Projet de programmation par contraintes: Séquencement des vols à l'arrivée

François Fages
email: Francois.Fages@inria.fr
http://contraintes.inria.fr/~fages/Teaching/

Module C2-4-1 Programmation par Contraintes
Master Parisien de Recherche en Informatique
6 Octobre 2009

Le but de ce projet de programmation est de modéliser et résoudre un problème combinatoire en programmation logique avec contraintes. Le sujet porte sur la recherche de stratégies optimales de séquencement des vols à l'arrivée des aéroports, qui peuvent être plus performantes que celles utilisées par les contrôleurs aériens. La méthode consiste à trouver une séquence qui minimise le temps total d'atterrissage des appareils, en associant la propagation des contraintes procédurales d'approche à l'exploration d'un arbre de recherche.

Ce projet de programmation comptera pour 30% de la note du module. Il est à remettre le 17 novembre 2009 par courrier électronique sous la forme

1. d'un programme GNU-Prolog (`http://www.gprolog.org/` à l'exclusion de tout autre langage pour des raisons de facilité de correction)
2. d'un jeu de tests (à deux ou trois couloirs aériens, et 15 à 30 avions)
3. d'un rapport succinct décrivant le format de représentation des données, le prédicat principal à appeler et éventuellement des choix particuliers de stratégies de résolution et leur évaluation.

1 Le problème du séquencement des vols

La gestion du trafic aérien dans la zone terminale, comprenant les vols à 30-40 minutes de la piste d'atterrissage, est une tâche critique lors des périodes de pointe. Sur un aéroport comme Orly, la régulation du trafic à l'arrivée peut conduire à retarder le décollage des vols nationaux vers Orly, mais la régulation au départ ne s'appliquant pas aux vols long-courrier, les problèmes les plus ardus peuvent se poser sur des aéroports comme Sydney où il arrive parfois que des nuages d'avions se présentent à l'arrivée.

La zone terminale d'un aéroport contient les différents couloirs d'approche de la piste, et est découpée en trois zones:

1. la zone critique (5 min. de la piste)
2. la zone régulée (5 à 20 min. de la piste)
3. la troisième zone (20 à 40 min. de la piste)

Dans la zone critique aucune permutation des vols n'est possible, dans la zone régulée les dépassements entre vols se trouvant dans des couloirs différents sont

possibles, dans la troisième zone tous les dépassements sont envisageables y compris à l'intérieur d'un couloir aérien.

Les changements de position sont réalisés en donnant des ordres au pilote qui peut modifier la vitesse de son appareil ou sa trajectoire (élargissement ou rétrécissement des courbes, ronds), mais nous ne traiterons pas directement ces aspects.

Chaque vol a une date prévue d'arrivée, autour de laquelle doit se situer la date réelle d'arrivée, dans une fourchette déterminée pas un facteur de retard (fixé par défaut à 3) et un facteur d'avance (0.8 par défaut).

Chaque vol se présente dans un couloir aérien, qu'il ne peut pas quitter. Les distances à respecter entre les vols dépendent des caractéristiques des appareils. On distinguera trois types d'appareils A (gros), B (moyens), C (petits), et la matrice des distances de sécurité (séparation temporelle entre les appareils exprimée en secondes) $\{d_{i,j}\}$ suivante:

$i \backslash j$	A	B	C
A	100	130	160
B	70	80	100
C	60	60	70

Dans un souci de simplification, on se contentera d'appliquer ces contraintes de distances à l'atterrissage, c'est-à-dire à la date calculée d'arrivée, et non dans les points intermédiaires.

La stratégie des contrôleurs aériens consiste essentiellement a conserver l'ordre prévu d'arrivée en cumulant les retards dus aux contraintes de distance de sécurité. Or la matrice des distances n'étant pas symétrique, il est possible de minimiser le temps global d'arrivée des vols de la séquence en effectuant des permutations. La recherche d'une solution de coût minimum devient alors un problème NP-difficile.

La figure 1 montre une copie d'écran d'un système de calcul des séquences d'arrivée optimales. La piste est à gauche, les trois zones sont délimitées par des teintes différentes, les trois couloirs aériens matérialisés par les bandes horizontales. Dans la partie inférieure de l'écran sont visualisées les temps prévus d'arrivée ainsi que la solution issue de la stratégie préservant l'ordre d'arrivée (28.67 min.). Dans la partie supérieure, une solution de coût minimum (26.5 min.) est visualisée, celle-ci consiste à permuter les vols 6 et 7-8, 12 et 13, 14 et 15. L'optimalité est prouvée par le système.

2 Contraintes d'ordonnancement

Le problème du séquencement des vols peut être modélisé comme un problème d'ordonnancement de tâches. Étant données les dates prévues d'arrivée $tp_i \in N$ de chaque vol, pour $1 \leq i \leq n$, on cherche à déterminer les dates réelles d'arrivée des vols, $t_i \in [0.8 * tp_i, 3 * tp_i]$, qui minimisent le temps d'arrivée global de la séquence (modélisé par une tâche fictive fin avec pour tout $1 \leq i \leq n$, $t_i \leq fin$).

Les contraintes de précédence entre deux vols i, j s'expriment en GNU-Prolog par une simple contrainte d'inégalité

$$t_i + d_{i,j} \leq t_j$$

Ces contraintes sont posées d'une part entre les vols i de la zone critique ($tp_i < 300$) et les vols j avec $tp_j > tp_i$, et d'autre part entre les vols i de la zone régulée ($300 \leq tp_i < 1200$) et les vols j se trouvant dans le même couloir aérien avec $tp_i < tp_j$.

Les possibilités de permutation sont exprimées par des contraintes disjonctives

$$t_i + d_{i,j} \leq t_j \vee t_j + d_{j,i} \leq t_i$$

posées entre les autres paires de vols.

3 Espace de recherche et procédure d'optimisation

Les contraintes disjonctives sont testées par des points de choix qui constituent l'espace de recherche sous la forme d'un arbre binaire.

Les contraintes de précédence $t_i + d_{i,j} \leq t_j$ sont utilisées de façon active pour réduire le domaine des inconnues (initialement $t_i \in [0.8 * tp_i, 3 * tp_i]$), à chaque fois que la borne inférieure $\underline{t_j}$ (resp. supérieure $\overline{t_i}$) d'une des variables de l'inégalité change. Une impossibilité est détectée lorsque le domaine d'une variable devient vide. Dans ce cas l'exploration de la branche correspondante de l'arbre de recherche s'arrête immédiatement grâce à la propagation des contraintes, et la recherche continue en explorant une autre alternative en restaurant le domaine des variables du dernier point de choix ("backtracking").

S'agissant d'un problme d'optimisation (minimisation de la date d'arrive du dernier avion) on utilisera le prdicat `fd_minimize` du GNU-Prolog pour itrer la recherche de solutions jusqu' l'obtention d'une preuve d'optimalité.

Plusieurs heuristiques pourront éventuellement être essayées pour améliorer les performances (choix des contraintes disjonctives à considérer en premier, en particulier celles devenues déterministes, choix des alternatives à essayer en premier, propagation de contraintes redondantes).

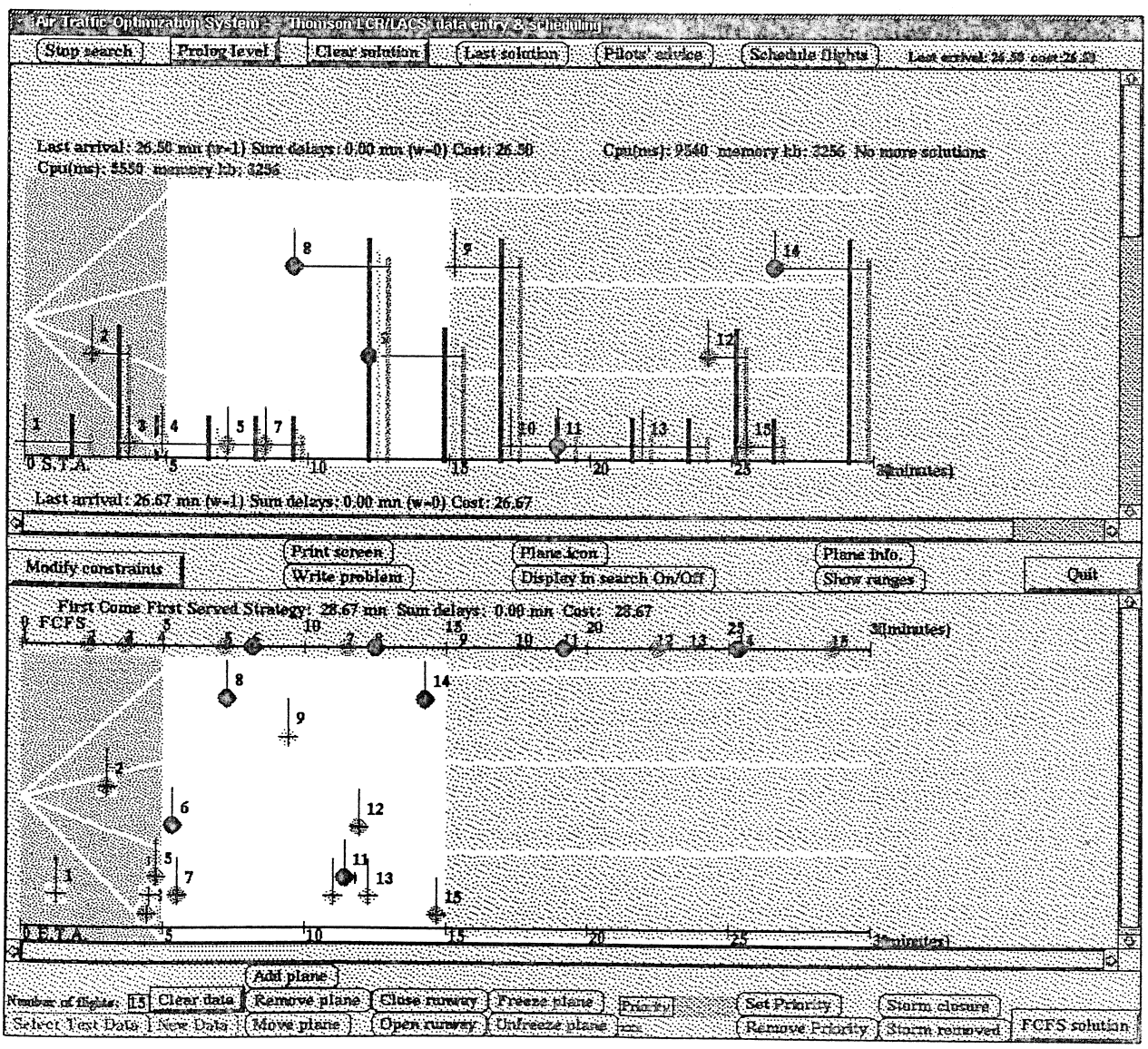

Fig. 1. Copie d'écran d'un système de séquencement des vols sur un exemple comprenant trois couloirs aériens et 15 avions (en haut séquence d'arrivée prouvée optimale, en bas séquence prévue d'arrivée et solution préservant l'ordre prévu des arrivées.