

Computing Optimal Schedules for Landing Aircraft

J. Abela §
D. Abramson ‡
M. Krishnamoorthy †
A De Silva §
G. Mills †

October 19, 1995

§ Department of Computer Systems Eng., R.M.I.T. P.O. Box 2476V, Melbourne, 3001.

‡ School of Computing and Information Technology, Griffith University, Kessels Rd,
Nathan, Queensland, 4111.

† Division of Mathematics and Statistics, C.S.I.R.O., Private Bag 10, Clayton, VIC 3168.

Contact Information: davida@cit.gu.edu.au

The 12th National Conference of the Australian Society for Operations Research. Adelaide, July 7-9, 1993

Abstract

In this paper we describe two approaches for solving the problem of scheduling aircraft landing times. In this problem, the air traffic controller must compute a landing time for each aircraft in the *horizon*, and must attempt to minimise costs whilst observing a number of hard constraints. The costs are associated with aircraft landing times varying from the preferred landing time. The hard constraints are concerned with preventing aircraft from landing within a prescribed minimum time of each other. Given a certain number of aircraft within the air traffic controllers' horizon at a certain point in time, we develop a heuristic for the problem using a genetic algorithm (GA). We also formulate this problem as a mixed integer program (MIP) and develop a branch and bound algorithm for its solution. We compare the quality of the two solutions obtained through computational tests performed on a wide range of problems.

1 Introduction

In this paper we describe two solution methods for computing optimal schedules for landing aircraft. One is an approximate solution approach using a genetic algorithm (GA) idea. The other is an exact, branch and bound algorithm based on an exact formulation of the problem as a 0–1 mixed integer programming problem.

The GA mimics the process of natural selection. It operates on a population of individual solutions to produce subsequent generations of better solutions. The branch and bound algorithm involves the relaxation of the 0–1 constraints in the problem to produce linear programming subproblems at each node of the search tree. The GA solutions to the problem are used as upper bounds in this search to prune the resulting tree. We are not aware of any previous exact solution approaches to solve this problem.

In this study, we only consider the static case of scheduling landing times for aircraft in the air at a given time. The dynamic case of incorporating new aircraft entering the system and ignoring aircraft that have landed and, hence left the system, is not considered here. Furthermore, in this study, we are only concerned with the issue of assigning landing times to incoming aircraft. In other words, we assume that the runway is always available for landing aircraft.

In Section 1 we introduce the problem in detail. We explain the costs involved, the objective and the constraints. In Section 2, we introduce the general principles of genetic algorithms. We describe the GA to solve the problem in Section 3. This algorithm provides upper bounds on the problem. In Section 4, we give an exact mathematical formulation of the problem. The branch and bound algorithm to solve it incorporates the GA upper bounds and an LP-based lower bounding scheme, along with some clever branching rules. This is described in Section 5. In Section 6, we provide extensive computational results on problems involving systems of up to 20 aircraft. Comparisons between the solution quality and the run times of the two algorithms are made and some conclusions are arrived at in Section 7. We conclude with suggestions for further research in Section 8.

2 The Aircraft Landing Problem

Upon entering within the air traffic controller’s radar range (*horizon*), an aircraft requires the control tower to assign it a landing time, known as the *broadcast time*. This time must exist within a specied time window which is dependent on the type of aircraft. This time window is bound by an *earliest time* and *latest time*. The earliest time represents the the time required to land if the aircraft flies at its maximum airspeed. The maximum airspeed requires the uneconomical overuse of fuel per nautical mile. The latest time corresponds to the time the aircraft will reach the runway if it slowed to its slowest possible airspeed while also *holding* for the maximum allowable time. This implies a substantial delay. Each type of aircraft has a most economical, preferred

speed referred to as the cruise speed. An aircraft is said to be assigned its *preferred time* if it is required to fly in to land at its cruise speed. If the control tower requires the aircraft to either slow down (a process which is also known as *holding*) or speed up, a cost will be incurred. This cost will grow as the difference between the preferred and broadcast time increases. Figure 1 illustrates the variation in cost within the time window of a particular aircraft.

In the example shown in Figure 1, we have assumed that the cost per unit time of speeding up (delaying) an aircraft prior to (respectively, past) its preferred landing time is linear. This need not necessarily be the case. Most real examples show that the delay cost per unit time past an aircraft's preferred landing time normally has a step-wise characteristic. Likewise, the cost per unit time of speeding up an aircraft is sometimes nonlinear. However, although genetic algorithms can handle nonlinear cost structures, we have assumed linear speeding-up and holding costs in this study, for the sake of simplicity.

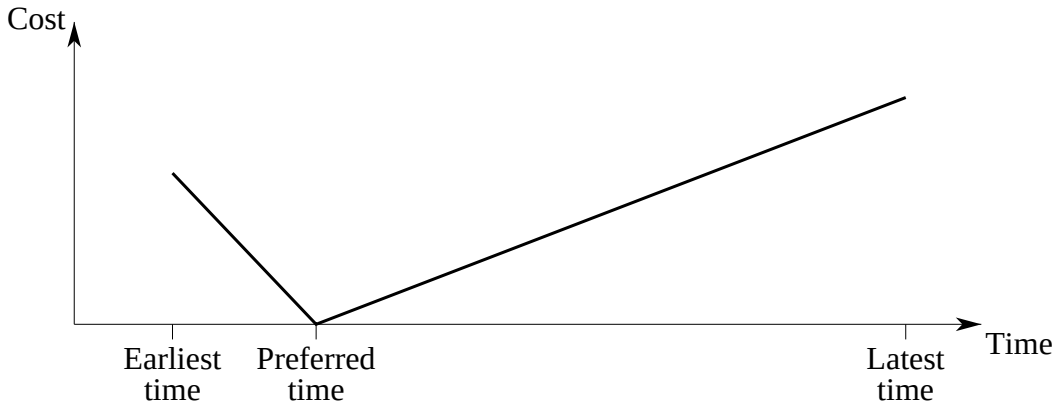


Figure 1: Variation in cost for an aircraft within its time window

The inter-landing time between any two successive landing aircraft is bounded by a minimum which is dependent upon their size. A Boeing 747 generates a great deal of rear wind disturbance and therefore requires that the next aircraft be a safe distance in time behind it. However, a light aircraft generates little rear wind disturbance and therefore the following aircraft need not be as far back.

The requirement of a schedule for landing aircraft is, therefore, to assign each aircraft in the air traffic controller's horizon a broadcast landing time, which minimises the overall costs of speeding up or slowing down of all aircraft in the horizon, while ensuring that all aircraft land within their respective time windows and while observing the minimum interlanding times.

In the next section we describe a heuristic, genetic algorithm to compute such schedules.

3 The Genetic Algorithm

As background to the GA that we develop for the problem of landing aircraft, we describe in this section the general principles of GA's.

In biological evolution, fit individuals are more likely to survive their environment. They are therefore more likely to generate offspring. Unfit individuals will tend to die before they can produce offspring. Therefore, successive generations will tend to be fitter than the previous ones. This process of natural selection may be viewed as being a search for the optimal combination of characteristics of a species that will survive best in a given environment.

The particular characteristics of an individual are determined by its genetic makeup which is encoded in its chromosome. It is the chromosome that uniquely identifies an individual. The chromosome is made up of an array of genes that represent the type of characteristic, that is, whether the individual has brown or blue eyes, is tall or short, muscular or weak, etc. The actual characteristic in the gene is referred to as the allele.

When two individuals mate to produce offspring, the offspring's chromosome will be copied from both parents so that the offspring is similar to both parents but not exactly identical to either. Thus, this process has generated a new unique individual that has a fitness comparable to both parents. The characteristics of an offspring may be further randomly altered by mutation.

Genetic Algorithms apply the same principle of survival of the fittest to optimisation problems. A random sample of initial solutions are generated to form an initial population. The initial generation process involves that each gene in the chromosome be set a random allele (within a specified range so as not to make the solution infeasible). The number of genes in the chromosome is equal to the number of variables in the problem.

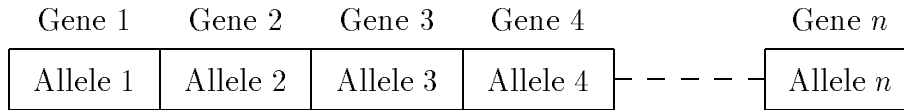


Figure 2: Chromosome for GA for problem with N variables

From this initial population two parents are selected, based on their fitness, to produce offspring. The fitter the individual the greater the likelihood of selection.

Once the parents are selected, they are combined to form the offspring. This process of generating the offspring is referred to as *crossover*, and involves the crossing over, at a random point, of different parts of the parent's chromosome to construct the offspring's. This results in the production of an offspring whose chromosome is similar, but not identical, to each of the parents. Parents may be selected any number of times.

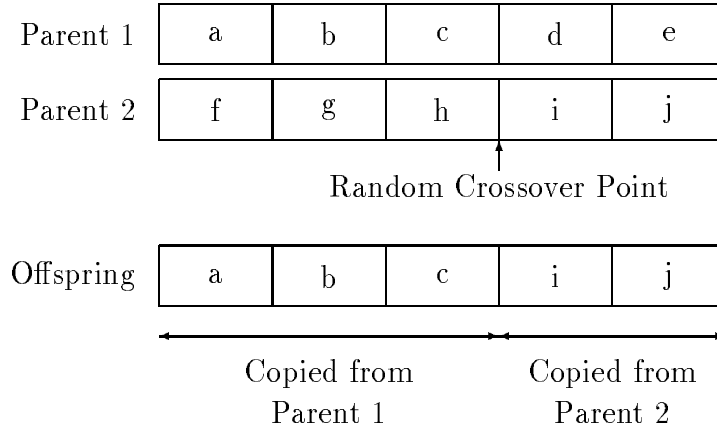


Figure 3: Using crossover to generate offspring

After crossover has generated the offspring, the resultant chromosome may be further altered by the *mutation* process. Mutation involves the random alteration of a gene. This random alteration introduces new characteristics into the population. These new characteristics may be beneficial or detrimental to the offspring's fitness.

This process of parent selection, crossover and possible mutation is repeated until a whole new population is generated. This new population constitutes the *next generation*, and parents are now selected from it to produce the next subsequent generation. This process of evolution from one generation to the next continues until an individual whose fitness is greater than some pre-determined threshold is obtained.

This simulation of evolution has been applied to many wide ranging optimisation problems (Davis [1991], Michalewicz [1992], Goldberg [1989], Soucek, *et al* [1992]) and found to be very effective at avoiding local optima and quick at obtaining solutions which are optimal or close to optimal.

4 Implementation of the GA to Schedule Landings

The implementation of a Genetic Algorithm solution requires that a chromosomal representation for a particular problem be selected. This chromosomal representation should characterise a particular solution along with a calculable fitness. Each gene in the chromosome will represent a problem variable and the allele will represent the value of that variable. Once the chromosomal representation has been decided upon, the mechanics of parent selection, crossover and mutation need to be considered.

For the aircraft landing problem each gene will represent an aircraft, and the allele value will be the landing time for that aircraft. Therefore, the chromosome will represent an array of aircraft and their proposed landing times. This constitutes a schedule, i , with an associated cost, and a fitness value, F_i .

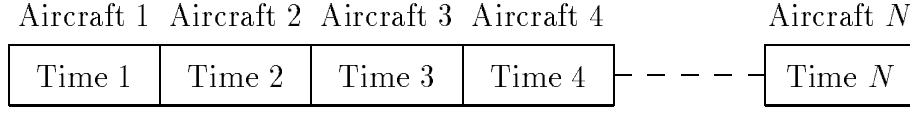


Figure 4: Chromosome representing a schedule for landing N aircraft

Two parents are selected based on their fitness. The greater their fitness, the greater the probability of selection. Thus, fit individuals containing good qualities are more likely to be selected as parents than unfit individuals. Given that there are, at any point in time M individuals in the population, where M is the maximum population size, the exact probability, P_s , of an individual i being selected is given by:

$$P_s(i) = \frac{F_i}{\sum_{i=1}^M F_i}$$

Crossover uses two parents to produce two offspring. For the aircraft landing problem, crossover will produce two new schedules, constructed from the parent schedules. However, (without intervention) these resultant schedules may not be feasible in that the minimum interlanding times may not be observed. Also a group of aircraft may be scheduled unnecessarily far out in time. These two factors bring about the need for two heuristic operators to process the offspring after crossover. These are the *force feasible* and *squash heuristics*.

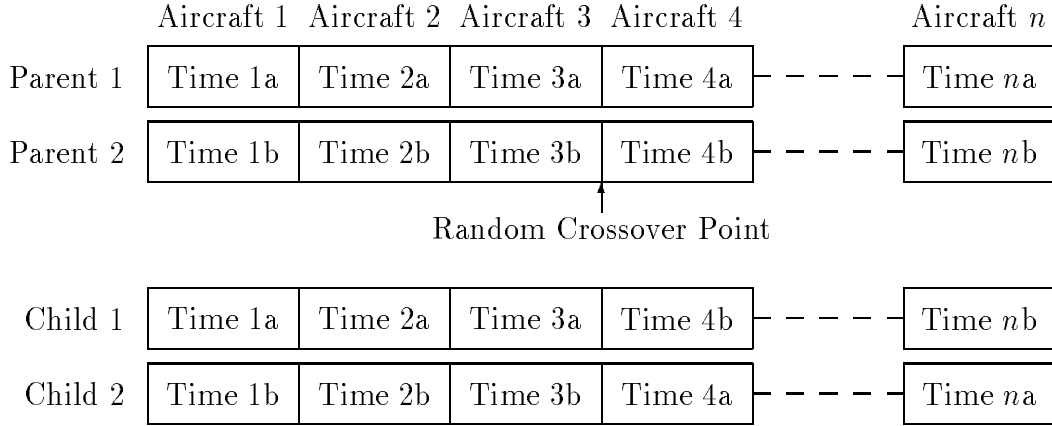


Figure 5: Crossover for aircraft landing problem

The force feasible heuristic forces the interlanding time between two aircraft to be at least the minimum allowable. The aircraft are first ordered. Then the first two aircraft are examined and if a conflict occurs, the rest of the schedule is moved back until a conflict no longer exists. The next pair of aircraft are then considered, and this process continues until all the interlanding times are at least the minimum allowable.

The squash heuristic locally optimises the schedule while preserving the order of the aircraft. If the interlanding time between any two aircraft is greater than the mini-

mum required separation then the later aircraft would be unnecessarily delayed. Thus, moving the later aircraft forward will decrease cost.

After an offspring is generated, we sometimes perform a mutation operation on it. Mutation is the process of introducing new characteristics into the population. It involves the random alteration of an individual's chromosome. For the aircraft landing problem, the mutation process involves randomly changing an aircraft's landing time. This has the effect of changing the landing order, and thus a new schedule is generated. If the alteration was beneficial to the individual then the individual's fitness will increase, making it more likely for selection as a parent. If the alteration was detrimental, then the fitness of the individual would lower and reduce the likelihood of selection as a parent. The better alterations are more likely to be passed on from one generation to the next. Mutation is not carried out on all offspring. It occurs on some offspring selected randomly, with a certain fixed probability that is referred to as P_{mut} .

The relative success of a GA run is very dependent on maintaining a diversity of individual fitness when forming successive generations. It is therefore important to minimise the tendency of an extraordinary individual to dominate successive populations by scaling the fitnesses of all members such that the ratio between the maximum fitness and mean fitness is controlled. This ratio is referred to as the fitness scaling co-efficient, C_{mult} , and is introduced in Goldberg [1989] (pages 76-79). Goldberg states that for small populations of 50 to 100, a scaling co-efficient of 1.2 to 2.0 have been used successfully.

In this study, the GA is not used in isolation, but is combined with a branch and bound algorithm. The GA is run first to provide an upper bound on the optimal solution to the problem. This value is used during the branch and bound tree search based on a relaxation of an exact formulation of the problem which is now described.

5 An Exact Formulation of the Problem

Let us consider the static case of the air traffic controller's problem, involving the scheduling of N aircraft in the horizon. Let t_i^e, t_i^p and t_i^l denote the earliest landing time, the preferred landing time and the latest landing time, respectively, of the i th aircraft, $i = 1, \dots, N$.

Further, let us assume that the cost per unit time of speeding up the i th aircraft is the same as the cost per unit time of holding it. Denote this cost by C_i . We will show later on that this assumption is not binding on the formulation and that we can easily incorporate situations where the cost per unit time for speeding up of the i th aircraft, say, S_i , is different to the cost per unit time of holding it, say, H_i . However, in this study, we have assumed that, for each $i = 1, \dots, N$, $S_i = H_i = C_i$. Here, C_i merely refers to the *deviation cost* per unit time.

Let X_i be the broadcast time for the i th aircraft. These are then our decision variables.

Let d_i denote the *deviation time* for the i th aircraft. An aircraft's deviation time is the deviation from its preferred landing time. Hence,

$$d_i = |X_i - t_i^p|$$

Our objective should be, therefore, to minimize the sum of the weighted deviations of all aircraft in the horizon, where the weights are C_i , the deviation costs per unit time. The constraints that need to be satisfied are the time window constraints for each aircraft and the inter-aircraft separation constraints.

The objective function can be represented as:

$$\begin{aligned} \min \quad & \sum_{i=1}^N C_i d_i \\ \text{i.e.,} \quad & \sum_{i=1}^N C_i |X_i - t_i^p| \end{aligned} \tag{1}$$

Equation (1) is an absolute value function which cannot be solved easily because of its discontinuous nature (See Shanno & Weil [1971]). We could resolve the situation by introducing 0–1 variables which control whether the deviations are positive or negative. However, this would make the objective function non-linear.

We model (1) by introducing two new continuous variables Z_i and Y_i for each aircraft, $i = 1, \dots, N$. The objective function is then accompanied by constraints which determine the values which Z_i and Y_i assume, depending on whether or not d_i is negative or positive. The system of equations is represented as follows:

$$\min \sum_{i=1}^N \left[\frac{C_i}{2} \{(X_i - t_i^p) + Y_i\} - \frac{C_i}{2} \{(X_i - t_i^p) - Z_i\} \right] \tag{2}$$

We add the constraints:

$$Y_i \geq (X_i - t_i^p) \tag{3}$$

$$Y_i \geq -(X_i - t_i^p) \tag{4}$$

$$Z_i \geq (X_i - t_i^p) \tag{5}$$

$$Z_i \geq -(X_i - t_i^p) \tag{6}$$

Constraints (3), (4), (5) and (6) ensure that the absolute value difference between X_i and t_i^p is used in (2). If the difference is negative, the second term in (2) contributes a positive value to the function. The first term contributes a positive value to (2) if the difference is positive. Thus, it is also possible to include different linear costs for positive and negative deviations from t_i^p , for all i . However, in this paper, we have not included these effects.

The real constraints in the problem can be written down as:

$$X_i > t_i^e, \quad \forall i, \quad (7)$$

$$X_i < t_i^l, \quad \forall i, \quad (8)$$

$$(X_i - X_l) \geq I_{il}, \quad \forall i, l = 1, \dots, N \quad (9)$$

Constraints (7) and (8) specify the lower and upper bounds respectively on the landing time of the i th aircraft. Constraint (9) are the *separation constraints* for each aircraft-pair in the horizon. Here, I_{il} is the minimum prescribed separation between the i th and the l th aircraft in the horizon. However, the constraint (9) is only valid if the broadcast time for aircraft i is greater than that of aircraft l . The constraint should not hold if aircraft l lands after aircraft i . In other words, the separation constraints (9) should be written down as:

$$\text{IF } X_i > X_l \text{ THEN } (X_i - X_l) \geq I_{il}$$

$$\text{IF } X_l > X_i \text{ THEN } (X_l - X_i) \geq I_{li}$$

The above IF-THEN constraints can be modelled using binary 0–1 variables δ_{il} for each aircraft-pair (i, l) such that,

$$\delta_{il} = \begin{cases} 1, & \text{if flight } i \text{ follows flight } l \\ 0, & \text{if flight } l \text{ follows flight } i \end{cases}$$

We can then replace constraint (9) with the following constraints:

$$(X_i - X_l) - M(\delta_{il} - 1) \geq I_{il}, \quad \forall(i, l) \quad (10)$$

$$(X_l - X_i) + M\delta_{il} \geq I_{li}, \quad \forall(i, l) \quad (11)$$

$$\delta_{il} \in \{0, 1\}, \quad \forall(i, l) \quad (12)$$

Here, M is a large positive integer. If the solution requires $X_i > X_l$ then δ_{il} has to assume the value 1 (and $\delta_{li} = 0$). While, if the solution requires $X_l > X_i$ then δ_{il} has to assume the value 0 (and $\delta_{li} = 1$). The above constraint does not imply immediate succession. In other words, if $\delta_{kj} = 1$, it does not necessarily imply that aircraft k immediately succeeds aircraft j in the sequence of landing aircraft. It merely specifies that X_k , the broadcast landing time of aircraft k , is greater than X_j , the broadcast landing time of aircraft j .

Equations (10) and (11) applies for each aircraft pair (i, l) . We can, however, see that, when implementing the above system of constraints to aircraft pair (i, l) , the constraint (11) actually applies for aircraft pair (l, i) . Similarly, when the above system of constraints are implemented for aircraft pair (l, i) , the constraint (11) applies to aircraft pair (i, l) . We can collect the terms and simplify the constraints that apply to an aircraft pair (i, l) by a single constraint represented as follows:

$$x_l - x_i + M\delta_{il} + s_{il} = M, \quad \forall(i, l) \quad (13)$$

$$\text{With } s_{il} \text{ such that } I_{il} \leq s_{il} \leq M - I_{li} \quad \forall(i, l) \quad (14)$$

The formulation of the problem, which we denote as P , is, therefore, objective function (2) and the constraints (3), (4), (5), (6), (7), (8), (13), (14) and (12). Problem P is a 0–1 integer programming problem.

6 A Branch and Bound Algorithm

We solve Problem P by relaxing the 0–1 restrictions on δ_{il} . In other words constraint (12) is re-written as:

$$\delta_{il} \geq 0 \tag{15}$$

$$\delta_{il} \leq 1 \tag{16}$$

At the root node the relaxed problem, P_R is a trivial LP whose solution values will be $X_i = t_i^p, \forall i = 1, \dots, N$. This is so since all the separation constraints are made redundant by relaxing the values that the 0–1 variables can take. The optimal solution of P_R , which will be denoted by $Z(P_R)$, will equal 0, since there is no speed up or holding required for any aircraft in this solution.

The violated separation constraints are introduced into the problem successively via an intelligent branching scheme. Branching constraints are used to generate LP subproblems. Given a solution X , at a particular node K of the tree search procedure, we sequence the aircraft according to their broadcast landing times. We then look at the separation times between successive aircraft in the schedule and compare these with the prescribed separation times. If the difference between the broadcast times of successive aircraft is less than the prescribed minimum separation times for these aircraft, the broadcast times for this aircraft pair is said to be *critical*. If none of the aircraft pairs are critical with respect to their separation, a feasible solution is found at node K . Otherwise, we pick out the aircraft pair, say (j, k) , whose broadcast times are most critical. This choice reflects the biggest violation of the separation constraints at node K . We branch from node K by setting δ_{jk} to either 1 or 0, thus enforcing the violated separation constraints.

Although a breadth-first tree search could have led to fewer nodes being searched, the size of the subproblem storage that this required led us to using a depth-first strategy.

The revised simplex algorithm was used to solve the LP subproblems in order to determine lower bounds at each node of the tree. The LP implementation incorporated the product form of the inverse. Sparse data structures were used to store the problem as well as the basis inverse. The algorithm handles bounds on the decision variables and ranges of the right side of the constraints. We solve the LPs at nodes of the tree search by fixing the upper bound and the lower bound of the 0–1 variables at either one or zero. As the parent node is solved to optimality, the optimal basis is passed on to the child node. Thus the initial basis is dual feasible but not primal feasible as the bounds of some variables are changed by the branching constraints. Ideally the dual

simplex method would have solved the LP at child nodes very efficiently. The dual simplex method requires the pivot row information (elements of all non basic variables of this row). As the revised simplex method does not store any nonbasic columns, all the nonbasic columns have to be reconstructed; an extremely time consuming process. Although the dual simplex takes fewer number of iterations, it takes much longer to solve. Thus the primal simplex was used to solve the LP subproblems at each node. It is hoped that the existing code will be improved by efficiently accessing the pivot row.

7 Computational Results

We tested the GA and the branch and bound algorithms on randomly generated test data. Given a value N for the number of aircraft, the complexity of the problem depends on the number of different types of aircraft in the horizon, denoted by T . We simulated different test problems using an aircraft traffic simulator that we developed. The simulator generated aircraft randomly and assigned them a type. The type of an aircraft is randomly chosen, in such a manner that there would be, on average, N/T aircraft of type $t = 1, \dots, T$ in a problem of size N with T aircraft. Once a type has been assigned, the simulator assigns an early landing time, a preferred landing time and a late landing time for the aircraft depending on its flying speed and on randomly perturbed tail and head winds.

In Table 1, we present results for experiments aimed at optimizing the GA parameters $Cmult$ and $Pmut$. The results show the GA solutions for exactly the same problem with $N = 25$ aircraft and $T = 2$ types. Each μ and σ signify the mean and standard deviation of the solution values over 10 runs.

The results in Table 1 illustrate that if $Cmult$ is too high then extraordinary individuals dominate subsequent generations and cause a detrimental narrowing in diversity in the population. If $Cmult$ is too low then very fit individuals are not being promoted sufficiently, resulting in a evolutionary search among the mediocre.

The mutation rate determines the level at which diversity is introduced into the population. If the mutation rate is too low, an insufficient amount of diversity is introduced to achieve near optimal results. The quality of the result depends very much on the diversity of the initial population. A higher mutation rate allows a greater proportion of diversity to be introduced, thus making the final solution relatively independent of the initial population.

CMult	PMut	μ	σ
1.5	0.01	1391	179
1.5	0.25	1107	84
1.5	0.5	1063	49
1.5	0.8	1231	85
1.5	1.0	1237	112
1.8	0.01	1583	269
1.8	0.25	1050	82
1.8	0.5	1031	37
1.8	0.8	1182	124
1.8	1.0	1226	77
2.0	0.01	1413	221
2.0	0.25	1070	81
2.0	0.5	1078	29
2.0	0.8	1127	66
2.0	1.0	1305	151

Table 1: Results for the GA experiments

The results show that we are justified in using $Cmult = 1.8$ and $Pmut = 0.5$. We used these values in the GA that provided the upper bound at the root node of the branch and bound algorithm.

The results for the branch and bound algorithm are provided in Table 2. Although problem-dependent values of $Cmult$ and $Pmut$ would lead to better GA solutions, we used the parameter values mentioned above. In Table 2, N represents the number of aircraft in the problem. T represents the number of different types of aircraft. The values of N and T provide a measure of the complexity of the problem.

In Table 2, GAP represents the relative percentage deviation of the GA upper bound to the optimal solution value. GA Time shows the time (in CPU Secs) for the GA to complete 1000 generations, while BB Time shows the time for the branch and bound algorithm to run to its completion. TFI represents the proportion of time for the branch and bound algorithm to find the first improvement on the GA upper bound. NFI represents the node at which this first improvement occurred, as a proportion of the total number of nodes. TFO depicts the proportion time at which the algorithm first finds the optimal solution and NFO depicts the node at which this first optimal solution was found as a proportion of the total number of nodes. Nodes shows the total number of tree nodes.

N	T	GAP		GA Time (Secs)		BB Time (Secs)		TFI	NFI	TFO	NFO	Nodes
		$\mu\%$	σ	$\mu\%$	σ	$\mu\%$	σ	$\mu\%$	$\mu\%$	$\mu\%$	$\mu\%$	
10*	2	0.00	0.00	63.20	0.75	4.27	5.44	77.13	84.26	77.13	84.26	8
10*	3	4.18	11.95	63.60	0.80	3.75	6.39	63.19	85.68	63.19	85.68	7
10*	4	1.64	3.02	64.20	1.08	8.38	7.82	52.21	69.49	52.21	69.49	17
13*	2	0.67	2.00	80.70	0.46	7.63	9.44	60.98	80.45	60.98	80.45	6
13*	3	1.29	2.26	81.50	1.57	50.13	56.98	35.10	49.62	35.10	49.62	41
13*	4	9.21	8.83	88.50	15.30	69.26	97.17	49.12	68.07	58.06	77.01	61
15*	2	8.85	11.96	133.80	46.11	203.07	354.80	41.36	59.72	46.36	64.85	98
15*	3	7.80	7.18	99.10	10.39	175.44	204.72	46.55	59.63	57.33	70.93	93
15*	4	6.21	5.65	95.90	1.30	4090.66	7780.89	23.90	36.77	31.02	43.70	2244
17*	2	3.03	5.04	106.50	2.06	3878.45	11439.25	42.80	54.78	54.68	65.78	1312
17*	3	7.90	8.09	108.00	1.79	525.02	791.15	42.76	51.21	49.10	57.95	184
17*	4	14.44	9.32	114.70	17.51	12944.47	15368.83	19.17	22.32	54.39	57.38	4499
20#	2	11.36	7.08	334.75	79.38	455.33	532.21	16.25	28.71	41.52	53.79	99
20#	3	7.50	6.50	285.00	31.76	3069.14	1705.12	26.33	27.47	42.82	43.24	555
20#	4	8.15	3.68	329.25	99.01	17419.26	12797.44	44.77	43.83	64.15	63.78	3254

Table 2: Results for the Branch and Bound algorithm

*: results tabulated for 10 problems

#: results tabulated for 4 problems

We can see from Table 2 that the GA solves to optimality only for the case with $N = 10, T = 2$. For the more complex data sets, the GA performs sub-optimally. Although the worst GA solution is 14% away from the optimal, the mean GAP is approximately 6%. The mean standard deviation for all the problems is also approximately 6%. We can therefore conclude that, for the problem sizes we have chosen, the GAP is not excessively large. The GA time to solve increases as the complexity of problems increase. This behaviour is also shown by the branch and bound algorithm. However, even for the larger (20 aircraft) problems, the GA provides solutions within 5 minutes approximately. Moreover, this increase in time is orders of magnitude below the corresponding increase in the time required for the branch and bound to solve to optimality.

The branch and bound algorithm consumes a large amount of time to solve some of the larger problems, while it solves some of the smaller problems faster than the GA. Moreover, there is a significantly large deviation in the times to completion for the branch and bound algorithm. Thus, we can conclude that some of the larger problems solve quickly while a few of them take a very long time to complete. This is probably due to congestion in the sky (in these difficult problems), thereby leading to many separation-time conflicts that the algorithm has to resolve.

The average proportion of total time for the first improvement produced by the branch and bound algorithm is approximately 43% of the overall time required by the algorithm over all problems. We also see that this is not too different from the average

proportion of total time required by the branch and bound algorithm to find the optimal solution (average of 52% over all the runs). Thus, it is highly probable that one of the first improvements on the GA solution produced by the branch and bound algorithms usually results in the optimal solution. In other words, we could conclude that the GA solution is only a few perturbations away from optimality (although it is an average of 6% away from optima).

8 Conclusions and Further Research

In this paper, we have described two different solution approaches (one approximate and the other, exact) for computing optimal schedules for landing aircraft. We have tested the algorithms on a large data set and provided results of these experiments. From the results, we can conclude that the approximate algorithm performs reasonably for small problems, while the exact algorithm takes a long time to solve larger problems. It is possible that this is more a result of the type of problem we generated. In further experiments, we intend to experiment using real aircraft data to see how the algorithm performs on real data sets.

Given a tolerance percentage, we can terminate the branch and bound algorithm using the current least lower bound in the tree and the current best upper bound. If the relative difference between them is below the specified tolerance the algorithm can be terminated. This would provide a good feasible solution in a reasonable amount of computing time. Although this idea would be more applicable in a breadth-first tree search, it is worth investigating the effects of it in a depth-first search, since, on average, 50% of the time is spent in proving optimality.

The branch and bound algorithm can be easily speeded up by pre-processing. For example, if we can somehow determine that some of the δ_{il} values could never assume a value of 1 in the optimal solution, this would restrict the search space that the branch and bound algorithm has to investigate. Alternatively, if we can fix some of the δ_{il} values to 1 *a priori*, we can further restrict the branch and bound search space. This type of preprocessing is possible with the help of the LP reduced costs that are readily available.

Acknowledgements

The work described in this paper was performed as part of a joint project between the CSIRO Division of Information Technology, the CSIRO Division of Mathematics and Statistics, the Royal Melbourne Institute of Technology, and Griffith University. John Abela is supported by a CSIRO/CITRI Postgraduate Research Award. This project was also part of the Co-operative Research Centre for Intelligent Decision Systems. Thanks go to Magnus Ljungberg (from AAI) for describing the original scheduling problem. We also thank David Sier and Kate Smith, both from DMS-CSIRO, for their help with the preparation of this paper.

References

- Davis.L, (1991), **Handbook of Genetic Algorithms**, Van Nostrand Reinhold.
- Goldberg.D.E., (1989), **Genetic Algorithms in Search, Optimization & Machine Learning**, Addison-Wesley Publishing Company.
- Michalewicz.Z., (1992), **Genetic Algorithms + Data Structures = Evolution Programs**, Springer-Verlag.
- Shanno.D.F., and Weil.R.L., (1971), *Linear Programming with Absolute-Value Functionals*, **Operations Research**, 19, 1, 120–124.
- Soucek.B., *et al*, (1992), **Dynamic, Genetic, and Chaotic Programming**, John Wiley & Sons.