

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | O Padeiro da Vila em Época Covid | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | A picture containing text  Description automatically generated |  | | | |
|  | | | | 07 / 04 / 2021  —  Conceção e Análise de Algoritmos  —  Bruno Rosendo  Domingos Santos  João Mesquita |  | | | |
|  | | |  | | |  | | |

|  |
| --- |
|  |

# Índice



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | |
|  | | Descrição do Problema | | |  | |
|  | |  |  |  |  | |
|  | O trabalho incide sobre a distribuição de pães, começando na padaria e passando pela casa de todos os clientes que aderiram. Por fim, o padeiro tem ainda de voltar à padaria.  Cada cliente indica a hora preferencial para que o pão lhe seja entregue, tendo uma margem de tolerância definida (antes e depois da hora). Quando o padeiro chega a uma habitação, demora uma certa quantidade de tempo a efetuar a entrega.  A solução ótima é uma rota a seguir, minimizando o tempo total do itinerário e equilibrando o tempo de atraso nas entregas. Para simplificar o problema, dividimo-lo em diversas fases: 1ª Fase: Minimizar o tempo do itinerário sem considerar hora de entrega e apenas uma carrinha de capacidade ilimitada Numa primeira fase, despreza-se a hora preferível dos clientes e as entregas são todas feitas por uma única carrinha de capacidade ilimitada. Assim, o problema resume-se a encontrar o trajeto mais curto (neste caso, o menos demorado) que começa na padaria, passa por todas as moradas dos clientes e volta à padaria.  É importante notar que as entregas só conseguem ser efetuadas se existir pelo menos um caminho que liga as moradas de todos os clientes e a padaria (tanto de ida como de volta), ou seja, todos os vértices de entrega devem fazer parte do mesmo componente fortemente conexo. Assim, é necessário avaliar a conetividade do grafo subjacente à zona considerada, com o fim de identificar moradas com pouca acessibilidade, cujas estradas podem ter sido obstruídas por imprevistos como obras públicas (se alguma morada não fizer parte deste componente, a entrega a esse cliente será cancelada). 2ª Fase: Entrega considerando a hora preferencial, equilibrando o tempo de atraso, com uma carrinha de capacidade ilimitada Numa segunda fase, teremos em consideração a ordem de entrega aos clientes, de modo a satisfazer a hora preferencial deles.  Como consequência desta nova restrição, teremos que minimizar dois parâmetros: o tempo total do itinerário e o tempo de atraso das entregas, sendo que se dará prioridade ao último critério, de modo a que os clientes não tenham de esperar mais do que precisam. 3ª Fase: Entrega com múltiplas carrinhas de capacidade limitada Numa última fase, passa-se a considerar múltiplas carrinhas com capacidade limitada, ou seja, cada cliente pede X pães e cada carrinha entrega Y pães, sendo que o objetivo é minimizar o número de carrinhas utilizadas. Assim sendo, no início do algoritmo, cada carrinha será alocada aos seus respetivos clientes.  Com esta adição, temos três critérios de minimização, sendo eles, por ordem decrescente de prioridade: número de carrinhas alocadas, tempo de atraso das entregas e tempo total do itinerário.  Além disso, como cada carrinha tem um condutor diferente, o tempo de cada entrega também será diferente (o condutor pode ser mais ou menos falador) Formalização do ProblemaDados de Entrada Ci- Sequência de carrinhas na padaria, sendo Ci(i) o seu i-ésimo elemento. Cada um é caracterizado por:   * Q- quantidade total de pães que a carrinha consegue transportar (infinito nas 1ª e 2ª fases) * T- tempo que o condutor demora a fazer a entrega   Gi = (Vi, Ei) – Grafo dirigido pesado, composto por:   * V- vértices, representando pontos da rede rodoviária, com:   + ID- identificador do vértice   + Adj ⊆ E, arestas que partem do vértice   + Lat- latitude real do ponto no mapa   + Long- longitude real do ponto no mapa * E- arestas, representando estradas da rede rodoviária, com:   + ID- identificador da aresta   + W- Peso da aresta, que no contexto do projeto representa o tempo aproximado que se demora a percorrer a aresta   + Dest ∈ V- destino da aresta   Ui- Sequência de clientes da padaria, sendo Ui(i) o seu i-ésimo elemento. Cada um é caracterizado por:   * Name- nome do cliente * ID- identificador do cliente * Addr- morada do cliente (deve pertencer aos vértices do grafo) * H- hora da entrega * Q- quantidade de pães na encomenda   S ∈ Vi - vértice da padaria (inicial)  Ra – Raio de ação dos veículos (centro na padaria)  Upper- Tolerância de atraso nas entregas  Lower- Tolerância de adiantamento nas entregas Dados de Saída Gf = (Vf, Ef) – grafo dirigido pesado, tendo Vf e Ef os mesmos atributos que Vi e Ef (à excessão de atributos utilizados pelos algoritmos).  Cf- Sequência de carrinhas com a informação das entregas realizadas, sendo Cf(i) o seu i-ésimo elemento. Cada carrinha é composta por:   * Qf- quantidade final de pães (sobras) * Qe- quantidade entregue de pães * T- Tempo total do itinerário * Ta- Tempo total de atraso nas encomendas * Uf (u ∈ Ui | 1 <= j <= |Uf|) - sequência de clientes a quem a carrinha realizou entregas, sendo Uf(i) o seu i-ésimo elemento. Cada cliente é caracterizado por:   + Name- nome do cliente   + Addr- morada do cliente   + Ht- hora marcada para a encomenda   + Hp- hora real da encomenda * I (i ∈ Ei | 1 <= j <= |I|) – sequência de arestas a percorrer no itinerário, sendo P(j) o seu j-ésimo elemento.  RestriçõesAos dados de entrada:  * ∀ e ∈ Ei, w(e) > 0, visto que w representa o tempo necessário para percorrer uma aresta. * Ra > 0, visto este valor representar a zona de entregas. * S ∈ Vi, visto que a padaria tem que pertencer ao grafo. * Upper > 0 e Lower > 0, visto que é virtualmente impossível chegar a todos os locais no instante exato que foi marcado. * ∀ c ∈ Ci, Q(c) > 0 e T(c) > 0, visto que não faz sentido usar uma carrinha que não tem pães para entregar e essas entregas não podem ser instantâneas. * ∀ u ∈ Ui, Q(u) > 0, visto que não faz sentido haver encomendas em que não se entregam pães.   Nota: Não é uma obrigatório a morada do cliente fazer parte do grafo e da zona de entrega, mas, se tal acontecer, essas encomendas serão canceladas. O mesmo acontece com moradas fora da mesma componente fortemente conexa do grafo. Aos dados de saída:  * |Cf|<=|Ci|, visto que pode não ser preciso usar todas as carrinhas. * ∀ vf ∈ Vf, ∃ vi ∈ Vi tal que vi e vf têm os mesmos atributos (à exceção de atributos utilizados pelos algoritmos). * ∀ ef ∈ Ef, ∃ ei ∈ Ei tal que ei e ef têm os mesmos atributos (à exceção de atributos utilizados pelos algoritmos). * ∀ cf ∈ Cf,   + Qe(cf) > 0   + T(cf) > 0   + Ta(cf) >= 0   + ∃ ci ∈ Ci tal que Q(ci) = Qf(cf) + Qe(cf)   + ¬ ∃ c2 ∈ Cf tal que cf = c2   + ∀ i ∈ [1, |Uf|-1], Hp(Uf[i]) < Hp(Uf[i+1]) * ∀ uf ∈ Uf,   + ∃ ui ∈ Ui tal que name(uf)=name(ui), addr(uf)=addr(ui) e Ht(uf)=H(ui) * Seja e1 o primeiro elemento de I, é necessário que e1 ∈ adj(S), pois cada carrinha parte da padaria. * Seja ef o último elemento de I, é necessário que dest(ef)=S, pois cada carrinha regressa à padaria.  Funções Objetivo A solução ótima do problema passa por minimizar o número de carrinhas usadas, o tempo de atraso das entregas e o tempo total do itinerário. Logo, é necessário minimizar as seguintes funções:  f = |Cf|  g = ∑c ∈ Cf Ta(c)  h = ∑c ∈ Cf T(c)  Tal como foi dito na descrição do problema, dar-se-á prioridade às funções na ordem f->g->h. | | | | |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | |
|  |  | | | | |  |

Demonstram-se de seguida os principais obstáculos encontrados ao longo das diferentes etapas e as suas respetivas soluções. Serão identificadas as técnicas de conceção e os algoritmos a serem desenvolvidos.

# Perspetiva de Solução

## Pré-Processamento dos Dados de Entrada

### Tratamento do grafo

De modo a melhorar a eficiência temporal dos algoritmos que serão aplicados posteriormente, o grafo G deve ser pré-processado, reduzindo o seu número de vértices e arestas.

Inicialmente, devem ser eliminadas as arestas do grafo que se encontrem inacessíveis, devido a fatores externos, tal como obras públicas ou demasiado afastadas da padaria, tendo em conta que a padaria do Sr. Sílvio apenas efetua encomendas na vila da Tocha, ou seja, num raio (Ra) à volta da padaria (simplificando).

Após a primeira filtragem, será necessário remover os vértices que não pertencem ao mesmo componente fortemente conexo do grafo onde a padaria se encontra. Este segundo fator de eliminação resulta da natureza circular do trajeto, dado que as carrinhas regressam à sua origem, após a finalização das entregas.

Assim, é possível aumentar significativamente a eficiência temporal dos algoritmos que serão aplicados de seguida, garantindo que o grafo não terá vértices nem arestas que não podem ser percorridos pelas carrinhas.

### Tratamento dos Clientes

Inicialmente, a sequência de clientes C deve ser percorrida à procura de clientes cuja morada não pertença ao grafo G. Nos casos em que isto acontece, o respetivo cliente deve ser removido, dada a impossibilidade de efetuar a entrega. Durante esta iteração, será calculado o intervalo de tempo segundo o qual a empresa deverá efetuar a entrega para cada cliente (hora mínima e máxima).

## Identificação dos problemas encontrados

Na primeira fase, com uma única carrinha e sem restringir a ordem de entrega pela hora das encomendas, o problema inerente é o de passar por todos os pontos de interesse e regressar à partida, gastando o menor tempo possível, assemelhando-se vivamente ao problema *NP-hard* chamado ***Travelling Salesman Problem (TSP).***

Numa segunda fase, adiciona-se a restrição da hora de entrega preferível pelos clientes, pelo que o algoritmo concebido para o problema em cima terá de ser modificado, alterando a ordem de passagem pelos pontos de interesse, de modo a equilibrar o tempo de atraso nas entregas.

(Acho que vamos ter de explicar melhor o algoritmo a usar)

Passando para uma frota de carrinhas, na última fase do projeto, o algoritmo deve conseguir decidir o número de carrinhas a usar e a alocação ótima delas aos clientes, de modo a minimizar a função objetivo. Assim sendo, o problema tem parecenças com o ***Vehicle Routing Problem***, uma generalização do TSP, que tínhamos nos pontos anteriores.

## Algoritmos considerados para a 1ª Fase

Como este problema se assemelha ao TSP, existem vários algoritmos que procuram alcançar soluções para este problema. Assim, existem soluções exatas de elevada complexidade temporal e, por outro lado, soluções mais eficientes, que alcançam resultados aproximados.

### Soluções Aproximadas

O **nearest neighbour (NN)** é um algoritmo heurístico, que fornece uma solução razoavelmente eficiente, mas que não é a mais próxima da ótima (em média, possui um desvio de 25%). Este algoritmo, de natureza gananciosa, começa escolhendo um vértice aleatório e, repetidamente, visitando o vértice mais próximo que ainda não foi visitado, até ter percorrido todos os nós.

Observe-se que, neste caso de estudo, o vértice inicial será sempre a padaria, o que remove a aleatoriedade associada a este passo.

O **Bitonic tour** **(BT)** é um algoritmo baseado em programação dinâmica que calcula a cadeia poligonal fechada, de perímetro mínimo, que inclui todos os vértices de um grafo. Este poderá ser aplicado nos vértices de interesse, resolvendo o problema desta fase.

### Soluções Exatas

O algoritmo **naïve** (*brute force*) consistiria em calcular todas as permutações entre vértices do grafo, escolhendo a solução que possui o caminho mais curto. No entanto, possui uma complexidade de O (n!).

O algoritmo **Held-Karp** é outra solução possível, baseada em programação dinâmica, com complexidade O (n2 2n).

Desta forma, se for pretendida uma solução ótima para este problema, será tido em conta o número de vértices (N) a analisar no caso em questão, dado que para N < 8 a técnica de *brute force* apresentará um tempo menor de execução, relativamente ao segundo método apresentado. Consequentemente, para N ≥ 8, o Held-Karp revela-se superior.

## Integração da hora de entrega do pão (2ª Fase)

Como se pretende minimizar o tempo de atraso das entregas para além do tempo de viagem total, podemos reutilizar os métodos previamente enunciados, modificando-os, de modo a ter este parâmetro em conta e a priorizá-lo.

No entanto, nos algoritmos gananciosos não podemos dar prioridade absoluta ao tempo de atraso das entregas, pois iria ser um resultado muito pouco diferente de uma simples ordenação pela ordem de preferência de entrega de cada cliente, que provavelmente estaria muito longe da solução ótima. Para contornar este problema, pode-se calcular um novo peso para cada aresta que chega a casa de um cliente, a partir do tempo de viagem (peso antigo) e do tempo de atraso na iteração atual.

Por exemplo, atribuindo uma prioridade de 80% aos atrasos e 20% ao tempo de viagem.

Isto só me faz sentido se o grafo for feito SÓ por casas e padaria. Se for uma rede rodoviária, temos que por a aresta mais leve enquanto o tempo de atraso aumenta?

Em relação às soluções exatas para o TSP, podemos simplesmente escolher o caminho com menor tempo de atraso e, em caso de empate, menor tempo de viagem.

Devido a caminhos mais curtos e horas de entrega diferentes, é também possível passar por casa de um cliente mais do que uma vez (embora se faça uma única entrega)

## Utilização de múltiplas carrinhas com capacidade limitada (3ª Fase)

TO DO

## Análise da conectividade

Tal como já foi referido anteriormente, durante a fase de pré-processamento do grafo, os vértices que não se encontram no componente fortemente conexo da padaria devem ser removidos do grafo.

Consequentemente, torna-se necessário analisar a conectividade do grafo, de forma a excluir os nós com pouca acessibilidade. Apresentam-se de seguida dois algoritmos para resolver este problema:

### Algoritmo de Kosaraju

Este algoritmo encontra todos os componentes fortemente conexos de um grafo e consiste em fazer uma pesquisa em profundidade, numerando os vértices em pós-ordem, e fazendo, após a inversão de todas as arestas, uma segunda pesquisa em profundidade, começando pelos vértices de numeração superior. No final desta sequência de passos, cada árvore obtida é um componente fortemente conexo. Contudo, apenas interessa o componente que inclui a origem, ou seja, a padaria.

Este método tem uma complexidade O(|V| + |E|), ou seja, corre em tempo linear.

### Algoritmo de Tarjan

O método proposto por Robert Tarjan surge como uma alternativa ao algoritmo de Kosaraju. Apesar de executar em tempo linear, executa apenas uma pesquisa em profundidade, sendo por isso mais eficiente.

## Casos de utilização

A aplicação a desenvolver terá uma interface simples onde o utilizador poderá inserir os dados de entrada e escolher as opções que pretende, tais como:

* Se pretende carregar um ficheiro de texto com toda a informação necessária para o algoritmo ou se quer inserir a informação manualmente.
* Se quer guardar o resultado do programa num ficheiro de texto.
* Se quer utilizar uma única carrinha com capacidade infinita ou várias com capacidades limitadas.
* Se prefere uma solução ótima (e possivelmente mais demorada) ou aproximada (e mais rápida).
* Nome do ficheiro contendo os dados de entrada ou apenas do grafo (coordenadas reais retiradas do [OpenStreetMaps](https://www.google.com/url?q=http://www.openstreetmap.org&sa=D&source=editors&ust=1617931022735000&usg=AOvVaw3H6Cpih-yc3b7jhwuTu6Uk)).
* Caso escolha fazê-lo manualmente, pode inserir informação sobre a(s) carrinha(s), os clientes, o raio de entrega da padaria e as margens de falha para a hora preferencial dos clientes. O vértice de partida será o primeiro no ficheiro do grafo.

Além disso, o programa terá as seguintes funcionalidades:

* Cálculo do caminho ótimo (ou aproximado) para realizar as encomendas.
* Verificação da acessibilidade dos destinos pretendidos (moradas dos clientes).
* Organização e alocação de encomendas a uma lista de carrinhas.
* Toda a informação obtida (trajeto, entregas, tempo total, etc.) pelos pontos anteriores será mostrada ao utilizador (e guardada, se assim for pretendido).

## Conclusão

## Bibliografia

* Slides utilizados nas aulas teóricas de CAL, pelos professores: R. Rossetti, A. Rocha, L. Ferreira, G. Leão, F. Ramos e J. Fernandes
* “Introduction to Algorithms”, Thomas H. Cormen
* Travelling Salesman Problem- [https://en.wikipedia.org/wiki/Travelling\_salesman\_problem](https://en.wikipedia.org/wiki/Travelling_salesman_problem#Computing_a_solution)

<https://en.ppt-online.org/31503>

* Vehicle routing problem- <https://en.wikipedia.org/wiki/Vehicle_routing_problem>
* Strongly Connected Component- <https://en.wikipedia.org/wiki/Strongly_connected_component>
* Heuristics and Local Search- <https://paginas.fe.up.pt/~mac/ensino/docs/OR/CombinatorialOptimizationHeuristicsLocalSearch.pdf>
* Nearest neighbour algorithm- <https://en.wikipedia.org/wiki/Nearest_neighbour_algorithm>
* Tarjan's strongly connected components algorithm- <https://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm>
* Kosaraju's algorithm- <https://en.wikipedia.org/wiki/Kosaraju%27s_algorithm>
* Held–Karp algorithm- <https://en.wikipedia.org/wiki/Held–Karp_algorithm>
* Bitonic tour- [https://en.wikipedia.org/wiki/Bitonic\_tour](https://en.wikipedia.org/wiki/Nearest_neighbour_algorithm)