

**2ª iteração: um único veículo com capacidade limitada**

Nesta iteração pretende-se encontrar o caminho mais rápido (ou mais curto) respeitando a capacidade de transporte de um único veículo. Assim, este veículo poderá ser submetido a várias viagens se não conseguir **meter toda a carga na bagagem**.

O veículo parte da sede da empresa e faz as entregas correspondentes às refeições que transporta (utiliza-se a 1ª iteração, encontrando o caminho mais curto) e retorna ao ponto de partida para recarregar a bagagem se necessário (ciclo até não haver mais refeições por entregar).

**3ª iteração: frota de veículos com capacidades diferentes**

Nesta terceira fase são considerados vários veículos com capacidades independentes. O objetivo é minimizar o número de veículos utilizados e a distância percorrida.

É importante salientar que um cabaz não pode ser dividido, ou seja, só pode ser transportado por um veículo.

# Formalização do problema

## **Dados de entrada**

**baskets** - lista de cabazes

**C** – cabaz:

* **clientName** - nome do destinatário
* **numPack** - número de embalagens
* **idDest** – identificador do vértice de destino
* **numFat** - número de fatura

**fleet** – lista de camiões

**T** – camião:

* **maxCap** - capacidade de cada veículo
* **baskets** – lista de cabazes dentro do veículo
* **path** - lista de vértices do caminho deste veículo

**Gi = (Vi, Ei)** - grafo da área metropolitana do Porto:

* **V** – vértices (pontos de interesse em que é possível a entrega de cabazes)
  + **id** – identificador do vértice
  + **adj ∈ E** – conjunto de arestas adjacentes
* **E** – arestas (representam as ligações entre os pontos, vértices)
  + **w** - Peso da aresta (pode variar conforme o tipo de estrada)
  + **dest** [**∈**](https://pt.wiktionary.org/wiki/%E2%88%88) **V** – Destino
  + **name** – identificador da aresta (por ex: rua)

**S** - vértice inicial (representa a sede da empresa em Vila do Conde)

## **Dados de saída**

fleet - lista dos veículos com caminho percorrido com um os parâmetros “baskets” e “path” atualizados.

## **Restrições**

Dados de Entrada:

* **∀ T ∈ fleet ∧ T.path.size = 0: T.baskets.size = 0** , porque cada camião que ainda não esteja em movimento não pode conter cabazes inicialmente.
* **∀ T ∈ fleet: T.maxCap > 0**, já que o camião tem de poder carregar no mínimo um cabaz.
* **∀ C ∈ baskets: ∃ V.id = C.idDest** pois o id que identifica o vértice no cabaz tem de identificar um vértice no grafo.
* **∀ C ∈ baskets: C.numPack > 0** sendo que cada cabaz tem de ter no mínimo uma embalagem.
* **S ∈ V**, logo o ponto inicial tem de ser um vértice do grafo.
* **S.adj.size > 0**, sendo que tem de se garantir a possibilidade de os camiões saírem do vértice inicial, havendo, necessariamente, pelo menos um caminho de saída da sede.
* **∀ E.w > 0**, uma vez que se se associar cada vértice a uma distância, tem de existir uma distância entre cada vértice (não pode haver dois vértices na mesma localização).

Dados de Saída:

* **∀ T ∈ fleet ∧ T.path.size ≠ 0: T.path[0] = S**, porque o início do caminho do camião é sempre a empresa.
* **∀ T ∈ fleet ∧ T.path.size ≠ 0: T.path ⊆ V**, uma vez que todos os vértices do caminho do camião têm de ser vértices do grafo.
* **∀ T ∈ fleet ∧ T.path.size ≠ 0: ∃ T.path[1] = S.adj.dest**, já que o vértice a seguir ao inicial tem de ser adjacente e esta lógica mantém-se em todos os vértices pertencentes ao “path”.
* **∀ T ∈ fleet: T.maxCap(antes) = T.maxCap(depois)**, sendo este um dado que não será alterado ao longo do algoritmo.
* **baskets.size = 0**, porque todos os cabazes foram atribuídos a um camião.
* **baskets.size(antes) = ∑ T.baskets.size**, já que todos os cabazes existentes têm de ser distribuídos pelos camiões.

Funções objetivo:

# Descrição da solução

## **Algoritmos a implementar**

**Aceitação de pedidos de Cabazes**

Na receção do pedido de um cabaz, o sistema deve procurar pelo vértice ao qual o cabaz deve ser entregue através do id fornecido (morada). Desta forma, pensamos que seria melhor utilizar uma adaptação do algoritmo de pesquisa em profundidade.

O algoritmo de pesquisa DFS é um algoritmo utilizado pata percorrer ou buscar itens dentro de um grafo. Define-se por percorrer todos os nós filho ao nó raiz o mais profundo possível para somente retroceder quando não for possível continuar nessa direção. É um algoritmo recursivo e assumindo que um grafo tem V vértices e A arcos, consome tempo proporcional a V+A.

**Distribuição de Cabazes pelos Camiões**

Na primeira iteração, este algoritmo consiste em passar todos os cabazes para o único camião. No entanto, nas restantes iterações, a ideia é dividir os cabazes entre cada um dos camiões da frota.

Primeiro, encher um camião. Se ainda sobrar cabazes e houver mais camiões, encher os outros de seguida. Caso haja mais cabazes do que a soma da capacidade máxima de todos os camiões, o primeiro camião a retornar estará encarregue de o que conseguir e assim em diante até todos os cabazes serem entregues.

**Identificação do pedido mais perto do camião**

A cada camião utilizado, procurar dentro dos seus pedidos o cabaz com o local de entrega mais perto do vértice atual do camião. Utilizando um algoritmo de pesquisa de vértices.

**Determinar o percurso mais curto entre os dois vértices e fazer o camião percorrê-lo**

Após determinar a entrega mais próxima do local atual do veículo, procede para a entrega, retirando a entrega da lista de cabazes do camião e adicionando este vértice à lista de vértices do camião em questão.

Estes dois últimos passos são repetidos até todos os cabazes do camião serem entregues. Aí, o camião tem de retornar à empresa e voltar ao estado zero (estar pronto para ser utilizado novamente).

# Casos de utilização

# Conclusão

# Participação de cada membro

# Bibliografia