Inteligencia Artificial

Informe Final: Aircraft Landing Scheduling Problem

Victor Gonzalez Rodriguez

13 de agosto de 2013

Evaluación

Mejoras 1ra Entrega (10 %):	_____
Código Fuente (10 %):	_____
Representación (15 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100):	_____

Resumen

El Aircraft Landing Scheduling Problem, es un problema que se presenta en todos los aeropuertos del mundo, y a medida que la cantidad de aviones y el tráfico aumenta, este problema se vuelve cada vez mas crítico y necesario. En la actualidad existe gran cantidad de articulos científicos que abordan esta problemática, cada una con un enfoque distinto y para casos específicos, como los es para la calendarización con solo una pista de aterrizaje o input dinámico. Lo cierto, es que la mayor parte de estas implementaciones son dificiles de comparar entre ellas debido a la gran diferencia de enfoques, pero si entregan soluciones aceptables.

En el presente documento, abarcaremos el uso de la técnica de backtracking basado en grafo de restricciones (BT-GBJ), el cual si bien no es el más óptimo para este tipo de casos, si es buen candidato a la sencillez de implementación.

1. Introducción

El propósito de este informe, es conocer, entender y ver una implementación del problema de calendarización de aterrizajes. Analizaremos el estado del arte sobre este tema, analizando cada uno de los enfoques y algoritmos que son más utilizados en la actualidad, para luego entregar una presentación más formal del problema, mediante un modelo matemático, su representación mediante la implementación de un algoritmo y sus respectivos casos prácticos.

El Aircraft Landing Scheduling Problem (ALSP), se puede enfrentar como un problema combinatorial, donde se debe encontrar el mejor tiempo de aterrizaje para una lista de aviones con restricciones específicas. Esto se puede atacar mediante distintos algoritmos, ya sea mediante heurísticas, o métodos de búsqueda completa, como el que veremos en este informe.

Buscamos mediante esto, entregar una visión generalizada del Aircraft Landing Scheduling Problem, pero analizada en profundidad mediante la implementación usando el algoritmo BT-GBJ.

2. Definición del Problema

El Aircraft Landing Scheduling Problem, es un problema que busca elegir un tiempo de aterrizaje para distintos aviones con el menor costo posible.

En la realidad, si un avión llega atrasado, genera molestias a los pasajeros y genera gastos que deben ser evitados. Por otro lado, si un avión se adelanta a su hora programada de aterrizaje, también genera gastos, porque puede obligar a adelantar procesos que pueden ser costosos.

Junto a todo esto, se debe sumar, el hecho de que entre aterrizaje y aterrizaje, debe existir una separación de tiempo, que permita que la pista quede despejada y esté lista para recibir al siguiente avión. Esto varía entre tipos de naves, ya que aviones más grandes generan más tiempo de uso de la pista, mientras que naves más pequeñas tienden a ser más agiles. Además considerando la turbulencia residual que dejan los aviones entre aterrizajes, sobretodo cuando se realizan aterrizajes paralelos en distintas pistas.

3. Estado del Arte

El ALSP es tan viejo como lo son los aeropuertos en el mundo. El problema surge cuando se debe decidir el tiempo de aterrizaje entre distintos aviones considerando un listado de restricciones de tiempo y de costo para cada aterrizaje.

Para resolver este problema se han desarrollado distintas implementaciones, las cuales en su mayoría son híbridos entre heurísticas y métodos de busqueda completa, ya que es ampliamente conocido, que mezclar lo mejor de esos 2 mundos es el mejor camino.

Es dificil decir a ciencia cierta cuales serían los mejor algoritmos, porque casi no existen puntos de comparación debido a que cada una de las implementaciones utiliza una instancia distinta del ALSP, por ejemplo, resolver el problema para aterrizajes en una sola pista o considerando restricciones entre aterrizajes paralelos, aunque la tónica común es la utilización de técnicas híbridas.

Uno de los métodos que se utilizan es el Branch-and-Price [1], el cual en la implementación, se mejora con una heurística previa para así reducir el espacio de búsqueda de una solución. El caso que se utiliza para este método es el de múltiples pistas de aterrizaje, con separacion entre aterrizajes ignorando aterrizajes paralelos.

Otro de los métodos es el uso de busquedas del camino más corto [2], el cual usa una versión modificada de ese algoritmo para encontrar soluciones bajo la restricción

específica de un máximo reprogramaciones de aterrizajes. Este es quizas el que más se asemeja al método que se empleará en este informe, ya que hace uso de *fuerza bruta* para encontrar una solución.

4. Modelo Matemático

Esta sección presenta una formulación presentada por J. Beasley [3] Dado un set de aviones P, cada avión i, tiene una ventana predeterminada de aterrizaje $[bef_i, last_i]$, y además un tiempo objetivo $target_i(bef_i \leq target_i \leq last_i)$ donde el avión aterriza con costo 0. sep_{ij} es la separación requerida entre el avión i y j al aterrizar en una misma pista. p_bef_i y p_last_i denotan los costos unitarios para el avión i al adelantarse o atrasarse respectivamente.

Las variables de decisión son:

$x_i =$ el tiempo de aterrizaje para el avión i
$\alpha_i =$ que tan temprano el avion i aterriza antes de $target_i$
$\beta_i =$ que tan tarde el avión i aterriza despues de $target_i$
$\delta_i = \begin{cases} 1 \text{ si el avión } i \text{ aterriza antes que } j \\ 0 \text{ si no} \end{cases}$
$z_{ij} = \begin{cases} 1 \text{ si } i \text{ y } j \text{ aterrizan en la misma pista} \\ 0 \text{ si no} \end{cases}$
$y_{jr} = \begin{cases} 1 \text{ si avión } j \text{ aterriza la pista } r \\ 0 \text{ si no} \end{cases}$

El problema es determinar el tiempo de aterrizaje x y asignar la variable y para avión que entregue el mínimo costo satisfaciendo lo siguiente:

- cada avión aterriza en su respectiva ventana de tiempo

$$x_i \in [bef_i, last_i] \ \forall i \in P;$$

- criterio de separación entre el aterrizaje de un avión y todos los aviones sucesivos en la misma pista. Eso es, si $\delta_{ij} = 1$ y $z_{ij} = 1$, entonces se tiene que:

$$x_i \geq x_i + sepij \ \forall i, j \in P; i \neq j$$

5. Representación

La representación matemática es la siguiente:

$$
\begin{aligned}
\min \quad & \textstyle\sum_{i=1}^{P} (p\_bef_i \alpha_i + p\_last_i \beta_i) \\
\text{s.t.} \quad & bef_i \leq x_i \leq last_i \\
& \delta_{ij} + \delta ji = 1 \\
& \textstyle\sum_{r=1}^{R} y_{ir} \\
& z_{ij} = z_{ji} \\
& z_{ij} \geq y_{ir} + y_{jr} - 1 \\
& x_j \geq x_i + sep_{ij} z_{ij} - (last_i + sep_{ij} - bef_j)\delta_{ji} \\
& \alpha_i \geq target_i - x_i \\
& 0 \leq \alpha_i \leq target_i - bef_i \\
& \beta_i \geq x_i - target_i \\
& 0 \leq \beta_i \leq bef_i - target_i \\
& x_i = target - \alpha_i + \beta_i \\
& x_i, \alpha_i, \beta_i \geq 0 \\
& \delta_{ij}, y_{ij}, z_{ij} \text{ binarios}
\end{aligned}
$$

La función objetivo es minimizar la suma de los costos de las desviaciones de los tiempos objetivo.

En la implementación del problema, se utilizó una clase llamada *Airplane*, la cual almacena los datos críticos de un avión, como el bef_i, $last_i$, $target_i$, sep_i (notar que es solo la lista de separaciones para el avión i en vez de sep_{ij}), p_bef_i y p_last_i.

Todas las demás estructuras, son específicas al algorítmo, por lo que no aportan a la representación, aunque muchas de las restricciones que aparecen en la representación matemática fueron incluidas explicitamente en la implementación.

Supongamos que el tiempo más temprano que puede aterrizar un avión es b, y que el tiempo más tarde que puede aterrizar un avión es l, sea N la cantidad de aviones, y sea R la cantidad de pistas en el problema. Luego el espacio de búsqueda es aproximadamente $N^{b-l}2^R$

6. Descripción del algoritmo

El algoritmo de Backtracking (BT) es una técnica de búsqueda completa. Este algoritmo en particular tiene 2 variantes: Grap-based Jumping (GBJ) y Conflict-base Jumping (CBJ).

El BT-GBJ hace uso del grafo de restricciones de un problema, para hacer saltos dirigidos al nodo padre de una variable conflictiva, a diferencia de el BT simple, el cual solo se devuelve un nodo cronológicamente. Para la implementación, el grafo se representa mediante una matriz de NxN, donde N es la cantidad de aviones del problema.

Como no existe documentación apropiada para elegir los criterios para generar el grafo de restricciones, se procedió a crearlos bajo el criterio que si la ventana $[bef_i, last_i + sep_{ij}]$ comparte elementos de dominio con $[bef_j, last_j + sep[ji]]$, entonces se tratan de nodos conflictivos, y se marcan dentro del grafo de restricciones.

La implementación genérica del GBJ no recursivo[4] es la siguiente:

```
procedure graph-based-backjumping
Input: A constraint network P = (X; D; C). Output: Either a solution, or a decision that t
    compute anc(xi) for each xi
    i <- 1 (initialize variable counter)
    D'i <- Di (copy domain)
    Ii <- anc(xi) (initialize induced ancestor set)
    while 1 <= i <= n
        instantiate xi <- selectValue
        if xi is null (no value was returned)
            iprev <- i
            i latest in Ii (backjump)
            Ii <- Ii u Iiprev - {xi} (merge to update Ii)
        else
            i <- i + 1
            D'i <- Di
            Ii <- anc(xi)
    end while
    if i = 0
        return "inconsistent"
    else
        return instantiated values of {x1,...,xn}
end procedure

subprocedure selectValue (same as backtracking's)
    while D'i is not empty
        select an arbitrary element a in D'i, and remove a from D'i
        if consistent(a{i-1}; xi = a)
            return a
    end while
    return null (no consistent value)
end procedure
```

En el subprocedimiento del selectValue, se desarrolló una implementación que recibe un avión específico, se navega por todos los valores del dominio de aterrizaje (como mejora se introdujo entregar primero los que tienen menor costo), para luego ir probando para cada una de las pistas de aterrizajes, si hay una combinación de tiempo/pista que sea consistente. Si es asi, las instanciaciones de los valores se guardan en una matriz de NxR donde N es la cantidad de aviones y R es la cantidad de pistas.

Para el algoritmo general del GBJ, el trabajo de los padres se mejoró utilizando una lista de largo N, la cual indica cual es el padre mas cercano que apunta a ese nodo; esto es lo que se menciona en el seudocódigo como anc(x1), el cual entrega el padre del avion xi.

El resto del algoritmo permaneció generalmente intacto, para asi preservar la simplicidad de lectura.

La mayoría de las restricciones blandas, se implementaron dentro de la clase Air-

plane, la cual verifica si hay colisiones, si los tiempos instanciados son correctos, etc.

7. Experimentos

Lamentablemente el programa por alguna razón desconocida, nunca entrega la solución óptima, sino mas bien una donde utiliza todo el dominio de la primera variable, y a partir de eso, la solución de menor costo. Es decir:

Avión	aterrizaje	pista	Comentario
1	559	2	← posibilidad más alta de instanciación
2	195	1	
3	89	1	
4	105	1	
5	121	2	
6	120	1	
7	136	2	
8	560	1	
9	135	1	
10	160	1	

A pesar de los esfuerzos, no se logró arreglar el problema a nivel de código.

8. Resultados

No Aplica

9. Conclusiones

A partir de todo lo que se ha visto sobre el ALSP, hemos visto que uno de sus tanto acercamientos también puede ser el de un problema combinatorial, mediante el uso de BT. Si bien el método no es rápido ni eficiente, si ayuda a entregar un resultado que será completamente el mejor, debido a que su naturaleza lo obliga a navegar todas las posibilidades.

El ALSP es un problema complejo de resolver, sobretodo considerando las variables que se tienen que tomar en cuenta para que el problema sea resuelto eficientemente.

10. Bibliografía

[1] Min Wen, *Algorithms of Scheduling Aircraft Landing Problem*, Department of Informatics and Mathematical Modelling Technical University of Denmark, 2005, 45–48, `http://etd.dtu.dk/thesis/185882/imm4124.pdf`

[2] Hamsa Balakrishnan, Bala Chandran, *Scheduled Aircraft Landings Under Constrained Position Shifting*, MIT, `http://www.mit.edu/~hamsa/pubs/BalakrishnanChandranGNC06.pdf`

[3] Scheduling aircraft landings - the static case (with M. Krishnamoorthy, Y.M. Sharaiha and D. Abramson) Transportation Science, vol. 34, 2000, pp180-197

[4] Rina Dechter and Daniel Frost, *Backjump-based Backtracking for Constraint Satisfaction Problems*, Department of Information and Computer Science, University of California, Irvine, USA, 2001,16–18, `http://www.ics.uci.edu/~csp/R56.pdf`