Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-98279

Evolutionary 3D-air traffic flow management

J.M. van den Akker, C.H.M. van Kemenade and J.N. Kok

Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-98279

Evolutionary 3D-air traffic flow management

J.M. van den Akker, C.H.M. van Kemenade* and J.N. Kok**

* CWI, Department of Software Engineering
** Leiden University, Department of Computer Science

This paper is a contribution to the *Handbook of Evolutionary Computation,* IOP Publishing Ltd. and Oxford University Press.

A paper based on the same work appeared in the proceedings of *Parallel Problem Solving from Nature IV,* held in Berlin, September 22-26, 1996.

Division:	Informatics
Issued:	June 1998
Classification of title:	unclassified

Summary

The increasing amount of air traffic requires more efficient use of airspace. One possible way to do this is to use a new planning model which is called the free-route model. New optimization tools are required to create a planning according to this type of model. We present an evolutionary tool to solve free-route Air Traffic Flow Management problems within a three-dimensional airspace. To the best of our knowledge, this is the first (evolutionary) tool that can solve free-route planning problems involving a few hundred aircraft. The performance of the tool has been tested on a set of randomly generated problem instances, where we were especially interested in the robustness of the tool and the scaling of the amount of computation with respect to the size of the problem instances.

Contents

6 Figures

(23 pages in total)

1 Project overview

Air Traffic Flow Management is the general name for the routing and the scheduling of aircraft from half a year before until shortly after departure. An important task is the creation of a planning for trajectories of aircraft in a certain area for a certain time horizon (for example for the next twelve hours). Other tasks include for example dynamic replanning. Though some automated tools are used to make a global planning long before departure, especially the detailed planning just before or after departure is still mainly done manually. Due to the increasing volume of air traffic, improved planning methods and models become necessary, and automated tools to create plans become very useful. One of the new planning models is the free-route model. Under this model the trajectories of aircraft are less restricted than under the more traditional corridor-model. Currently, there are no tools that are able to create good solutions for the free-route model for a large number of flights. In a cooperation between the Centre for Mathematics and Computer Science CWI and the National Aerospace Laboratory NLR the free-route model was studied with the objective to develop a planning tool for this model. Figure 1 shows the environment of such a planning tool. On the left we see the data to be input to the tool. The data consists of a list of

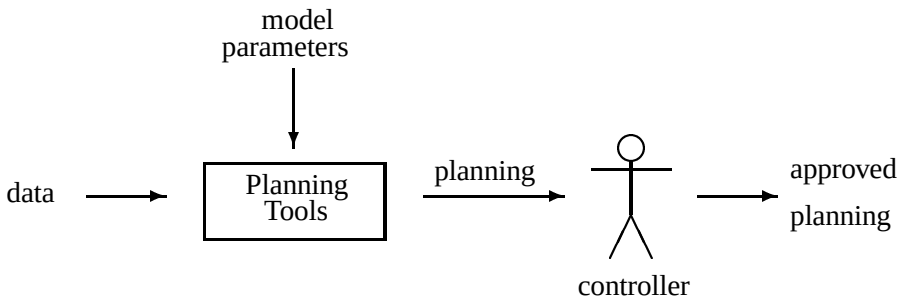

Fig. 1 Environment of the ATFM planning tool

flights that have to be scheduled. For each flight the coordinates of departure, the time of departure and the coordinates of the destination are given. We also have to give the parameters of the free route model. The box represents the planning tool that creates a plan given the current data. The plan describes a schedule for a period of several hours and will be passed to a human controller for approval. Only after this approval will the plan be used.

2 Air Traffic Flow Management Problem

A good reference to Air Traffic Flow Management (ATFM) is the book 7, (Field 1985). A plan describes the trajectories of all involved aircraft. A trajectory defines the exact position of an aircraft as a function of time, so it corresponds to a path with additional temporal information. Two trajectories are conflicting when the separation standards, as stated by the ICAO (International Civil Aviation Organization), are violated. Depending on the area and weather circumstances different separation standards apply. In this paper we use the following standards. The minimal required separation between different routes in the horizontal plane is about 16 nautical miles (1 nautical mile = 1,852 meters), the vertical separation for the higher flight levels is 2000 ft (1 foot = 0.3048 meters). The airspace is partitioned into a set of sectors. A sector can contain a number of layers, called flight-levels. The controller assigned to a sector is responsible for the planning of the trajectories of all the aircraft that fly in this sector. When an aircraft wants to pass a boundary between two sectors the corresponding controllers first have to agree on the time and location of the crossing of the boundary.

An ATFM plan assigns a single trajectory to each aircraft. The primary goal is to choose in such a way that there are no conflicts between aircraft. Secondary targets are:

- to keep all trajectories as short as possible,
- to minimize the number of manoeuvres (especially those manoeuvres that are uncomfortable to the passengers), and
- to have a fair plan with respect to all involved aircraft (i.e. the additional flying distance should be divided over all involved aircraft in a reasonable manner).

Currently ATFM is based on a network model. This model assumes a fixed network of corridors within the airspace, each containing a number of flight-levels. An aircraft is assumed to fly through a corridor from beacon to beacon. Intersections of corridors are always marked by beacons. Only near those beacons, an aircraft is allowed to switch to a different corridor. So this model can be seen as a kind of three-dimensional highway network.

The network model restricts the number of possible trajectories in order to make the planning manageable. In this sense, the network model does not use the available airspace in an efficient manner. Due to the increasing amount of air traffic, the airspace above Europe is almost saturated. Increased accuracy of navigation equipment and the availability of better computers allows for other airspace models. One such model is the free-route model that allows arbitrary shaped trajectories, has a larger degree of flexibility. A different model is the free-flight model, in which the aircraft are autonomous instead of being guided by air traffic control. It is clear that new methods

have to be developed for planning problems in these models. This was the starting point of the research described in this paper.

3 Design process

Currently no practical applications of the free-route model for ATFM exist, hence we have to use artificially generated problem instances. Moreover, if the free-route planning model is going to be used, it might be used in a different form than the form that is studied in the present paper. Therefore one of the requirements is that the tool should be robust in the sense that is able to handle new constraints. Moreover, a certain degree of flexibility is desired. For example, a tool should offer a way to handle soft constraints. A characteristic of the ATFM problems is their dynamic nature: it is possible that additional aircraft have to be added to the plan and planned aircraft can deviate from their planned trajectory. If the current plan violates contraints due to these changes a modified plan has to be created on the fly. The number of affected trajectories should not be too large. It is also desirable that a tool can create alternative plans, among which a human controller can choose. It should also be adaptive in the sense that it can cope with additional constraints imposed by a human controller. Hence, the construction of a tool for the free-route model is a challenge. An evolutionary algorithm is used to have the flexibility described above and to handle the many additional constraints that can arise during the development of a plan.

ATFM and Evolutionary Computation

The first paper about the application of evolutionary techniques to ATFM problems was 1, (Alliot *et al* 1993). In this paper a binary genetic algorithm (GA) was used to do (short-term) conflict resolution. The problems handled involved two or three aircraft and the GA was shown to outperform A^*-search. In France research on conflict resolution continued, and currently they can handle problems involving up to 20 aircraft all having a conflict at approximately the same location 3, (Durand *et al* 1995, 1996a) This genetic conflict solver is used as a part of their so-called ATC test bench. The ATC test bench has been applied to a problem involving 4835 aircraft. It detects all conflicts within a time window of five minutes, partitions the conflicts over independent subsets, applies the genetic solver to each of these subsets independently, and combines the results to obtain a new planning. This procedure is repeated until no further progress is possible. Evolutionary algorithms have also been applied to ATFM problems using the network model 2, (Delahaye *et al* 1997). For each flight a small set of possible routes and departure times is created. An evolutionary algorithm is used to assign the departure time and the route for each flight in order to minimize the workload over all involved sectors. Problems involving up to 3000 aircraft have been handled. Also research on the combination of neural networks and GA's for (short-term) conflict resolution involving two aircraft 5, (Durand *et al* 1996b) was investigated. In Germany at the DLR evolutionary algorithms were used to solve ATFM problems involving the free-flight model and the free-route model 8, (Gerdes 1994, 1995, and 1996).

Our first system was a mutation based hybrid EA that could handle two-dimensional problem instances, so all trajectories were restricted to a horizontal plane. The system could solve problem instances involving up to 20 aircraft within a square 2D sector of size $200 \times 200\, km$ 13, (Kemenade *et al* 1995). Recombination operators were used, but these operators did not increase the performance of the system. This is expected to be a consequence of the strong dependence on the context of the trajectories that are close to other trajectories. A new system was developed that could handle three-dimensional problem instances where the altitude of the aircraft is the additional degree of freedom 14, (Kemenade *et al* 1996). This system could easily find conflict-free solutions to problems involving up to 800 aircraft in a 3D-sector of size $2000 \times 2000\, km$ that were spread over an interval of four hours. These problems can be handled on a standard workstation within roughly 15 minutes of computation. For these larger instances recombination is useful because on these larger problem instances many trajectories are either temporal or spatial independent of each other. We expect that some conflicts can be solved in parallel and the resolutions to these conflicts can then be merged by means of recombination. A more direct way to exploit this kind of parallelism would be to make a decomposition of the problem in a set of smaller independent problems. Unfortunately finding an optimal decomposition is far from trivial and the choice of the decomposition will restrict the set of possible solutions. In order to exploit the potential parallel conflict resolution we apply an evolutionary algorithm with recombination. This system is described in the rest of the paper.

Evolutionary Algorithm

The elitist recombination algorithm 12, (Thierens and Goldberg 1994) is a simple evolutionary algorithm, that involves a direct competition between the offspring and both of their own parents. Figure 2 shows a single iteration of this algorithm. It shows how the next population P_{t+1} is produced from the current population P_t. To the left we see the current population P_t, and

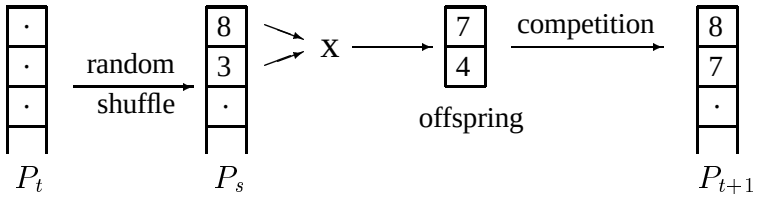

Fig. 2 Elitist recombination

each box represents a single individual. The values in the boxes denote the fitness of the corresponding individuals. An intermediate population P_s is generated by doing a random shuffle of the individuals in population P_t. Partition population P_s in a set of adjacent pairs, and for each

pair apply the recombination operator in order to obtain two offspring. Next, hold a competition amongst the two offspring and their two parents, and transfer the best two out of these four to the next population P_{t+1}. This competition between parents and offspring prevents rapid duplication of relatively fit individuals, and as a result decreases the probability of premature convergence. Elitist recombination resembles the deterministic crowding scheme 11, (Mahfoud 1992), but deterministic crowding lets each offspring compete with only one of its parents, i.e. the most similar parent.

We are using the elitist recombination algorithm as basis for our evolutionary planner. The (population-wide) elitism prevents a decay of the average fitness. Inferior offspring will not enter the population and due to the direct competition between offspring and its parents the duplication of the best few individuals is slowed down. Furthermore elitist recombination tends to be less sensitive to undersized populations than most other evolutionary algorithms 12, (Thierens and Goldberg 1994).

Requirements

The ATFM planning tool should be able to create a plan for an airspace under the free-route model. Such a planning should be free of conflicts, the amount of additional distance traveled by all aircraft should be minimized and the number of manoeuvres should be kept low. The planning tool should have appropriate scaling properties with respect to the number of flights to be scheduled. In order to be applicable such a tool should be able to create a planning for problem instances involving a few thousand flights in less than an hour.

Representation

A single individual represents a complete plan describing the trajectories of all the flights that have to be scheduled. We represent a single trajectory by means of an ordered list of four-dimensional coordinates. Each coordinate is a tuple (x, y, l, t) where x and y represent a location in a two-dimensional plane, l represents the flight-level of the aircraft, and t represents the time of passing the location. An aircraft is assumed to fly in a straight line and at a constant velocity from one coordinate to the next coordinate. A complete plan is represented as:

flight 1	$(x,y,z,t)_{1,1}$	$(x,y,z,t)_{1,2}$	$(x,y,z,t)_{1,3}$	$(x,y,z,t)_{1,4}$	$(x,y,z,t)_{1,5}$
flight 2	$(x,y,z,t)_{2,1}$	$(x,y,z,t)_{2,2}$			
$\vdots$		$\vdots$			
flight n	$(x,y,z,t)_{n,1}$	$(x,y,z,t)_{n,2}$	$(x,y,z,t)_{n,3}$	$(x,y,z,t)_{n,4}$	

Note that the number of coordinates can differ for each trajectory.

The fitness $f(I)$ of the individual I is defined by

$$f(I) = -f_c(I) - \frac{1}{d}\left(\sum_{x \in I} f_m(x) + f_d(x)\right),$$

where $f_c(I)$ is the number of conflicts between trajectories in individual I, $f_m(x)$ is the number of manoeuvres in trajectory x, and $f_d(x)$ is the relative length of trajectory x with respect to the shortest possible trajectory for the corresponding flight. The constant d is a large constant chosen such that the value of the second term is in the range $[0, 1]$. Hence minimization of the number of conflicts is the primary goal, while minimization of the distances and the number of manoeuvres is of secondary importance.

Operators

For each conflict we can identify a set of involved flights. Such a set consists of the two aircraft that are actually in conflict augmented with aircraft that are in their vicinity. If two conflicts involve disjoint sets of aircraft then often it is possible to resolve these conflicts independently. The possibility to resolve conflicts independently and to merge the results, increases the efficiency of the planner significantly. This is the reason to emphasize on the recombination operator in the evolutionary algorithm. Recombination considers the trajectories as atomic entities and therefore recombination will not generate new trajectories, but will only generate new combinations of trajectories.

We first experimented with a uniform recombination operator which took approximately half of the trajectories from the first parent while the other trajectories were taken from the second parent. This operator worked fine for small problems, but did not scale well when increasing the size of the problem. The main reason for this bad scaling behavior is that a uniform recombination operator can not keep large sets of conflict-free trajectories together as the probability that m trajectories are taken from the same parents is $(1/2)^m$.

In order to handle large problems we need a better scaling behavior. Therefore we developed a heuristic recombination operator. This operator starts with an empty plan for the offspring. Then it iteratively selects a nonscheduled flight and a primary parent; both are selected at random. The trajectory corresponding to this flight is taken from the selected parent and it is checked whether this trajectory will introduce a conflict in the current plan of the offspring. If no conflict is introduced then the trajectory is added to the offspring, otherwise the trajectory from the other parent is added to the offspring. This process continues until all flights of the offspring are scheduled. This recombination operator is unbiased in the sense that on average half of the flights will be taken from one of the parents.

The mutation operator is the only operator that introduces new trajectories. A new trajectory is created by making a copy of one of the existing trajectories and add one additional manoeuvre. The mutation operator introduces a detour in a trajectory by changing its heading of the aircraft by $\pm 45^o$, let it fly in this new direction for a random duration, and then change the heading of the aircraft such that it flies to its destination along a straight line. Note that the mutation operator can cancel previously inserted manoeuvres. Figure 3 shows a trajectory obtained after applying the mutation operator one time. It might seem a bit restricted to consider only changes of $\pm 45^o$, but in practice this works well. A much smaller change of heading results in a new trajectory that is quite similar to the non-mutated trajectory, and a much larger change of heading will move the aircraft too far off course. Although this mutation operator is relatively easy to implement one can not

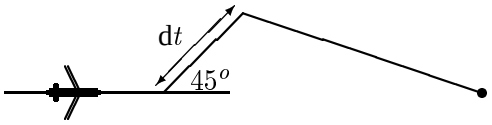

Fig. 3 Trajectory obtained after applying the mutation operator once

guarantee that a single application of the operator will produce a good trajectory. New trajectories that do not resolve any conflicts will rapidly be thrown out of the population by the EA as such trajectories only introduce a penalty and therefore a decrease of fitness.

Evolutionary planner

In order to get an efficient evolutionary planner we have to obtain a good balance between the creation of new trajectories and the assessment of the quality of each of the trajectories. If too few new trajectories are generated then the search will lack diversity and we are likely to get premature convergence. On the other hand a too rapid production of new trajectories will result in a waste of computational effort and will likely result in the introduction of unnecessary complex trajectories.

In order to balance the two processes we have split the inner loop of the evolutionary algorithms in two parts that are alternated. The first part is a elitist recombination algorithm involving recombination only. Here the quality of the available trajectories is assessed by mixing trajectories in order to get a complete plan. The second part contains the mutation operator. Here we take an arbitrary plan and select an arbitrary conflicting trajectory within this plan. Mutation is applied to this trajectory until a better trajectory is obtained within the context of the selected plan or the maximal number of trials for a single mutation is exceeded. So the amount of computational effort used to generate new trajectories will increase when the generation of new trajectories becomes more difficult, thereby resulting in a more constant rate of introducing new trajectories. The fol-

lowing pseudo-code gives a more detailed description of the evolutionary planner.

Input: data, maxEval, N_{gen}, F_{mut}, and N_{try}
Output: best (the best individual found)

```
1  t ← 0;
2  P(t) ← initialize(μ);
4  while (ι(P(t), maxEval) ≠ true) do
       { Part 1: apply Elitist recombination for N_gen generations }
5      for i ← 1 to N_gen do
6          P(t+1) ← ElitistRecombinationStep(P(t));
7          t ← t+1;
       od
       { Part 2: introduce new trajectories by means of mutation }
8      best ← SelectBest(P(t));
9      for i ← 1 to F_mut f_c(best) do
10         plan ← RandomPlan(P(t));
11         flight ← randomConflictFlight(plan);
12         k ← 0;
13         repeat
14             k ← k+1;
15             if k < N_try
16                 then newPlan ← plan − flight + mutate(flight);
17                 else newPlan ← plan − flight + randomStraight(flight);
               fi
18         until f_c(newPlan) < f_c(plan) or k ≥ N_try;
19         if f_c(newPlan) < f_c(plan)
20             then P(t) ← P(t) − plan + newPlan;
           fi
       od
   od
21 best ← SelectBest(P(t));
```

The function $f_c(\text{plan})$ counts the number of conflicts in a plan and randomStraight(flight) creates

a straight-line trajectory for the flight at a random flight level. The number of mutations is related to the number of conflicts in the current best solution by means of the parameter F_{mut}.

Domain knowledge

Domain knowledge is used in several places in our algorithm. During initialization we have to choose an initial set of trajectories for all flights. Because we know that a good plan should involve as few manoeuvres as possible we initialize all plans with straight line trajectories or trajectories involving only a single manoeuvre. More complex trajectories will be introduced by mutation and recombination only if necessary. Domain knowledge is also incorporated in the mutation operator that generates relatively smooth trajectories, that do not contain too large deviations from the straight line trajectories.

4 Development and implementation

The evolutionary planner has been implemented in C^{++}. All experiments were performed on a Silicon Graphics Indy 120 MHz workstation. A single run is terminated as soon as the fitness of the best individual is higher than -1, which indicates that a conflict-free solution is found.

Generation test problems

The free-route model is currently not in use in practice. There is no practical situation nor a golden standard to compare results to. Usually models are compared on the basis of the number of aircraft they can handle on random problem instances. Hence we generated a reasonable set of randomly generated test problems for this model.

We decided to model a square sector of $2000 \times 2000\,km$, containing 12 independent flight-levels. This sector is assumed to be located at an altitude of roughly $10\,km$ above sea-level. Independent flight levels imply that aircraft located in different flight levels will never be in conflict. The source and the destination of the aircraft are 2D-locations, chosen at random within the sector, using a uniform distribution. The flight-level of the aircraft at the given entry and exit locations can be chosen freely.

The entry and exit locations of flights in a plan do not need to correspond to actual locations of airports. The reason for this is as follows. The sector we are modeling is located at an altitude of approximately $10\,km$ above sea level. Our tool plans flights only for higher regions of the airspace. A local Air Traffic Control center at an airport usually manages lower regions of the airspace. When aircraft depart or approach an airport their trajectory is managed by controllers at the airport. When aircraft reach the sector we are interested in, then their locations are already spread over a rather large area. Moreover, in Western Europe airports are relatively close to each other. So the entry and exit locations can be considered as uniformly distributed over the 2D-space.

The evolutionary algorithm can handle a problem where all aircraft have a different, but known, velocity. For convenience we assume that all the aircraft have a velocity of $900\,km/hr$. The flights are to be planned within an interval of 4 hours. The time of entry is selected at random within this interval. This time of entry is accepted if the aircraft can reach its exit location within the interval of 4 hours, when flying in a straight line from the entry to the exit location.

In order to predict the number of conflicts we can use physical models describing the number of collisions between a set of gas molecules in a box per unit of time 6, (Endoh and Odoni 1983).

Application of such a model to our case results in

$$E[\#Conf] = \alpha \frac{t_{hor} d_{sep} v}{s^2 l} n^2.$$

where t_{hor} is the interval of the planning, d_{sep} is the minimal horizontal separation between any two aircraft, v is the average velocity of the aircraft, s is the length of the side of the sector, l is the number of independent flight levels, and n is the number of aircraft. The constant α is introduced to account for the non-uniform distribution of aircraft over the sector. The density of aircraft will be highest near the center of the sector, and lowest near its borders. Within the gas model this constant is one as the gas molecules are distributed uniformly over the complete volume. The validity of the formula has been tested experimentally 14, (Kemenade *et al* 1996).

5 Handling large-scale problems

The expected number of conflicts scales quadratically with respect to the number of aircraft. Therefore it can be beneficial to split a large-scale problem in a set of smaller independent subproblems and merge the solutions of these subproblem to a solution of the large-scale problem. Whether we can find a set of independent subproblems will depend on the set of trajectories that we allow. The class of possible trajectories is determined by the mutation operator. Because the initial population contains trajectories that are only within a single level and the mutation operator does not introduce level changes, the search is restricted in this case to solutions that only involve trajectories located in a single flight level. A population consists of flight plans and within the different flight plans aircraft can have different flight levels. The recombination operator combines trajectories from different plans.

However, quite some efficiency can be gained by splitting the flight levels in disjoint subsets. We can first do an assignment of flights over these subsets, and then use independent evolutionary algorithms for each of the subsets. In this way, each flight is first assigned a restricted number of flight levels, among which a good one is choosen by the evolutionary algorithm. This approach reduces the search space significantly. However simultaneously the set of solutions is reduced.

There are two extremes: take one subset, or take as many subsets as there are flight levels. The first one would take a lot of time and memory, and the second one is too restrictive and does not give very good results. In our experiments we tried different values for the number of subsets. We considered 12, 6, 4, and 3 subsets. In fact, in our formulae and figures we use a variable k that ranges from 1 to 4, where k stands for the number of flight levels within the subproblem. The actual approach can be represented schematically as shown in figure 4. On the left we see all

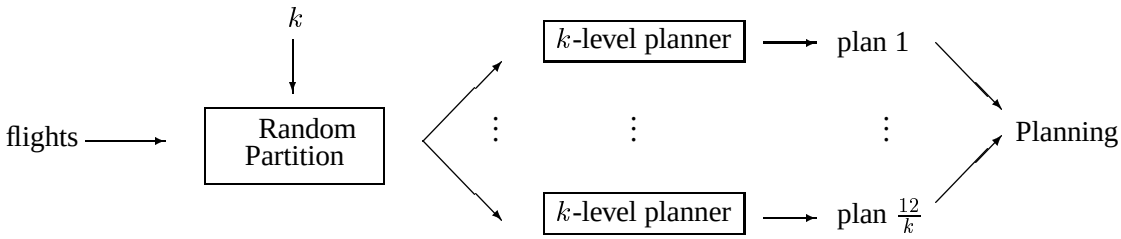

Fig. 4 Approach to solve large scale problems

flights that have to be scheduled. The number of subsets k is given as an input and the flights are randomly partitioned over $12/k$ sets. For each set a k-level planner is applied that schedules the flights within a range of k consecutive flight levels. The resulting $12/k$ plans are combined to a

plan for the 12-level problem.

Performance measures

We were interested in feasibility of tackling free-route problems using evolutionary computation. By feasibility we mean that we were interested to obtain evidence that this type of problem can be solved in a reasonable amount of time and a conflict-free solution will be found with a reasonable probability of success.

Moreover, we are interested in the quality of the solutions. The quality of a conflict-free solution is determined by the expected total number of manoeuvres and by a distance penalty with respect to a situation that all flights go along a path of minimal length.

6 Results

During the experiments we varied the number of aircraft to be planned thereby varying the complexity of the problem instances. We have applied the procedure described in section 5 to partition large-scale problems into a set of subproblems.

In order to measure the expected performance of the system we did run the k-level for different values of k. All results have been extrapolated to a 12-level problem instance.

During a single iteration each of the k-level planners is allowed to perform a single fitness evaluation. If $\sigma'_{iter,k}$ is the standard deviation in the number of iterations used to find a conflict-free solution for a problem with k levels then we can estimate the standard deviation for the 12-level problem by

$$\sigma_{iter,k} = \frac{\sigma'_{iter,k}}{(12/k)^{1/2}}$$

The success-rates are determined experimentally. The 90%-confidence intervals for the success-rate have been computed. Given the confidence interval of the success rate for the k-level planner we can estimate the confidence interval for the 12-level problem when using a set of k-level planners by

$$p_{succ,k} = p'^{(12/k)}_{succ,k}$$

Given the expected value $\mu'_{man,k}$ of the total number of manoeuvres for the k-level subproblems, we have for the whole problem

$$\mu_{man,k} = (12/k)\mu'_{man,k}$$

$$\sigma_{man,k} = (12/k)^{1/2}\sigma'_{man,k}$$

Similar estimates can be used for values $\mu_{dev,12}$ and $\sigma_{dev,12}$ for the cumulated deviations from the shortest paths.

All experiments were performed with an upper bound of 5000 fitness evaluations, a maximal number of allowed avoidance manoeuvres of 12 per aircraft, $N_{gen} = 2$, $F_{mut} = 0.5$, $N_{try} = 10$, $\#I = 16$. All results are averaged over 20 independent runs.

Figure 5 shows the number of iterations and the success rate. The horizontal axis shows the number of aircraft routed in 12 levels. The error bars in the left graph correspond to the standard deviation. These error bars are only given for the case $k = 4$. Note the large standard deviation for the number of iterations. It is remarkable that the actual number of iterations does not change much

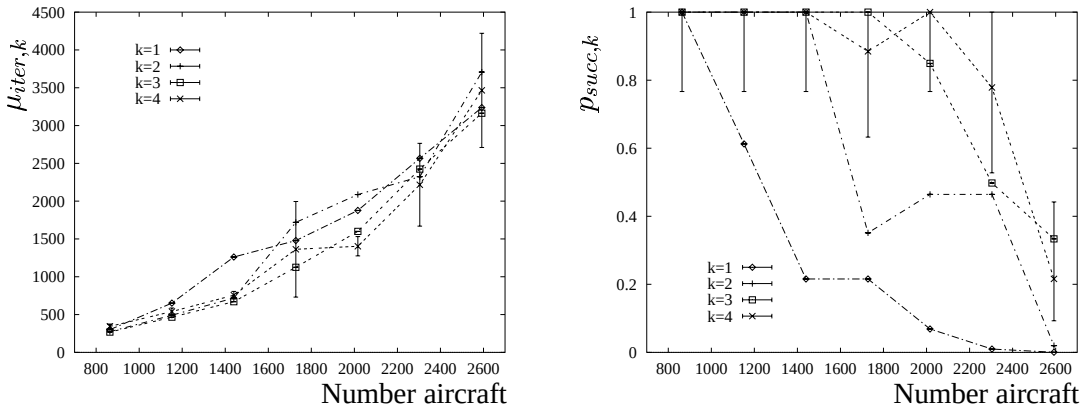

Fig. 5 Number of iterations required (left) and fraction of successful runs (right) when extrapolated to the 12-level planning problem

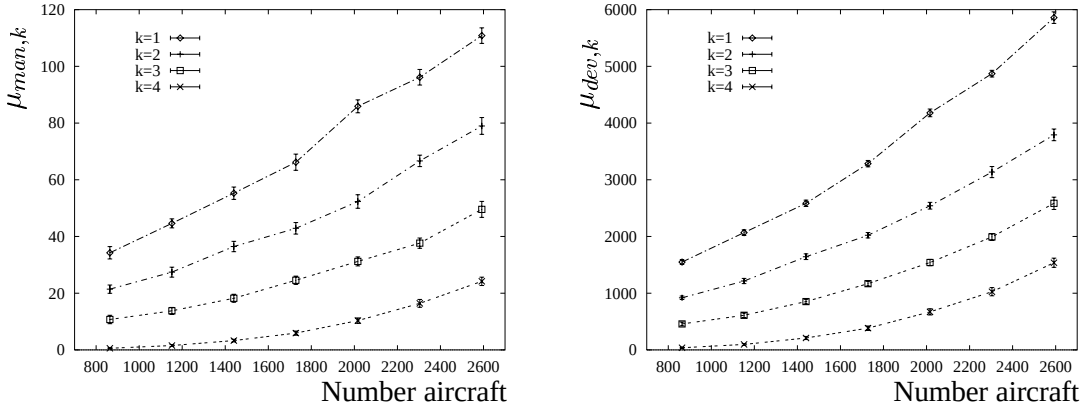

Fig. 6 Average number of manoeuvres(left) and average total deviation (right) when extrapolated to 12-level planning problem

as the the number of levels handled simultaneously increases. The error bars for the success rates (right graph) correspond to the 90%-confidence intervals. Again the error bars are only given for the case $k = 4$. Optimizing multiple levels simultaneously results in a significantly higher success rate. For 2592 aircraft $k = 3$ performs best. We expect this to be a result of the rather arbitrary limit of 5000 fitness evaluations. The experiments for $k = 4$ terminate 10 times by reaching this upper limit while the experiments for $k = 3$ terminate only 6 times by reaching this limit. Figure 6 shows the extrapolated number of manoeuvres and the amount of deviation. As expected, using larger values of k results in a lower value of the number of manoeuvres needed and a smaller total distance being traveled.

7 Conclusions

The free-route planning problem has a search space that grows exponentially in the number of aircraft. To be able to handle large scale ATFM problems it is important to incorporate knowledge regarding the problem domain. We have done so by means of a non-uniform seeding of the initial population and by designing problem-specific evolutionary operators. Introduction of such operators has to be done carefully in order to prevent certain good solutions from being ignored and to prevent premature convergence.

Large problem instances can be handled relatively easily by splitting them in a number of smaller problem instances each involving a limited range of flight levels. We have investigated this approach and observed that optimizing multiple levels simultaneously results in a significant improvement of the probability of finding a solution and also in the quality of the obtained solutions.

Using the current tool we are able to generate a planning involving up to 2592 flights within a 4-hour interval. Straightforward extrapolation suggests that the tool can create a plan for 7776 flights in a 12-hour interval.

References

1. Alliot J, Gruber H, Joly G, and Schoenauer M 1993 Genetic algorithms for solving air traffic control conflicts *Ninth conference on Artificial Intelligence for Applications* (IEEE Computer Society press) pp 338–344
2. Delahaye D and Odoni A R 1997 Airspace congestion smoothing by stochastic optimization *proceedings of Evolutionary Programming VI* ed. Angeline P J, Reynolds R G, and McDonnell J R (Springer) pp 163–176
3. Durand N, Alliot J-M, and Chansou O 1995 Optimal resolution of en route conflicts *Air Traffic Control Quarterly* **3** pp 139–161
4. Durand N, Alliot J-M, and Noailles J 1996a Automatic aircraft conflict resolution using genetic algorithms *proceedings of the* 11^{th} *Annual ACM conference on applied computing* pp 289–298
5. Durand N, Alliot J-M, and Noailles J 1996b Collision avoidance using neural networks learned by genetic algorithms *proceedings of the Ninth International Conference on Industrial & Engineering Applications* pp 595–601
6. Endoh S and Odoni A R 1983 A generalized Model for Predicting the Frequency of air conflicts. *Proceedings of the Conference on Safety Issues in Air Traffic Systems Planning and Design* (Department of Civil Engeneering, Princeton University) pp 226–251
7. Field Obe A 1985 *International Air Traffic Control; Management of the World's Airspace*. (Pergamon Press, Oxford)
8. Gerdes I S 1994 Application of genetic algorithms to the problem of free-routing for aircraft *proceedings of the First IEEE conference on Evolutionary Computation* (IEEE press) pp 536–541
9. Gerdes I S 1995 Application of genetic algorithms for solving problems related to free routing for aircraft *Evolutionary Algorithms in Management Applications* ed. Biethahn J and Nissen V (Springer) pp 328–340
10. Gerdes I S 1996 Application of evolutionary algorithms to the free flight concept for aircraft. *proceedings of the fourth European Congress on Intelligent Techniques and Soft Computing (EUFIT)* pp 1445–1449
11. Mahfoud S W 1992 Crowding and preselection revisited *Parallel Problem Solving from Nature II* ed. Männer R and Manderick B (Springer) pp 27–36.
12. Thierens D and Goldberg D E 1994 Elitist recombination: an integrated selection recombination GA *proceedings of the First IEEE conference on Evolutionary Computation* (IEEE Press) pp 508–512

13. van Kemenade C H M, Hendriks C F W, Hesselink H H, and Kok J N 1995 Evolutionary computation in air traffic control planning *proceedings of the Sixth International Conference on Genetic Algorithms* ed. Forrest S (Morgan Kaufmann) pp 611–616
14. van Kemenade C H M, van den Akker J M, and Kok J N 1996 Evolutionary air traffic flow management for large 3D-problems. *Parallel Problem Solving from Nature IV* ed. Voigt H-M, Ebeling W, Rechenberg I, and Schwefel H-P (Springer) pp 910–919