

**Optimização da Aterragem de Aviões**

*Relatório Intercalar*

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:

Carlos Teixeira – código – ei11145@fe.up.pt

Leonel Rocha – código – ei11130@fe.up.pt

Pedro Silva – código – [ei222@fe.up.pt](mailto:ei222@fe.up.pt)

24 de Abril de 2014

Índice

[Introdução 3](#_Toc386069274)

[Objetivo 3](#_Toc386069275)

[Descrição 3](#_Toc386069276)

[Estruturas de Dados 3](#_Toc386069277)

[Avião *3*](#_Toc386069278)

[Algoritmo *4*](#_Toc386069279)

[Solução *5*](#_Toc386069280)

[Trabalho Efetuado 5](#_Toc386069281)

[Resultados esperados 6](#_Toc386069282)

[Conclusões 6](#_Toc386069283)

[Recursos 6](#_Toc386069284)

# Introdução

Quer sobre a forma de dispositivos avançados como smartphones, *tablets* ou computadores quer embebidos em dispositivos aparentemente mais simples como eletrodomésticos e televisões os agentes inteligentes ocupam atualmente um papel muito importante no nosso dia a dia.

Com o intuito de compreender como funcionam estes agentes e as bases teóricas que os suportam propusemos desenvolver um projeto em C++ o qual fará uso das técnicas ligadas à inteligência artificial(algoritmos genéticos, arrefecimento simulado e *Custo Uniforme*).

Este projeto irá consistir no desenvolvimento de algoritmos de otimização para um problema de escalonamento relacionado com aterragens de aviões.

# Objetivo

Tal como referido anteriormente, tencionamos desenvolver um projeto com o objetivo de escalonar aterragens de vários aviões numa pista de aeroporto. O escalonamento procurará minimzar o custo da aterragem de cada avião, o qual é calculado de acordo com uma função definida para cada avião que por sua vez depende da hora a que o avião aterra.

Assim sendo, esperamos ser capazes de, no final do projeto, apresentar soluções para o problema utilizando diferentes algoritmos e estabelecendo relações entre a eficiência dos vários métodos implementados.

# Descrição

Descrição dos Algoritmos

# Estruturas de Dados

Para uma mais sucinta interpretação do trabalho e da sua estrutura, apresentamos, de seguida, a estrutura adotada e uma breve explicação da mesma. **Esta compreende três grandes componentes:**

1. *Avião*

A primeira estrutura de dados representa um avião e para este definimos as seguintes variáveis :

- **Hora preferencial de aterragem(HPA)** : hora a que o avião pretende aterrar com custo de aterragem igual a zero.

- **Janela temporal :** intervalo de tempo que o avião tem para fazer a aterragem.

- **Valor da função de custo de aterragem retardada(VCAR) :** valor utilizado para calcular o custo de uma aterragem após HPA.

- **Valor da função de custo de aterragem adiantada(VCAA) :** valor utilizado para calcular o custo de uma aterragem antes de HPA.

- **Perído de ocupação da pista :** após a aterragem existe um período de tempo durante o qual mais nenhum avião poderá aterrar.

E o seguinte método:

-**Função de custo:** A função de custo é computada de acordo com a fórmula:

1. *Algoritmo*

Cada um dos algoritmos implementados utiliza uma classe homónima para diferenciar as chamadas e permitir que todos possam correr num único executável.

**Algoritmo Genético**

[A Preencher]

**Arrefecimento Simulado**

O Arrefecimento Simulado, por sua vez, apenas utiliza a classe **Solução** para organizar num vetor o caminho seguido até à solução final.

Para além desse vetor de instanciações da classe **Solução** a implementação do Arrefecimento Simulado utiliza os seguintes campos:

* **fatorReducao**: contém o fator de redução a aplicar à temperatura a cada iteração;
* **temperatura**: guarda a temperatura a dado momento do algoritmo;
* **variacaoEnergia**: guarda a variação que ocorreu na energia de uma solução em determinada iteração;
* **solucaoInicial**: contém uma instanciação da classe **Solução** que reflete a solução utilizado no início da iteração;
* **solucaoAtual**: contém uma instanciação da classe **Solução** que reflete a solução após ser submetida a uma perturbação;

No que toca a funções, a classe que executa o Arrefecimento Simulado tem as seguintes funções auxiliares:

* **geraEstadoInicial():** que gera um estado inicial aleatório a ser usado na primeira iteração;
* **perturbacao():** que exerce uma perturbação na solução da iteração atual;
* **condExpAleatoria():** que calcula a expressão aritmética de aceitação da perturbação efetuada. Tem em conta a variação de energia e a temperatura atual.

**Custo Uniforme**

O algoritmo Custo Uniforme faz uso de duas estruturas **Node** e **TimeInterval** assim como da estrutura **Avião** supramencionada para a definição do seu espaço de soluções.

**Node -** esta estrutura de dados é usada para representar um nó no grafo a ser precorrido pelo algoritmo, a esta estrutura estão definidos os seguintes campos:

* **level**: Campo utilizado para especificar a que profundidade da raiz do grafo, o nó se encontra.
* **departTime**: Valor contendo a hora para o aviao contido neste nó (ver “plane” abaixo).
* **parent**: Apontador usado para fazer a ligação entre o nó atual e o seu pai, efetivamente funcionando como uma aresta no grafo.
* **branches**: Vetor de apontadores para os nós-filhos de um determinado nó, tendo a mesma funcionalidade que o campo parent (usado para navegar no grafo de pai para filho).
* **plane**: Apontador para o avião a ser representado neste nó, contêndo todas as suas caracteristicas já mencionadas.
* **restrictions**: Vetor de timeInterval usado para representar todas as restrições temporais que existem para chegar ao nó. Herda as restrições dos nós ancestrais para validar ou não uma possivel junção a outros nós.

**TimeInterval** - esta estrutura de dados é usada para representar o intervalo de tempo, ao qual um avião está associado. Isto é, indica o tempo em que o avião associado irá aterrar assim como o tempo que a pista está indesponível para utilização.

* **start**: Valor indicativo do inicio do periodo a que um avião está vinculado.
* **finish**: Valor correpondente ao fim do periodo ao qual o avião está associado.

1. *Solução*

A representação da solução é dependente do método utilizado sendo implementados diferentes métodos de acordo com o algoritmo a que se destinam.

Faz parte do esqueleto comum um recipiente que contém informação relativa às aterragens a qual relaciona um avião com uma hora da sua aterragem e um método capaz de calcular o valor de cada solução tendo em conta penalizações para soluções inválidas.

**Algoritmo Genético**

[A Preencher]

**Arrefecimento Simulado**

Tal como referido anteriormente aquando da especificação da estrutura, a implementação do algoritmo de Arrefecimento Simulado conta com um vetor de instanciações da classe Solucao que guarda o caminho seguido pelo algoritmo.

Quanto à classe **Solução** em si, o algoritmo de Arrefecimento Simulado apenas adiciona a função intersectaIntervalo() que se é utilizada para detetar sobreposições de aterragens.

**Custo Uniforme**

Para representar a solução ao problema proposto, utilizamos para o algoritmo Custo Uniforme, um vetor de apontadores de **Nodes**, sendo que este se encontra ordenado de fim para início pois é construido durante a execução do programa.

A cada elemento deste vetor o melhor tempo para o aviao contido no elemento encontra-se no campo *departTime*. A esta estrutura para se descobrir o custo associado à solução apenas é utilizar a função *getTotalCost* do primeiro elemento, sendo que este navega posteriormente pelos seus antecessores somando o custo de cada *departTime*.

# Trabalho Efetuado

Encontram-se no momento presente implementadas versões preliminares dos três algoritmos em C++. As implementações ainda não apresentam as melhores soluções e, por conseguinte, necessitam de melhoramentos ao nível da otimização.

**Algoritmo Genético**

[A Preencher]

**Arrefecimento Simulado**

Até ao presente ponto de avaliação, o algoritmo de Arrefecimento Simulado já foi correctamente implementado e é capaz de devolver uma solução boa para os problemas apresentados. No entanto, na nossa opinião, a implementação ainda pode ser mais agilizada e os resultados podem se aproximar ainda mais da solução ótima.

O programa não apresenta problemas no que toca a tempo de execução e corre sem problemas qualquer tipo de problema, sendo que, os problemas impossíveis apresentarão um custo elevado e imprático e terminarão com a restrição de número máximo de iterações.

**Custo uniforme**

O trabalho realizado neste algoritmo conta já com a especificação e correto funcionamento das estruturas de dados inerentes, assim como um esboço do preliminar do algoritmo, sendo que está já gera ramificações apartir de um nó com as restrições inerentes ao próprio e avalia corretamente o custo da solução temporária encontrada.

Neste momento o algoritmo está a funcionar para os casos mais simples, isto é o algoritmo consegue calcular a solução, embora para um numero substâncial de aviões, cada um com um intervalo de tempo considerável, o programa torna-se bastante lento, pelo é necessário melhorar a performance a nível de código para evitar fazer a funções redundantes assim como possivelmente limitar mais à fila de espera que contêm os nós a serem percorridos pelo algoritmo. Embora o algoritmo em si, não seja eficaz para resolver o problema proposto, o grupo acredita que é possível melhorar a implementação atual.

# Resultados esperados

Espera-se que no final do projeto sejamos capazes de gerar soluções viáveis usando diferentes algoritmos com o objetivo de comparar a eficiência e validade das mesmas.

Adicionalmente seria interessante criar uma interface que permitisse visualizar estatísticas sobre os diferentes algoritmos e o caminho que seguiram para alcançar a solução.

Para além destas interfaces, tencionamos criar um suporte para importação de aviões utilizando ficheiros de texto.

# Conclusões

Relativamene ao presente projeto, assumimos um balanço extremamente positivo no que toca consolidação os conhecimentos obtidos e no aumento do domínio sobre problemas de escalonamento.

Conseguimos ainda familiarizarmo-nos com as diferenças entres os diferentes algoritmos e as respetivas implementações, desta feita, em C++.

Concluindo, esperamos não só conseguir cumprir as tarefas esperadas mas também melhorar as otimizações já implementadas relativamente ao problema dado.

# Recursos

Para além dos slides das aulas teóricas que foram disponibilizados na página da unidade curricular, no desenvolvimento do projeto, utilizámos o Microsoft Visual Studio como ambiente de desenvolvimento integrado.

Adicionalmente, de notar a importante orientação prestada pelo monitor da disciplina Tiago Azevedo.