Graph coloring for Air Traffic Flow Management

Nicolas Barnier, Pascal Brisset

Annals of Operations Research 130, 163–178, 2004

Graph Coloring for Air Traffic Flow Management

NICOLAS BARNIER and PASCAL BRISSET {barnier;brisset}@recherche.enac.fr
École Nationale de l'Aviation Civile, 7, avenue Édouard Belin, BP 4055, 31055 Toulouse Cedex 4, France

Abstract. The aim of Air Traffic Flow Management (ATFM) is to enhance the capacity of the airspace while satisfying Air Traffic Control constraints and airlines requests to optimize their operating costs. This paper presents a design of a new route network that tries to optimize these criteria. The basic idea is to consider direct routes only and vertically separate intersecting ones by allocating distinct flight levels, thus leading to a graph coloring problem. This problem is solved using constraint programming after having found large cliques with a greedy algorithm. These cliques are used to post global constraints and guide the search strategy. With an implementation using FaCiLe, our Functional Constraint Library, optimality is achieved for all instances except the largest one, while the corresponding number of flight levels could fit in the current airspace structure. This graph coloring technique has also been tested on various benchmarks, featuring good results on real-life instances, which systematically appear to contain large cliques.

Keywords: air traffic flow management, route network, graph coloring, cliques, greedy algorithm, constraint programming

Introduction

Airspace congestion is today the most critical issue European Air Traffic Flow Management (ATFM) has to face. French en-route Air Traffic Control (ATC) centers capacities are far exceeded by a constant growth in air traffic demand, resulting in ever increasing flight delays.[1] These time and management costs are such a nuisance for all airlines and passengers that the European Commission has released a special statement (European Commission, 1999) acknowledging that current ATFM systems are unable to support high traffic loads and unscalable for the predicted growth. Several measures have been taken at national and European scale to enhance ATFM, among which is the design of a new route network.

The current French (and European) route network has been chronologically designed by small local additions and modifications over time to meet users demand and ATC regulation requirements (and some other constraints like avoiding military areas accordingly to a given schedule). Obviously, such a process does not lead to an efficient solution and underestimates the global capacity of the French airspace. Furthermore, it generates excessive deviations from optimal (direct) routes.

Several research studies have been and are currently being run at Eurocontrol (Letrouit, 1998; Maugis et al., 1998) and CENA (French Civil Aviation Research Center) (Mehadhebi, 2000) to overcome this issue. This work addresses the feasibility of the following "ideal" approach: only consider direct routes (straight lines between origin and destination) for flows of more than a given number of aircrafts[2] and separate

intersecting ones vertically to avoid conflicts (and thus their resolution by ATC). This scheme does not take any additional operational side constraints into account, so that the model used leads to a graph coloring problem.

The graph coloring problem is a well-studied NP-hard problem and an active research topic with many practical applications (see (Trick, 1994) for examples). As it is a difficult problem, especially for large random graphs with poor structure, heuristic techniques are widespread among the best approaches, like greedy algorithms (Brélaz, 1979; Leighton, 1979) or local search (Hertz and de Werra, 1987; Morgenstern and Shapiro, 1986; Fleurent and Ferland, 1994). However, for real-life problems, complete enumeration algorithms can be made very efficient by taking advantage of the structure of the graph. (Coudert, 1997) notices that all the real instances he could find were *1-perfect*, i.e. the size of their maximum clique is equal to their chromatic number. By computing maximum clique with an efficient algorithm, his branch and bound exhibits excellent results on various benchmarks. Constraint Programming (CP) fits well to implement such exact algorithm as reported in (Van Hentenryck, 1990): vertices are simply mapped to variables and edges to disequality constraints. Furthermore, branch and bound is natively provided within (most) CP systems and the versatility of search goals eases the design of search heuristic. Last but not least, additional constraints can be quickly added to the model thanks to the high level of abstraction of CP.

The main idea of our coloring algorithm is based on the search of large cliques by a greedy algorithm during a first stage. This static step provides a lower bound and feeds our constraint program with global "all different" constraints. Then, disequality constraints are posted among the remaining intersecting flows and an optimal solution is searched with a standard branch and bound algorithm. The set of cliques found by the greedy algorithm is also used to guide the search towards the parts of the graph which are the most difficult to color. Through the search, the symmetry between allocated colors is removed by cutting the choice-points associated with the instantiation of a flow to an unused color.

The combination of these techniques allows to obtain optimality proofs for all the instances tried, except the largest one, within short computation time. The optimal number of distinct allocated flight levels for flows of relevant (large enough) sizes could fit within the upper airspace. Our algorithm is efficient on the real-life instances of the DIMACS benchmark (Johnson and Trick, 1996) as well, and compares well on our problem with DSATUR (Brélaz, 1979), a standard greedy coloring method.

The description of the flight level allocation problem and its modeling with flows of minimal given size using constraint programming is presented in section 1. Section 2 then details the use of a greedy algorithm to find cliques in order to improve the efficiency of propagation during the search. The strategy used to obtain the results presented in section 4 is exposed in the preceding section 3. We discuss the relevance of our approach with respect to related work in section 5, which ends with the benchmark. Conclusion and future work are featured in last section.

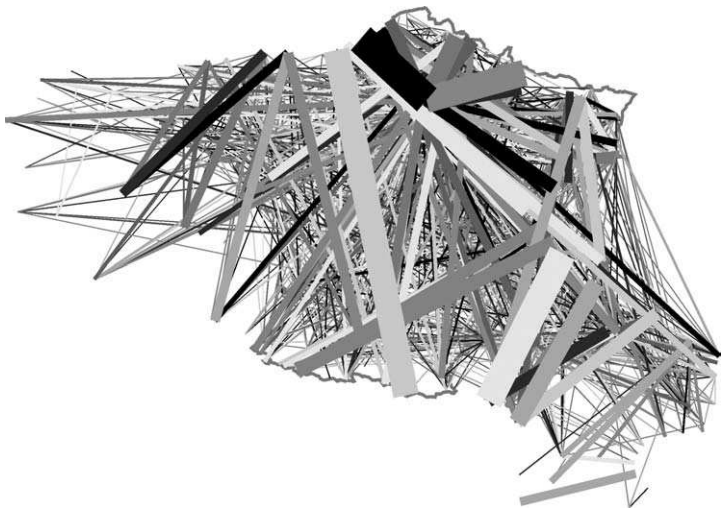

Figure 1. Direct routes (with flow size) in the French Airspace.

1. Problem description and modeling

The design of the routes in the French and European airspace is an incremental process determined by air traffic demand and the evolution of ATC ground facilities, aircraft equipment and flight procedures. The regulation of flights to ensure separation between aircrafts (usually 4–8 NM^3 horizontally and 1000–2000 ft vertically) is handled by Air Traffic Control Centers, which are divided in numerous sectors. The shape of these sectors allows controllers to manage the complexity of the traffic and solve the conflicts that may occur. But the resulting network is made up of a large number of short segments which force airliners to deviate from the direct route joining origin and destination. This Air Traffic Management scheme obviously is not very flexible and is costly for the airlines.

An ideal network, i.e. direct routes for each flight, would be too complex to manage for (human) controllers if the aircrafts were not vertically separated: conflicts occurring in the horizontal plan would form a very tight web, as shown in figure 1 where the width of each route is proportional to the number of flights that takes it. The main idea is therefore to allocate different flight levels to (horizontally) intersecting routes in order to avoid conflicts. With this scheme, flights sharing the same origin and destination would sequentially take the same direct route at a given flight level, distinct from all other intersecting routes. But only aircrafts whose flight time windows overlap will be taken into account for potential vertical separation.

Our model implements this strategy through the notion of flows: flights with the same origin/destination pair are gathered in one flow of size equal to the number of aircrafts involved, and its start and end dates are computed with respect to its earliest and latest flights. For a given day, the flows are precomputed such that the input data of the model are:

- $\mathcal{F}$: the set of flows of cardinality $|\mathcal{F}| = n$;
- r_i: the route of flow i;
- n_i: the size of flow i;
- t_i: the time window of flow i.

The decision variables are the flight levels of the flows:

$$L_i, \quad \forall i \in [1..n]$$

Their common initial domain is $[1..n]$. Finally, the cost to minimize is the number of distinct flight levels:

$$c = card(\mathcal{F})$$

with initial domain $[1..n]$.

However, to control the size of the problem, the set of flows is filtered according to parameter $n_{\min}$: only the flows of more than $n_{\min}$ flights will be kept. Actually, the traffic can usually be divided into two main categories. Daily (or scheduled) commercial flights correspond to most of the big flows and request a high flight level to optimize their fuel consumption. Private IFR[4] flights are responsible for the flows of small size and can reach low flight levels only, so that they hardly interfere with the first category. To take into account this operational behavior, it may be relevant to solve the problem for flows of more than 5–10 flights only, assuming that smaller flows will fly below the lowest flight level granted to the main ones.

The constraint model is straightforward: disequality constraints are posted between the levels of intersecting flows whose time windows overlap.

$$\forall i, j,\ 1 \leqslant i < j \leqslant n, \quad L_i \neq L_j \text{ if } intersect(r_i, r_j) \wedge overlap(t_i, t_j).$$

The objective simply is to minimize the number of distinct allocated levels, such that this model is equivalent to a graph coloring problem. Figure 2 shows a route network and its corresponding graph obtained by mapping routes and intersections to nodes and edges.

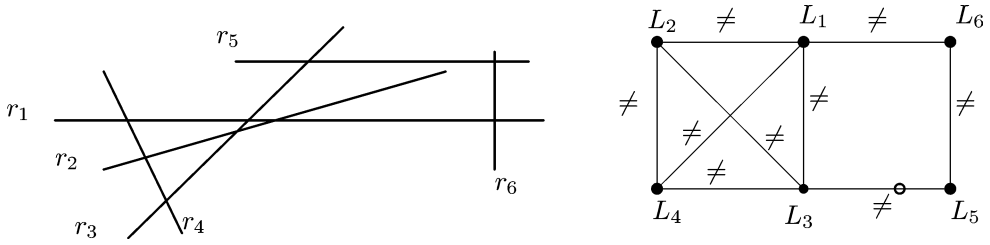

Figure 2. From air traffic network to constraint graph (where $\{L_1, L_2, L_3, L_4\}$ is a clique).

2. Searching for cliques

The preceding constraint model only posts local disequality constraints between two flows. However, for our intersection problem, large complete sub-graphs (cliques) of the induced constraint graph are very likely to occur. They can be used to post global "all different" constraints and thus to detect inconsistencies earlier during the search. Though the search for the maximum clique (the largest one) or all maximal cliques[5] of a graph is well known to be NP-hard, greedy or approximation algorithms can be used to provide a subset of them, assuming that the heuristic will be efficient for the particular structure of the graph.

We use a very simple greedy algorithm, similar to the ones featured in (Bomze et al., 1999), to build a set of cliques, trying to find the largest ones. From each node of the graph, a clique is incrementally built from candidates belonging to its adjacency list, starting with the node which has the largest intersection between its neighbors and the other candidates. This algorithm has the property to produce only maximal cliques so that no strict subset of a generated clique will be returned. However, identical cliques might be discovered through different initial nodes. The set of found cliques is then filtered to obtain distinct cliques only, of size (number of nodes) strictly greater than 2 (as cliques of size 2 will be handled with regular disequality constraints).

The theoretic worst case complexity for this algorithm is $\mathcal{O}(n^4)$ and occurs when the graph is complete, i.e. when the number of edges belongs to $\mathcal{O}(n^2)$. This seems to be a fair estimation for the small instances of our problem. For the biggest ones, only small flows with tight time windows are added, so they only intersect few other flows. For these less dense graphs, the number of candidates to augment a clique dramatically falls when the clique grows. So the complexity becomes much less than the one of the worst case. Figure 3 shows the time to compute the cliques as a function of the number of nodes. Each point corresponds to a value of $n_{\min}$ (the minimum number of flights in a flow, see section 1) ranging from 1 to 20: the plot corresponds to the expected $\mathcal{O}(n^4)$ growth for the small flows, and lesser for the big ones.

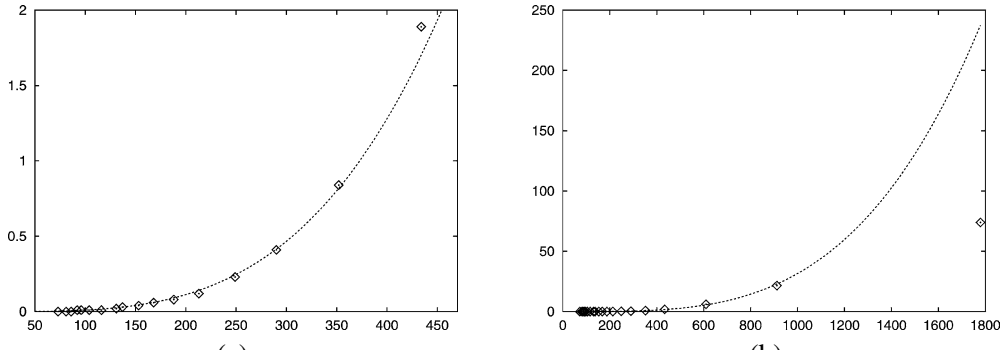

Figure 3. Computation time (seconds on a PIII 700 MHz) of the cliques as a function of the number of nodes n. Complexity in $\mathcal{O}(n^4)$, but lower for the largest instances (b).

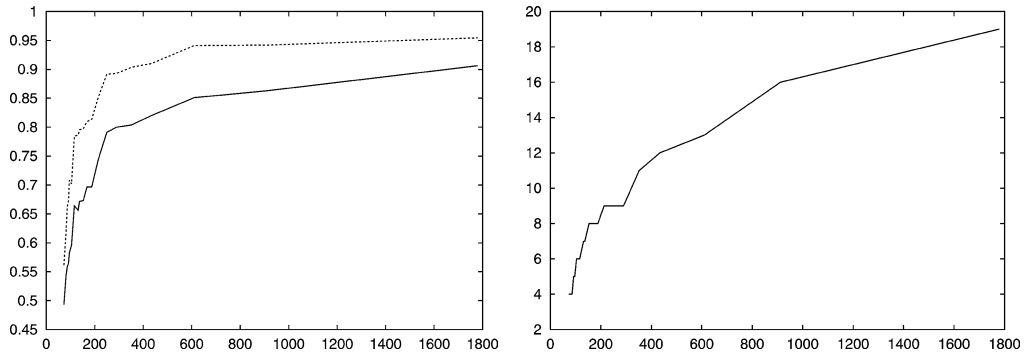

Figure 4. Ratio "*number of cliques*"/n and "*number of distinct cliques*"/n (a), size of the largest clique (b), as functions of the number of nodes n.

Furthermore, a high percentage (between 50% and 90%) of the starting nodes generates cliques which are candidates for posting global constraints (distinct cliques with at least three nodes). This percentage grows with n as shown in figure 4(a) where the ratio between the number of cliques and n is plotted as a function of n, for all found cliques (upper curve) and for distinct cliques (lower curve). Note that only a small number of the cliques are redundant.

For each distinct clique returned by the greedy algorithm, an "all different" constraint is posted, using the bin-matching algorithm provided by FaCiLe. This global filtering provides a more powerful consistency checking than the equivalent set of disequality constraints. Of course, disequalities on variables already involved in a clique are no longer posted to avoid useless (and time-costly) redundancy. The size of the largest clique is also used as a lower bound for the cost variable. Figure 4(b) features the size of the largest clique as a function of the number of nodes: cliques of significant size are found as this lower bound seems to be close from the optimal solution for the instances solved (see sections 4 and 5).

3. Search strategy

The search strategy that seems to be the most efficient of the ones we tried on our problem is guided by the cliques found during the first static step (see section 2). The basic idea is that large cliques correspond to the most constrained part of the flow network and that early instantiation of their variables can produce efficient domain reductions, making advantage of the global "all different" constraints.

The results featured in the next section are obtained by a standard depth first search combined with a branch and bound to minimize the number of flight levels. The largest clique is first deterministically colored, then the other cliques are chosen by decreasing size. All the uninstantiated variables that belong to a given clique are selected with a standard minimum domain size ordering. This strategy is robust enough for most of the problem sizes.

During the search, the symmetry between the remaining equivalent free colors is also broken. When branching on the instantiation of a flow, if there is no more already chosen color in its domain, then the flow is deterministically instantiated to the smallest possible value. Otherwise a choice point is created.

4. Results

The implementation of the search for cliques and the constraint program has been made in Objective Caml (`caml.inria.fr`) with FaCiLe (Barnier and Brisset, 2001), our open source constraint library (`www.recherche.enac.fr/opti/facile`). The results presented in this section were obtained with all the techniques described in the previous sections:

- Greedy algorithm to find distinct cliques of size strictly greater than two flows.
- "All different" constraints on all the cliques.
- Disequalities on the remaining intersecting flows.
- Search strategy based on the cliques.
- Breaking of the symmetry between equivalent colors.

The program was run on the problem with the minimum size of the flows $n_{\min}$ ranging from 1 to 20. All sizes were optimally solved except for the biggest problem, i.e. for $n_{\min} = 1$ (1779 flows, 165859 intersections), with an optimal solution between 20 and 24.

The optimal number of flight levels is plotted in figure 5 (upper curve) as a function of the number of flows, and dots are the corresponding $n_{\min}$ values. Solving the problem for flows of more 5 flights (less than 352 flows) requires no more than 12 flight levels. So with a vertical separation of 1000 ft, the flows involved could fit in the upper airspace, which roughly ranges between 20000 ft and 40000 ft. On the same figure, the lower

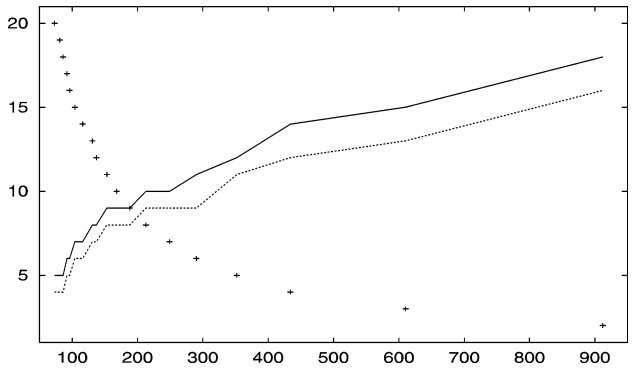

Figure 5. Optimal number of flight levels (upper curve) and size of the largest clique found (lower curve) as a function of the number of flows. Dots are the corresponding minimum size of the flows ($n_{\min}$).

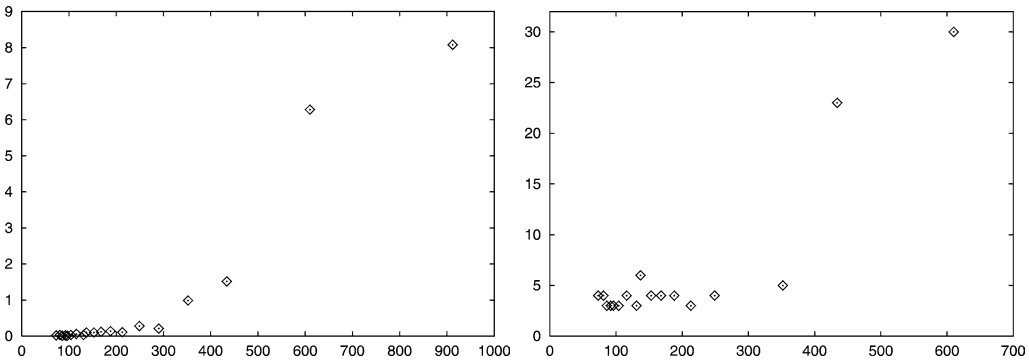

Figure 6. CPU time in second (a) and number of backtracks (b) for optimality proof as a function of the number of flows.

curve shows that the size of the biggest clique found is a good lower bound for the problem.

CPU time for the proof of optimality (a) and required number of backtracks (b) are shown in figure 6. However, for $n_{\min} = 2$ (912 flows, 68759 intersections) the optimal solution could only be obtained by further constraining the cost variable to its optimal value 18 (with largest clique of size 16). Note that this problem size does not appear on the backtracks graph (252 backtracks). Furthermore, the proof of optimality for $n_{\min} = 6$ could only be obtained by using the Brélaz heuristic within the labelling by cliques.

5. Related work

The two following sections compare our algorithm to previous works on route network design for Air Traffic Management and on graph coloring. The third one presents the results obtained on standard graph coloring benchmarks, followed by the performance of DSATUR on our route network problem.

5.1. Route network design for ATFM

The need to increase European airspace capacity has led to various approaches regarding the design of a new route network. The Eurocontrol project TOSCA (Maugis et al., 1998) considers the possibility of such a direct route network. However, the report assesses its intractability for current ATC because the model does not include vertical separation of the flows. It rather focuses on ATFM tools as well as ATC devices (ground and on board) and procedures to add flexibility and capacity within the current network. Yet such measures appear to have poor capacity growth potential as they stick too much to the current "over-constrained" system. An alternative work presented in (Mehadhebi, 2000) tries to simultaneously minimize the ATC complexity within dense areas and the length of the routes: a Voronoi diagram is built around dense crossing points and a complexity indicator of these points is devised to model ATC; local moves are then

performed on this structure to optimize the network. Our approach is quite different, as we get rid of the ATC modeling by ensuring separation with distinct flight levels. However, it is far from existing ATFM procedures and its operational integration would require a lot of additional features (see section 6).

A very different concept that tries to address the same issues is "free flight". This approach allows aircrafts to freely choose their route, then the conflicts are solved in real-time with various algorithms. (Alliot and Durand, 1998) introduces a free flight distributed algorithm which uses priority tokens to order the conflicts and solve each one with an A^* algorithm using simple manoeuvres only. This algorithm seems to be very efficient for the upper airspace. But it is again quite different from our work, as it only focuses on automated distributed ATC, a concept which is also far from the current operational system.

The approach of (Letrouit, 1998) is certainly the most similar to ours. This work includes a technique to recognize subnetworks for which the associated graph is a permutation graph. These graphs admit a polynomial algorithm to compute the maximum clique. Cliques are then used to guide an incomplete coloration algorithm. The proposed algorithms are integrated in an interactive graphic tool for designing a new European air traffic network.

5.2. Graph coloring

As our very simple model leads to a pure graph coloring problem, a well-studied and active research topic, our algorithm can be compared with previous approaches. Because the graph coloring problem is NP-hard, most of them are heuristic methods like greedy algorithms (DSATUR (Brélaz, 1979), RLF (Leighton, 1979)), local search (tabu (Hertz and de Werra, 1987), simulated annealing (Morgenstern and Shapiro, 1986), genetic algorithm (Fleurent and Ferland, 1994)), possibly featuring a certain amount of enumeration. Most techniques generate the coloring either by sequentially assigning a color to all the vertices, or by partitioning the vertices into independent sets mapped to color classes. Some of them also search for a clique to provide a lower bound and initiate the coloring. Loosely related with our search strategy using cliques, (Culberson, 1992) mentions the use of cliques as a tie breaker to order the variables within DSATUR, but without success. Note that finding a maximum independent set or clique is NP-hard as well, such that most techniques used for these tasks are greedy.

These algorithms and refined/hybrid versions (e.g., (Dorne and Hao, 1998) presents a "genetic tabu search" initialized with DSATUR) are successful at finding "good" solutions for large random graphs such as the DIMACS challenge instances (Johnson and Trick, 1996), for which enumeration may have a prohibitive cost. However, complete algorithms relying on enumeration and backtracking techniques have been devised as well to provide optimality or failure proof (yielding lower bounds), like the early works featured in (Christofides, 1971) and (Brown, 1972) and their refined sequels (e.g., Brélaz, 1979; Peemöller, 1983). More recently, Coudert (1997) proposes a complete algorithm which first efficiently finds a maximum clique, then uses an exact sequential coloring.

Table 1
Hardware benchmark.

r100.5	r200.5	r300.5	r400.5	r500.5
0.01	0.23	2.06	12.78	47.78

The author notices that every real-life instances he could find are *1-perfect* graphs. Such graphs have a chromatic number equal to the size of their maximum cliques, so that the search is always bounded by the discovery of a coloring of the same cardinality. Although our problems seem to exhibit the same property,[6] our technique rather takes advantage of the numerous cliques found by the greedy algorithm (the largest one being suboptimal). For such application problems, (Coudert, 1997) reports excellent results. However, graphs which are not 1-perfect heavily impedes the search, and finding a maximum clique still is exponential in the worst case.

Constraint programming has already been used as well to color graphs of "reasonable" size. They are most often implicitly represented, mapping nodes on variables and edges on disequality constraints. Actually, CSPs are often considered as constraints graphs to reason about them, possibly with hyper-arcs to model global constraints. The most popular variable ordering strategies were inspired by heuristics developed for "pure" coloring problems, e.g., see (Smith, 1999) about the Brélaz heuristic (Brélaz, 1979). The search is then generally handled by a branch and bound algorithm to obtain optimality or failure proof. Van Hentenryck (1990) presents this framework with an ordering strategy similar to the Brélaz's one and the same kind of symmetry breaking among colors than the one we use. However, this early approach did not use cliques or global constraints to speed up the search. A very different CSP approach is presented in (Caramia and Dell'Olmo, 2002) to obtain better lower bounds than the one provided by the maximum clique for solving instances which are not 1-perfect. The algorithm is extended to yield upper bounds as well and optimality proofs are obtained when both collapse. However, this technique would be hard to implement using the high level primitives of a standard CP system only.

Unlike previously mentioned ad hoc techniques, the Constraint Programming paradigm provides a high level of abstraction which allows to easily refine the model by adding other various constraints. This is a highly suitable feature for our application (see section 6). One of its important limitations is the computational cost of standard backtracking search on large instances. Various attempts can be found to overcome this issue like the Incomplete Dynamic Backtracking algorithm presented in (Prestwich, 2001) with good results on the DIMACS challenge, but at the cost of loosing completeness.

5.3. Benchmarks

5.3.1. Graph coloring benchmarks

We have compared the performance of our algorithm with various benchmark problems of the DIMACS challenge[7] (Johnson and Trick, 1996). Our hardware is rated in table 1

Table 2
Coloring random instances with cliques.

Name	χ	Best	bt (sol)	CPU (sol)	bt (proof)	CPU (proof)	Clique
le450_5a	5	**5***	225	4.18	228	4.24	4
le450_5b	5	**5***	3564	51	3567	51	4
le450_5c	5	**5***	10	0.79	13	0.89	4
le450_5d	5	**5***	7	0.78	10	0.86	4
le450_15a	15	**15**	3905043	37976	∞	∞	14
le450_15b	15	**15**	26509	240	26512	240	14
le450_15c	15	23	4	5.96	∞	∞	14
le450_15d	15	23	2	6.04	∞	∞	14
le450_25a	25	**25**	0	0.97	3	1.09	24
le450_25b	25	**25**	0	0.95	3	1.05	24
le450_25c	25	28	0	2.00	∞	∞	24
le450_25d	25	28	2	0.96	∞	∞	24
myciel3	4	**4**	0	0.0	5	0.01	2
myciel4	5	**5**	0	0.0	50	0.06	2
myciel5	6	**6**	0	0.02	4417	11	2
myciel6	7	**7**	0	0.06	∞	∞	2
myciel7	8	**8**	0	0.24	∞	∞	2
flat1000_76_0	76	113	33	28	∞	∞	13

with the standard DIMACS machine benchmarks, running the `dfmax`[8] program written by Johnson and Applegate (complete search of the maximum clique).

Results are gathered in tables 2 and 3. They feature the problem name, the chromatic number (if known), the number of backtracks needed to obtain the best solution found and the corresponding CPU time, the number of backtracks needed to reach optimality proof and the corresponding CPU time, and the size of the largest clique found by our greedy algorithm. The search times (excluding data reading and the static greedy search for cliques) are obtained on a PIII 700 MHz running Linux 2.4.5.

Unlike for the route network problems, various heuristics including the Brélaz's one and `dom/deg` (cf. Bessière and Régin, 1996), often combined with the cliques-driven strategy (described in section 3) have been tried, as the latter alone did not provide good results on some instances.

Table 3 reports the results on real-life instances from industrial (register allocation, class scheduling) or academic ("books" graphs, geometric graphs from real maps, etc.) applications: optimality proofs are quickly obtained except on the `queen` problems[9] on which only lower bounds (indicated by LB in the chromatic number column) and upper bounds have been found above size 8. However, the results reported by (Coudert, 1997) and (Kirovski and Potkonjak, 1998) suggest that these problems are very difficult. Coudert does not even consider them as "real-life" problems (but as random ones: not all instances are 1-perfect graphs) or does not mention results above size 9. For the latter, its algorithm takes more than $561 \cdot 10^6$ backtracks and five hours to complete (whereas other real-life problems only take a couple of seconds).

Table 3
Coloring application instances with cliques.

Name	χ	Best	bt (sol)	CPU (sol)	bt (proof)	CPU (proof)	Clique
fpsol2.i.1	65	**65**	0	0.33	3	0.51	64
fpsol2.i.2	30	**30**	0	0.77	3	1.26	29
fpsol2.i.3	30	**30**	0	0.23	3	0.36	29
inithx.i.1	54	**54**	0	0.74	3	1.11	53
inithx.i.2	31	**31**	0	0.58	3	0.89	30
inithx.i.3	31	**31**	0	0.60	3	0.90	30
mulsol.i.1	49	**49**	0	0.08	3	0.12	48
mulsol.i.2	31	**31**	0	0.09	3	0.12	30
mulsol.i.3	31	**31**	0	0.09	3	0.12	30
zeroin.i.1	49	**49**	0	0.08	3	0.13	48
zeroin.i.2	30	**30**	0	0.06	3	0.11	29
zeroin.i.3	30	**30**	0	0.07	3	0.12	29
school1	14	**14**	27	13.23	30	13.36	13
school1_nsh	14	**14**	272	8.91	275	9.04	13
games120	9	**9**	0	0.02	3	0.02	8
anna	11	**11**	0	0.05	3	0.07	10
david	11	**11**	0	0.02	3	0.03	10
homer	13	**13**	0	0.15	3	0.16	12
huck	11	**11**	0	0.01	3	0.01	10
jean	10	**10**	0	0.00	3	0.00	9
miles250	8	**8**	0	0.03	3	0.03	7
miles500	20	**20**	0	0.03	3	0.04	19
miles750	31	**31**	0	0.07	3	0.09	30
miles1000	42	**42**	0	0.09	3	0.12	41
miles1500	73	**73**	0	0.17	3	0.17	72
queen5_5	5	**5**	0	0.00	3	0.00	4
queen6_6	7	**7**	89	0.08	115	0.10	5
queen7_7	7	**7**	122	0.12	125	0.12	6
queen8_8	9	**9**	19088	17	97964	89	7
queen9_9	9	10	77218	453	∞	∞	8
queen10_10	LB: 10	11	12494627	10741	∞	∞	10
queen11_11	LB: 11	13	1855	17	∞	∞	10
queen12_12	LB: 12	14	172119	187	∞	∞	11
queen13_13	LB: 13	17	20	0.67	∞	∞	12
queen14_14	LB: 14	17	4597	4.91	∞	∞	13
queen15_15	LB: 15	18	5094	84	∞	∞	14

These results are obtained with the full algorithm. However, we tried to separately observe the effects of the global "all different" constraints and of the cliques-driven labelling because they are supposed to be complementary: their combination should allow earlier pruning by selecting variables involved in the biggest global constraints. Surprisingly, for a few instances, optimality could only be achieved by the cliques-driven heuristic when the global constraints were disabled, therefore emphasizing the interest of computing a set of large cliques.

Table 4
Performance of DSATUR on the route network problem.

Size	nb vars	nb edges	DSATUR	Cliques
10	168	2151	9	9
9	188	2720	9	9
8	213	3562	10	10
7	249	5156	10	10
6	290	7274	11	11
5	352	10581	13	**12**
4	434	16106	15	**14**
3	610	33059	16	**15**
2	912	68759	21	**18**
1	1779	165859	24	24

But our algorithm less efficiently behaves on random graphs which often have a low maximum clique cardinality (not 1-perfect) or many sub-optimal colorings. The size of the problem impedes the search as well, because standard backtracking search does not scale well (especially on problems with poor structures (Smith and Grant, 1995)).

5.3.2. DSATUR on the route network problem

We finally report in table 4 the performance of a version of DSATUR, a standard greedy coloring algorithm, implemented by Culberson.[10] For each flow size are presented the corresponding number of vertices (or variables) and edges, and the size of the best coloring found by DSATUR and by our algorithm. We show the result of the best heuristic available with this implementation for each problem size, but note that a more sophisticated version of the algorithm could provide better results. All the computation times fall below one tenth of a second on the same system than the one used for the benchmarks. However, DSATUR cannot provide optimality proof or finds larger colorings than our algorithm on large instances (typeset in bold in table 4).

6. Conclusion and future work

A simple model to handle the air traffic on a direct route network over France has been presented. This model, unlike most other approaches (Maugis et al., 1998; Mehadhebi, 2000), statically avoids conflicts and ATC constraints by vertically separating intersecting flows. The resulting optimization problem is an allocation of flight levels to flows (aircrafts sharing the same origin and destination) and its size can be tuned by restricting the minimum size of the flows. This allocation problem is equivalent to a graph coloring minimization problem and this paper presents a combination of techniques to tackle it.

First, a greedy algorithm is used to find maximal cliques on which global constraints are posted. The algorithm reveals to be efficient on the structure of our problem: a large number of cliques are found and the size of the biggest one is a good lower bound of the minimum number of flight levels. Then a variable ordering based on the set of

cliques returned by the first step is used in a branch and bound to search for the minimal number of flight levels. During this search, the symmetry between the equivalent flight levels is broken when the instantiation of a flow to a new level occurs.

With an implementation in FaCiLe of these techniques, we were able to optimally solve in short computation time most of the sizes of the problem on a typical day of traffic, except for the biggest one. The optimal solutions for flows of "operational" size suggest that the traffic could fit within the upper airspace. This traffic would be separated from the flows of smaller size assumed to fly at lower flight levels and differently handled by ATC. We have also shown that our method behaves well on real-life benchmark instances for which "large" cliques can be found. It also finds better solutions than DSATUR on the large instances of the route network problem.

However, this model is over-simplified and only aims at providing a feasibility study for long-term innovative redesign of ATC like the "sectorless" project[11] (Gawinowski and Guichard, 2002). To be more realistic, some additional constraints may be taken into account. For example, aircrafts have a preferred flight level to optimize their performances, depending mostly on the type of the aircraft and on the length of the route. This could be modeled by restricting the domains of the flows or adding the preferences in the cost. Current flight procedures, like the use of existing beacons, the avoidance of military areas or the parity of the flight level depending on the heading, may also be considered to allow a smoother integration of a new route network. The first two procedures may be taken into account in the horizontal precomputing of the route. The routes would then be made of a limited number of segments instead of straight lines, but the resulting intersection graph should not contain less or smaller cliques. The third one can be modeled simply by constraining the parity of flows. Note that adding some of these constraints to the model would make the flight levels non-equivalent, so that the symmetry breaking technique could not be used anymore. Furthermore, French and European air traffics are tightly coupled, so that the operational implementation of a direct route network should affect at least neighboring countries. The problem should be globally solved but optimality may be unreachable because of the huge size of the resulting instances, so alternative local or incremental CP methods could be more efficient.

Further work includes the use of more efficient techniques to find larger cliques, the design of criteria to dynamically select the cliques during search or use them inside standard strategies, the addition of more flexibility to the model by taking into account some of the previously mentioned constraints, and trying to solve it at an European scale. A further considered refinement of the model would be to allow the splitting of flows with noncontinuous time windows to change their allocated flight level during the day.

Acknowledgment

The authors would like to thank the referees for their very useful comments.

Notes

1. Note that the situation is quite different in the US where congested areas subject to ATC regulations are mostly located within terminal sectors around big airports (vs. en-route sectors for Europe).
2. Smaller flows only would then be managed by ATC. If all the flows are considered (i.e. with at least one flight), no ATC would ideally be required any more.
3. 1 NM = 1852 m.
4. Instrument Flight Rule: flight for which a flight plan must be posted and which is systematically controlled by ATC.
5. A maximal clique has the property that it is not a strict subset of any other clique.
6. A complete algorithm has been used to check this property for the small instances.
7. `ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/`.
8. `http://www.cs.sunysb.edu/~algorith/implement/dimacs/distrib/color/graph/`.
9. These graphs correspond to a square chessboard of size $n \times n$ on which n sets of n queens must be placed such that no pair of queens within the same set is in conflict.
10. `http://www.cs.ualberta.ca/~joe/Coloring/Colorsrc/`.
11. A controller would not be responsible of one single small sector anymore, but of an entire flow from origin to destination.

References

Alliot, J.-M. and N. Durand. (1998). "Faces: A Free Flight Autonomous and Coordinated Embarked Solver." In *Proceedings of the Second Air Traffic Management R&D Seminar ATM-2000*, Orlando, USA. Eurocontrol & FAA.

Barnier N. and P. Brisset. (2001). "FaCiLe: A Functional Constraint Library." In *Colloquium on Implementation of Constraint and LOgic Programming Systems CICLOPS'01 (Workshop of CP'01)*, Paphos, Cyprus, December.

Bessière, C. and J.-C. Régin. (1996). "MAC and Combined Heuristics: Two Reasons to Forsake FC (and CBJ?) on Hard Problems." In *Principles and Practice of Constraint Programming*, pp. 61–75.

Bomze, I., M. Budinich, P. Pardalos, and M. Pelillo. (1999). "The Maximum Clique Problem." In D.-Z. Du and P.M. Pardalos (eds.), *Handbook of Combinatorial Optimization*, Vol. 4. Boston, MA: Kluwer Academic.

Brélaz, D. (1979). "New Methods to Color the Vertices of a Graph." *Communications of the ACM* 22, 251–256.

Brown, R.J. (1972). "Chromatic Scheduling and the Chromatic Number Problem." *Management Science* 19, 451–463.

Caramia, M. and P. Dell'Olmo. (2002). "Constraint Propagation in Graph Coloring." *Journal of Heuristics* 1(8), 83–108.

Christofides, N. (1971). "An Algorithm for the Chromatic Number of a Graph." *Computer Journal* 14, 38–39.

Coudert. O. (1997). "Exact Coloring of Real-Life Graphs is Easy." In *Design Automation Conference DAC97*, Anaheim, CA, June.

Culberson, J.C. (1992). "Iterated Greedy Graph Coloring and the Difficulty Landscape." Technical Report TR 92-07, University of Alberta, Edmonton, Canada.

Dorne, R. and J.K. Hao. (1998). "Tabu Search for Graph Coloring, T-Colorings and Set T-Colorings." In *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, chapter 3, pp. 77–92. Dordrecht: Kluwer Academic.

Fleurent, C. and J.A. Ferland. (1994). "Genetic and Hybrid Algorithm for Graph Coloring." Technical Report, Université de Montréal.

Gawinowski, G. and L. Guichard. (2002). "Sectorless-1 Operational Concept Document 0.1." Technical Report, Eurocontrol Experimental Centre.

Hertz, A. and D. de Werra. (1987). "Using Tabu Search Techniques for Graph Coloring." *Computing* 39, 345–351.

Johnson, D.S. and M.A. Trick (eds.). (1996). *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, DIMACS Series on Discrete Mathematics and Theoretical Computer Science. Providence, RI: Amer. Math. Soc.

Kirovski, D. and M. Potkonjak. (1998). "Efficient Coloring of a Large Spectrum of Graphs." In *Design Automation Conference*, pp. 427–432.

Leighton, F.T. (1979). "A Graph Colouring Algorithm for Large Scheduling Problems." *Journal of Research of the National Bureau of Standards* 84(6), 489–503.

Letrouit, V. (1998). "Optimisation du réseau des routes aériennes en Europe." Ph.D. Thesis, Institut National Polytechnique de Grenoble.

Maugis, L., J.-B. Gotteland, R. Zanni, and P. Kerlirzin. (1998). "TOSCA-II - WP3: Assessment of the TMA to TMA Hand-over Concept." Technical Report TOSCA/SOF/WPR/3/03, SOFREAVIA.

Mehadhebi, K. (2000). "A Methodology for the Design of a Route Network." In *Proceedings of the Third Air Traffic Management R & D Seminar ATM-2000*, Napoli, Italy, June. Eurocontrol & FAA.

Morgenstern, C. and H. Shapiro. (1986). "Chromatic Number Approximation Using Simulated Annealing." In *ACM Mountain Regional Meeting Proceedings*.

Peemöller, J. (1983). "A Correction to Brélaz's Modification of Brown's Coloring Algorithm." *Communications of the ACM* 26, 595–597.

Prestwich, S. (2001). "Local Search and Backtracking versus Non-systematic Backtracking." In *Papers from the AAAI 2001 Fall Symposium on Using Uncertainty within Computation*, pp. 109–115. Cape Cod, MA: North Falmouth.

Smith, B.M. (1999) "The Brélaz Heuristic and Optimal Static Orderings." Research Report 1999.13, School of Computer Studies, University of Leeds, June.

Smith, B.M. and S.A. Grant. (1995). "Where the Exceptionally Hard Problems Are." Technical Report 95.35, University of Leeds.

The European Commission. (1999). *Fifteen Countries, a Single European Sky*. IP/99/924.

Trick, M. (1994). "Network Resources for Coloring a Graph." `mat.gsia.cmu.edu/COLOR/color.html`

Van Hentenryck, P. (1990). "A Logic Language for Combinatorial Optimization." *Annals of Operations Research* 21, 247–274.