Bruno Coswig Fiss
Kauê Soares da Silveira

GRASP aplicado ao problema de aterrissagem de aviões

INF05010 – Otimização Combinatória

Professor: Marcus Rolf Peter Ritt

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

1º de julho de 2010

Sumário

1 Descrição do problema

1.1 Descrição formal

O problema de aterrissagem de aviões consiste em definir um momento no tempo para a aterrissagem de cada avião $i \in P$, sendo P o conjunto de aviões. Cada avião possui os seguintes dados:

E_i: Momento mais prematuro em que o avião i pode realizar pouso.

T_i: Momento ideal para pouso do avião i.

L_i: Momento mais tardio em que o avião i pode realizar pouso.

g_i: Penalidade por unidade de tempo da diferença do pouso para o tempo ideal se o avião chegar mais cedo do que o ideal.

h_i: Penalidade por unidade de tempo da diferença do pouso para o tempo ideal se o avião chegar mais tarde do que o ideal.

S_{ij}: Distância de tempo requerida após o pouso do avião i para que o avião j possa pousar.

O objetivo é encontrar uma solução com somatório de todas as penalidades mais baixo possivel.

1.2 Formulação como um programa inteiro

O seguinte programa inteiro descreve o problema acima descrito:

$$
\begin{aligned}
\textbf{minimiza} \quad & \sum_{i \in P} g_i \alpha_i + h_i \beta_i & \\
\textbf{sujeito a} \quad & x_i = -\alpha_i + \beta_i + T_i, & \forall i \in P \\
& E_i \leq x_i \leq L_i, & \forall i \in P \\
& x_j - x_i \geq S_{ij}\delta_{ij} + (E_j - L_i)\delta_{ji}, & \forall i, j \in P \\
& \delta_{ij} + \delta_{ji} = 1, & \forall i, j \in P \\
& x_i \geq 0, x_i \in \mathbb{R}, & \forall i \in P \\
& \alpha_i \geq 0, \alpha_i \in \mathbb{R}, & \forall i \in P \\
& \beta_i \geq 0, \beta_i \in \mathbb{R}, & \forall i \in P \\
& \delta_{ij} \in \mathbb{B}, & \forall i, j \in P
\end{aligned}
$$

A variável x_i representa o momento da aterrissagem do avião i. As variáveis α_i e β_i indicam a diferença para menos ou mais, respectivamente, do momento da aterrissagem para o momento ideal de aterrissagem do avião i. δ_{ij} é uma variável binária que indica se o avião i aterrisa antes do avião j.

A formulação do programa inteiro foi realizada pelos autores antes da leitura de artigos relacionados com o problema, que por sua vez continham uma formulção similar. Para facilitar a compreensão, tornamos a simbologia utilizada por nós semelhante à utilizada nesses artigos [1, 2].

1.3 Representação de um solução

O programa inteiro que descreve o problema aqui tratado é misto, possuindo variáveis reais além de inteiras. Como o objetivo desse trabalho é aplicar uma meta-heurística a um problema inteiro, separamos a solução em duas partes, uma inteira e outra real.

A parte inteira da solução é a que contém as variáveis δ_{ij}, que descrevem, em conjunto, a ordem de chegada dos aviões. O caminho inverso também é válido, ou seja, uma determinada ordem de chegada dos aviões descreve completamente as variáveis δ_{ij}. Já a parte linear são as demais variáveis.

Portanto, representaremos uma solução como uma sequência de aviões que descreve a ordem de chegada desses. Após obter a solução inteira do problema, basta resolver o programa linear restante para obter a solução completa do problema.

2 *Algoritmo proposto*

2.1 Idéia geral

Nosso algoritmo utiliza a biblioteca GLPK (GNU linear programming kit) para resolução das partes lineares do problema sendo tratado.

Primeiramente, definimos o problema inteiro utilizando os dados lidos da entrada e as funções para criação de restrições disponibilizadas pela biblioteca GLPK. Após esse passo é possível utilizar o resolvedor interno do GLPK para resolver o problema inteiro, utilizando o método branch-and-cut. Essa característica foi requisitada na especificação do trabalho da disciplina, e também permite a comparação entre as soluções atingidas pelo algoritmo do GLPK e pelo método GRASP, além da comparação entre o desempenho na obtenção dessas.

Após isso, utilizamos a meta-heurística GRASP para criar soluções para a parte inteira do problema, ou seja, sequências de aviões que definem a ordem de chegada desses. Para cada solução inteira, modificamos as restrições do problema inteiro de forma que ele respeite a ordem presente na sequência e torne-se, por consequência, linear. O programa linear é então resolvido através do método simplex utilizando uma rotina da biblioteca GLPK. Isso é feito para cada solução inteira gerada pelo método GRASP, tanto para as soluções geradas durante a construção gulosa aleatória quanto para as soluções geradas na busca local (soluções vizinhas da solução atual). A melhor solução encontrada é armazenada e impressa ao fim do algoritmo.

2.2 GRASP

2.2.1 Criação de soluções

[3].

2.2.2 Vizinhança e busca local

3 Experimentos

3.1 Configurações

3.2 Resultados

3.3 Análise

Referências Bibliográficas

[1] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha, and D. Abramson. Scheduling aircraft landings–the static case. *Transportation Science*, 34(2):180–197, 2000.

[2] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha, and D. Abramson. Displacement problem and dynamically scheduling aircraft landings. *Transportation Science*, 55(1):54–64, 2004.

[3] Mauricio G. C. Resende, Mauricio G. C, Resende, and Celso C. Ribeiro. Greedy randomized adaptive search procedures, 2002.