Optimization of air traffic control sector configurations using tree search methods and genetic algorithms

David Gianazza, Jean-Marc Alliot

OPTIMIZATION OF AIR TRAFFIC CONTROL SECTOR CONFIGURATIONS USING TREE SEARCH METHODS AND GENETIC ALGORITHMS

David Gianazza and Jean-Marc Alliot, C.E.N.A., Toulouse, France

Abstract

Optimization of Air Traffic Control sector configurations using tree search methods and genetic algorithms.

Keywords

Air Traffic Flow Management, ACC schedule, Sector Configuration, Branch&Bound, A*, Genetic Algorithm.

Introduction

The first step of the Air Traffic Flow Management process is to define the predicted schedule for each Air Traffic Control Center one or two days ahead. This is done by forecasting the traffic and by comparing this estimation to the available resources, in order to determine what sector configuration would be the most adapted to the predicted traffic and if some overloads can be foreseen.

In Europe, the ACC schedule is elaborated by the Flow Management Position (FMP) of each ATC Center. Some automated tools are available to help the operators to perform this task. For example the French FMP operator can choose a sector configuration among a set of pre-defined configurations, and match it with the traffic demand so that traffic overloads appear immediately. But apart from his own experience, he has no way of knowing if the chosen configuration is the most adequate, or if another one could better balance the traffic between the manned control positions. An additional drawback of the current method is that it is limited to a set of statically defined configurations, which is a fairly small subset of all the possible ways to combine sectors, as we will see later.

The present paper describes two classical tree search algorithms and a genetic algorithm that take as input the traffic flows and build optimal sector configurations considering the airspace capacity constraints and also the maximum number of control positions that can be manned at each time of the day. This optimization is made in a realistic context, using the airspace description data, the traffic data, the number of available control positions, and the sector capacities of the French ATC centers.

In order to estimate what profit may be expected from optimization, two strategies are then compared in terms of delays and resources, for a peak day of 1999. The first strategy is a slot allocation with the raw traffic demand as input and with constraints issued from the deposited ACC schedules of all French ATCCs. The second consists in smoothing the traffic with all elementary sectors armed, and then find an optimal ACC schedule by combining these elementary sectors.

Related work

The problem of optimizing sector configurations has already been addressed in [2], although with a less realistic model, and also in [1]. In [2], genetic algorithms were used in a toy ATC network to balance workloads by combining sectors, with a chosen number of control positions. Only convex sectors were considered.

In [1], integer programming techniques were used in a realistic context to minimize the sum of traffic overloads (called deficits of capacity) by selecting patterns among a reduced set of statically defined configurations. A pattern is a configuration associated to a time period. The number of control positions is an input parameter, as in [2]. An attempt is made to consider the traffic throughput as a variable, by allowing some macroscopic shifting

of traffic loads along the time axis or from a sector to another. The convergence of the iterative algorithm which is used in that case seems not sure. However, the results show significant improvements when compared to the current manual methods.

The optimization problem addressed in the present paper is different : the optimum we are trying to reach is the sector configuration for which the traffic load is as close as possible to the capacity of each sector or group of sectors of the configuration, so traffic overloads as well as traffic under-loads are considered. The search of an optimal configuration is not restricted to a subset of manually entered configurations, but explores the whole set of possible configurations which can be obtained by combining operational ATC sectors. The number of control positions is a variable of the cost function we are trying to minimize, constrained by an upper limit as there may not always be enough ATC controllers to man all the positions that would be needed.

Model

A "good" configuration is a configuration for which there are no traffic overloads (or the smallest possible ones), and for which the traffic load is balanced as well as possible between the manned control positions, while arming the minimum number of positions. Let us formulate this as a minimization problem.

Let us define the function Δ by $\Delta(x,t)=workload(x,t)-capacity(x,t)$ where x is a sector or group of sectors, t is the time, *workload* is the traffic load (for example N aircraft flying through sector x between t and $t+w$, where w is a chosen time window), and *capacity* is the threshold value for the workload. The capacity of each sector or group of sectors may also depend on specific criteria (like military activity...).

The operator may allow some tolerances around the value of the capacity. These lower and upper tolerances l and u are taken into account in the evaluation of a configuration. For this, we will need to define the overloads and underloads as follows :

$$\Delta_{+}(x,t) = \begin{cases} \Delta(x,t) & if \quad 0 \le \Delta(x,t) \\ 0 & otherwise \end{cases}$$

$$\Delta_{++}(x,t) = \begin{cases} \Delta(x,t) & if \quad \Delta(x,t) > u \\ 0 & otherwise \end{cases}$$

$$\Delta_{-}(x,t) = \begin{cases} |\Delta(x,t)| & if \quad l \le \Delta(x,t) \\ 0 & otherwise \end{cases}$$

$$\Delta_{--}(x,t) = \begin{cases} |\Delta(x,t)| & if \quad \Delta(x,t) < \\ 0 & otherwise \end{cases}$$

Let us then define the following functions:
$N_{pos}(t)$ the number of control positions in the configuration

$$C_{+}(t) = \sum_{x \in config} \Delta_{+}(x,t)$$

$$C_{++}(t) = \sum_{x \in config} (\Delta_{++}(x,t))^2$$

$$C_{-}(t) = \sum_{x \in config} \Delta_{-}(x,t)$$

$$C_{--}(t) = \sum_{x \in config} (\Delta_{--}(x,t))^2$$

The problem will then consist in minimizing the following cost function :

$$cost_{config} = a.C_{++} + b.N_{pos} + c.C_{--} + d.(C_{+} + C_{-}) \quad (1)$$

while respecting the following constraint : $N_{pos}(t) \le M_{pos}(t)$ where a, b, c, and d are chosen factors of decreasing value and M_{pos} is the maximum number of control positions available at each time of the day.

Instead of minimizing $cost_{config}$, and in order to take better account of the relative weights of the different costs, we will in fact maximize $eval_{config}$, such that the k_1 most significant digits of $eval_{config}(x,t)$ refer to the cost C_{++}, the next k_2 digits refer to N_{pos}, and so on:

$$\underbrace{xxxxx}_{k_1} \quad \underbrace{xx}_{k_2} \quad \underbrace{xxxxx}_{k_3} \quad \underbrace{xxx}_{k_4}$$

$$
\begin{aligned}
eval_{config} &= 10^{k_2+k_3+k_4} \times N(k_1, C_{++}) \\
&+ 10^{k_3+k_4} \times N(k_2, N_{pos}) \\
&+ 10^{k_4} \times N(k_3, C_{--}) \\
&+ N(k_4, C_+ + C_-)
\end{aligned} \qquad (2)
$$

where N is a function such that $\lfloor N(k_i,C) = \max(0,(10^{ki}\text{-}1)\text{-}C) \rfloor$ and k_1, k_2, k_3, and k_4 are chosen such that $(10^{ki}\text{-}1)$ is an upper bound of the corresponding cost, if possible.

Problem complexity

The difficulty of the problem is mostly due to the high number of possible configurations which can be built from a set of sectors. A configuration is the mapping of n sectors onto k control positions.

Let us first try to find how many ways there are to part a set of n elements into k subsets. If $P(n,k)$ is this number, it verifies the following equations:

- $\forall$ n≥1
- $P(n,1) = 1$ (one group of n elements)
- $P(n,n) = 1$ (one partition of n groups of one element)
- $P(n,k) = 0$ if $k>n$ (we cannot make more than n groups)
- $P(n,k) = k*P(n\text{-}1,k) + P(n\text{-}1,k\text{-}1)$ if $1<k<n$

The number of all possible partitions will then be $P(n) = \sum_{k=1}^{n} P(n,k)$. For 17 sectors (Brest ATCC), this would give around 83 billion possibilities. However, this value is not realistic: many of the partitions could not be used in an operational context. For example, a partition containing a group in which one sector is a neighbor of no other sector in the group is not a valid configuration. The set of all possible partitions (that we have counted above) could theoretically be obtained by exploring the tree of all possibilities as shown in figure 1 with an example of 5 sectors, although this would become quickly impractical when the number of sector increases.

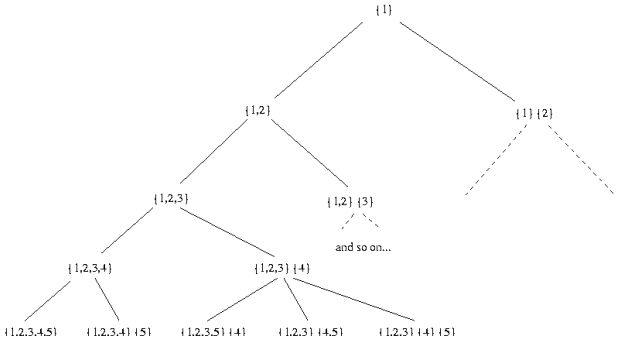


Figure 1: Partitions of a set of 5 elements

In order to estimate the difficulty of the real problem, let us consider only the operational sectors and groups of sectors defined in each ACC database. In figure 1, a tree node was a list of groups under construction. In the case of operational configurations, a node is a list of couples (g,G) (cf figure 2 illustrating the Branch&Bound), where g is a group under construction and G is the set of valid groups compatible with g in the context of the configuration under construction. An element h of G is ``compatible" with g if it contains all the sectors of g, but no sector of the other groups in the configuration (the other "g"'s of the node). If one of the G sets is empty, then there is no need to continue the search from the considered node: it will lead to no valid configuration.

	Number of sectors		Number of partitions	Number of operational configurations
	elementary sectors	operational groups		
Aix	24	42	4.4610^{17}	123965
Bordeaux	22	65	4.4510^{15}	551032
Brest	17	52	8.2810^{10}	14832
Paris(west)	11	17	678570	192
Paris(east)	12	22	4213597	399
Reims	12	17	4213597	249

Table 1: Number of possible sector configurations for the French ATCCs

The methods presented above allow us to count the number of configurations and estimate the difficulty of the problem. The results for the French

ATC centers are shown in table 1, which highlights the combinatorial relation between the number of partitions and the number of sectors.

The number of operational configurations that we can build with the sectors or groups of sectors described in the ATCCs airspace data goes from a few hundreds to a half million. So, we can expect that the use of exhaustive tree search techniques will lead to optimal configurations within a reasonable computation time, at least for relatively small ATC centers and with sector combinations restricted to the groups described in the ATCCs airspace data. However, as the sector configuration optimization problem may be extended to larger ATCCs, and as we may wish to use a wider range of sector combinations, genetic algorithms were also experimented with to solve the problem.

Description of the algorithms

A basic algorithm:

In the previous section, we have presented an algorithm which builds all the possible operational configurations. A basic optimization method consists in evaluating the cost (cf definition 1) of each configuration in order to find the best one. The basic algorithm explores all the configurations. It is memory and time consuming, but complete: we can be sure that it returns the optimum of the evaluation function. So it will be used as a reference for the other algorithms presented hereafter.

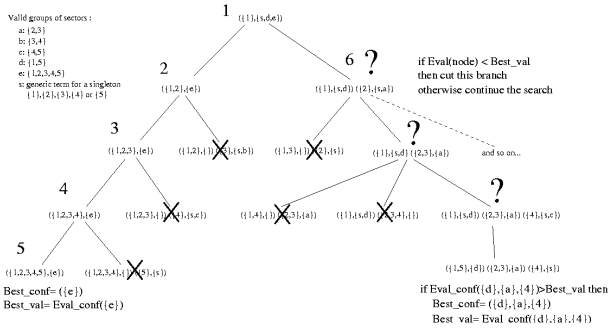


Figure 2: Branch & bound search for an optimal sector configuration

A Branch & bound algorithm:

The idea of the *branch & bound* algorithm illustrated in figure 2 is to avoid exploring every branch of the tree. To do so, it needs to evaluate the nodes in order to decide, by comparison with the best configuration found so far, whether to continue or not the search along a given branch. The cost (resp. evaluation) of a node must be a lower bound (resp. an upper bound) of all the costs (resp. evaluations) of configurations which can be reached from that node. A node of our tree search algorithm is a configuration under construction (see figure 2), i.e. a list of couples (g,G), where g is a group under construction and G is the set of valid groups compatible with g in the context of the configuration under construction.

For a better understanding of this notion of compatibility, let us consider node 6 of the example shown in figure 2. This node is represented by~: ({1},{s,d}) ; ({2},{s,a}). The valid groups compatible with the group under construction {1} are the singleton s, which is {1}, and the group d={1,5}. These groups are the only ones which contain {1} but not {2}.

The cost function for a node is similar to the cost of a configuration. Let us define a function *best* such that *best*(G) returns the element h (a sector or group of sectors) of G, for which the difference *workload*(h,t)-*capacity*(h,t) is the smallest possible underload, or the smallest possible overload if all groups of G are overloaded. We will then define the cost (resp. evaluation) of a node as in definition 1 (resp. 2), except that N_{pos} will be the number of couples (g, G) in the node, and that only Δ_{++} and Δ_{+} will be considered, taking *best*(G) as input instead of an operational sector x of a configuration.

An algorithm inspired from A:*

Like the *Branch & bound*, the A* is a tree search algorithm. However, instead of simply storing the best leaf found so far, all the explored nodes are stored and sorted by valuation.

In order to do so, the A* as described in [4] needs a cost function defining the cost of each state transition, and a function (called heuristic) underestimating the cost of the remaining transitions between the current node and the end of the search.

For our sector configuration problem, we will try to minimize the $cost_{config}$ function described in definition 1, by searching a path in the tree

illustrated in figure 2. But instead of comparing the node evaluation with the evaluation of the best configuration found so far like in figure 2, the nodes already explored are stored into a priority queue sorted according to the node evaluation. The A* will then iteratively consider the node with lowest cost, evaluate its children nodes and insert them in the priority queue, until a leaf of the tree is reached. This leaf is then the optimal sector configuration. The evaluation of a configuration and a node is the same as in 1 and the *Branch & bound* algorithm respectively.

A genetic algorithm:

A genetic algorithm has also been tested to solve the sector configuration problem. The genetic algorithm considers a population of chromosomes, which evolves by crossover, mutation, and selection of the fittest individuals, as described in [3] and [6].

A *chromosome* will be a sector configuration. Each chromosome is composed of several genes. A gene is either an elementary sector or a group of sectors. A fitness value is assigned to each chromosome. The raw fitness f of a configuration will be given by $eval_{config}$ (see definition 2).

The crossover operator splits the two parent configurations and completes each half configuration with the other parent's genes. The resulting incomplete chromosome is then completed with valid groups or sectors. The mutation operator first randomly chooses one gene of the chromosome. Another gene is randomly chosen among a list comprising the first chosen gene and its neighbors (sectors or groups of sectors of the configuration which have a common border with the first chosen sector or group of sector). The sectors of the chosen genes are then recombined into one or several new genes (up to three).

Results

A graphical interface has been developed to display the results of the optimizations. The languages Ocaml and OcamlTk were used to code respectively the algorithms and the interface. The program runs on a PC Pentium IV (1.8 GHz), with Linux as operating system.

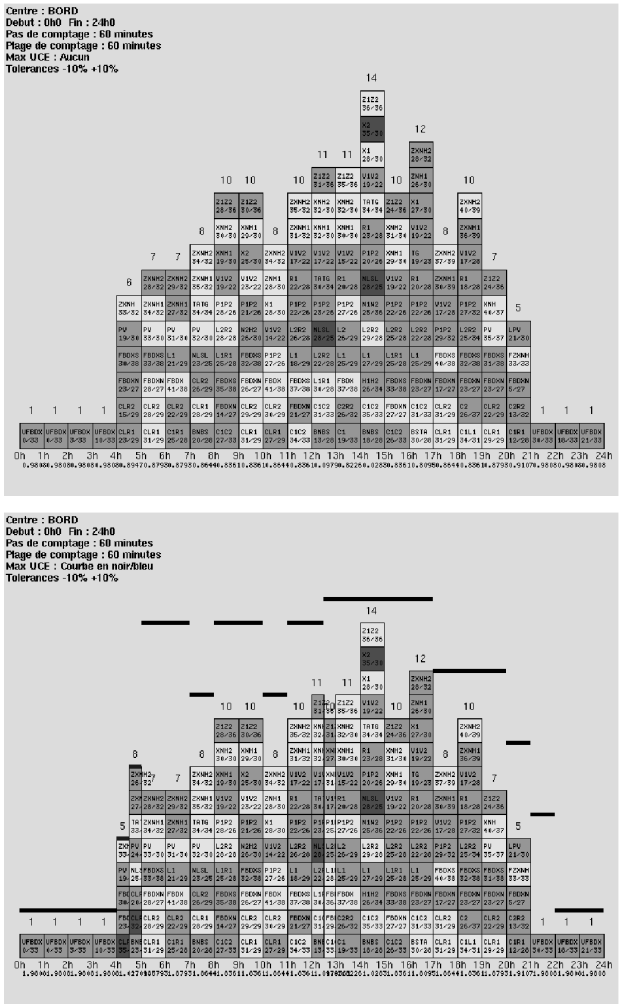


Figure 3: Optimal sector configurations with (down) or without (up) constraints on the number of available positions (example of Bordeaux ACC)

The input parameters are the date, the selected ATCC, the type of traffic (raw demand, final demand...), the capacity tolerances, the maximum number of available control positions, the chosen time step and time horizon (for the computation of traffic flows). The program displays an optimal sector configuration for each time subdivision of the day. The color code is: Green (grey) when the traffic load is under the capacity minus the lower tolerance for the capacity; Yellow (white) when the traffic load is between the margins of tolerance; Red (dark grey) when the traffic load is over the capacity plus the upper tolerance.

The upper part of figure 3 shows the result of the optimization for Bordeaux ATCC, with the raw

traffic demand and with no constraint on the available number of control positions. In this case, the remaining overloaded sectors are necessarily elementary sectors, or operational groups which cannot be split (some elementary sectors are defined and used only to improve flexibility by choosing different ways to combine two of them into an operational group).

The lower part of figure 3 shows the same optimization but with constraints on the available number of positions. These constraints are taken into account in the evaluation functions, by dividing by two the evaluation when the number of control positions in the configuration is greater than the available maximum, thus penalizing these configurations. The constraints used in the lower part of figure 3 are directly issued from the ACC schedule that was sent to the CFMU by Bordeaux FMP for that day. These constraints induce additional overloads (like between 4 and 5 o'clock in the example) on combined sectors which the algorithms are unable to split because there are not enough available control positions.

By comparing the two schedules we see that a few more controllers on the spot could be useful in the early morning. However, this straight-forward analysis is valid only if hourly capacities and traffic flows are significant indicators, which is subject to discussion.

Comparison of the algorithms

The tree search algorithms as well as the genetic algorithm provide optimal sector configurations for each ATCC, verified with the basic algorithm when possible (for Brest, Paris and Reims ATCCs).

	Basic method	branch&bound	A*
Aix	Unfinished	34.06	134.88
Bordeaux	Unfinished	6.06	10.12
Brest	14.44	1.45	10.05
Paris(east)	0.06	0.08	0.08
Paris(west)	0.03	0.07	0.04
Reims	0.02	0.04	0.02

Table 2: Computation times (in seconds) for the deterministic algorithms

The table 2 shows the computation times for the deterministic algorithms, with the raw traffic demand and no constraint on the number of maximum positions. The fastest is the *branch & bound*. The A* is slower because the chosen heuristic is not a very good estimate of what it really costs to reach the best configuration achievable from a given node, thus inducing high backtracking.

For the genetic algorithm, ten different values of the random generator were tested for Bordeaux ATCC, with a crossover probability of 0.6 and a mutation probability of 0.2.

Table 3 shows for each time step of the day and for several sets of population parameters the number of configurations which where different from the optimum. The lines labelled (+1) show the number of occurrences when there was exactly one more control position in the configuration found by the GA than in the optimal configuration. The lines labelled (> +1) show the number of occurrences when the difference was greater than one control position.

When they were not optimal, the solutions found were qualitatively close to the optimum.

	Step	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
220 gen. 120 el..	fails	0	2	0	1	5	2	0	1	3	0	1	0	0	1	0	0
	+1	0	0	0	0	2	0	0	1	0	0	1	0	0	1	0	0
	> +1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
220 gen. 130 el.	fails	0	3	0	0	3	3	0	2	2	0	0	0	0	1	0	0
	+1	0	0	0	0	1	0	0	2	0	0	0	0	0	1	0	0
	> +1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
300 gen. 220 el.	fails	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	+1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	> +1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3: GA results for Bordeaux ATCC

The genetic algorithm is slower than the tree search methods, but provides several optimal or near-optimal solutions. It is not very significant to compare the overall computation times of the genetic algorithm and the tree search algorithm when the minimization problem is easily solved. In

such cases (at night for example), the genetic algorithm will run the same number of generations whatever the difficulty of the problem, whereas the B&B will find an optimal solution among the first branches it explores. Table 4 compares for Bordeaux ATCC the computation times of each time step between 11 a.m and 5 p.m., for 220 generations and a population size of 120 configurations:

Step	11	12	13	14	15	16	17
B&B	0.38	0.54	0.21	0.66	0.59	0.58	0.35
GA	15.75	16.42	15.62	17.41	16.85	15.65	14.98

Table 4: GA and B&B detailed computation times for Bordeaux ATCC

Comparison of optimal and filed ACC schedules

We have seen in figure 3 that, with no constraint on the number of positions, there may remain some overloaded elementary sectors when optimizing the ACC schedule with the raw traffic demand as input. This can be avoided by smoothing the raw traffic demand first, considering that all elementary sectors are armed. Let us compare the following strategies in terms of produced delays and used resources, for a peak day of 1999 (May 21st) :

DEP : delay allocation considering the raw traffic demand and the deposited schedules of the French ATCCs, modified according to the capacities stored in the ACC databases, when these were found different from the declared capacities.

OPT.0.0 : it consists of two steps. The first step is to run a delay allocation with all elementary sectors armed and with zero tolerance on the capacity values. The second step is to find an optimal ACC schedule(s) taking as input the resulting smoothed traffic.

One must be cautious in analyzing the results : capacity values may be slightly different in the filed opening scheme than what was found in the ACC databases. This is due to several factors : in 1999 the capacity databases were not stored on a regular basis and the dataset we used may not correspond to the chosen day. Furthermore, in the deposited ACC schedule, the operator chose among several capacity values which depend on the context of the day (military activity,...). However, this does not change the nature and the validity of the results.

		DEP	OPT0.0	Profit
Delay minutes	Total	211335	65505	69
	Max	460	225	
Cumulated time of activity, in minutes, for all control positions	Aix	19230	18240	5.1
	Bordeaux	14610	11400	19.5
	Brest	14700	8520	42
	Paris(east)	9600	7020	26.9
	Paris(west)	9150	7620	16.7
	Reims	10500	9120	13.1
	All ATCC	77340	61920	19.9

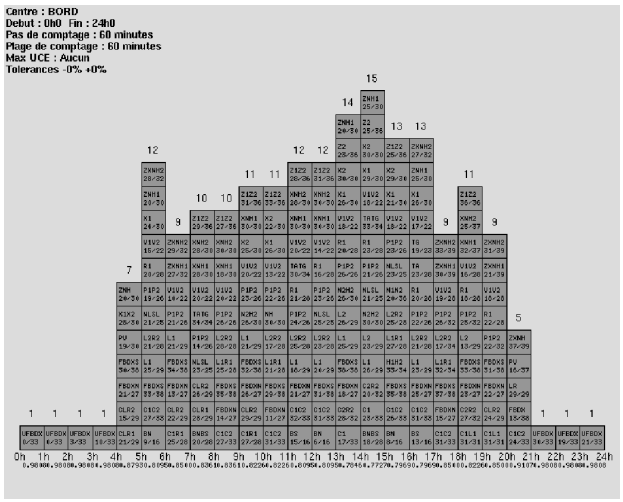
Table 5: Compared slot allocation strategies

We used the basic "SHAMAN" slot allocation method described [5] which aims at minimizing the maximum delay. The slot allocations as well as the ACC schedules optimizations are run with a time step of 60 mn and a time window for the flow computation of 60 mn. The delays generated by each method are shown in the upper part of table 5, and the resources used in each case are detailed for each ATC center in the lower part of the same table. Figure 4 shows the result of the OPT.0.0 strategy for Bordeaux ATCC: on the upper schedule we see that there is no remaining traffic overloads, and on the lower part of the figure we can compare the optimized ACC schedule with the filed one.

Conclusion

The tree search algorithms as well as the genetic algorithm provide optimal sector configurations, using the same parameters and constraints as in the current FMP/CFMU process. The results are computed in a time short enough for an operational use. The tree search algorithms are

faster than the genetic algorithm when applied to the French ATC centers and while restricting the ways to group sectors to a set of operational groups (issued from each ATCC airspace description database).



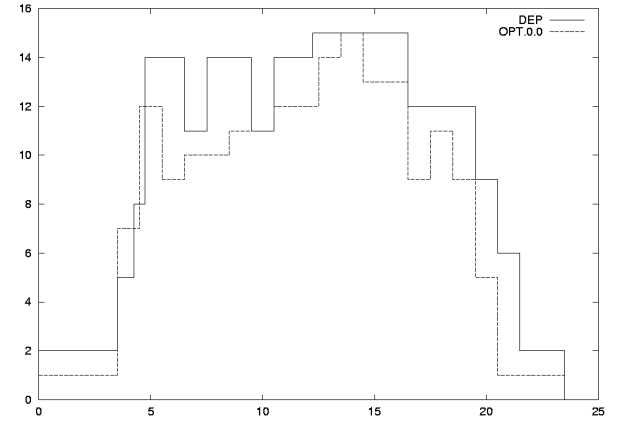


Figure 4:Optimal sector configurations with prior slot allocation (OPT.0.0) for Bordeaux ATCC. Comparison with DEP

However, the combinatorial relation that we have shown between the number of sectors and the number of partitions and configurations is such that the tree search algorithms may prove impractical in the context of larger ATC Centers or if a wider set of operational groups is used. In such a context, the genetic algorithm is a good alternative. It provides several optimal or near-optimal configurations in a chosen computation time.

The results concerning the slot allocation strategies are pretty good : about 70 percent less delays while using about 20 percent less resources. However, they do not take into account the uncertainties on the traffic forecast. Furthermore, the traffic flows and hourly capacities are fairly bad indicators of the controller workload and capacity. Consequently, it is not sure that a slot allocation strategy optimized on the basis of such indicators can lead to a real smoothing of the controller workload. Let us note that these remarks also apply to the current method, although this one has at least the advantage to be validated by an operational feedback.

So far, we have considered the sector configuration optimization only in the context of the pre-tactical ACC schedule estimation, based on the traffic flows and the capacity constraints. The proposed algorithms could as well be envisioned, with adapted workload and constraints definitions, to tactically propose sector configurations to control room managers.

Biography

David Gianazza graduated from the Ecole Nationale de l'Aviation Civile (ENAC) in 1989, obtained a master degree in computer science in 1996. After beginning his career in the operational field in Brest ATC center, he became deputy head of the division ``Computer-assisted control tools'' of the technical service STNA. He is now working in the Global Optimization Laboratory (LOG), a laboratory supported by CENA and ENAC.

Jean-Marc Alliot graduated from the Ecole Polytechnique de Paris in 1986 and from the Ecole Nationale de l'Aviation Civile (ENAC) in 1988. He also holds a Ph.D. in Computer Science (1992). He is currently in charge of the global optimization laboratory of CENA and ENAC in Toulouse.

References

[1] Céline Verlhac and S. Manchon. Optimization of opening schemes. In Proceedings of the fourth USA/Europe Air Traffic Management R&D Seminar, 2001.

[2] Daniel Delahaye, Jean-Marc Alliot, Marc Shoenauer and Jean-Loup Farges. Genetic

algorithms for automatic regroupement of air traffic control sectors. In proceedings of the conference on Evolutionary Programming, 1995.

[3] David Goldberg. Genetic Algorithms. Addison Wesley, 1989. ISBN: 0-201-15767-5.

[4] Judea Pearl. Heuristics. Addison Wesley, 1984. ISBN: 0-201-05594-5.

[5] Nicolas Barnier, Pascal Brisset and Thomas Rivière. Slot allocation with constraint programming: Models and results. In Proceedings of the fourth USA/Europe Air Traffic Management R&D Seminar, 2001.

[6] Zbigniew Michalewicz. Genetic algorithms+data structures = evolution programs. Springer-Verlag, 1992. ISBN: 0-387-55387.