

**Pesquisa Sistemática/Informada de Soluções:**

Pesquisa em Serviço de Transfer

Relatório Final

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:

Andreia Rodrigues – up201404691 – up201404691@fe.up.pt

Eduardo Leite – gei12068 – gei12068@fe.up.pt

Francisco Queirós – up201404326 – up201404326 @fe.up.pt

21/05/2017

[**Objetivo**](#_qm4vx08q5y8o) **3**

[**Especificação**](#_mwa88afugocj) **4**

[Detalhes](#_cvksexmmerqf) 4

[Representação](#_puq08m3mw7z) 4

[Resolução](#_yiceaqsqc08) 4

[Representação interna do problema](#_89obw3gojji3) 5

[Algoritmo](#_1vltonbgdt3w) 5

[Heuristica](#_jm2kcwwjds3m) 6

[**Desenvolvimento**](#_olig01n2uo9z) **7**

[Ferramentas e ambiente utilizados](#_8e3yomym9xkv) 7

[Estrutura do programa](#_1g3htbjwlwxm) 7

[Implementação](#_f60z0mii4uqe) 7

[**Experiências**](#_k6vjqdiffrd2) **9**

[Resultados primeira experiência](#_rrhg2mutblee) 9

[Resultados segunda experiência](#_xmz4kzp0n8s6) 10

[**Conclusões**](#_wg4ms6qykeqv) **11**

[**Melhoramentos**](#_9mni9odm5y6b) **12**

[**Recursos**](#_a83iazn2fc5y) **13**

# Objetivo

Pretende-se com o projeto realizado implementar um sistema que permita determinar o melhor percurso no transfer entre locais específicos e um aeroporto, representando através de um grafo os determinados locais e os caminhos que ligam os mesmos.

Os clientes deste serviço terão um horário a respeitar, definida pela quantidade máxima de tempo que o serviço pode demorar a levantar o cliente e a colocá-lo no aeroporto.

O melhor percurso vai corresponder ao percurso que minimiza a distância percorrida pelo transfer.

# Especificação

### Detalhes

Este projeto consiste em, imaginando uma situação real, auxiliar uma empresa que fornece serviços de transfer de clientes entre hotéis ou outros locais da região e um aeroporto. O objetivo é determinar os melhores percursos a efectuar pelos veículos de transporte desta empresa dado um certo número de clientes e as suas exigências. Para cada percurso, é necessário determinar o caminho mais vantajoso , ou seja, o caminho que minimiza a distância percorrida e consequentemente os gastos da empresa, sem que as condições em relação ao tempo que cada cliente tem para fazer o percurso sejam violadas.

Sendo que este serviço pretende transportar os clientes de um local até ao aeroporto, cada um destes tem naturalmente uma hora a que tem necessariamente de estar no aeroporto para que não perca o seu voo. É da responsabilidade do serviço de transporte garantir que o cliente chega ao aeroporto antes dessa hora máxima.

Assim, dado um conjunto de pedidos de serviços de transporte para o aeroporto a prestar num determinado dia, com os locais de recolha já definidos e o número de clientes em cada um desses locais de recolha, pretende-se planear as rotas a fazer pelos veículos da empresa, de forma a satisfazer o maior número de pedidos, minimizando a distância a percorrer.

### Representação

Para representar o problema inicial o utilizador deve construir, usando a interface do programa ou através da estrutura de ficheiro utilizado, um grafo cujos nós representam os pontos que o utilizador quer considerar. Estes pontos devem representar intersecções de vias ou locais relevantes e as arestas entre os nós serem as vias que ligam esses pontos. Os custos destas arestas devem representar a distância a percorrer para atravessar a via. O programa cria uma relação direta entre distância e tempo para a percorrer, não tendo em consideração variação de velocidade. O utilizador deve depois marcar os pontos que se referem aos locais pertinentes ao problema, o aeroporto e os hoteís.

### Resolução

Para resolver o problema, reduzimos o grafo original a um grafo que é apenas constituído pelos nós que representam os locais importantes e arestas ligam cada nó para todos os outros nós sendo a distância de cada uma a distância a percorrer para o caminho determinado pelo algoritmo usado e, no caso dos algoritmos de pesquisa com solução ótima para este problema, a distância mínima entre estes locais.

### Representação interna do problema

Formalizamos o problema em 3 objetos:

* Veículos
* Clientes
* Lugares

Cada veículo contém:

* Um identificador único
* Uma lista de clientes dentro do mesmo.
* Local onde está.
* Tempo total gasto a navegar nas vias até o estado atual.

Cada cliente contém:

* Um identificador único
* Local onde está originalmente. Se o cliente estiver num veículo, esta informação deixa de ser relevante.
* Unidade de tempo máxima até estar no aeroporto.

Cada lugar contém:

* Um identificador único

Cada estado do problema vai ter:

* Uma lista de veículos
* Uma lista de clientes
* Uma referência ao grafo reduzido fabricado no passo anterior
* Uma referência ao lugar referente ao aeroporto
* Capacidade que cada veículo deve ter, definido pelo utilizador

### Algoritmo

O estado inicial será haver 0 veículos e ter a lista dos clientes.

Este estado vai gerar um novo estado com um veículo e a mesma lista.

A partir daqui, o algoritmo gera outro estado com 2 veículos mas também gera outros estados referentes à pesquisa de caminho com um veículo.

A partir do momento que um ramo começa a pesquisar caminho com um certo número de veículos, não pode aumentar o seu número de veículos.

Ou seja, vão ser gerados até n estados iniciais, onde n é o número de clientes. Um veículo para cada cliente será o limite máximo de veículos necessários.

Na pesquisa, o algoritmo gera todos os casos combinatórios possíveis.

Se existem 5 hotéis, o estado inicial com 1 veículo cria 5 estados, um para cada hotel; o estado inicial com 2 veículos cria 25 estados; e assim sucessivamente até 5 veículos, por exemplo.

Quando um veículo chega a um hotel, são carregados o máximo número possível de clientes.

Quando um veículo chega ao aeroporto, descarrega todos os clientes. Ao descarregar os clientes, estes desaparecem da memória do estado e por consequência, dos estados descendentes.

Este é o fundamento do algoritmo.

No entanto, existem vários métodos para ignorar estados:

* Ignorar estados similares. Por exemplo, no caso de um veículo que levante clientes do hotel 1 num estado, depois do hotel 2 no estado a seguir e depois do hotel 3 no estado a seguir gera um estado similar a ir 2->1->3. Ao encontrar o segundo destes estados, este é descartado, com a excepção dos tempos gastos forem diferentes.
* Ignorar estados inválidos. Se em algum estado houver um cliente dentro de um veículo e o tempo que o veículo já gastou a navegar nas vias for superior ao tempo do cliente, este estado também é descartado ao ser encontrado.
* Ignorar estados inúteis. Não existe valor nenhum em um veículo ir para um hotel quando este hotel já não tem clientes em si ou quando o veículo está cheio.

O estado final vai surgir quando não houver clientes e todos os veículos se encontrarem no aeroporto.

### Heuristica

A heurística é uma métrica importante para os métodos informados para os apoiar a preferir estados que têm maior potencial de se encontrarem mais perto da solução.

Uma restrição a heurística no caso deste problema é a heurística ter a necessidade de ser admissível. Isto permite a algoritmos como o A\* encontrar a solução ótima.

A heurística selecionada pelo grupo para este trabalho resume-se a somar as distâncias de todos os hotéis com clientes ao aeroporto. Esta heurística é admissível visto que no mínimo, um veículo tem de ir ao hotel e depois para o aeroporto.

# Desenvolvimento

### Ferramentas e ambiente utilizados

Para o desenvolvimento deste programa foi utilizada a linguagem de programação Java junto com o IDE IntelliJ IDEA em ambiente Microsoft Windows 10. Foi também utilizada a ferramenta GraphViewer que foi apresentada na unidade curricular de Concepção e Análise de Algoritmos (CAL) devido ao conhecimento prévio da API por parte da equipa.

### Estrutura do programa

A aplicação é estruturada em 3 partes principais:

* *Front-end* do programa (manipulação do GraphViewer e *input/output* nas *streams* principais do programa, sendo direcionado para o uso com uma interface por linha de comandos)
* Esqueleto para o fabrico, manipulação de, e cálculo sobre grafos. Aqui é onde estão presentes todos os algoritmos principais para pesquisa.
* Formalização do problema com o uso de classes que contém informação relevante para cada ator (e.g. clientes, veículos) e para a representação de cada estado do problema. Encontra-se aqui também a algoritmia necessária para a resolução do problema que não o algoritmos de pesquisa em grafo.

Para cada uma destas 3 partes existe um módulo no código designado para atender às respetivas responsabilidades.

Existe mais um módulo adicional que contém apenas código utilitário de pequena responsabilidade.

### Implementação

Na especificação que nos foi fornecida era sugerido abordar o problema em 3 fases de desenvolvimento:

1. Assumir que haveria apenas um veículo de transporte e que este tinha capacidade limitada de clientes, podendo ser necessário fazer várias viagens para transportar todos os clientes.
2. Poder agora usar vários veículos de transporte mas que a sua multiplicidade deve ser alvo de minimização.
3. Agora ter em conta que os clientes têm a limitação de tempo referida anteriormente na descrição detalhada do problema.

O grupo durante o desenvolvimento acabou por seguir o desenvolvimento pelas limitações 1->3->2. O uso de vários veículos foi adiado para o final devido à dificuldade de concepção do algoritmo que permite a combinatória de todas as decisões de percurso dos veículos. Mas apesar desta dificuldade, a equipa conseguiu encontrar uma solução satisfatória, embora podendo ser melhorada, como iremos referir na secção adequada deste relatório.

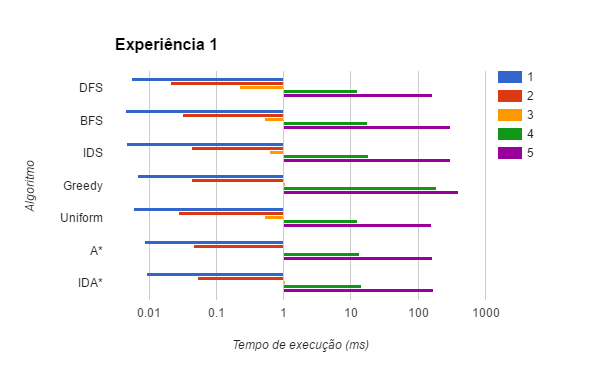
# Experiências

O grupo decidiu fazer experiências para determinar a variação de tempo a resolver o problema.

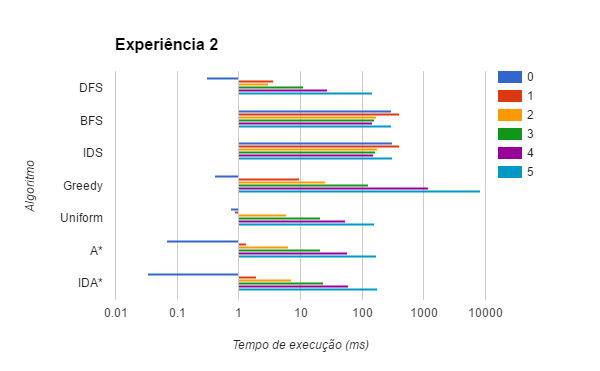
Decidimos então fazer duas experiências:

* Na primeira decidimos criar os pior casos possíveis. Estes são aqueles onde existem n clientes que todos eles têm um tempo limite equivalente a duas vezes o tempo mínimo para chegar ao aeroporto ao seu hotel. Nestes estados o algoritmo vai ser obrigado a usar mais veículos, n veículos, e com mais veículos, mais fator de ramificação e então mais estados e mais complexidade tanto temporal como espacial. Vamos aumentando o n a partir de 1 e ver o crescimento temporal.
* Na segunda vamos usar um número fixo de 5 hotéis com um cliente cada um. No primeiro teste todos os clientes dão tempo o suficiente para o veículo levantar a todos. No teste a seguir, um dos clientes tem a necessidade de imediatamente ser levado ao aeroporto como os da experiência anterior. No a seguir, dois destes clientes. E assim sucessivamente.

### Resultados primeira experiência



### Resultados segunda experiência



# Conclusões

Na experiência 1 consegue-se observar o crescimento muito rápido com o aumento de hotéis. Embora essa experiência tenha também a variável de todos os clientes serem urgentes, na experiência 2 usa-se um número fixo de hotéis/clientes aumentando o número de clientes urgentes.

Conclui-se que o BFS e o IDS são relativamente estáveis se o número de hotéis for constante ao contrário dos outros algoritmos. No entanto, apesar dessa estabilidade, o valor base é bastante alto comparativamente aos outros métodos, nomeadamente os de custo uniforme e A\*.

O algoritmo guloso cresce muito rapidamente à medida que a quantidade de clientes urgentes aumenta. Devido à maneira que a heurística está feita, o greedy vai querer levantar o máximo de clientes na mesma viagem e os clientes urgentes impedem isso.

É importante notar no entanto que dos algoritmos usados apenas os algoritmos de custo uniforme, A\* e IDA\* retornam soluções ótimas.

Este problema, pelo menos com esta abordagem, tem a inconveniência de ter a característica de ter um crescimento muito rápido nas suas complexidades tanto temporais como espaciais.

# Melhoramentos

Para trabalho futuro, há espaço para melhoria nas seguintes áreas:

* Aperfeiçoamento nas estruturas de dados, particularmente nas que estão relacionadas com os grafos. Isto com o objetivo de reduzir a pegada de memória do programa durante a execução.
* Desenvolver uma heurística mais próxima do custo real, mantendo um custo de cálculo baixo, de forma a evitar estados menos favoráveis.
* Descobrir novas formas de determinar se um estado não vai ter descendentes que sejam uma solução. Se descobrirmos mais destes estados, podemos podar todos os descendentes e reduzir o espaço de pesquisa mais cedo, poupando em tempo e espaço em memória.
* Os custos das arestas dos grafos são interpretados como distâncias de custo igual, ou seja, não usam a possibilidade da velocidade de veículo ser variável dependendo de fatores como trânsito elevado nas vias, vias com melhores condições ou com menos restrições de velocidade.
* Poderá haver implementações não perfeitas dos algoritmos de pesquisa que podem levar a incoerências nas complexidades temporais ou espaciais dos mesmos. Melhorar estes aspetos poderia trazer bastante benefício ao bom desempenho do programa.

# Recursos

Não foi usado outro software que não o já mencionado para o desenvolvimento.

Foram utilizados os apontamentos das aulas para adquirir conhecimento sobre os algoritmos utilizados.

O grupo acha que em contribuições houve um equilíbrio com todos os elementos, ou seja:

* Andreia Rodrigues: ~33%
* Eduardo Leite: ~33%
* Francisco Queirós: ~33%