Building insulation renovation: a process and a software to assist panel layout design, a part of the ISOBIM project.

Michel Aldanondo, Julien Lesbegueries, Andrea Christophe, Élise Vareilles, Xavier Lorca

Building insulation renovation: a process and a software to assist panel layout design, a part of the ISOBIM project.

Michel Aldanondo[1] Julien Lesbegueries[1] Andréa Christophe[1,2] Elise Vareilles[3] Xavier Lorca[1]

[1] Université de Toulouse / IMT Mines Albi / Centre Génie Industriel – Albi, France
[2] Armines – Paris, France
[3] Université de Toulouse / Isae-SUPAERO / DISC – Toulouse, France
firstame.lastname@mines-albi.fr, firstame.lastname@isae-supaero.fr

Abstract –

In order to lower the huge carbon dioxide emissions due to building heating, the goal of this communication is to show how the design of insulation panel layout can be strongly computer-aided during façade building energy renovation projects.

The work has been achieved during an applied collaborative research project (ISOBIM), whose purpose is to develop open web integrated decision-aided tools to support a whole digital engineering retrofitting processes using modular timber framing panels.

First, the panel layout design problem is analyzed and permit to identify a four steps design process supported by three data models. Then, once the problem is preprocessed to remove some singularities, an automatic layout design algorithm is proposed and discussed. An example illustrates the propositions and support discussions.

Keywords –

Building insulation renovation, Panel layout design, Aiding design system, Constraint-based problem

1 Introduction

According to the European commission [1] "Today, roughly 75% of the European Union (EU) building stock is energy inefficient. This means that a large part of the energy used goes to waste. Such energy loss can be minimized by improving existing buildings". The same report enhances that such building renovation can reduce the EU's total energy consumption by 5-6% and lower carbon dioxide emissions by about 5%.

Consequently, many European and national research agencies have launched research programs to encourage the systematic renovation of buildings. In France the national research agency (ANR) supports a collaborative project called ISOBIM. The goal of the project is to develop open web integrated decision-aided tools to support a digital engineering retrofitting processes using modular timber framing panels. In this project we consider "heavy" retrofitting panels, meaning that they integrate doors and windows in opposition with "light" solutions that by-pass them, the reader can consult [2] or [3] for more details.

It is important to note that "light solutions" require a significant amount of work in sizing, cutting and fitting the panels which must be done on the building site during the renovation. In contrast, "heavy solutions" assume that all panels are prefabricated in the factory and then assembled without rework on site. The "light solutions" are more often related to a craft approach while the "heavy solutions" correspond more to an industrial activity. The solutions proposed in this article are fully within the scope of an industrial approach which allows to reduce the manufacturing and assembly costs and the duration of the renovation on site.

In order to provide ideas about "heavy solution", some order of magnitude about panels used in "heavy solutions" are provided. In terms of dimensions, maximum panel sizes are around 3.5 meters by 13 meters because of transportation constraint. In terms of weight such a panel (3.5x13) can weigh between 1.5 to 4 tons. For cost, including manufacturing and installation, a panel without any door or window is between 150 to 250 €/m2 in Europe. In terms of energetic performance improvement, it is too much case dependent to provide exact result, but such solution can be used only if the heating cost are at least divided by 2 to 4.

Considering the "heavy solution", some global retrofitting process models can be found in the literature. For our work, we consider the one published in [4] that considers four main steps: (i) building digitalization, (ii) panels layout design, (iii) panel manufacture planning (iv) on site project management (figure 1).

Within the framework of the ISOBIM collaborative research project, these 4 steps have been distributed to

various partners and we are in charge of the second step (panels layout design) and with the transition between the first and second step (preparation of the layout design just after building digitalization). Consequently, this paper is dedicated to panel layout design.

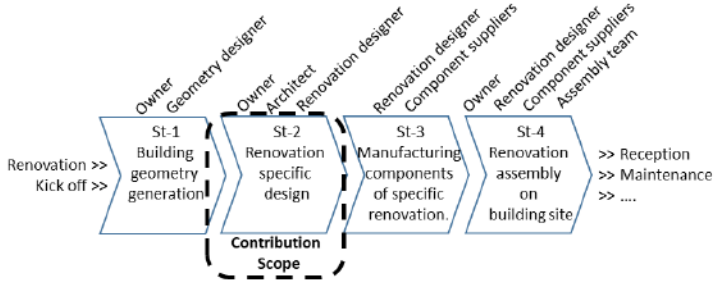

Figure 1. Work situation

The research work and this communication follow a progressive problem-solving approach where three issues are successively addressed. First, the tasks of the complete design process are defined. Then the data feeding and resulting from these tasks are identified and modeled. Finally, the core task of the design is instrumented with a solution finding algorithm.

Consequently, the contribution is organized as follows. In a first section, we will discuss and propose a process model that starts after the building digitalization and finishes after the final design of the panel's layout, shown as the "contribution scope" in figure 1. In a second section, we will discuss and propose three building facade models that support the tasks of the previous process. In a third part we will discuss and propose an algorithm that can automate the panel layout design.

2 The layout design process

The goal of this section is to clarify and to detail the tasks necessary before the panel layout design. We will first detail the need of a manual design step and then the need of a preparation step before automatic design. We will conclude with the global process and add a final correction step.

2.1 About a manual layout design step

According to some authors, as Hillebrand et al. [5] or Barco et al. [6], the algorithms that are able to solve the panel layout design problem requires strong hypothesis relevant to the considered entities.

We follow these ideas and first assume that our automatic layout design algorithm considers only rectangle building facade and only rectangle panels. Consequently, facades with triangular gable ends will require manual layout.

We also assume that the panels must be fixed at the top (mainly for support) and bottom (mainly for wind). As a consequence, specific parts of the facade as acroterion (facade top) or sub-basement (facade bottom) that don't have resisting area at their top or bottom will also require a manual layout design.

2.2 About preparation before manual design

If building facades have windows and doors that will be replaced and integrated into the panels, they also can have multiples singularities as: air conditioning unit, solar panel, balcony, lights, garage door, pipes... For all these singularities, it must be decided during the preparation task, if they should be associated with a hole in the panel or if they should be by-passed by the panels. Rectangle elements gathering: (i) singularities associated with panel holes, (ii) windows and (iii) doors will be called from now panel "outin". Elements gathering singularities by-passed by panels will be called from now panel "outout".

As we have assumed before, panel are fixed at their top and bottom with mechanical supports. These supports are sealed in the facade at specific places called resisting areas that can handle high effort, note that a large isolating panel (12 meters by 3 meters) can weight up to 4 tons. During the preparation task, these specific areas must be identified on the building geometry by building experts and correspond most of the time with floor-end, partition-wall or their crossing. Furthermore, for automatic layout algorithm, they should be approximated by support lines where panel supports will be sealed. These support lines can be either vertical or horizontal.

2.3 Proposed layout design process

Given previous assumptions, we propose the following panel layout process in four steps (figure 2).

The first one, "Preparation" identifies and transforms resisting areas in support lines and associates any facade element with either outin (that will be integrated in a single panel) or outout (that will be by-passed by the panels).

A second one, "Manual design" that allows the user to manually design the panels that can not be taken into account by the automatic algorithms as: non-rectangular panels, panels for acroterion or sub-basement.

A third one, "Automatic design" that will be detailed in a following section.

A last step, "Manual touch up" allows the architect or building owner to manually correct the proposed panel layout.

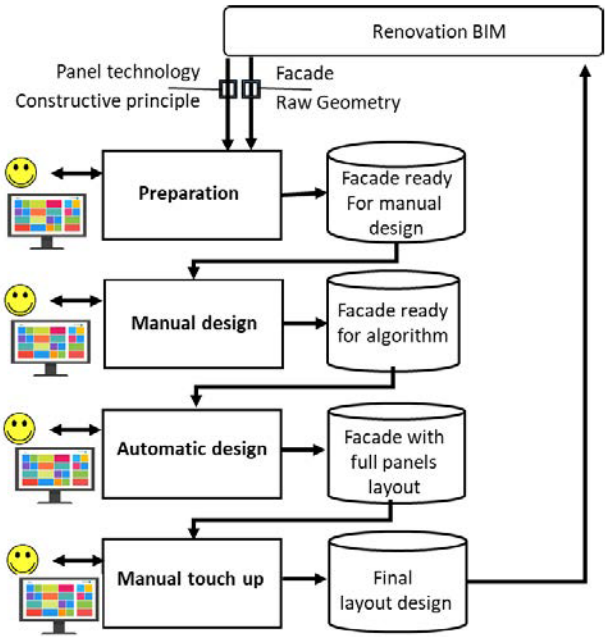

Figure 2. Panel layout design process

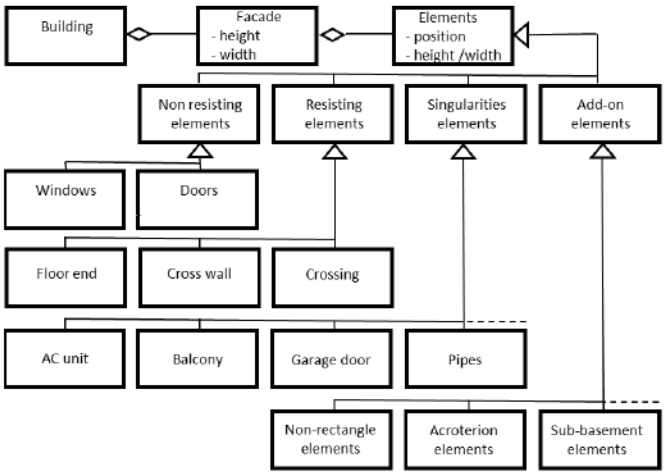

Figure 3. Facade raw geometry model

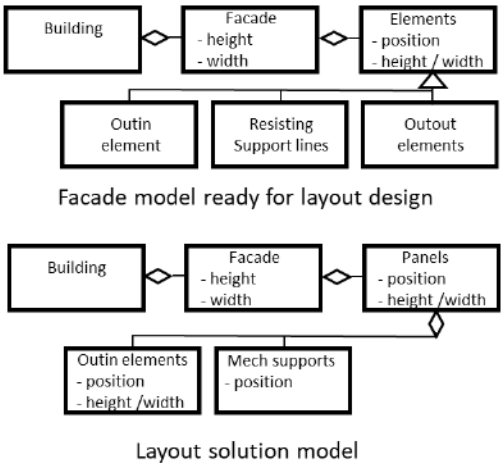

Figure 4. Facade model before and after design

3 The facade models

Given previous process, we describe in this section the tree facade models that are used during the layout design.

In order to represent the façade models, we use the standard class diagram of the Unified Modeling Language that can show how systems can be decomposed in sub-systems. We use mainly the two relations : (i) composition aggregation (AND represented with a diamond symbol) that shows the physical composition of the system and (ii) Generalization/Inheritance (OR represented with a triangle symbol) that shows various possibilities for each sub-system.

The first model (figure 3) is the first description or raw geometry of the building's facades. The building is a set of facades, where each facade gathers four kinds of elements: non-resisting, resisting, singularity and add-on. All of them are rectangle except the add-on non-rectangle element.

The second model (upper part of figure 4) is the facade ready for manual and automatic layout design. This model gathers only three kinds of elements: Resisting Support lines, Outin elements (that should be integrated in a single panel) and Outout elements (that should be by-passed by panels).

The last model shows the layout solution as a set of panels including outin elements and mechanical supports (lower part of figure 4).

4 The proposed layout design algorithm

In a first part, we summarize the problem inputs, the panel technology constraints and optimization criteria. Then we propose a small problem analysis in order to introduce our heuristic approach. Finally, we will present the main ideas of our layout design algorithm.

4.1 Inputs, panel constraints and criteria

Given previous elements, the layout design automatic algorithm only considers as an input the three elements: support lines, outin and outout, an example is given in figure 5. This case does not need any manual panel design because everything is rectangular and there is no special panel for facade top and bottom. According to most panel manufacturers each couple of balcony and window element must be associated in a single outin rectangle.

In terms of panel technology, a panel has minimum and maximum dimensions (width and height). For transportation constraints, each dimension should be less than the truck length (around 14 meters) and one dimension should be lower than the truck height (around 3.5 meters). The layout solution, is a set of panels that covers the facade except "outout". Panel dimensions should be larger than the integrated elements outin. For panel rigidity, a minimum distance between the outin edge and the panel border can exist. Each panel has a weight more or less proportional to its area. Lower panel border can be supported either by a lower horizontal support line or by its lower corners on a vertical or horizontal support line. Panels can have two orientations, horizontal or vertical. Horizontal panels have a width larger than height while it is the opposite for vertical ones. Panel providers prefer most of the time horizontal panels if this is possible.

The problem is consequently to find a set of panels that cover the facade. As many solutions can exist, different criteria can be considered. The most frequent are: minimize the number of panels, maximize the panel size or minimize the length of joins between panels. They all globally go towards the same direction that is to minimize heat loss.

4.2 Problem analysis

Given the proposed elements, this problem can be classified with respect to two types of problems: "cutting and packing" [7] and "product configuration" [8].

Cutting and packing problems deal with small and large objects and try to place the small ones into the large ones. Classic examples are: steel bars and plates production, fabrics cutting, vehicle loading, memory allocation... For problem with two dimensions, as our panel layout problem, four characteristics impact the solution complexity [9] :(i) the quantity of large objects, 1 or more than 1 (ii) the quantity of small objects, given as an input or to be identified, (iii) the size of small objects, identical or not, and (iv) the sizes of small objects, given as an input or to be identified.

Product configuration considers that a product is a set of predefined components (standard or configurable) that respect various constraints (coming from marketing, design, manufacturing…). The characteristics impacting the problem complexity are: (i) the quantity of components in a product: given as an input or to be identified, (ii) the fact that the components are: standard (frozen dimensions) or configurable (dimensions to be identified but limited) and (iii) the constraints complexity acting on all of these objects.

For these two approaches, when the quantity of small objects and relevant dimensions are given as an input, it is quite easy to define the space of solutions as a tree and to define systematic search algorithms. When the problem size and/or complexity are moderate, it is possible to scan the total solution space and to identify optimal solutions by combining solving and filtering algorithms. When we quit these kinds of situations, it is much more difficult to define systematic algorithm and the definition of heuristics becomes most of the time necessary.

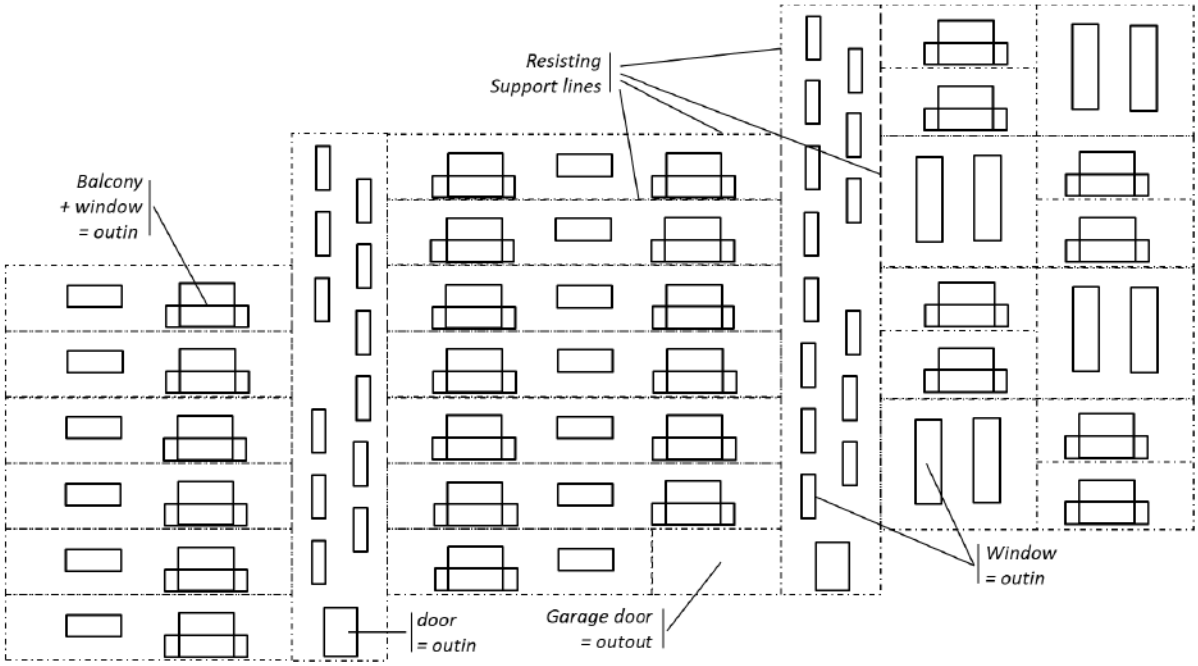

Figure 5. Example of facade model ready for automatic layout design

It sounds rather obvious that our panel layout design problem is at the top of the complexity scale due to: (i) the quantities, dimensions and positions of all the small objects (panels) than must be identified and (ii) the diversity and quantity of geometric constraints that should be respected. The most delicate characteristic to handle is that the number of small objects is to be identified. This effectively implies that the variable quantity of the problem (or problem size) is not known when the algorithm is launched. Authors, as Barco et al. [10], have modeled this whole problem as a constraint satisfaction problem and shown that there is no generic constraint-based algorithm able to handle it.

As this work belongs to a collaborative operational project, it is necessary to provide a robust solution that works with all previous characteristics, we therefore designed a heuristic-based algorithm. But we also try to figure out how constraint-based solvers could deal with our problem. The next section presents the main ideas of our automatic panel layout design algorithm.

4.3 Layout design algorithm

After a small introduction, we describe the general structure algorithm then we detail the algorithm for horizontal panels and finish with some experimental results and discussion.

4.3.1 Algorithm key parameters

The heuristic we propose is parameterized according to two key parameters:

- Orientation strategy: HORIZONTAL or VERTICAL. This parameter defines the preferred orientation of the panels defined by the user. A panel can be horizontal: in this case it is attached at the bottom and top of support lines, or vertical: in this case, the panel is fixed on its 4 corners.
- Direction strategy: RIGHT or LEFT. This parameter defines if the heuristic path starts by going to the right or to the left.

However, regardless of the chosen parameters, the heuristic as a whole travers the facade from bottom to top, i.e., the first panels will be positioned at the bottom and the last ones at the top. This is compliant with the assembly process that always start at the bottom and goes up floor by floor.

In order to start the design algorithm, we need to set some initial starting points. These points will be stored in a list, that will be filled while traveling through the facade, and removed once taken into account. The termination predicate of the algorithm is then the emptiness of this list. Initially, the starting points list is filled with each lower right and left corner of the perimeter of the facade.

4.3.2 Algorithm general structure

The general algorithm behavior is a loop running while starting points remain in a stack ordered from the bottom to the top (Algorithm 1). While running, it tries to find a panel in the desired orientation and direction. If the first attempt fails, another try is made with the other available orientation. For instance, if the algorithm is launched with RIGHT and HORIZONTAL parameters, the algorithm will first look for a right-directed horizontal panel, and if it's not possible a right-directed vertical panel. If a panel is found, it is added to the solution list that will be returned at the end of the loop. Consequently, possible new starting points are added based on the new panel : for right direction the two added points are panel top left and bottom right, while for the other they are top right and bottom left.

Algorithm 1 General layout design structure.

```
1: Let Panel := x, y, width, height, out-ins
2: Let p a Point variable x, y
3: Let panel a panel variable
4: direction ← RIGHT
5: orientation ← HORIZONTAL
6: startingPoints ← findStartingPoints(F, direction) // ordered stack
7: Let solution := [Panel]
8: solution = []
9: while startingPoints is not empty do
10:     p ← selectNextPoint(startingPoints) // bottom to top, then direction
11:     panel = addPanel(p, F, cp, direction, orientation)
12:     if panel is null then
13:         panel = addPanel(p, F, cp, direction, other(orientation))
14:     end if
15:     if panel is null then throw Error (No solution found.)
16:     else
17:         solution.add(panel)
18:         addStartingPoints(startingPoints, panel)
19:     end if
20: end while
21: return solution
```

Lines 1 to 8 are variable definitions and input parameters:

1. A Panel data structure is defined (basically a rectangle with some out-ins)
2. A p variable of type Point
3. A panel variable of type Panel
4. Direction parameter set
5. Orientation parameter set
6. A set of points is calculated to start the algorithm, using one of the preparation algorithms
7. The type of the return value is set (an array of panels)
8. Return value set

The addPanel() function used in Algorithm 1 is a simplification for addHorizontalPanel() and addVertical Panel() combination. They are very similar and differs only on their supporting technique. Both functions are given a starting point and have for main goal to find the best second point defining the largest valid panel. For clarity, the first function is detailed in the next paragraph.

4.3.3 Horizontal panels detailed algorithm

The horizontal panel design algorithm is divided into 2 main actions (see Algorithm 2). First it builds a hash map of possible coordinates - for each possible y, possible x are stored (line 2). Then the biggest valid panel is found by browsing ordered possible coordinates (line 25 to 35).

Given a starting point p of coordinates {xp, yp}, the algorithm is looking for a support-line that touches p. Then a geometry collection of all contiguous support-lines found from the first one is gathered (findSupportLines() line 3). As a result, we can obtain a horizontal line that has got minimum and maximum X coordinates.

We store then the maximum x (or minimum x if direction is LEFT) in the bottomMaxX variable (bottomMinX). This gives us the range [xp, bottomMaxX] ([bottomMinX, xp]) of all the possible X coordinates regarding the bottom side of the panel (line 5 and 16). But we have to test also the possible Xs regarding the top side of the panel.

Then the possible Y coordinates are calculated, given the initial p point. For each possible y, we can deduce the maximum x (minimum x) for both bottom and top support lines (by using findSupportLines() again - line 9 and 19). Once ordered, possible coordinates are stored.

The second step of the algorithm is a double loop trying to build and return the biggest valid panel candidate. By ordering the coordinates we ensure that the first valid panel found will be the biggest one. The valid function evaluates the validity of the following constraints of each panel candidate:

Algorithm 2 Horizontal panel design algorithm.

```
1: function addHorizontalPanel(point, facade, cp, direction)→ Panel
2:         coordCandidates := Map{Y, [X]} = {} // possible x list for each y
3:     bottomSupportLines ← findSupportLines(point)
4:     if direction == RIGHT then
5:         bottomMaxX ← maxX(bottomSupportLines)
6:         possibleYs ← findPossibleYs(point)
7:         possibleYs ← possibleYs.reverse()
8:         for y in possibleYs do
9:                 maxX ← min(bottomMaxX, maxX(findSupportLines(point.x, y)))
10:            possibleXs ← findPossibleXs(point, maxX, y)
11:            possibleXs ← possibleXs.reverse()
12:            coordCandidates.put(y, possibleXs)
13:        end for
14:    else/* direction == LEFT */
15:        bottomMinX ← minX(bottomSupportLines)
16:        possibleYs ← findPossibleYs(point)
17:        possibleYs ← possibleYs.reverse()
18:        for y in possibleYs do
19:                minX ← max(bottomMinX, minX(findSupportLines(point.x, y)))
20:            possibleXs ← findPossibleXs(point, minX, y)
21:            coordCandidates.put(y, possibleXs)
22:        end for
23:    end if
24:    /* Browses possible coordinates for the 2nd point of panel */
25:    for x in coordCandidates.keys() do
26:        for y in coordCandidates[x] do
27:            panel ← createCandidate(point, x, y)
28:            if isValid(panel, facade, cp) then
29:                addResistingElements(panel)
30:                addFixingElements(panel)
31:                return panel
32:            end if
33:        end for
34:    end for
35:    return null
36: end function
```

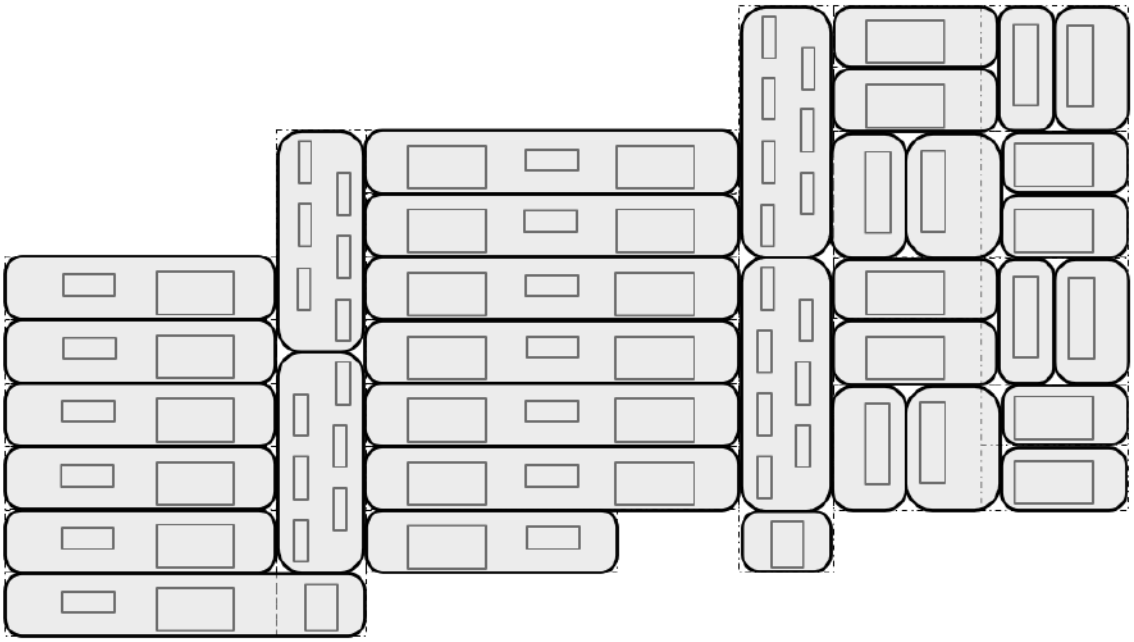

Figure 6. Horizontal solution for the example of figure 5

- Dimensions validation: panel dimensions must respect the minimum and maximum values of width and height coming from the constructive principle,
- Borders validation: if the distance between a panel border and a facade edge or outout edge is greater than 0 but lesser than the minimal width / height of a panel, the panel is not valid,
- Outin containment validation: outin must be fully contained in one panel,
- Other panels overlapping validation: panels must not overlap the other panels (the ones of the solution or the ones manually placed initially),
- Resistance validation: the minimal resistance of the resisting support lines must be compliant with the weight of the panel (computed from its area).

4.3.4 First results and discussion

When this algorithm is used with the example of figure 5, a solution gathering 34 panels is found and shown in figure 6 (panels have rounded corner to see them easily). It can be seen that even if horizontal panels are preferred, at least 12 vertical panels are necessary for staircase and duplex with two floors windows.

In terms of optimality, it must be pointed out that the panel orientation has a major influence on the solution and can sometimes decrease the optimality level of a solution. In the example of figure 6, it is possible to see that when the panel at the bottom of the right staircase is designed, the horizontal orientation forces a small horizontal panel (with a door in the center). Architects would most of the time prefer a single large vertical one that would replace two panels as shown in figure 7.

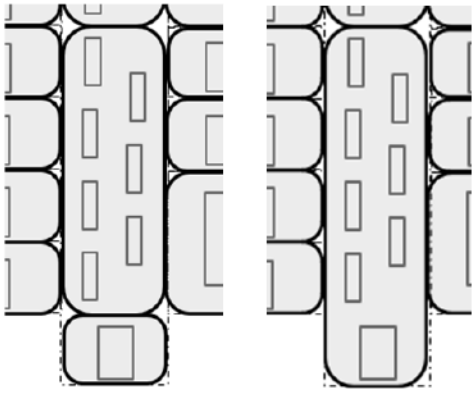

Figure 7. A more optimal result

A more optimal algorithm could be to compute for each panel design the two alternatives (horizontal and vertical) and to keep the best one. This could have been easily done, but this would lead to layout solutions mixing the two orientations. These solutions can be confusing because panels don't seem to be logically arranged and are very delicate to manage during on-site assembly due to a lack of a logical assembly.

This is why we have kept the strength of the orientation parameter in the algorithm, but add a last manual correction, with the manual touch up step, at the end of the design process after automatic design (figure 2). During this last step, manual modification can be inputted in order to change some panel size, to regroup panels or to line up panel edges. This can improve the optimality of the solution and also take into account specific requirements of the building architect and/or owner.

In order to avoid the previous hazardous mix of panel orientations, some authors, as Aldanondo et al. [11], propose to decompose the whole facade in sub-facades either fully horizontal or vertical (meaning accepting only horizontal or vertical panels) before launching the layout algorithm. The main interest is that the problem size can be decreased significantly. But the drawback is that some cases cannot be considered. For example, the right part of our example cannot be processed with fully vertical or horizontal panels. But a mix of the two approaches would merit to be investigated.

The presented solution was obtained with a direction from left to right. When the opposite direction is used (left to right), the modification of the layout concerns only the right part of the facade and are shown in figure 8. This is the result of the outin (windows and balconies) that forbid to use horizontal panels with a larger width.

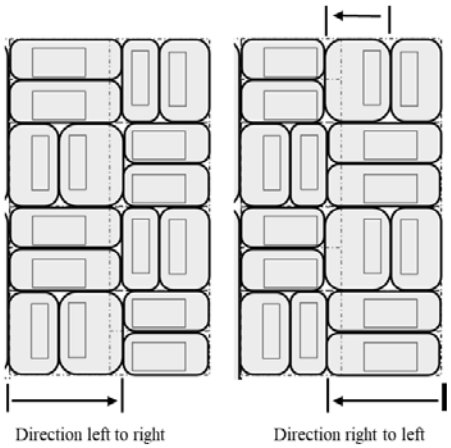

Figure 8. Comparing different directions

5 Conclusion

Our goal was to present a global supported approach to assist the panel layout design for Building insulation renovation. We have proposed a four steps process, three data models and a layout design algorithm that allows to find solutions with a preferred orientation.

As far as we know, we couldn't find in the communities close to the subject, "cutting and packing" and "product configuration", any exact approach able to model and to optimally solve the problem as defined. The proposed algorithm is a greedy heuristic which purpose

is to design panels as larger as possible. As panels get larger, the panel quantity and joint length get smaller. Consequently, the building energetic performances are much better which fulfill the ultimate goal. However, our automatic algorithm only considers rectangular shapes limited by resisting zones. In order to have an operational solution that works, a manual design task where more or less "every design is possible" can be achieved before automatic layout design.

At an operational level, the result is quite robust, it has been used on various facades with no problem. The feedback from the project partners is quite good, they can generate in less than one hour panel layout solutions that require manually at least a day of work. Consequently and this is a key interest, our approach allows us to quickly test and compare different solutions, which is usually never done when layout design is achieved manually.

Some comments about the sustainability aspects of the proposition can address two aspects : the technical "heavy" panel-based solution, the proposed process and software. About the panel solution, we cannot provide any detail figures about energy efficiency, we just know that such project is launched when the cost reduction is at least 50%. But we can say that this industrial process is the only solution in order to reduce building carbon dioxide emissions. For example, in France : (i) much more than 12 million of building apartments are considered as energy strainer (ii) each year no more than 300 thousand of new building apartments are built, (iii) consequently it will take more than 40 years to upgrade le French building stock without industrial retrofit. About the process and software, it is not possible to quantify the benefit of the computerized assistance on sustainability. But as we said in the previous section, the ease and rapidity in the elaboration and comparison of insulation solutions offered by our proposal undoubtedly allows the development of more effective solutions.

At the present time we are trying to use the constraint-based solver Choco Solver library [12] to reprogram the algorithm in order to have a more declarative program. This will allow to update or modify much more easily the constraints relevant to the panel technology and/or constructive principle of the panels. Another research direction is to develop an approach to analyze the "manual touch up" steps in order to systematize the search for improvements concerning the proposed process and algorithm.

Acknowledgments

The authors want to thank the French Research National Agency (ANR) for funding the ISOBIM project that has supported the presented works.

References

[1] European Commission. In focus: Energy efficiency in buildings. In https://ec.europa.eu/info/news/focus-energy-efficiency-buildings-2020-lut-17_en

[2] Urbikain MK. Energy efficient solutions for retrofitting a residential multi-storey building with vacuum insulation panels and low-E windows in two European climates. *J. of Cleaner Production*, Vol 269, Elsevier, 2020.

[3] International Energy Agency. Prefabricated Systems for Low Energy Renovation of Residential Buildings. *IEA ECBCS Annex 50* or https://nachhaltigwirtschaften.at/resources/iea_pdf/iea_ecbcs_annex_50_anhang10b-moduledesign.pdf 2011.

[4] Aldanondo M., Barco-Santa A., Vareilles E. and Falcon M. Towards a BIM Approach for a High Performance Renovation of Apartment Buildings. In *Proceedings of the PLM 2014 conference*, Springer, pages 21–30, Yokohama, Japan, 2014.

[5] Hillebrand G., Arends G., Streblow R., Madlener R. and Müller D. Development and design of a retrofit matrix for office buildings. *Energy and buildings*, Vol 70, pages 516-522, Elsevier, 2014.

[6] Barco A., Vareilles E., Aldanondo M. and Gaborit P. A Recursive Algorithm for Building Renovation in Smart Cities. *Lecture Notes in Computer Science*, LNAI Vol 8502 Springer, pages 144–153, 2014.

[7] Kendall, K., Daniels, K. and Burke, E.. Cutting, packing, layout, and space allocation. *Special issue Annals Operation Research*. n°179, p. 1-3, 2010.

[8] Felfernig, A., Hotz, L., Bagley, C., and Tiihonen, J. *Knowledge-Based Configuration: From Research to Business Cases*. Morgan Kaufmann, 2014.

[9] Huang, E., and Korf, R.E. Optimal rectangle packing: An absolute placement approach. *J. of Artificial Intelligence Research*. Vol. 46, p. 47-87, 2012.

[10] Barco-Santa, A., Fages, J.G., Vareilles, E., Aldanondo, M., and Gaborit, P. Open packing for facade-layout synthesis under a general purpose solver. *In CP conference - Lecture Notes in Computer Science*, vol. 9255, pages 508-523. 2015.

[11] Aldanondo M., Vareilles E., Lesbegueries J., Christophe A. and Lorca X. Buildings energetic improvement: first elements about isolating panels layout design. In *Proceedings of the conference 14th IFAC Workshop on Intelligent Manufacturing Systems*, pages 487–492, Tel Aviv, Israel, 2022.

[12] Prud'homme C. and Fages J. An introduction to choco 3.0 an open-source java constraint programming library. In *CP solvers: modeling, applications, integration, and standardization*. Int. workshop, Uppsala Sweden, 2013.