Heuristic Methods for the Antenna-Constrained Beam Layout Optimization on Multibeam Broadcasting Mission

Camille Lescuyer, Christian Artigues, Jean-Thomas Camino, Cédric Pralet

Heuristic methods for the antenna-constrained beam layout optimization on multi-beam broadcasting mission

Camille Lescuyer[1,2,3] Christian Artigues[2]

Jean-Thomas Camino[1] Cédric Pralet[3]

[1] Airbus Defence and Space, Space Systems, Telecommunication System Department, 31 Rue des Cosmonautes, 31402, Toulouse, France

[2]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France

[3] ONERA, DTIS, 2 Avenue Edouard Belin, 31 400, Toulouse, France

Abstract

Nowadays, telecommunication satellites are becoming more and more complex to increase the quantity and quality of the services provided by the operators. This makes the design phase of these satellites very challenging and in practice the manufacturers must face highly combinatorial problems to reach the performance required. In this paper, we consider one of these combinatorial problems. The latter involves a satellite that must provide television services to distinct regions, that each come with specific television channels requirements is a complex one for the telecommunication system engineers in charge of defining the most appropriate satellite payload. Indeed, it cumulates the difficulties of standard broadcasting missions where the same content must be transmitted to large regions with a stable quality of service, and of broadband missions where several beams have to coexist on the coverage with the risk that they mutually degrade their telecommunication performance while also being hard to accomodate mechanically on the spacecraft. We propose here two methods to solve this problem, that is to determine a set of non-conflicting beams that cover all the polygons and optimize a performance metric related to the sizes of the beams used. The first method is a matheuristic exploiting iterative resolutions of an ILP model. The second method, called the merge-and-split heuristic, is inspired by Iterated Local Search and reuses a fast graph coloring algorithm to analyze conflicts among selected beams. These two methods are evaluated on realistic instances, the largest one involving more than one hundred polygons to cover.

Keywords: Telecommunication Satellite, Television broadcasting, Linguistic Beam, Heuristic, Graph Coloring

1 INTRODUCTION

Nowadays, telecommunication satellites are major assets used by telecommunication operators to cover large zones on the Earth's surface and remote areas, with a cost

that is low compared to the installation of a terrestrial network. However, given the increasing demand in terms of quantity and quality of telecommunication services, these satellites have to be more and more efficient concerning the usage of their onboard resources. As a direct result, telecommunication satellite manufacturers need the support of optimization techniques to help them find payload design solutions that reach the mission's requirements while complying with the full set of operational and design constraints.

On this point, a classical design solution is to increase the capacity of the telecommunication satellites by using multiple beams instead of a single one, where a beam refers to a signal emitted by a satellite to cover a limited geographical area (see Fig. 1). Basically, such a multi-beam configuration allows the same frequency band to be used by several beams and is a direct and efficient way to increase the total bandwidth transmitted to the users, without enlarging the frequency spectrum of the satellite.

Moreover, using multiple beams requires using dedicated antenna technologies such as the one we focus on here, called Single-Feed-Per-Beam (SFPB), where each beam is created by a so-called feed horn (or feed, see Fig. 1), whose size and shape can be computed beforehand given the region to be served. The SFPB technology is relatively cost-efficient and has proven to behave well in terms of radio-frequency performance, but it comes with constraints that impact the possible configurations of the beams. Indeed, the feeds that generate beams exploiting a common frequency band are arranged into a feed cluster, and all the feeds composing a cluster must emit their beams towards a reflector. The latter is a parabolic antenna available onboard the satellite (typically three or four reflectors per satellite). As shown in Fig. 2, one issue is that the position of the target region of a beam allocated to a given reflector induces hard positioning constraints for the feed that generates this beam, and it can turn out that the positions required for two feeds are not compatible with each other for geometrical reasons. Said differently, if two beams share the same reflector and serve two regions that are close on the Earth's surface, then the beam layout might be unfeasible geometrically speaking. Additionally, reducing the nominal feed sizes is not acceptable performance-wise.

To optimize the design of telecommunication satellites while satisfying such beam layout constraints, several approaches are studied in the literature. In (Kyrgiazos et al., 2013), the authors consider both a non-uniform bandwidth allocation among beams and non-uniform beam sizes, and search for a beam-to-reflector allocation that maximizes the inter-beam distance. In (Camino et al., 2014), a layout of non-uniform beams is built using a randomized multi-start greedy algorithm enhanced with local search and simulated annealing. In this work, the beam compatibility constraints are represented by a graph and graph coloring techniques are used to handle the allocation of beams to reflectors. Such an idea is reused in some of the approaches described in the next sections. In (Camino et al., 2016), the possible positions of the centers of the beams and their radius are discretized, so that the possible beam layouts belong to a finite set of solutions. The authors solve an ILP formulation of the problem taking into account antenna constraints. In the works mentioned above, only sets of points are considered in the constraints and the objective of the optimization problem solved, marking a clear difference with the need to cover polygons that we propose to investigate further in this paper. Next, an existing patent ((Hammill and Dishaw, 2004)) presents a method to generate a set of non-uniform beams covering a specific polygon of interest. The sizes and frequency spectrum of each beam are chosen according to the population density of its covering area. Our problem is also close to the work described in (Contardo and Hertz, 2021), where the authors propose an exact algorithm to cover a set of polygons

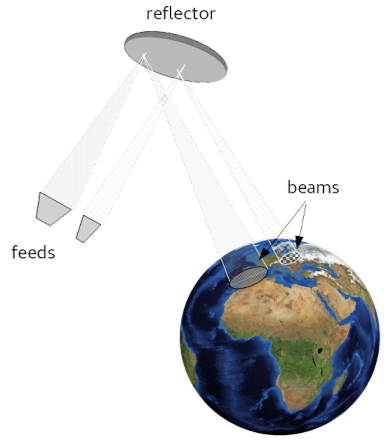


Figure 1: Beams of different sizes and their associated feeds

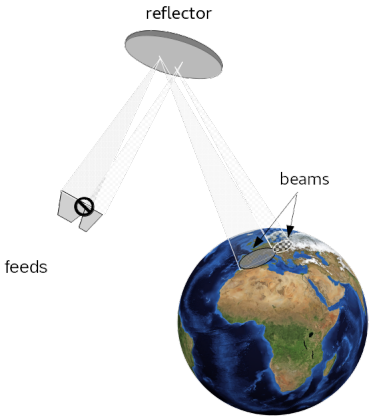


Figure 2: A conflict between two feeds

with a finite set of disks. But they do not consider the beam-to-reflector allocation problem.

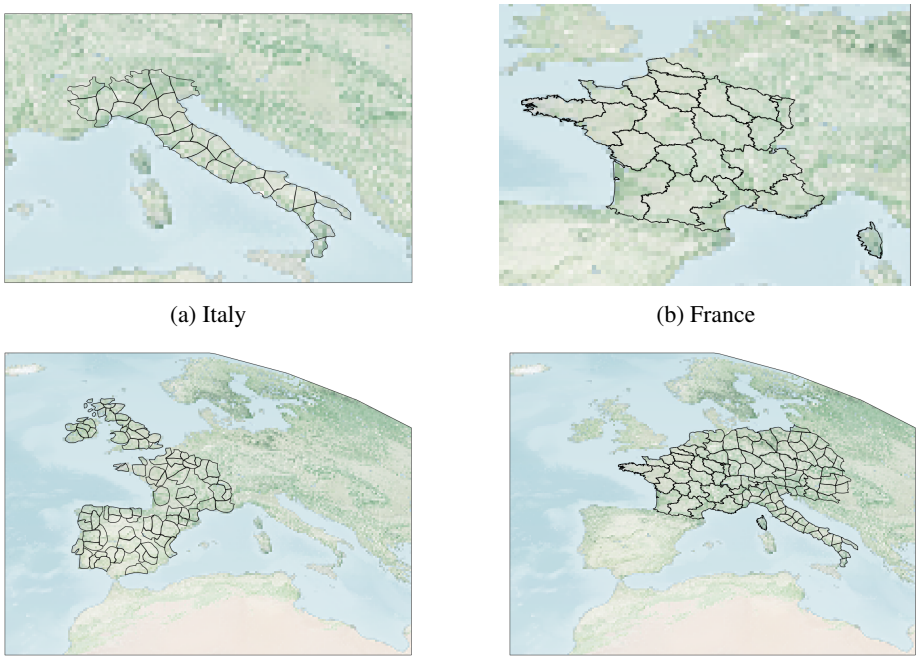
In this paper, we introduce two heuristic methods to help designing the beam layout for a SFPB antenna while meeting the telecommunication mission requirements. Our main objective is to provide the service on as many requested regions as possible. Additionally, we want to provide a good quality of service. In practice, this quality of service can only be accurately assessed once the design of the full payload has converged, after several important design steps subsequent to the beam layout optimization discussed here (frequency allocation, definition of all payload routes, TV channels assignment, frequency conversion hardware definition, amplifiers sizing, power distribution, etc.). In this decomposed optimization scheme where the beam layout is the first step, we actually incorporate constraints whose only purpose is to prepare for the best the following optimizations that are out of the scope of this paper. To that purpose, we aim at creating beams that have a small size. The reason for this is that, for a given power capacity available on-board, it allows the signals to have higher power spectral densities, which for the signal receivers on the ground leads to a service of higher quality. In the end, our goal is to select beams covering as many regions as possible and having minimum sizes, while ensuring that two beams that are too close to each other cannot be assigned to the same reflector to avoid conflicts on the positions of their respective feeds.

The rest of the paper is organized as follows. Section 2 formalizes the problem using Integer Linear Programming (ILP). Section 3 introduces a matheuristic method where a set of candidate beams is iteratively enlarged. Section 4 presents an heuristic called the merge-and-split algorithm. Section 5 provides experimental results on several instances. Last, Section 6 concludes and gives some perspectives.

2 Problem definition

2.1 Telecommunication mission

The telecommunication mission is defined by a set of polygons on the Earth's surface. Each of these polygons is defined by the set of successive vertices placed on its boundary and has its own demand expressed in terms of number of television channels. The segmentation of the market in geographical areas can be made for linguistic reasons or because of the economical context. Figures 3a to 3d provide four examples of instances involving an increasing number of polygons to be covered (from 22 polygons for the "France" instance to 103 polygons for the "Central Europe" instance). These four instances are those used in our experiments. In the following, the set of polygons to cover is referred to as P.



(a) Italy

(b) France

(c) Western Europe

(d) Central Europe

Figure 3: Instances considered

2.2 Beams

General beams In the literature, we can find several definitions of what a beam is, according to different contexts and perspectives. In this paper, a beam is represented as a disk that covers a specific area on the Earth's surface served by the satellite communication system. More formally, a beam is a pair $b = (c_b, r_b)$ where c_b is the center of the beam (defined by a longitude and a latitude) and r_b is the radius of the disk associated with the beam. From the features of a beam, it is also possible to compute the set $P_b \subseteq P$ that contains all the polygons covered by beam b. For signal quality reasons, the radius of all the beams considered must not be greater than a maximum value referred to as $MaxRadius$.

Smallest covering beams In the following, we consider specific kinds of beams that are directly defined by the set of polygons they cover. For this, given a subset of polygons $P' \subseteq P$ to be covered, we can compute the smallest beam $b = (c_b, r_b)$ that covers all the polygons in P'. Such a computation can be done in polynomial time using Welzl algorithm (Welzl, 2005). The latter returns in time $O(n)$ the smallest enclosing circle of a set of n points in the Euclidian space. In our case, the relevant points are the vertices of the polygons in P', and the smallest enclosing circle for these points directly provides the features of the smallest beam covering the polygons. In the following, the Smallest Covering Beam for a set of polygons $P' \subseteq P$ is referred to as $SCB(P')$.

We can therefore identify one of the combinatorial issues of our problem: if a beam can be defined from any subset of polygons $P' \subseteq P$ and if the radius of each smallest covering beam obtained does not exceed *MaxRadius*, then there are $2^{|P|}$ candidate beams.

Conflicts between beams As mentioned in the introduction, the antenna we consider is Single-Feed-Per-Beam (SFPB). This antenna technology is mature for this type of telecommunication mission and has the advantage of being among the less expensive solutions. In this case, the antenna is characterized by a set of *reflectors* and each reflector is composed of a set of feeds and a dish. Each feed is the hardware equipment associated with a single beam, and the dish focuses the signal and reflects the beam on the Earth's surface. A common order of magnitude for the number of reflectors in such systems is three or four.

Two beams whose disks are close on the Earth's surface and that are allocated to the same reflector can create a conflict on their associated feeds. To avoid any conflict, we impose a minimum separation distance between two beams associated with the same reflector. More formally, two beams $b_1 = (c_{b_1}, r_{b_1})$ and $b_2 = (c_{b_2}, r_{b_2})$ cannot belong to the same reflector if and only if the CONFLICT$(.,.)$ predicate defined below returns value true for beams b_1, b_2:

$$\text{CONFLICT}(b_1, b_2) : dist(c_{b_1}, c_{b_2}) < \kappa(r_{b_1} + r_{b_2}) \tag{1}$$

where $dist(c, c')$ denotes the Euclidian distance between two points c, c' in the longitude-latitude plane and $\kappa > 1$ is a fixed parameter defined by the satellite manufacturer. Equation 1 expresses that there is a conflict between b_1 and b_2 if the distance between the centers of these beams is less than κ times the sum of the two beam radii. For this article, we consider $\kappa = \sqrt{3}$ as it has been assessed to be representative of what has been observed recently on related activities on the satellite manufacturer side.

From an operations research perspective, the beam separation constraints can be represented using a graph coloring problem. Indeed, let B_s denote the set of beams selected. We can build a graph $G(B_s)$ called the feed conflict graph, containing one node per beam in B_s and one edge between two nodes associated with beams b_1, b_2 such that CONFLICT(b_1, b_2) takes value true. Then, we consider as many colors as the number of reflectors available, and our goal is to color graph $G(B_s)$ so that two adjacent nodes do not have the same color. In other words, a set of beams is mechanically implementable if the chromatic number of $G(B_s)$, referred to as $\gamma(G(B_s))$, is less than or equal to the number of reflectors, so that the beams can be distributed among the different reflectors.

2.3 Input data

For a particular beam layout problem, we consider the following input data:

- $P = \{p_1, p_2, \ldots, p_{N_P}\}$: set of polygons to cover;
- $R = \{1, 2, \ldots, N_R\}$: set of reflector indices;
- $B = \{b_1, b_2, \ldots, b_{N_B}\}$: set of candidate beams; the radius of all these candidate beams is assumed to be consistent with parameter *MaxRadius*;
- $I \subset B \times B$: pairs of beams (b_1, b_2) such that b_1 and b_2 cannot be assigned to the same reflector (I for incompatible beams);
- $B_p \subseteq B$ (for $p \in P$): subset containing all the beams covering polygon p, that is the beams $b \in B$ such that $p \in P_b$; set B_p can be directly computed from the features of the beams in B and the features of polygon p, hence it can actually be omitted from the input data.

2.4 ILP definition

Defining a solution of the beam layout problem means (1) selecting a subset of the candidate beams so as to cover polygons, and (2) associating a reflector with each selected beam so that the reflector-to-beam allocation is feasible. Among the candidate solutions, our main objective is to cover as many polygons as possible, and our secondary objective is to minimize the size of the beams used, so as to optimize the quality of the signal received on the ground.

Such a problem can be formalized as an Integer Linear Program by introducing the following variables:

- $z_p \in \{0,1\}, p \in P$: binary variable taking value 1 if and only if the polygon p is covered by a selected beam;
- $x_{b,r} \in \{0,1\}, b \in B, r \in R$: binary variable taking value 1 if and only if beam b is allocated to reflector r.

Then, the ILP model proposed is given in Equations 2 to 5. The objective function given in Equation 2 tries to both maximize the number of polygons covered (term $\sum_{p \in P} z_p$) and minimize the size of the beams used (term $-\sum_{r \in R} r_b^2 x_{b,r}$). To express that the main goal is to maximize the coverage of the polygons, a large constant $M = \sum_{b \in B} r_b^2 + 1$ is used to weight the first term, which leads to a lexicographic objective function. Constraint 3 ensures that if a polygon is covered, then at least one of its covering beams is selected. Constraint 4 ensures that if a polygon p is not covered, then none of the beams covering p is selected. As z_p is a binary variable, it also states that a beam is associated with at most one reflector. Last, Constraint 5 expresses that two conflicting beams cannot use the same reflector.

$$\text{maximize} \quad M \cdot \sum_{p \in P} z_p - \sum_{b \in B, r \in R} r_b^2 x_{b,r} \tag{2}$$

$$\text{subject to} \quad \forall p \in P, \sum_{b,r \in B_p \times R} x_{b,r} \geq z_p \tag{3}$$

$$\forall p \in P, \forall b \in B_p, \sum_{r \in R} x_{b,r} \leq z_p \tag{4}$$

$$\forall r \in R, \forall (b_1, b_2) \in I, x_{b_1,r} + x_{b_2,r} \leq 1 \tag{5}$$

2.5 Complexity

Proving the NP-hardness of the beam layout problem considered is in our perspectives, but we can mention close problems from the literature whose NP-hardness is already known. Notably, we can find similarities with several NP-complete covering and packing problems (Fowler et al., 1981). For instance, given a set of points to cover, the 3-colorable unit disk covering problem consists in finding a set of unit disks and a color for each disk, so that the union of the disks cover all the points and overlapping disks have distinct colors. This problem is proven NP-hard in (Biedl et al., 2021). In another direction, the beam layout problem tackled in (Camino, 2017) considers a set of points to be covered by beams, where each point has a traffic demand. In this context, the problem of searching for a beam layout (including the beam-to-reflector assignment) that maximizes the total traffic is proven NP-hard, based on a polynomial reduction of the Circle-Covering problem. One key difference with our problem however is that we want to maximize the number of polygons covered by beams of small sizes, instead of maximizing the total traffic associated with a set of covered points.

3 Matheuristic method

Ideally, the set of candidate beams B considered in the ILP model should contain the smallest covering beam associated with each possible subset of the polygons in P. However, as mentioned before, this leads to a number of beams that is exponential in the number of polygons, hence enumerating all these beams is not practicable. For example, for an instance involving only 20 polygons, there are more than 1048576 possible subsets of polygons and as many potential beams. This is why we propose a matheuristic method that solves the ILP several times, on a restricted pool of candidate beams B that evolves during the iterations. Iterations are performed until a solution covering all the polygons is found or until a maximum CPU time is reached.

3.1 Detailed description

The matheuristic is provided in Algorithm 1. Initially, the set of candidate beams B only contains all the smallest beams covering a single polygon in P and all the smallest beams covering two polygons in P. Based on these candidate beams, it is possible to compute the set I containing the pairs of incompatible beams and to solve the ILP model given before for input data P, R, B, I. In Algorithm 1, this is achieved through a call to function solveILP, that returns the set of beams B_s selected in the solution of the ILP presented in Section 2.4. More formally, if $\overline{y}$ denotes the value of a variable y after optimization, then $B_s \leftarrow \text{solveILP}(P,R,B,I)$ is equivalent to $B_s \leftarrow \{b \in B \mid \sum_{r \in R} \overline{x_{b,r}} = 1\}$. Altogether, the beams in B_s cover a set of polygons $P_s = \cup_{b \in B_s} P_b$.

Then, as long as a polygon in P is not covered and there is some computation time left, the algorithm tries to improve the current solution. To do this, it generates a set of new beams B_g, to enlarge set B, and updates the set of beam incompatibilities to take into account these new beams. After that, the enlarged ILP model is solved again, which may update the beam selection strategy. Finally, the last beam selection found is returned by the algorithm. In case there is a maximum CPU time for each call to solveILP, it is also possible to return the best solution found along the iterations, according to the objective function provided in Equation 2. Note that in Algorithm 1, cpuTime() is a function returning the current computation time and *TimeLim* stands for

the global CPU time limit.

Input: P: set of polygons to cover; R: set of reflector indices
$B \leftarrow \text{initBeams}(P)$;
$I \leftarrow \text{initIncomp}(B)$;
$B_s \leftarrow \text{solveILP}(P,R,B,I)$;
$P_s \leftarrow \cup_{b \in B_s} P_b$;
while $P_s \neq P$ *and* $\textit{cpuTime}() \leq \textit{TimeLim}$ **do**
  $B_g \leftarrow \text{generateNewBeams}(B_s, P)$;
  $I_g \leftarrow \text{generateNewIncomp}(B_g, B)$;
  $(B,I) \leftarrow (B \cup B_g, I \cup I_g)$;
  $B_s \leftarrow \text{solveILP}(P,R,B,I)$;
  $P_s \leftarrow \cup_{b \in B_s} P_b$
end
return B_s

Algorithm 1: Pseudocode of the matheuristic

3.2 Beam generation heuristic

In the matheuristic proposed, the key challenge is to define a good beam generation strategy, that is to find beams that should be added to the current pool of beams to help converge towards a solution covering as many polygons as possible. Several options have been tested in our experiments. In this paragraph, we detail two methods to augment the pool with new beams if the current pool is not sufficient to cover all polygons.

Beam Generation Method 1 (BGM1) The first method proposed adds beams covering subsets of polygons of increasing cardinality throughout the iterations. For this, we define a parameter *NbPolyLimit* that is increased by one at each iteration. Initially, we set $\textit{NbPolyLimit} = 2$ so as to generate all beams covering one and two polygons. Then, at each iteration, solving the ILP leads to a solution selecting a set of beams B_s. If the set B_s does not cover all polygons, then we increase *NbPolyLimit* by one unit and generate all beams b obtained by merging two beams of the solution and such that b covers at most *NbPolyLimit* polygons. Formally, we compute

$$\begin{aligned} B_g = \{b = \text{SCB}(P_{b_1} \cup P_{b_2}) \mid & \\ & (b_1, b_2 \in B_s) \wedge (r_b \leq \textit{MaxRadius}) \\ & \wedge (|P_b| \leq \textit{NbPolyLimit})\} \end{aligned} \tag{6}$$

Beam Generation Method 2 (BGM2) The second method proposed consists in using the set of beams B_s selected in the solution produced for the ILP, and adding a single polygon to each of these beams. More precisely, let us consider a beam $b_s \in B_s$ and a polygon $p \in P$ that is not covered by b_s. If the smallest beam covering p ($\text{SCB}(\{p\})$) is in conflict with b_s, then we generate the smallest enclosing beam b that covers p and the polygons in P_b, the underlying idea being to try and cover p thanks to beam b.

Formally, the set of beams generated after each iteration is:

$$B_g = \{b = \text{SCB}(P_{b_s} \cup \{p\}) \mid (b_s \in B_s) \wedge (p \in P \setminus P_{b_s}) \wedge (r_b \leq MaxRadius) \wedge \text{CONFLICT}(b_s, \text{SCB}(\{p\}))\} \quad (7)$$

4 Merge-and-split heuristic

The ILP manipulated in the previous sections has limitations on instances containing more than 100 polygons. To overcome this limitation, we define a specific heuristic that is independent of the ILP formalization, called the merge-and-split heuristic.

4.1 Global description

The merge-and-split heuristic proposed is described in Figure 4. It starts from a set of beams B containing the smallest covering beam associated with each polygon in P, i.e. $B = \{SCB(\{p\}) \mid p \in P\}$. By definition, this set of beams covers all the polygons in P. If the feed conflict graph $G(B)$ can be colored using a number of colors that does not exceed the number of reflectors available (N_R) then, even if this is not explicit in Figure 4, the initial set of beams B is directly returned as the solution found, as it only uses beams of minimum sizes.[1]

Otherwise, there are conflicts between the beams selected so far. In this case, the algorithm tries to iteratively merge pairs of beams used in the current solution. Again, merging two beams b_1 and b_2 means building the smallest beam covering all the polygons covered by b_1 and b_2. The idea here is that by merging two beams, we hope to get a new conflict graph that is easier to color than the current graph. If the chromatic number of the new feed conflict graph obtained after a merging operation is still too high, another beam merging operation is performed, and so on until reaching a colorable graph or until a maximum number of merging operations is reached.

At that point, the algorithm splits some of the beams contained in the current solution, where splitting a beam b covering a set of polygons P_b means replacing b by the set of individual beams $\{SCB(\{p\}) \mid p \in P_b\}$ covering each polygon served by b. After that, the merging phase is performed again, and doing so other regions of the search space can be explored since the decisions made at the level of the merging module involve some randomness. Moreover, as soon as a new solution is obtained by the merging process, we update the best solution found so far. Note that if the current solution is not feasible (not colorable using N_R colors), we can still extract a feasible solution by keeping only the subset of beams that are consistently colored by the graph coloring method. Last, we fix a total time limit, and we return the best solution found at the end of the process.

Globally, the merge-and-split heuristic is inspired from Iterated Local Search (Lourenço et al., 2003), which alternates between optimization phases where local moves are performed to try and improve the current solution (*beam merging* moves in our case), and a perturbation phase where the features of the current solution are randomly updated (*split* operations in our case). In the following, we detail the three main components of the algorithm, that is the coloring, merging, and split procedures.

[1] Technically speaking, it can be shown that in this case, the set of beams in B corresponds to a solution that is leximin-optimal if the evaluation of a solution is provided by the vector containing the radii of all the beams used.

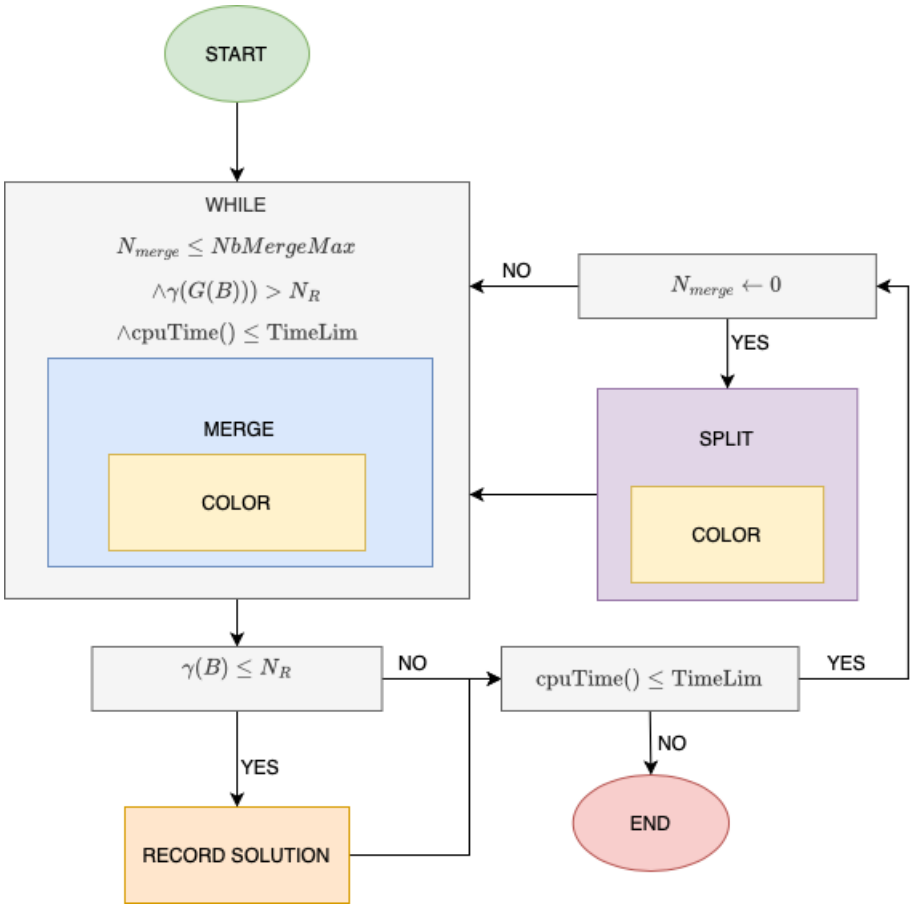


Figure 4: Merge-and-split algorithm

4.2 Coloring method

Each time the current set of beams B is updated, we try to color the corresponding feed conflict graph. This function is called many times during the algorithm and needs to be fast, even if determining whether a graph can be colored using a restricted number of colors is NP-complete. To quickly evaluate the colorability of a graph, we reuse DSATUR (Brélaz, 1979), a greedy algorithm that runs in polynomial time. Basically, DSATUR colors the nodes of highest degree first (the nodes that have the highest number of neighbors). The algorithm is allowed to use any number of colors, but to limit the number of colors used, each node n is colored at each step using a color that has a minimum index and that is feasible given the neighbors of n that are already colored. Coming back to the feed conflict graph, if DSATUR manages to color $G(B)$ using no more than N_R colors, then there exists a consistent beam-to-reflector allocation. Otherwise, when DSATUR fails to find a solution using at most N_R colors, the feed conflict graph might have a consistent coloring (since DSATUR is incomplete), but this is ignored by the merge-and-split algorithm whose goal is to quickly find good quality solutions. In the following, we denote by $\hat{\gamma}(G(B))$ the number of colors used by DSATUR to color $G(B)$, and this number of colors is an upper bound on the actual

chromatic number $\gamma(G(B))$.

4.3 Merging mechanisms

The pseudocode of the merging process is sketched in Algorithm 2. As expressed in the condition of the while loop, merging operations are performed while the chromatic number of current feed conflict graph exceeds N_R and there is some computation time left and the number of merge operations does not exceed a limit referred to as $NbMergeMax$. As we only merge beams two by two, the highest number of merging operations is always $|P|-1$ (if this number is reached, then we have produced the unique beam covering all the polygons), hence we always have $NbMergeMax \leq |P|-1$. In practice, we consider a smaller value for *NbMergeMax* in order to favor the exploration of solutions containing a larger number of beams. Parameter *NbMergeMax* can also be updated during search if needed.

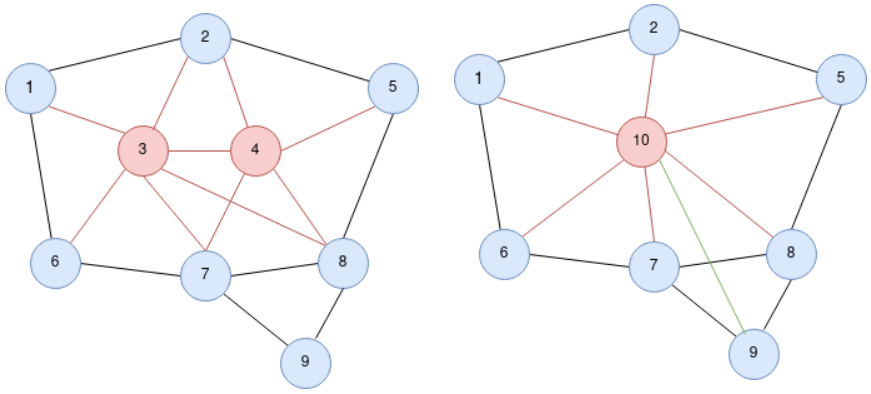
During the merging phase, the algorithm maintains the set of beam pairs that are candidates for being merged (set *Cand* is the pseudocode). Initially, this set contains all pairs (b_1,b_2) such that b_1 and b_2 are two distinct beams used in the current solution (and we use an ordering $\prec$ over beams to avoid considering equivalent pairs (b_1,b_2) and (b_2,b_1)). At each step, the merging loop selects a candidate pair of beams (b_1,b_2) in *Cand*, according to a merging method randomly chosen (more details later on this point). If merging b_1 and b_2 is feasible according to parameter *MaxRadius*, then we consider the new set of beams $B' \leftarrow (B \setminus \{b_1,b_2\}) \cup \{b_3\}$ where b_3 is obtained by merging b_1 and b_2. As illustrated in Figure 5, this leads to a new feed conflict graph $G(B')$ that is likely to contain fewer edges and that may be easier to color. However, in some cases, the chromatic number of $G(B')$ can be greater than the chromatic number of $G(B)$ since merging two beams can also create new edges in the feed conflict graph. The reason for this is that beam b_3 can cover regions that are covered neither by b_1 nor by b_2, hence there may exist another beam that is sufficiently separated from b_1 and b_2 but that is in conflict with b_3. This occurs in Figure 5 where beam 10 created by merging beams 3 and 4 has a conflict with beam 9 that was not in conflict with the two beams merged. As a result, the chromatic number of $G(B')$ may be higher than the chromatic number of $G(B)$. On this point, to avoid making moves leading to a conflict graph that is harder to color, the merging procedure accepts B' as the new set of beams only if for DSATUR, the estimated chromatic number of $G(B')$ is not greater than the estimated chromatic number of $G(B)$ (i.e., $\hat{\gamma}(G(B')) \leq \hat{\gamma}(G(B))$). If set B' is accepted as the new set of beams, the algorithm updates the set of candidate beam pairs to take into account the presence of b_3 and increments the number of merging operations done so far.

In the following, we define three heuristics to select the two beams b_1,b_2 to merge at each step.

Merging method 1 (M1) The first merging heuristic favors the selection of two beams b_1,b_2 such that merging b_1 and b_2 leads to a beam that is small. The underlying idea is not only to try and keep small beams, but also to merge beams that are the less likely to create new edges in the feed conflict graph. Therefore, for all pairs of distinct beams $b_1,b_2 \in B$, we compute the smallest covering beam b_3 obtained after merging b_1 and b_2. To diversify the search process, we do not systematically select a pair (b_1,b_2) leading to the smallest beam b_3. Instead, merging method M1 randomly selects a pair (b_1,b_2) leading to a merged beam b_3 that is among the $\lceil |B|^2 \times RatioSelectedBeams \rceil$

Input: B: set of beams in the current solution
$NbMerge \leftarrow 0$
$Cand \leftarrow \{(b_1,b_2)\,|\,b_1,b_2 \in B, b_1 \prec b_2\}$
while $\hat{\gamma}(G(B)) > N_R$ *and*
    $Cand \neq \emptyset$ *and*
    $cpuTime() \leq TimeLim$ *and*
    $NbMerge \leq NbMergeMax$ **do**
  $m \leftarrow$ select merging method$(ProbMerge)$
  $(b_1,b_2) \leftarrow$ select a pair in $Cand$ given m
  $Cand \leftarrow Cand \setminus \{(b_1,b_2)\}$
  $b_3 \leftarrow SCB(P_{b_1} \cup P_{b_2})$
  if $r_{b_3} \leq MaxRadius$ **then**
    $B' \leftarrow B \cup \{b_3\} \setminus \{b_1,b_2\}$
    if $\hat{\gamma}(G(B')) \leq \hat{\gamma}(G(B))$ **then**
      $B \leftarrow B'$
      $Cand \leftarrow Cand \cup \{(b_3,b)|b \in B' \setminus \{b_3\}\}$
      $NbMerge \leftarrow NbMerge + 1$
    end
  end
end

Algorithm 2: Merging loop



(a) Before merging

(b) After merging

Figure 5: Impact of beam merging on the feed conflict graph

smallest ones. Here, $RatioSelectedBeams \in]0,1]$ is a parameter that allows us to control the number of candidate pairs considered at each step, that is the degree of diversification.

Merging method 2 (M2) The second merging heuristic favors the selection two beams b_1, b_2 that have the highest number of common neighbors in graph $G(B)$. As illustrated in Figure 5b, the underlying idea is to make the coloring of these common neighbors easier. In Figure 5, beams 3 and 4 have four common neighbors, the merging of this pair of beams deletes 10 edges and creates 7 ones, reducing the total number of edges in the graph.

As a result, in merging method M2, the quality of a beam obtained by merging two beams b_1 and b_2 is the number of common neighbors of b_1 and b_2 in the feed conflict graph. To diversify the search process, we do not systematically select the merged beam that has the highest quality. Instead, merging method M2 randomly selects a pair of beams (b_1, b_2) leading to a beam b_3 that is beam among the $\lceil |B|^2 \times RatioSelectedBeams \rceil$ highest quality ones. Again, parameter $RatioSelectedBeams \in]0,1]$ allows us to control the degree of diversification of the method.

Merging method 3 (M3) This third method first selects a beam b for which the index of the color assigned by DSATUR is strictly greater than the number of reflectors available. This beam is then merged with a beam b' that is selected following either a rule inspired from method M1, or a rule inspired from method M2, or a rule that merges b with its closest neighbor.

Formally, DSATUR associates a color index $color_b \in \mathbb{N}^*$ with each beam b, and the set of beams whose color is not consistent with N_R is

$$HighColorBeams = \{b \in B \mid color_b > N_R\}.$$

Then, to preferentially select small beams, we associate a selection probability

$$p_b = \frac{\frac{1}{radius(b)}}{\sum_{b \in HighColorsBeams} \frac{1}{radius(b)}}$$

with each beam $b \in HighColorBeams$, and we randomly select a beam b in $HighColorBeams$ according to these probabilities.

To select the second beam b', we have three different methods:

- in the first method, we select a beam b' such that the beam created by merging b and b' has the smallest radius;
- in the second method, we select a beam b' that has the highest number of common neighbors with b;
- in the third method, we select a beam b' whose center is as close as possible to the center of b.

The method for selecting the second beam is chosen randomly among these three methods. In the end, the pair (b_1, b_2) selected is (b, b') if $b \prec b'$ and (b', b) otherwise.

4.4 Splitting mechanisms

The pseudocode of the splitting procedure is given in Algorithm 3. The main idea is to split some of the largest beams in order to remove the contribution of these beams to the objective function in terms of squared radius. The method does not split a unique beam. Instead, given the current solution that contains a set of beams B, it splits a certain proportion of these beams β. More precisely, it selects $\lceil \beta \times |B| \rceil$ beams in B. The value of parameter β depends on the current set of beams B and can take two possible values, referred to as β_m and β_M. The former is used when the current set of beams B provided by the merging operations done before corresponds to a feasible solution (i.e., DSATUR manages to color the current set of beams using no more than N_R colors). Typically, parameter β_m is low enough to try and improve the current

solution and not restart all over again, but not too low to favor the exploration of other parts of the search space. On the contrary, parameter β_M should be higher in order to try and remove inconsistencies in the current solution.

Then, as expressed in Algorithm 3, the beam to be split at each step is selected according to a probability distribution that is proportional to the beam size. Doing so, splitting the largest beams is the preferred options, but there is some randomness in the process to promote diversification.

Input: B: current set of beams
if $\hat{\gamma}(G(B)) \leq N_R$ **then** $\beta \leftarrow \beta_m$;
else $\beta \leftarrow \beta_M$;
$NbSplit \leftarrow \lceil \beta \times |B| \rceil$
$Cand \leftarrow B$;
for $i = 1$ *to* $NbSplit$ **do**
  $b_s \leftarrow$ select a beam $b \in Cand$ following a selection probability $p_b = r_b^2 / (\sum_{b \in B} r_b^2)$
  $Cand \leftarrow Cand \setminus \{b_s\}$
  $B \leftarrow (B \setminus \{b_s\}) \cup \{SCB(\{p\}) \mid p \in P_{b_s}\}$
end

Algorithm 3: Splitting loop

4.5 Parameters

In the end, the merge-and-split heuristic uses several parameters. The latter are summarized below and may be adjusted according to the instance.

- TimeLim: maximum duration time of the search process;
- *NbMergeMax*: maximum number of merging operations performed before splitting some beams;
- $ProbMerg = [p_{M1}, p_{M2}, p_{M3}]$: selection probability for each merging method;
- *RatioSelectedBeams*: ratio used to define the proportion of beam pairs among which the beam merging method selects a good alternative;
- β_m: ratio of beams split when the current solution is feasible (colorable using no more than N_R colors);
- β_M: ratio of beams split when the current solution is not feasible.

5 Experiments

Experimental setup The matheuristic and merge-and-split method have been implemented in Python. The ILP problems have been solved using CPLEX 12.10. The runs were made on a server with 96 cores of an Intel(R) Xeon(R) Gold 5318Y CPU @2.10GHz processor and 62GB of RAM. CPLEX exploits the 96 cores while the merge-and-split heuristic runs on a unique core. This difference must be considered in the analysis of the experimental results, but for the heuristic our goal is to build a

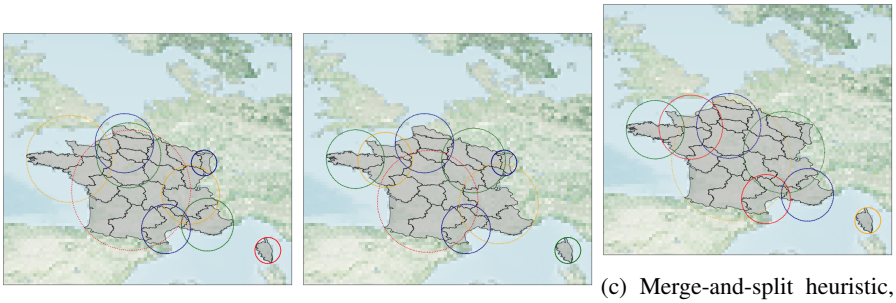
fast method anyway. We consider a telecommunication satellite having $N_R = 4$ reflectors, and the instances tackled are those presented in Figures 3a, 3b, 3c, and 3d. If we covered each of the polygons of the four instances with one unique beam, we would obtain a feed conflict graph whose chromatic number computed with DSATUR algorithm would be respectively 10,8, 9 and 12. To evaluate the efficiency of each algorithm, we analyze the number of polygons covered by the solution found and the number of beams selected in this solution. We also analyze the Mean Squared Radius Sum given by

$$\text{MSRS} = \frac{\sum_{b \in B_s} r_b^2}{|B_s|}$$

where B_s stands for the set of beams selected in the final solution. For the merge-and-split heuristic, we use the settings given in Table 1.

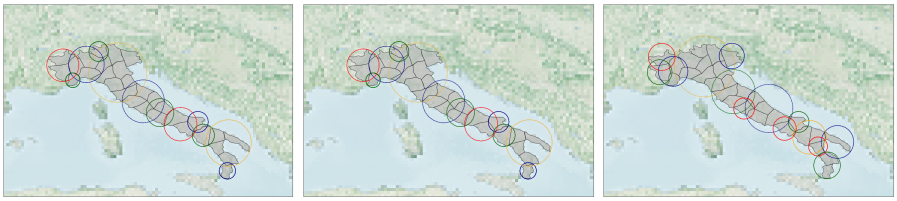
Table 1: Parameters of the merge-and-split heuristic

β_m	0.2
β_M	0.8
ProbMergMethod	$[0.2, 0.7, 0.1]$
RatioSelectedBeams	0.2



(a) Matheuristic using BGM1 (b) Matheuristic using BGM2 (c) Merge-and-split heuristic, run 2

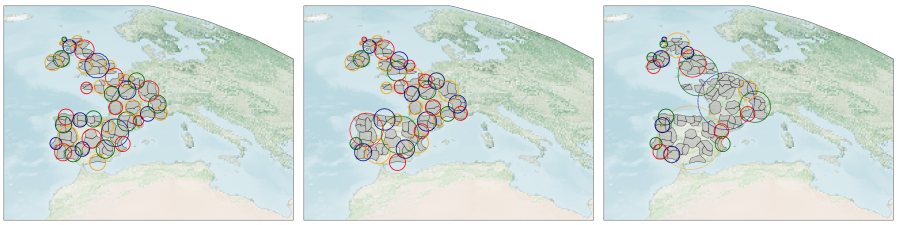
Figure 6: Solutions on the France instance



(a) Matheuristic using BGM1 (b) Matheuristic using BGM2 (c) Merge-and-split heuristic

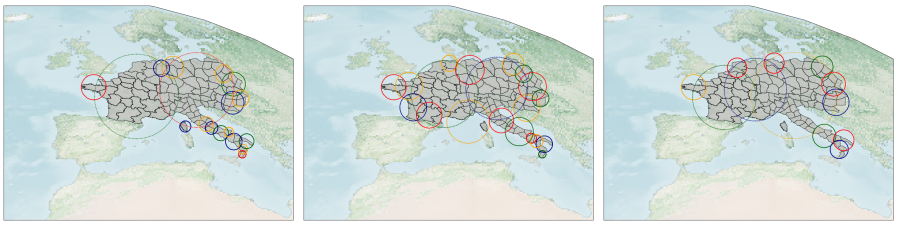
Figure 7: Solutions on the Italy instance

Results of the matheuristic The solutions found by the matheuristic method are illustrated in Figures 6a, 6b, 7a, 7b, 8a, 8b, 9a, 9b. These figures depict the set of beams



(a) Matheuristic using BGM1 (b) Matheuristic using BGM2 (c) Merge-and-split heuristic

Figure 8: Solutions on the Western Europe instance



(a) Matheuristic using BGM1 (b) Matheuristic using BGM2 (c) Merge-and-split heuristic

Figure 9: Solutions on the Central Europe instance

obtained in the best solution found, each beam being colored according to the index of the reflector to which it is assigned. From an operational point of view, the solutions appear to be of good quality for the telecommunication satellite designers. The detailed results of the matheuristic are given in Table 2, where BGM stands for "Beam Generation Method" to indicate which method we used to fill in the pool of beams after each iteration. We recall that *BGM*1 fills in the pool of beams by merging beams from the current solution with each other, while *BGM*2 fills in the pool by merging a beam belonging to the current solution with a beam covering a unique polygon. In the table, we also indicate the time limit of the matheuristic methods and the time limit specified for the call to the ILP solver at each iteration (column *ILP time limit*). This parameter is set manually following preliminary experiments. The results obtained show that globally, for the instances considered, BGM1 is less efficient than BGM2. Indeed, both methods manage to cover all the polygons before the time limit (see column $|P_s|$), but BGM2 leads to a higher number of beams and a lower MSRS value (see columns $|B_s|$ and MSRS). This is true for the four instances except from the Italy instance where the two methods give the same final solution. It is also worth noting that the matheuristic method manages to find good solutions on all instances, but can have a long computational time on the instance containing more than 100 polygons, where the time limit is increased to 1000 seconds.

Results of the merge-and-split heuristic Examples of solutions found by the merge-and-split heuristic are given in Figures 6c, 7c, 8c, 9c. The detailed results are given in Table 4, which shows that the solutions found cover all polygons using a number of beams that is rather low given the number of polygons to be covered. Moreover, the merge-and-split heuristic finds several feasible solutions during the process.

Figure 10 shows how the quality of the best solution found evolves for the *Western*

Table 2: Matheuristic results

Instance name	Time limit	ILP time limit	BGM	Nb it	Best solution			
					$\|P_s\|$	$\|B_s\|$	$MSRS$	Fig.
France	100	10	BGM1	7	22/22	7	0.0624	6a
France	100	20	BGM2	13	22/22	9	0.05137	6b
Italy	10	5	BGM1	3	30/30	12	0.0168	7a
Italy	10	5	BGM2	3	30/30	12	0.0168	7b
West Europe	200	50	BGM1	3	92/92	37	0.0223	8a
West Europe	200	50	BGM2	3	92/92	40	0.0187	8b
Central Europe	1000	300	BGM1	7	103/103	18	0.0692	9a
Central Europe	1000	300	BGM2	3	103/103	20	0.0677	9b

Table 3: Results of the matheuristic and the merge-and-split heuristic

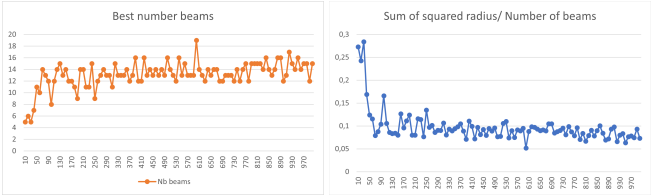
Instance name	Time limit	Matheuristic results					Merge-and-split results				
		ILP time limit	Nb it.	Best solution			$NbMergeMax$	Nb sol.	Best solution		
				$\|P_s\|$	$\|B_s\|$	$MSRS$			$\|P_s\|$	$\|B_s\|$	$MSRS$
F	100	10	7	22/22	7	0.062	5	8	22/22	7	0.069
I	10	5	3	30/30	12	0.017	5	88	30/30	15	0.027
W	200	50	3	92/92	37	0.022	20	14	92/92	21	0.49
C	1000	300	7	103/103	18	0.069	50	5	103/103	12	0.12
C1	50	10	4	52/52	19	0.037	5	15	52/52	18	0.38
C2	50	10	2	51/51	6	0.13	5	1	51/51	8	0.10

Table 4: Merge-and-split heuristic results

Instance name	Time limit	$NbMergeMax$	Nb solutions	Best solution			
				$\|P_s\|$	$\|B_s\|$	$MSRS$	Fig.
France	100	5	9	22/22	6	0.0556	/
France	100	5	8	22/22	7	0.0685	6c
Italy	10	5	88	30/30	15	0.02719	7c
Italy	10	5	126	30/30	12	0.01919	/
West Europe	200	15	64	92/92	17	0.0644	/
West Europe	200	20	14	92/92	21	0.488	8c
Central Europe	200	15	3	103/103	10	0.1381	/
Central Europe	1000	50	5	103/103	12	0.120	9c

Europe instance. More precisely, we consider a list of different *TimeLimit* from 10s to 1000s with a step of 10s. For each run, we report the best values found for the number of selected beams (Figure 10a) and the mean squared radius sum (Figure 10b). The results obtained show that the heuristic method manages to globally decrease the squared radius sum and increase the number of beams, especially during the first iterations. But the improvement is not significant when increasing the total time limit.

Comparison between the two methods Globally, given the time limit to consider for each instance, the matheuristic method provides solutions of better quality: the number of beams selected is higher and the mean squared radius sum is lower. However, the computation time is higher for the matheuristic. Indeed, the merge-and-split heuristic manages to find several solutions covering all the polygons, while the matheuristic



(a) Number of beams (b) Squared radius sum

Figure 10: Impact of the time limit on the performance of the merge-and-split heuristic (time limit on the x-axis)

takes a few iterations before finding a solution covering all the polygons. The heuristic method is also capable of directly considering the MSRS minimization objective, while the matheuristic only minimizes the squared radius sum (SRS). Based on the $x_{b,r}$ variables used in the ILP model, MSRS can be expressed as $\sum_{b\in B, r\in R} r_b^2 x_{b,r} / \sum_{b\in B, r\in R} x_{b,r}$, but such a formulation is not directly linear, hence the ILP model only considers the SRS objective function. To show the difference between MRSR and SRS, Figure 11 gives the evolution of these two quantities along the iterations of the matheuristic, for the Central Europe instance. On this figure, we can see that a solution covering all the polygons is found at the second iteration, and after that decreasing the squared radius sum of the beams does not improve the mean squared radius sum. On the contrary, for the merge-and-split heuristic, the criteria optimized can be easily changed. Indeed, each time a solution is found, we can evaluate its quality according to the selected criterion.

Figure 11: Evolution of different criteria through iterations with matheuristic method

For both methods, the difficulty of an instance does not entirely rely on the number of polygons to cover. For example, it is easier to cover the 30 polygons of the Italy instance than the 22 polygons of the France instance. The reason for this is that the polygons in Italy are more dispatched, hence it is easier to create small beams while respecting the beam separation constraints. As an illustration, for the Italy instance, the heuristic finds 10 times more solutions in 10 times smaller CPU time than for the France instance. It is also much easier to solve the Western Europe instance than the Central Europe instance, because the polygons involved are more spread.

6 Conclusion

In this paper, we considered a problem associated with the design of a telecommunication satellite used for television broadcasting on geographic areas, for which a set of beams of different sizes must be defined to cover a set of polygons, considering

antenna mechanical constraints represented as a graph coloring problem. To face the combinatorial issues and find solutions that are industrially feasible, we proposed two different methods. The first one is a matheuristic method that is built upon an ILP formulation and uses an evolving pool of candidate beams until finding a feasible solution. This matheuristic produces good-quality solutions but can require a long computational time. The second method, called the merge-and-split heuristic, iteratively constructs the beam layout by updating a set of beams step-by-step through local merging operations. This second method is faster and robust to large scale instances. Several perspectives can be listed for this work. For the matheuristic approach, some unused beams could be deleted from the pool of candidate beams throughout the iterations, to alleviate the ILP model. We could also design other methods to fill in the pool of beams with potential relevant ones. Moreover, to handle the mean squared radius sum in the matheuristic instead of the squared radius sum, we could use Linear-fractional Programming. For the merge-and-split heuristic, other merging methods should be looked for, in particular to better identify the reasons for the non-colorability at each merging step. A last perspective is to define a hybrid method where the matheuristic and the merge-and-split heuristic could share solutions or sets of relevant beams.

REFERENCES

Biedl, T., Biniaz, A., and Lubiw, A. (2021). Minimum ply covering of points with disks and squares. *Computational Geometry*, 94:101712.

Brélaz, D. (1979). New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256.

Camino, J.-T. (2017). *Co-optimisation charge utile satellite et systeme télécom*. PhD thesis, Toulouse 3.

Camino, J.-T., Artigues, C., Houssin, L., and Mourgues, S. (2016). Mixed-integer linear programming for multibeam satellite systems design: Application to the beam layout optimization. In *2016 Annual IEEE Systems Conference (SysCon)*, pages 1–6.

Camino, J.-T., Mourgues, S., Artigues, C., and Houssin, L. (2014). A greedy approach combined with graph coloring for non-uniform beam layouts under antenna constraints in multibeam satellite systems. In *2014 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, pages 374–381.

Contardo, C. and Hertz, A. (2021). An exact algorithm for a class of geometric set-cover problems. *Discrete Applied Mathematics*, 300:25–35.

Fowler, R. J., Paterson, M. S., and Tanimoto, S. L. (1981). Optimal packing and covering in the plane are np-complete. *Information processing letters*, 12(3):133–137.

Hammill, C. W. and Dishaw, K. O. (2004). Satellite beam pattern for non-uniform population distribution. US Patent 6,813,492.

Kyrgiazos, A., Evans, B., and Thompson, P. (2013). Irregular beam sizes and non-uniform bandwidth allocation in hts multibeam satellite systems. In *31st AIAA International Communications Satellite Systems Conference (ICSSC)*.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer.

Welzl, E. (2005). Smallest enclosing disks (balls and ellipsoids). In *New Results and New Trends in Computer Science: Graz, Austria, June 20–21, 1991 Proceedings*, pages 359–370. Springer.