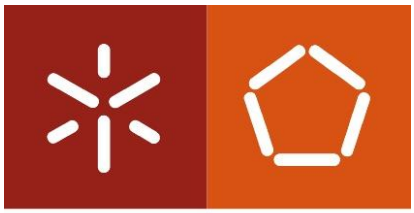

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Inteligência Artificial - Trabalho Prático (Fase Final) - Ano Letivo 2021/2022

Grupo 42

Gonçalo Braz (a93270) Simão Cunha (a93262)
Tiago Silva (a93277) Gonçalo Pereira (a93168)

6 de janeiro de 2022

Conteúdo

1 Introdução

Nesta fase final do trabalho prático da UC de Inteligência Artificial, foi-nos incentivado a utilizar técnicas de formulação de problemas e aplicar diversas estratégias para a resolução dos problemas formulados, com recurso a algoritmos de procura e com o desenvolvimento de mecanismos de raciocínio adequados a esta problemática.

O código foi, de igual modo à primeira parte, escrito recorrendo à linguagem PROLOG.

Para tal, adaptamos o nosso sistema de entregas de encomendas desenvolvido na 1º fase de forma a poder simular os circuitos dos estafetas através de 5 algoritmos de pesquisa: DFS(em profundidade), BFS(em largura), busca iterativa limitada em profundidade, *greedy* e A*.

A forma como fizemos essa adaptação irá ser descrita nas próximas secções.

2 Formulação do problema

No nosso projeto existem quatro funcionalidades: efetuar pesquisas, comparar circuitos de entrega quanto ao volume e peso de encomendas, saber o circuito mais rápido e saber o circuito mais ecológico.

2.1 Representação do estado

O primeiro passo a efetuar nesta fase do projeto consiste em elaborar um grafo para cada zona existente, cujos vértices são as ruas da sua zona associada. Além disso, inserimos um vértice `Centro de distribuições` em cada grafo, uma vez que é dito no enunciado que o estafeta começa e acaba o seu circuito neste ponto.

Pensámos, de início, em elaborar um grafo de dimensões consideráveis com as várias arestas das diversas zonas mas, uma vez que o estafeta só pode efetuar entregas numa zona previamente definida, considerámos não existir necessidade em ter esse grafo e decidimos então simplificar em grafos mais pequenos.

Assim, criamos um dos predicados essenciais para esta fase: o `vertice`, que possui um nome, a zona onde se insere e as coordenadas no grafo.

```
% Exemplo
vertice("Ruilhe", "Rua 1", 3/10).
```

No exemplo acima, temos a representação de um vértice da zona `Ruilhe`, com nome `"Rua 1"` e com coordenadas `x = 3` e `y = 10`.

2.2 Estado inicial

Quando o utilizador pretende efetuar os diferentes algoritmos de pesquisa, decidimos que o estafeta escolhido vai passar por todos os pontos de entrega, começando e terminando o seu circuito no centro de distribuições. Ora, podemos considerar que o estado inicial refere-se ao estafeta possuir todas as encomendas a entregar antes de efetuar a travessia (quer sejam uma ou várias), situando-se no centro de distribuições.

2.3 Estado objetivo

O objetivo do utilizador efetuar os algoritmos de pesquisa é observar que o estafeta escolhido entrega, efetivamente, as suas encomendas associadas. No final do processo, o estafeta deve regressar ao centro de distribuições sem as encomendas que tinha inicialmente.

2.4 Operadores

Para o estafeta mudar de estado, isto é, de mudar de rua em rua, irá utilizar as arestas do grafo. Assim, surgiu o predicado `aresta`. Este possui um nodo de origem, um nodo de destino e um custo associado a essa aresta. Além disso, possui uma zona, de forma a impedir ambiguidades entre ruas com nomes iguais de zonas diferentes.

```
% Exemplo
aresta("Ruilhe", "Centro de distribuições", "Rua 3", 7).
```

Aqui, temos, a título de exemplo, a representação de uma aresta da zona `Ruilhe`, com origem `Centro de distribuições` e destino `Rua 3`, com custo associado `7`.

2.5 Custo da solução

Na nossa aplicação efetuamos 2 tipos de pesquisas: não informada e informada. Nos algoritmos de pesquisa informada, não é considerado qualquer custo, uma vez que o intuito deste tipo é encontrar uma solução qualquer. No entanto, para podermos comparar os diferentes circuitos gerados pelos diferentes algoritmos, poderemos considerar como o número de nós visitados até chegar ao nodo destino. Já na pesquisa informada, temos 2 tipos de custo (também chamados de heurísticas): tempo e distância. Para a comparação entre circuitos quanto ao tempo temos em atenção a velocidade do estafeta e a distância em linha reta entre dois nós. Já para a comparação entre circuitos quanto à distância, temos apenas em atenção à distância em linha reta das coordenadas de 2 nós.

3 Estratégias de procura

3.1 Procura não informada

Este tipo de estratégia pode-se dizer que apenas procura uma solução para o problema de procura, não tomando em conta quaisquer tipos de custo na procura de caminhos solução visto que, o algoritmo é executado às "cegas". Assim, esta pesquisa não é boa para encontrar a solução ótima, mas sim para encontrar uma solução.

Para este tipo de procura adaptámos os algoritmos: Depth-First Search, Breadth-First Search e Busca Iterativa Limitada em Profundidade.

3.1.1 Profundidade (DFS - Depth-First Search)

A primeira estratégia que adaptámos expande sempre um nó para passar ao próximo nível da árvore de procura, sendo portanto menos custoso em termos de memória, em problemas onde existem múltiplas soluções, caso contrário, este pode ficar perdido num ramo em que não existe solução durante muito tempo antes de encontrar o ramo certo.

Além disso no caso de uma árvore ter profundidade infinita, o algoritmo falha.

3.1.2 Largura (BFS - Breadth-First Search)

Neste tipo de pesquisa o algoritmo expande primeiro todos os nós de um nível da árvore de procura antes de passar para o próximo nível, deste modo, garantindo que percorrerá todo o grafo de modo a encontrar uma solução porém, do mesmo modo poderá demorar muito tempo a encontrar a solução e, como expande todos os nodos de cada nível, ocupará muito espaço.

3.1.3 Busca Iterativa Limitada em Profundidade

A busca iterativa procura limitar o uso de memória do DFS nos casos em que este se perde por ramos profundos sem solução.

Este algoritmo começa por limitar o dfs a encontrar uma solução de tamanho i e, caso isto resulte em falso, é realizada uma nova iteração, desta vez limitando o tamanho da solução a i+1, e assim recursivamente até que uma solução seja encontrada.

3.2 Procura informada

A procura informada beneficia do conhecimento dos custos das operações que permitem a resolução do problema, por exemplo, a distância de uma aresta ou o tempo que demora a percorrê-la são informações que esta pesquisa pode utilizar para procurar a melhor solução ou pelo menos uma solução boa para minimizar estes custos.

3.2.1 Gulosa (*Greedy*)

A gulosa na sua procura pela solução, em cada iteração, busca pelo caminho que fornece a melhor estimativa de custo para alcançar o estado objetivo do problema.

Esta estimativa, na nossa solução é calculada através do cálculo da distância até ao ponto objetivo, no caso de ser procurada uma boa solução de distância. No caso de se procurar um solução em termos de tempo, a estimativa é calculada através da divisão da distância pela velocidade do estafeta em questão de modo a devolver o tempo estimado até chegar ao objetivo.

3.2.2 A*

Por último, adaptámos o algoritmo de procura A*, este que, em relação aos seus predecessores, encontra a melhor solução.

De igual forma que o a gulosa, ele utiliza as estimativas de modo a encontrar o melhor caminho mas, toma uma precaução para o caso de estas não serem fiáveis, analisando também os próprios custos.

A nossa implementação aproveita dos predicados de cálculo de estimativas utilizados pela gulosa.

4 Resultados

Nesta secção, iremos mostrar exemplos concretos da aplicação dos algoritmos em situações em que o estafeta entrega uma só encomenda, regressando ao centro de distribuições, e no caso em que o estafeta tem de passar por vários pontos da zona para entregar as encomendas. Decidimos exemplificar com recurso ao grafo de `Semelhe`, uma vez que é o grafo mais complexo existente no nosso sistema.

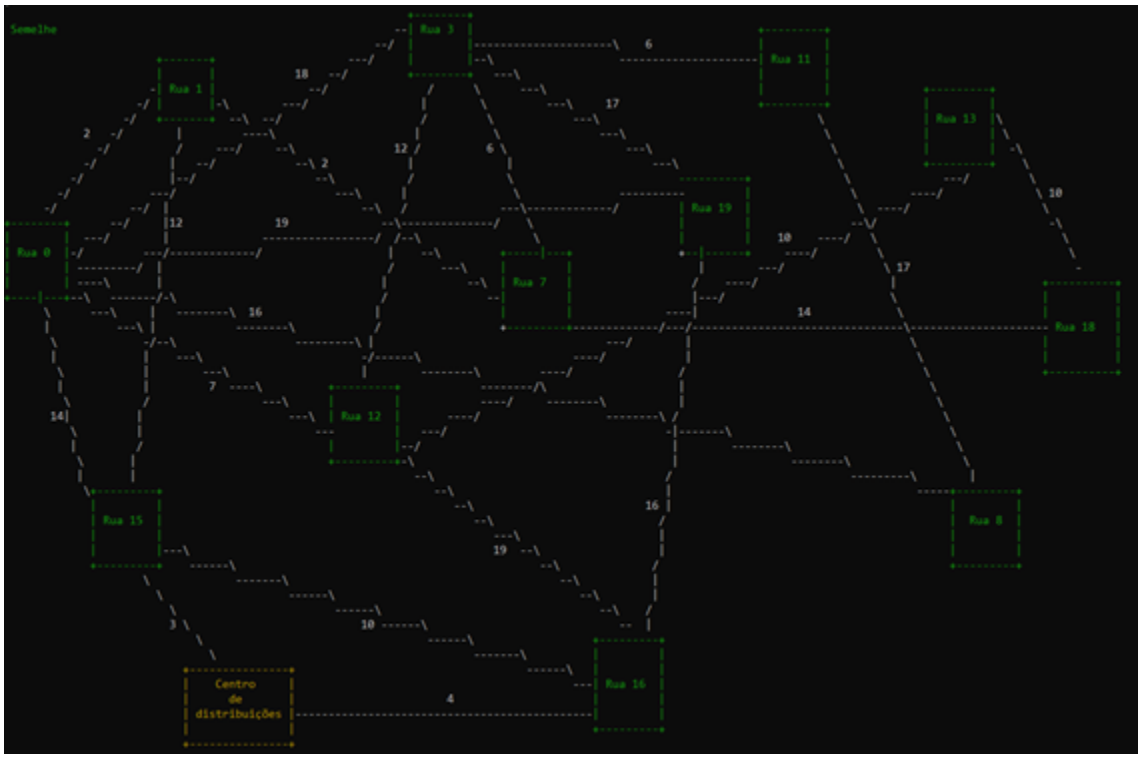

Figura 1: Representação em grafo da freguesia de `Semelhe`

4.1 Entregar uma encomenda

Algoritmo	Tempo de execução (ms)	Espaço	Indicador/ Custo	Encontrou a melhor solução?	Vértice destino	Estafeta
DFS	0.4561	$O(b^d)$	Nº de nós visitados	Não	Rua 19	Tina
BFS	2.9538	O(bm)	Nº de nós visitados	Não	Rua 19	Tina
BILP	17.2288	O(bd)	Nº de nós visitados	Não	Rua 19	Tina
Gulosa						
➢ Distância	0.5319	Melhor caso: $O(b^d)$ Pior caso: O(bd)	Distância	Não	Rua 19	Tina
➢ Tempo	0.2310	Melhor caso: $O(b^d)$ Pior caso: O(bd)	Tempo	Não	Rua 19	Tina
A*						
➢ Distância	0.4852	Todos os nós em memória	Distância	Sim	Rua 19	Tina
➢ Tempo	0.6223	Todos os nós em memória	Tempo	Sim	Rua 19	Tina

Legenda:
b: o máximo fator de ramificação (o número máximo de sucessores de um nó) da árvore de pesquisa
d: a profundidade da melhor solução
m: a máxima profundidade do espaço de estados

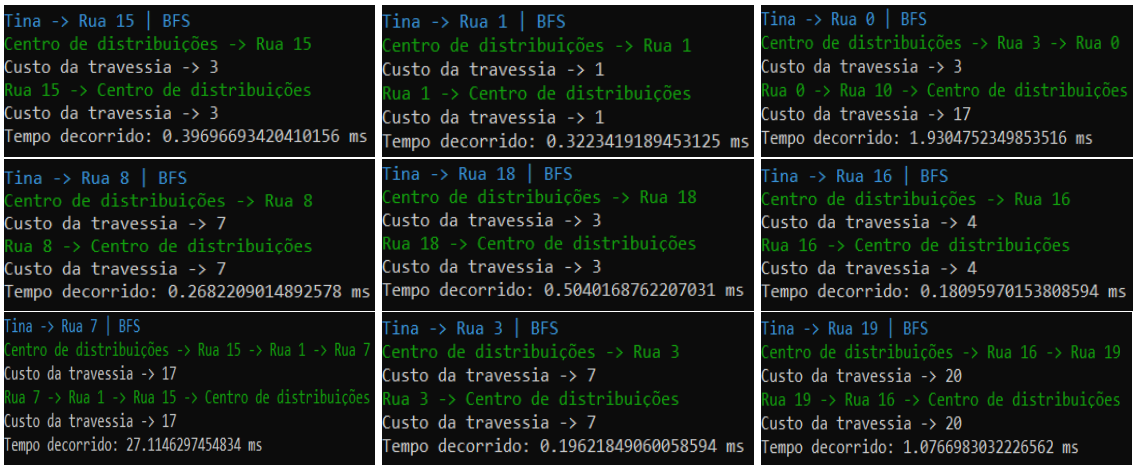

Figura 2: Circuitos do estafeta até todos os vértices do grafo (**BFS**)

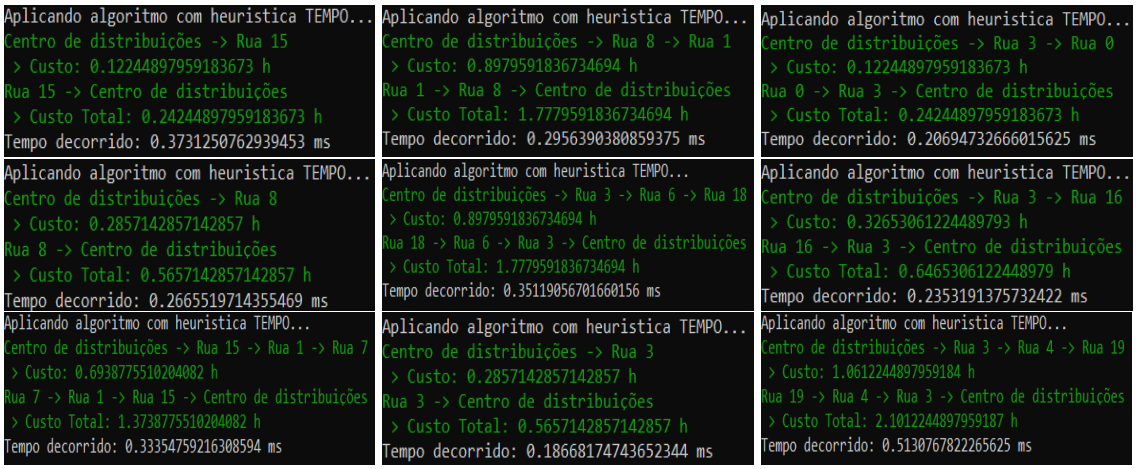

Figura 3: Circuitos do estafeta até todos os vértices do grafo (**A*** com heurística de tempo)

4.2 Entregar várias encomendas

Conforme sugerido no enunciado, testámos que influencia teria tanto no programa, como nas próprias soluções dos algoritmos, colocar um estafeta a realizar múltiplas entregas em uma só viagem.

Para tal, tivemos de novamente adaptar os algoritmos de forma a que tal função fosse obtível.

Começámos então por, para um dado estafeta, obter todos os seus pontos de entrega, excluindo entregas no mesmo local, sendo que contamos que ele as realiza todas ao mesmo tempo caso ele lá chegue. No caso dos três primeiros algoritmos, esta informação é suficiente para obter os resultados esperados, sendo que cada iteração implica usar como origem o primeiro local da lista de pontos que está a ser iterada, e o destino, o segundo da lista. Ao fim de cada iteração, é observado o caminho obtido como solução e, retirados os locais de entrega da lista, caso eles estejam presentes no caminho, considerando então que ele realizou a entrega ao passar por eles.

O estafeta começa esta viagem no centro de distribuições e, do mesmo modo, após realizar todas as entregas, retorna ao mesmo.

No caso do algoritmo greedy e A*, como estes podem ser definidos de modo a procurarem a melhor solução em termos de tempo e, tendo em conta que este depende da velocidade do estafeta e que, por sua vez esta depende do peso que o estafeta carrega no momento, teremos de, em cada iteração, recalcular a quantidade de peso que ele ainda leva e, consequentemente a nova velocidade dele, de modo a calcularmos custos e estimativas realistas para o algoritmo usar. Para isto, precisamos de, ao mesmo tempo que calculamos os pontos em que o estafeta necessita passar, a quantidade de peso correspondente aos pedidos desses pontos, e, em cada iteração utilizamos esta lista para atualizar os valores anteriormente mencionados.

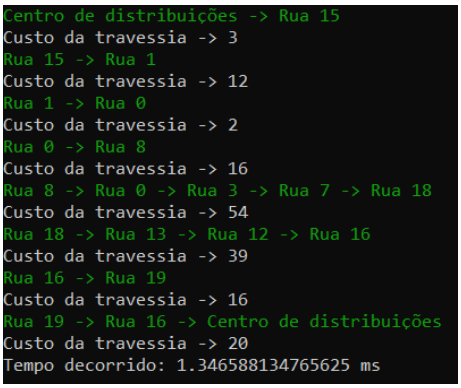

Figura 4: Estafeta efetua BFS

Comparando a utilização do algoritmo BFS para a entrega de todos os pedidos em simultâneo com a utilização repetida para entregar todos os pedidos separadamente, podemos observar que a primeira encontra um caminho-solução com tamanho muito mais eficiente, isto é, o tamanho do primeiro caso é 15 e no segundo caso o tamanho é 35.

Figura 5: Estafeta efetua Gulosa com heurística de distância e de tempo

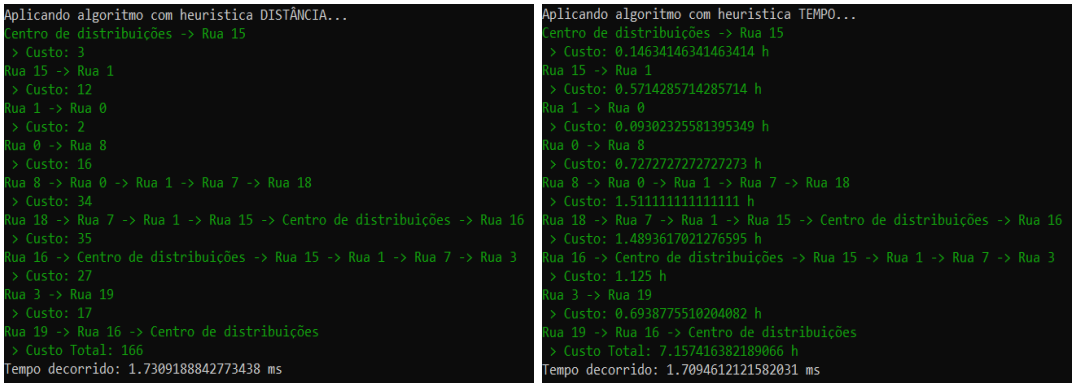

Figura 6: Estafeta efetua A* com heurística de distância e de tempo

Comparando a utilização do algoritmo A* com heurística relativa ao tempo para a entrega de todos os pedidos em simultâneo com a utilização repetida para entregar todos os pedidos

separadamente, podemos observar que a primeira possui um menor custo-tempo em relação à segunda, isto é, o tempo do primeiro caso é 7.15 horas e no segundo caso o tempo é 9.25 horas.

5 Comentários Finais e Conclusão

Através da realização deste trabalho, que visou o aprofundamento do estudo da resolução de problemas através da aplicação de algoritmos de procura, apurámos a nossa compreensão sobre a implementação destes e as diferentes vantagens e desvantagens entre os diferentes métodos de pesquisas.

Além disso, tendo em conta a problemática em questão, a resolução de caminhos através de grafos predefinidos, aumentámos o nosso conhecimento sobre estes ambientes, por exemplo, a obtenção de estatísticas provenientes dos resultados das procuras ou a otimização de travessias com vários destinos, de modo a não repetir destinos que já foram previamente visitados.

Consideramos portanto, termos concluído os objetivos requeridos do trabalho com sucesso, incluindo alguns dos objetivos adicionais, através de uma implementação que seguiu os conhecimentos de programação lógica.