Universidade Federal da Paraíba
Centro de Informática
Disciplina: Análise e Projeto de Algoritmos
Professor: Bruno Bruck

Projeto Final

Descrição do problema

O Aeroporto APA, um dos principais *hubs* de transporte aéreo da região, passou recentemente para a administração de uma empresa privada. Com essa mudança, a nova gestão busca modernizar as operações do aeroporto e tornar sua logística mais eficiente. Um dos desafios identificados é a programação dos pousos e decolagens, um fator essencial para evitar atrasos, otimizar o uso das pistas e garantir a segurança das operações.

Para resolver esse problema, você foi contratado para desenvolver um algoritmo que ajude a programar os pousos e decolagens de forma otimizada. O objetivo é evitar congestionamentos nas pistas, reduzir os atrasos e garantir que os intervalos de segurança entre os voos sejam respeitados. Esse intervalo é fundamental porque, entre um voo e outro, é preciso considerar dois fatores críticos: o tempo de taxiamento e a turbulência de esteira.

O tempo de taxiamento refere-se ao deslocamento das aeronaves entre a pista e os portões de embarque ou desembarque. Uma pista não pode ser liberada para outro voo imediatamente após um pouso ou decolagem, pois a aeronave anterior precisa de tempo para sair do local com segurança. Além disso, a turbulência de esteira ocorre quando uma aeronave deixa um rastro de ar turbulento que pode comprometer a estabilidade da próxima, especialmente se for menor e mais leve. Por isso, é essencial garantir um intervalo adequado entre voos consecutivos na mesma pista.

O desafio é distribuir os voos entre as pistas disponíveis, garantindo que nenhum fique esperando mais do que o necessário e que as operações ocorram da maneira mais eficiente possível. Sua solução ajudará a nova administração a transformar o Aeroporto APA em um modelo de eficiência, melhorando a experiência dos passageiros e a pontualidade dos voos.

Formalmente, o problema pode ser descrito da seguinte forma:

> "Seja V o conjunto de voos que devem pousar e decolar de um aeroporto com m pistas. Cada voo $i \in V$ possui um horário $r_i \geq 0$ a partir do qual a decolagem ou o pouso pode ser autorizado, e um tempo $c_i \geq 0$, necessário para se completar o pouso ou a decolagem uma vez iniciado. Para cada par de voos distintos $i, j \in V$, há um tempo obrigatório de espera $t_{ij} \geq 0$, que deve ser respeitado se o voo j for programado imediatamente após o voo i na mesma pista. Esse tempo considera restrições operacionais de segurança relacionadas ao taxiamento (deslocamento) e turbulência de esteira entre aeronaves. Além disso, cada voo $i \in V$ possui uma penalidade por atraso $p_i \geq 0$, que corresponde ao custo a ser pago por cada unidade de tempo de atraso em relação ao tempo de liberação r_i. O objetivo do problema é determinar um escalonamento dos voos nas m pistas do aeroporto de forma a minimizar as penalidades por atraso, respeitando-se os tempos de liberação de cada voo."

Exemplo de instância e solução

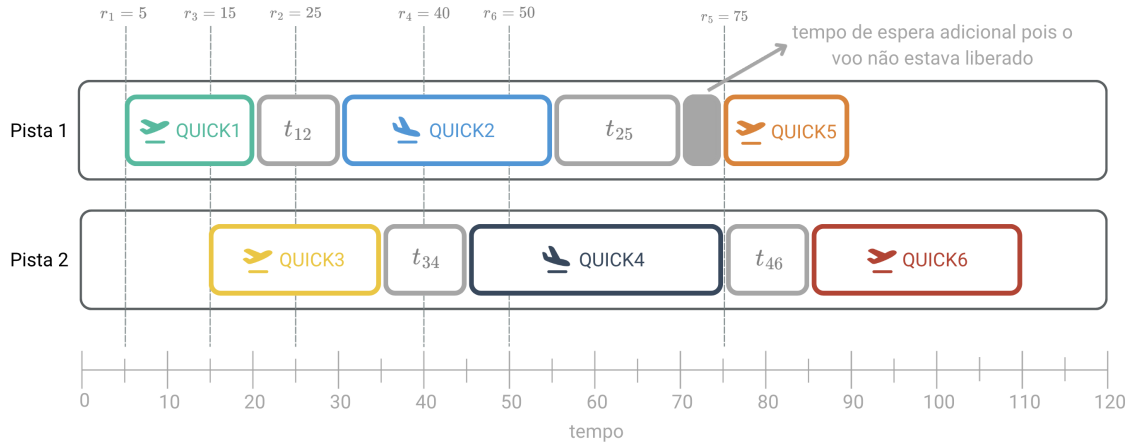
Para exemplificar o problema, considere uma instância (cenário) do aeroporto APA, o qual possui 2 pistas e 6 voos diários como a seguir:

Código voo	Itinerário	r	c	p
QUICK1	APA → JPA	5	15	55
QUICK2	REC → APA	25	25	90
QUICK3	APA → GRU	15	20	61
QUICK4	GRU → APA	40	30	120
QUICK5	APA → REC	75	15	45
QUICK6	APA → FOR	50	25	50

A matriz t contendo os tempos de espera entre dois voos consecutivos é como a seguir:

	QUICK1	QUICK2	QUICK3	QUICK4	QUICK5	QUICK6
QUICK1	-	10	15	8	21	15
QUICK2	10	-	10	13	15	20
QUICK3	17	9	-	10	14	8
QUICK4	11	13	12	-	10	10
QUICK5	5	10	15	20	-	12
QUICK6	5	10	15	20	28	-

Considerando essa instância, o seguinte planejamento é uma solução viável (não necessariamente a ótima) para o problema.



Para calcularmos o valor da solução acima é necessário calcular as possíveis multas por atraso de cada voo usando a seguinte fórmula:

> Multa = Valor Multa por Minuto *(Tempo de início do pouso ou decolagem − Tempo de liberação do voo)

Sendo assim, os valores de multa por voo são os seguintes:

QUICK1: Multa $= 55 * (5 - 5) = 0$

QUICK2: Multa $= 90 * (30 - 25) = 450$

QUICK3: Multa $= 61 * (15 - 15) = 0$

QUICK4: Multa $= 120 * (45 - 40) = 600$

QUICK5: Multa $= 45 * (75 - 75) = 0$

QUICK6: Multa $= 50 * (85 - 50) = 1750$

Portanto, no exemplo, o valor da solução é $0 + 450 + 0 + 600 + 0 + 1750 = 2800$.

Instruções

O projeto deve ser realizado em grupo de **3 integrantes (da mesma turma)** e vale 10 pontos, relativos à terceira nota da disciplina. Cada grupo deve desenvolver um algoritmo eficiente de busca local (ou meta-heurística) para o problema de otimização descrito acima. O código-fonte deve ser **obrigatoriamente** escrito na linguagem **C/C++**.

Note que o seu programa deve ser capaz de ler um arquivo contendo os dados de uma instância (cenário) do problema e utilizar tais dados como entrada para o algoritmo. O formato de arquivo a ser utilizado é o seguinte:

```
numero_de_voos
numero_de_pistas

array r
array c
array p

matriz t
```

A instância utilizada na seção anterior, por exemplo, poderia ser representada pelo seguinte arquivo:

```
6
2

 5  25  15  40  75  50
15  25  20  30  15  25
55  90  61  120 45  50

  0    10    15     8    21    15
 10     0    10    13    15    20
 17     9     0    10    14     8
 11    13    12     0    10    10
  5    10    15    20     0    12
  5    10    15    20    28     0
```

Ao final da execução, seu código deve produzir um arquivo de saída, no seguinte formato, contendo a melhor solução encontrada:

```
<valor da solucao>
<lista de voos alocados na pista 1>
<lista de voos alocados na pista 2>
...
<lista de voos alocados na pista m>
```

Por exemplo, a solução mostrada na seção anterior geraria o seguinte arquivo de saída:

```
2800
1 2 5
3 4 6
```

Etapas e prazos

Este projeto contém os seguintes entregáveis:

- Implementação de **ao menos um algoritmo guloso** para a geração de uma solução viável.
- Implementação de **pelo menos 3 movimentos de vizinhança**.
- Implementação do algoritmo de busca local chamado VND (Variable Neighborhood Descent).
- Implementação de uma meta-heurística (OPCIONAL). Sugestões: GRASP ou ILS.

- Resultados computacionais: **criar uma tabela** que contenha os resultados obtidos pela(s) heurística(s) construtiva(s) e pelo VND, e que compare tais resultados com a solução ótima de cada instância. Essa tabela deverá conter os seguintes dados para cada heurística construtiva e para o VND:
  - Média do valor da solução (em no **mínimo 10 execuções** para cada instância caso exista algum fator aleatório no algoritmo);
  - Melhor solução encontrada;
  - Média do tempo gasto pelo respectivo algoritmo;
  - GAP para a solução ótima.

  Observação: Caso decida implementar a meta-heurística, é necessário adicionar colunas de resultados para ela na tabela.
- Todas as implementações devem vir acompanhadas de um arquivo ***makefile*** para a compilação. Tal arquivo deve ser preparado de forma a funcionar em sistemas UNIX.
- Criar uma pasta contendo os arquivos de saída gerados durante os testes com cada instância. Favor incluir somente os resultados dos testes finais, com a versão a ser entregue.

O projeto deve ser entregue até as **23:59 do dia 20 de abril de 2025**. Devem ser enviados via SIGAA, o código-fonte do projeto e um relatório em *pdf* contendo o nome dos integrantes do grupo e a tabela de resultados computacionais. Note que é necessário somente uma entrega por grupo e não serão aceitos envios por e-mail ou fora do prazo.

Avaliação

Cada grupo deverá apresentar presencialmente o projeto em data a ser agendada pelo professor. A nota do projeto é individual e leva em consideração diversos critérios, como demonstração de entendimento do código na apresentação, qualidade do código, eficiência dos algoritmos implementados, qualidade dos resultados obtidos, dentre outros. **Não apresentar o projeto implica em nota zero**.

Dicas

Como calcular o valor da medida GAP: Suponha que desejamos calcular o valor GAP para o resultado da heurística construtiva para a instância chamada *nome_instancia*. Supondo que o valor encontrado pela heurística para essa instância é dado por $valor_{heuristica}$ e o valor ótimo para essa instância é $valor_{otimo}$, o cálculo do GAP é realizado da seguinte forma:

$$gap = \left(\frac{valor_{heuristica} - valor_{otimo}}{valor_{otimo}} \right) \times 100$$

Note que o valor do gap é dado em percentagem (%) e indica a "distância" da solução, no caso, da heurística construtiva para o valor ótimo.

Para calcular o GAP dos resultados obtidos pelo VND basta substituir $valor_{heuristica}$ pelo valor encontrado pela VND.

Exemplo de tabela de resultados:

		Heurística construtiva			VND		
	ótimo	valor solução	tempo	gap	valor solução	tempo	gap
instancia1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
instancia2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
instancia3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
instancia4	0.0	0.0	0.0	0.0	0.0	0.0	0.0