An exact algorithm for Aircraft Landing Problem

Min Wen Jesper Larsen Jens Clausen
Informatics and Mathematical Modelling
Technical University of Denmark
2800 Kgs. Lyngby
Denmark

September 4, 2005

Abstract

This paper addresses the problem of scheduling aircraft landings at an airport. Given a set of planes and runways, the objective is to minimize the total (weighted) deviation from the target landing time for each plane. There are costs associated with landing either earlier or later than a target landing time for each plane. Each plane has to land on one of the runways within its predetermined time windows such that separation criteria between all pairs of planes are satisfied. This type of problem is a large-scale optimization problem, which occurs at busy airports where making optimal use of the bottleneck resource (the runways) is crucial to keep the airport operating smoothly.

This paper is the first attempt to develop a column generation-based exact decomposition algorithm for the Aircraft Landing Problem (ALP). We formulate the problem as an mixed integer program, then reformulate it, using Dantzig-Wolfe decomposition, as a set partitioning problem with side constraints. We also present a mixed integer formulation for the subproblem to generate the columns with negative reduce cost. Based on the set partitioning formulation, a branch-and-bound algorithm is developed to obtain the exact solution for the problem. Computational results are presented for publicly available test problems involving up to 50 aircrafts and 4 runways. The initial implementation shows that optimal solutions can be produced with less than 500 columns generated in acceptable CPU time.

Keywords: Aircraft Landing Problem, Branch-and-Price.

1 Introduction

In this paper, we consider the problem of scheduling aircraft landings at an airport. Given a set of planes and runways, the objective is to minimize the total (weighted) deviation from the target landing time for each plane. There are costs associated with landing either earlier or later than a target landing time for each plane. Each plane has to land on one of the runways within its predetermined time windows such that

separation criteria between all pairs of planes are satisfied. This type of problem is a large-scale optimization problem, which occurs at busy airports where making optimal use of the bottleneck resource (the runways) is crucial to keep the airport operating smoothly.

Upon entering within the radar range (or horizon) of an air traffic control (ATC) at an airport a plane requires ATC to assign it a landing time and also a runway if more than one runway is in use. The landing time must lie within a predetermined time windows, bounded by an earliest landing time and a latest landing time. The time windows are different for different planes. The earliest time represents the time required if a plane flies at its maximum airspeed. The latest time corresponds to the landing time of a plane flying at its most fuel efficient airspeed while holding (circling) for the maximum allowable time.

Each plane also has a most economical, preferred speed, referred to as the cruise speed. The preferred or target time of a plane is the time it would land if it is required to fly at cruise speed. If ATC requires the plane to either slow down, hold or speed up, a cost will be incurred.

Furthermore, planes have to keep separation distance to avoid turbulences caused by preceding planes. Thus, there are certain minimum constraints on the separation delay between aircrafts of different types, e.g. a Boeing 747 can handle (and generates) more turbulence than a Hawker 700.

Beasley, Krishnamoorthy, Sharaiha and Abramson (2000) presented a mixed integer formulation of the ALP. After relaxing binary variables and strengthening the formulation with additional constraints the problem is solved optimally with a linear programming based tree search algorithm. Jung and Laguna (2003) proposed a heuristic algorithm based on the segmentation of time. The time horizon is divided into time segments that determine subproblems of ALP. Cheng, Crawford and Menon (1999) addressed four genetic schemes for solving the multiple runway ALP. Ernst, Krishnamoorthy and Storer (1999) solved the multiple runway ALP through a heuristic approach involving a genetic algorithm to search a perturbation space, and an exact approach which is implememted as a branch-and-bound algorithm. Pinol and Beasley (2004) first applied the scatter search and bionomic algorithm for the multiple runway ALP. Bianco, Ricciardelli, Rinaldi and Sassano (1988) adopt a job-shop scheduling view of the ALP. The ALP may also be viewed as an open traveling salesman problem with time window when only one runway is considered (Bianco *et al.* 1993).

In this research, we investigate the multiple runway case. We use column generation combined with branch-and-bound (so called branch-and-price) to solve the problem. The contribution of this paper is that our algorithm is the first attempt to develop a branch-and-price algorithm for the problem.The remainder of the paper is structured as follows: In section 2, we give the problem description and the mathematical formulation for the ALP. In section 3, we reformulate it as a set partitioning formulation with side constraints. The mathematical model for the subproblem is also given. The branch-and-price algorithm is developed in section 4. In section 5, some implementation details are given. Computational results are reported in section 6. Conclusion are given in Section 7.

2 The Aircraft Landing Problem

This section presents a mixed integer formulation of the static multiple runway Aircraft Landing Problem that is based on the formulaiton presented in Beasley *et al.* (2000). Given a set of planes P, each plane i has a predetermined landing time windows $[E_i, L_i]$, and also, a target time T_i $(E_i \leq T_i \leq L_i)$ at which time the plane is landed with cost 0. S_{ij} is the required seperation time between plane i and j (where i lands before j) for landing them on the same runway. As custom in the multiple runway case, we assume the seperation time between two planes on the different runways is 0. g_i and h_i denote the unit costs for plane i landing earlier and later than the target time respectively. Figure 1 depicts this variation in cost within the time windows of a plane i.

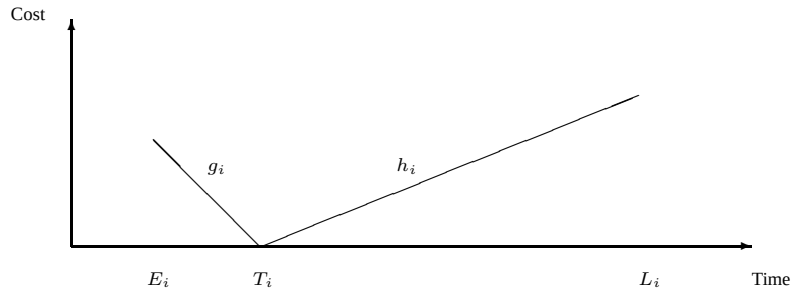

Figure 1: Variation in cost for a plane within its time window

We use the decision variables:

$$
\begin{aligned}
x_i &= \text{the landing time for plane } i \ (i \in P); \\
\alpha_i &= \text{how soon plane } i \text{ lands before } T_i \ (i \in P) \\
\beta_i &= \text{how late plane } i \text{ lands after } T_i \\
\delta_{ij} &= \begin{cases} 1 & \text{if plane } i \text{ lands before } j \ (i, j \in P; i \neq j); \\ 0 & \text{otherwise} \end{cases} \\
z_{ij} &= \begin{cases} 1 & \text{if } i \text{ and } j \text{ land on the same runway } \ (i, j \in P; i \neq j); \\ 0 & \text{otherwise} \end{cases} \\
y_{jr} &= \begin{cases} 1 & \text{if plane } j \text{ lands on runway } r \ (j \in P; r \in R); \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

The mathematical formulation can now be written as:

MIP:

$$\min \quad \sum_{i=1}^{P}(g_i\alpha_i + h_i\beta_i) \tag{1}$$

$$\text{s.t.} \quad E_i \le x_i \le L_i \qquad \forall i \in P; i \neq j \tag{2}$$

$$\delta_{ij} + \delta_{ji} \le 1 \qquad \forall i, j \in P \tag{3}$$

$$\sum_{r=1}^{R} y_{ir} = 1 \qquad \forall i \in P; r \in R \tag{4}$$

$$z_{ij} = z_{ji} \qquad \forall i, j \in P; i \neq j \tag{5}$$

$$z_{ij} \ge y_{ir} + y_{jr} - 1 \qquad \forall i, j \in P; i \neq j \tag{6}$$

$$x_j \ge x_i + S_{ij}z_{ij} + (L_i + S_{ij} - E_j)\delta_{ji} \forall i, j \in P; i \neq j \tag{7}$$

$$\alpha_i \ge T_i - x_i \qquad \forall i \in P \tag{8}$$

$$0 \le \alpha_i \le T_i - E_i \qquad \forall i \in P \tag{9}$$

$$\beta_i \ge x_i - T_i \qquad \forall i \in P \tag{10}$$

$$0 \le \beta_i \le L_i - T_i \qquad \forall i \in P \tag{11}$$

$$x_i = T_i - \alpha_i + \beta_i \qquad \forall i \in P \tag{12}$$

$$\delta_{ij}, y_{ij}, z_{ij} \text{ binary} \qquad \forall i, j \in P; i \neq j \tag{13}$$

The objective function (Eq.1) minimize the sum of the costs of deviation from the target times (T_i). Constraints (Eq.2) ensure that each plane lands within its time windows. Constraints (Eq.3) indicate that either plane i must land before plane j ($\delta_{ij} = 1$) or plane j must land before plane i ($\delta_{ji} = 1$). Constraints (Eq.4) ensure that each plane lands on exactly one runway whereas constraints (Eq.5) are symmetry constraints (if i and j land on the same runway so do j and i). Constraints (Eq.6) ensure that, if there is any runway r on which plane i and j are both landed, then we force z_{ij} to be 1 (i and j land on the same runway). If $z_{ij} = 0$, then (Eq.6) ensure that planes i and j can not land on the same runway. Constraints (Eq.7) are the separation constraints. Constraints (Eq.8) and (Eq.9) ensure that α_i is at least as big as zero and the time difference between T_i and x_i, and at most the time difference between T_i and E_i. Constraints (Eq.10) and (Eq.11) are similar equations for β_i. Constraits (Eq.29) relate the landing time (x_i) to the time i lands before (α_i), or after (β_i), target (T_i).

The linear programming relaxation (denoted as **LMIP**) is obtained by relaxing the integer constraints (Eq.13) with

$$\delta_{ij}, z_{ij}, y_{ir} \in [0, 1] \qquad \forall i, j \in P; i \neq j \tag{14}$$

3 Decomposition and Column Generation

As the experimental result presented in Beasley *et al.* (2000), the **LMIP** provides a poor bound on the optimal of **MIP**. In order to get a better formulation, we formulate it as a set partitioning formulation. Let S denote the set of all feasible landing sequences.

Let a_i^s be 1 if plane i appears in the landing sequence s ($s \in S$) and 0 otherwise. Let c_s be the cost of the landing sequence s, which is given by,

$$c_s = \sum_{i=1}^{P} (g_i \alpha_i a_i^s + h_i \beta_i a_i^s)$$

The resulting model becomes:
SP:

$$\min \quad \sum_{s \in S} c_s z_s \tag{15}$$

$$\text{s.t.} \quad \sum_{s \in S} a_i^s z_s = 1 \quad \forall i \in P \tag{16}$$

$$\sum_{s \in S} z_s = R \tag{17}$$

$$z_s \text{ binary} \tag{18}$$

where the z_s is the binary variable which is 1 if the landing sequence s is selected and 0 otherwise.

(Eq.15) is the objective function minimizing the accumulated costs of all the planes. Constraints (Eq.16) ensure that each plane lands on exactly one runway, while the constraint (Eq.17) indicates the limit on the number of the runways.

Consider a small example involving 4 planes and 2 runways. It is possible to enumerate all possibilities of the landing sequences. Fig.2 shows the coefficient matrix for this small problem.

1:	1				1	1	1				1	1	1		=1
2:		1			1			1	1		1	1		1	=1
3:			1			1		1		1	1		1	1	=1
4:				1			1		1	1		1	1	1	=1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	=2

Figure 2: Coefficient matrix for a small example involving of 4 planes and 2 runways

However, for a problem with 50 planes, there exsit $\sum_{k=1}^{50} \binom{50}{k} \approx 1.1259 \times 10^{15}$ feasible columns. It is computational inefficient to enumerate all these columns. Instead, we use column generation method to solve the problem. The column generation method split the problem into two problems: a master problem and a subproblem. We start by solving the master problem which is an LP relaxation with only a small set of the columns (denoted as **LSP**). This problem is also denoted the restricted linear program. Afterwards, we check if the addition of one or more columns, currently not in the restricted linear program, might improve the LP-solution. Similar to the simplex algorithm, a variable of negtive reduce cost can improve the solution. The column generation can be stopped as soon as no further columns with negative reduced cost exist. The general framework of column generation is presented in Fig.3.

generate an initial feasible solution (columns)
repeat
solve the master problem by linear programming
find out the columns with negetive reduce cost
add the column(s) into the master problem
until termination, when there exists no column with negative reduce cost

Figure 3: The framework of column generation

When solving the **LSP** by the column generation procedure described earlier, the aim of solving subproblems is to find the column with the minimum reduced cost, which can be given by,

$$\min \quad \sum_{i=1}^{P}(g_i\alpha_i a_i + h_i\beta_i a_i) - \sum_{i=1}^{P}\pi_i a_i - \lambda \tag{19}$$

π_i denote the dual variable value corresponding to plane i, for each $i \in P$, in constraint (Eq.16), and λ denote the dual variable value corresponding to the (Eq.17) in the master problem. a_i is the binary variable which is 1 if plane i appears in the landing sequence and 0 otherwise.

The constraints for obtaining a landing sequence are:

$$\begin{aligned}
\text{s.t.} \quad & a_i E_i \le x_i \le a_i L_i && \forall i \in P && (20)\\
& \delta_{ij} \le a_i && \forall i,j \in P; i \ne j && (21)\\
& \delta_{ij} \le a_j && \forall i,j \in P; i \ne j && (22)\\
& 1 \ge \delta_{ij} + \delta_{ji} \ge a_i + a_j - 1 && \forall i,j \in P; i \ne j && (23)\\
& x_j \ge x_i + S_{ij}\delta_{ij} - (L_i - E_j)\delta_{ji} - (a_i L_i - a_j E_j) \\
& \qquad + (L_i - E_j)(\delta_{ij} + \delta_{ji}) && \forall i,j \in P; i \ne j && (24)\\
& \alpha_i \ge a_i T_i - x_i && \forall i \in P && (25)\\
& 0 \le \alpha_i \le T_i - E_i && \forall i \in P && (26)\\
& \beta_i \ge x_i - a_i T_i && \forall i \in P && (27)\\
& 0 \le \beta_i \le L_i - T_i && \forall i \in P && (28)\\
& x_i = a_i T_i - \alpha_i + \beta_i && \forall i \in P && (29)\\
& \delta_{ij}, a_{ij} \text{ binary} && \forall i,j \in P && (30)
\end{aligned}$$

The landing time (x_i), earliness (α_i) and tardiness (β_i) are active only if the plane i appears in the landing sequence (i.e. $a_i = 1$). This is ensured by constraints (Eq.20) Similarly, constraints (Eq.21) and (Eq.22) show that δ_{ij} is active if i and j both are in the landing sequence (i.e. $a_i = a_j = 1$). Constraints (Eq.23) show that either plane i must land before plane j ($\delta_{ij} = 1$) or plane j must land before plane i ($\delta_{ji} = 1$) if both of them are active (i.e. $a_i = a_j = 1$). Constraints (Eq.24) enforce the separation time

between two planes. There are 4 cases considered here:

a. if both of plane i and j are active (i.e. $a_i = a_j = 1$), then either i lands before j (i.e. $\delta_{ij} = 1$) or j lands before i (i.e. $\delta_{ij} = 0$). Therefore, (Eq.24) becomes either $x_j \geq x_i + S_{ij}$ ensuring that separation is enforced, or $x_j \geq x_i - (L_i - E_j)$ a constraint that is always true,

b. if $a_i = 1$ $a_j = 0$, from (Eq.21) and (Eq.22), we have that $\delta_{ij} = \delta_{ji} = 0$. Therefore, (Eq.24) becomes $0 \geq x_i - L_i$,

c. if $a_i = 0$ $a_j = 1$, similar to case b, (Eq.24) becomes $x_j \geq E_i$,

d. if $a_i = 0$ $a_j = 0$, it becomes $0 \geq 0$.

The subproblem is denoted as **SUB**. If the objective function value is non-negative, then the master problem **LSP** has been solved, otherwise the corresponding column a is added to the master problem and the master problem is then solved again.

For Integer programs (**IP**) like the one presented in this work, however, column generation may not find the optimal solution as linear programming duality theory is not valid for **IP**s. In this case, we use column generation together with branch-and-bound (called branch-and-price) to guarantee that we end up with an integer solution.

4 Branch-and-Price

For solving our problem **SP** traditional branching on the z-variable may cause trouble along a branch where a variable has been set to be zero. Recall that z_s in **SP** represents a partial landing sequence on a single runway generated by solving a single runway subproblem, the branching $z_s = 0$ means that this partial landing sequence is excluded and hence no such schedule can be generated in the subsequent subproblems. However, it is very hard to exclude a landing sequence when solving a single runway subproblem.

Instead of branching on the z-variable in the set partitioning formulation, we branch on whether plane j is landed immediately after i or not. Let y_{ij} denote the new variable which is 1 if plane j is landed immediately after i and 0 otherwise. For any feasible solution of **LSP**, ($\bar{z}_s, s \in S$), the corresponding $\bar{y}_{ij}$ is given by

$$\bar{y}_{ij} = \sum_{s \in S} w_{ij}^s \bar{z}_s \geq 0 \qquad \forall i, j \in P; i \neq j \tag{31}$$

where w_{ij}^s is 1 if $s \in S$ covers both i and j and i is immediately after j, and 0 otherwise.

Consider a branch-and-bound node suppose its **LSP** solution (denoted as $\bar{z}_s, s \in S$), is fractional and is not pruned. Compute the corresponding $\bar{y}_{ij}$ value by (Eq.31). Then a branching decision can be made on this node. A pair (m, n) is selected such that $\bar{y}_{mn}$ is fractional value closest to 1. (i.e.$\bar{y}_{mn}$ = max($\bar{y}_{ij}, \bar{y}_{ij} \in (0, 1)$)). Then two son nodes are created, one along the branch with $\bar{y}_{mn}$ fixed as 1 and the other as 0. Constraints enforcing this requirement need to be added to the problem.

a. If $\bar{y}_{mn}$ is fixed as 1, then the initial restricted master problem of the corresponding son node consists of all the columns of its father node where plane n is landed immediatly after plane m if they are active. At the same time, the structure of the

subproblem **SUB** is updated. The following two constraints should be imposed:

$$a_m = a_n \tag{32}$$

$$\sum_{k\in P} \delta_{km} = \sum_{k\in P} \delta_{kn} - a_i \tag{33}$$

Constraint (Eq.32) indicates that either both or none of them is covered by the landing sequence. Constraint (Eq.33) ensures n lands immediatly after m if they are both active (i.e.$a_m = a_n = 1$).

b. If $\bar{y}_{mn}$ is fixed as 0, then the initial restricted master problem of the corresponding son node consists of all the columns of its father node except where plane n is landed immediatly after plane m. At the same time, the following two constraints should be added to the subproblem **SUB**:

$$a_m + a_n \leq 1 + Md_{mn} \tag{34}$$

$$2 - M\delta_{nm} \leq \sum_{k\in P} \delta_{kn} - \sum_{k\in P} \delta_{km} + M(1 - d_{mn}) \tag{35}$$

where d_{mn} is a binary variable. M is a large number. In this case, either constraint (Eq.34) is active (indicating only one or none of them is in the sequence), or constraint (Eq.35) is active (indicating either n lands before m or there are planes landing after m and before n).

The node selection strategy we use here is the *depth-first Search* (DFS). If the current node is not pruned (i.e. the solution is fractional and is less than the upper bound *UB* of the search tree), then the DFS is applied such that the son node with $\bar{y}_{mn} = 1$ is selected as the next node to be solved.

5 Implementation Details

5.1 The Upper Bound

Before starting the exploration of the search tree, we have to set the upper bound for the problem. It is usually provided by a feasible solution through an approximation algorithm or a heuristic method. Beasley *et al*. (2000) presented a simple heuristic for the problem. Specifically, the planes are ordered in nondecreasing target time, then assigned one by one to the runway with the least cost. The landing cost on a runway is calculated according to the previous landings and the corresponding separation time. Let B_{ir} be the cost of airplane i landing on runway r, then

$$B_{ir} = max(T_i, x_k + S_{ki} | \forall k \in A_r)$$

where A_r is the previous landings on runway r and plane i is the one being considered for a runway assignment. The current plane is assigned to the runway r^* with the minimum B_{ir}. Hence, after this step, the temporary assignment of landing times are always on or after the target times. To improve the solution obtained, we solve the LP problem with the order of landing on the runway fixed. The optimal objective function value of the LP problem is a true upper bound that we denote with Z_{UB}. This can also be used for tightening the time windows as follows:

$$E_i = max[E_i, T_i - Z_{UB}/g_i] \qquad \forall i \in P$$

$$L_i = min[L_i, T_i + Z_{UB}/h_i] \qquad \forall i \in P$$

5.2 The initial columns for Column Generation

The initial columns for the the column generation consist of three parts. The first part is P 'dummy' columns. Each column contains one plane without taking account of the runways, that is, the column contains a single 1 for the i'th row ($a_i^c = 1$). They are added to ensure a feasible LP upon branching and they are assighed a cost sufficiently high in order to force them out of the basis in the optimal solution. In order to get a good starting objective value, we also add the columns corresponding to the heuristic solution in the master problem. We also generate some columns with cost 0. Specifically, the planes are ordered in nondecreasing target time (stored in $ordind[]$), then considered one by one whether to be landed or not. If the cost of landing the current plane is 0, in other words, the separation time between its target time and the previous landings is satisfied, we assign it on the runway, otherwise we discard it and store it into $nonexist[]$. Let k denote the number of non-existing planes. The next step is to generate k columns, in each column, we first fix one of the non-existing planes to be landed at its target time, then insert the rest P-1 planes as many as we can. The detail of the algorithm for generating the columns is shown in Fig.4.

initialization
ordindT = $sortindex$(T);
cost0_col = $zeros$(num,P);
nonexist = [];
col0_col(1, ordindT(1)) = 1;
for $j = 2$ to p **do**
  if the seperation time between T(ordindT(j)) and the previous landings is satisfied **then**
    cost0_col(1, ordindT(j)) = 1;
  else
    nonexist = [nonexist, j];
  end if
end for
for $i = 1$ to $length$(nonexist) **do**
  for $j = 1$ to P **do**
    if $j \neq i$ **then**
      if the seperation time between T(ordindT(j)) and the previous landings is satisfied **then**
        cost0_col(i+1, ordindT(j)) = 1;
      end if
    end if
  end for
end for

Figure 4: The algorithm for generate the columns with cost 0

6 Experimental Results

The decomposition algorithm were implemented in Matlab on a ..Hz Pentium PC with ..GB of memory. The linear programming problems **LSP** and the subproblems are solved by GAMS. Computational results are presented in this section for 8 instances, publicly available from the OR-Library, involving from 10 to 50 aircrafts.

Table 1 shows the result of our first computational experiment. The first column is the number of the problem. The following two column 'P' and 'R' represent the number of planes and the number of runways respectively. Each problem was solved with an increasing number of runways until the optimal solution value dropped to zero (indicating that we have sufficient runways to enable all planes to land on target). The column 'LMIP-IP gap' represents the gap in percentage between the linear relaxation solution value of the mixed integer formulation and the integer solution value, while the column 'LSP-IP gap' represents the gap between the linear relaxation solution value of column generation and the integer solution value. The rest information regarding the number of tree nodes searched for solving the problem, the number of columns generated, and the total running time of the algorithm.

Problem Number	P	R	LMIP-IP Gap(%)	LSP-IP Gap(%)	Number of Tree nodes	Number of Columns	Total Time (sec)	Optimal Soluiton
1	10	2	100	0.0	1	23	27.1	90
		3	-	-	-	-	-	0
2	15	2	100	0.0	1	39	64.0	210
		3	-	-	-	-	-	0
3	20	2	100	0.0	1	24	4.9	60
		3	-	-	-	-	-	0
4	20	2	100	0.0	1	92	260.8	640
		3	100	0.0	1	85	96.7	130
		4	-	-	-	-	-	0
5	20	2	100	0.0	9	153	1066.1	650
		3	100	0.0	12	121	496.5	170
		4	-	-	-	-	-	0
6	30	2	100	0.0	9	440	3207.7	554
		3	-	-	-	-	-	0
7	44	2	-	-	-	-	-	0
8	50	2	100	0.0	1	204	909.3	135

Table 1: Computational results for the multiple runway problem

It is clear from Table 1 that our algorithm found the optimal solution for all the problems. In addition, the optimal solutions were found early in the tree search in our branch-and-bound algorithm with less than 12 tree nodes. Compared the column 4 and 5, we can see that, for every problem we tested, the LMIP-IP gap is 100, while the LSP-gap is 0 indicating that the lower bound provided by the column generation

is exact the same as the solution value of the integer problem **SP**. The LSP-IP gap is much smaller than the LMIP-IP gap. Moreover, the optima are produced with very few columns generated. For example, if we consider the last problem with 50 planes, there exsit $\sum_{k=1}^{50} \binom{50}{k} \approx 1.1259 \times 10^{15}$ feasible columns in the entire set partitioning master problem. However, in our algorithm, only 204 columns were used to get the optimality for this problem. It can also be seen from the table that less than 450 columns were generated for all the problems.

Compared with the computational performances of the same problem in some other literatures, the running time in this paper is substantially larger. One main factor is that we used too much time for calling GAMS from Matlab. Therefore, we believe that using other languages (e.g. JAVA or C++) will significantly shorten the running time.

7 Conclusion

In this paper, we presented a set partition formulation for the ALP and an exact algorithm using branch-and-price method which have not been applied to the ALP previously in the literature. The computational experiments show that the bound based on the column generation is much better than the LP relaxation of the mixed integer formulation and the algorithm can solve the problem successfully. Further more, it is worthy to note that all of our problems we solved generating less than 450 columns. A potential problem as the problem size increases is that, it will not be a computationally effective procedure to solve the subproblem by mathematical model **SUB**. Research efors can be made in future to develop an effective heuristic method for the subproblem.

References

[1] J.E. Beasley, M. Krishnamoorthy, Y.M. Sharaiha, and D. Abranmson, Scheduling Aircraft Landings-the static case, *Transportation Science*, 34(2):180-197, 2000.

[2] A.T. Ernst, M. Krishnamoorthy, and T.H. Storer, Heuristic and Exact Algorithms for Scheduling Aircraft Landings, *Networks*, 34:229-241, 1999.

[3] H. Pinol, and J.E. Beasley, Scatter Search and Bionomic Algorithms for the Aircraft Landing Problem,

[4] G. Jung, and M. Laguna, Time segmenting heuristic for an aircraft landing problem,

[5] J. Abela, D. Abramson, M. Krishnamoorthy, A. De Silva, and G. Mills, Computing optimal schedules for landing aircraft, In *Proceedings of the 12th National ASOR Conference*, pages 71-90, 1993

[6] J.E. Beasley, J. Sonander, and P. Havelock, Scheduling aircraft landings at London Heathrow using a population heuristic. *Journal of the Operational Research Society*, 55:54-64, 2004.

[7] V. Ciesielski, and P. Scerri, Real time genetic scheduling of aircraft landing times, In D. Fogel, editor, *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation (ICEC98)*, pages 360-364. IEEE, NewYork, USA, 1998.

[8] V.H.L. Cheng, L.S. Crawford, and P.K. Menon, Air traffic control using genetic search techniques, *IEEE International Conference on Control Applications*, 1999.

[9] L. Bianco, S. Ricciardelli, G. Ginaldi and A. Sassano, Scheduling tasks with sequence dependent processing time, *Naval Research Logistics*, 35:177-184, 1988.

[10] L. Bianco, A. Mingozzi and A. Ricciardelli, The Traveling Salesman Problem with Cumulative Costs, *Networks 23*, 81-91, 1993.