Airspace block organization with metaheurisitics and partitioning packages

Charles-Edmond Bichot, Nicolas Durand

Airspace block organization with metaheurisitics and partitioning packages

Charles-Edmond Bichot and Nicolas Durand
Laboratoire d'optimisation globale
ENAC - DSNA/DTI-SDER — Toulouse, FRANCE
{lastname}@recherche.enac.fr

***Abstract*— In this paper, different metaheuristics applied on an air traffic control problem. This problem is a graph partitioning problem. It can be solved by classical methods which are spectral and multilevel methods. State-of-the-art public-domain graph partitioning packages, CHACO and METIS are used to resolve it. A comparison between results return by these packages and metaheuristics implementations is made for different objective functions of the literature. Metaheuristics used are simulated annealing, ant colony and a new one called fusion fission developed in the LOG laboratory. Experimental results show that metaheuristics find better results than classical packages.**

I. Introduction

In this paper, different metaheuristics applied on an air traffic control problem are presented. This problem is a graph partitioning problem. It can be solved by classical methods which are spectral and multilevel methods. State-of-the-art public-domain graph partitioning packages, CHACO [1] and METIS [2] are used to resolve this problem. A comparison between results return by these packages and metaheuristics implementations is made for different objective functions. However, whereas classical methods are designed for particular objective function, that can hardly be modified, metaheuristics can easily be changed of objective function, which is an advantage for this problem.

The air traffic control problem is presented in section II. Different objective functions already used to solve partitioning problems are described in section III. In section IV are shortly presented three metaheuristics, which are simulated annealing, ant colonies and a new one developed in the LOG laboratory, fusion fission. The comparison between results of metaheuristics and classical libraries is presented in section V.

II. An Air Traffic Control problem

"The primary purpose of the Air Traffic Control (ATC) system is to prevent a collision between aircraft operating in the system and to organize and expedite the flow of traffic" [3]. The first objective of ATC is safety, the second is efficiency.

The Functional Airspace Block Optimized Process (FABOP) project consists in dividing the European airspace into blocks. Let us explain some air traffic control mechanisms. An air traffic controller supervises the traffic in an limited area, called air traffic sector. Controllers are qualified for working on a set of sectors, which is called a functional airspace block. The FABOP project consists to partition the European airspace into functional airspace blocks. Actually, just a few blocks are crossing countries' frontiers. In this paper we study a new organization of blocks based on flows of aircraft and not on countries' frontiers.

Controllers only know air traffic sectors on which they are qualified, and rarely other sectors. Because "controller-controller coordination is easier and more effective inside an ATC unit (a block) than between ATC units" [4], *we try to maximize flow of aircraft inside blocks and to minimize flow of aircraft between blocks.* For human factors, *each block must have the same "size".* This is the aim of the objective function to minimize.

Let us describe more formerly the problem. We compare each air traffic sector to a vertex and each flow of aircraft between sectors to an edge. We have a graph $G = (V, E)$ with a vertex set V and edge set E. Each edge $e = (v_1, v_2)$ has a weight $w(e)$ which is the flow of aircraft which fly from v_1 to v_2 and *vice versa.* Each vertex v_i has a weight, which is the sum of the weights of connected edges plus aircraft departures and landings if the sector is connected with an air-

port. A partition of G into k distinct partitions must respect the following constraints : $P_k(G)$ must be a partition of G into k non-empty subgraphs $V_1, \ldots, V_k$ with $\forall\ i, j$ included in $1 \ldots k, i \neq j, V_i \cap V_j = \varnothing$ and $\bigcup_{i=1}^{k} V_i = V$.

The problem is to partition the vertices of the graph G into k roughly equal parts (or blocks), such that a certain objective function is optimized. Intuitively, equal parts are parts with the same number of vertices, *ie.* blocks should have the same number of sectors. Because the number (flow) of aircraft which go through a sector is very different for each sector, we can not use this equality between parts. More probably, comparing the amount of flows of aircraft between the sectors of a block with the same amount of flows with another block is better. The number of conflicts or potential conflicts in a sector increases considerably the difficulty for a controller to coordinate aircraft. Thus, the computation of the equality should take into account the number of conflicts or potential conflicts. The size and the design of a sector influence the number of coordinations, maybe the computation of the equality should take into account this too. Other operationals constraints can be easily find. As we can see, finding what is the equality criteria between parts is difficult.

In the experimental tests, the number of vertices is $|V| = 759$ and the number of edges is $|E| = 3,165$. This number of vertices is the number of air traffic sectors of the European countries core area defined in [5]. This area is the set of countries which have the highest flows of traffic in Europe. The countries core area is composed of Germany, France, United Kingdom, Switzerland, Belgium, Netherlands, Austria, Spain, Denmark, Luxembourg and Italy.

We have seen that finding equal parts for this problem is difficult. Indeed, like image segmentation, we should use objective functions which itself equalize the different parts of the partition.

III. Objective functions

In this section we present objective functions used for partitioning problems. The simplest and the oldest of them is the *Cut* function, which is the sum of edges' weight between partitions. This objective function is designed for spectral graph partitioning. Let $A \in P_k(G)$, $V - A = \{u \in V, u \notin A\}$, we define

$$cut(A, V - A) = \sum_{u \in A, v \in V - A} w(u, v) \qquad (1)$$

and

$$W(A) = \sum_{u \in A, v \in A} w(u, v) \qquad (2)$$

Thus,

$$Cut(P_k(G)) = \sum_{A \in P_k(G)} cut(A, V - A) \qquad (3)$$

Hagen and Kahng [6] defined the ratio cut :

$$Rcut(P_k(G)) = \sum_{A \in P_k(G)} \left(\sum_{B \in P_k(G) - \{A\}} \frac{cut(A, B)}{|A|} \right) \qquad (4)$$

which removes the requirement $|A| = |B|$ and minimizes $cut(A, V - A)$ when the number of vertices in each part is roughly equal.

Shi and Malik [7] propose the normalized cut :

$$Ncut(P_k(G)) = \sum_{A \in P_k(G)} \frac{cut(A, V - A)}{cut(A, V - A) + W(A)} \qquad (5)$$

which minimizes $cut(A, V - A)$ while maximizing $Assoc(A, V) = cut(A, V - A) + W(A)$, the sum of the weights of each partition.

And the min-max cut function was introduced in [8] :

$$Mcut(P_k(G)) = \sum_{A \in P_k(G)} \frac{cut(A, V - A)}{W(A)} \qquad (6)$$

which minimizes $cut(A, V - A)$ while maximizing $W(A)$ which is the sum of the weights of edges between vertices of the same partition.

The $Mcut$ objective function seems the most appropriate of the objective functions presented, regards to the objective presented in section II : maximizing flows of aircraft (edges weight) inside blocks (partitions) and minimizing flow of aircraft between blocks.

IV. Metaheuristics

The three metaheuristics described in this section are more precisely explained in [9]. More information about a new method called fusion fission can be found in [10].

A. Simulated annealing

This metaheuristic was first introduced in [11]. Let us present very shortly the simulated annealing method. The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution in order to escape from local minima.

The probability of doing such a move is decreased during the search. In metallurgy, a very hot metal is cooled very slowly to increase its solidity. In the same way, vertex are moved among partitions, one by step. The temperature T is decreased when moves are increasing the objective function ($T\frac{t_{max}-t_{min}}{t_{max}}$ with $t_{max}(t_{min})$, maximal (minimal) temperature). The objective function e gives the "energy" of the solution. The result of a move of a vertex which changes of partition, builds a new move s'. s' is kept if $\exp\frac{e(s)-e(s')}{T}$ is upper a random number in $[0,1]$ or if the objective function is increasing. Else, the old state s is kept. The algorithm stops when $T \leq t_{min}$.

B. Ant colony

Ant colony optimization is inspired by the foraging behavior of real ants. This metaheuristic proposed by Dorigo is explained in [12]. The algorithm uses the ability of ants to find the shortest path between food source and their nest. While walking from food sources to the nest and *vice versa*, ants deposit a substance called pheromone on the ground. When they decide a direction to go, they choose with a higher probability paths that are marked by stronger pheromone concentrations.

The ant colony algorithm is based on three (one optional) steps. While a termination condition is not satisfied, the three steps are executed, but not necessarily in the following order :

- The first step is the ants motion. Ants are moving through nodes of the graph G by applying a stochastic local decision policy which uses pheromone values and a local heuristic. While moving, the ant keeps memory of the path it was walking on the graph.
- The second step consists in updating pheromones. Ants always update the pheromone trails they are using. But if a path lead to "food" (a local solution), the ant can update backward the path it used by using its memory. Finally, like real pheromones, pheromone trail intensity decreases over the time (to avoid convergence into a sub-optimal region).
- The last step is optional. It is used to implement centralized actions which cannot be performed by single ants.

The adaptation of ant colony to the partitioning problem uses k colonies, one for each partition of the graph. These colonies are competing for food. It is important to notice that an ant can only distinguish pheromones of its colony. The weights of vertices correspond to ants food. Ants put down pheromones on edges. A vertex is owned by a colony if the sum of his pheromones on adjacent edges is greater than for other colonies. A local heuristic forces ants to explore edges which have no pheromone. Ants can go where they want, so ants from different colonies can be in the same vertex at the same step. Thus, the connectivity of sets is not forced. As connected sets often produces best results, we do not need to force this connectivity.

To conclude, the approach of k-partitioning with ant colony algorithm is very different than precedent works [13], [14].

C. Fusion fission

The fusion fission method is a new method developed at the LOG laboratory. Its aim is graph partitioning. The fusion fission method is inspired by nuclear fusion and fission. The organization of a molecule can be compared to a k-partition. The molecule is the graph G, nucleons are vertices and atoms are partitions. Fusion and fission are two processes which organize atoms. Fusion is the process which assembles two atoms in one. On the contrary, fission is the process which breaks atoms into two parts. The fusion fission method consists in assembling and breaking atoms successively. Like in the natural process, fusion and fission can eject alone vertices (atoms) from a partition. Such vertices can be added to a different partition, or cut another partition in two (chained fission).

Because the number of partitions continuously change, the energy (returned by the objective function) is smaller for a small number of partitions. Thus we use a function to increase the value of the objective function if the number of partitions is not k.

The fusion fission algorithm is presented figure 1. *cpart* is the partition of G into sets of vertices. *cpart* is initialized with a near k-partition of G. t is the temperature. The higher the temperature, the easier the fusion of big atoms and the easier the fission of small atoms. $part_i$ is a partition of *cpart*. $part_i$ is randomly chosen between all partitions of *cpart*. A new partition *npart* of G is made by $fusion$ or $fission$. The *laws* correspond to probability lists, which are probabilities to eject zero, one, two or three nucleons as described upper. Then nucleons (in *ln*, a list of nucleons) are added to another

partition, or, in the fission case, if the temperature is high (*high_energy* function), these nucleons can cut partitions. The *laws* are updated if the result of the objective function is lower than the preceding partition. The function which decreases temperature is $decrease(t) = t\ \frac{t_{max}-t_{min}}{nbt}$ where nbt is the number of steps of the temperature decrease. The best result is memorized in *best_part*. If the temperature is lower than a minimal temperature, the algorithm is run with the best partition *best_part* and the upper temperature. Else, it is run with the partition *npart* find and the new temperature *new_t*.

Algorithm 1 Fusion / Fission

```
cpart ⇐ initial_partion;
t ⇐ t_max;
while Stop condition is not fulfilled do
  part_i ⇐ choose_partition(cpart);
  if choice(part_i) < random number then
    — fusion —
    (npart, ln) ⇐ fusion(part_i, cpart, laws);
    for all n ∈ ln do
      npart ⇐ nfusion(n, npart, laws);
  else
    — fission —
    (npart, ln) ⇐ fission(part_i, cpart, laws);
    for all n ∈ ln do
      if high_energy(n, t) then
        npart ⇐ nfission(n, npart, laws);
      else
        npart ⇐ nfusion(n, npart, laws);
  new_laws ⇐ update(laws, t);
  new_t ⇐ decrease(t);
  if Energy(npart) < Energy(cpart) and
  Energy(npart) < Energy(best_part) then
    best_part ⇐ npart;
  if low_temperature(t) then
    cpart ⇐ best_part; t ⇐ t_max;
  else
    cpart ⇐ npart; t ⇐ new_t;
```

The process of choice between fusion and fission is function of the number of nucleons x of the atom choose :

$$choice(x) = \begin{cases} 1 & if\ x > n + \frac{1}{2\alpha(t)} \\ 0 & if\ x < n - \frac{1}{2\alpha(t)} \\ (x-n)\alpha(t) + \frac{1}{2} & else \end{cases}$$

where $n = \frac{nbv}{k}$, nbv is the number of vertices of the graph, k is the number of partitions, and $\alpha(t) = q\frac{t_{max}-t}{t_{max}-t_{min}} + r$ where q, r are adjusted by the user.

TABLE I
OBJECTIVE FUNCTIONS RESULTS FOR EACH ALGORITHM

	Cut	*Rcut*	*Ncut*	*Mcut*
CHACO Spectral	202,353	8504	22.31	75.45
CHACO Multi.	202,095	**8492**	22.42	76.93
METIS Multi.	208,224	9962	22.76	80.49
S. annealing	203,946	9385	22.34	74.44
Ant colony	203,308	9689	22.30	74.22
Fusion Fission	**197,955**	8508	**21.83**	**69.03**

V. RESULTS

Classical partitioning methods were first used to solve the air traffic control problem. Kenighan and Lin [15] have created a very efficient algorithm, which uses a local optimization strategy. Spectral methods have been popularized by the work of Pothen, Simon and Liu [16]. Multilevel methods have been developed by Hendrickson and Leland [17] and by Karypis and Kumar [18]. In this example, the CHACO [1] library was used. This library includes Kernighan-Lin algorithm, a spectral method and the multilevel method of Hendrickson and Leland. The METIS [2] library was also used. This library includes the multilevel method of Karypis and Kumar.

CHACO's objective is to minimize the *Cut* objective function, and minimize the difference between the number of vertices in each partition. METIS's objective is to partition the vertices into k disjoint subsets such that the sum of the vertex weights in each subset is the same, and to minimize the *Cut* objective function. The metaheuristics algorithms use the *Mcut* objective function which is the most appropriate for the air traffic problem (see section III). In table I the best results of the different methods are presented. These results are computed for a partition into 32 sets, on a 3 GHz Intel Pentium 4 running with Linux. The objective function of the three metaheuristics is *Mcut*. CHACO spectral method uses RQI/Symmlq eigen solver, with the octasection partitioning method and multiple Kernighan-Lin refinement. CHACO multilevel method uses the bisection partitioning method. METIS multilevel method uses *pmetis* with default parameters. *kmetis* was also tested but without better results. Note that, if spectral and multilevel algorithms use local refinement (Kernighan-Lin), metaheuristics do not. The best results for each objective functions are strewn in bold.

TABLE II
VARIANCES

	$\sigma(spb)$	$\sigma(vw)$	$\sigma(ewbb)$
CHACO Spectral	0.4	1448	1680
CHACO Multi.	0.4	1573	1685
METIS Multi.	8.5	384	779
S. annealing	11.4	1392	1778
Ant colony	12.8	1500	1901
Fusion Fission	15.7	2620	2567

Regards to this results, the fusion fission is the best algorithm (with $Rcut$ nearest the multilevel method). If METIS and CHACO software are extremely fast (a few seconds to compute), metaheuristics are running 30 minutes to give these results.

Figure 1 and 5 details the map of the partitioning result of the fusion fission algorithm corresponding to table I result. In the same way, figure 2, 6 and 3, 7 present the partitioning results of the CHACO spectral algorithm and the METIS multilevel algorithm. Figure 4 and 8 present real partitions of the European airspace on February 2002. The three maps are vertical cuts of the European countries core are, at flight level 380 (11,000 m). Regarding to blocks of other figures, blocks of figure 5 have disproportionate shapes and sizes. Especially, the block in the center of France, which has a lot of sectors (88, compared to an expected value which is around 23.8). This result is irrelevant for air traffic control, because no controller can be qualified for so many sectors.

Table II presents the variance $\sigma(spb)$ of the number of sectors per block, the variance $\sigma(vw)$ of the sum of vertex weights $W(A)$, per block, and the variance $\sigma(ewbb)$ of the sum of edges weights between blocks $cut(A, V-A)$, per block. Expected values corresponding to these variances are: $E(spb) = 23.8$, $E(vw) \simeq 2700$, and $E(ewbb) \simeq 3500$. Because of its goal, the CHACO software produces the smallest variance of the number of sectors per block. But with 1,446 aircraft to control for the lowest block and 10,346 aircraft for the highest, it seems that this result is irrelevant for air traffic control too. METIS multilevel partitioning methods have the lowest variance of $W(A)$ and $cut(A, V - A)$. These results correspond to METIS goal. Considering METIS results, the maximal number of sectors in a block is 48, which makes these results more relevant.

Thus, a new approach of the problem is necessary. Future work on the air traffic problem will deal with minimizing the $Mcut$ objective function, but with the following constraints :

1) $\sigma(spb) < 6$
2) $\sigma(vw) < 500$
3) $\sigma(ewbb) < 500$

This defines a multi-objective problem.

VI. Conclusion

An application of three metaheuristics (simulated annealing, ant colony and fusion fission) to an air traffic control problem was presented in this paper. This air traffic control problem is a k-partitioning problem. These metaheuristics were compared to the CHACO and the METIS libraries which are classical methods to solve partitioning problems (Kernighan-Lin, spectral and multilevel methods). We compare results of these methods to results of metaheuristics with the $Mcut$ function. Compared to all of the objectives functions, metaheuristics return better results, especially the fusion fission algorithm. But the best result found with fusion fission is irrelevant for air traffic control. The analysis of CHACO and METIS partitioning results showed that these results are not relevant either. Consequently, to be applied, a solution of the air traffic control problem must respect some constraints, while minimizing the $Mcut$ objective function.

References

[1] B. Hendrickson and R. Leland. *The Chaco user's guide.* Sandia National Laboratories, 2 edition, 1994.

[2] George Karypis and Vipin Kumar. *MeTis: A software package for partitioning*, 4 edition, 1998.

[3] Federal Aviation Administration (U.S. Department of Transportation). *Air Traffic ControL : FAA Order 7110.65K*, July 1997.

[4] Anders Hallgren. Restructuring european airspace: functional airspace blocks. *Skyway*, pages 20–22, autumn 2005.

[5] Charles-Edmond Bichot and Jean-Marc Alliot. A theoretical approach to defining the european core area. Technical report, LOG - ENAC/CENA, 2005.

[6] Lars Hagen and Andrew Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, 11(9):1074–1086, 1992.

[7] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[8] Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of ICDM 2001*, pages 107–114, 2001.

[9] Charles-Edmond Bichot, Jean-Marc Alliot, Nicolas Durand, and Pascal Brisset. Optimisation par fusion et fission. application au problème du découpage aérien européen. *Journal Européen des Systèmes Automatisés*, 38(9-10):1141–1173, 2004.

[10] Charles-Edmond Bichot. A metaheuristic based on fusion and fission for partitioning problems. In *9th International Workshop on Nature Inspired Distributed Computing, In conjunction with IEEE IPDPS 2006*, Rhodes Island, Greece, 2006.

[11] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[12] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):29–41, 1996.

[13] P. Kuntz, P. Layzell, and D. Snyers. A colony of ant-like agents for partitioning in vlsi technology. In *the Fourth European Conference on Artificial Life*, pages 417–424. MIT Press, 1997.

[14] A. E. Langham and P. W. Grant. A multilevel k-way partitioning algorithm for finite element meshes using competing ant colonies. In *the Genetic and Evolutionary Computation Conf.*, volume 2, pages 1602–1608, Orlando, Florida, USA, 1999.

[15] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970.

[16] Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, 1990.

[17] Bruce Hendrickson and Robert Leland. A multi-level algorithm for partitioning graphs. In *Supercomputing*, 1995.

[18] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

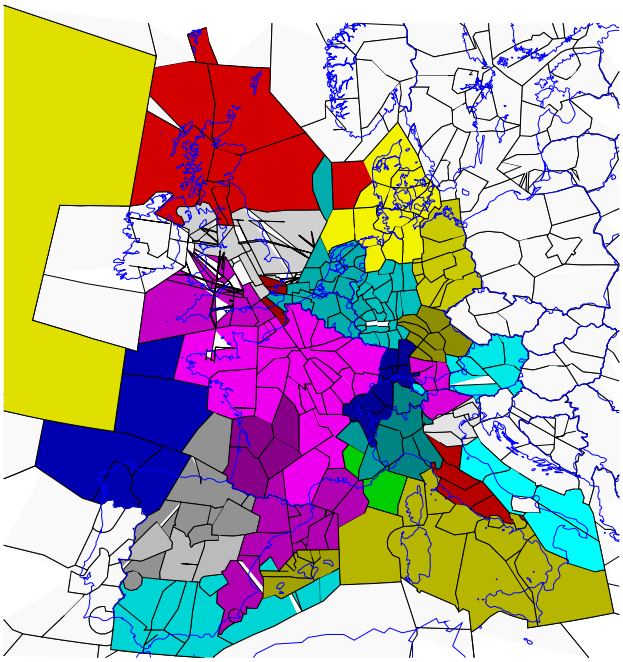

Fig. 1. Partitioning result of fusion fission (FL240)

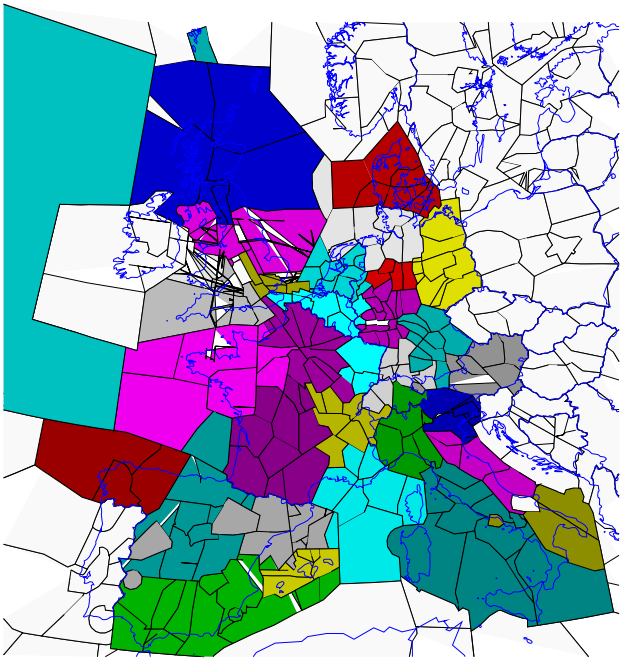

Fig. 2. Partitioning result of the CHACO package (FL240)

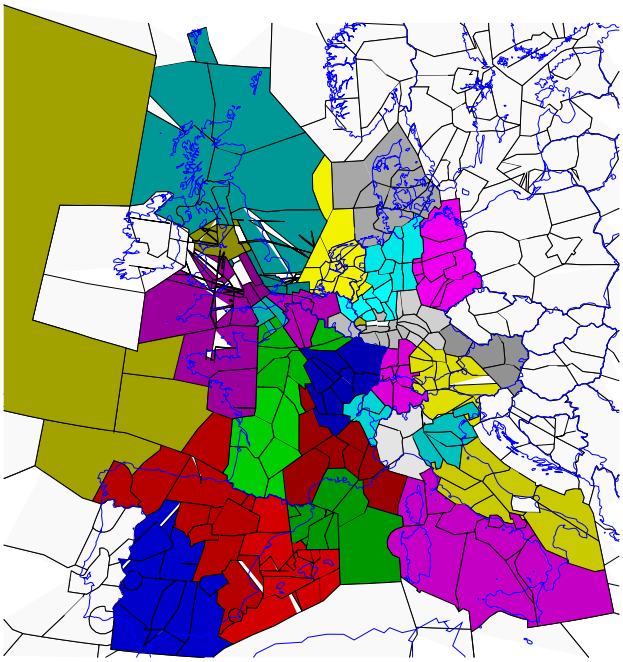

Fig. 3. Partitioning result of the METIS package (FL240)

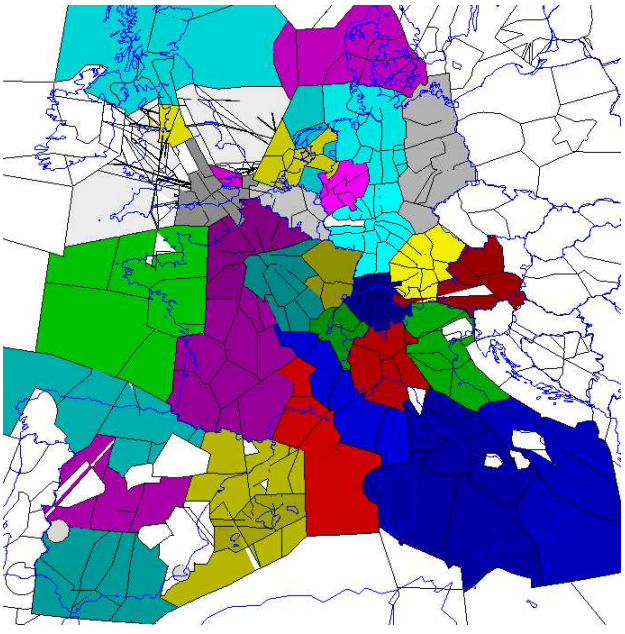

Fig. 4. Real block partitioning in February 2002 (FL240)

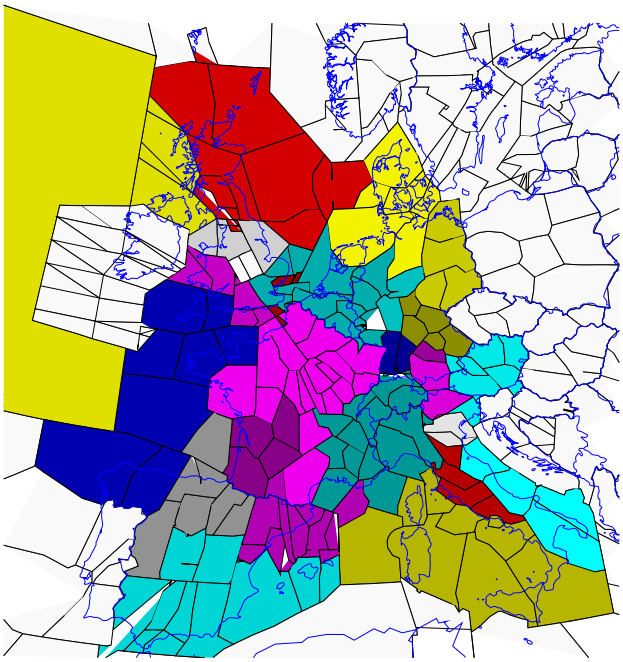

Fig. 5. Partitioning result of fusion fission (FL320)

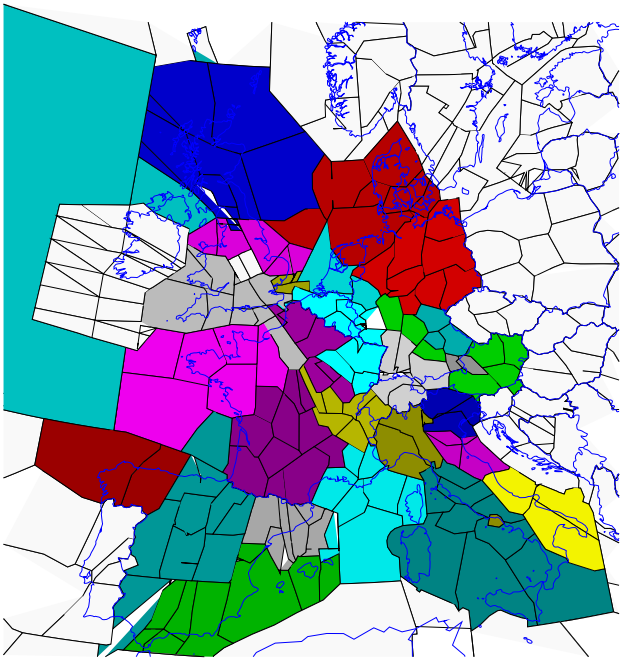

Fig. 6. Partitioning result of the CHACO package (FL320)

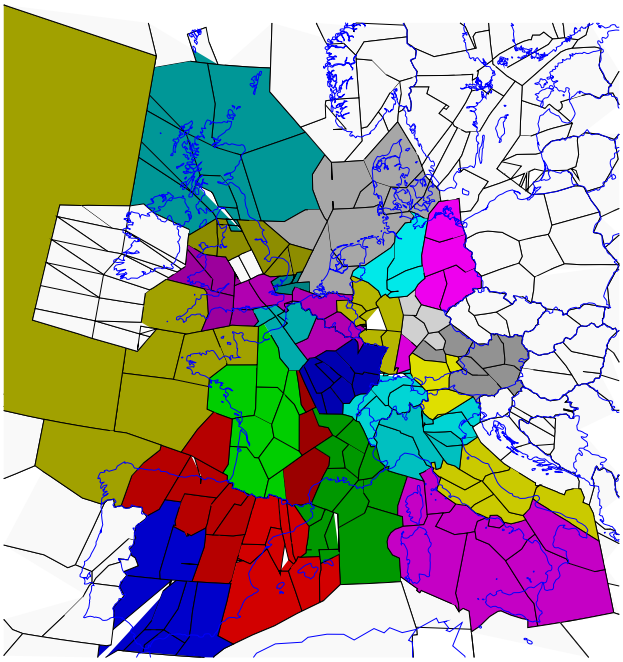

Fig. 7. Partitioning result of the METIS package (FL320)

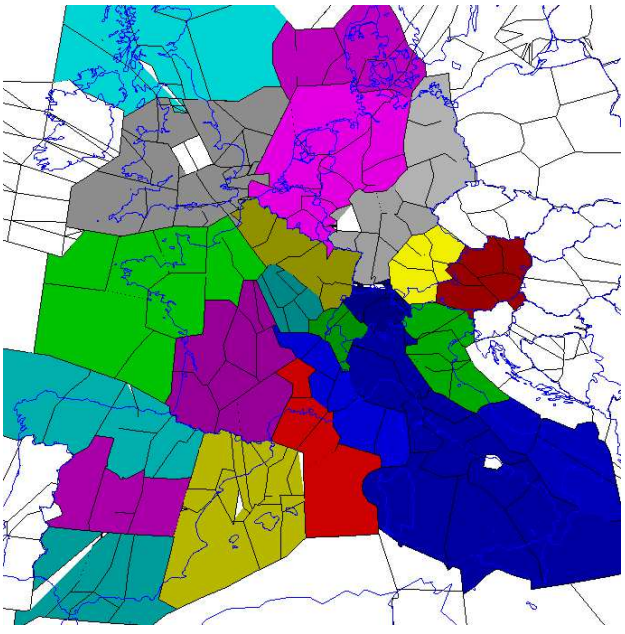

Fig. 8. Real block partitioning in February 2002 (FL320)