

**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

24 de Abril de 2014

# 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 *Branch and bound*).

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

**A Estruturas de Dados:**

**A estrutura de dados adotada compreende três grandes componentes:**

**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:

**Algoritmo :**

**Algoritmo Genetico:**

**Arrefecimento Simulado**

**Branch and Bound:**

O algoritmo Branch and Bound 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.

**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 genetico**

**Arrefecimento Simulado**

**Branch and Bound:**

Para representar a solução ao problema proposto, utilizamos para o algoritmo *Branch and Bound*, 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 genetico**

**Arrefecimento Simulado**

**Branch and Bound:**

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 acredito 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

# Recursos

No desenvolvimento deste projeto foi utilizado o Microsoft Visual Studio com ambiente de desenvolvimento integrado.

Adicionalmente o monitor da disciplina Tiago Azevedo foi uma grande ajuda permitindo-nos fazer amor com ele ;) . lolololol