

Índice

- 1. Introdução
 - 1.1 Descrição do Tema
- 2. Formalização
 - 2.1 Problematização
 - 2.2 Estratificação
 - 2.2.1 Fase I
 - 2.2.2 Fase II
 - 2.2.3 Fase III
 - 2.2.4 Fase IV
 - 2.3 Dados de Entrada
 - 2.4 Dados de Saída
 - 2.5 Restrições
 - 2.5.1 Dados de Entrada
 - 2.5.2 Dados de Saída
 - 2.6 Função Objetivo
- 3. Perspetiva de Implementação
 - 3.1 Caminho mais curto entre dois vértices
 - Algoritmos considerados
 - 3.1.1 Dijkstra Unidirecional
 - 3.1.2 Dijkstra Bidirecional
 - 3.1.3 Algoritmo A*
 - 3.1.4 Algoritmo Floyd-Warshall
 - 3.1.5 Abordagens consideradas na Fase III

1. Introdução

1.1 Descrição do Tema

A EatExpress é um sistema de entrega de comida entre os restaurantes registados na plataforma e os utilizadores da sua aplicação. Quando um utilizador acede à aplicação, escolhe o restaurante e o prato que quer, e o seu pedido é entregue no local onde o utilizador se encontra, por um estafeta que utiliza o seu próprio meio de transporte para lá chegar (a pé, bicicleta, mota, carro etc).

Neste projeto procuramos criar um sistema para gerar os percursos mais rápidos a ser percorridos pelos estafetas, entre as localizações dos restaurantes e dos clientes.

Como base para o estudo, formalização, interpretação, e garantia da eficácia dos algoritmos usados, apoiar-nos-emos nas bases da teoria dos grafos. Adotaremos uma estratégia *bottom-up*, começando com variantes mais simples do problema, e avançaremos, à medida que forem testados e escolhidos novos e mais eficientes algoritmos, até ao sistema mais complexo de funcionamento da aplicação.

Para garantir a plausibilidade da resolução, informação real e atualizada será usada, para garantirmos a sua aplicabilidade em problemas verdadeiramente reais, do nosso mundo.

2. Formalização

2.1 Problematização

Em primeiro lugar, convém dividir todas as questões relativas aos vários componentes da aplicação.

O que se pretende com esta análise é garantir a máxima eficiência na implementação de um algoritmo que permita criar e gerir, de forma rápida e precisa, as rotas dos estafetas de entrega de comida, entre as localizações dos restaurantes e as localizações dos clientes que efetuaram os pedidos.

Não serão equacionadas as implementações da interface da aplicação, ao nível do utilizador, nem serão alvo de análise todos os aspetos não relacionados com grafos, ou algoritmos não abordados.

Assim sendo, o problema reduzir-se-á a encontrar os caminhos mais curtos entre pontos/vértices, num mapa de atuação pré-escolhido, e, por fim, tendo em conta as diversas variantes do problema, definir rotas que permitam navegar pelos percursos, percorrendo a menor distância possível.

■

2.2 Estratificação

A estratégia a seguir, para a resolução deste problema, sugere a divisão do conceito em fases de análise, de implementação e de deliberação. No entanto, como pré-estabelecida tomamos a ideia de que um

estafeta em trabalho, antes de receber a sua rota, pode estar localizado num qualquer ponto do mapa. Esta é a generalização de que existirão sempre, no mínimo, um restaurante e uma localização de um cliente como pontos integrantes da rota, serão o ponto de partida para a análise que se segue.

Convém definir, também, o perfil do estafeta, enquanto entidade que se move pelo percurso estabelecido, com características próprias que serão definidas pelo meio de transporte em que se desloca e pela capacidade máxima que pode transportar. Estas condicionantes influenciarão a escolha da rota e serão tidas em conta, antes de o estafeta iniciar a entrega dos pedidos.

É imperativo realçar também que, em qualquer percurso, o levantamento dos pedidos nos restaurantes terá de ser efetuado antes da sua entrega aos clientes, por uma questão de fiabilidade e praticabilidade real da situação. Este pormenor irá criar certas restrições, que serão exploradas, ao detalhe, mais adiante.

Outro aspeto importante é a existência de zonas de desconectividade, nos mapas - a existência de obstáculos à circulação, particularmente, a ocorrência de obras na via pública, que pode inviabilizar o acesso a certas moradas e restaurantes, ao tornar zonas inacessíveis. Também isto será tido em conta, aquando da esquematização dos algoritmos.

Numa última nota, destaca-se a possível predeterminação de todos os restaurantes e estafetas registados na plataforma, elegíveis de tratar um pedido. Este detalhe será substancialmente importante para a adoção de determinados algoritmos, sobretudo se se impuser, ou não, com base na estratégia a utilizar, um pré-processamento dos dados, em particular, a cada novo registo, de uma destas entidades, na plataforma.

2.2.1 Fase I

Na sua simplicidade, consideramos a existência de apenas um estafeta, que realiza, sequencialmente, os percursos que lhe são impostos. Nesta fase, de modo a focar a análise na essência da problemática, será considerado um meio de transporte qualquer e desvalorizada a capacidade de transporte. Na sua medida, o único funcionário terá rotas na qual concretizará o levantamento de pedidos, nos restaurantes, e a sua entrega, na morada dos clientes. Poderá ser considerado, como parte da exemplificação, o caso atómico de um estafeta que entrega apenas um pedido, entre um restaurante e a morada de um cliente.

2.2.2 Fase II

Nesta fase, serão considerados vários estafetas que utilizam o mesmo meio de transporte e para os quais é delineado um percurso. Aqui, estaremos perante a distribuição simultânea de pedidos, com a avaliação de todas as condicionantes relacionadas com o problema, nesta fase.

Os pedidos estarão, logicamente, organizados numa escala temporal, ficando encarregue por um dado pedido o estafeta que se encontre mais perto do restaurante a ele associado.

2.2.3 Fase III

Esta fase coincidirá com a implementação de variados meios de transporte, o que poderá corresponder à utilização de diferentes mapas, para atender às características das múltiplas entidades que entregarão os pedidos e, em simultâneo, às características das diversas vias.

Para além disso, será imprescindível ter em conta a capacidade máxima que cada estafeta pode

transportar, pois a dimensão da encomenda passa a ser determinante na atribuição dos pedidos. Assim, nesta fase, a atribuição de um pedido a um estafeta fica a depender não só da sua proximidade ao restaurante, mas também das características próprias do meio de transporte por ele utilizado, preferindo os meios mais rápidos para encomendas que impliquem deslocações maiores e atribuindo as encomendas de maior dimensão aos estafetas com maior capacidade de transporte.

Tendo em conta os múltiplos fatores envolvidos na seleção de um estafeta para a realização de um pedido, vai-se optar por, numa pré-seleção, excluir estafetas que não tenham capacidade para o transportar.

Seguidamente, como não se pretende ocupar estafetas com encomendas de dimensão muito inferior à sua capacidade máxima de transporte, por poderem ser necessários para os pedidos seguintes, terá que se avaliar, simultaneamente, a capacidade máxima de transporte de cada estafeta e a sua distância ao restaurante. Para isso, será atribuído um peso de 50% a cada um destes fatores aquando da escolha entre estafetas para a realização de um pedido.

Para além disso, será necessário, ao avaliar a possibilidade de escolha dos estafetas, definir uma distância máxima para aqueles que se desloquem a pé, ou de bicicleta, já que a utilização destes meios de transporte para longas distâncias, apesar de não se refletir na distância total percorrida, resultaria, em situações reais, num tempo de entrega extremamente longo.

Por fim, consideraremos também a possibilidade de um pedido englobar vários restaurantes. Isto implicará, mais uma vez, escolher, em primeiro lugar, os estafetas elegíveis, com base na sua disponibilidade para transportar a dimensão do pedido. Posteriormente, delinear-se-á a melhor opção de percurso para cada estafeta recolher o pedido dos vários restaurantes e entregá-lo ao cliente, considerando para a decisão final todas as restrições descritas anteriormente, no que diz respeito à capacidade, distância e tipo de transporte utilizado.

2.2.4 Fase IV

Aqui, entrará em consideração a existência de vários obstáculos nas vias, identificados em cima, o que levará a uma seleção mais restritiva do percurso e do tipo de estafeta encarregue de determinado pedido. Além disso, o tratamento dos grafos poderá sofrer alterações, sobretudo ao nível do pré-processamento.

■

2.3 Dados de Entrada

Os dados a recolher, antes da execução de qualquer dos algoritmos, para os percursos de cada um dos estafetas, são generalizáveis na seguinte lista:

- Grafo $G(V, E)$, dirigido, representando o mapa de vias (poderemos estar perante múltiplos grafos como opção para vários tipos de veículo), a percorrer pelos estafetas, onde estão localizados, como parte integrante, os pontos de recolha e entrega dos pedidos.
 - Cada vértice $v \in V$ terá os seguintes atributos:
 - Uma lista de arestas $Adj(v) \in E$, que partam desse vértice;
 - Um id , identificativo;
 - Um par de coordenadas $coords$, representando a localização real do respetivo ponto;

- Cada aresta $e(v, u) \in E$, que parta de um dado vértice v será caracterizada por:
 - Um vértice $u \in V$ de destino;
 - Um peso *weight* associado, relacionado com o seu comprimento real, expresso numa medida de comprimento espacial;
 - Um estado *state*, que representará a transitabilidade da via;
 - Um campo *name*, sem valor expressivo, servindo de identificador;
- Conjunto de pontos R , representando os restaurantes registados na plataforma:
 - Restaurante $r \in R$ com atributos:
 - O seu *id*, único, identificativo;
 - A referência para o vértice v do grafo, que representa a sua posição;
- Conjunto de estafetas *Employees*:
 - Estafeta $employee \in Employees$, com a informação sobre:
 - A sua posição inicial/no momento, s , um vértice/ponto no mapa;
 - A sua capacidade máxima de transporte, *maxCargo*;
 - O seu meio de transporte, *type*;
 - A sua disponibilidade, *ready*, para iniciar uma nova tarefa;
- Lista ordenada de Pedidos P :
 - Pedido $p \in P$ com a informação detalhada sobre:
 - Data e Hora, *time*, em que foi realizado;
 - Lista, *checkPoints*, de pontos/vértices que façam, obrigatoriamente, parte do percurso (referência apenas aos restaurantes e localização dos clientes);
 - Carga total, *cargo*, ocupada pelos itens que o constituem;

Estes dados poderão fazer parte de um pré-processamento existente, consoante o algoritmo escolhido e o problema a resolver. Se esses cálculos iniciais existirem, no caso de um algoritmo que estabeleça as distâncias mínimas entre cada par de vértices, seguir-se-á, então, a inevitável instanciação dos pedidos, com a sua organização em tarefas e a requisição de estafetas para as realizar, terminando no cálculo do caminho final a ser percorrido pelos mesmos.

2.4 Dados de Saída

Tendo sido o grafo analisado, tratado e traduzido numa das várias formas plausíveis de representação, poderá ser retornado, como dado de saída.

O importante, realmente, aqui, é a entrega das tarefas aos estafetas, o que resultará nos seguintes dados:

- Conjunto de tarefas *Tasks*:
 - Tarefa $task \in Tasks$, com informação sobre:
 - A lista completa e ordenada dos vértices consecutivos, *path*, por onde passará o estafeta;
 - O pedido P , que faz parte da tarefa;
 - O estafeta *employee*, encarregue da sua realização;

O que se segue, nomeadamente, o ato de percorrer o caminho, com a identificação dos pontos já visitados, da entrega dos pedidos, do cálculo da distância percorrida, é um tanto independente desta

análise algorítmica inicial, mas poderá ser, eventualmente, alvo de estudo e reflexão, para interpretação de resultados.

2.5 Restrições

A primeira restrição prende-se com o tamanho do grafo. Em termos reais, aplicações deste género, por uma questão de praticabilidade, limitam as zonas de atuação a áreas urbanas, onde exista um número razoável de restaurantes registados e de estafetas em operação. Assim sendo, os grafos aqui analisados também terão a sua área limitada.

As outras restrições são como se seguem:

2.5.1 Dados de Entrada

- $\forall v_1, v_2 \in V, v_1 \neq v_2$, no sentido em que não haverá vértices repetidos;
- $\forall e_1, e_2 \in E, e_1 \neq e_2$, no sentido em que não haverá arestas repetidas;
- $\forall e \in E, e.weight > 0, e.state \in \{0, 1\}$, já que uma aresta - via/rua - tem um comprimento definido e encontra-se transitável, ou não transitável;
- $\forall r_1, r_2 \in R, r_1 \neq r_2$, no sentido em que não haverá restaurantes repetidos;
- $\forall r \in R, r.v \neq null$, sendo que um restaurante tem que ter uma posição estabelecida;
- $\forall employee_1, employee_2 \in Employees, employee_1 \neq employee_2$, no sentido em que não haverá estafetas repetidos;
- $\forall employee \in Employee, employee.s \neq null$, sendo que um estafeta tem que ter uma posição estabelecida;
- $\forall employee \in Employee, employee.maxCargo > 0, employee.type \in \{'car', 'bike', 'foot'\}, employee.ready \in \{0, 1\}$;
- $\forall p_1, p_2 \in P, p_1 \neq p_2$, no sentido em que não haverá pedidos repetidos;
- $\forall p \in P, p.time \neq null, |p.checkPoints| > 1, p.cargo > 0$;

2.5.2 Dados de Saída

- $\forall task_1, task_2 \in Tasks, task_1 \neq task_2$;
- $\forall task \in Tasks, task.path \neq null, task.P \neq null, task.employee \neq null, |task.path| > 1$;
- $\forall task \in Tasks$, em $task.path$, os restaurantes terão de ser visitados antes da localização do cliente, em relação ao pedido associado a ambos;

2.6 Função Objetivo

A solução ótima para este problema reside, genericamente, na minimização da distância percorrida, em cada rota, por cada estafeta, numa dada tarefa. Assim, será necessário encontrar o mínimo de:

$$f(task) = \sum_{e(v_k, v_{k+1})} e.weight, v_k, v_{k+1} \in task.path \wedge e \in E$$

Adicionalmente, para múltiplas tarefas a realizar em simultâneo, num dado intervalo de tempo, poderá surgir a necessidade de minimizar, também, as funções:

$$g = \sum_{task} f(task) , task \in Tasks$$
$$h = |Tasks|$$

Para a função g , o mínimo poderá, eventualmente, ser atingido pela simples minimização de f , em todas as instâncias, sendo, por isso, a função f de maior prioridade, entre todas.

3. Perspetiva de Implementação

3.1 Caminho mais curto entre dois vértices

Em cada uma das fases de análise anteriormente deliberadas pode-se, seguramente, realçar como problema comum a determinação do caminho mais curto entre dois vértices numa rede viária. Em particular, enquadra-se perfeitamente nesta problemática a necessidade de, aquando da realização de um pedido, determinar o percurso que permite proceder à sua entrega, recorrendo ao trajeto de menor distância. Isto implicará:

- determinar o percurso transitável mais curto, que permita a cada um dos estafetas alcançar um ou mais restaurantes dos quais o pedido foi solicitado;
- calcular, no caso de pedidos realizados num só restaurante, o trajeto de menor distância entre o restaurante e o ponto de entrega especificado pelo cliente.

Assim, é pertinente analisar a variedade de algoritmos e alternativas de implementação disponíveis para o cálculo do caminho mais curto entre dois vértices. Só a realização desta análise prévia tornará possível perceber o impacto das vantagens e desvantagens de cada algoritmo e das suas variantes, no contexto da temática a ser trabalhada.

Salienta-se, portanto, que nenhum algoritmo a ser considerado na fase de implementação deve ser, ainda numa fase experimental, tomado como solução ideal, pois a sua aplicabilidade está dependente da dimensão e tipologia dos grafos de entrada e de uma análise cuidada da sua eficiência temporal e espacial, para cada caso.

Destaca-se ainda que, numa terceira fase, os resultados obtidos através da implementação dos algoritmos a ser analisados vão ser conciliados com as múltiplas restrições impostas pela temática. Pretende-se que, no processo de escolha de um estafeta para a realização de um pedido, a distância mínima que este precisa de percorrer para entregar o pedido em questão não seja o único fator determinante na sua escolha em detrimento da escolha de outro estafeta. Assim, o estafeta cuja distância seja a mínima obtida para a concretização de uma encomenda, não será obrigatoriamente selecionado para o realizar, sendo, também, fatores relevantes para a escolha do estafeta a capacidade que este pode transportar e o tipo de veículo por ele utilizado.

Algoritmos considerados

3.1.1 Dijkstra Unidirecional

A primeira alternativa de solução que pretendemos testar, no que diz respeito à eficiência temporal e aplicabilidade, no contexto do problema, passa essencialmente pela implementação de uma das múltiplas variantes do algoritmo de Dijkstra.

Este algoritmo ganancioso poderá ser vantajoso, na medida em que procura, a cada iteração, obter o melhor resultado possível, minimizando a distância percorrida. Para além disso, é logicamente aplicável ao grafo que será utilizado como dado de entrada, já que este se trata de um grafo dirigido - cada aresta representa uma via unidirecional; e pesado - o peso de uma aresta representa uma distância.

A variante a implementar permite, recorrendo a uma fila de prioridades (heap com mínimo à cabeça) como estrutura auxiliar, obter a distância mínima desde o vértice de origem até todos os outros vértices do grafo, garantindo-se um tempo de execução de $O((|V| + |E|) \times \log|V|)$. Este tempo resulta das $|V|$ operações executadas na fila de prioridade (inserções/extrações), realizadas, cada uma, em tempo logarítmico, e da utilização da operação "Decrease-Key", realizada, no máximo, uma vez por aresta, com um tempo também logarítmico, para a fila de $|V|$ elementos.

Note-se que, nas fases previamente definidas, há a necessidade de aplicar o algoritmo mais do que uma vez:

- considerando como vértice de origem a posição atual de cada um dos estafetas disponíveis para efetuar a entrega de um pedido;
- assumindo que o vértice de origem é o restaurante do qual foi solicitado o pedido.

No entanto, o que se pretende realmente é utilizar o algoritmo de Dijkstra como base para encontrar o caminho mais curto entre dois pontos, pelo que se vai descobrir o caminho mais curto do vértice de origem para todos os outros, terminando o algoritmo quando se for processar o vértice que procuramos, uma otimização que evita continuar a processar vértices, quando já se descobriu o caminho pretendido.

Assim, como, nas duas primeiras fases de implementação, a escolha do estafeta a realizar o pedido depende exclusivamente da sua proximidade ao restaurante do qual este foi solicitado, é necessário determinar, em primeiro lugar, o caminho mais curto para cada um dos estafetas se deslocar até aos respetivos estabelecimentos, sendo, em cada situação, o vértice de destino o restaurante. De seguida, aplica-se, novamente, o algoritmo, para encontrar o percurso mais curto do restaurante à morada escolhida para o ato de entrega, usando o vértice de destino correspondente à localização indicada pelo cliente.

Espera-se, ao aplicar este algoritmo, em alguns casos específicos, entre os quais, mapas de estradas com poucos vértices e distâncias curtas, obter tempos de execução razoáveis. No entanto, para trajetos longos e mapas de estradas complexos, a utilização do algoritmo de Dijkstra, na sua variante unidirecional, gerará, certamente, tempos de execução demasiado longos, o que nos levará, provavelmente, a experimentar otimizações e outros algoritmos que permitam obter, com maior eficácia temporal, o caminho mais curto para a entrega de um pedido.

3.1.2 Dijkstra Bidirecional

Uma das otimizações que se procura explorar com mais detalhe na fase de implementação, para o algoritmo de Dijkstra, tendo em conta que o resultado esperado para a variante unidirecional não é o desejado, é a variante bidirecional. A execução alternada do algoritmo de Dijkstra, no sentido do vértice de origem para o de destino, e no sentido inverso, poderá trazer vantagens no que diz respeito ao tempo de execução, esperando-se que este seja reduzido para metade, em comparação com a variante do algoritmo que utiliza pesquisa unidirecional.

Esta melhoria deriva do facto de a área processada diminuir para metade, já que a pesquisa passa a ser executada nas duas direções, mantendo-se sempre a distância mais curta descoberta até ao momento e verificando-se, ao processar uma aresta já processada previamente, na direção oposta, se foi descoberta uma distância menor.

Deste modo, a variante bidirecional do algoritmo de Dijkstra vai ser aplicada com o intuito de melhorar o tempo de cálculo do percurso mais curto entre o estafeta e o restaurante e, posteriormente, entre o restaurante e o local de entrega, no caso de pedidos que envolvam um só restaurante.

3.1.3 Algoritmo A*

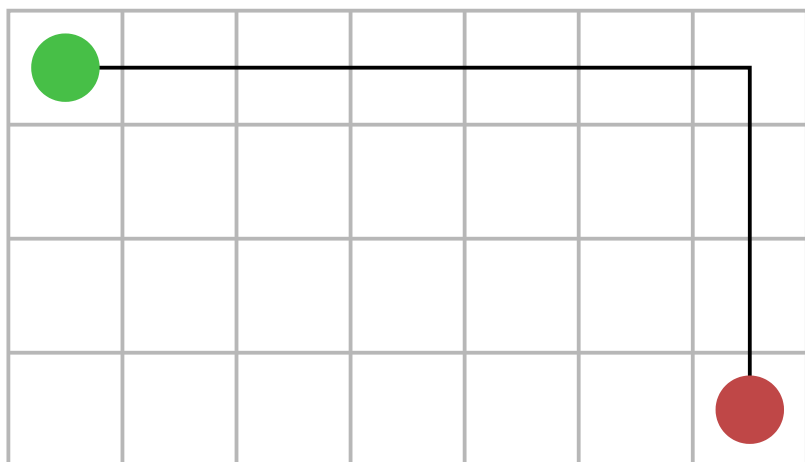
O algoritmo A* será, também, objeto de estudo, na fase de implementação, não só por ser mais um algoritmo que permite, igualmente, calcular o caminho mais curto entre dois vértices, mas, essencialmente, pela sua performance se diferenciar do algoritmo de Dijkstra, pelo facto de recorrer a heurísticas para manipular os pesos das arestas e, assim, afetar os nós que são expandidos.

Antes de mais, é essencial escolher, cuidadosamente, a heurística a utilizar, porque esta desempenhará um papel determinante no comportamento do algoritmo e nos resultados obtidos.

Posto isto, tendo em conta as propriedades dos vértices dos grafos a tratar, foram consideradas três funções de heurística diferentes que farão parte do processo de implementação e análise deste algoritmo em concreto. Todas elas atuarão em função das coordenadas dos pontos no mapa e darão, sempre, um valor aproximado menor que o custo real do percurso, entre os vértices escolhidos.

A primeira estratégia tem o nome de **Manhattan Distance** e calcula a distância, como número de quadrados percorridos, em ambas as direções, desde o ponto inicial, até ao final. Tem particular interesse em mapas com a forma de grelha e usa, para o efeito, a seguinte fórmula generalizada:

$$h = |x_{start} - x_{destination}| + |y_{start} - y_{destination}|$$



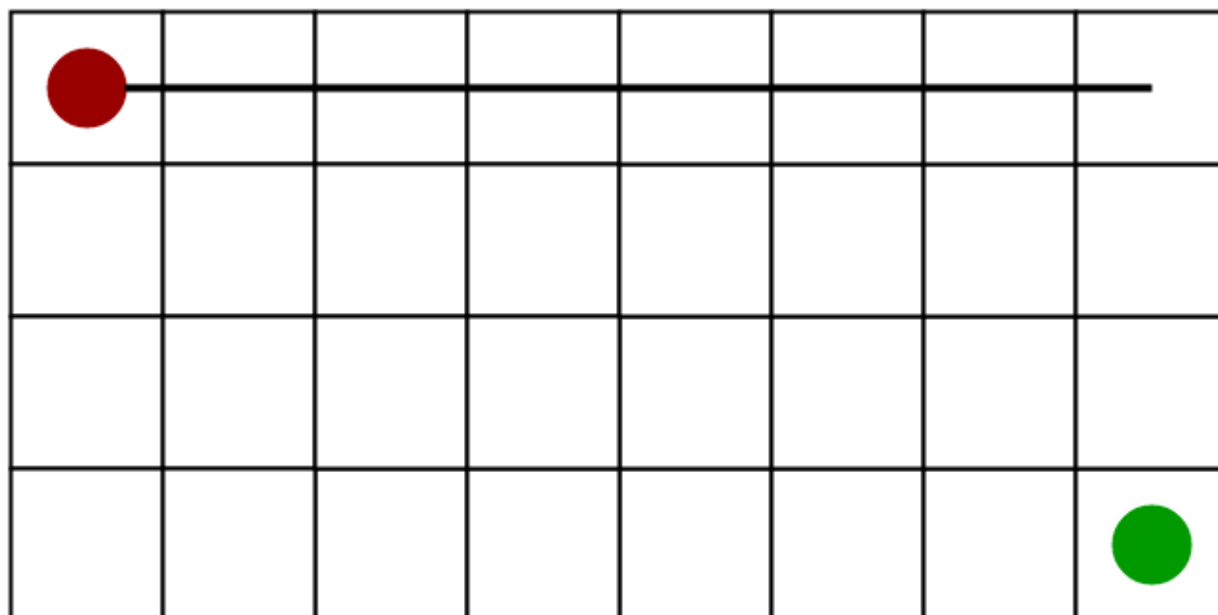
Outra estratégia, conhecida como **Euclidean Distance**, um pouco mais precisa que a anterior, explora o percurso em linha reta, demorando, em contrapartida, mais tempo a executar, por necessitar de explorar uma área maior. A sua função é representada na forma:

$$h = \sqrt{(x_{start} - x_{destination})^2 + (y_{start} - y_{destination})^2}$$



Por fim, existe também a **Diagonal Distance**. Esta última perde interesse real, quando comparada com as outras funções, porque tem a limitação de só poder ser usada em movimentos realizados numa direção apenas:

$$h = \max(abs(x_{start} - x_{destination}), abs(y_{start} - y_{destination}))$$



Este algoritmo em particular é conhecido por não garantir uma solução ótima em muitos casos. No entanto, os seus resultados serão avaliados, a par com os restantes algoritmos, tendo em conta, também, as diferentes funções de heurística aqui referidas.

Ver, para fontes das heurísticas referidas:

<https://brilliant.org/wiki/a-star-search/#heuristics>

<http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html>

<https://www.redblobgames.com/pathfinding/a-star/implementation.html>

<https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2>

<https://www.growingwiththeweb.com/2012/06/a-pathfinding-algorithm.html>

<https://www.geeksforgeeks.org/a-search-algorithm/>

As mais usadas são Manhattan Distance, Euclidean Distance, ou até a Diagonal Distance , mas esta última é limitada

3.1.4 Algoritmo Floyd-Warshall

O último algoritmo analisado será o de Floyd-Warshall. Este atua diretamente no problema da distância mais curta, entre vértices, e baseia-se numa matriz, onde se encontra a menor distância entre cada par de vértices do grafo. O algoritmo retorna, como resultado, uma tabela de distâncias mínimas e, com uma simples modificação, garante, facilmente, informação adicional, como a reconstrução da matriz de predecessores de um determinado caminho, entre dois pontos.

Os custos são mais notórios em termos de memória, mas a compensação em tempo permite que este seja um algoritmo ideal para ser usado no pré-processamento dos grafos. Depois de efetuado o pressuposto, qualquer cálculo de distância se resumirá a uma simples *table lookup*, na matriz de distâncias mínimas, ao contrário dos algoritmos anteriores, que necessitariam de ser executados, de novo, a cada situação. A única inconveniência deste algoritmo poderá ser, nos casos específicos de mudança da constituição atômica dos grafos, como quando se registam arestas intransitáveis, com obras, ou obstáculos na via pública, ter que se pré-processar e reestruturar, de novo, os respectivos mapas, de maneira a identificar outros trajetos que evitem essas arestas.

Apesar disso, este algoritmo será, certamente, o escolhido para ser aplicado na fase final de implementação da aplicação, como descrito de seguida.

3.1.5 Abordagens consideradas na Fase III

Na terceira fase, o caso em que se consideram múltiplos restaurantes no mesmo pedido, poderá ser visto como uma generalização do típico "Travelling Salesman Problem", pois o objetivo é encontrar o menor caminho possível para que cada estafeta visite todos os restaurantes associados a um pedido apenas uma vez. É de notar, no entanto, que o problema original considera que após visitados todos os pontos de interesse é necessário regressar ao ponto de origem, o que não será o caso, pois pretende-se que o destino seja a morada do cliente.

Acrescenta-se, assim, a restrição de existir um ponto de origem e de destino predeterminados, respetivamente a posição inicial do estafeta e a morada de entrega do pedido.

Tendo em conta a dificuldade de conseguir uma solução ótima para este problema, vamos procurar abordá-lo recorrendo a diferentes estratégias, escolhendo a que apresentar o comportamento mais próximo do esperado para cada caso.

A alternativa mais direta seria aplicar um algoritmo "brute-force" que testasse todas os percursos possíveis, preservando, a cada iteração, aquele caminho que apresentasse uma distância mais curta. No entanto, esta solução é impraticável pelo seu elevado tempo de execução, $\Theta(n!)$, até para grafos pouco densos.

Também são conhecidas alternativas de solução para o problema recorrendo a algoritmos de programação dinâmica que, apesar de apresentarem melhorias em comparação com a solução "brute-force", ainda apresentam um tempo exponencial $O(2^n * n^2)$, apresentando, para além disso, complexidade espacial elevada, $O(2^n * n)$, o que nos leva a descartá-lo para o efeito desejado.

A alternativa, que, à partida, nos parece a mais viável tendo em conta os conhecimentos adquiridos até agora, baseia-se numa abordagem gananciosa, já que procura, em cada iteração, escolher a solução ótima, ou seja, neste caso, escolher o restaurante mais perto do anterior.

Assim, tendo como vértice de origem a posição de cada estafeta, procuraria-se construir o percurso até à morada de entrega do pedido, escolhendo, a cada iteração, o restaurante de menor distância.

Para esta abordagem gananciosa será, no entanto, necessário ter calculado previamente as distâncias entre os vários pontos do grafo. Para isso, pensamos recorrer ao algoritmo de programação dinâmica Floyd Warshall, referido anteriormente, obtendo, assim, o caminho mais curto entre todos os pares de vértices, a usar posteriormente para determinar o caminho de cada estafeta para alcançar todos os restaurantes e, por fim, a morada do cliente.

Uma abordagem semelhante mas no sentido inverso poderá ser realizada no caso de os estafetas utilizarem o mesmo meio de transporte, isto é, quando se conhece previamente o grafo de entrada. Nestes casos, ao escolher um estafeta para realizar um pedido, pode tomar-se como ponto de partida a morada de entrega do pedido e, no grafo invertido, construir o percurso a realizar com base na estratégia referida anteriormente: escolhendo, neste caso, o restaurante mais perto da morada do cliente numa primeira iteração e, posteriormente, optando sempre pelo restaurante mais próximo do anterior. Por fim, inclui-se no percurso a localização do estafeta.