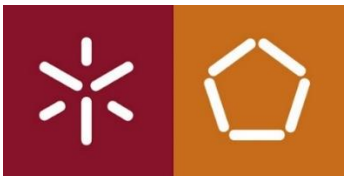

Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

INTELIGÊNCIA ARTIFICIAL

2021/2022

MÉTODOS DE RESOLUÇÃO DE PROBLEMAS E DE PROCURA

Serviço de Encomendas - Green Distribution

GRUPO 22:

Joana Alves A93290
Maria Cunha A93264
Samuel Lira A94166
Vicente Moreira A93296

Data de Entrega: 05/01/2022

Índice

Índice de Figuras

Introdução

Esta segunda fase do trabalho consistiu no desenvolvimento de um sistema de recomendação de circuitos de entrega de encomendas. Este executa simulações de entregas, de uma forma razoavelmente realista.

Para o seu desenvolvimento foi necessário expandir a base de conhecimento elaborada na primeira fase do trabalho acrescentando alguns predicados, não tendo alterado os pré-existentes. Como tal, implementamos predicados necessários ao teste e cálculo de circuitos de entrega como algoritmos de pesquisa informada e não-informada. Para além disto, foram também implementados predicados necessários para a representação da ligação das ruas na forma de um grafo não orientado.

Assim, reuniram-se os conhecimentos adquiridos sobre pesquisas informadas e não-informadas e procedeu-se à conceção de uma estratégia para o problema proposto.

Formulação Do Problema

Figura 1 - Grafo contendo as várias ruas do sistema.

Identificamos este problema como um problema de **Estado Único**, tendo em vista que o agente (um estafeta) se encontra num **ambiente determinístico** e totalmente **observável**. Certas variantes, como alterações de morada durante entrega, desvios de rota, entre outras, não foram consideradas neste caso de estudo.

O **Estado Inicial** coincide com o **Estado Final** estabelecido: o centro de entregas *GreenDistribution*, sendo este localizado na 'Rua da Universidade', estabelecendo-se esta como o nó inicial do grafo concebido.

O **Estado Objetivo** estabelecido é o ponto de entrega, neste caso, os vários nós do grafo acima representado (exceto o nó correspondente ao estado inicial e final).

Ações possíveis: conduzir entre as ruas.

Este grafo (Figura 1), representa as várias ruas que farão parte da simulação de entregas, estando limitadas à cidade de Braga, não sendo importante a freguesia a que pertencem.

O grafo desenvolvido denota a relação de distância (em km) entre cada rua/nodo, sendo estes valores reais uma vez que recorremos à ferramenta *Google Maps* para o cálculo dos mesmos.

Recorremos à mesma ferramenta para obter uma estimativa de tempo médio que levaria a chegar de qualquer um dos nós do grafo até ao nó 'Largo de São Pedro', obtendo os resultados presentes na Figura2. Estas estimativas de tempo serão usadas na procura de solução utilizando os algoritmos de procura informada, mais concretamente no algoritmo A*.

Tempos de viagem médios (Estimativa)	
Local	Tempo (minutos)
Rua da Universidade	13
Avenida Dom João II	9
Rua São Vicente	12
Avenida Padre Julio F.	7
Rua da Lage	12
Rua da Igreja	9
Rua Conselheiro Lobato	5
Rua Conselheiro Januário	10
Rua da Mouta	1
Rua António Ferreira Rito	7
Avenida da Liberdade	7
Campo das Hortas	8
Rua da Boavista	10
Largo São Pedro	0

Figura 2 - Estimativas de Tempo

Estratégias De Procura

Uma estratégia de pesquisa é definida pela escolha da ordem de expansão dos nós de uma árvore de pesquisa. Estratégias de pesquisas não informadas não possuem conhecimento extra ao da definição do problema. Estas vão conhecendo o ambiente ao passo que pesquisam. Em contraste, as estratégias de pesquisa informadas possuem algum conhecimento extra sobre a adequação de diferentes estados. Assim, procedemos a uma breve explicação das várias estratégias consideradas para o nosso projeto, inserindo, para cada, uma imagem representativa do comportamento do algoritmo de expansão de nós da mesma (retirada de um documento cedido pelos docentes, referido na Bibliografia).

Procura Não-Informada

Profundidade Primeiro (DFS – *Depth First Search*)

Esta estratégia dá preferência à expansão dos nós mais profundos da árvore, recorrendo a pouca memória, sendo indicada para problemas com várias soluções. No entanto, deve ser evitada quando as árvores geradas são muito profundas ou geram caminhos infinitos.

- **Completude**: Não, falha em espaços de profundidade infinita, ou também com repetições.
- **Tempo**: $O(b^m)$, piora caso m>d, sendo **b** o fator de ramificação e **d** a profundidade da melhor solução e **m** a máxima profundidade do espaço de estados.
- **Espaço**: O(bm), linear.
- **Otimalidade**: Não, devolve a primeira solução encontrada.

Para uma melhor perceção do funcionamento do algoritmo, apresentamos uma sequência de imagens que ilustra o modo de escolha do algoritmo dos nodos a expandir. Como referido acima, as imagens foram retiradas de um conjunto de *slides* fornecidos pelos docentes.

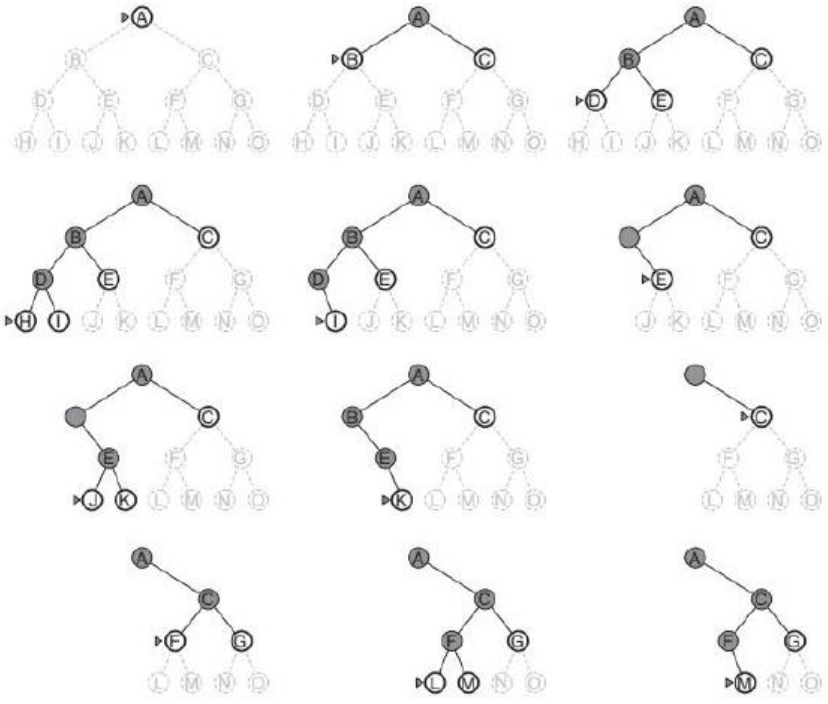

Figura 3 - Profundidade Primeiro (DFS)

Largura Primeiro (BFS – *Breadth First Search*)

A estratégia de pesquisa em largura define-se por expandir os nós de menor profundidade primeiro, sendo benéfico apenas em problemas de pequena dimensão, pois demora muito tempo e ocupa muito espaço.

- **Completude**: Sim, se o fator de ramificação (nºmáximo de sucessores de um nó) for finito.
- **Tempo:** $O(b^d)$.
- **Espaço**: Guarda cada nó na memória, $O(b^d)$.
- **Otimalidade**: Sim, se o custo de cada passo for igual a 1.

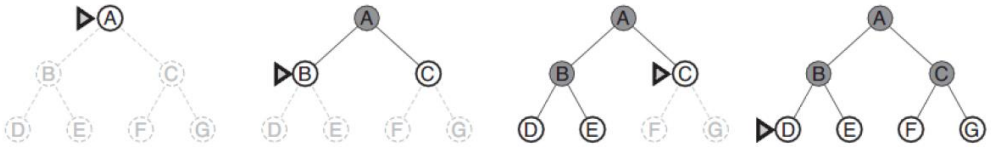

Figura 4 - Largura Primeiro (BFS)

Busca Iterativa Limitada em Profundidade

Executa-se uma pesquisa em profundidade limitada, iterativamente, aumentando sempre o limite da profundidade, no geral melhor estratégia (não-informada ou cega) para problemas com um grande espaço de pesquisa e em que a profundidade da solução não é conhecida. Não implementada no código.

- **Completude**: Sim.
- **Tempo**: $O(b^d)$.
- **Espaço**: $O(b^d)$.
- **Otimalidade**: Sim, se o custo for 1.

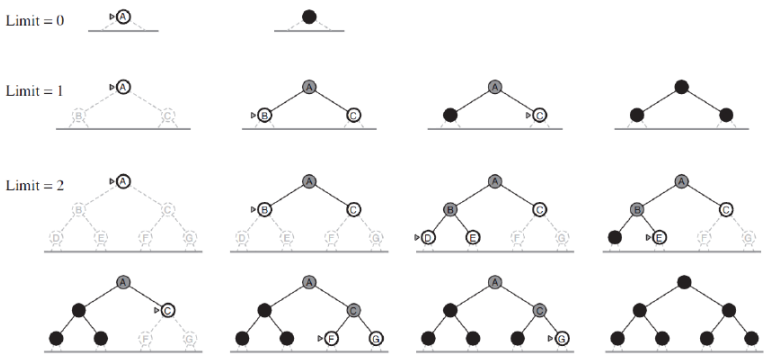

Figura 5 - Busca Iterativa Limitada em Profundidade

Procura Informada

Pesquisa Gulosa (*Greedy-Search*)

Expande o nó que parecer estar mais próximo da solução, através de uma função heurística.

- **Completude**: Não, pode entrar em ciclos.
- **Tempo**: O(b^m), pode reduzir significativamente com uma boa função heurística.
- **Espaço**: Guarda cada nó na memória, O(b^m).
- **Otimalidade**: Não.

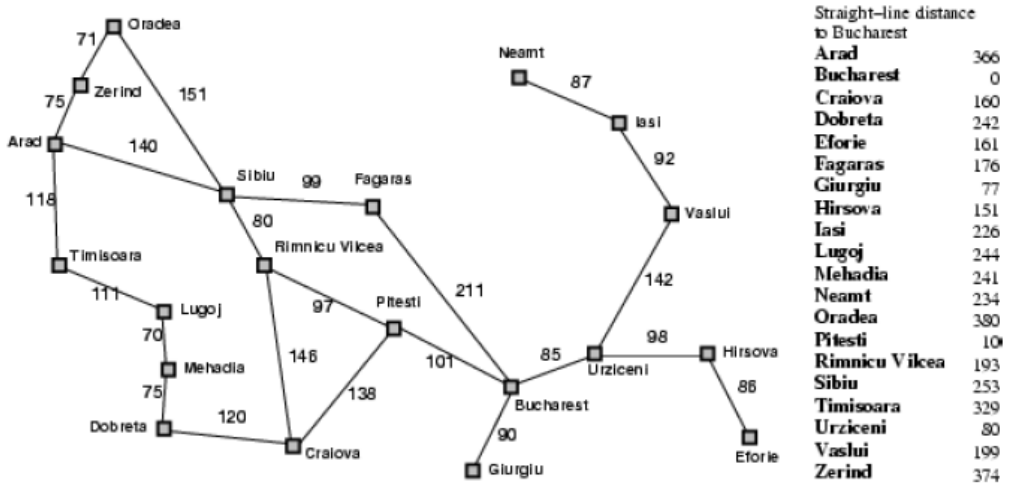

Figura 6 - Grafo Exemplificativo

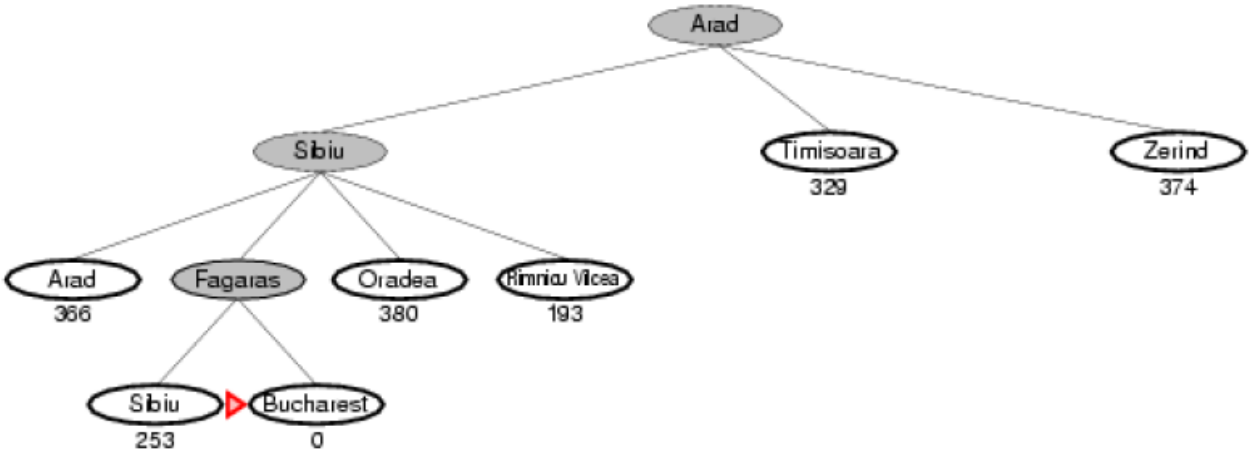

Figura 7 - Árvore de Pesquisa do algoritmo Gulosa através do grafo da Figura 6

Algoritmo A*

Evita expandir nós de custo alto, combinando a pesquisa gulosa com a uniforme com o objetivo de minimizar a soma do caminho já efetuado com o mínimo previsto que falta até a solução.

- **Completude**: Sim.
- **Tempo**: Erro relativo de h*comprimento da solução.
- **Espaço**: Guarda cada nó na memória, O(b^m).
- **Otimalidade**: Sim.

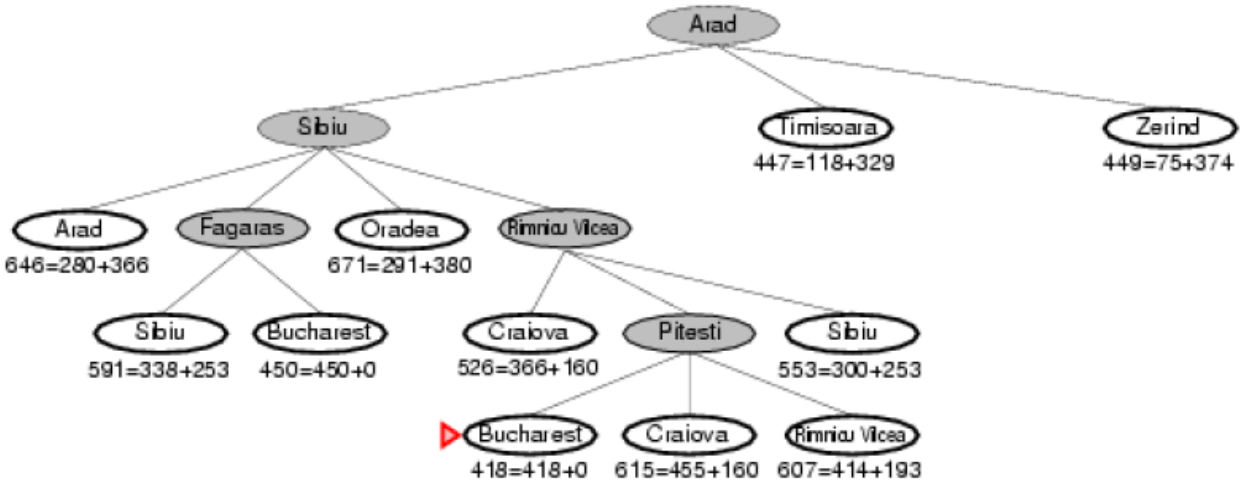

Figura 8 - Árvore de Pesquisa do Algoritmo A através do grafo da Figura 6*

Resultados dos Algoritmos de Pesquisa

Para a realização de testes de *performance* da execução dos vários algoritmos de pesquisa informada e não-informada, realizamos cinco iterações de execução para cada um, tendo todos em comum o nodo objetivo: 'Largo de São Pedro'. A solução ótima em termos de distância seria: Rua da Universidade -> Avenida Dom João II -> Rua da Igreja -> Rua da Mouta -> Largo de São Pedro, com um custo total de 7,65 km.

Para a demonstração da veracidade dos resultados obtidos, apresentamos de seguida *prints* da resolução de cada um dos algoritmos implementados através da ferramenta *SWI-Prolog*.

Pesquisa Não-Informada

Para a execução dos algoritmos **Largura Primeiro** e **Profundidade Primeiro**, era necessário passar como argumento um ID de entrega (desenvolvido na primeira fase do trabalho). A partir deste, o ponto de entrega, o peso da encomenda e o tempo limite imposto pelo cliente são obtidos, começando a análise do problema.

```
?- calculaCaminhoBFS(13).

Peso da Encomenda: 14 Kg.
Tempo de Entrega Max: 51 minutos.
Ponto de Entrega: largoSaoPedro.

IDVeiculo: 5 (carro)
VelocidadeMedia: 23.6 km/h.

Caminhos:
   Ida: [ruaDaUniversidade,ruaDaLage,ruaDaBoavista,largoSaoPedro].
   Volta: [largoSaoPedro,ruaDaBoavista,ruaDaLage,ruaDaUniversidade]

Custo Ida: 14.2 km.
Custo Total: 28.4 km.

Tempo do Cliente: 51 minutos.
Tempo Real de Entrega: 36.101694915254235 minutos.
true .
```

Figura 10 - Resultado do Algoritmo BFS

```
?- calculaCaminhoDFS(13).

Peso da Encomenda: 14 Kg.
Tempo de Entrega Max: 51 minutos.
Ponto de Entrega: largoSaoPedro.

IDVeiculo: 5 (carro)
VelocidadeMedia: 23.6 km/h.

Caminhos:
   Ida: [ruaDaUniversidade,ruaDaLage,ruaDaBoavista,largoSaoPedro].
   Volta: [largoSaoPedro,ruaDaMouta,ruaDaIgreja,avenidaDomJoaoII,ruaDaUniversidade]

Custo Ida: 14.2 km.
Custo Total: 21.85 km.

Tempo do Cliente: 51 minutos.
Tempo Real de Entrega: 36.101694915254235 minutos.
true .
```

Figura 9 - Resultado do Algoritmo DFS

Pesquisa Informada

Para a execução dos algoritmos **Gulosa** e **A***, o nodo objetivo foi pré-definido devido à necessidade de conter informação extra para este, usando para isso um ID de entrega fixo. No entanto, para o propósito do Estado Objetivo comum entre os vários algoritmos (para assim obtermos resultados relativos ao mesmo processo), alteramos o ID de entrega para coincidir com o usado nos algoritmos de pesquisa não-informada, ou seja, o ID de entrega nº 13.

```
|    calculaCaminhoEstrela().

ID da Entrega: 13.
Peso da Encomenda: 14 Kg.
Tempo de Entrega Max: 51 minutos.
Ponto de Entrega: largoSaoPedro.

IDVeiculo: 5 (carro)
VelocidadeMedia: 23.6 km/h.

Caminhos:
   Ida: [ruaDaUniversidade,avenidaPadreJulioFragata,ruaConselheiroJanuario,avenidaDaLiberdade,largoSaoPedro].
   Volta: [largoSaoPedro,ruaAntonioFerreiraRito,ruaConselheiroLobato,avenidaDomJoaoII,ruaDaUniversidade]

Custo Ida: 9.3 km.
Custo Total: 21.700000000000003 km.

Tempo do Cliente: 51 minutos.
Tempo Real de Entrega: 23.64406779661017 minutos.
true .
```

*Figura 11 - Resultado do Algoritmo A**

```
?- calculaCaminhoGulosa().

ID da Entrega: 13.

Peso da Encomenda: 14 Kg.
Tempo de Entrega Max: 51 minutos.
Ponto de Entrega: largoSaoPedro.

IDVeiculo: 5 (carro)
VelocidadeMedia: 23.6 km/h.

Caminhos:
   Ida: [ruaDaUniversidade,avenidaDomJoaoII,ruaDaIgreja,ruaDaMouta,largoSaoPedro].
   Volta: [largoSaoPedro,ruaDaMouta,ruaConselheiroLobato,avenidaDomJoaoII,ruaDaUniversidade]

Custo Ida: 7.65 km.
Custo Total: 16.0 km.

Tempo do Cliente: 51 minutos.
Tempo Real de Entrega: 19.44915254237288 minutos.
true .
```

Figura 12 - Resultado do Algoritmo Gulosa (Solução Ótima)

Tabela de Resultados:

Estratégia	Tempo Médio (ms)	Espaço	Indicador/Custo	Encontrou a melhor solução?
DPS	10	O(5*4)	Distância	NÃO
BFS	9	$O(5^4)$	Distância	NÃO
Estrela	10,4	$O(5^4)$	Distância + Tempo	SIM
Gulosa	8,4	$O(5^4)$	Distância	SIM

ESPECIFICAÇÃO DA MÁQUINA:

Sistema Operativo: Windows 10 Home
Memoria RAM: 16GB
Processador: Intel Core i7-8550U @ 1.80Ghz

Comentários dos Resultados Obtidos

Deparados com os resultados, concluímos que $\boldsymbol{\Delta t_{Gulosa} < \Delta t_{BFS} < \Delta t_{DFS} < \Delta t_{A*}}$. Notamos que não houve mudanças significativas observáveis entre o tempo de execução dos vários algoritmos devido à baixa complexidade do grafo desenvolvido. No entanto, é de notar que os tempos médios foram obtidos numa só máquina com várias aplicações a correr em *background*, influenciando a *performance* da execução dos algoritmos.

Relativamente à solução ótima de **distância**, apenas o algoritmo de pesquisa informada **Gulosa** a encontrou. Os algoritmos de pesquisa não-informada (**BFS** e **DFS**) apenas devolveram a primeira solução calculada, tendo sido, em panorama geral, a pior solução, ou seja, a mais longa das quatro soluções obtidas, com um total de 14,2 km.

O algoritmo **A***, tal como a sua informação teórica sustentava, encontrou a melhor solução para minimização da soma das distâncias com os tempos.

Estatísticas Dos Circuitos

Outro requisito desta segunda fase de desenvolvimento era a implementação de predicados que calculassem o circuito com o maior número de entregas (peso e volume). Assim, tivemos a necessidade de expandir a base de conhecimento, adicionando os predicados 'circuito' e 'circuitoEntrega'. O predicado 'circuito' corresponde à associação de um identificador a um circuito possível do sistema. Já o predicado 'circuitoEntrega', associa um ID de entrega (desenvolvidos na primeira fase do trabalho) a um ID de circuito.

De seguida, foram implementados três predicados distintos, mas com um comportamento semelhante, ou seja, que comparam os circuitos apenas diferindo no critério de comparação (número de entregas, peso ou volume).

Circuito Com Maior Número de Entregas

Para este predicado, a estratégia seguida é comum aos três predicados referidos acima. Começamos por verificar quantos circuitos existem no sistema (quantosCircuitos), para, de seguida, compararmos cada um deles de forma sequencial (maiorNumEntregasAux), obtendo, no final, um par (IDCircuito, Nº Entregas) correspondente ao identificador do circuito com maior número de entregas associadas e o número de entregas respetivo.

```
% maiorNumEntregasCircuito: (Id,Quantos) -> {V,F}
% descobre qual o circuito com o maior mumero de entregas associado.
maiorNumEntregasCircuito(Res) :-
    quantosCircuitos(Max),
    maiorNumEntregasAux(Max,0,Pares),
    maxPar(Pares,Res).

maiorNumEntregasAux(Max,Max,[]).
maiorNumEntregasAux(Max, Act, [(ID,Tamanho)|T]) :-
    ID is Act+1,
    findall((IDEntrega), circuitoEntrega(IDEntrega,ID), IDs),
    remove_duplicates(IDs,Res),
    length(Res, Tamanho),
    Next is Act+1,
    maiorNumEntregasAux(Max, Next, T).
```

Figura 13 - Código de implementação do predicado 'maiorNumEntregasCircuito'

```
?- mainMaiorNumEntregas().
Circuito ID:  6.
Caminho: [ruaDaUniversidade,avenidaDomJoaoII,ruaConselheiroLobato].
Numero de Entregas: 6.
true .
```

Figura 14 - Resultado do cálculo 'maiorNumEntregasCircuito'

Circuito Com Maior Peso/Volume de Entregas

Para estes predicados, começamos também por verificar o número de circuitos presentes na base de conhecimento, recorrendo ao mesmo predicado auxiliar referido acima. De seguida, é calculado para cada circuito as entregas associadas, e, para cada entrega, o peso/volume associado à mesma. Termina retornando o par com (IDCircuito, Total), que corresponde ao identificador do circuito com maior peso/volume de entregas e o peso/volume total do mesmo.

```
% maiorPesoCircuito: (Id,Peso) -> {V,F}
% calcula o circuito com as encomendas mais pesadas.
maiorPesoCircuito(Res) :-
    quantosCircuitos(Max),
    maiorPesoAux(Max,0,Pares),
    maxPar(Pares,Res).

maiorPesoAux(Max,Max,[]).
maiorPesoAux(Max, Act, [(ID,Soma)|T]) :-
    ID is Act+1,
    findall((IDEntrega), circuitoEntrega(IDEntrega,ID), IDs),
    remove_duplicates(IDs,Res),

    maplist(getPeso(),Res,ListaPesos),
    sumlist(ListaPesos,Soma),

    Next is Act+1,
    maiorPesoAux(Max, Next, T).
```

Figura 15 - Códido de implementação do predicado 'maiorPesoCircuito'

```
% maiorVolumeCircuito: (Id,Peso) -> {V,F}
% calcula o circuito com as encomendas mais volumosas.
maiorVolumeCircuito(Res) :-
    quantosCircuitos(Max),
    maiorVolumeAux(Max,0,Pares),
    maxPar(Pares,Res).

maiorVolumeAux(Max,Max,[]).
maiorVolumeAux(Max, Act, [(ID,Soma)|T]) :-
    ID is Act+1,
    findall((IDEntrega), circuitoEntrega(IDEntrega,ID), IDs),
    remove_duplicates(IDs,Res),

    maplist(getVolume(),Res,ListaVolumes),
    sumlist(ListaVolumes,Soma),

    Next is Act+1,
    maiorVolumeAux(Max, Next, T).
```

Figura 16 - Códido de implementação do predicado 'maiorVolumeCircuito'

```
?- mainMaiorPesoEntregas().
Circuito ID:  9.
Caminho: [ruaDaUniversidade,avenidaPadreJulioFragata,ruaConselheiroJanuario].
Peso Total das Entregas: 92 Kg.
true .

?- mainMaiorVolumeEntregas().
Circuito ID:  17.
Caminho: [ruaDaUniversidade,ruaSaoVicente,ruaConselheiroLobato,ruaAntonioFerreiraRito,largoSaoPedro].
Volume Total das Entregas: 92 dm3.
true .
```

Figura 17 - Resultado do cálculo 'maiorPeso/VolumeCircuito'

Comparar Distância/Tempo Circuitos

Após a implementação dos predicados acima, prosseguimos para o próximo ponto: comparar circuitos relativamente ao seu comprimento total ou tempo médio para percorrer o mesmo. Assim, implementamos dois predicados muito semelhantes entre si, sendo a única diferença o critério de comparação (distância ou tempo). A restrição destes predicados envolve a comparação entre apenas dois circuitos, tendo estes de ser passados como argumentos assim como o seu ID.

```
% circuitoMaisCurto: Circuito1, Circuito2, CircuitoMaisCurto -> {V,F}
% Dado dois circuitos, calcula o mais curto dos dois.
circuitoMaisCurto((ID1,Circuito1,Primeiro), (ID2,Circuito2,Segundo), (ID,Circuito)) :-
    calculaCusto(Circuito1, Primeiro),
    calculaCusto(Circuito2, Segundo),

    minPar([(ID1,Primeiro),(ID2,Segundo)], (ID,Circuito)).
```

Figura 18 - Código de implementação do predicado 'circuitoMaisCurto'

```
% circuitoMaisRápido: Circuito1, Circuito2, CircuitoMaisRapido -> {V,F}
% Dado dois circuitos, calcula o mais rápido dos dois.
circuitoMaisRapido((ID1,Circuito1,Tempo1), (ID2,Circuito2,Tempo2), (ID,Circuito)) :-
    calculaTempo(Circuito1, Tempo1),
    calculaTempo(Circuito2, Tempo2),

    minPar([(ID1,Tempo1),(ID2,Tempo2)],(ID,Circuito)).
```

Figura 19 - Código de implementação do predicado 'circuitoMaisRapido'

```
?- mainComparaDistanciaCircuitos(1,14).
Caminho 1: [ruaDaUniversidade,ruaSaoVicente].
Distancia: 3.2 km.

Caminho 14: [ruaDaUniversidade,avenidaPadreJulioFragata,ruaConselheiroJanuario,campoDasHortas].
Distancia: 7.1 km.

Caminho Mais Curto: 1 (3.2 km).
true .

?- mainComparaTempoCircuitos(1,14).
Caminho 1: [ruaDaUniversidade,ruaSaoVicente].
Tempo: 12 minutos.

Caminho 14: [ruaDaUniversidade,avenidaPadreJulioFragata,ruaConselheiroJanuario,campoDasHortas].
Tempo: 25 minutos.

Caminho Mais Rapido: 1 (12 minutos).
true .
```

Figura 20 - Resultado do cálculo 'circuitoMaisCurto/Rapido'

Conclusão

Com a realização deste trabalho reconhecemos a necessidade e a importância da representação de conhecimento num contexto informático e da utilidade que uma ferramenta capaz de realizar inferências poderá facilitar no desenvolvimento de novo conhecimento.

Também aprofundamos o nosso conhecimento teórico e prático acerca dos variados níveis de complexidade das heurísticas lecionadas, assim como as suas propriedades, vantagens e desvantagens na sua utilização entre os vários cenários possíveis.

Com a conclusão do desenvolvimento deste projeto, acreditamos que alcançamos de forma satisfatória as várias metas propostas pelos docentes, tendo adquirido resultados e estatísticas credíveis no universo que definimos.

No entanto, encontramos alguma dificuldade na adaptação inicial no uso da ferramenta *SWI-PROLOG* para a realização do projeto. Para além disso, achamos que o grafo desenvolvido não foi suficientemente complexo de forma a observar diferenças significativas no uso dos vários algoritmos de pesquisa.

Bibliografia

- Russell and Norvig, (2009) Artificial Intelligence - A Modern Approach.