Relatório: Gerenciamento de Aterrissagem de Aeronaves em Aeroporto

Autores

Guilherme Henrique Silva Barbara;

Lucas Malachias Furtado;

Orientadores

Andreza Cristina Beezao Moreira;

Mayron Cesar de Oliveira Moreira.

1. Introdução

O presente trabalho tem como objetivo otimizar o processo de gerenciamento de aterrissagem de aeronaves em um grande aeroporto brasileiro. Com a recente rodada de investimentos privados, o aeroporto busca automatizar a operação de aterrissagem de aeronaves no pátio principal. Diariamente, um conjunto de aviões pousa, e cada aeronave possui informações detalhadas sobre o tempo ideal de pouso, penalidades por antecipação ou atraso em relação ao tempo ideal, entre outros dados.

A solução para esse problema envolve a sequência de aterrissagem das aeronaves de maneira que se minimize as penalidades relacionadas aos tempos de aterrissagem antes ou depois do tempo ideal. Além disso, deve-se respeitar a separação necessária entre os aviões. Para isso, utilizamos dois métodos principais: programação linear inteira (com a ajuda de solvers) e meta-heurísticas, especificamente a **Variable Neighborhood Search (VNS)**.

Este trabalho apresentará os resultados obtidos a partir da aplicação de ambos os métodos de resolução, discutindo as vantagens e limitações de cada um no contexto da otimização do processo de aterrissagem. Além disso, serão abordadas as dificuldades encontradas durante a modelagem do problema e a escolha dos solvers e heurísticas, como questões relacionadas ao tempo de execução e à qualidade das soluções obtidas. Por fim, o relatório irá avaliar a eficácia das abordagens, comparando-as com soluções tradicionais, destacando as melhorias no desempenho do sistema de aterrissagem e propondo possíveis caminhos para futuras pesquisas e aprimoramentos.

2. Formulação do Problema

O problema foi formulado de acordo com as informações fornecidas no enunciado. Cada avião **i** possui as seguintes variáveis:

- **Ri**: Tempo de detecção pelo radar;
- **Ei**: Tempo inicial de pouso;
- **Ti**: Tempo ideal para o pouso;
- **Li**: Tempo final que o avião pode pousar;
- **gi**: Penalidade por unidade de tempo se o avião pousar antes do tempo ideal;
- **hi**: Penalidade por unidade de tempo se o avião pousar depois do tempo ideal;

Além disso, temos a matriz **S = [sij]**, onde **sij** representa o tempo de separação necessário entre o pouso do avião **i** e o pouso do avião **j**.

O objetivo é minimizar as penalidades associadas aos tempos de aterrissagem, respeitando as separações entre os pousos dos aviões.

O modelo de programação linear pode ser descrito como:

$$\text{Min } Z = \sum (g_i \cdot e_i + h_i \cdot d_i)$$

Sujeito a:

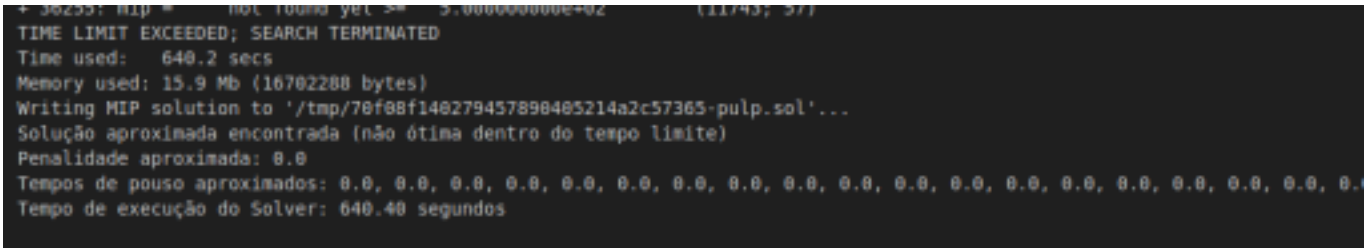
- $t_j \geq t_i + s_{ij} - M(1 - x_{ij}), \forall i,j, i \neq j;$
- $t_i \geq t_j + s_{ji} - M(x_{ij}), \forall i,j, i \neq j;$
- $x_{ij} + x_{ji} = 1, i,j, \forall i \neq j;$
  Onde **xij** e **xji** são variáveis binárias.
- $e_i \geq T_i - t_i, \forall i;$
- $d_i \geq t_i - T_i, \forall i;$
- $e_i \leq T_i - E_i, \forall i;$
- $d_i \leq L_i - T_i, \forall i.$

3. Descrição da Solução

3.1 Resolução com Solver (CBC)

Primeiramente, começamos o desenvolvimento com o GLPK (GNU Linear Programming Kit), uma biblioteca para resolver problemas de programação linear e inteira. No entanto, como ele utiliza métodos como o **Simplex** e o **Branch-and-Bound**, que, em problemas com muitas variáveis e restrições, podem ter um desempenho muito lento, logo enfrentamos problemas relacionados ao tempo de execução, que estava bastante elevado, especialmente para instâncias maiores do problema (08.dat). O GLPK, embora seja uma ferramenta poderosa, não foi eficiente para o nosso caso.

Figura 1: Resultados obtidos na execução da instância 08.dat usando o solver GLPK.



Diante disso, optamos por utilizar o **CBC (Coin-or Branch-and-Cut)**, que é um solver desenvolvido pela **COIN-OR** (Computational Infrastructure for Operations Research). O CBC é um solver de programação inteira e linear, projetado para ser mais eficiente em termos de tempo de execução, especialmente quando lidamos com problemas de otimização de grande escala.

Ele é bastante conhecido pois utiliza um algoritmo de **Branch-and-Cut**, que é uma extensão do **Branch and-Bound**, que além de realizar a busca por ramificação, também corta regiões do espaço de soluções que não são promissoras, o que pode acelerar significativamente a resolução de problemas de programação inteira.

3.2 Resolução com Heurística (VNS)

A Variable Neighbourhood Search (VNS) é uma metaheurística baseada na exploração sistemática de diferentes estruturas de vizinhança para escapar de ótimos locais e encontrar soluções melhores. O algoritmo começa com uma solução inicial e, a cada iteração, perturba essa solução de diferentes formas, buscando uma melhora.

Funcionamento do VNS

1. **Geração da solução inicial**: Cria-se uma solução viável, geralmente baseada em alguma heurística simples.
2. **Exploração de vizinhança**: Aplicam-se diferentes perturbações na solução atual para gerar novas soluções candidatas.
3. **Avaliação da solução**: Se a solução perturbada for melhor, ela substitui a solução atual e o processo recomeça.
4. **Critério de parada**: O processo continua até atingir um número máximo de iterações ou uma outra condição de convergência.

No contexto do nosso problema, o objetivo é minimizar penalidades causadas por pousos antecipados ou atrasados, respeitando restrições de separação entre aviões.

Escolhas na implementação do VNS

1. **Geração da solução inicial**

A solução inicial é ordenada pelo tempo ideal de pouso de cada avião:

indices_ordenados = sorted(range(n), key=lambda i: tempos[i][2])

Isso gera uma solução inicial razoável, já que respeita parcialmente as preferências de pouso e reduz penalidades grandes logo no início.

2. **Função de avaliação**

A penalidade total é calculada com base no horário real de pouso dos aviões e penalidades associadas a adiantamentos e atrasos:

```
atraso = max(0, tempos_pouso[i] - tempos[i][2])

adiantamento = max(0, tempos[i][2] - tempos_pouso[i])

penalidade_total += atraso * penalidades[i][1] + adiantamento * penalidades[i][0]
```

3. **Estruturas da vizinhança**

A perturbação da solução é feita de três formas diferentes:

- **Troca simples entre dois aviões aleatórios**: Isso gera pequenas alterações e permite explorar soluções próximas.
- **Troca de blocos de aviões**: Essa estratégia altera um conjunto maior de posições, ajudando a escapar de ótimos locais.
- **Deslocamento circular de um avião**: Essa técnica evita que um único avião permaneça fixo, permitindo mais variedade nas soluções.

Essas três abordagens são alternadas a cada iteração do **VNS**, garantindo uma boa diversificação na busca.

4. **Estratégia de busca**

O VNS busca uma nova solução a cada iteração, sempre aceitando uma solução melhor:

```
if nova_penalidade < melhor_penalidade:
    melhor_solucao, melhor_penalidade = nova_solucao, nova_penalidade
    iteracao = 0
else:
    iteracao += 1
```

Isso mantém a exploração ativa e reinicia o contador de iterações sempre que uma melhoria for encontrada.

5. **Critério de parada**

O algoritmo para quando atinge 5000 iterações sem melhoria, garantindo que o tempo de execução não seja excessivamente longo.

4. Resultados Obtidos e Análise

Para as diferentes instâncias de teste fornecidas, aplicamos tanto o solver quanto a meta-heurística e obtivemos os seguintes resultados:

Instância	Solução Ótima	Valor da Solução Final (SF) (CBC)	Valor da Solução Inicial (SI) (VNS)	Valor da Solução Final (SF) (VNS)	Desvio Percentual (SI - SF)/SI (VNS)	Desvio Percentual da SF em relação à Solução Ótima (CBC)	Desvio Percentual da SF em relação à Solução Ótima (VNS)	Tempo Computacional (Heurística)	Tempo Computacional (Solver)
1	700	699.99	2830.00	1500.00	47	0.0014	114	0.05	0.22
2	1480	1480.00	4520.00	3380.00	25	0.00	56.21	0.08	1.75
3	820	820.00	6990.00	3190.00	54	0.00	74.29	0.15	0.51
4	2520	2520.00	5610.00	2950.00	47	0.00	14.57	0.13	16.87
5	3100	3100.00	8330.00	3580.00	57	0.00	15.48	0,15	55.42
6	24442	24442.00	24442.00	10668.00	56	0.00	-12.91	0.31	0.09
7	1550	1550.00	1550.00	1463.00	5.61	0.00	-5.94	0.18	1.83
8	1950	1950.00	58525.00	13550.00	76	0.00	85.60	0.96	17.11

Solução Inicial (SI): Valor obtido na solução gerada inicialmente;
Solução Final (SF): Valor final encontrado após a execução do solver e da meta-heurística;
Desvio Percentual 100 x (SI - SF)/SI: Cálculo do desvio percentual entre a solução inicial e a solução final;
Desvio Percentual em relação à solução ótima: Cálculo do desvio em relação à solução ótima fornecida;
Tempo Computacional: Tempo total gasto para encontrar a solução com a heurística e o solver.

5. Dificuldades enfrentadas

- **Dificuldade em entender a estrutura do arquivo**: Inicialmente, a entrada dos dados no formato fornecido era difícil de interpretar. A necessidade de separar corretamente os dados para alimentar o solver foi uma das etapas mais desafiadoras. Após compreender melhor o formato e adaptar a entrada, conseguimos prosseguir com a implementação.
- **Definição do modelo e restrições**: A definição das restrições do problema no modelo linear exigiu cuidado para garantir que as separações entre os aviões fossem respeitadas e que as penalidades fossem calculadas de maneira correta. Cada restrição de separação foi incorporada com base na matriz **S**.
- **Desempenho do Solver**: Utilizando o solver GLPK, observamos que a solução com instâncias maiores (a partir da instância 4, que tem 20 aviões) demorava significativamente. O tempo de processamento aumentava exponencialmente à medida que o número de aviões aumentava. Em instâncias grandes, o solver demorava a encontrar uma solução viável, o que levou à necessidade de explorar alternativas, como o solver CBC.

• **Desafios na implementação da Metaheurística:**

- **Balanceamento entre diversidade e exploração local:** Uma das dificuldades enfrentadas foi escolher o tipo e a frequência das perturbações na solução. Inicialmente, tentamos várias abordagens, como a troca de blocos grandes de aviões ou perturbações mais agressivas, o que gerava movimentos muito grandes no espaço de solução. Isso acabava prejudicando a busca, já que muitas vezes o algoritmo se afastava de regiões promissoras do espaço de soluções.

  Reduzir a diversidade de perturbações e adotar movimentos mais suaves foi uma tentativa de controlar a exploração, mas a escolha de movimentos pequenos e suaves também leva ao problema da lentidão na convergência para uma solução satisfatória. Isso implica em uma busca local muito restrita, onde pode ser difícil escapar de ótimos locais ou mínimos locais subótimos.

- **Tamanho do problema e tempo de execução:** Outra dificuldade enfrentada na implementação da metaheurística foi a definição da quantidade ideal de iterações para alcançar bons resultados. O ajuste desse parâmetro exigiu um equilíbrio delicado, pois uma busca muito curta poderia não ser suficiente para encontrar soluções de qualidade, enquanto uma busca prolongada poderia aumentar significativamente o tempo de execução sem garantir melhorias expressivas. Além disso, a geração da solução inicial se mostrou um ponto crítico, pois uma escolha inadequada poderia comprometer o desempenho do algoritmo, resultando em tempos de execução elevados e soluções insatisfatórias.

  Outro fator que impactou o processo foi a convergência lenta das penalidades, que dificultou a obtenção de melhorias rápidas. Um algoritmo muito conservador poderia progredir de forma excessivamente gradual, enquanto um número elevado de iterações poderia consumir tempo sem garantir avanços significativos.

- **Estratégias de ajuste dos parâmetros:** Durante a implementação da metaheurística, outro desafio enfrentado foi o ajuste dos parâmetros. A solução final depende diretamente de fatores como o número de iterações, a quantidade de perturbações e o tipo de movimentos aplicados. No entanto, definir esses valores de forma adequada se mostrou uma tarefa complexa, exigindo múltiplas tentativas e refinamentos. O processo de ajuste fino demandou experimentação contínua para equilibrar a exploração do espaço de soluções e o tempo de execução, buscando garantir um desempenho satisfatório do algoritmo.
- **Resultados subótimos e a busca por melhores resultados:** Um outro ponto importante foi a dificuldade em atingir o resultado esperado. A metaheurística é uma ferramenta muito boa para encontrar soluções boas o suficiente dentro de um tempo razoável, mas não há garantias de que ela encontrará a solução ótima. O algoritmo pode estar preso em um ótimo local, ou pode não ter explorado todas as possibilidades relevantes. Isso significa que, mesmo com ajustes, o algoritmo pode não alcançar o resultado ótimo sem mais explorações no espaço de soluções.

6. Conclusão

Durante a execução deste trabalho, enfrentamos diversas dificuldades que proporcionaram importantes aprendizados. Inicialmente, a definição do modelo e suas restrições foi um ponto crítico. Garantir a correta separação entre os aviões e calcular as penalidades corretamente demandou uma abordagem cautelosa, particularmente ao incorporar as restrições baseadas na matriz S.

O desempenho do solver GLPK se mostrou insuficiente para todas as instâncias, o que nos levou a explorar alternativas, como o solver CBC. Essa mudança possibilitou a obtenção de resultados em tempos mais satisfatórios.

A implementação da metaheurística apresentou uma série de desafios relacionados ao equilíbrio entre diversidade e exploração local. Experimentamos diferentes tipos e frequências de perturbações, enfrentando dificuldades tanto com movimentos muito grandes que afastavam o algoritmo de regiões promissoras, quanto com movimentos pequenos que resultavam em lentidão na convergência. Apesar dos ajustes realizados, os valores obtidos com a metaheurística não foram satisfatórios, ficando aquém das expectativas em termos de qualidade das soluções e tempos de execução.

Além disso, outro desafio importante foi o ajuste de parâmetros. Definir o número ideal de iterações, a quantidade de perturbações e os tipos de movimentos demandou uma experimentação contínua. Mesmo com múltiplas tentativas e refinamentos, a convergência para soluções de qualidade permaneceu um ponto crítico, evidenciando a necessidade de futuras melhorias no modelo da metaheurística.

Em conclusão, os resultados obtidos destacaram as limitações da metaheurística aplicada, evidenciando a importância de novas estratégias para explorar o espaço de soluções de forma mais eficiente. O trabalho, entretanto, proporcionou uma experiência valiosa em modelagem, otimização e ajuste de metaheurísticas, reforçando o aprendizado sobre os desafios práticos na resolução de problemas complexos.

7. Bibliografia

SUMIKA, Fernanda. ***Minicurso I: Introdução às Metaheurísticas****. Apresentação de slides. Minicurso realizado em 5 e 6 de abril de 2024. Docente da UFOP.*