Constraint-based design : two-dimensional insulating panels configuration

Andres Felipe Barco Santa

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

délivré par
École Nationale Supérieure des Mines d'Albi-Carmaux

---

présentée et soutenue par

Andrés Felipe Barco Santa

le 20 septembre 2016

Constraint-Based Design : Two-Dimensional Insulating Panels Configuration

---

École doctorale et discipline ou spécialité :
EDSYS : Informatique et Génie Industriel

Unité de recherche :
Centre Génie Industriel, Mines Albi

Directeur(s) de Thèse :
M. Michel Aldanondo, Professeur, Mines Albi
M^me^ Élise Vareilles, Maître-Assistant HDR, Mines Albi

Autres membres du jury :
M^me^ Nadège Troussier, Professeur, Université de technologie de Troyes *(Présidente)*
M. Pierre-Alain Yvars, Professeur, Institut supérieur de mécanique de Paris *(Rapporteur)*
M. André Thomas, Professeur, Ecole des Technologies et Industries du Bois *(Rapporteur)*
M. Paul Gaborit, Maitre-Assistant, Mines Albi *(Examinateur)*
M. Juha Tiihonen, Opettaja Konferenssit, University of Helsinki *(Examinateur)*
M. Lars Hvam, Professeur, Technical University of Denmark *(Examinateur)*

Para mi mamá por obligarme a levantarme para ir al colegio.

Pour Élise pour supporter mon visage grincheux et mes blagues.

Para Katherine, por todo.

Abstract

The two-dimensional Cutting & Packing problem consists in allocating a set of two-dimensional items into a set (possible singleton) of two-dimensional large objects in such a way that items are completely contained in the objects without overlapping. A particular instance of two-dimensional Cutting & Packing problem arises in the context of thermal building renovation. This special case deals with the design of an *insulating envelope* by packing a set of rectangular panels over a rectangular façade surface. The envelope is used to reduce the thermal transfer between the interior and the exterior of the building in the aim of reducing the building's energy consumption. Five particularities makes this problem novel and interesting. Firstly, the number of panels to design an envelope and their size are not known a priori (tough their size is bounded). Secondly, due to manufacturing conditions, each window and door over the façade surface must be covered by one and only one panel. Thirdly, panels are attached in specific areas over the façade strong enough to support the weight added by the envelope. Fourthly, in order to guaranty a perfect external insulation, no holes and no panels overlapping are allowed. Lastly, to reach a good building thermal performance while minimizing the retrofit global cost, the envelopes should be composed of the minimum number of panels.

The motivation behind this dissertation lies in the existing need for designing external insulating envelopes for residential buildings. The design of an envelope depends on geometry and strength of the façade to be renovated as well as in the architects preferences and artistic flair. The result of such design is a numerical model (nomenclature) used as input for the manufacturing of panels (respectively envelopes). In consequence, the problem is not only treated as a two-dimensional Cutting & Packing problem but also as a *configuration* one. These facts have led us to use the constraint satisfaction framework to reason and to solve the envelopes design problem. We consider that the mathematical model of constraint satisfaction fits neatly in the constrained nature of Cutting & Packing problems as well as configuration problems. In particular, its declarative view allows a clear knowledge representation, it divides modelling from solving and it allows the inclusion of an objective function. The thesis then builds upon the constraint satisfaction framework to solve design problem under different techniques from artificial intelligence and operational research. As such, the dissertation presents two major contributions.

Due to the unusual characteristics of the industrial case, the description and constraint-based model of the design problem becomes the first contribution of the thesis. In particular, the industrial renovation description, the envelopes design as a Cutting & Packing problem and the support system for architects decision-making as a product configuration software are discussed. Requirements, limitations and design knowledge extracted by stakeholders and architects have been compiled into a constraint satisfaction model that acts as foundation of the algorithmic solutions and a resulting decision support system.

The second contribution dwells in five *constraint-based algorithmic solutions* for the design problem; InDiE, a manual interactive solution to guide architects design; GaLaS, an automatic solution that packs panels on the fly in a greedy fashion; CaSyE, an automatic solution that packs panels considering the geometry of the façade with a cutting approach; SkEdE, a manual solution for architects sketching that uses Choco as underlying solver; OpackS, an automatic solution that applies filtering and search from the underlying solver Choco and treats the problem as a constraint optimization problem.

Additionally, the dissertation introduces a product configuration software, specialized version of a decision support system, for the design/configuration of panels (respectively envelopes). The system makes a division between key configuration tasks that allows the transparent manipulation of the architects preferences. Also, it includes some recommendation capabilities for some features in the configuration of panels. Evaluation is performed over some real-life facades in France and prove the validity of our methods. Conception and implementation of the thermal retrofit of buildings are then supported by this work.

Résumé

Les travaux de recherche présentés dans cette thèse se situent dans une problématique d'aide à la conception d'enveloppes isolantes pour la rénovation thermique de bâtiments résidentiels collectifs. Ces enveloppes isolantes sont composées de panneaux multifonctionels rectangulaires, configurables et préfabriqués en usine. Leur conception repose sur les cinq caractéristiques suivantes. Premièrement, le nombre de panneaux nécessaires pour concevoir une enveloppe ainsi que leur taille respective ne sont pas connus au début de la rénovation (mais leur taille est cependant bornée). Deuxièmement, en raison des contraintes de fabrication, chaque fenêtre et chaque porte présentes sur la façade à rénover doivent être insérées dans un et un seul panneau. Troisièmement, les panneaux sont fixés à des endroits spécifiques de la façade, assez résistants pour supporter leur poids, nommés zones d'accroche. Quatrièmement, ni trous (zone non couverte), ni chevauchements entre panneaux ne sont autorisés. Cinquièmement, afin de garantir une isolation thermique performante tout en minimisant son coût, les enveloppes doivent être composées d'un nombre minimal de panneaux. Au vue de la complexité de ce problème, nous restreignons nos travaux de recherche aux façades rectangulaires portant des menuiseries et des zones d'accroche rectangulaires.

Compte tenu des cinq caractéristiques énoncées et de l'hypothèse de forme rectangulaire des éléments traités (panneaux, façades, menuiseries, zones d'accroche), la conception des enveloppes est à la fois un problème de découpe et de conditionnement à deux dimensions et un problème de configuration. Ce problème est formalisé et traité comme un problème de satisfaction de contraintes et a pour but d'aider la conception dédites enveloppes isolantes. En tant que tel, la thèse présente deux contributions majeures.

En raison des caractéristiques originales du problème de calepinage de façades, sa description et sa formalisation comme un problème de satisfaction de contraintes est la première contribution de ces travaux de thèse. En particulier, la description du processus de rénovation industriel, la conception des enveloppes perçue comme un problème de découpe et de conditionnement et le système d'aide à la décision pour les architectes sont discutés. L'ensemble des exigences, des limites et des connaissances « métier » recueillies auprès des parties prenantes et des architectes a été intégré dans un seul modèle de satisfaction de contraintes et sert de base aux solutions algorithmiques et un système d'aide à la décision.

Deuxièmement, les solutions algorithmiques basées sur les contraintes constituent notre seconde contribution. En particulier, ces travaux de thèse présentent deux solutions manuelles et trois automatiques pour le problème de conception ; `InDiE`, une solution interactive manuelle pour guider pas à pas la conception des enveloppes ; `GaLaS`, une solution automatique pour générer des envelopes à partir d'un algorithme glouton ; `CaSyE`, une solution automatique pour générer des enveloppes à partir d'un algorithme de type « guillotine » ; `SkEdE`, une solution manuelle pour esquisser des enveloppes qui s'appuie sur le solveur `Choco` ; `OpackS`, une solution automatique pour générer des enveloppes exploitant le filtrage des contraintes et recherche du solveur `Choco`.

De plus, une maquette logicielle du système d'aide à la décision pour concevoir différentes solutions de calepinage, intégrant l'ensemble de nos solutions a été développée. Les solutions générées par les architectes qui utilisent notre maquette d'aide à la décision sont des modèles numériques d'enveloppes isolantes, dont la nomenclature peut être envoyée aux usines de fabrication. En tant que tel, notre maquette, qui utilise la satisfaction de contraintes comme modèle sous-jacent et met en œuvre les solutions algorithmiques proposées dans cette thèse, peut être classée à la fois comme un système d'aide à la conception par ordinateur ou *CAO* et comme un logiciel de configuration de produit ou *PCS*.

Acknowledgements

I would like to thank the people who make my PhD possible and, after a conscious reflection, I have realized that such a task is impossible. This has to do with the well-known phrase of, in my opinion, the most intelligent person ever existed: "If I have seen further, it is by standing on the shoulders of giants". Indeed, this phrase gets short when you realize all the human work needed to reach the society we currently are. Further, the phrase gets small when you comprehend the amount of human efforts needed to allow a human being as yourself, to eat, dress, travel, communicate and, in essence, be an integral human being. The chain of humans that allowed me to be who I am is vast in such an extend that a full acknowledgements is impossible. Thus, I would like to give you my thanks in a more general way. I am sorry if this acknowledgement disappoints you but is the only way to put in words what is in my mind and heart.

I would like to thank the pleasant and harmonic environment inside the Centre de Genie Industrial (CGI) at Mines d'Albi. Without the effort of its teachers/researchers, leaders and secretary, my tiny work would never be finished. Likewise, this research centre can only function properly because of the efforts of the administrative staff of the school and, in a great extend, to all the other workers that make the school a living community (cleaning staff, garage service, telecommunications office, etc.). I have to thank all of them. Among the same lines, the school is part of a small (really small!) warm city called Albi, whose progress and development is only possible by the common efforts of every citizen, every bus driver, baker, chef, waiter, gardener and so on. My deepest appreciation for all those honest workers who allowed me to have the highest life quality that I ever had (particular thanks to those who make possible to drink clean water from the comfort of my home). Over and above, none of the people living in Albi, including me, would have the necessary energy to get up every morning if the rice, meat and fruits that we consume were not produced by other kind of workers. People that take from our hands the responsibility to produce the wealth we consume so to allow us to put our minds in less critical aspects of human life. To those people in Albi, France or any part of the world, I say thanks, I promise you some day I will help you free your hands so we put our minds in the construction of a better world.

Andrés Felipe Barco
20 September 2016

Contents

The past lies like a nightmare upon the present.

The 18th Brumaire of Louis Bonaparte
Karl Marx, 1852

1 Introduction

Contents

This dissertation studies the framework of *constraint satisfaction* as a model for analyzing constrained industrial scenarios. The thesis is that the declarative model of constraint satisfaction is suitable for describing, reasoning and implementing consistent solutions for orthogonal *two-dimensional* `Cutting & Packing` problems with unknown number of configurable entities. The work is intended to assist architects decision-making in the context of industrialized building thermal renovation.

1.1 Foreword

The two-dimensional `Cutting & Packing` problem consists in allocating a set of two-dimensional items into a set (possible singleton) of two-dimensional large objects in such a way that items are completely contained in the objects with no-overlapping (Bennell et al., 2013). Different characteristics of the problem may vary depending on the targeted industry. Typical variations include the possibility of items rotation, maximizing the number of allocated items, minimizing the uncovered space or even the mandatory covering of space (Hopper and Turton, 2000). These kinds of problems appear in industries as lumber processing, glass and metal cutting, leather cutting, web-page design, microcircuits design and apartment design.

Given that two-dimensional Cutting & Packing problems are difficult problems, classified as NP-hard, scientists have been developing computer-based solutions to support decision-making in different industry sectors. In particular, techniques from artificial intelligence and operation research have shown their robustness in addressing these problems (Hopper and Turton, 2001a). For instance, the cutting stock problem, a particular case of Cutting & Packing often present in steel and aluminum industries, has been treated as a linear programming model, dynamic programming model and evolutionary model (Imahori et al., 2007). These kinds of solutions, however, are either to general or to specific, to solve all Cutting & Packing problems that arise in the industry. Above all, it is difficult to create general purpose heuristics that exploit the expertise knowledge of any given problem.

A particular instance of Cutting & Packing problems arises in the context of thermal building external renovation (Vareilles et al., 2013). This special case, termed as *Façade-Layout Synthesis Problem*, deals with the allocation of rectangular configurable panels over a rectangular façade surface. The set of allocated panels covering a façade is called an insulating envelope and is used to reduce the thermal transfer between the interior and the exterior of the building. The motivation behind this insulation is to achieve a reduction of the buildings energetic consumption that currently exceeds transportation and industry sectors (Pérez-Lombard et al., 2008). Five particularities make this problem novel and interesting:

1. The main characteristic is that the number of panels to design an envelope, as well as their sizes, is *not known a priori*. The size of panels, however, is bounded to a given interval.
2. Second, some rectangular areas inside façades (existing windows and doors) are meant to be *completely overlapped* by panels. Each area must be covered by one and only one panel at a time.
3. Third, panels are attached in specific *rectangular areas* that are strong enough to support their added weight.
4. Fourth, in order to guaranty a perfect external insulation, in addition to the non-overlapping condition, no holes are allowed in a packing solution.
5. Finally, to reach a good building thermal performance while minimizing the renovation global cost, the envelopes should be composed of the minimum number of panels.

One of the key problems in this setup is to propose a computational process that allows configuring the specific set of non-overlapping panels with respect to each façade to renovate. This process is referred to as *design*. The output of the design, i.e., an insulating envelope, is called a *layout plan*.

The motivation behind this dissertation lies in an existing need for designing external insulating envelopes for residential buildings. Considering only plane rectangular façades, panels, frames and supporting areas, the envelopes design can be considered as a constrained two-dimensional Cutting & Packing problem. Each of the façades in a building has potentially different size, number and position of frames (windows and doors) and different strength (how much additional weight the façade can support and where this additional weight can be located). Clearly, the design of an insulating envelope depends on the geometry and strength of the façade to be renovated. Also, architects preferences, such as panel size and orientation, play an important role as they must be considered in the design process. Then, the problem is also a *product configuration* problem. These facts have led us to apply several techniques from operation research (OR) and artificial intelligence (AI) to tackle this Cutting & Packing problem.

From the set of techniques from OR and AI, the constraint satisfaction framework has proven to be remarkably robust for addressing two-dimensional Cutting & Packing problems (Beldiceanu et al., 2011). Also, it has been proven that constraint satisfaction fits neatly in the constrained nature of layout-synthesis problems and configuration problems. In particular, its declarative view allows a clear knowledge representation of items as constraint decision variables and their relations as constraints. Such declarative knowledge model is independent of the implemented solution and its underlying programming language. Thus, the constraint model, which is the most critical for solving a problem, can be implemented using different techniques such as

mixed integer linear programming, *constraint programming* and optimized with meta-heuristic techniques such as *genetic algorithms*.

The modeling and solving of two-dimensional `Cutting & Packing` problems is considered, mainly for building renovation purposes. The main scientific question addressed in this dissertation is: *How to generate close to optimal packing solutions, w.r.t. minimum number of panels, for the façade-layout synthesis problem while taking into account the stakeholders expectations and industrial limitations?* Among the same lines the dissertation answers the following specific scientific questions:

1. How to model the `Cutting & Packing` problem as a constraint satisfaction problem when the number of configurable items and their size are unknown?
2. How to cut/pack/cover a rectangular surface with unfixed number of rectangular configurable items?

The thesis then builds upon the constraint satisfaction framework to solve the façade-layout synthesis problem under different techniques from AI and OR. It is intended to model and solve the cutting or packing of two-dimensional insulating panels under building renovation setups. The constraint satisfaction model has been mapped into consistent solutions implemented under an object-oriented programming language. The diverse set of solutions here presented deals with the problem when the number of rectangular items (panels) and their size are unknown. As such, the dissertation presents two major contributions:

1. Due to the unusual characteristics of the problem, the *description* and *constraint-based model* of the design problem becomes the first contribution of the thesis. In particular, the industrial renovation description, the envelopes design as a `Cutting & Packing` problem and the support system for architects decision-making as a product configuration software are discussed. Requirements, limitations and design knowledge extracted by stakeholders and architects have been compiled into a constraint satisfaction model that acts as foundation of a decision support system.
2. The *constraint-based algorithmic solutions* that are the gist of the support system, is our second contribution. In particular, the dissertation introduces two manual solutions (i, iv) and three automatic ones (ii, ii, v) for the design problem:

   i First, `InDiE`, a manual interactive solution to guide architects design. It uses the capabilities of a graphical user interface along with validation algorithms, to allow an interactive configuration of each panel on the envelopes. The interaction with the architect is made visually by presenting different colors for well-configured panels and ill-configured ones.

   ii Second, `GaLaS`, an automatic design of panels by packing on the fly with a greedy solution. It follows the greedy approach by making local decisions when assigning size and positions to panels. It solves constraints conflicts locally when allocating each panel. Design knowledge is taken into account when assigning size to panels thus providing solutions close to optimal.

   iii Third, `CaSyE`, an automatic design of panels by considering the geometry of the façade with a cutting solution. It exploits the geometry of the façade (size, windows, doors, supporting areas) to execute vertical and horizontal cuts in order to partition the façade. Once the façade has been partitioned it follows a greedy approach to allocate panels in non-conflictive areas, thus avoiding constraints conflicts from happening. In this solution, the alignment of junctions between panels is taken as aesthetics criterion.

   iv Four, `SkEdE`, a manual solution for architects sketching design. It starts by allowing architects to draw in a draft way sketched panels and creates a constraint model by considering the number, size and position of panels from the predefined sketch. Then, the constraint model is used as input to a constraint solver that applies filtering and search to find compliant solutions.

- v Five, `OpackS`, an automatic design that solves constraint conflicts by means of filtering algorithms. This solution assumes a maximum number of panels potentially in an insulating envelope. Then, it proceeds by placing one panel at the time as does the greedy approach. When a panel is placed, a filtering of inconsistent values is performed for the remaining panels. If the façade surface is covered then the unused panels are discarded. A dedicated heuristic boots the performance of the solution.

From the industrial perspective, the dissertation introduces a product configuration software, specialized version of a decision support system, for the configuration of panels (respectively envelopes) in the renovation project. The system makes a division between key configuration tasks that allows the transparent manipulation of the architects preferences. Also, it includes some recommendation capabilities for some features in the configuration of panels.

Each solution presents an analysis with respect to the underlying allocation process. Conversely, benefits for each solution are discussed and clearly exemplified.

The backbone of the dissertation is the application of the constraints satisfaction framework to solve the industrial problem of façade-layout synthesis. In consequence, the significant part of the thesis is practical rather than theoretical. The results may be applied to two-dimensional `Cutting & Packing` problems in different industries.

The remaining of the chapter is structured as follows. The next section presents a literature review of the fields the thesis builds on, namely, constraint satisfaction, `Cutting & Packing`, product configuration and layout synthesis. Afterwards, we describe in a more detailed way the contributions of the dissertation, list of publications result of the research and the structure of the document.

⋆ ⋆ ⋆

This research is supported by the French agency ADEME - Agence de l'Environnement et de la Maîtrise de l'Énergie - under the project CRIBA (for its French acronyms of *Construction and Renovation in Industrialized Wood Steel*): A joint effort between academics at École des Mines d'Albi - Carmaux and several French companies. Conception and implementation of the industrialized buildings renovation are supported by this work.

1.2 Background & Related Work

This dissertation unifies work from the following fields:

— Constraint satisfaction: The modelling and some of the solving techniques used on the dissertation are taken from this field (section 1.2.1).

— `Cutting & Packing`: The problem at hand falls into this category. The problem definition as well as solving techniques are taken from this field (section 1.2.2 on the next page).

— Product configuration: The support system introduced in this work, conceived for the envelopes design and to support architects decision-making, is built using the concepts from this field (section 1.2.3 on page 11).

— Layout synthesis: Relevant works, in which some solutions (greedy and cutting) have been inspired, are taken from this field (section 1.2.4 on page 14).

This section presents related work on of these fields. The constraint programming concepts, however, will be deeply explained latter in a dedicated chapter for solutions exploiting the capabilities of a constraint solver (Chapter 4 on page 93).

1.2.1 Constraint Satisfaction Problems

Constraint satisfaction problems (CSPs) and constraint programming (CP) have been identified as key paradigms in the expansion of applied computer science. It is, arguably, the most used framework at the intersection of artificial intelligence and operational research. The combination of the modeling capabilities of CSP and the solving techniques of CP have provided to scholars and practitioners with a powerful framework for addressing combinatorial problems whereas their ties with logic allow to reason on the behavior of different systems. In this section we present a brief background on this constraint satisfaction framework and their use to tackle two-dimensional packing problems.

Constraint satisfaction problems are problems in which their solutions must satisfy a given set of requirements known as constraints. The CSP modeling focuses on problems that may be described in terms of requirements or constraints, among which are found several combinatorial problems and were conceived to allow the end-user to state the logic of the computation rather than its flow. For example, in the context of scheduling, instead stating a set of steps to avoid tasks overlapping, the user states *"for any pair of tasks they must not overlap"*. Thus, CSP is part, and good representative, of the declarative modeling frameworks. This declarative view makes the modeling straightforward in many applications where experts have clearly identified the requirements a solution to their problem has.

To model a problem under CSP it is necessary to state the elements of the problem as variables, their potential instantiation values and, a set of relations (named constraints) over the stated variables (Montanari, 1974). Definition 1 formally describes a CSP.

Definition 1 (Constraint Satisfaction Problem): *A CSP is described in terms of a tuple* $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$*, where*

1. $\mathcal{V}$ *is a set of constraint variables, also referred to as decision variables,*
2. $\mathcal{D}$ *is a collection of potential values associated for each variable in* $\mathcal{V}$*, also known as domains and,*
3. $\mathcal{C}$ *is a set of relations over the variables in* $\mathcal{V}$*, referred to as constraints. A constraint is a relation representing partial information over the variables of the problem.*

A CSP solution is a unique assignment of values in $\mathcal{D}$ to the variables in $\mathcal{V}$, in such a way that all constraints in $\mathcal{C}$ are satisfied (see Barták (1999), Brailsford et al. (1999), Montanari (1974) for further references). Variable domains may take different representation such as integer domains, real domains and boolean domains. The domain representation allows to tackle different kinds of problems. If a CSP is based only on integer domains the model is called a *finite domain constraint*

satisfaction problem and is the most common representation for many problem instances (Schulte et al., 1998, Schulte and Carlsson, 2006).

Now, Definition 1 on the preceding page describes the satisfaction of a problem, meaning that the constraints in $\mathcal{C}$ must be *simply* satisfied. By contrast, real-world applications may involve optimization criteria. In such cases, a satisfaction to $\mathcal{C}$, prerequisite to find optimal solutions, is transformed into a constraint optimization problem (COP). Definition 2 describes a constraint optimization problem (COP).

Definition 2 (Constraint Optimization Problem): *A COP is a CSP $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$ in which a function $\mathcal{F}$ is optimized:*

1. *$\mathcal{F}$ is a cost function, or loss function, if the objective is to minimize a given value.*
2. *$\mathcal{F}$ is an utility function, or reward function, if the objective is to maximize a given value.*

In addition, multi-criteria optimization is possible by considering a Pareto-like optimization approach (Marler and Arora, 2004).

Although the modeling is language independent, the solving of such models is done with respect to an underlying constraint system and its corresponding logic. In other words, a declarative model can be solved using different constraint programming systems as `SWI-Prolog` (Wielemaker et al., 2012), `Mozart-Oz` (Müller, 2001), `ECLiPSe` (Schimpf and Shen, 2010), `IBM CP optimizer` (Laborie, 2009), `Minion` (Miguel, 2006), `Choco` (Prud'homme et al., 2014) and the `C++` library `Gecode` (Gecode Team, 2006), among others.

1.2.2 Cutting & Packing

According to Wäscher et al. (2007), any `Cutting & Packing` problem possesses a common structure that allows identify it as such. The structure relates two aspects. In the first place, there exist two elements taking part on the `Cutting & Packing` problem:

— A set, possibly singleton, of large objects: These objects are the spatial entities in which the packing takes place.

— A set of small items: These are the spatial entities to be packed into the large objects.

On the other hand, as part of the `Cutting & Packing` problems structure, every problem must satisfy:

— no overlapping among small items and

— all small items, or a selection of them, must be completely contained in the large object(s).

These elements are specifically defined in one, two, three or more dimensions. Also, both elements may take regular or irregular shapes: Most studies deal with regular shapes (such as rectangles (Korf, 2003) and circles (George et al., 1995)) although small items with irregular shapes appear often in the industry (Julia A. Bennell, 2001, Hopper and Turton, 2001b). Further, regardless of the shape of the large object and the small items, the packing must satisfy a set of requirements inherit from the application domain. These requirements are typically known as *side constraints*. These problems with particular side constraints appear, among other industries, in:

— Lumber processing (Yanasse et al., 1991), glass and metal cutting (Pantalé et al., 2004, Pan and Rao, 2009), leather cutting (Senel et al., 2015): The material is meant to be cut into different shapes, although typically rectangular, in such a way that the waste is minimum.

— Web-page design (Ben Amor and Valério de Carvalho, 2005, Cintra et al., 2008): It addressed appearance of web-pages by considering the browser window as an area to be filled with different shapes. Colors, adjacency and small items size are among other the packing criteria.

— Microcircuits design (Fan et al., 2005, Ziesemer et al., 2014): The design is made with sizes measured in micrometers and thus is an error prune task. The packing of different logical items is done by taking into account, for instance, the total covered area, flow and production cost.

— Apartment design (Liggett, 2000): Convex shaped small items must be packed following adjacency and areas requirements. Here, the rotation of small items is possible as doors communicating rooms (for instance the principal room with a door communicating a bathroom) must be considered.

A subcategory of `Cutting & Packing` problems is the set of two-dimensional problems. Withing it, a special case of two-dimensional `Cutting & Packing` problems is called the *rectangular* `Cutting & Packing` problem (Huang and Korf, 2009, 2014). In this case, both the small items and the large objects are of rectangular geometry. Further distinctive versions of the rectangular `Cutting & Packing` problem forbids the small items rotation and in some case address only the orthogonal case. By orthogonal we meant that the borders of the rectangles are parallel to the enclosing large object. The orthogonal property is exhibit by the `Cutting & Packing` problem addressed in the dissertation.

A classification of the two-dimensional `Cutting & Packing` problems (independent of the shapes) is useful to understand the novelty of our problem (to be presented in Chapter 2 on page 23). The classification presented in Figure 1.1 is based on the typology introduced in Wäscher et al. (2007). It expresses the properties exhibited by particular instances of two-dimensional `Cutting & Packing` problems found in the literature during the last three decades. These properties are related to the number of large objects (multiple or single), the size of the small items (different or identical) and the fact that their size is variable or fixed.

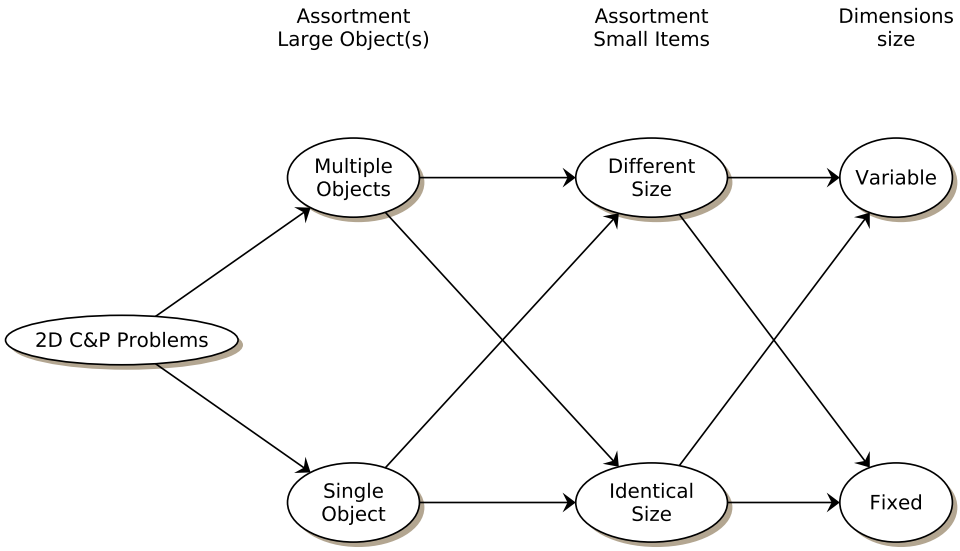


Figure 1.1 – Rectangular `Cutting & Packing` problems classification. A given path in the graph defines a problem category.

The classification is not representative of all `Cutting & Packing` problems but it works as a comparison basis for our work. Nonetheless, well-known classifications, such as the those presented in Dyckhoff (1990) and Wäscher et al. (2007), cannot fully describe the problem addressed in this dissertation; for example, open dimension problems are only represented those with *one* open dimension with fixed number small items, which is not enough to represent the particularities of our problem. Then, as part of the scientific problem description, in Section

2.1.1, we further extend the classification to include the novelties of our Cutting & Packing problem.

Now, as stated before, regarding the difficulty, the set of two-dimensional Cutting & Packing problems are classified as NP-Hard (Korf, 2003). This means that there exists no algorithm to solve them in polynomial time (provided that P ≠ NP) (Imahori et al., 2007). In other words, as the number of small items increases, enumerate all the possibilities for allocating them becomes insolvable in a human life. To understand the difficulty of the problem here considered, we will later study the combinatorics within a minimalistic example (Example 1 on the facing page). It does not show only that Cutting & Packing problem are complex, but also, it provides a valid argument for the use of computer-based solutions for assisting real-world industrial problems.

Review

Along the past three decades, Cutting & Packing problems have been tackled using a broad spectrum of models and techniques. To our knowledge, the more advanced and recent survey of Cutting & Packing literature is presented in Wäscher et al. (2007). It is with respect to that survey that our Cutting & Packing analysis is based on. In our work, however, we focus our attention on constraint-based techniques. Brief discussion about greedy approaches and cutting ones for Cutting & Packing, are presented in their respective sections in Chapter 3 on page 47.

Constraint satisfaction is one of the most used frameworks to address Cutting & Packing problems (Beldiceanu et al., 2011), having among other abstractions, the geometrical constraint GEOST (Beldiceanu et al., 2007), which is a generalization of non-overlapping of small items in k-dimensions ($k \geq 1$). The framework capabilities allow to efficiently model and solve; one-dimensional Cutting & Packing problems such as planning and scheduling problems (Barták et al., 2008, Letort et al., 2014); two-dimensional Cutting & Packing problems such as the two-dimensional bin packing (Beldiceanu et al., 2011); three-dimensional Cutting & Packing problems such as container loading problem (Bortfeldt and Wäscher, 2013); and k-dimensional Cutting & Packing problems (Beldiceanu et al., 2007).

Due to the Cutting & Packing problem object of our study, we are only interested in the orthogonal two-dimensional case. This implies that we do not survey studies on the two-dimensional Cutting & Packing of irregular shapes as it is of marginal interest to our research (but they are described as future research). Also, given the specific rectangular Cutting & Packing problem at hand and the fact that it handles a fixed number of small items, we consider the GEOST constraint not well adapted to our needs.

Common two-dimensional orthogonal Cutting & Packing problems addressed with CP technology are:

— Two-dimensional Bin Packing (2BP) (Lodi et al., 2002): It consists in allocating a set of small items into a set of large objects called *bins*. The objective is to minimize the number of used bins. Entities rotation are possible.

— Two-dimensional Pallet Loading (2PL) (Neliben, 1995): It consists in allocating a set of small items into a large object called *pallet*. Usually, the small items have identical size and may be rotated. The objective is to maximize the number of packed small items.

— Two-dimensional Bin Design (2BD) (Lodi et al., 2002): It consists in finding the minimal size for packing a set of small items. Items rotation and they are typically of different sizes.

— Two-dimensional Item Design (2ID) (Hifi and Ouafi, 1998, Martello et al., 2003): It consists in setting the size for small items, in at least one dimension, in order to fill a given area. This category is called *Open Dimension Problem*. Problems in which small items must be defined in two dimensions are rare in the literature (Wäscher et al., 2007). As we will study in Section 2.1.1 on page 25, the problem addressed in this dissertation is close to the open dimension problem category.

Generally, industrial problems are variations or extensions of the previously listed `Cutting & Packing` problems. These variations include a set of (side) constraints, typically geometrical. Common variations are:

— Vessel Loading Problem (Brown, 2014, Iris and Pacino, 2015): A problem that arises in container terminals and container ships. Side constraints state that, to avoid containers damage, a minimal distance must separate the containers (vessels). As we will study in Chapter 2 on page 23, the problem addressed in this dissertation includes the side constraint of minimal distance.

— Floor Planning Problem (Zawidzki et al., 2011, Izadinia et al., 2014): A problem that arises in design. The packing of rooms, however usually not rectangular, is made with respect to some adjacency, connections, area and more topological constraints. The particular field of these `Cutting & Packing` problems is called layout synthesis or space planning.

— Circuits Design Problem (Fan et al., 2005, Ziesemer et al., 2014): A problem that arises in the devices-manufacturing industries. The design is made with sizes measured in micrometers and thus is an error prone task. The `Cutting & Packing` of different logical items is done taking into account, for instance, the total covered area, flow and production cost.

— Vehicle loading Problem (Leung et al., 2011): A problem that arises in the transportation sector. The `Cutting & Packing` of small items must be done in such a way that the loading and unloading respect a given order. Precedence constraints are then one kind of side constraints for this problem. The problem is often extended in three-dimensions.

The problem of façade-layout synthesis is a particular orthogonal two-dimensional `Cutting & Packing` problem. Its main characteristic, as we will study latter, is that the number of small items and their size are not known at any stage of the `Cutting & Packing` process. This makes the problem an instance of open dimension problems (Wäscher et al., 2007). From the constraint programming viewpoint, this peculiarity represents a challenge: Not having a predefined number of small items becomes a drawback given that the great majority of CP environments implement global constraints and search engines with a fixed set of input variables (OpenRules, Inc., 2013). In fact, performing filtering and searching using an unfixed number of variables, i.e., a dynamically changing problem, is an open research topic in the constraint programming community. For instance, in Barták (2003), the author solves the problem of unknown variables by dynamically adding variables while exploring the search tree. In essence, it introduces a setup in which constraints may be deactivated to be replaced with new activated constraints involving more or less variables. This is a CP implementation of a CSP extension called dynamic CSP (Mittal and Falkenhainer, 1990). Nonetheless, even though the idea seems simple, a good implementation is intricate. Our approach, as we will study latter, exploits the notion of open global constraints and optional variables (van Hoeve and Régin, 2006) and a dedicated search heuristic to address this gap.

Packing & Cutting Example

In order to consistently tie the theory with the `Cutting & Packing` problem, we introduce the following example used as reference in some parts of the document.

Example 1 (Two-dimensional packing problem): *Given a set $\mathcal{R}$ of rectangles described by their width w_i and height h_i and a rectangle area S, find all solutions on how pack the rectangles in $\mathcal{R}$, or a subset of $\mathcal{R}$, in S with no-overlapping.*

The previous statement presents the basic two-dimensional packing problem which is a NP-hard problem (Korf, 2003). Variations of the problem, depending on the targeted industry, attach different constraints and often optimization features. For illustration purposes and as it is compliant with our `Cutting & Packing` problem, we will study the orthogonal case when no rotation is allowed. First, let us instantiate the problem by setting the size of 10 rectangles.

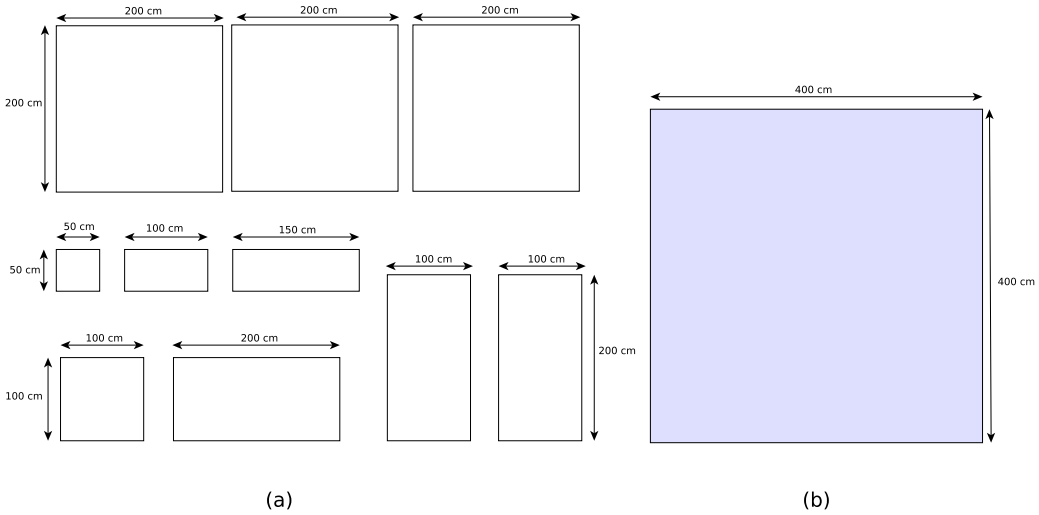


Figure 1.2 – a) Small items and b) large object.

Figure 1.2 shows this instantiation. In addition, let us consider the rectangular area to be synthesized with a size of 4×4 meters.

The process of allocating rectangles over the surface must choose, among the set of rectangles, which one to pack first. Afterwards, it must choose where to pack it. The first combinatorial property emerges in this setup: How many possibility for positioning a given rectangle exists assuming that the discretization is in centimeters? In general, the number of possibilities for allocating the rectangle is in proportion with its size and the size of object. For instance, let us select as the first rectangle the one at the bottom-left whose size is 100×100 centimeters. In this case the rectangle may be placed along the vertical axis in 30 different locations. Conversely, along the horizontal axis, the rectangle can be placed in 30 possible locations. In consequence, to allocate the rectangle $30 \times 30 = 900$ locations are possible. Now, the problem statement in Example 1 on the previous page does not discard solutions with holes. Thus, some of the rectangles may be left out of a packing solution, increasing even more the combinatorics within the problem. Figure 1.3 illustrates three valid solutions for the problem.

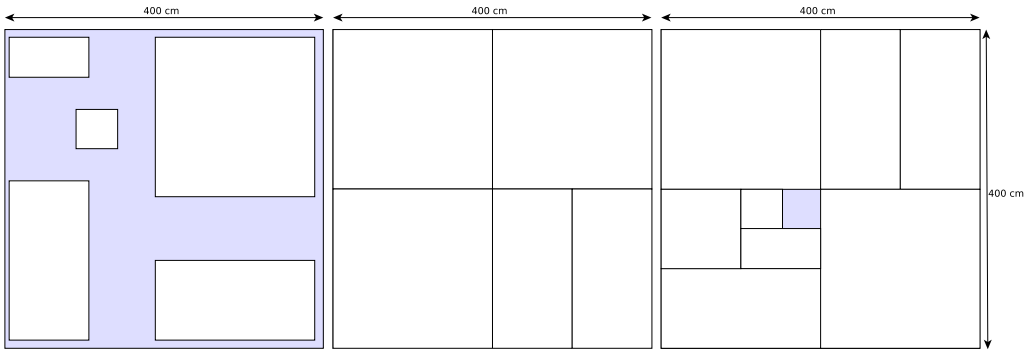


Figure 1.3 – Three possible packing solutions for Example 1 on the preceding page.

Consider now the three conceptually different solutions in Figure 1.4 on the next page, that are in fact the same practical solutions. In fact, although having the same size, rectangles 1, 2 and 3 are actually different small items. Thus, swapping locations between two rectangles with the same size is valid and will lead to a symmetrical solution. These solutions are often

discarded by applying symmetry breaking constraints (Ågren et al., 2009), typically applying certain ordering (Walsh, 2009). The problem undertaken in this thesis deals with symmetry as small items (rectangular panels) are indistinguishable from each other.

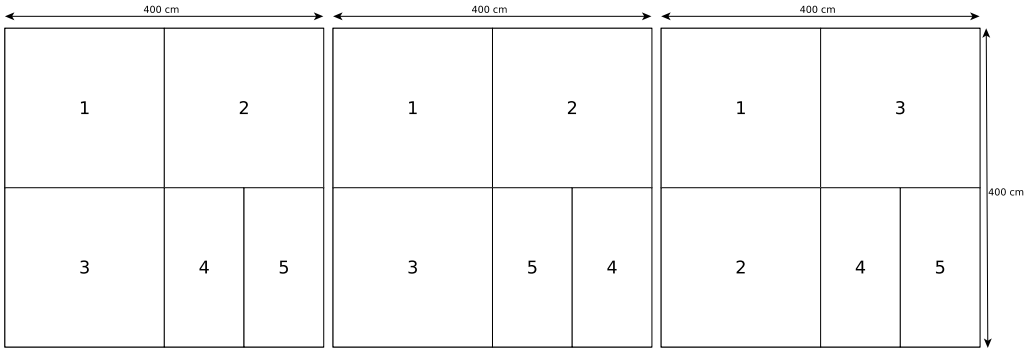


Figure 1.4 – Three symmetrical solutions for Example 1 on page 9.

To conclude with Example 1 on page 9, let us add one degree of difficulty. Suppose that the 10 rectangles previously stated do not have fixed sizes. Instead, assume that the rectangles have a given lower bound and upper bound for the possible sizes they can take for both width and height. This fact increases the possibilities of rectangle allocation with the size of the domain, i.e., upper bound less lower bound. This particular case, of unfixed size, is a peculiarity of our packing problem (to be described in Chapter 2 on page 23). On top of it, let us further add another degree of difficulty shared by our `Cutting & Packing` problem: The number of small items to allocate is *unknown*. Simply put, the number of possible packing solutions increases with these new incognitos.

Tribunes

Tribunes for literature on `Cutting & Packing` is as wide as the engineering field due to the diverse set of `Cutting & Packing` problems that arise in different scenarios. From these tribunes, we highlight the ones dedicated to AI and OR. Notably, international conferences *International Conference on Principles and Practice of Constraint Programming*, *International Conference on Integration of Artificial Intelligence and Operational Research Techniques in Constraint Programming*, *INFORMS Conferences*, *International Joint Conference on Artificial Intelligence*, *European Conference on Artificial Intelligence*, *The AAAI Conference on Artificial Intelligence* and *The European Conference on Operational Research* are the main meetings for `Cutting & Packing` academics and practitioners. In addition, international journals as *Constraints*, *European Journal of Operational Research*, *Operations Research - INFORMS* and *Computers and Operations Research* among other, present the state-of-the-art developments of theory and practice related to `Cutting & Packing` problems.

Link to thesis

The problem addressed in this work is an instance of orthogonal two-dimensional `Cutting & Packing` problems. The set of peculiarities on the problem, to be discussed in next chapter, makes the problem novel. Further, we will propose an extension of the classification for `Cutting & Packing` problems; a classification that allow us to represent this unstudied case.

1.2.3 Product Configuration

With the advent of Internet and the World Wide Web, individual user's preferences and needs have pullulated today's commerce. Current trends show that customers demand products

to be adapted to their requirements while keeping, or improving, traditional delivery times and costs (Da Silveira et al., 2001). This challenge has led academics and industry representatives to work on knowledge models with expressive power to capture complex configurable products and with the ability to search for solutions (Sabin and Weigel, 1998), to optimize (Nica et al., 2014), to debug and to diagnose (Felfernig et al., 2014b) while taking into account needs of particular customers. The body of knowledge models for product configuration is referred to as *knowledge-based configuration*.

Product configuration refers to the process of building a personalized target product using predefined components, respecting requirements from customers and following some rules that shape a correct configuration (Soininen et al., 1998, Yang et al., 2008). This process has been increasingly supported by intelligent systems given the complexity and size of relations within a single product. The possible number of outputs for a configuration is in proportion to the number of components and relations within the product, and is inversely proportional to the number of rules that restrict combinations. Actually, solving a configuration problem is complex in such an extend that specialized techniques from *Artificial Intelligence* (AI) and *Operation Research* (OR) have been used, often extended, to handle these ubiquitous industrial problems (Felfernig et al., 2014a). One of these techniques is constraint satisfaction problems.

Traditionally, a configuration problem may be described as a tuple $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$. Semantics in $\mathcal{C}$ is subject to the configuration problem as customers requirements are taken into account.

Definition 3 (Configuration Problem): *A configuration problem may be described in terms of a tuple* $\langle \mathcal{V}, \mathcal{D}, \mathcal{C}, \mathcal{P} \rangle$*, where*

1. $\mathcal{V}$ *is a set of variables describing components,*
2. $\mathcal{D}$ *is a collection of component types associated to each variable in* $\mathcal{V}$*,*
3. $\mathcal{C} = \mathcal{T} \cup \mathcal{R}$ *where*
   - $\mathcal{T}$ *are the compatibility constraints between components that describes a correct configuration (from the technical viewpoint) and,*
   - $\mathcal{R}$ *are the constraints describing the customers preferences over the configuration.*
4. $\mathcal{P}$ *is a set of parameters customizing some components of* $\mathcal{V}$*, defining types of* $\mathcal{D}$ *or personalizing some constraints of* $\mathcal{C}$*.*

This traditional configuration definition is the one used along the document. But, different configuration problems may be described using particular extensions of this definition. For instance, determining a hierarchical structure of the problem (Kokeny, 1994), using activation constraints (Mittal and Falkenhainer, 1990), defining a configuration problem as a set of subproblems with composite CSP (Sabin and Freuder, 1996), and more. Nevertheless, for the problem at hand, only the traditional view is considered as it allows us to completely express the problem.

Let us now create the configuration for the `Cutting & Packing` problem presented in Example 1 on page 9. Following our example, we may construct a model as

$\mathcal{V}$ For every rectangle i the position (x, y) of top-left corner over the rectangular area S.

$\mathcal{D}$ Domain for x is $[0, S_w]$ and for y is $[0, S_h]$, where S_w and S_h are, respectively, the width and height of the rectangular area S.

$\mathcal{C}$ What are the constraints and requirements?

- $\mathcal{T}$ Panels overlapping is not allowed.
- $\mathcal{R}$ The customer rather prefers square panels.

Review

The development of robust techniques for product configuration problems influenced the use of Product Configuration Software (PCS) in different industry domains. A PCS is a specialized version of decision support system that focuses on configuring products and services. These

systems support different stages of the production process, such as design (Myung and Han, 2001), sales (Haag, 1998) and manufacturing (Aldanondo et al., 1999). As such, at companies eyes, product configuration software is a source of profit as it improves decision-making and manufacturing. Industry decision support systems for product configuration include but are not limited to railways interlocking systems (Falkner and Schreiner, 2014), equipment and services on cement and minerals industries (Orsvärn and Bennick, 2014), sales and marketing (Höfling, 2014), mobile networks (Nica et al., 2014) and technical documents (Rabiser et al., 2014). Nonetheless, the industry continuously provides new challenges on this domain, so the field of configuration is far from closed. This dissertation involves the design of insulating envelopes that may be seen as the configuration of each of the panels on the envelope. Ergo, the prototype support system result from the thesis is a PCS for insulating envelopes.

Now, PCSs have their foundations in different techniques from applied computer science, such as integer linear programming (Frutos et al., 2004, Feinerer, 2013) and mixed integer linear programming (Hutter et al., 2010), answer set programming (Schenner et al., 2013) and constraint satisfaction problems (CSPs) (Felfernig et al., 2011). Although each of these techniques has provided certain benefits on regard to the particular configuration problem, it is the constraint satisfaction model the one originating the research on configuration problems with AI in the first International workshop of Configuration collocated in the *AAAI'96 Fall Symposium*. This is one of the most used techniques for solving configuration problems. In fact, CSPs have been used in product configuration problems in such an extend that there is an entire chapter to product configuration in the *Handbook of Constraint Programming* (Junker, 2006). The author, Ulrich Junker, presents a complete overview of the application of CSP on configuration problems. The constrained nature of packing problems makes an additional argument for the adoption of CSP to develop the PCS for configuring insulation envelopes.

Now, despite the fact that CSP modeling is actually very simple and intuitive, product configuration problems involve difficulties that sometimes exceeds the capabilities of model (e.g. dynamic view and interactivity). Thus, it was argued that CSPs paradigm as original conceived was not well suited for addressing most of the configuration problems, due to the lack of mechanisms to handle variables that in some cases are relevant and in other cases not. Mittal et al. in their seminal work, enhanced CSP with a dynamic view (Mittal and Falkenhainer, 1990): Constraint variables may be either activated or deactivated. In the first case variables take part of the problem and hence in the solving process. This dynamic view is important to our work as the number of panels for configuring an envelope is not known and thus new instances of panels may be demanded and generated dynamically.

The work started a cascade effect of research on product configuration and CSP. For instance, in Sabin and Freuder (1996), the authors have developed Composite CSP, incorporating into CSP the possibility to have model complex configuration relations such as *whole-part*, *is-a* and *part-of*. Thus, if a sub-problem variable is activated, all the problem is dynamically changed to handle all information in the problem and sub-problem. Hierarchical relations in the renovation process, such as building part-of block, are easily modeled under this CSP view. Among the same lines of work is Gelle and Weigel (1996) in which CSP is enhanced with the manipulation of continuous variables in order to address a wide range of real-life configuration problems. Critical to model the problem at hand as it mixes different variable representation such as discrete variables, continuous variables, boolean variables and even symbolic ones. Studies for treating constraints as formulae and computable procedures using a constraint modeling tool (Xie et al., 2005) and to handle advanced structural relationships (Yang et al., 2012), are also found in the literature.

Given the efforts commented above, constraint satisfaction is now a mature framework to address combinatorial problems including configuration problems. More extensions and applications of CSP to product configuration have been and are still proposed constantly. However, regardless the considerable body of literature on configuration and constraint satisfaction, there is no generic solution capable of addressing all configuration problems that rise in the industry. It is necessary, then, to develop a dedicated computer-based solution for the configuration problem of insulating envelopes.

Tribunes

Due to the broad spectrum of techniques used to solve configuration problems, tribunes for this research include the journals *IEEE Intelligent Systems*, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *AI Communications*, *AI Magazine* and the *International Journal of Mass Customization*, and successful events like *IEEE International Conference on Industrial Engineering and Engineering Management*, *European Conference on Artificial Intelligence*, *ACM Conference on Recommender Systems*, *International Conference on Mass Customization and Personalization in Central Europe* and one of the oldest, most constant and important meetings, the *International Configuration Workshop*.

Link to thesis

The reason for which product configuration is part of the underlying theories of this dissertation is two-fold. First, the manufacturing of envelopes is a configuration task. A given envelope must be designed taking into account geometrical and structural aspects of the façades as well as the preferences of architects. The design of envelopes is a configuration problem where the first actor to impose its requirements is the façade itself. In addition, architects' preferences are taken into account as much as possible (soft constraints). For example, in the support system, a personalization feature is provided; the preferred orientation of panels in an envelope configuration solution. At the end of the design process, the specific envelope design for a given façade is given to the panels provider so the manufacturing of these products can start.

On the other hand, the support system whose core is the result of this dissertation, is meant to be used in the buildings systematic thermal renovation. Panels are products that need to be designed before their manufacturing, properly manufactured and then delivered to the working site. The support system is then a PCS.

1.2.4 Layout Synthesis

Layout synthesis, also known as space planning, involves the placement of objects into a given space following certain criteria. Although sometimes these objects may refer to abstract entities (such as departments or services inside company), the majority of cases the placement means real object. As far as we know, there is no formal definition of layout problems but rather instantiations under well-defined theories (such as CSP). Nonetheless, according to the literature (Liggett, 2000), any layout or placement problems must have well defined:

1. Space S: Entity in which the placement will take place. This space may take different shapes and may be defined in $n = 1$ or more dimensions (most practical works however has $n \leq 3$). In addition, the space may be seen as an area or a set of discrete objects.
2. A set of indivisible objects Obs to be placed in the space S. These objects, in most cases, have well defined height and width although they may be configured (set in run-time) according to the problem specification.
3. A set of side-constraints that describe a correct layout-plan. Instance of such constraints are:
   - — Adjacency constraints between all or some objects.
   - — Size constraints to objects.
   - — Distance constraint between objects (may include flow analysis).
   - — Overlapping constraints between objects or portion of the space S.
   - — ...

These problems may be considered as configuration problems (Drira et al., 2007) and thus may take advantage of the techniques and finding in the knowledge-based configuration field (see previous section). As a consequence, no other notion is necessary for understanding layout problems. Further, these problems may be modelled using CSP and may be solved using constraint technology. The following review shows the methods used in the field.

Review

As pointed out researchers in Liggett (2000) layout synthesis, commonly referred to a space planning, "...is concerned with the allocation of activities to space such that a set of criteria (for example, area requirements) are met and/or some objective optimized...". Here "activities" may refer to two concepts. On the one hand it may refer to spatial entities such as panels, rooms or machines. On the other hand, it may refer to more abstract concepts such as departments inside an office or services inside an hospital facility. The allocation of entities in space is classified as a combinatorial problem (Baykan and Fox, 1991, Dutta and Sarthak, 2011).

The leading research topic in the area is that of synthesis or planning algorithms. From the body of these algorithms several approaches may be distinguished, such as direct solving with Linear Programming and Constraint Programming or user-designed with an interactive layout design. Among the most used approaches are the constructive and the iterative one (Liggett, 2000). The former follows an incremental fashion, i.e., each activity (e.g. room, office, panel) is put in the space following a given criteria and, if all conditions are satisfied, the algorithm proceeds by placing the next activity until the construction of the entire layout has been done. The iterative improvement approach, on the other hand, is based on the improvement of an already configured space that, although fulfill some requirements, does not satisfy all criteria. Of course, some studies go further by mixing the constructive and iterative approaches.

Despite the great body of literature on the field, work on layout synthesis for façades using any kind of spatial entity is scarce. Actually, works relating to façades and their layout focus over predefined layout solutions and not on the actual composition or structure of the façade. For instance, in Teboul et al. (2010) authors use shape grammars and supervised classification to generate a segmentation of the façades in order to model buildings and understand patterns as part of urban planning. Among the same line of research, in Wu et al. (2014) is used an approach based on inverse modeling to build knowledge on how façade layouts are generated and how elements are related. Also, researches have used the available partial structure of some buildings to determine the total layout of façades and thus enable image capturing software and automatic layout design to capture urban reconstruction (Fan et al., 2014). None of these works intent to allocate entities over the façade and thus we consider them interesting from the design point of view but not suitable for the problem at hand.

Concerning the actual allocation of activities over space, the majority of studies focus on layout plans for buildings floors, rooms inside apartments, machines inside factories and departments inside offices. In fact, such kind of research has a profound impact on energy consumption over buildings, production efficiency and management (Rodrigues et al., 2014, Ansary and Shalaby, 2014). These works, however, present a setup where the number of entities to allocate is known in advance making them no suitable for façade-layout synthesis. Nonetheless, these works present interesting algorithms and optimization features that have inspired our work although not all of them are constraint-based.

The simplest version of layout synthesis problem is called the *Quadratic Assignment Problem*. The problem focuses on assigning n entities over n possible locations, where each pair of entities has a distance cost and associated flow. The goal is to minimize the total distance times flow on the layout plan. Application of this problem can be found in hospital departments locations (Elshafei, 1977, Helber et al., 2015), numerical analysis (Brusco and Stahl, 2000), cellular manufacturing systems (Solimanpur et al., 2004), among others.

Evolutionary computation, from which the most representative is the meta-heuristic genetic algorithm, may be used to tackle layout problems as they avoid the problem of local optimum by allowing bad chromosomes to continue to reproduce and thus helping explore the entire search space. Then, this is a particular realization of the iterative approach. Such technique is found in Ansary and Shalaby (2014) when optimizing the position of house inside a finite space prioritizing aspects such as incidence light and indoor privacy. A similar approach is used in Krishnan et al. (2014) where a swarm optimization technique is applied to the layout planning of flexible manufacturing systems. The main optimization feature is to minimize the travel distance between key entities on the layout. Another study using an evolutionary approach is Garcá-

Hernández et al. (2015) where authors capture expertise knowledge into the genetic algorithms thus helping to considerably reduce the search space.

Some studies exploit interactivity as a way to include designer's preferences in the layout generation process. For instance, in Michalek and Papalambros (2002) authors use an object-oriented approach to allow the designer modify elements of the layout meanwhile the underlying mathematical model adapts the solution with respect to optimization features. Similarly, authors in Goetschalckx (1992) explore the interactive approach with adjacency graphs. In essence, the heuristic designs a set of layouts and the designer selects the more appropriated one. In a second stage an heuristic improves the selected layout based on integer programming optimization. Another study under mathematical programming is found in Izadinia et al. (2014) and addresses the floor layout plan in an uncertain environment where flow among entities may vary. One of our proposed solutions, in Section 3.1 on page 48, allows interactive design by providing to the architect a visual feedback when drawing panels thus guiding them through the design process.

Additionally, well-known search strategies have also been applied to facility layout problems. It is the case of Liang and Chao (2008) where tabu search is used to optimize layout plans and Solimanpur and Jafari (2008) where the branch and bound technique is used to explore the solution space for optimizing two-dimensional entities over a two-dimensional space by using a mixed-integer nonlinear mathematical programming model. Although the authors acknowledged that the solution is not well suited for large-size problems, it is efficient and provides optimal solutions on small and medium scale problems. As well as most works, these models and solutions are conceived for a fixed number of entities.

Regarding layout synthesis under a constraint satisfaction perspective, we have selected a set of studies that we consider interesting and related to our work. Indeed, several studies on layout synthesis under the scope of CSP are found. For example, a theoretical framework for addressing layout synthesis problems using an interactive based approach (Shikder et al., 2010), another that uses topological constraints (e.g. adjacency) and ordering heuristics to analyze the search space in a room apartments setup (Medjdoub and Yannou, 2001), and even the combination of CSP technology with shape grammars and data structures to determine the interior layout of buildings (Yue et al., 2008).

A common scenario in the field is the planning of rooms within a residential apartment and departments within offices. In these works, a grid/matrix space division and heuristics to select optimal solutions are common. In (Zawidzki et al., 2011) an apartment space is divided into a matrix in which rooms are placed following area and adjacency constraints. Authors used backtrack search to find solutions and then an heuristic to rank all the solutions found. In Charman (1993) is introduced Geometric CSP, which is an extension of CSP for handling geometric domains, and presents some consistency methods as well as backtracking algorithms. Another work on layout synthesis and CSPs is found in Baykan and Fox (1997). The authors have developed a constraint-based framework to tackle layout synthesis problems in a 2D reference plane. Their work, called Disjunctive CSP, uses disjoints of atomic constraints to represent spatial relations among entities. Solutions are found by applying search to each one of the disjuncts in the constraint model. Here, the performance of the solving process depends largely on the underlying solver. In Liggett (2000) the reader can found a comprehensive survey of layout synthesis models and applications.

Our problem includes five particularities that have been never consider simultaneously and thus, no support system nor computer-aided design software is well suited for addressing the problem. Nevertheless, the field of space planning counts with several systems implemented using different approaches, here we name a few of them. Let us start with Shikder et al. (2010) where is introduced a prototype for the interactive layout synthesis of rooms inside facilities which includes design information and an iterative design process. Interactive behavior is strongly welcome in design and thus one manual solution presented in this dissertation implements interactive behavior (Section 3.1 on page 48). In Rodrigues et al. (2014) a tool for the design of house interior layouts that applies a hybrid stochastic-evolutionary technique to allocate rooms is presented. This tool runs simulations of energetic performance of the house for each floor

plan thus assisting architects to make decisions. Our dedicated support system do not performs simulations but supports architect decision-making by computing the cost and length of junctions of each generated insulating envelope. In Baykan and Fox (1992) is introduced WRIGHT, a constraint-based layout generation system that exploits disjunctions of constraints to manage the possibilities on positioning two-dimensional objects in a two-dimensional space. Another system, LOOS (Flemming, 1990), is able to configure spaces using rectangles that cannot be overlapped but that may have holes. It uses test rules applied by steps to the rectangles in order to reach a good plan based on its orientation and relation with other rectangles. The same authors have developed SEED (Flemming and Woodbury, 1995): A system based on LOOS used for early stages on design. A comparison between WRIGHT and LOOS can be found in Flemming et al. (1992). The system HeGel (Akin et al., 1992) (for Heuristic Generation of Layouts) is yet another space planning tool that simulates human design based on experimental cases. In (Medjdoub and Yannou, 2000) is presented the system ARCHiPLAN which integrates geometrical and topological constraints to apartment layout planning.

Again, the particularities of our façade-layout packing problem would make necessary to deeply extend these tools, or any other packing related software, which is potentially more time consuming. More software such as computer-aided design systems and support systems may be found in the literature, a brief review of different tools for site layout planning can be found in Kumar and Bansal (2014).

Tribunes

The field of layout synthesis is closely related to architecture and design but, their underlying theories are taken from computer science and applied mathematics. Thus, tribunes for this research include the journals *Environment and Planning B: Planning and Design*, *Journal of Architectural Engineering*, *Artificial Intelligence in Engineering* and *Automation in Construction* and successful events like *Conference on design computing and cognition*, *Conference on Computer-Aided Architectural Design Research in Asia* and *Conference on Education and Research in Computer Aided Architectural Design in Europe*, among others.

Link to thesis

Layout synthesis is part of the dissertation mainly because it provides many works on constructive algorithms. The design of an insulating envelope is seeing here as a novel layout problem where the space plan is vertical and the number of entities to allocate is unknown. Further, one of the allocation processes presented in the dissertation, the greedy solution GaLaS, follows the very same constructive approach of these algorithms whereas one manual solution InDiE exploits interactive behavior to provide compliant layout-plan solutions.

1.3 Contribution & Structure

This section summarizes the contribution of the dissertation, its communications results and the structure of the remaining of the document.

Clarifying note #1

From a `Cutting & Packing` perspective, a solution is called a *cutting solution* or *packing solution*. From a layout synthesis perspective, a solution is called a *layout-plan solution*. From the industrial scenario perspective, a solution is called an insulating *envelope*. Thus, the following equivalence holds:

$$\textit{cutting/packing solution} \equiv \textit{layout plan} \equiv \textit{insulating envelope}.$$

Along the document, however, we will use only the terms insulating envelope or just envelope.

1.3.1 Contribution

As stated previously, the main scientific question addressed in this dissertation is *How to generate close to optimal packing solutions, w.r.t. minimum number of panels, for the façade-layout synthesis problem while taking into account the stakeholders expectations and industrial limitations?* As such, the central contributions of this dissertation are summarized as follows.

1. Chapter 2 on page 23:
   - ☐ This dissertation introduces the problem of Façade-layout Synthesis as a novel non-studied orthogonal two-dimensional `Cutting & Packing` problem and layout synthesis problem (Section 2.1 on page 23).
   - ☐ This dissertation develops a solid constraint-based model for the `Cutting & Packing` problem of façade-layout synthesis (Section 2.2 on page 37).
2. Chapter 3 on page 47:
   - ☐ This dissertation introduces an interactive manual design for insulating envelopes, named `InDiE`. Validation algorithms are implemented for each constraint in the constraint-model and a visual communication guides the architects through the design process (Section 3.1 on page 48).
   - ☐ This dissertation presents a dedicated on-line greedy solution for `Cutting & Packing` problem, named `GaLaS`. The greedy solution is part of the constructive algorithms family and uses stakeholders knowledge to generate layout plans close to optimal (Section 3.2 on page 56).
   - ☐ This dissertation presents a dedicated cutting algorithm for the packing problem, named `CaSyE`. Its underlying mechanism is that of guillotine vertical and horizontal cuts. No optimal solutions are generated with this approach but it has other benefits such as aesthetics of insulating envelopes (Section 3.3 on page 75).
3. Chapter 4 on page 93:
   - ☐ This dissertation presents a fully declarative model to assist architects sketching, named `SkEdE`. A constraint model is created from a hand-made sketch and is executed by a constraint programming environment. Its generates the more compliant and aesthetics envelopes from an architectural point of view (Section 4.2 on page 97).
   - ☐ This dissertation presents constraint programming filtering algorithms and a dedicated search heuristic for the problem under the model of *optional decision variables*, named `OpackS`. The heuristic exploits stakeholders knowledge to arrive rapidly to solutions on façades where most predefined search heuristics fails (Section 4.3 on page 106).
4. Chapter 5 on page 127:

- ☐ This dissertation introduced a novel decision support system for the manual and automatic design of façades insulating envelopes. Configuration aspects of the renovation project, such as the inclusion of architects preferences and recommendation features, and division of configuration tasks are included in the conception and development of the system (Chapter 5 on page 127).

1.3.2 Scientific Dissemination

The fundamental aspects of this thesis have been previously reported in the following publications whose first author is Andrés Felipe Barco.

— Building Thermal Renovation Overview. In *22nd International Symposium on Methodologies for Intelligent Systems (ISMIS'15)*, pages 379-385.
Joint work with Élise Vareilles, Paul Gaborit and Michel Aldanondo.

Contributions associated with Chapter 2 on page 23.

— A Recursive Algorithm for Building Renovation in Smart Cities. In *21st International Symposium on Methodologies for Intelligent Systems (ISMIS'14)*, pages 144-153.
Joint work with Élise Vareilles, Paul Gaborit and Michel Aldanondo.

Contributions associated with Chapter 3 on page 47.

— External Buildings Retrofit: Employing Guillotine Cuts for Aesthetic Envelopes. In *IEEE International Conference on Industrial Engineering and Engineering Managements (IEEM'16)*
Joint work with Michel Aldanondo, Paul Gaborit and Élise Vareilles.

Contributions associated with Chapter 3 on page 47.

— Open Packing for Façade-Layout Synthesis Under a General Purpose Solver. In *21st International Conference in Principles and Practice of Constraint Programming (CP'2015)*, pages 508-523.
Joint work with Jean-Guillaume Fages, Élise Vareilles, Paul Gaborit and Michel Aldanondo.

Contributions associated with Chapter 4 on page 93.

— Industrialized building renovation: Manufacturing through a constraint-based on-line support system. In *IEEE International Conference on Industrial Engineering and Engineering Managements (IEEM'15)*, pages 947-951.
Joint work with Élise Vareilles, Paul Gaborit and Michel Aldanondo.

Contributions associated with Chapter 5 on page 127.

— Building renovation adopts mass cutsomization. Configuring insulating envelops. In *Journal of Intelligent Information Systems (JIIS'16)*
Joint work with Élise Vareilles, Paul Gaborit and Michel Aldanondo.

Contributions associated with Chapter 5 on page 127.

Three additional conference publications that have been part of the research, but whose content is not included in this dissertation, are:

— Constraint-based Decision Support System: Designing and Manufacturing Building Facades. In *7th Joint Conference on Mechanical Design Engineering and Advanced Manufacturing (JCM'14)*, pages 276-282.
Andrés F. Barco, Élise Vareilles, Michel Aldanondo, Paul Gaborit and Marie Falcon.

— Towards a BIM Approach for a High Performance Renovation of Apartment Buildings. In *11th IFIP WG 5.1 International Conference on Product Lifecycle Management (PLM'14)*, pages 21-30.
Michel Aldanondo, Andrés F. Barco, Élise Vareilles, Marie Falcon, Paul Gaborit and Linda Zhang.

— Layout Synthesis for Symmetrical Facades: Constraint-Based Support for Architects Decision-Making. In *11th International Conference on Artificial Intelligence Applications and Innovations (AIAI'15)*, pages 293-306.
Andrés F. Barco, Élise Vareilles, Michel Aldanondo and Paul Gaborit.

1.3.3 Road Map

In what follows we describe the structure of this dissertation. Figure 1.5 shows the dependency among chapters.

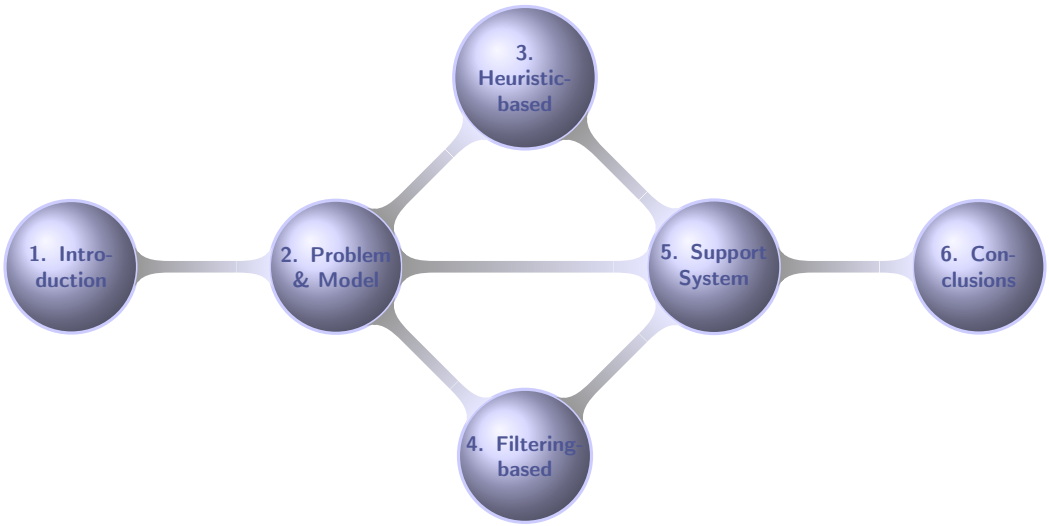


Figure 1.5 – Organization of the dissertation.

Chapter 2 on page 23 [Facade-Layout Synthesis Problem] In this Chapter, the specification of the problem at hand from both scientific and industrial viewpoints are presented.

— The scientific problem of the dissertation, orthogonal two-dimensional `Cutting & Packing` problems, is described in this section.

— The industrial problem of designing insulating envelopes is described in detail. The knowledge provided by stakeholders is synthesized here in specific limitations for the packing problem.

— The model gives a formal declarative view of the restrictions and limitations of the packing problem. To do this, constraint decision variables, their domains and a set of constraints are stated.

— The three façade examples that are presented are used along the document to evaluate our algorithmic solutions. Two of them are models of real existing façades from the pilot renovation site whereas the last one is a realistic instance.

Chapter 3 on page 47 [Interactive & Heuristic-Based Design] In this chapter, a manual interactive solution and two automated heuristic-based ones are introduced.

— A first manual solution `InDiE` aims to guide the user on the envelope design. It does so by testing constraints each time the user draws a panel thus providing immediate feedback and interactive design.

— A second solution `GaLaS` is based on a greedy approach, making local decision for packing panels. Local decisions respect stakeholders knowledge to assign size for panels thus providing solutions close to optimal.

— Finally is presented a solution that uses a guillotine approach `CaSyE`, performing vertical and horizontal cuts while respecting constraints. Albeit the algorithm is restricted for generating at most two solutions, it is smarter than the greedy in the sense that the geometrical structure of the façade is taken into account when allocating panels. Further, the algorithm throws aesthetics of envelopes with respect to their symmetry.

Chapter 4 on page 93 [Filtering-Based Design] Filtering-based solutions that exploit the capabilities of a constraint solver are discussed in the chapter.

— The first solution is a declarative model that throws a packing solution using an user-defined sketch `SkEdE`. In this setup, the user may draw panels with constraint conflicts (a hand-made sketch). A consistent solution is generated by performing filtering and search over the defined user sketch.

— As a second approach, we present an automated filtering solution for the problem `OpackS`. It uses the notion of optional decision variables for performing filtering. It includes a simple but efficient search heuristic and its comparison against traditional CP search heuristics.

Chapter 5 on page 127 [Decision Support System] The solutions presented in the dissertation are included in an on-line support system introduced in this chapter. The on-line system is based on a service-oriented architecture that allows the configuration tasks within the renovation process.

— An overview of the system capabilities and its internal design are discussed.

— The specific configuration tasks within the renovation process are described here.

Chapter 6 on page 139 [Concluding Remarks] The last chapter presents a comparison of results and summary of the research presented in this dissertation. In addition, some perspectives for future work are presented in this chapter.

For me, it is far better to grasp the Universe as it really is than to persist in delusion, however satisfying and reassuring.

The Demon-Haunted World: Science as a Candle in the Dark
Carl Sagan, 1995

2 Facade-Layout Synthesis Problem

Contents

The façade-layout synthesis problem is classified among the family of `Cutting & Packing` problems, more concisely, among the orthogonal two-dimensional `Cutting & Packing` problems. It arises in the context of industrialized building renovation and hence some particularities, modelled as constraints, are inherit from this domain. This chapter presents the problem and its modelling with constraint satisfaction problems. A brief report of this chapter has been presented in Barco et al. (2015a).

2.1 Problems

This section presents the scientific issues, in regard to the `Cutting & Packing` problem, addressed in the dissertation. Likewise, details of the industrial problem are introduced along with assumptions and optimality notions used in the algorithmic solutions. The section systematically breaks down the problem, industrial limitations and the packing requirements from a design process viewpoint.

2.1.1 Scientific Problem

As commented in Section 1.1 on page 1, the problem subject of our study has five particularities. One of these is key for defining the `Cutting & Packing` problem whereas the others

only imposes specific side constraints. Then, to understand the scientific issues addressed in the dissertation, we will briefly analyze four of the particularities and, in a more deeply analysis, the main characteristic that differentiates our problem from the literature works. Here, and ignoring the industrial details, it is worth noticing that façades, panels, windows, doors and supporting areas have all rectangular shapes and their edges are parallel to the façade edges (orthogonal problem). Therefore, the problem belongs to the family of orthogonal two-dimensional `Cutting & Packing` problems.

Mandatory overlapping (of frames). The first side constraint of the problem is that specific rectangular areas (existing windows and door) over the large object (façade) must be completely overlapped by the small items (panels). Any of these areas must be covered with only one specific small item, meaning that partially overlapping any of these areas is not allowed. As far as we know, this property is inexistent in previous works on `Cutting & Packing` or layout-synthesis literature.

Panels' installation (over supporting areas). The second side constraint involves the installation of the small items (panels) over the large object (façade surface). In fact, due to the added weight of the set of small items (envelope composed of panels) and given the vertical orientation of the large object (façades), the small items can only be attached in specific rectangular areas that will uniformly distribute their weight thus preventing the small items to fall and the large object to collapse. To the best of our knowledge, the second characteristic of the problem, as well as the previous one, is inexistent in the literature of `Cutting & Packing` and layout synthesis.

No overlapping, no holes. Likewise most `Cutting & Packing` problems, small items (panels) overlapping is forbidden. In addition, given the renovation context, the existence of holes in a solution is infeasible (holes are impractical for the thermal insulation). In consequence, the small items in a solution (envelope) must be adjacent to each other. This third side constraint appears often in `Cutting & Packing` and layout synthesis problems typically as the minimization of uncover space (respectively maximization of packed small items).

Desired thermal performance and cost equals minimum number of panels. As we will study in next section, the ranking of solutions (insulating envelopes) is made with the number of small items (panels) on it. This fact introduces a goal for the algorithmic solutions developed along the thesis. As expected, many `Cutting & Packing` and layout-synthesis problems involve the optimization of a given feature (sometimes more than one value, multi-criteria optimization). Within the set of envelopes with the minimum number of small items, the best solutions, from the thermal perspective, are those in which the length of junctions is minimal. These facts will be explained in detail in Section 2.1.2 on page 36.

Now, we have found that last particularity of our problem is unique. This means that, to the best of our knowledge, it has never been studied in the `Cutting & Packing` and layout-synthesis literature.

Number and size incognito. The more distinguishable characteristic of our problem is that the number of small items and their size are not known a priori. This property makes most of the work on `Cutting & Packing` and layout synthesis not well suited for tackling it. Further, in layout-synthesis literature it is rather logic that entities to create layout plans are well-defined before the design starts due to the specific layout requirements. For instance, when designing a residential floor plan, the customer demands the architect a certain number of rooms (e.g. one for each family member) and a given number of bathrooms, kitchens, etc. For services inside companies, or components in an electronic circuit, small items are well-defined and have

specific flow between them that must be considered when designing the layout-plan solution. In consequence, works on the field address the design with fixed number of entities (small items).

Now, one would assume that the Cutting & Packing literature may solve this gap as it studies a wider set of problems. However, the classification presented in Section 1.2.2 on page 6 Figure 1.1 on page 7 is not adapted to our problem. In fact, the more closely related works tackle open dimension problems and, as we have shown in Section 1.2.2 on page 6, the open dimension problem only relates one of the dimension of the small items to be packed, leaving the other dimension and the number of small items fixed. In consequence, we consider that the classification for two-dimensional Cutting & Packing problems should be modified in order to represent the properties of unknown number of small items and unfixed size (in both dimensions). Figure 2.1 illustrates the proposed classification.

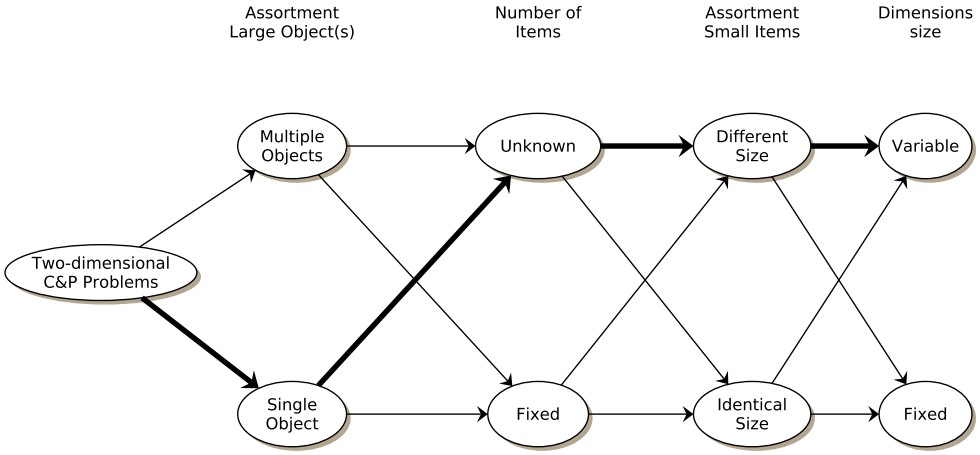


Figure 2.1 – Proposed classification for two-dimensional Cutting & Packing problems.

In the proposed classification are included all characteristics of the façade-layout synthesis problem, the thick-black path expresses all of them. An alternative to a new classification would be to consider the façade-layout synthesis problem as a variation of open dimension problem. However, we consider that there is enough additional elements to improve the classification and in that way represent a wider set of problems that may appear in the future. Our particular Cutting & Packing problem, that has been named as façade-layout synthesis, is defined as follows:

Definition 4 (Façade-layout synthesis problem): *Given a rectangular façade surface and an unbounded set of rectangular configurable panels, find a solution to determine the number of panels, assign size to them and place them over the façade under the next conditions:*

- *i Rectangular windows and doors over the façade must be* completely overlapped *by panels, each of them covered by one and only one panel (no partial overlapping).*
- *ii Panels must be attached in specific rectangular areas that are strong enough to support their weight.*
- *iii No holes are allowed in order to guaranty a perfect external insulation.*
- *iv The best solutions are composed of the minimum number of panels.*

Clarifying note #2

In the remaining of the document, we use *façade(s)* instead of large object(s) and *panel(s)* instead of small item(s) or rectangle(s).

Now, without further delay we introduce the details of the industrialized building renovation.

2.1.2 Industrial Problem

The addressed problem appears in a large French multi-partner project called CRIBA (for its acronym in French of Construction and Renovation in Industrialized Wood Steel) that aims to industrialize building renovation in order to reduce energy consumption (Vareilles et al., 2013, Aldanondo et al., 2014). One of the major problems for the industrialization project is to propose a computational process to assist the manual and automatic design of insulating envelopes that respect the set of limitations while minimizing the cost and maximizing the thermal performance.

This renovation is firstly based on a complete, true and accurate description of each of the façades in terms of geometry and structure and, secondly, on a precise description of the configurable panels. This first-hand knowledge enriched by the user's expectations about the renovation leads to an insulating envelope taking into account all the stakeholders' requirements. In this section we present the problem from the industrial point of view: The industrial context, industrial process, elements description and restrictions are made and a clear definition is laid down. The main purpose of the section is to understand the key issues of the industrial case. We shall begin by describing the context and process of the renovation.

Renovation Environment and Process

Although the core elements of the renovation are façades and panels, spatial entities are important for the manufacturing of panels, their shipment and their installation. This section then describes how spatial entities, for instance a building, imposes limitations over the configurable items, i.e., rectangular panels.

Façades habitats. Façades, as well as any spatial entity, are not isolated structures as they always belong to another bigger entity. For instance, a neighborhood belongs to a city and a city to a country. This *part-of* relation may be described as a hierarchical relation, the smaller part being lower in the hierarchy. This holds true in the renovation process for its spatial entities. This hierarchy is crucial for setting panels lower and upper bounds for a given façade renovation. For instance, if a building is in a region with high seismic activity, panels and attaching devices must be specially designed to deal with earthquakes. The hierarchy of the renovation is as follows. A façade is part of a building, a building is part of a block, a block part of a working site. An instance of such hierarchical structure is presented in Figure 2.2.

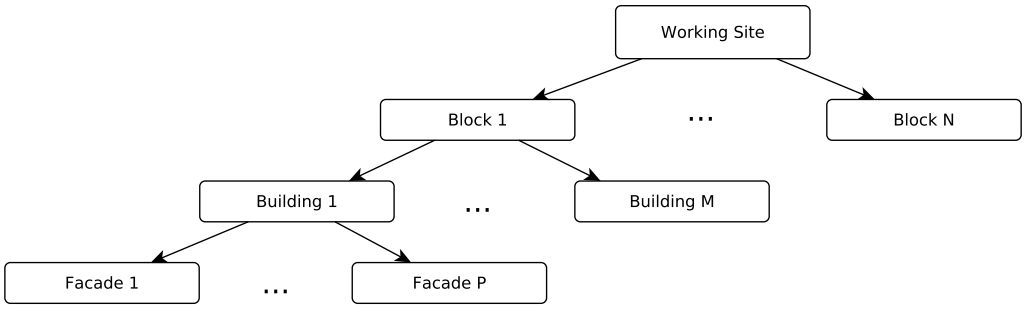


Figure 2.2 – Hierarchical view of renovation.

Now, each node in the Figure 2.2 is described by environmental properties. These environmental properties, for now assumed to be independent between nodes, impose some limitations on the size of panels. These environmental properties are the accessibility and weather conditions.

The accessibility conditions refers to the surroundings circumstances that allow to get in and out of the spatial entity. For instance, a working site may have wide streets and gates, railways and even a seaport. The more possibilities and the more easy to access the spatial entity the less limitations impose over the panels. Let us study this fact further.

Suppose two buildings A and B with the following accessibility conditions. The building A has big dual carriageway whereas the building B has a small one-way road. In addition, A has no trees in its surroundings whereas B has one garden and several trees in its proximities. Clearly, the building B is restricted in the size and number of trucks that can access it as well as the kind of supporting machinery to install panels. As expected, the smaller the trucks the smaller the panels. Now consider that surroundings circumstances may include playground, water sources, electric lines, unstable ground, sculptures and so on.

At the other end of the spectrum are the weather conditions. These conditions refer to the climatological circumstances in which the renovation takes part. From these circumstances the wind and the season are taken into account. On the one hand, if a given working site is subject to strong wind then either panels must be specially attached or their size must be further limited (reduced) in order to properly do the on-site assembly process. In this particular renovation the panels' size are limited. Conversely, if the season in which the on-site work takes place is winter, then it is physically harder for working, as well as technically harder, to move and install bigger panels than small ones. Moreover, the cost of the renovation may increase because more workforce is needed or because non-working days may come up due to the weather.

In short, the environmental properties of spatial entities play an important role in the renovation. A crucial aspect remains to be measured: The impact of these environmental conditions over the hierarchical structure of the renovation. But before that, the renovation process may provide a better idea of why this hierarchical view is important from the industrial perspective.

Renovation Process. To get a clear understanding about the limitations and constraints imposed over the `Cutting & Packing` problem by each one of the stages of the renovation (Aldanondo et al., 2014), it is necessary to have a global view of it. The following stages describe the renovation process.

Stage 1. Information collection. In this stage a certified surveyor uses laser range-finder devices, along with pattern recognition software, to create a numerical model describing the building. In essence, the model includes the dimension of each façade on the building, their windows, doors and any other element visually accessible to the lasers, i.e., the buildings' geometry.

Limitation *Façade size and panel size*

Each façade is potentially different from the others. Thus, to be consistent, panels size must be set according to the façade it will be allocated on. In addition, in this stage an early check is done: If frames (windows and doors) over the façade are bigger than the panel size, then no solution is possible.

Stage 2. Semantic enrichment. Once the building information model, or *BIM*, has been built, it is necessary to enrich it with three crucial information. This information is, first, the specific areas where it is possible to attach panels, along with their load bearing capabilities, second, the areas that are not part of `Cutting & Packing` problem as they need specific panel's design and, third, the architects requirements such as deletion or addition of windows, doors and balconies.

Limitation *Load bearing capability and panel size*

Assuming that the bigger a panel the heavier, size of panels may be limited by the load bearing capabilities of the façade they will be attached onto. If the envelope, plus the future look (optional cladding) plus new windows, is not successfully supported by the façade, it may collapse.

Stage 3. Envelope definition. Using the complete numerical model, the generation of envelopes is executed. In this stage, each of the façades to renovate is synthesized using `Cutting & Packing` algorithms.

Limitation *Cutting & Packing algorithms and envelopes*

The insulating envelopes design is limited by two facts. First, the computational tools (hardware and software) used in the design. And second, the algorithmic solutions to tackle this specific problem.

Stage 4. Manufacturing. Once an insulting envelope has been designed in the previous stage, the panels composing it are manufactured in the factories. Each panel is manufactured one by one, with their specific size, including new frames (windows and doors), and with every detail for its installation.

Limitation *Manufacturing and panel size*

A given panels' provider has limitations on the manufacturing process. Thus, each manufacturer provides:

— Two bounds on the size for panels; one bound indicating the minimum panel size and another bound indicating the maximum panel size.

— A minimum distance between panels' border and frames' border must be assured for a correct panel manufacturing.

— A constraint between width and height stating the possible combination of size that may be manufactured.

Stage 5. Supply chain. In this stage panels are shipped to the working site for their installation. Additionally, support for installing the panels is decided. This support includes cranes, harnesses and any additional machinery needed by workers.

Limitation *Supply and panel size*

Panels size may be limited by the available machinery. For instance, the size of panels is constrained w.r.t. the size of trucks available for their transportation (e.g., the smaller the trucks the smaller the panels).

Stage 6. On-site installation. The final stage is the installation of panels on each one of the façades. The supply chain planning of the CRIBA project is the core of the PhD thesis of a colleague in Mines d'Albi (Gholizadeh-Tayyar et al., 2015).

Limitation *Installation and panel size*

Accessibility conditions may restrict the use of transportation, cranes or other supporting machinery. As a bandwagon effect, supporting machinery affects panels size.

The aforementioned stages describe the global process of the renovation. The dissertation focuses on the **Stage 3**. From an envelope-definition point of view, the limitations on each stage have an impact on the limits for panel size. When executing the `Cutting & Packing` algorithms, panel size bounds, product of the environmental conditions, hierarchy and each stage limitation, are unequivocally set and are not changed during the `Cutting & Packing` process. For the time being, let us study further each one of the elements taking part on the renovation.

Renovation Elements

In the interest of understanding the `Cutting & Packing` problem, it is necessary to deeply describe the elements (façades and configurable panels) taking part on the building renovation process. Limitations on the renovation process and the manufacturing of items are considered as *constraints* that are included in the constraint-based model of the `Cutting & Packing` problem.

Rectangular Façades. The description of the façades is one of the key points in building renovation. Their description has to be complete, true and accurate in terms of geometry and structural properties. Indeed, each of the façades in a building has potentially different size, number and position of frames (windows and doors) and different strength (how much additional weight the façade can support and where this additional weight can be located). Clearly, the design of an insulating envelope depends on the geometry and strength of the façade to be renovated.

Information about façades is essentially acquired from a certified surveyor for their geometry and from a structural engineer for their structural properties. Only specific elements are kept for the design of the envelope: (1) frames, such as windows, doors and garage entrances, and (2) supporting areas, such as shear walls and concrete slabs. All the rest, such as rain gutters, balconies or street lights directly attached to the building, are removed and thus not considered in the problem. Also, the building's gable (triangular portion of a wall between the edges of intersecting roof pitches) is considered out of configuration as it needs specific panels shapes that will be manually tuned by the architect. Lastly, upon architects request and for any reason, any given portions of the façade may be leave out of the `Cutting & Packing` process. As a restriction to these portions, that we call *zones out of configuration* (ZOC), their shape must be rectangular as well so to not interfere with the `Cutting & Packing` process and to treat these zones as already-defined panels.

Façade Restrictions: In the problem addressed here, we only consider rectangular elements: façades, frames and supporting areas are approximated by rectangular shapes. Indeed, gables and round frames make the formalization of the envelopes design problem as well as the research of solutions very complex and are therefore out of scope as we target a first solving approach.

Façade Definition: A façade, as illustrated in Figure 2.3 on the next page is represented by a two-dimensional rectangular vertical plane, with origin of coordinates at the bottom-left corner of the façade ($fac_{xo} = 0, fac_{yo} = 0$), containing rectangular zones defining:

- Perimeter of façade with its size: height fac_h and width fac_w in meters.
- Frames. Existing windows and doors over the façade play the first key role as they are meant to be completely overlapped by *one and only one* panel. Frames fr^j are defined by:
  - Origin point, i.e. the bottom-left corner, (fr^j_x, fr^j_y) with respect to origin of façade (fac_{xo}, fac_{yo}).
  - Width fr^j_w and height fr^j_h in meters.
- Supporting areas. Supporting areas over the façade play the second key role as they support the weight of the panels composing the insulation envelope. Supporting areas sa^k have well-defined:
  - Origin point, i.e. the bottom-left corner, (sa^k_x, sa^k_y) with respect to origin of façade (fac_{xo}, fac_{yo}).
  - Width sa^k_w and height sa^k_h in meters.
  - Load bearing capability sa^k_l in kg/m^2.
- Rectangular zones out of configuration zoc_m with:
  - Origin point, i.e. the bottom-left corner, (zoc^m_x, zoc^m_y) with respect to origin of façade (fac_{xo}, fac_{yo}).
  - Width zoc^m_w and height zoc^m_h in meters.

  These zocs are treated as user-defined panels and thus will affect by not being considered in the envelope's design.

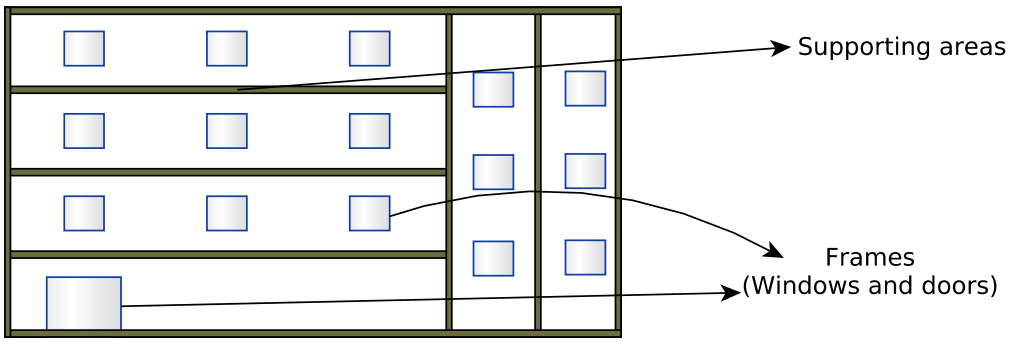


Figure 2.3 – Façade description: frames and supporting areas.

Cascade effect over façades: As commented previously, spatial entities inherit environmental conditions of entities on superior levels of the hierarchy. This fact is decisive for setting the panels bounds for a given façade. A simple illustration helps in understanding this fact.

Let us consider again the buildings A and B now belonging to the working site W. Suppose three levels accessibility conditions: easy, medium and hard. Also, suppose three types of trucks: big, medium and small. Further, let us assume three types of panel size: big, medium and small. Now let us instantiate the accessibility of the working site W to medium, and the accessibility to the building A easy and building B hard. Then access to the working site W can be done with medium size and small size trucks. These trucks can carry medium size and small size panels. Given that A building's accessibility conditions is easy, it can be accessed using both medium size and small size trucks. But it is not the case for building B with hard accessibility conditions which can only be accessed by small trucks with small panels. This is a rather logical consequence.

From the point of view of the building A, it can be accessed with big size trucks and big size panels; the accessibility condition is easy. But, here is where dependency in the hierarchy originates. For the working site cannot be accessed by big size trucks due to his accessibility conditions and, in order to access the building, it is mandatory to access the working site. Therefore, the accessibility condition of the working site is inherited by the building, restricting it to medium size and small size trucks. This property of inheritance is a monotonic operation, i.e., it can only reduce panels size.

Bear in mind during the rest of the document that all these environmental conditions, and their inherit property, have an impact on the panels size and consequently in the thermal performance of the insulating envelopes.

Rectangular panels. Panels are rigid rectilinear rectangles (see Figure 2.4 on page 32) with sides parallel to the façade reference axis. The problem, as explained before, is an instance of a orthogonal two-dimensional `Cutting & Packing` problems. The description of the insulated panels is the second key point in building renovation. Their description has to be precise in terms of admissible sizes, weights and thermal properties. Indeed, each panel is configurable and has therefore a range of different sizes (width and height), different weights (depending on several factors relative to their size, insulation characteristics, included frames, chosen cladding), a certain number of included frames and different thermal performances (taking into account insulation type, insulation thickness and included frames). Clearly, the design of an insulating envelope depends on the admissible size of panels. We have to highlight the fact that each panel is especially configured (size fitted, weight computed, insulation selected, and so on) and manufactured with the appropriated structure containing the new frames for each façade. There is no on-site fitting (the accuracy is around 1mm).

Information about panels is essentially acquired from the panels manufacturers. They have

to specify the panels range of widths p_w (lower and upper bounds), heights p_h (lower and upper bound), the constraints linking the width p_w and the height p_h of the panels (width upper bound not compatible with height upper bound, for instance), the possible insulation thickness I_{th} (lower and upper bound), the insulation types I_{ty} and the constraints linking thickness I_{th} and type I_{ty}, and the range of external cladding types Cl_{ty} which are allowed (range of colors, from natural to exotic ones, and materials and textures, such as wood, vinyl, stone or aluminum).

Panels are manufactured prior to shipment. The chosen mode of transportation (special convoys or small trucks) has an impact on their size: the use of special convoys to deliver the panels will not affect their admissible sizes (every combination is allowed), on the contrary, the use of small trucks will impact them: the panels have to be smaller in order to fit the truck size.

Panels have an orientation (horizontal or vertical) depending on the ratio between their width p_w and height p_h. If the ratio $\frac{p_w}{p_h}$ is less than one, the panel is vertical, otherwise, it is horizontal. This information impacts the inner structure of the panel and its laying direction onto the façade, even more pertinent if the panel contains frames. When a panel contains frames, given its internal structure, a minimal distance d must be respected between the panel borders and each frame required space. Panels are also characterized by a thermal performance TP_p and a weight W_p both depending on its size (p_w, p_h), its insulation thickness (I_{th}), its insulation type (I_{ty}), the frames it contains and the cladding Cl_{ty} only for the weight computation.

Panels can be attached onto a façade in different manners: hanged, supported or stapled along their perimeter. The way to attach panels onto the façade has an impact on the way to spread their weights over the supporting areas and consequently on the insulating envelope. They are attached by means metallic devices called *fasteners*. Each fastener by its own is defined by the length of its threaded shaft: The part that goes into the wall. Additionally, these elements consist of two parts: One fixed directly onto the façade (support bracket) and one installed on the panel at the factory (panel bracket). The exact position of fasteners depend on the position of panels. Thus, fasteners definitions are only taken into account in the support system (Chapter 5 on page 127) when the insulating envelopes are already designed.

Panels Restrictions: In the problem addressed here, we only consider two main characteristics of the panels: their size and weight. Indeed, the insulation type and its thickness are determined for the whole renovation by energy performance simulations at the beginning of the renovation process. Therefore, panels thermal performance becomes unnecessary to compute by itself but thermal leaks become critical to estimate. These leaks appear mainly at the junction between panels. Our objectives of maximizing panels size and minimizing their number is therefore strengthened. Considering the weight, the cladding is chosen upstream for the whole renovation similarly to the insulation type and thickness. Its weight is directly included in the panels weight. The panel's weight is, potentially, supported by two supporting areas. If the panel is supported by its bottom edge, then the weight is concentrated on the bottom-left and bottom right corners whereas the remaining attaching act mainly as a stabilizing mechanism. If the panel is hanged in its top edge, then its weight if concentrated on the top-left and top-right corners whereas the remaining attaching act mainly as a stabilizing mechanism. Finally, if the panel is attached by the lateral edges, then its weight is concentrated on the lowest attaching point in both lateral edges whereas the remaining attaching act mainly as a stabilizing mechanism. Assuming that a fastener can support two contiguous panels, we check for each panel if its total weight can be supported by the relevant supporting areas of each attaching point.

Panel Definition: A panel p^i (see Figure 2.4 on the following page) is represented by a two-dimensional rectangular shape with origin of coordinates at the bottom-left corner of the panel (p^i_{xo}, p^i_{yo}) and is defined by:

— Size: width p^i_w and height p^i_h in meters. Its size is restricted by given limits: $p^i_w \in [p_{wl}, p_{wu}]$ and $p^i_h \in [p_{hl}, p_{hu}]$. The domains for p^i_w and p^i_h are defined by two intervals: $[min^{\downarrow}, max^{\downarrow}]$ and $[min^{\uparrow}, max^{\uparrow}]$, with $max^{\downarrow} < max^{\uparrow}$. Manufacturing limitations state that one panel dimension must take as domain the first interval and the other dimension the

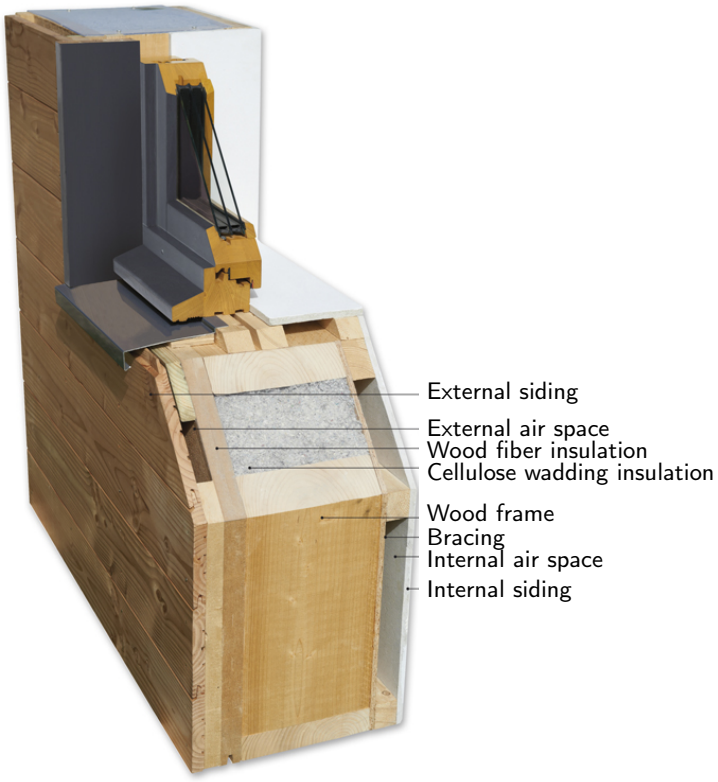


Figure 2.4 – Technical concept: Rectangular configurable panel.

second interval. In other words, it does not matter which interval is taken by one dimension as far as the other dimension takes as domain the other interval. This limitation forces the panel to have an orientation except when $p^i_w = p^i_h$, in which case $p_{wu} = p_{hu} = max^{\downarrow}$. An illustrative example of this limitation is presented in Section 2.2.3 on page 38.

— Orientation. The orientation is determined by the ratio between its width p^i_w and height p^i_h. If the ratio $\frac{p^i_w}{p^i_h} < 1$ then orientation is vertical, else horizontal.

— New frames with defined origin point (nfr^r_x, nfr^r_y), corresponding to the bottom-left corner of the frame, with respect to the panel origin (p^i_{xo}, p^i_{yo}) they belong to. Each frame nfr^r has a width nfr^r_w and a height nfr^r_h in meters. Also, these new frames must respect the minimum distance d to the panels borders.

— Weight. The weight of a given panel p^i_{we} depends on different factors: size (p^i_w, p^i_h), thickness (I_{th}), insulation type (I_{ty}) and new frames (ne^r_{we}). Regarding our restrictions on panels, its weight is computed with:

$$p^i_{we} = (p^i_w \times p^i_h) \times \beta \tag{2.1}$$

where p^i_w and p^i_h are respectively the panel p^i width and height and β is a factor which depends on the panel material, its internal structure, the chosen thickness I_{ty}, the chosen insulation type I_{ty} and the chosen cladding Cl_{ty}.

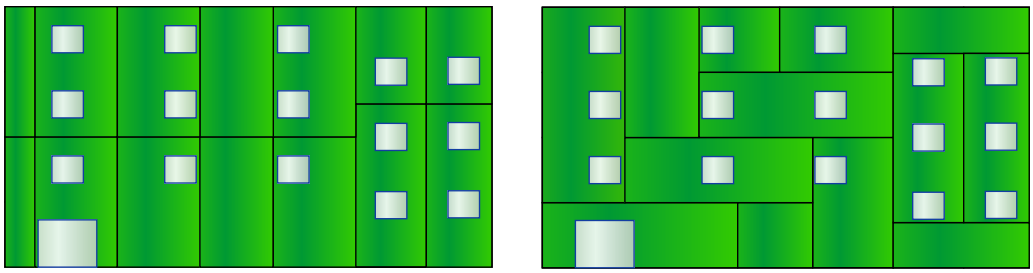
The weight of panels is supported in a square meter of the façade surface where the fasteners devices are located. Ergo, the weight (in kilograms) is compared to the load bearing capabilities (in kilograms) of a square meter supporting area where the fastener is located.

Renovation Requirements

Renovation requirements or expectations have also an impact on the insulating envelope design and style. In order to give to the renovated building a specific good looking with respect to urban design guidelines, building owners' expectations and tenders' wishes, architects need a complete overview on the renovation and have to achieve acceptable compromise solutions.

Firstly, they have to select from the large range of cladding CL_{ty} the one composing the new skin of the renovated building. This aesthetic choice has an impact on the weight of the panels p_{we} and therefore, on their maximum size (p_{wu},p_{hu}) but also on the orientation of the panels (horizontal or vertical).

Secondly, despite all possible care taken for the building renovation, the joints between panels are still visible and draw simple geometric patterns onto the façade. Architects may turn to advantage these visible joints by imposing a preferred orientation for the panels P_o. The new envelope is therefore composed mainly by horizontal or vertical panels, all aligned and drawing good looking patterns, such as straight vertical lines as illustrated in Figure 2.5 (a), straight horizontal lines, brickwork style, 90 degree Herringbone bond as illustrated in Figure 2.5 (b), etc.



(a) Vertical lines pattern

(b) 90 Degree herringbone bond pattern

Figure 2.5 – Façade geometric patterns examples.

Thirdly, when designing the envelope, architects keep constantly in mind as a target the beauty of the façade. The panels size are consequently thought out with the utmost care and tuned if necessary (reduction of their size, modification of their location onto the façade, etc) with always the most minimal impact on the final envelope design and style.

Fourthly, always for aesthetic reasons, the architects start designing the new envelope by a specific point ($start_x$,$start_y$), such as the middle of the façade meaning that $start_x = \frac{fac_h}{2}$ and $start_y = \frac{fac_w}{2}$ or the bottom-left corner meaning that $start_x = fac_{x0}$ and $start_y = fac_{y0}$, depending on the expected final design result and architect's artistic flair.

Renovation Requirements Restrictions: In the problem addressed here, no aesthetic aspect is considered as hard requirement as the process to design a good looking envelope is based on high human skill, know-how and flair, really difficult to formalize (tacit knowledge). A proposed cutting solution `CaSyE` presented in Section 3.3 on page 75, however, is conceived to generate more aesthetic envelopes by considering symmetry. Additionally, the chosen cladding CL_{ty} is taken into account directly into the panels weight computation and consequently on the permissible upper bounds.

Considering the resultant geometric patterns due to the joints between panels, when using the support system the architect has the possibility to mention which panels orientation P_o (horizontal, vertical, none) is preferable to the other. This preferred orientation can be either used as a soft or hard constraint:

— Using it as a soft constraint, it allows the algorithms to swap the orientation of a panel (for instance, from horizontal to vertical) when they fail to find a solution.

— By contrast, using it as a hard constraint, it forces the algorithms to find envelopes respecting the architect's orientation preference.

— When no orientation is specified (none), the algorithms are able to find close to optimal solutions by always selecting the largest panels whatever their orientation.

Optimality issues: The target is to reduce energy consumption of buildings by means of an external panels-made insulating envelope. This insulating envelope has as goal to reduce the heat transfer between the interior and the exterior of the building. Thus, an ideal envelope will have zero heat transfer, i.e., the best possible thermal performance. Given that all panels in an envelope are produced by a single manufacturer, it is the case that all panels in a given envelope has the same internal structure in such a way that any panel is indistinguishable from any other except for its size and position over the façade. In that way the performance of a given envelope is computed only with respect to panels allocation and sizing.

Now, differences on thermal performance for different envelopes for a given façade do not depend on the façade size. Considering that a valid envelope is the one covering all the façade, then it is the case that the summation of all panels areas equals the façade area, no matter how many panels or in which position they are. Then, the panels insulation is constant over a given façade, but it is not the same for an envelope. In fact, the junction between two consecutive panels associates to the envelope a *thermal leak*. So, from the energy conservation perspective, the thermal performance of an envelope depends only on the *length of junctions* plus façade perimeter. Hence, an envelope composed by one panel will provide better insulation that an envelope of two panels over the same façade. Let us study an illustration of this fact.

In Figure 2.6 we can see that both envelopes have the same number of panels, six of them. Although both envelopes are valid because they cover the entire façade and respect all limitations, they do not have the same thermal performance and cost. The façade at the left has a bigger thermal transfer because its junctions length is bigger (13 meters×5 + 5 meters×2 + 10 meters ×4 = 115 meters) than the envelope of façade at the right (18 meters×2 + 10 meters×7 = 106 meters).

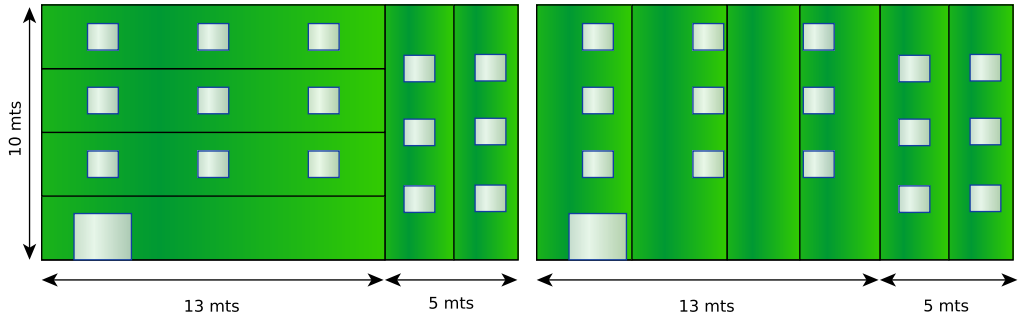


Figure 2.6 – Two valid envelopes with different junctions length: Envelope at the left has 115 meters and envelope at the right has 106 meters.

Computing the length of the junctions for a given envelope is straightforward, Formula 2.2 expresses this knowledge

$$fac_{loj} = \sum_{i=1}^{N} (p_w^i + p_h^i) + (fac_w + fac_h) \tag{2.2}$$

where fac_{loj} stands for length of junctions of façade fac, fac_w and fac_h are, respectively, the width and height of the façade, p_w^i and p_h^i represent the width and height of panel i, respectively, and N is the number of panels composing the envelope.

Clarifying note #3

The cost of an envelope is computed with respect to the cost of a panel of a given manufacturer. One of the associated companies in the CRIBA project has provided us some information on how to compute the costs c_p for panels they manufacture. Thus, knowledge on how to compute the cost of a given façade envelope fac_c is represented in the Formula 2.3

$$fac_c = \sum_{i=1}^{N} (p_w^i \times p_h^i \times \alpha) \tag{2.3}$$

where p_w^i and p_h^i represent the width and height, respectively, of panel i. The constant α is a factor provided by the manufacturer that depends on the particular manufacturing process.

Recall that an insulating envelope is characterized by three related aspects. First, the number of panels regardless the position. Second, the length of junctions between panels. Finally, the associated cost of the envelope. From these characteristics the number of panels gives a good approximate of the length of junctions and cost. Actually, minimizing the number of used panels entails the junctions length and cost minimization. Then, the optimization is reduced to minimizing the number of panels used on an envelope. However, as we saw in Figure 2.6 on the facing page, two insulating envelopes with the same number of panels may be compared using the length of junctions, preferring always the one with minimum length of junctions.

Support system requirements

Within the CRIBA project different needs on the insulating envelopes design, logistics and interoperability between partners have been identified. In this thesis we focus our attention in supporting the architects decision-making by modeling the design problem and developing computer-based solutions to tackle it. The computer-based solutions are merged together into a PCS dedicated to configure insulating envelopes. This means, on the one hand, that functional requirements for the support system must be isolated in order to fully satisfy the project needs. And, on the other hand, it means that non-functional requirements, to successfully implement the functional requirements, must be clearly defined. After a joint analysis with companies within the project and academics at the École des Mines d'Albi-Carmaux, these requirements have been defined as follows.

— Functional requirements:

1. Manual design of envelopes. In this setup the architects draws panels, one by one, on a graphical user interface. The system is responsible to either inform about constraint conflicts (as in `InDiE`) or, ideally, correct conflicts and throw a compliant solution (as in `SkEdE`). Interactive behavior and short delays are expected in this setup.
2. Automatic design of envelopes. In this setup, architects push a button and expect from the system compliant envelopes solutions.
3. Semi-automatic design of envelopes. This setup mixes the two previous approaches by letting the system finish the design of a partial envelope manually drawn by an architect.
4. Given that the architect looks for both valid envelopes as well as aesthetic ones, the system should provide different insulating envelopes solutions for a given façade.
5. Computing the cost and length of junctions is a task delegated to the system. Further, if the system throws several solutions for a given façade, then a ranking w.r.t. to the number of panels, and within this envelopes with minimum length of junctions, must be guaranteed.

— Non-functional requirements:

1. The major non-functional requirement refers to portability. The decision is then to allow the support system to run over Internet browsers. The system is then a web-oriented application.
2. The input (spatial entities specification) and output (envelopes numerical models) format should be standard as to allow the communication with other systems. The decision is then to use a web-compliant format called JSON, that is used in many applications and may be parsed effortless to many other formats.
3. Finally, the response time of the system, when generating automatic solutions, is not a critical requirement as the architects have enough time to plan the renovation. However, the expected time for the generation of insulating envelopes should not exceed 30 seconds for a façade of 20 meters width and 10 meters height.

2.2 Constraint Model

A constraint model using the above problem description is here introduced. The model is intended to reflect the key issues of the problem in a simple and declarative way. Thus, to formalize the packing problem as a CSP, the parameters describing the façade and panel's size lower and upper bounds, variables describing the packing solution and constraints describing the relations over panels and façade, are introduced.

2.2.1 Parameters

In order to give limit to variables, a set of parameters is necessary. In essence, the parameters contain all geometrical and structural information linked to the façade and needed to establish the relation among panels, and panels and façade. For instance, the position of frames, and their size, are important in order to avoid their partial overlapping. Further, in order to map the model into a computer-based solution, it is important to have an accurate description of the façade and its elements, regardless of the input format.

Each renovation being unique, the problem has to be adjusted considering some parameters:

- Considering the façade:
  - Height fac_h and width fac_w in meters.
  - Set F of frames and for each frame $fr^j \in F$:
    - Origin point (fr^j_x, fr^j_y) with respect to origin of façade (fac_{xo}, fac_{yo}).
    - Width fr^j_w and height fr^j_h in meters.
  - Set Sa of supporting areas and for each supporting area $sa^k \in Sa$:
    - Origin point (sa^k_x, sa^k_y) with respect to origin of façade (fac_{xo}, fac_{yo}).
    - Width sa^k_w and height sa^k_h in meters.
    - Load bearing capability sa^k_l in kg/m^2.
  - Set Zoc of zones out of configuration and for each zone $zoc^m \in Zoc$:
    - Origin point (zoc^m_x, zoc^m_y) with respect to origin of façade (fac_{xo}, fac_{yo}).
    - Width zoc^m_w and height zoc^m_h in meters.
- Considering the panels:
  - Minimal distance d between the panel borders and each frame required space,
  - Size limits (lower and upper bounds for each dimension) defined by $[min^{\downarrow}, max^{\downarrow}]$ and $[min^{\uparrow}, max^{\uparrow}]$, with $max^{\downarrow} < max^{\uparrow}$.
  - Weight factor β in order to compute the weight of the panels.
  - Cost factor α in order to compute the cost of the envelope.
  - Constraint between width and height stating the compatibility between sizes allowed by the manufacturing process.
- Considering the renovation requirements:
  - Preferred orientation for the panels P_o: vertical or horizontal.

2.2.2 Decision Variables

Decision variables are linked to the position and size of each panel in an insulating envelope. Let us assume that N represents the number of panels in a given insulating envelope. Then, each panel p^i with $i \in [1, N]$ is described by its origin and size attributes:

- $p^i_{x0} \in [0, fac_w]$ is the origin of rectangle p^i in the horizontal axis.
- $p^i_{y0} \in [0, fac_h]$ is the origin of rectangle p^i in the vertical axis.

— $p^i_w \in [min^{\downarrow}, max^{\downarrow}]$ or $p^i_w \in [min^{\uparrow}, max^{\uparrow}]$ is the length of the rectangle p^i the horizontal axis.

— $p^i_h \in [min^{\downarrow}, max^{\downarrow}]$ or $p^i_h \in [min^{\uparrow}, max^{\uparrow}]$ is the length of the rectangle p^i the vertical axis.

Domain for each variable depends on the variable semantics. As well, some result variables that are not part of the solving process but rather generated by a given solution are:

— The number of panels N composing the insulating envelopes,

— for each panel covering frames, for each frame it contains: Position (nfr_x, nfr_y) with respect to the panel origin (p_{xo}, p_{yo}) as well as width (nfr_w) and height (nfr_h) in meters,

— weight $p^i_{we} \in [p_{wel}, p_{weu}]$ of each panel, computed with formula 2.1 on page 32 ($p^i_{we} = (p^i_w \times p^i_h) \times \beta$),

— length of junctions fac_{loj} computed with the panels' size (p_w, p_h) and façade size (fac_w, fac_h) and,

— cost fac_c computed with the panels' size (p_w, p_h).

2.2.3 Constraints

In order to design the envelope of a given façade, stakeholders knowledge is mapped into constraints over the variables representing panels. The following constraints have been extracted from the problem domain by stakeholders (e.g., architects and building renovation companies). They state the properties well-designed panels, consequently envelope, must posses. Instances of good and ill panels design are illustrated for each constraint.

Design constraints

We shall begin the description of four constraints that are related to the problem specification.

Size *Panels have a range of width $p^i_w \in [p_{wl}, p_{wu}]$ and a range of height $p^i_h \in [p_{hl}, p_{hu}]$.*

$$\forall p^i, 1 \leq i \leq N: \quad p_{wl} \leq p_w \leq p_{wu} \wedge p_{hl} \leq p_h \leq p_{hu} \tag{2.4}$$

Size compatibility *This constraint links the width and the height of panels to reflect manufacturing limitations. It states a compatibility between the domains of p_w and p_h with respect to the panel sizes permitted by the manufacturing process and defined by $[min^{\downarrow}, max^{\downarrow}]$ and $[min^{\uparrow}, max^{\uparrow}]$, with $max^{\downarrow} < max^{\uparrow}$. More precisely, the constraint states that p_w must take one of these domains and p_h the other one. This knowledge is expressed as*

$$\begin{aligned} \forall p^i, 1 \leq i \leq N: \quad & (min^{\downarrow} \leq p_w \leq max^{\downarrow} \wedge min^{\uparrow} \leq p_h \leq min^{\uparrow}) \\ & \vee \ (min^{\uparrow} \leq p_w \leq max^{\uparrow} \wedge min^{\downarrow} \leq p_h \leq min^{\downarrow}) \end{aligned} \tag{2.5}$$

Then, for instance, $p_w \in [min^{\uparrow}, max^{\uparrow}]$ and $p_h \in [min^{\uparrow}, max^{\uparrow}]$ is not allowed. Figure 2.7.a, illustrates the two panel size possibilities. Thus, two panel instantiation are possible, as shown in Figure 2.7.b. Note also that when $max^{\downarrow} \neq max^{\uparrow}$ the constraint imposes a panel orientation.

Area *The entire façade must be covered with panels. It implies no holes in the envelope and that the summation of all panels areas equals the façade area. Figure 2.8.a presents an envelope with a hole and thus does not cover all façade areas as does Figure 2.8.b.*

$$\sum_{i=1}^{N} (p^i_w \times p^i_h) = fac_w \times fac_h \tag{2.6}$$

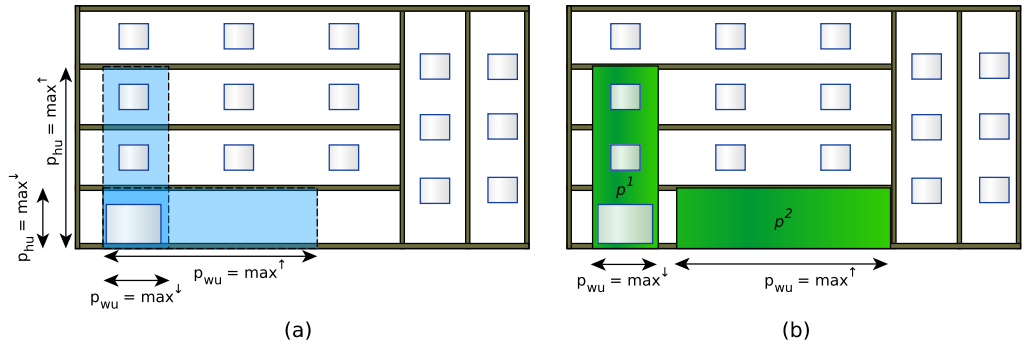


Figure 2.7 – a) Possible panel instantiations and b) valid instantiations w.r.t. size compatibility.

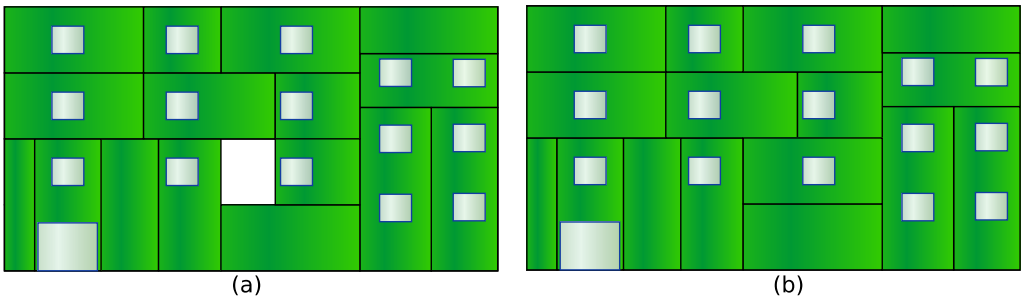


Figure 2.8 – a) Ill-defined envelope and b) well-defined envelope w.r.t. area constraint.

Non-Overlapping *Panels overlapping is forbidden. It means that for two given panels u and v there is at least one dimension (vertical or horizontal) where their projections do not overlap.*

$$\begin{aligned} \forall_{p^u,q^v} \mid\ & p^u_{x0} \geq p^v_{x0} + p^v_w \vee p^v_{x0} \geq p^u_{x0} + p^u_w \\ \vee\ & p^u_{y0} \geq p^v_{y0} + p^v_h \vee p^v_{h0} \geq p^u_{h0} + p^u_h \end{aligned} \tag{2.7}$$

The panel p^2 in Figure 2.9.a overlaps panel p^1. A correct allocation of the same panel is shown in Figure 2.9.b.

Frames *Each frame over the façade must be completely overlapped by one and only one panel. Additionally, frame borders and panel borders must be separated by a minimum distance denoted by d:*

$$\begin{aligned} \forall_{fr^j} \in F, \exists_{p^i}, 1 \leq i \leq N \mid\ & p^i_{x0} + d \leq fr_x \wedge fr_x + fr_w \leq p^i_{x0} + p^i_w + d \\ \wedge\ & p^i_{y0} + d \leq fr_y \wedge fr_y + fr_h \leq p^i_{y0} + p^i_h + d \end{aligned} \tag{2.8}$$

In Figure 2.10.a, the panel is not correctly designed as it partially overlaps a frame. A correct design would be either completely cover the window or, as presented in Figure 2.10.b, avoid the overlapping by leaving the responsibility to another panel.

Installation *Panels must be attached by their corners over supporting areas. Let $p_{x1} = p_{x0} + p_w$ and $p_{y1} = p_{y0} + p_h$, then we have*

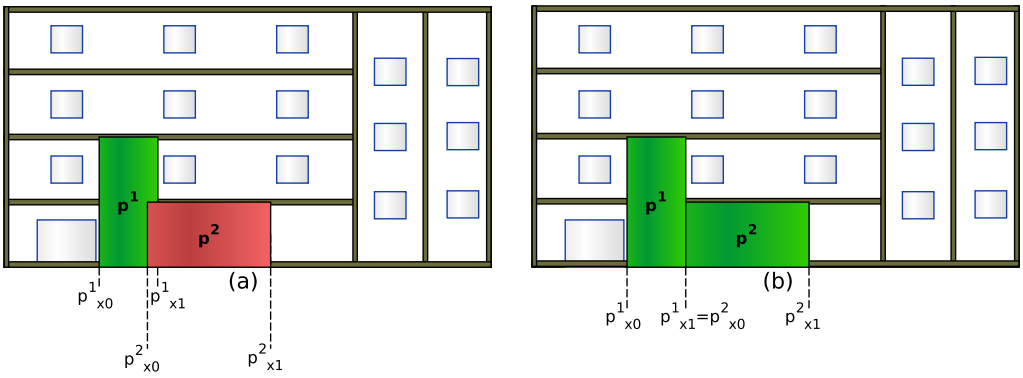


Figure 2.9 – a) Ill-defined panel and b) well-defined panel w.r.t. non-overlapping constraint.

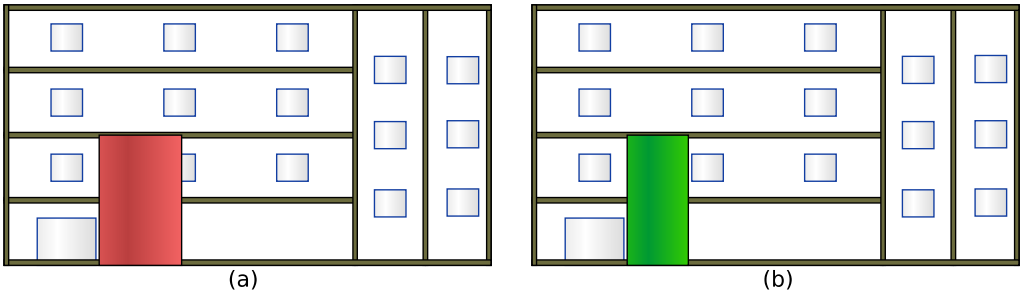


Figure 2.10 – a) Ill-defined panel and b) well-defined panel w.r.t. frames constraint.

$$
\begin{aligned}
&\forall_p, \\
&\exists_{sa^k} \mid (sa^k_x \le p_{x0} \wedge p_{x0} \le sa^k_x + sa^k_w \wedge \ sa^k_y \le p_{y0} \wedge p_{y0} \le sa^k_y + sa^k_h) \\
&\exists_{sa^l} \mid (sa^l_x \le p_{x0} \wedge p_{x0} \le sa^l_x + sa^l_w \wedge \ sa^l_y \le p_{y1} \wedge p_{y1} \le sa^l_y + sa^l_h) \\
&\exists_{sa^m} \mid (sa^m_x \le p_{x1} \wedge p_{x1} \le sa^m_x + sa^m_w \wedge \ sa^m_y \le p_{y0} \wedge p_{y0} \le sa^m_y + sa^m_h) \\
&\exists_{sa^n} \mid (sa^n_x \le p_{x1} \wedge p_{x1} \le sa^n_x + sa^n_w \wedge \ sa^n_y \le p_{y1} \wedge p_{y1} \le sa^n_y + sa^n_h)
\end{aligned}
\tag{2.9}
$$

Note $k = l = m = n$ is possible. The panel in Figure 2.11.a cannot be attached as its top-right corner does not match a supporting area. Conversely, Figure 2.11.b shows the same panel well designed.

Weight *Assuming that a fastener can support two contiguous panels, we check for each panel if its weight p_{we} can be supported by the relevant supporting areas sa^k_l (in a square meter). The following constraint expresses the knowledge*

$$p_{we} \le sa^k_l \tag{2.10}$$

Supplementary constraints

As we want to present to the end-user (e.g. an architect) a diverse set of good if not optimal insulating envelopes, we must avoid to enumerate symmetrical ones. Additionally, an

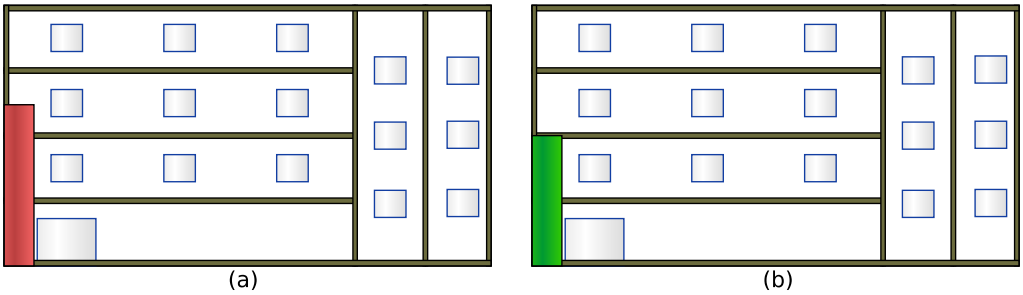


Figure 2.11 – a) Ill-defined panel and b) well-defined panel w.r.t. installation constraint.

interference constraint helps to avoid failure when positioning panel at the façade edges. The following constraints then improve the solving process and the solutions quality.

Interference *In order to allow forthcoming panels to be placed, a given panel p^i must either be at the façade edge or ensure that enough space is left to fix another panel.*

$$\begin{aligned} \forall p^i, 1 \leq p^i \leq N \ & (p^i_{x0} + p^i_w \leq fac_w - p_{wl} \vee p^i_{x0} + p^i_w = fac_w) \\ & \wedge \ (p^i_{y0} + p^i_h \leq fac_h - p_{hl} \vee p^i_{y0} + p^i_h = fac_h) \end{aligned} \tag{2.11}$$

The minimum space depends on the panels lower bound in a given dimension (respectively p_{wl} for panels width and p_{hl} for panels height). For example, in Figure 2.12.a the panel is not well-designed as it does not allow the allocation of a new panel below it. In Figure 2.12.b the same panel is well-designed as it leaves enough space for further panels.

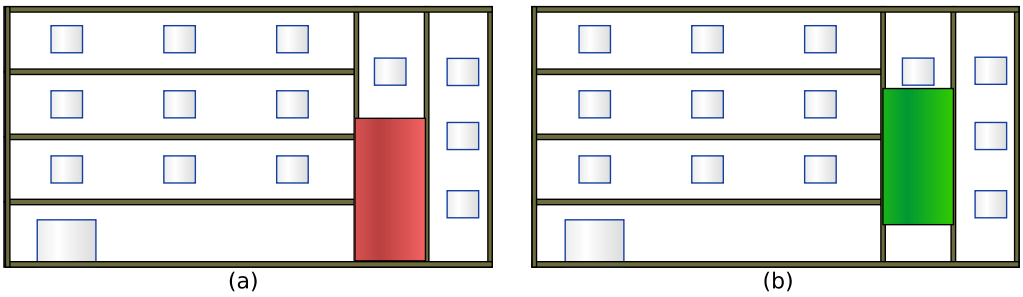


Figure 2.12 – a) Ill-defined panel and b) well-defined panel w.r.t. interference constraint.

Ordering *Panels are ordered: This is done by imposing an ordering on p_{x0} and p_{y0}*

$$\texttt{LexChainLessEq}(\{\{p^i_{x0}, p^i_{y0}\} | 1 \leq p^i < N\}) \tag{2.12}$$

This lexicographic constraint (van Hoeve and Hooker, 2009) ensures that priority is given to use the first rectangles and that rectangles are ordered. Assigning an order forbids the swapping of rectangles and thus avoid the generation of symmetrical solutions (as explained in section 1.2.2 on page 6).

An illustration of this constraint is presented in Figure 2.13 on the next page. In the first case, façade at the left, panels enumerated as p^1, p^2, p^3 are ordered, i.e., $p^1_{x0} \leq p^2_{x0} \leq p^3_{x0}$. This ordering prevents the second case, façade at the right, from happening and thus preventing the generation of symmetrical solutions.

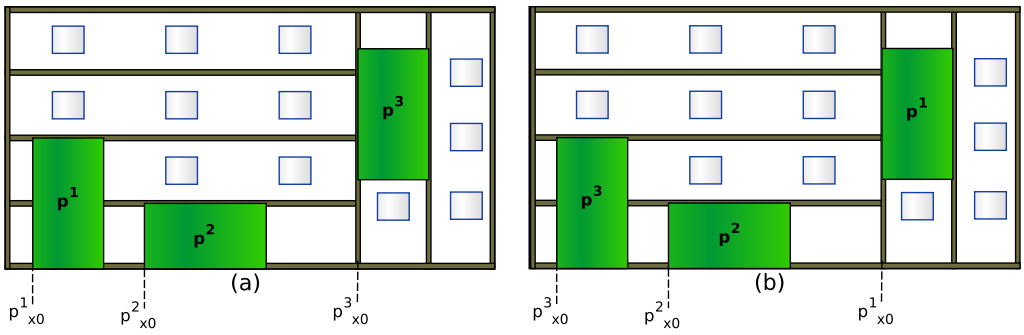


Figure 2.13 – a) Reachable panels' design and b) not reachable panels' design w.r..t. ordering constraint.

2.3 Evaluation Cases

In this section we introduce three façade instances on which the algorithmic solutions will be tested. The first two instances, Figures 2.14 on the following page and 2.15 on the next page, are real façades part of the working site *La Pince* in the commune Saint Paul-lès-Dax in the department of Landes, France. These two façades have:

— Several strong supporting areas (1000 kg/m^2) in shear walls and concrete slabs, whereas the remaining walls have weak load bearing capabilities (1 kg/m^2),

— a width equals to fac_w=18.95 meters and a height equals to fac_h=10.64 meters,

— 7 small windows of 1.8×0.4 meters for the façade presented in Figure 2.14 on the following page and 14 small windows of 1.4×1.3 meters for the façade presented in Figure 2.15 on the next page,

— 14 big windows of 1.4×1.3 meters for the façade presented in Figure 2.14 on the following page and 7 big windows of 2.09×2.2 meters for the façade presented in Figure 2.15 on the next page,

— 1 door of 0.8×2.2 meters for façade in Figure 2.14 on the following page and 2 doors of the same size for façade in Figure 2.15 on the next page,

— 1 zone out configuration of 3.5×2.7 meters for the façade presented in Figure 2.14 on the following page.

The last façade, Figure 2.16 on the next page is a realistic instance used in our tests with :

— the same load bearing properties,

— a width equals to fac_w=10 meters and a height equals to fac_h=7.5 meters,

— 10 small windows of 0.8×0.8 meters,

— 1 big windows of 3.5×0.5 meters,

— 1 door of 1.4×2.25 meters.

The façade in Figure 2.15 on the following page is used for the evaluation of the solutions over different façade sizes, i.e., scalability tests (Sections 3.2 on page 56, 3.3 on page 75 and 4.3 on page 106). Then, a set of 20 façades, that have the same geometrical structure, are generated under specifications where frames and supporting areas are uniformly distributed over the façade surface. Also, all tests use the same lower and upper bounds of panels. In particular we use the same lower bounds for both width and height of $p_{wl} = p_{hl} = 0.9$ meters and, we use 10 meters as upper bound for one dimension and 3 meters as upper bound for the other dimension.

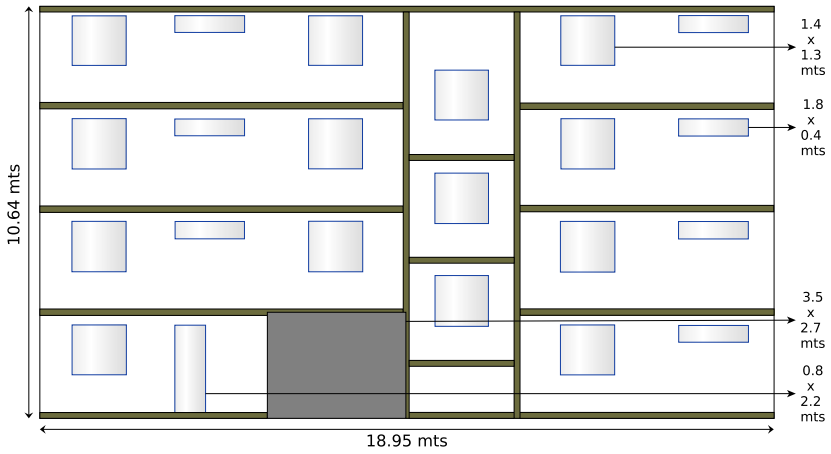


Figure 2.14 – Façade instance 1: fac_w=18.95 meters, fac_h=10.64 meters, small windows 1.8×0.45 meters, big windows 1.4×1.3 meters, door 0.8×2.25 meters, zone out configuration 3.575×2.74 meters.

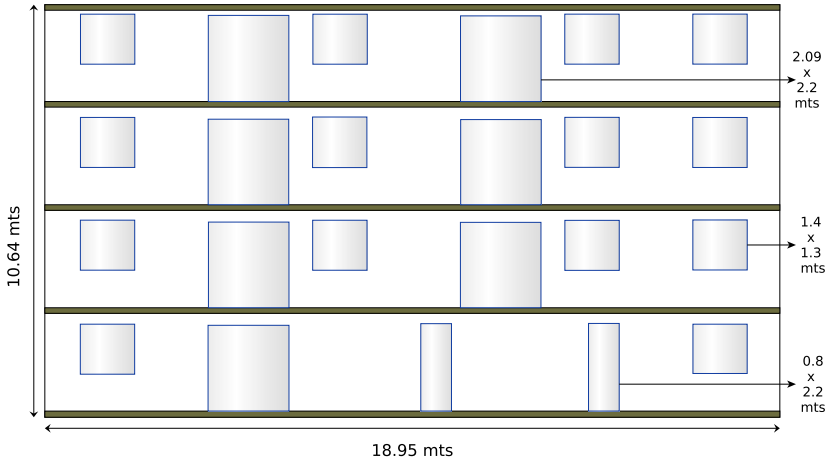


Figure 2.15 – Façade instance 2: fac_w=18.95 meters, fac_h=10.64 meters, small windows 1.4×1.3 meters, big windows 2.09×2.2 meters, doors 0.8×2.25 meters.

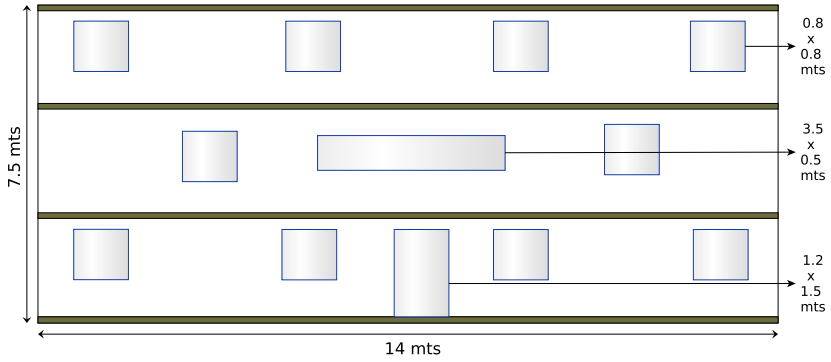


Figure 2.16 – Façade instance 3: fac_w=10 meters, fac_h=7.5 meters, small windows 1.4×1.3 meters, big window 4.9×0.97 meters, door 1.4×2.25 meters.

2.4 Digest

The industrial renovation of buildings has to cope with a multiple and diverse requirements, guidelines and constraints coming from urban design guidelines, owners' expectations, tenders' wishes, building geometry and structure, panels manufacturer and architects' skills and ability in design art. In this section we have presented the `Cutting & Packing` problem elements, limitations and requirements in regard to the industrial renovation. We have provided a detailed description of the elements taking part in a renovation along with its context and process. The requirements and limitations have been formalized as constraints under the CSP model. In addition, envelopes evaluation w.r.t. number of panels and length of junctions has been discussed and formulas representing the knowledge have been presented. The main characteristics and assumptions for tackling this problem may be summarized as follows.

— All the elements (façade, panels, frames and supporting areas) have a rectangular shape.

— Only two characteristics of the panels are fundamental in our problem: their position (p_{x0}, p_{y0}) and their size (p_w, p_h).

— Panel size lower and upper bounds may be different for each façade (depending on accessibility and impact of the spatial entities hierarchy).

— Panels are attached onto the façade by their corners.

— The design of insulating envelopes should be done in such a way that the number of panels is minimized.

— Aesthetics of envelopes is optional for the algorithms but a necessity for the architects point of view.

The goal of the renovation is to reduce energy consumption. The goal of our work is to model the `Cutting & Packing` problem under CSP and solve it using different techniques from operation research and artificial intelligence. The goal of the support system, product of our work, is to assist architects design and decision-making. As a consequence, our algorithmic solutions try to generate different envelopes solutions for façade in order to give the architect a wide spectrum of possibilities. Also, to give freedom to the architects intuitions, manual design and automatic design (although some panels may be drawn manually letting the system to complete the envelope) are identified as strong requirements.

Clarifying note #4

The core problem of the dissertation is the design of insulating envelopes. As we have seen, the design problem is treated as a two-dimensional `Cutting & Packing` problem as well as a configuration problem. Along the document, and when no confusion arises, we use *"design problem"*, *"`Cutting & Packing` problem"* and *"configuration problem"* indistinctly.

I don't know. How did Beethoven hear the Ninth Symphony in his head before he wrote it down? The brain's a pretty good computer, too, isn't it?

The Bicentennial Man and Other Stories
Isaac Asimov, 1976

3 Interactive & Heuristic-Based Design

Contents

Design is a complex process referring to the creation or elaboration of plans, blueprints, diagrams and process for objects or systems (Dinar et al., 2015). As such, design has been identified as an important conception phase in the Product LifeCycle Management of any product.

In this chapter three envelopes design algorithmic solutions are described. In the first place, we present a functional programming approach, named InDiE, supporting manual interactive design of panels, consequently insulating envelopes. As such, the functions implementing the conflict solving are discussed and exemplified. Benefits of this method are the interactive design provided to the user and the fast implementation setup for developers. In a second step, we introduce two heuristic approaches for supporting the automatic envelopes design. Here, instead of functions,

we develop two constraint-based heuristics, one based on the greedy approach, named GaLaS and one based on the cutting approach, named CaSyE. Again, constraint resolution is discussed and evaluation cases are provided.

3.1 Interactive Design: InDiE

This section presents a real-time manual interactive design of insulating envelopes, named InDiE.

3.1.1 Motivation

As previously established in Section 2.1.2 on page 35, one of the support system functional requirements is the support of manual design of panels and relevant insulating envelopes. A manual design involves creative efforts of the architect when designing each panel thus generating subjective insulating envelopes. Designing or dawning each panel is an application of the constructive approach from the layout synthesis field (Liggett, 2000). It is then the task of the support system to aid the construction process by informing or resolving constraint conflicts. Then, this design may be seen as a guided design. This means that the reactions of the support system to the user actions (interaction) must be clearly identified. Also, given the hand-eye coordination when dawning, the reactions of the system must be communicated to the user in a very short time (real-time). The first algorithmic solution for manual design is here introduced.

The solution presented in this section, that we have called InDiE for **In***teractive* **D***esign of* **I***nsulating* **E***nvelopes*, implements the constructive approach to interactively guide the architect in his/her design and see, in real-time, the impact of his/her own panels' dawning in regard to the industrial conditions. We do so by developing validation functions for each of the constraint in the model. The solution may be implemented in any functional language without relying on complex black-box tools as constraint solvers, linear programming libraries or meta-heuristics. Further, we propose a web-oriented Java-script implementation that gives the possibility to have a real-time interaction with the user (see concept in Figure 3.1 on the next page). What is more, partial envelopes design may be finished by the automated algorithms, presented latter in Chapter 5 on page 127, in a web-service setup.

3.1.2 Scheme

As explained before, our efforts focus on providing to the architects an interactive design of insulating envelopes in real-time. An interactive design refers to the system reactions to the user's actions in order to help him/her to reach her/his (design) goals (Dix et al., 2003). Interactive behavior have been widely study in many knowledge areas and industry sectors (Glover et al., 2005, Godin, 1978, Jankowski and Hachet, 2013, Shin and Ravindran, 1991). Among other things, the human interface allowing the interactive communication is one of the major study topics in computer science and informatics (Dix et al., 2003).

On the other hand we have real-time support. Real-time support refers to the capabilities of the support system to react to the user's actions in "no time". Real-time interaction is needed, mostly, when the user's actions require a response within the next 100 milliseconds (cf. Chapter 17 in Dix et al. (2003)). For instance, for activities involving hand-eye coordination, the system answers must be fast enough as to not block the activity or deteriorate the results. In the envelopes design case, immediate support must be given to the architects when designing each panel. This means that the underlying support system must execute validation or resolution algorithms in such a way that the design process is continuously fed by the system responses.

One fundamental concept functional programming is the tasks division. This concept is implemented by means of functions that are assembled together to provide a major functionality. The manual design of envelopes is addressed under the functional programming paradigm, meaning that the tasks to solve constraint conflicts have been delegated to different functions.

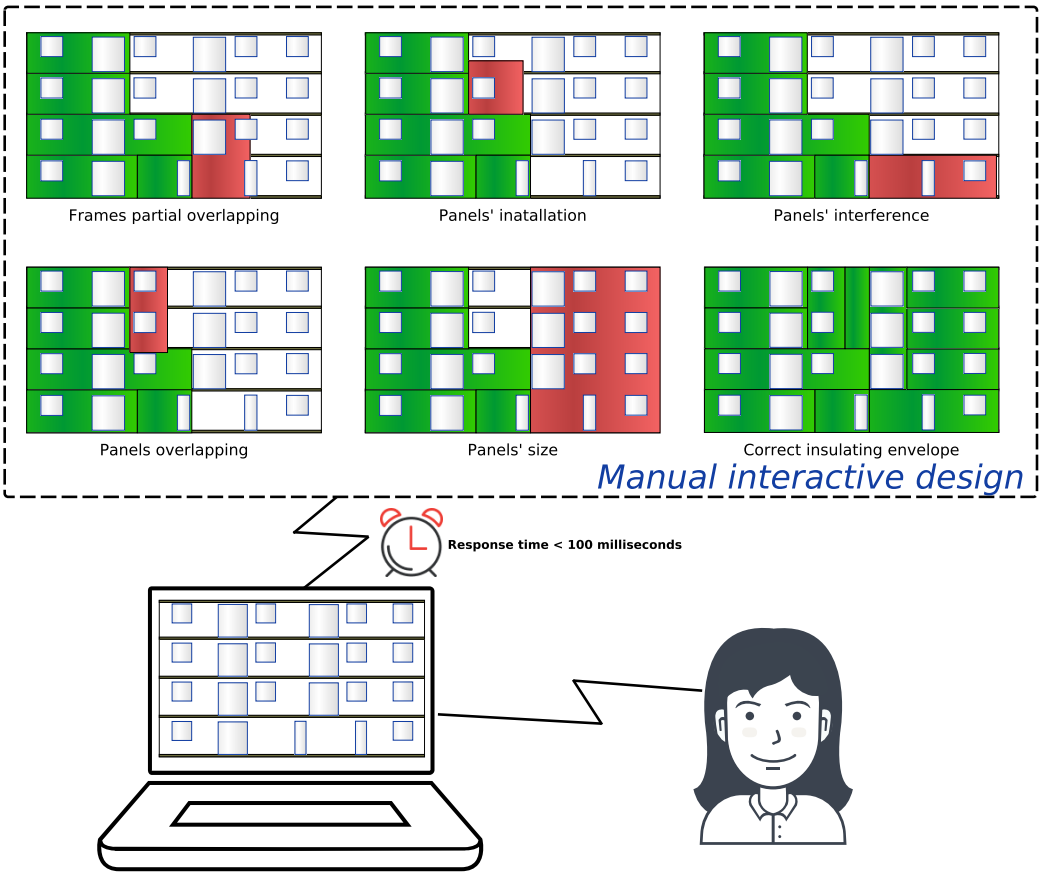


Figure 3.1 – Concept: Real-time manual interactive design.

In this case, and for convenience, several constraint up-to two constraints are linked to a unique function.

Now, an alternative for doing manual design is to provide instantaneous feedback to the architect drawing. Essentially this mean that the manual design may be *interactively* guided by the support system. When drawing a panel the system may restrict its possible size to be smaller than the upper bounds, bigger than the lower bounds, visually inform of conflicts with frames and supporting areas, and ideally, completely avoid panels overlapping or inform about such overlapping. An interactive architects manual design of envelopes would work as follows:

1. The system presents a drawing of the façade.
2. The architect draws a panel over the façade while the system informs:
   - If the size of the panel is too small or too big given panel bounds and the compatibility constraint betwen width and height,
   - if the panel is in conflict with windows and/or doors,
   - if the panel cannot be installed because its corners cannot be attached,
   - if the panel is in overlapping conflict with an already designed panel,
   - if the panel definition is blocking the definition of further panels.
3. The architects iterates step 2 until satisfaction.

As a matter of choice, and if the graphical user interface (GUI) capabilities allows it, a given drawn panel may be re-designed by the architect as part of aesthetics considerations or because as far the architect can tell current constraint conflicts may get solved. This means that in manual

design, ill definition of panels should be possible. Further, stopping ill definition of panels may be counterproductive for the architects aesthetic flair. The underlying validation algorithm informs the architect that constraints are being violated but it is the architect who decides lastly the exact size and position of panels. Among the set of alternatives to support architects manual design we have chosen the following:

— Informing about constraint conflicts is done visually: Our design choice is to set different colors for well-defined (green) panels and ill-defined ones (red).

— As an invariant, for drawing a panel, each of the previously drawn panels must be well defined.

— Re-design of a given well-define panel is possible. Colors of the panel are changed interactively depending on constraint conflicts.

— Gaps between panels are not conflictive for the result. In other words, when doing manual interactive design the area constraint (2.6 on page 38), that implies no holes, is ignored.

In consequence, the support system has two main responsibilities. Firstly, knowing that constraint conflicts exists for the selected/drawn panel. Secondly, it must inform the user about those conflicts. The former responsibility is fulfilled by the validation algorithms described in next Section. The latter responsibility is fulfilled with the GUI capabilities [1].

3.1.3 Implementation

In this section we present how functional programming is used to implement validation algorithms for the constraints presented in Section 2.2 on page 37. We give a brief behavior description for every function. The OUPUT for each function is **true** if the panel is well defined and **false** otherwise. Improved versions of these functions will be latter used in the implementation of the greedy algorithm. Examples of each function result are presented in the next Section.

Size constraint & size compatibility constraint

The size constraint (2.4 on page 38) states limits for the width and height of panels. The INPUTS of this function are a panel's current size (p_w, p_h) and, the lower $(min^{\downarrow}, min^{\uparrow})$ and upper bounds $(max^{\downarrow}, max^{\uparrow})$ for panel size. Algorithm 1, presented in Figure 3.2 on the facing page, implements the size constraint validation.

Recall that if the ratio $\frac{p_w}{p_h}$ is less than one, the panel orientation is vertical, otherwise, it is horizontal. Then, the size constraint is implemented respecting the compatibility constraint (2.5 on page 38) between width and height (i.e., orientation) and lower and upper bounds in both dimensions. Taking into account this information, the current panel width lies between $[min^{\uparrow}, max^{\uparrow}]$ whereas its height must lie between $[min^{\downarrow}, max^{\downarrow}]$, for horizontally oriented panels. Conversely, for vertically oriented panels a swap between width and height bounds is executed. This function is executed when decreasing or increasing the size of panel by means of the graphical interface.

Non-overlapping constraint

The non-overlapping constraint (2.7 on page 39), implemented as illustrates the Algorithm 2, introduced in Figure 3.3 on the next page, states that panels cannot overlap in at least one dimension. The INPUTS of this function are the position (p_{x0}, p_{y0}) and assigned size (p_w, p_h) of the panel currently being designed, and a list of already-defined panels (adp). Note that a given already-defined panel can be in fact a zone out of the configuration (zoc); these zones are of rectangular shape and thus their non-overlapping is managed the same as if they were a panels.

1. It is worth mentioning that the support system GUI has been implemented by the engineer Philippe Chantry when he was working in Mines d'Albi. No details about the GUI capabilities are provided as they are of marginal interest to our work.

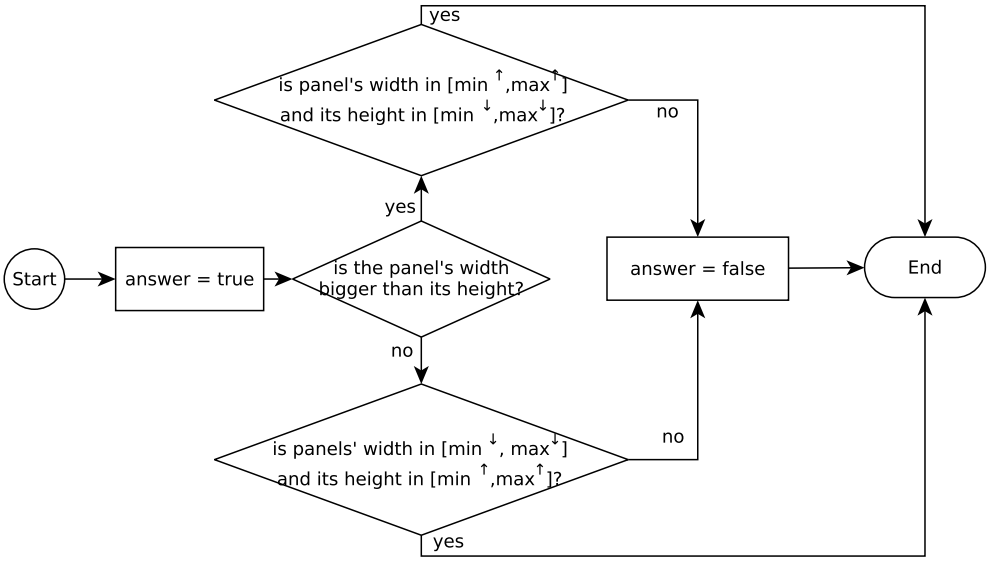


Figure 3.2 – Algorithm 1: Validation of size constraint.
***Inputs*:** Current panel size ($p_w,\ p_h$), panel size bounds ($max^{\uparrow}, min^{\uparrow}, max^{\downarrow}, min^{\downarrow}$).
***Output*:** Whether size constraint is being violated (**True/false**).

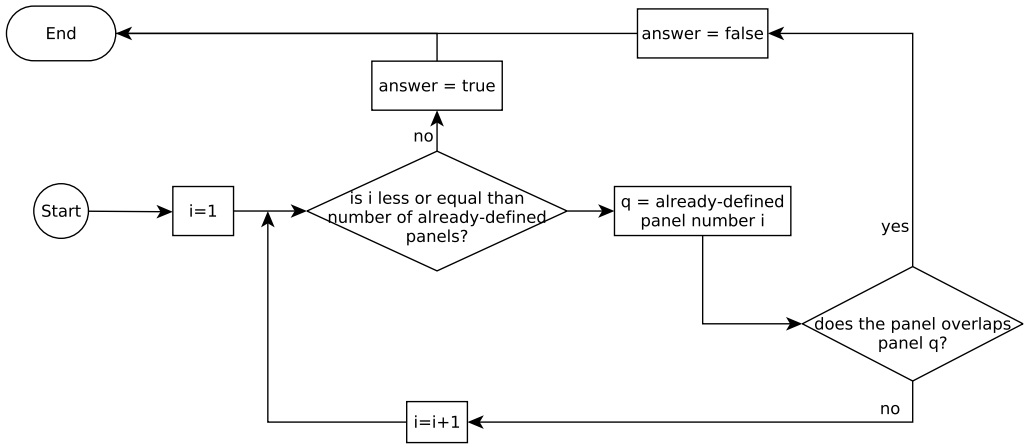


Figure 3.3 – Algorithm 2: Validation for non-overlapping constraint.
***Inputs*:** Current panel position (p_{x0}, p_{y0}), current panel size (p_w, p_h), list of already-defined panels (adp).
***Output*:** Whether non-overlapping constraint is being violated (**True/false**).

To verify panels overlapping, the current designed panel must be checked against every already-defined panel (adp). An overlapping exists between panels p and q, if the projection of their widths and heights overlaps. If an overlapping exists, the color of the current designed panel is changed to red while the already defined panel remains green. This function is executed in two cases. First, when decreasing or increasing the size of panel by means of the graphical interface. Second, when the defined panel is moved (re-designed) around the façade surface.

Frames and interference constraint

Algorithm 3, in Figure 3.4, checks at the same time whether an assigned panel violates interference constraint (2.11 on page 41) and frames constraint (2.8 on page 39). The INPUT of this function are the bottom-left corner (p_{x0}, p_{y0}), width (p_w) and height (p_h) of an assigned panel, the range of width and height (for instance, $[min^{\uparrow}, max^{\uparrow}]$ and $[min^{\downarrow}, max^{\downarrow}]$), the minimal distance between panels and frames borders (d) and the list of frames over the façade (F). The following flowchart depicts the behavior when checking conflicts of frames and interference in the vertical axis. A similar process is executed for the horizontal axis, i.e., for the left and right edges of the façade.

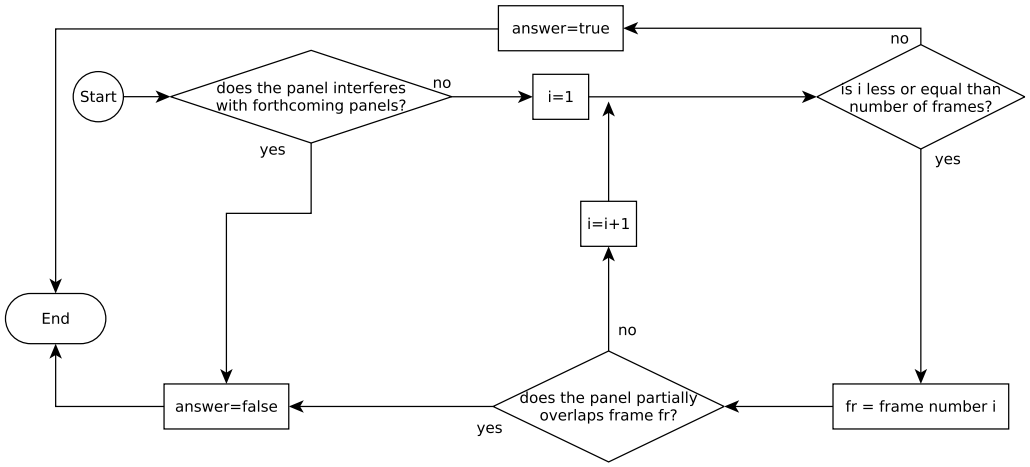


Figure 3.4 – Algorithm 3: Validation algorithm for frames and interference constraints.
***Inputs*:** Current panel position (p_{x0}, p_{y0}), current panel size (p_w, p_h), panel size bounds $([min^{\uparrow}, max^{\uparrow}])$, list of frames over façade (F), minimum distance between panel border and frames borders (d), façade size (fac_w, fac_h).
***Output*:** Whether frames or interference constraints are being violated (**True**/**False**).

The first step of Algorithm 3 is to validate the interference constraint (2.11 on page 41) by checking if there is enough space for forthcoming panels. If the panel does not interfere with forthcoming panels, the algorithm checks partial overlapping of the panel against all frames. In the case there is a conflict, the algorithm terminates informing that a conflict exists. This function is executed every time a new panel is drawn or re-designed.

Installation and weight constraints

The last function ties up a single panel definition and implements the validation of the installation (2.9 on page 40) and weight constraint (2.10 on page 40). The INPUT of this function are the bottom-left corner (p_{x0}, p_{y0}), width (p_w) and height (p_h) of an assigned panel and the list of supporting areas (sa). Recall that in order to attach panels, their corners must be located on supporting areas (sa). and that supporting areas at the bottom corners must be strong enough to support the weight of the panel.

In essence, the Algorithm 4, shown in Figure 3.5 on the facing page, checks that every corner match a supporting area. To do so, it tests corners against every supporting area, finishing the executing if at least one corner is not included in any supporting area. The panels' weight and the supporting area load bearing capabilities are checked after the corners check. Here, it is worth noticing that supporting areas do not overlap. This means that a given corner can be located in at most one supporting area.

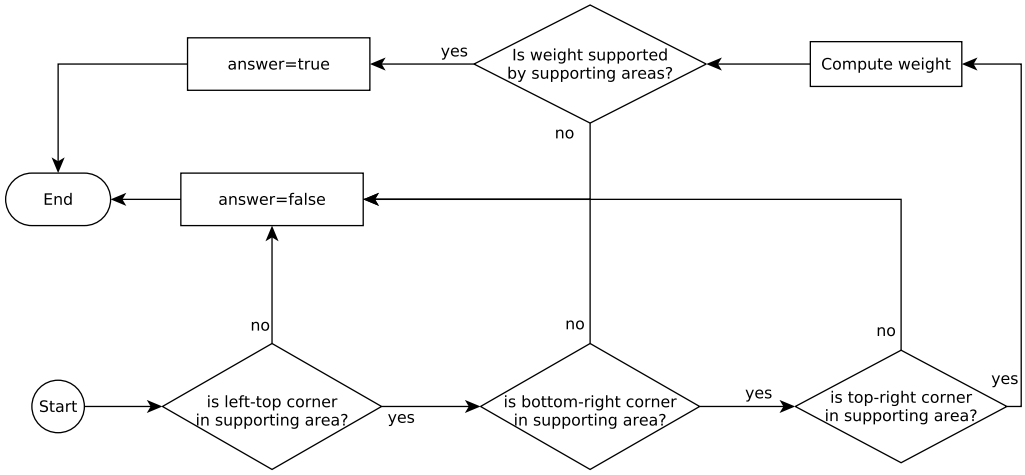


Figure 3.5 – Algorithm 4: Validation algorithm for installation and weight constraints.
***Inputs*:** Current panel position (p_{x0}, p_{y0}), current panel size (p_w, p_h), list of rectangular supporting areas (sa).
***Output*:** Whether weight or installation constraints are being violated (**True**/**False**).

The remaining constraints, i.e., area and symmetry breaking constraints, are overlooked as the manual design does not involve them.

3.1.4 Evaluation Cases

Examples of the previous validation algorithms are presented in this section. To do that, we use the façade illustrations introduced in Section 2.4 on page 45. Panels bounds are $[1,10]$ meters for one dimension and $[1,2]$ meters for the other dimension.

— Figure 3.6 presents two well-defined panels in green and one ill-defined panel in red. The figure illustrates the fact that the size of a panel is constrained to the inputed upper bounds.

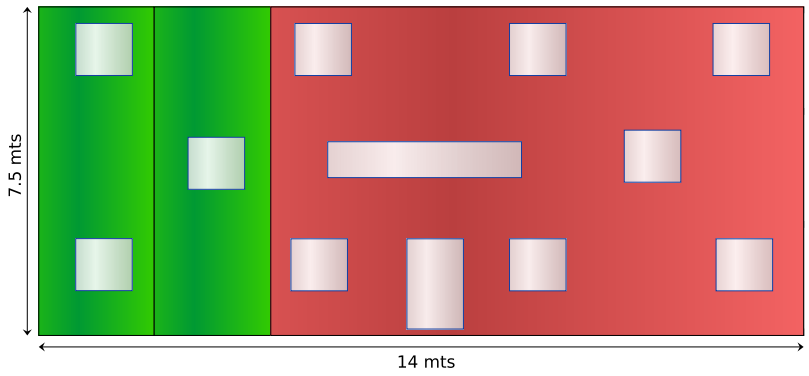


Figure 3.6 – Illustration of interactive support for size constraint constraint.

— Figure 3.7 on the next page presents six well-defined panels in green and one ill-defined panel in red. The figure illustrates the behavior of the support system when the designed

panel enters in conflicts with frames. The red panel is partially overlapping one door and one window on the façade.

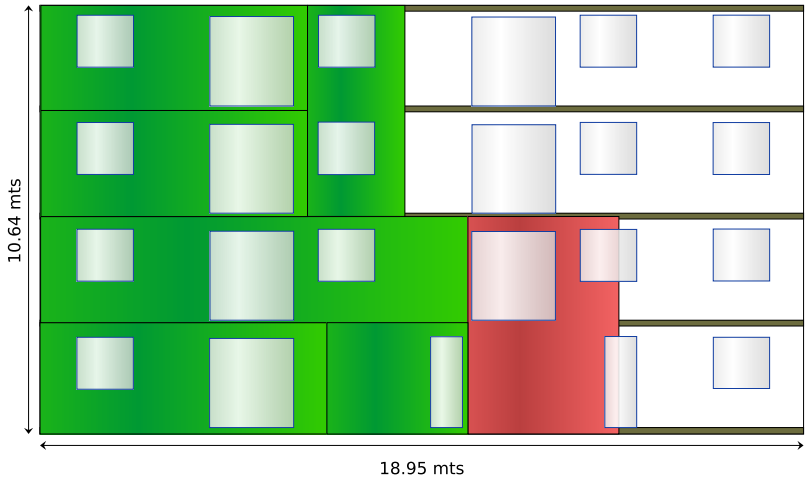


Figure 3.7 – Illustration of interactive support for frames constraint.

— Figure 3.8 presents four well-defined panels in green and one ill-defined panel in red. The figure illustrates the behavior of the support system when the designed panel cannot be installed over the façade. The red panel does not have its top-right corner in a supporting area.

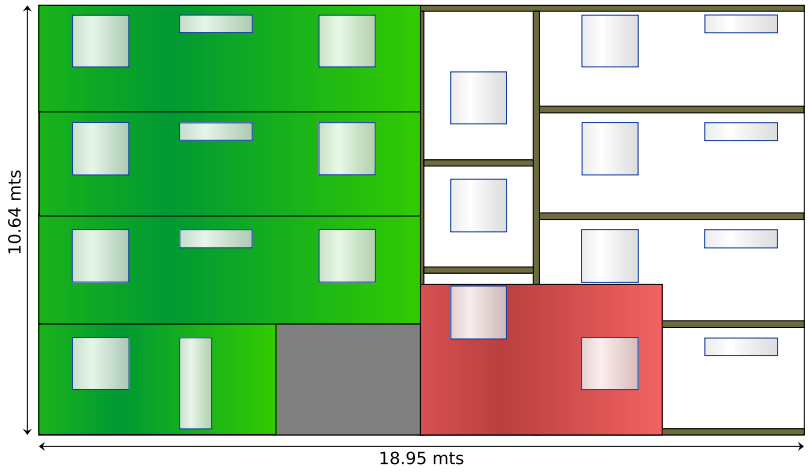


Figure 3.8 – Illustration of interactive support for installation constraint.

— Figure 3.9 on the next page presents four well-defined panels in green and one ill-defined panel in red. The figure illustrates the behavior of the support system when the designed panel interferes with the further definition of panels. The red panel cannot be designed there as the remaining space (at its left) is smaller than the panel width lower bound; no panel can be designed in that space.

— Figure 3.10 on the facing page presents six well-defined panels in green and one ill-defined panel in red. The figure illustrates the behavior of the support system when the designed panel overlaps one or more already defined panels.

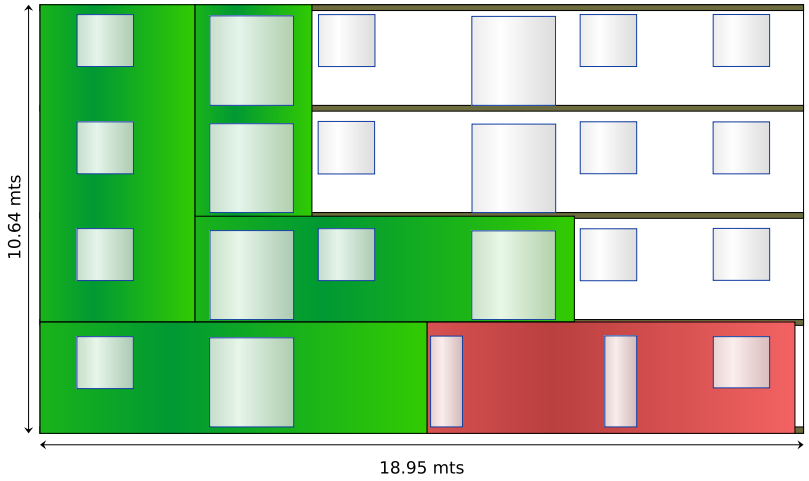


Figure 3.9 – Illustration of interactive support for interference constraint.

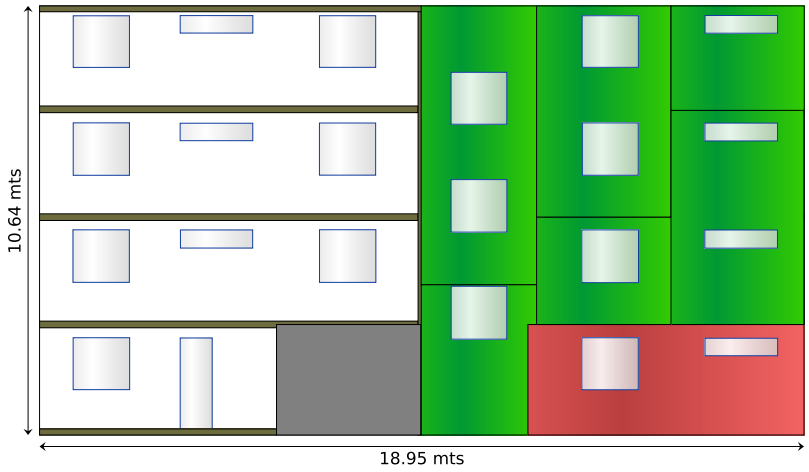


Figure 3.10 – Illustration of interactive support for panels overlapping constraint.

3.1.5 Discussion

In this section we have shown how the manual design may be made interactive with a visual communication with the user. Then, we have discussed the fact that functional programming allows the design support to be executed less than 100 milliseconds thus providing real-time interaction with the user. Finally, we have briefly presented four of the key functions to support the design problem.

At the beginning of our research, we have considered that constraint satisfaction techniques were appropriated for addressing the envelopes design problem. Our results have shown us right when assisting the automatic and automatic design of envelopes (studied in detail in forthcoming sections). Further, the framework of constraint satisfaction is known to address support interactive design (Gelle and Weigel, 1996). Nevertheless, for providing a real-time feedback we have rely on the simple yet powerful concept of task division instead of OR and AI techniques. Doing this we bypass the time expend in having an output that potentially exceeds the real-time requirements.

3.2 Greedy Design: GaLaS

This section presents a greedy approach for the automatic design of insulating envelopes, named GaLaS. Part of the content of this section has been presented previously in Barco et al. (2014).

3.2.1 Motivation

As previously established in the requirement Section 2.1.2 on page 35, one of the support system functional requirements is the support of automatic design of panels (respectively insulating envelopes). The automatic design refers to the capabilities of the system to provide compliant design solutions with the minimal human intervention. But, the fully automation of the building renovation process is not possible given that input (renovation specification) and output (envelopes) depends on human analysis, human manufacturing and even the human tuning of envelopes (for aesthetics reasons). Nevertheless, automation of the envelopes design is desirable to avoid non-compliant solutions, eliminate by hand configuration errors and to unload charges to the architects (to focus more on aesthetics properties). Besides, the algorithmic solution presented in this section, as well as the automatic design solutions to be discussed in Sections 3.3 on page 75 and 4.3 on page 106, allows a semi-automatic design: The architect draws well-designed panels on the façade and inputs the partial envelope the algorithmic solution.

The idea behind the GaLaS algorithm is to be as fast and simple as possible. The greedy heuristic strategy is based on very simple and powerful concepts. In essence, a greedy algorithm works by dividing the problem in steps. At each step the algorithm makes a choice that is assumed to be the best in the current state. At the core this is a *local optimal decision*. Then, by making the best local decision the algorithm expects to arrive to an optimal solution. However, in general, greedy algorithms do not achieve the optimal but close to optimal solutions. In consequence, the algorithm here developed is *"blind"* in the sense than it does not take into account the global façade geometry and structure, and it does not consider future states (except for the implementation of the interference constraint— 2.11 on page 41). One of its advantages is to generate solutions in a short computational time thus satisfying the execution requirements of the project (see 2.1.2 on page 35).

Now, according to Cormen et al. (2009a), to implement a greedy algorithm it is first necessary to determine the optimal substructure of the problem. Here, a substructure to the two-dimensional Cutting & Packing problem of designing optimal insulating envelopes is to design one optimal panel. Due that the minimization of number of panels is an objective, an optimal panel is the one that covers the largest area surface while respecting the set of constraints, including panel orientation soft or hard constraint, imposed by the problem domain. Thus, constraint conflicts are as well resolved using local optimal choices in regard to the panel surface. The algorithm decreases the failures possibilities by executing backtracking if a local decision cannot be made due to constraint conflicts.

Our algorithm, that we have called GaLaS for **G***reedy* **a***lgorithm for* **La***yout* **S***ynthesis*, follows an intuitive constructive approach (Liggett, 2000), i.e., optimally place one panel at a time without considering forthcoming panels (see greedy concept in Figure 3.11 on the facing page). The application of the greedy approach to Cutting & Packing problems is also known as *on-line packing* (Csirik and Woeginger, 1998, Jylänki, 2010). The on-line packing assumes that items arrive to the packing process on-the-fly, i.e., in runtime. This does not hold completely true in our case as there is no panels to be packed one by one. Instead, our on-line packing *"created"* panels on-the-fly in regard to the local façade geometry. This first algorithmic solution for automatic design is presented in this section.

3.2.2 Scheme

The main characteristics of greedy solutions is the local decision-making. This is, no matter in which stage the process is, the next step is computed with respect to present state and no

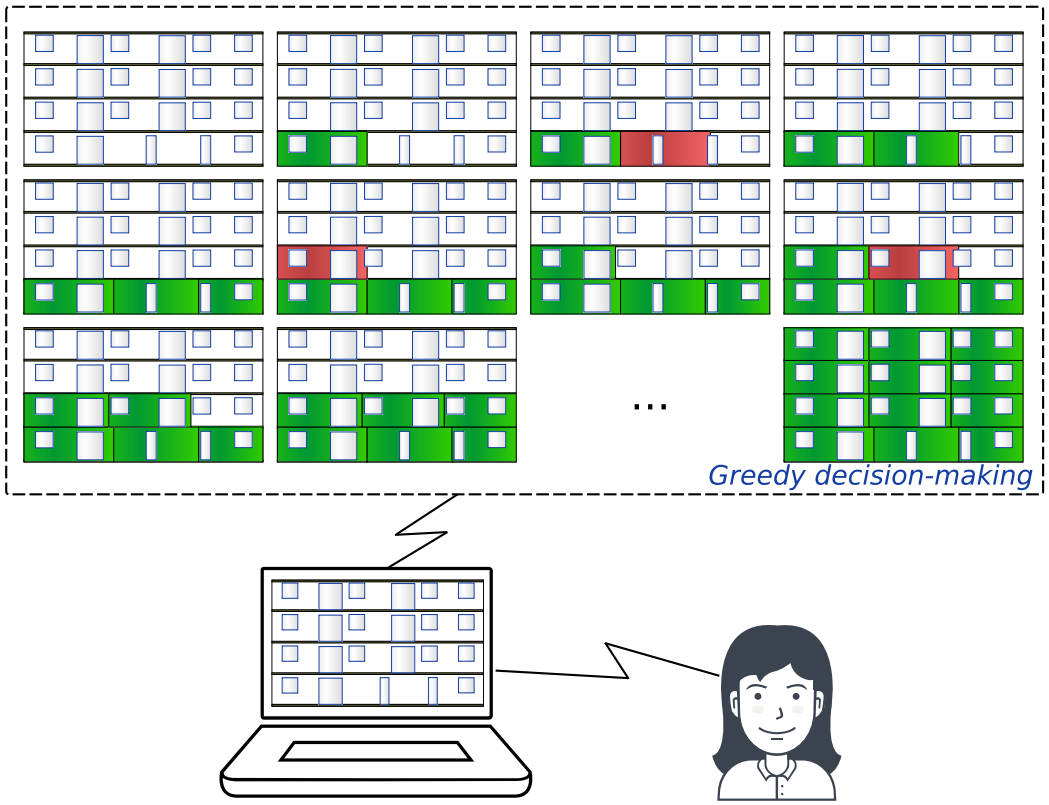


Figure 3.11 – Concept: Local decision-making for greedy design.

future states are taken into account. Thus, for developing a greedy algorithm that generates valid insulating envelope, several local decisions for designing panels must be isolated. These local decisions concern mainly the definition of a single panel, i.e., its origin point (p_{x0}, p_{y0}), its size (width p_w and height p_h).

This local decision-making must be done by taking into account the (constraint) knowledge inherit from the industrial scenario while trying to reduce the number of panels composing the resulting envelope. Before explaining the general scheme of the greedy algorithm, it is worth to remember that each panel must respect the compatibility constraint between width and height, constraint that imposes a given orientation for the panel (in the support system—Chapter 5 on page 127—the architect may choose his/her preferred orientation as a personalization feature). The local decisions concerning the design of a given panel are:

1. *How to choose an origin point to place a the panel?* The origin point (p_{x0}, p_{y0}) in which the panel is designed has a big impact on the envelope design. Considering the origin point of panels, Figure 3.12 on the following page shows two decision possibilities over a surface. Literal (a) illustrates the decision: Choose the lowest and leftmost point. Then, literal (b) illustrates the decision: the rightmost and lowest point. Note how these envelopes differ by changing the local decision of origin point. In our case, we have chosen to start from the bottom-left corner of the façade.

   In order to avoid holes, the origin point for the currently-designed panel is always adjacent to one or more panels (except for the first panel) or to zones out of configuration. The origin of the panel is set either at the bottom-right corner of the already-defined panel (illustrated in Figure 3.13 on the next page bottom-right point) or at the top-left corner of the already-defined panel (illustrated in Figure 3.13 on the following page top-left point).

   In our case, given that the panel has not defined size yet, we have chosen the bottom-

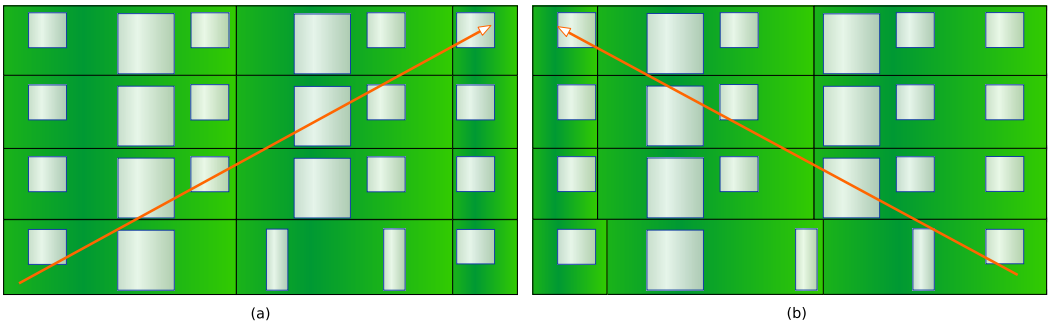


Figure 3.12 – Criteria: (a) Point bottom→left: (b) Point right→top

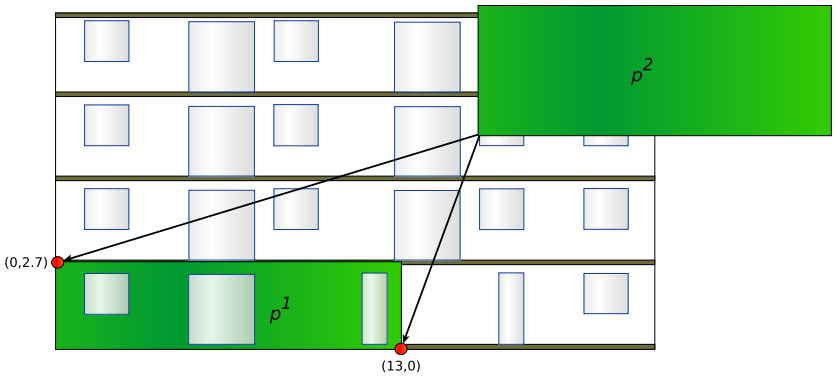


Figure 3.13 – Possible origin points for currently-designed panel.

first approach. In this approach, at each step the algorithm chooses as $(p_{x0},\ p_{y0})$ the first available bottom-left point of the façade with priority to the bottom. This option presents two advantages. Firstly, the panel may be extended up and to the right avoiding overlapping with the panels already designed by the process. Already-defined panels, such as user-designed panels and zocs, must be dealt with if they are at the right or top of the currently designed panel. And secondly, this allows to locate the largest panels at the bottom of the façade, which is a good choice for the on-site assembly process.

2. *How to choose the size assigned to a the panel?* The initial size assigned to any given panel is the maximum allowed, for both the width and the height, and respecting the compatibility constraint. Then, for a given execution, the algorithm sets either $p_w > p_h$ or $p_w < p_h$ and continues applying local decision respecting such ratio (orientation). If further local decisions make the panel enter in constraint conflicts, then the current ratio between p_w and p_h is changed, i.e., changes orientation in an attempt to prevent no solution setup. Now, the local decision of setting the maximum allowed size for both dimensions is done in order to reduce the number of panels and then have solutions close to optimal. In case of constraint conflicts using the maximum size, then the panel size is reduced in one or both dimensions. Trying to respect the current panel's orientation, the width is reduced if $p_w < p_h$, otherwise the height.
3. *How to solve panels-frames and panels-panels conflicts?* If a panel, with defined origin point and size, enters in conflict with a given frame (partial overlapping) or with already-defined panels or zocs (overlapping), then its width or its height are reduced in such a way that the conflict gets solved. To do so another local decision is made: It reduces the width if vertical orientation and the height if horizontal orientation (in order to keep consistency).

4. *How to choose where to attach panels?* If the panel's corner are not included in supporting areas, then reduce the width or height until a supporting area is found. The decision whether to reduce the width or the height is made taking into account the panel's current orientation and largest surface.

Note that defining different local decisions generates different insulating envelopes. Conversely, different envelopes can be drawn by choosing different panel's size or the way requirements are satisfied. For example, designing panels following an spiral pattern (Huang and Chen, 2008), i.e., following the perimeter of the façade, from the exterior to the interior.

3.2.3 Implementation

Bearing in mind the above scheme, the algorithm GaLaS that solves the packing of panels in a greedy fashion has been developed. It makes local decisions for positioning panels following the first-fit bottom-left approach. Also, it performs backtracking when the positioning of a panel is not possible due to requirements conflicts. To understand the algorithm internals, we present the subroutines that address most of the constraints in the model and that are used by the greedy algorithm. They are divided in three subroutines; one for frames and interference constraints, a second one for dealing with the installation and weight constraints and, one dedicated to solve panels overlapping with already-defined panels or zocs.

Subroutines

First, the Algorithm 5 presented in Figure 3.14 on the next page, called *Frames & Interference Resolution*, is introduced. This algorithm has two responsibilities. It checks whether an assigned panel violates the frames constraint (2.8 on page 39) and at the same time, the interference constraint (2.11 on page 41). The INPUTS of this function are therefore the current panel position (p_{x0}, p_{y0}), the current panel size $(p_w,\ p_h)$, the list of frames (F) of the façade, the minimum distance between panel borders and frames borders (d), the panel orientation (p_o) and the façade size $(fac_w,\ fac_h)$. In the case a conflict exists the algorithm returns a new value for the size of panels. The OUTPUT of this function is a new consistent size for the current panel. Note that the algorithm describes only the constraint conflicts using the vertical axis (y-axis) but it is executed for both axis sequentially.

The first step of this algorithm is to leave enough space for the next panel (interference constraint—2.11 on page 41) if needed. Then, it uses a stack to perform an ordered check of all frames covered by the panel in the axis. In case there is a conflict, the algorithm proceeds by adjusting one of the coordinates of the end point, i.e., reducing its size. In the case the frame is completely covered by the panel, it marks the panel as covered in the given axis. Once algorithm has been executed for both axis, every covered frame has been marked twice. The final step of the algorithm is to discard all frames successfully covered by the panel in order to avoid forthcoming checks. The new size for the panel is returned: A size which is consistent with all frames and forthcoming panels. Note here that the new size is consistent with respect to frames but not with respect to the panel size bounds. It is responsibility of the GaLaS algorithm to check the resulting panel size against the size bounds.

Algorithm 6 presented in Figure 3.15 on the next page, called *Weight Resolution*, introduces the resolution of the weight constraint (2.10 on page 40) in a greedy fashion. Note that any previously-defined panel must have its corners in supporting area in order to be properly attached. The origin point is therefore always adjacent to one or more panels or to zones out of configuration (expect for the first panel, which is always at the bottom-left corner of the façade). The INPUTS of this function are therefore the current panel position (p_{x0}, p_{y0}), the current panel size $(p_w,\ p_h)$, the panel's orientation (p_o) and the list of supporting areas (sa) of the façade. In the case a conflict exists the algorithm returns a new value for the size of panels. The OUTPUT of this function is a new consistent size for the current panel.

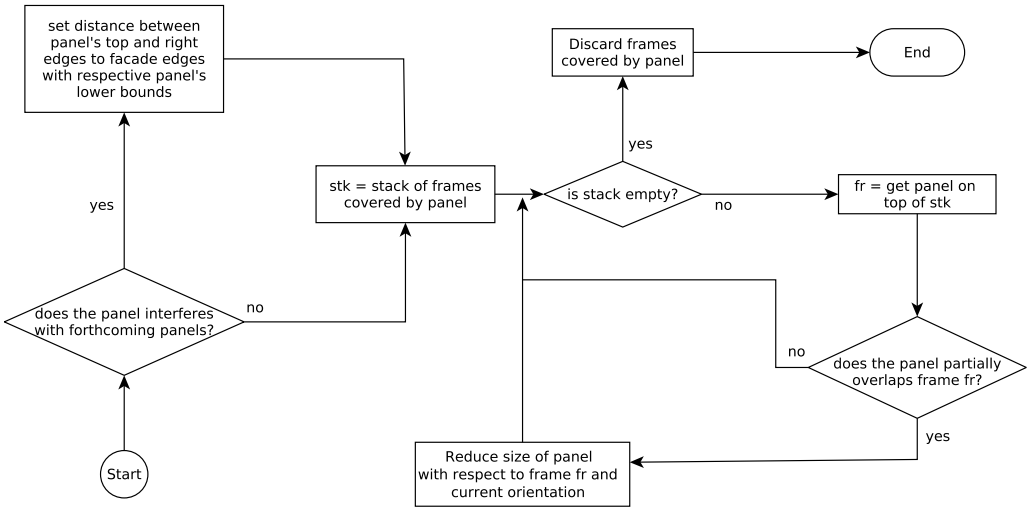


Figure 3.14 – Algorithm 5: Frames & interference resolution.
***Inputs*:** Current panel position (p_{x0}, p_{y0}), current panel size $(p_w,\ p_h)$, list of frames (F), minimum distance between panel borders and frames borders (d), panel orientation (p_o), façade size $(fac_w,\ fac_h)$.
***Output*:** New size of panel (p_w, p_h).

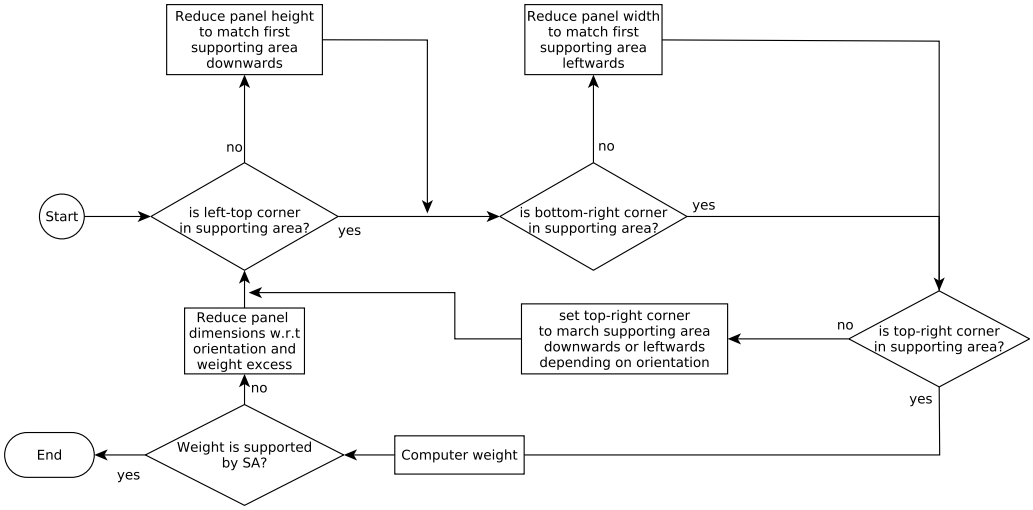


Figure 3.15 – Algorithm 6: Weight resolution.
***Inputs*:** Current panel position (p_{x0}, p_{y0}), current panel size (p_w, p_h), list of supporting areas (sa), current panel orientation (p_o).
***Output*:** New panel size (p_w, p_h).

Respecting our scheme choices, priority is given to the origin point at the bottom. In consequence, as an invariant, the origin point for every panel already belongs to a supporting area. For the first panel at the bottom-left corner of the façade this holds true as the existence of a supporting area in $(0,0)$ is mandatory to cover all the façade surface. What remains is then to choose a valid point for the remaining three corners.

In essence, the Algorithm 6 looks for valid attaching points for panel's corners. To do so, it takes the current panel definition (position and size) and checks if its corners match supporting

areas. If one or more corners fall out supporting areas it proceeds by reducing width and/or height until the next supporting area is found. The process is executed iteratively until all corners have been placed in valid attaching points. After computing the weight of the panel, it checks whether it is successful supported by supporting areas. In the case it is not possible, it reduces panels' size in proportion with the overweight, computes the weight and checks again with supported areas. As well as Algorithm 5, the resulting panel size may not be consistent with the panel size bounds. This constraint consistency is checked by the GaLaS algorithm.

The last subroutine solves overlapping conflicts between the currently-designed panel and the already-defined panels or zocs (non-overlapping constraint—2.7 on page 39). The INPUTS of this function are therefore the current panel position (p_{x0}, p_{y0}), the current panel size $(p_w,\ p_h)$, the panel's orientation (p_o) and the list of already-defined panels (adp) of the façade. In the case a conflict exists the algorithm returns a new value for the size of panels. The OUTPUT of this function is a new consistent size for the current panel.

Given that the panels designed by the automatic solution are all at the left and bottom of the currently-designed panel, the overlapping resolution is made with respect to the panels or zocs that have been manually designed by the architect. No assumption about the position of these panels or zocs is made and thus the need to solve overlapping conflicts. Algorithm 7 presented in Figure 3.16, called *Non-overlapping Resolution*, introduces the flowchart of this resolution.

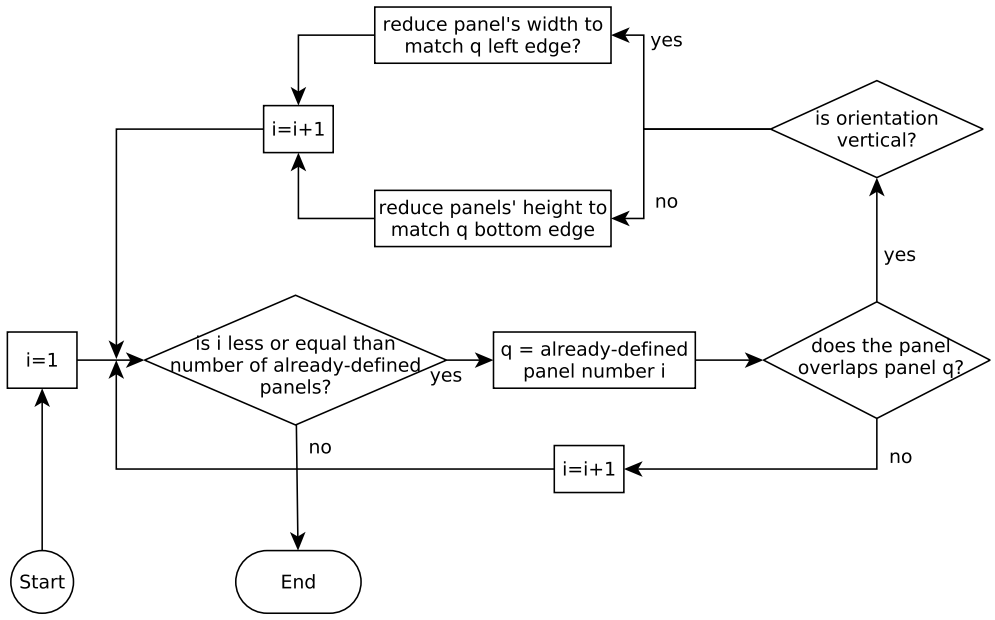


Figure 3.16 – Algorithm 7: Non-overlapping resolution.
***Inputs*:** Current panel position (p_{x0}, p_{y0}), current panel size (p_w, p_h), list of already-defined panels (adp), current panel orientation (p_o).
***Output*:** New panel size (p_w, p_h).

Essentially, the Algorithm 7 checks for overlapping in both axis between the currently-designed panel and each of the already-defined ones. If an overlapping in both axis exists, then one of the dimensions must be reduced. The chosen dimension is the one that respects the current ratio between width and height, i.e., the current orientation. Reducing a dimension that allows the largest panel area is also possible. However, the algorithm takes local decisions aiming as well to respect the architect preferences.

Greedy Solution

Algorithms 5 in Figure 3.14 on page 60, 6 in Figure 3.15 on page 60 and 7 in Figure 3.16 on the previous page, are used in the Algorithm 8 presented in Figure 3.17 on the facing page, called `GaLaS`. It is implemented for designing panels and re-designing them (backtracking) when the design of a given panel fails. The INPUTS of the algorithm are:

1. Façade size (fac_w, fac_h),
2. panels' width lower and upper bounds (p_{wl}, p_{wu}),
3. panels' height lower and upper bounds (p_{hl}, p_{hu}),
4. list of frames (F),
5. list of supporting areas (sa),
6. list of alread-defined panels (adp) which can be empty,
7. initial origin points (op, with $op = [(0,0)]$ if no already-defined panels nor zocs exist),
8. architects preferred orientation (p_o),
9. minimum distance between panel borders and frame borders (d),
10. an empty list where designed panels will be stored ($solution$) and,
11. a boolean variable indicating for the current panel if its size (width and height) has already been swapped due to backtracking.

The OUTPUT of the algorithm is either a set of envelopes or an empty set (meaning no solution has been found).

Before entering in the algorithms details, it is worth mentioning that at the beginning of the process exists a list of possible origin points. Initially, when there is no already-defined panels, the list contains only the bottom-left corner of the façade, i.e., $op = [(0,0)]$. This list op is dynamically changed when designing panels. For instance, the panel p^1 in Figure 3.13 on page 58, removes the origin point $(0,0)$ and adds two new points, namely, $(13,0)$ and $(0,2.7)$. Any new panel fetches its origin point from this list and adds new origin points (except for the last panel that matches the top and left edges of the façade). The algorithm `GaLaS` behaves as follows.

The stop condition is the absence of origin points or the failing at positioning the first panel over the façade, meaning that the façade cannot be covered at all by such an insulated envelope.

The algorithm begins by retrieving an available origin point for the panel p^i, the first panel p^1 been (0,0), and finding an end point given panel upper bounds and current orientation. Then it checks for panel overlapping against already-defined panels using Algorithm 7 (Figure 3.16 on the preceding page).

Afterwards, it checks the validity of the panel with respect to the façade frames using the Algorithm 5 (Figure 3.14 on page 60) which if necessary, generates a new valid end point for the panel. If the size of the panel p^i violates size limitations then it fails at positioning the panel. Upon failure, the current orientation is swapped and the design process starts again for panel p^i. Then, if no solution is found for p^i, the previous panel p^{i-1} is re-designed with size 10%[2] less and the design process starts again for panel p^{i-1}. This reduction is done in both dimensions.

If a valid end point has been set for panel p^i, the algorithm checks if the panel corners match supporting areas, otherwise reduces its size, using the Algorithm 6 (Figure 3.15 on page 60). If the panel p^i size has been modified by Algorithm 6, then it checks again the validity of the panel again frames by Algorithm 5. These two checks, panels vs frames and panels vs supporting areas, run sequentially until a consistent panel is designed. Once the panel p^i is designed respecting all the constraints, it proceeds by computing new origin points, using the top-left and bottom-right corners, and adding the next panel iteratively. If no more origin points are left it means that no more panels can be placed as the entire façade surface is already covered. Thus, a solution have been found.

2. The proportion of the size to be reduce may be set in a new parameter. We use 10% because it is a good approximated to the number of times a panel size may be reduced (maximum 10 times).

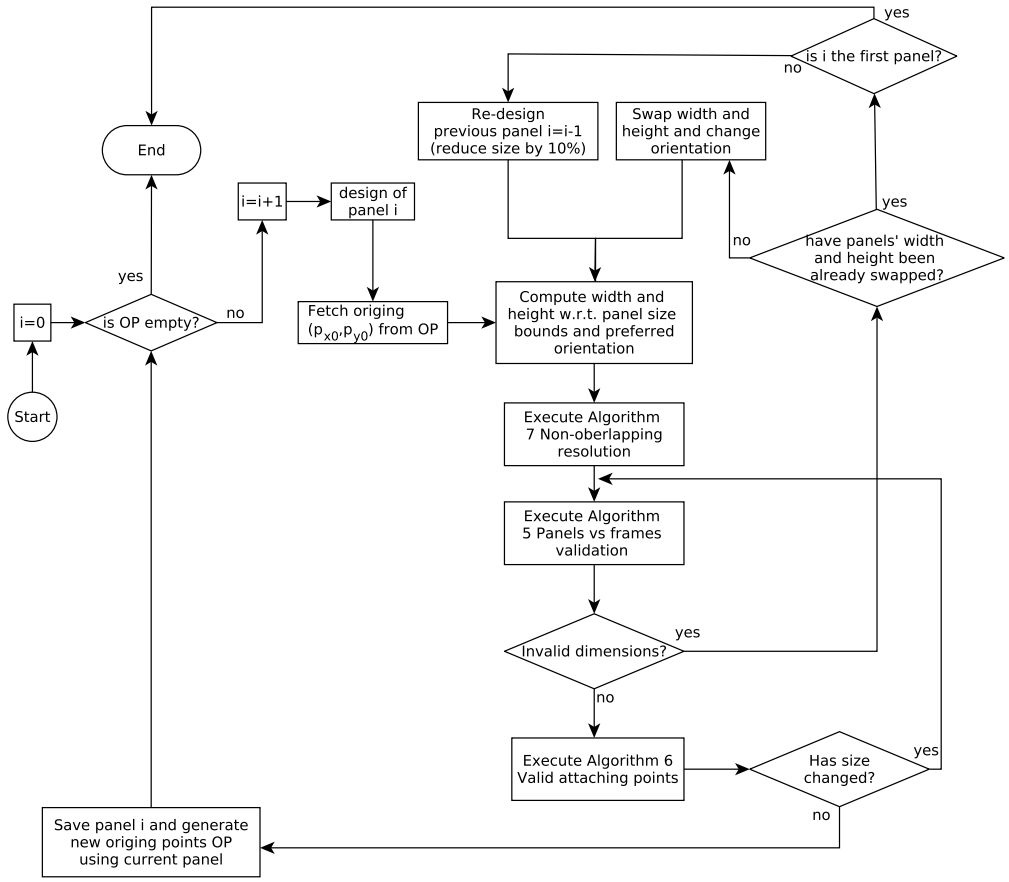


Figure 3.17 – Algorithm 8: GaLaS algorithm.
***Inputs*:** Panel size bounds ($p_{wl}, p_{wu}, p_{hl}, p_{hu}$), list of frames ($F$), list of supporting areas (sa), minimum distance between panel borders and frames borders (d), architects preferred orientation (p_o), façade size (fac_w, fac_h), list of origin points (OP=[(0,0)]), solution list=[].
***Output*:** Envelope solution (panels coordinates and size).

Solution diversity

Given that each façade has many different valid solutions and some valid solutions using the minimum number of panels, we need a way to generate the biggest possible set of insulating envelopes and take from them the best ones with respect to the number of panels. Therefore, we use a basic search strategy to explore the solution space and enumerate all solutions found. Our built search tree is binary: Left branch imposes the ratio $p_w > p_h$ for the panel's size whereas the right branch imposes the ratio $p_w < p_h$. This means that a given left node in the tree sets the current panel orientation as horizontal and the right node sets the current orientation as vertical. Figure 3.18 on the next page illustrates this search behavior.

For each node are executed the same local decisions discussed before but with different width and height ratio. If a solution using only horizontal panels exists, this solution is found at the left most leaf of the tree. Conversely, the right most leaf contains a solution with only vertical panels. Plausibly, these two leaves have a smaller height in the tree than those leaves with envelopes with panels in both orientations.

After all solutions in the search tree have been found, we proceed by ranking them according with a given criterion. For instance:

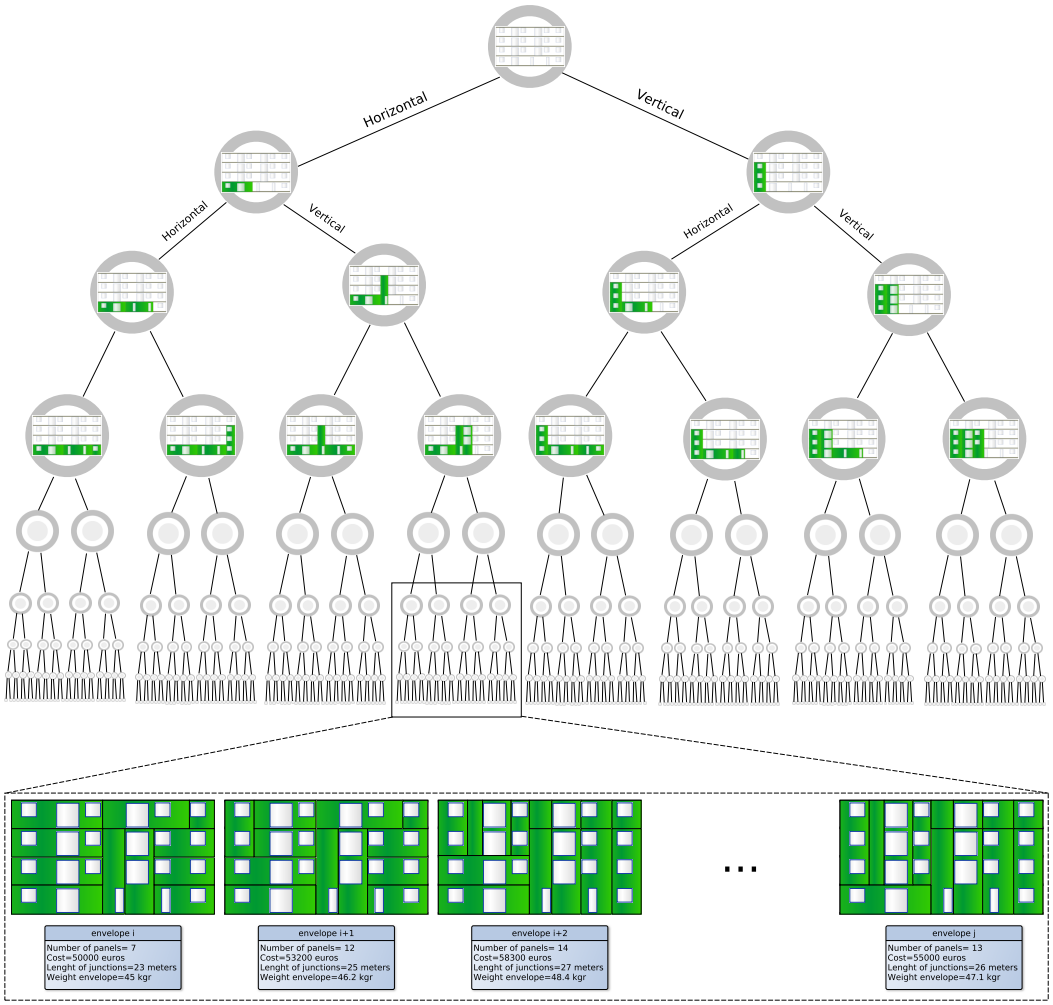


Figure 3.18 – Solution diversity using binary trees

— The number of panels.
— Length of junctions.
— Envelopes with panels in a given orientation.
— Envelopes with a given ratio between number of panels vertical oriented and number of panels vertically oriented.
— ...

Further, the set of solutions contain all numerical information about panels (size, position, etc). Ergo, the ranking in which solutions are ordered and presented to the user is only an ad hoc procedure that is executed at the end of the design process. Then, the raking of envelopes may be built with any criterion that depends on the insulating envelopes numerical model. Additionally, the solutions presented to the user maybe only a subset of all solutions found. Although, showing all acceptable envelope solutions is important in order to provide the user the possibility to choose her/his preferred envelope among several valid options. Indeed, it is the case that a given solution is adequate with respect to the thermal renovation goals but, according to architects request and aesthetic considerations, another solution may be implemented.

3.2.4 Evaluation Cases

In this section are presented some scalability tests and examples of envelopes generated by the greedy solution GaLaS.

Scalability

As stated in Section 2.3 on page 43, panel size upper bounds are 10 meter for one dimension and 3 meters for the other one. Consequently, for the left branch of the search tree the ratio $p_w > p_h$ is assured with $p_{wu} = 10$ meters and $p_{hu} = 3$ meters whereas the right node swaps these values, i.e, $p_{wu} = 3$ meters and $p_{hu} = 10$ meters. Following the non-functional requirement in Section 2.1.2 on page 35, the executing time is limited to 30 seconds. Figure 3.19 shows the time expended by the GaLaS algorithm to find the first solution (independently of the number of panels).

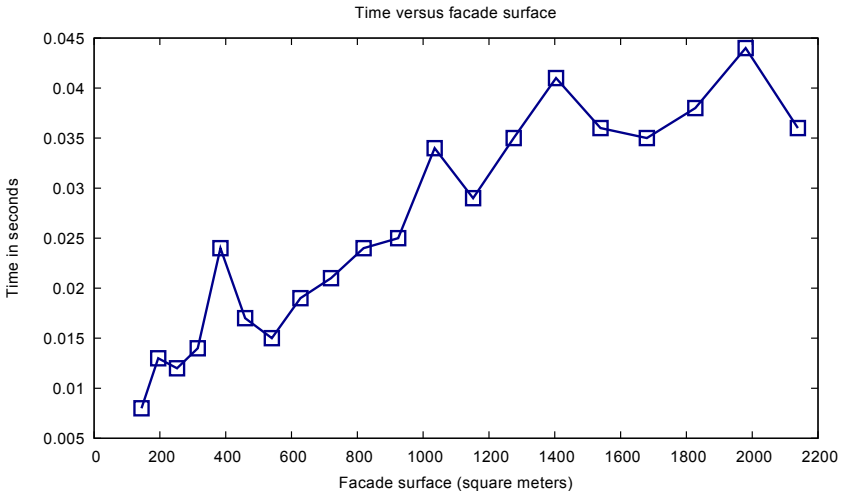


Figure 3.19 – Time first solution versus façade area.

As expected for greedy algorithms, and taking into account that no real-time interaction is presumed, the solution runs in competitive computational time, generating a envelope in less than a second for 2000 m^2 façade. From the figure is deduced that increasing the façade area does not necessarily more computational time. In fact, given that panels' corners must be matched with supporting areas in order to be properly attached, it may be the case that panels using their maximum width and height can be placed sooner in a large façade than in a small one. Lastly, the time depends on the conflicts with frames and the alignment on supporting areas which is dependent of the structure of the façade and panel size limits rather than the area of the façade.

Next, Figure 3.20 on the next page shows the number of solutions found in the time window of 30 seconds versus façade area. The number of solutions increases with the façade area provided the panel bounds remain the same. Then, from a façade with area 600 m^2, the number of solutions decreases as it has no time, in 30 seconds, to generate all possible solutions.

Finally, we have taken all solutions found in the 30-seconds time window and have ranked them with respect to the number of panels (increasing order). Figure 3.21 on the following page presents how many panels has the best ranked insulating envelopes (minimum number of panels). Naturally, the number of panels increases with the façade area.

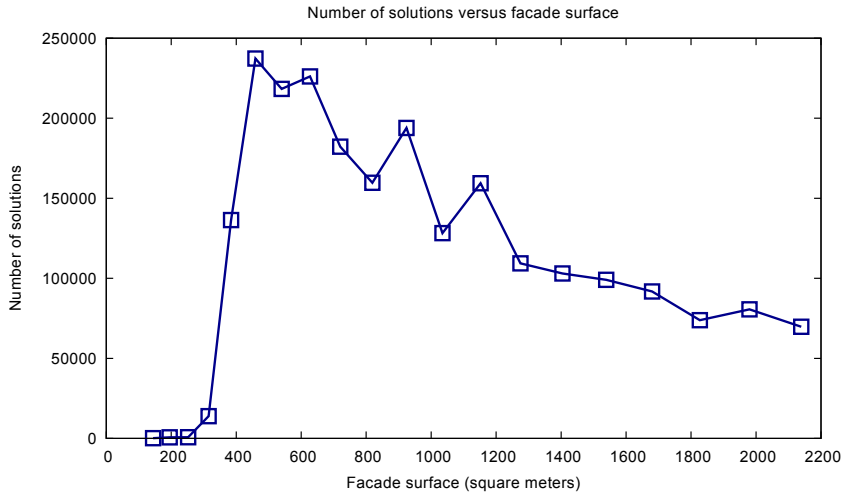


Figure 3.20 – Number of solutions (in 30 seconds) versus façade area.

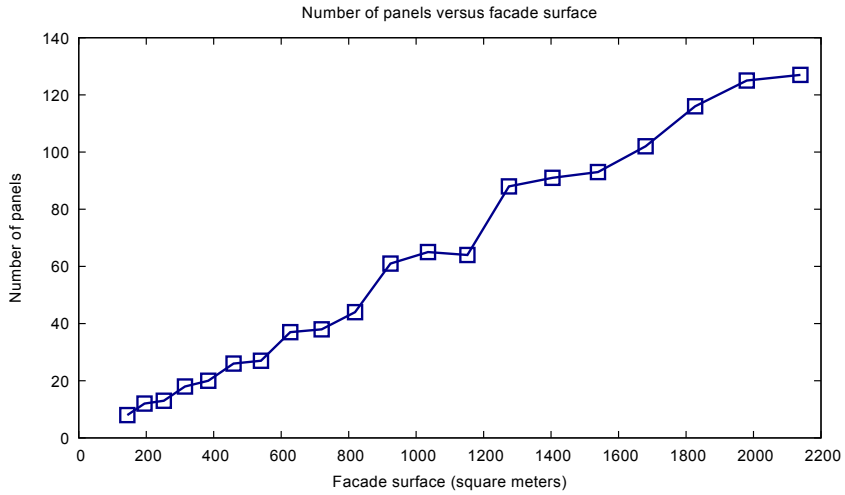


Figure 3.21 – Number of panels versus façade area.

Façades illustrations

The façades presented in Figures 2.15 on page 44, 2.14 on page 44 and 2.16 on page 44 in Section 2.4 on page 45, are used to illustrate the results generated by the algorithm. We present, before the resulting insulating envelopes, the original façade.

— Figure 3.22 on the next page presents the original façade number 1.

— Figure 3.23 on the facing page shows an envelope thrown by the GaLaS algorithm, using $p_{wu} = 3$ meters as width upper bound and $p_{hu} = 10$ meters as height upper bound, i.e., vertically oriented panels. The envelope is composed of 15 panels and its length of junctions is 113.2 meters.

— Figure 3.24 on page 68 shows an envelope thrown by the GaLaS algorithm, using $p_{wu} = 10$ meters as width upper bound and $p_{hu} = 3$ meters as height upper bound, i.e., horizontally oriented panels. The envelope is composed of 13 panels and its length of junctions is 106.9 meters. Note that the four small panels in the middle respect the constraint and are well designed but they are not optimal as they add more junctions to the envelope than only

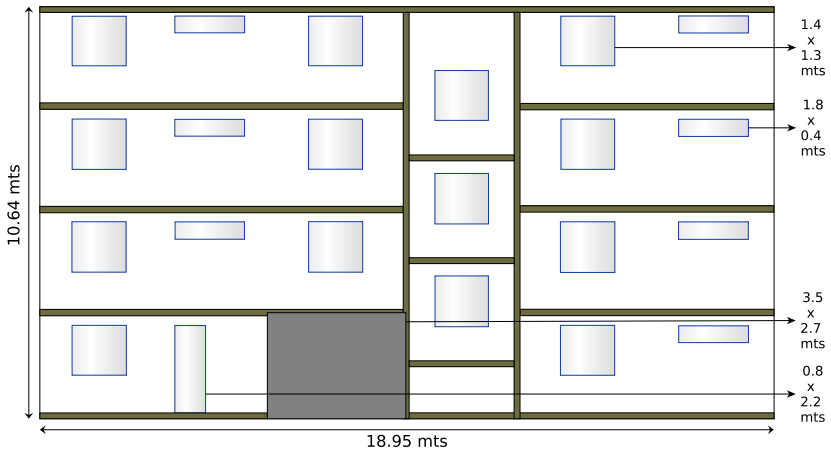


Figure 3.22 – Façade instance 1.

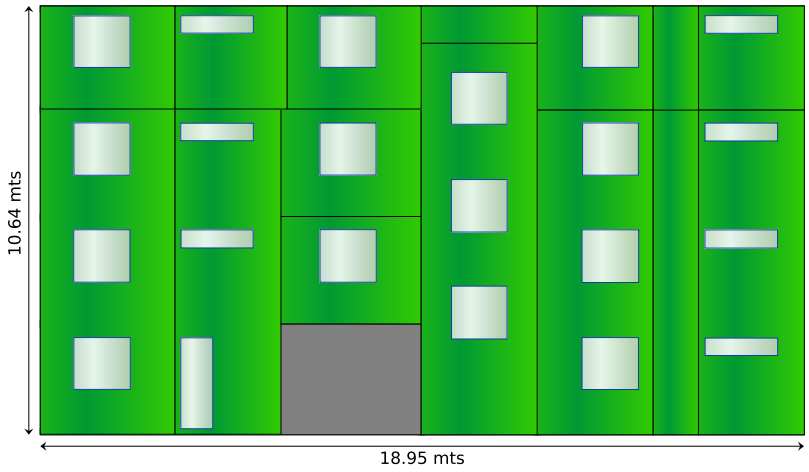


Figure 3.23 – Solution with $p_{wu} = 3$ meters and $p_{hu} = 10$ meters.

one big vertical panel. This may be overcome by the architect by designing a manual panel before launching the greedy algorithm.

— Façades in Figures 3.23 and 3.24 on the next page, however, are not the best envelopes (w.r.t. minimum number of panels or junctions length) thrown by the GaLaS algorithm; these are the best envelopes when respecting as much as possible the vertical or horizontal orientation. When asking the best solution, no matter the orientation, the generated envelope improves considerably. Figure 3.25 on the following page shows this resulting insulating envelope. The envelope is composed of 11 panels and its length of junctions is 100.555 meters. Another 16 envelopes composed of 11 panels have been generated, each of them with slightly bigger length of junctions. The envelopes are shown in Figure 3.26.

— Figure 3.27 on page 70 presents the original façade number 2.

— Figure 3.28 on page 70 shows a envelope solution made out with $p_{wu} = 3$ meters as width upper bound and $p_{hu} = 10$ meters as height upper bound, i.e., vertically oriented panels. The envelope is composed of 13 panels and its length of junctions of the envelope is 115.2 meters.

— Figure 3.29 on page 70 shows a envelope solution made out with $p_{wu} = 10$ meters as width

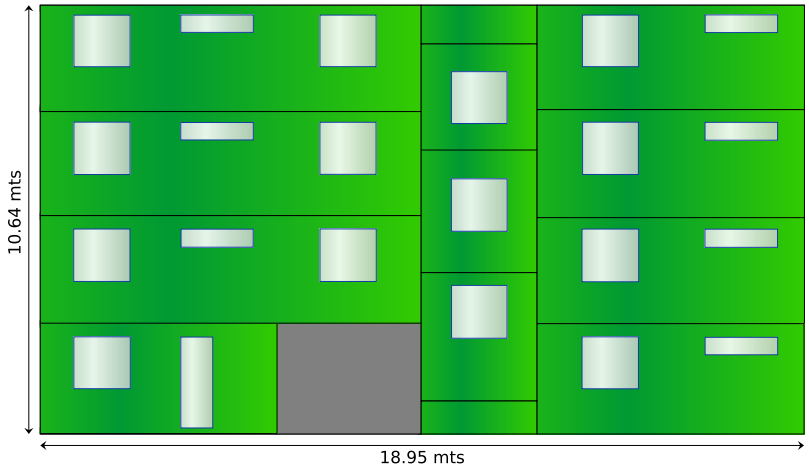


Figure 3.24 – Solution with $p_{wu} = 10$ meters and $p_{hu} = 3$ meters.

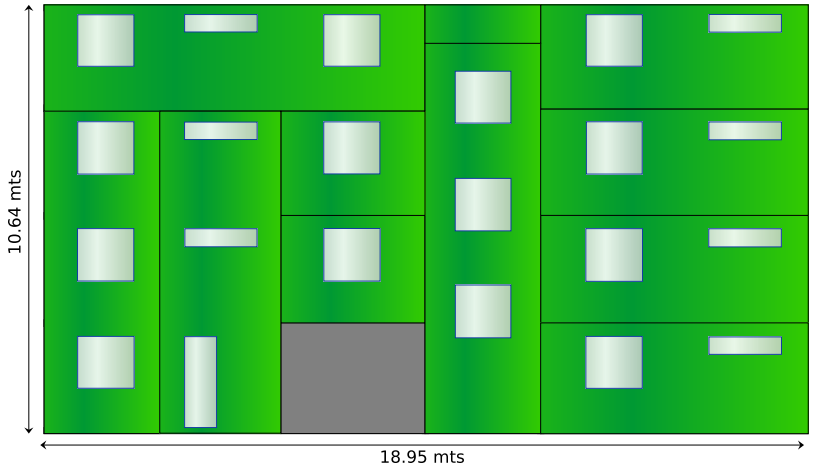


Figure 3.25 – Best solution of GaLaS algorithm for façade instance 1.

upper bound and $p_{hu} = 3$ meters as height upper bound, i.e., vertically oriented panels. The envelope is composed of 8 panels and its length of junctions of the envelope is 97.0 meters.

— Again, façades in Figures 3.28 on page 70 and 3.29 on page 70 are not the best envelopes thrown by the GaLaS algorithm. Figure 3.30 on page 71 shows the best solution thrown, independently of the orientation. The envelope is composed of 8 panels and its length of junctions is 95.6.

— Figure 3.31 on page 71 presents the original façade number 3.

— Figure 3.32 on page 71 shows a envelope solution made out with $p_{wu} = 3$ meters as width upper bound and $p_{hu} = 10$ meters as height upper bound, i.e., vertically oriented panels. Note that the blocking frame in the middle of the façade prevents the design with vertical panels from the bottom to the top of the façade. Thus, the GaLaS algorithm changes to overcome the situation. The envelope is composed of 8 panels and its length of junctions is 84.64 meters.

— Figure 3.33 on page 72 shows a envelope solution made out with $p_{wu} = 10$ meters as width upper bound and $p_{hu} = 3$ meters as height upper bound, i.e., horizontally oriented panels.

The envelope is composed of 6 panels and its length of junctions is 57 meters. This is the best solution thrown by the algorithm.

Figure 3.26 – Eight envelopes composed of 11 panels with different length of junctions.

a) Junctions length: 100.555 meters

b) Junctions length: 100.995 meters

c) Junctions length: 102.665 meters

d) Junctions length: 103.375 meters

e) Junctions length: 103.815 meters

f) Junctions length: 104.075 meters

g) Junctions length: 104.325 meters

h) Junctions length: 104.415 meters

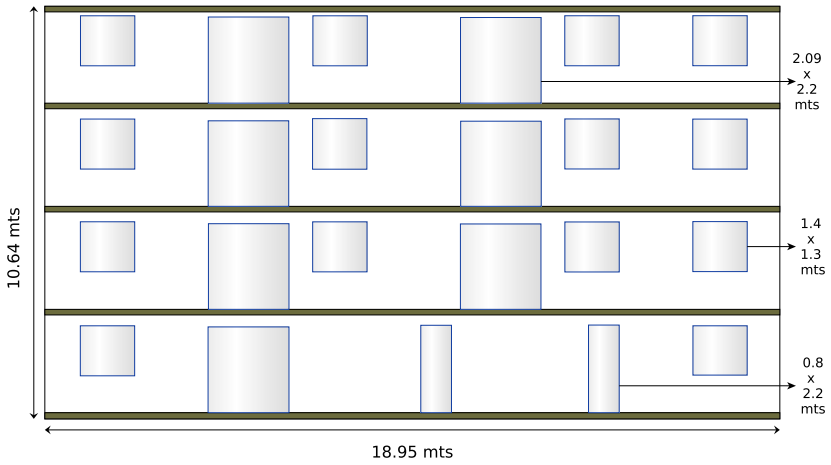


Figure 3.27 – Façade instance 2.

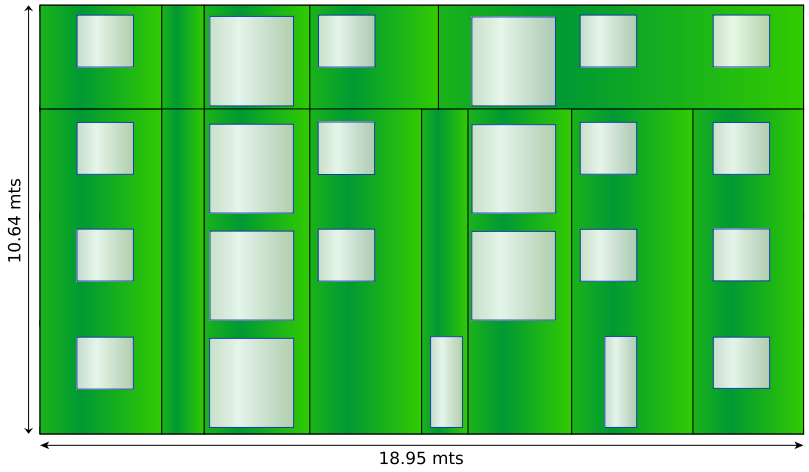


Figure 3.28 – Solution with $p_{wu} = 3$ meters and $p_{wu} = 10$ meters.

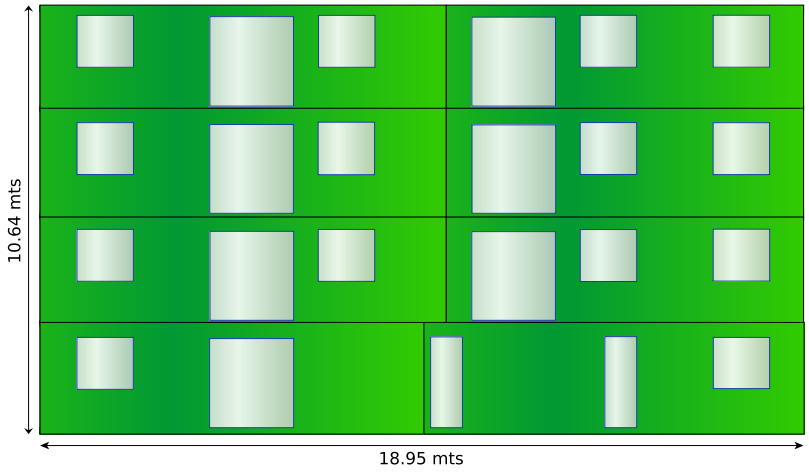


Figure 3.29 – Solution with $p_{wu} = 10$ meters and $p_{hu} = 3$ meters.

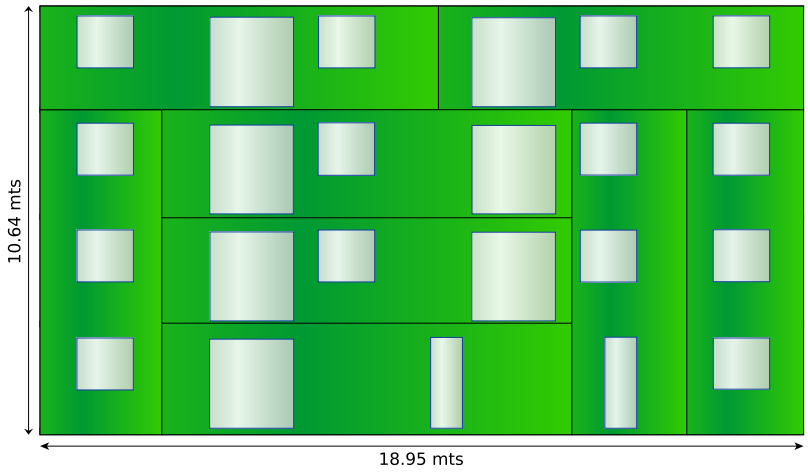


Figure 3.30 – Best solution of GaLaS algorithm for façade instance 2.

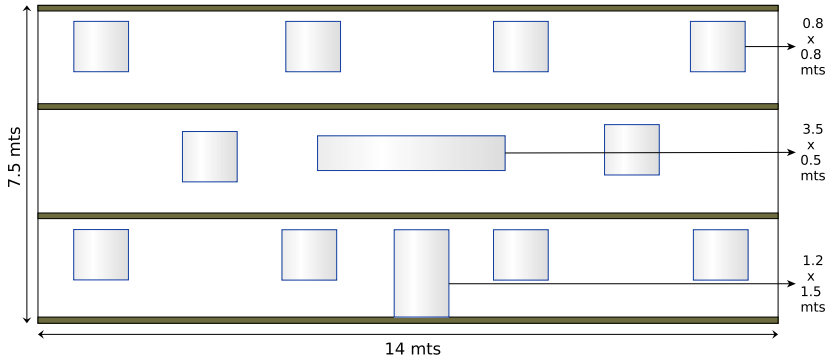


Figure 3.31 – Façade instance 3.

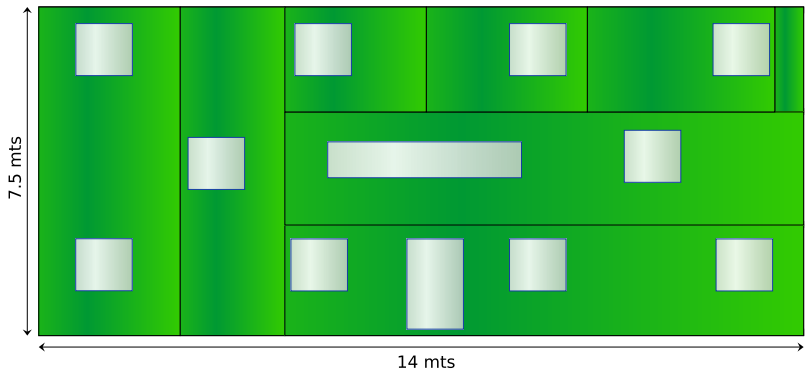


Figure 3.32 – Solution with $p_{wu} = 3$ meters and $p_{hu} = 10$ meters.

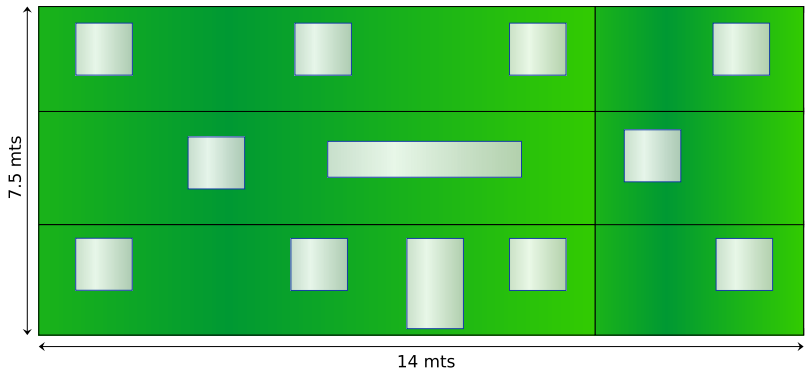


Figure 3.33 – Solution with $p_{wu} = 10$ meters and $p_{hu} = 3$ meters.

3.2.5 Discussion

In this section we have proposed a constraint-based heuristic approach, named GaLaS, based on optimal local decisions when assigning panels position and size. These local decisions allow to:

— Position (p_{x0}, p_{y0}): Choosing the first bottom-left available point of the façade makes each panel be adjacent to a previous one (or adjacent to a zoc). Then, the decision prevents holes from happening during the Cutting & Packing process.

— Size (p_w, p_h): Choosing the maximum allowed panel size allows to minimize the number of used panels. Also, due to the bottom-left origin point decision, the bigger panels will be, plausibly, at the bottom of the façade which facilitates the on-site assembly process.

This solution considers the already-defined panels (adp) or zones out of configuration (zocs) by avoiding overlapping (these adp or zocs must be well defined). Ergo, a partially designed envelope generated, for instance, by the interactive manual solution InDiE presented in Section 3.1 on page 48, may be finished by the GaLaS algorithm. Moreover, the generated relevant insulating envelopes can be adapted and tuned by a user allowing him/her to express his/her own artistic sense (discussed in Chapter 5 on page 127). This allows the user to start his/her design with a close to optimal solution and avoids starting from scratch.

The heuristic uses a given preferred panel orientation (soft constraint) to construct the envelope but it may change such orientation in order to avoid a no-solution setup. Additionally, we use a binary-tree scheme, in which left nodes positions horizontal panels and right nodes vertical ones, in the aim of generating solutions with lowest number of panels and different solutions which makes the approach more robust.

The GaLaS algorithm seeks to minimize the number of panels by always trying to place the largest ones first. Our heuristic approach provides good quality solutions, w.r.t. the number of panels and length of junctions, and is a source for strong design aiding tool for the following reasons:

Reason 1: The on-line packing is straightforward implemented under the greedy approach, addressing transparently the fact that the number of panels to design a given envelope is not known in advance.

Reason 2: The local-decision making allows the heuristic to find solutions fast whereas its combination with search trees allows the generation of solution diversity.

Reason 3: The different tests that have been carried out show the scalability of our approach as well as evaluation over different real-life French façades.

Reason 4: The design of the GaLaS algorithm makes it easily to adapt while keeping the same assumptions, to convex polygon shapes, such as triangles or pentagons to cover gables for instance, without modifying the core of the algorithm.

We acknowledge that the approach is *"blind"* in the sense that the façade geometry and structure, as well as future states in the packing process, are overlooked. Nevertheless, it is a fast and intuitive design and, as discussed, presents different benefits.

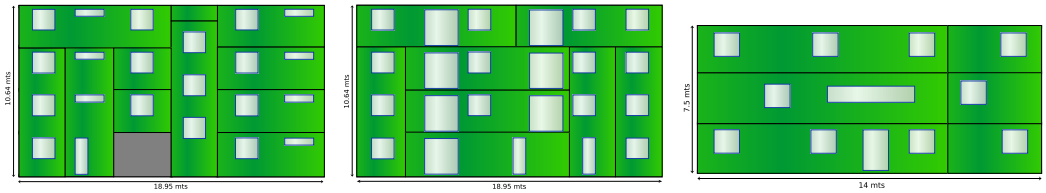
To conclude, we present a summary of the GaLaS algorithm results over the three façade instances introduced in Section 2.3 on page 43. Table 3.1 on the next page presents the results where N and Loj represents, respectively, the number of panels and the length of junctions in the envelopes. The columns vertical and horizontal show the results, respectively, of the GaLaS algorithm when asking envelopes with only one orientation and without combination of the binary tree. Finally, the column V & H shows the result when combined with binary tree and thus envelopes are a mix of horizontal and vertical panels. In addition, and for convenience, we present in Figure 3.34 on the following page the best solutions generated by the algorithm.

As the table shows, the combination of horizontal and vertical panels in an insulating envelopes decreases the number of used panels and consequently the length of junctions. The vertical solution for the façade 1 and 2, however, does not have all its composing panels in a vertical orientation (cf. façade in Figures 3.23 on page 67 3.28 on page 70). This makes the

Table 3.1 – Results of GaLaS algorithm over façade instances.

		Panels orientation		
		Vertical	Horizontal	V & H
Façade #1	N	15	13	**11**
	Loj	113.2	106.9	**100.5**
Façade #2	N	13	**8**	**8**
	Loj	115.2	97.0	**95.6**
Façade #3	N	8	**6**	**6**
	Loj	63.8	57.0	**57.0**

the vertical envelope for the façade 1 to have more panels and for the second façade less panels than it would if the orientation would not have been changed. This is consequence of the local decision-making and the soft constraint for panel orientation implemented by the GaLaS solution.



(a) 11 panels (b) 8 panels (c) 6 panels

Figure 3.34 – Best solutions founded for the three façades.

3.3 Cutting Design: CaSyE

This section presents a cutting solution approach for the automatic design of insulating envelopes. Part of the content of this section has been presented in Barco et al. (2016a).

3.3.1 Motivation

Tacit knowledge as aesthetics is a major challenge for both the modelling and implementation of computer-based solutions. It is known that aesthetics flair has no universally agreed standard. Nonetheless, properties like symmetry are well accepted as aesthetic concept in different domains such as human beauty (Jacobsen and Höfel, 2003), web design (Tuch et al., 2010), computer interfaces (Bauerly and Liu, 2008) and art (McMANUS, 2005). In consequence, we have adopted symmetry as a reference point for the aesthetics of insulating envelopes. This symmetry notion is considered as the *alignment of panel junction*. The second algorithmic solution for automatic design, that focuses on symmetry of envelopes, is here presented.

This second automatic algorithmic solution contrasts with the greedy approach in that it designs panels following the geometry and structure of the façade (e.g., it is not *"blind"*). Due to this fact, the solution designs envelopes with aligned junctions (symmetric). We do so by developing a rule-based heuristic, called CaSyE for **C***utting* **A***lgorithm for* **Sy***mmetrical* **E***nvelopes*, based on the well-known technique of guillotine cuts (Christofides and Hadjiconstantinou, 1995). Indeed, cutting is a technique from operations research widely used to solve different industrial problems (Bennell et al., 2013, Christofides and Hadjiconstantinou, 1995, Wäscher et al., 2007) and it has been applied on problems where symmetry is relevant for the final output. For instance, arranging items in a news paper (Strecker and Hennig, 2009), automatic mosaic generation (Battiato et al., 2013) and aesthetics photo post-processing (Greco and Cascia, 2013).

The guillotine approach refers to the process of executing orthogonal cuts, horizontal or vertical, from one edge of the object to the other edge (Christofides and Hadjiconstantinou, 1995, Ntene and van Vuuren, 2009). The non-guillotine technique, by contrast, may traverse only a portion of the object in a orthogonal manner (Baldacci and Boschetti, 2007, Alvarez-Valdes et al., 2007). The guillotine is often seen as a constraint, as well as the orthogonal property, imposed by the automated machines in particular industry sectors (Cui et al., 2008). Our motivations behind using the guillotine techniques are, in the first place, the orthogonality of the items (panels) and the large object (façades), and on the second place, the goal of generating (symmetric) solutions with aligned junctions. Then, we use the guillotine approach and show that it is able to generate aesthetic (aligned junctions) insulating envelopes not generated by the greedy approach, thus satisfying this crucial architectural requirement. The algorithm executes horizontal and vertical cuts to partition the façade and to find non-conflictive areas to attach panels. The concept of design using guillotine cuts is illustrated in Figure 3.35 on the next page.

3.3.2 Scheme

The main idea behind the algorithm is to generate envelopes "aesthetically pleasant" by considering their junctions alignment (symmetry) while respecting the industrial conditions commented above. To do this, and respecting the compatibility constraint between width and height, the set of panels in an envelope are configured according to a given ration between p_w and p_h, i.e., a given panel orientation (horizontal for $p_w > p_h$ and vertical for $p_w < p_h$). Then, for a given façade, the algorithm tries as much as possible to design panels using the chosen orientation. If due to the geometry of the façade, a portion of the envelope cannot be designed using the chosen orientation, the algorithm creates a partition of the façade (called here subfaçade) and tries to design that partition by changing the panel orientation. This means, on the one hand, that insulating envelopes may contain only vertical panels, only horizontal panels or a combination of both. And, on the other hand, that the final insulating envelope is the set of panels belonging to every subfaçade no matter their orientation. As it is shown later in the document, mixing panels orientation on the same envelope does not necessarily interferes with the aesthetics goal. The

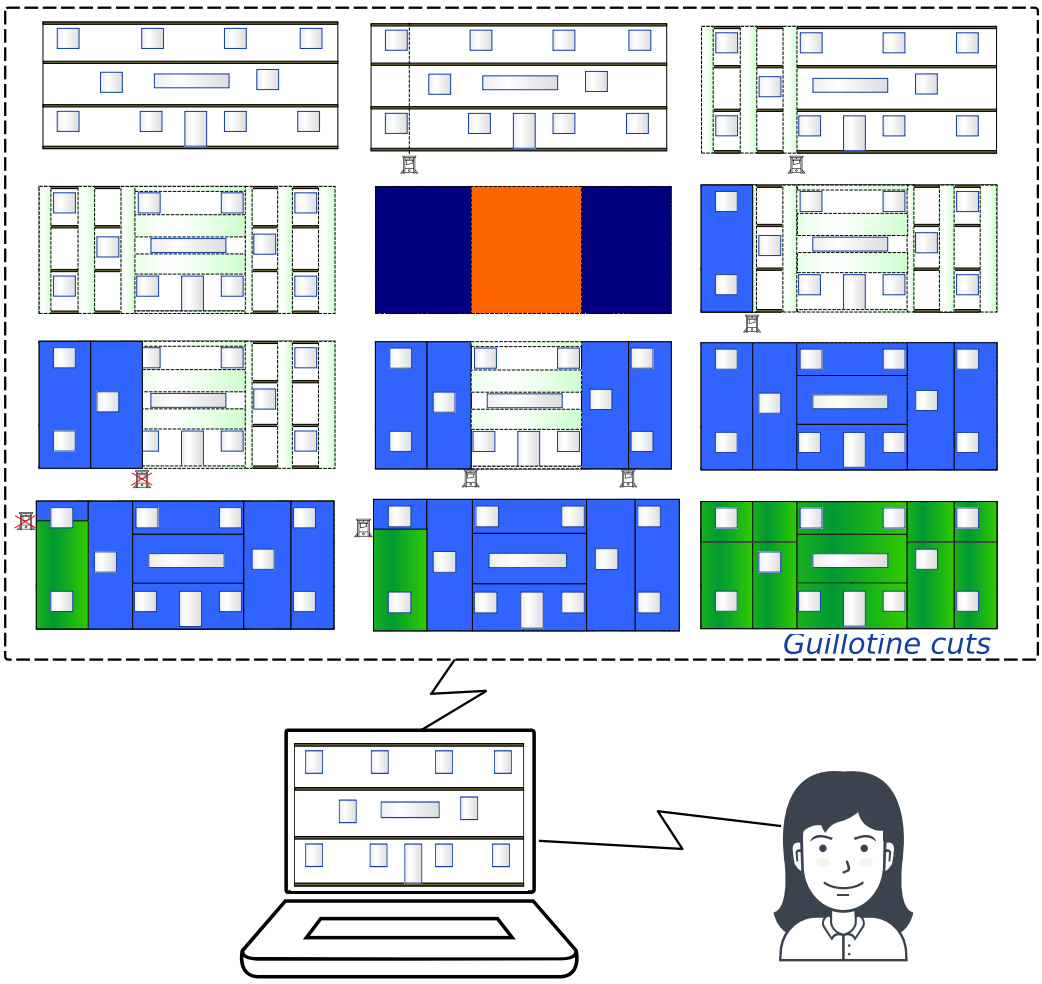


Figure 3.35 – Concept: Guillotine cuts for envelope design.

algorithmic solution is divided in three phases executed sequentially. Be aware that the preferred panel orientation is taken into account at the begging of the process but may be changed if needed.

Phase 1 The goal of the phase 1 is two-fold. First, using guillotine cuts, find the rectangular areas, vertical or horizontal, in which panels can be attached (i.e., those places with no conflicts with frames). And second, given the rectangular areas that have been found and the façade structure, determine if portions of the envelope cannot be designed with the preferred orientation. The façade then may be partitioned into pieces that we call subfaçades. An attempt to design all subfaçades using the other orientation is latter made.

Phase 2 For an envelope designed using vertical (respectively horizontal) panels, find the columns (respectively rows) where panels have to be placed. This columns (respectively rows) are generated taking into account the available areas to attach panels generated by the Phase 1 and the panel upper bounds.

Phase 3 For an envelope designed using vertical (respectively horizontal) panels, pack panels over the columns (respectively rows). The panels packing is done one by one following a greedy approach to assign the final size and to solve remaining conflicts with the frames over the façade.

Invariant. As an invariant for all phases, a guillotine cut in a given façade or subfaçade is done only if no frames conflicts exists.

Phase 1: Free zones and subfaçades

The goal of the phase 1 is to know whether the envelope can be designed using only vertical panels, only horizontal panels, both or to find the subfaçades and characterize them within the façade. Intuitively, when using guillotine cuts to design an envelope with vertical panels, for instance, the façade horizontal axis is traversed looking for points with absence of frames. When there is no frames in a given horizontal point, a vertical guillotine cut may trace from the bottom to the top of the façade (see Figure 3.36.a). A set of consecutive cuts make an interval (see Figure 3.36.b). These lines or intervals, termed as Zones Free of Conflicts (`ZoFCo`), are latter used to place panels' borders. Then, the algorithm of the phase 1 tries to deduce if the façade insulating envelope can be designed using vertical panels by checking the position and dimension of the `ZoFCos`. If portions of the façade (subfaçades) cannot be cover with panels in the chosen orientation, the algorithm makes a division into subfaçades and marks each of them with a type for further processing. In particular, subfaçade types are:

— `Vertical`: The envelope for the subfaçade can be designed using vertical panels.

— `Horizontal`: The envelope for the subfaçade can be designed using horizontal panels.

— `NotVertical`: The envelope for the subfaçade cannot be designed using vertical panels.

— `NotHorizontal`: The envelope for the subfaçade cannot be designed using horizontal panels.

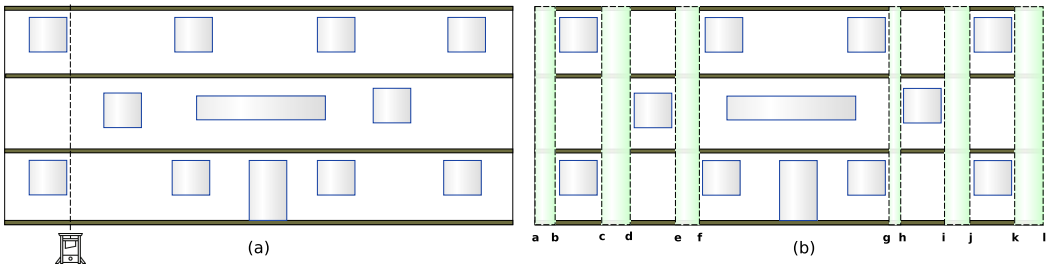


Figure 3.36 – Guillotine cuts for intervals and subfaçades definition.

A subfaçade may be marked more than one time. However, only some combinations are generated by the phase 2; `NotVertical-Horizontal` when a subfaçade has been marked as `NotVertical` and consequent processing has marked it as `Horizontal`; `NotHorizontal-Vertical` when a subfaçade has been marked as `NotHorizontal` and consequent processing has marked it as `Vertical` and; `NotVertical-NotHorizontal` when a subfaçade has been marked as `NotVertical` and consequent processing has marked it as `NotHorizontal`. In the last case the insulating envelope cannot be designed using the proposed algorithm. For instance, let us study the façade in Figure 3.37 on the following page. Here, let us assume panels' upper bounds of 3 meters for one dimension and 10 meters for the other dimension (i.e., 10×3 for a horizontal panel or 3×10 for a vertical panel). Envelopes for the subfaçades sub^1 and sub^2 may be designed using vertically and horizontally oriented panels. But, the subfaçade sub^3 at the right cannot be designed: One dimension (width or height) may be successfully covered with 10 meters but the other dimension, that can only takes as maximum size 3 meters, cannot be covered. Now, if a subfaçade cannot be designed then the original façade would have a hole in its insulating envelope, which is forbidden.

To clearly understand the scheme previously discussed, let us study the steps for the phase 1 using as example vertical orientation.

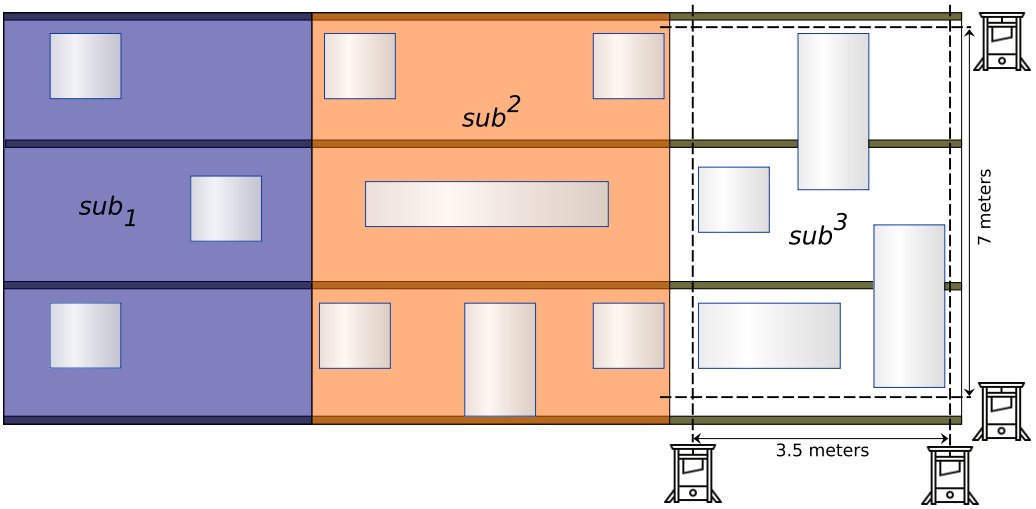


Figure 3.37 – Subfaçade (consequently façade) with no solution with guillotine cuts.

Step 1: Let $x_1 = 0$ (i.e., left border of façade) iterate to the right until the end of façade and do:

i) Let x_2 be equal to the origin point of the first frame found.

ii) Define a ZoFCo from x_1 to x_2.

iii) Set x_1 equals to the right edge of the frame. If another frame is blocking the guillotine cut at x_1, then update x_1 to the end point of that blocking frame and repeat until no blocking frames are found. For example, in Figure 3.36.b, after adding the ZoFCo defined by [e,f], it should start the new iteration with $x_1 = g$ as in this point a vertical non-conflictive cut is possible.

Step 2: For all ZoFCo found, make an ordered check: If the distance between the end of ZoFCo $_i$ and start of ZoFCo $_{i+1}$ is bigger than the maximum panel width p_{wu}, then at least two subfaçades and at most three subfaçades have been found (more subfaçades may be found when checking the remaining ZoFCos). This means that the space to be covered has a larger width than the maximum panel width. In our example, the first subfaçade sub^1, labeled as `Vertical`, goes from a until the point f (see Figure 3.38.b subfaçade at the left). The second subfaçade sub^2, labeled as `NotVertical`, goes from f until the point g (see Figure 3.38.b subfaçade in the middle). The third subfaçade sub^3, labeled as `Vertical`, goes from g until the point l (see Figure 3.38.b subfaçade at the right).

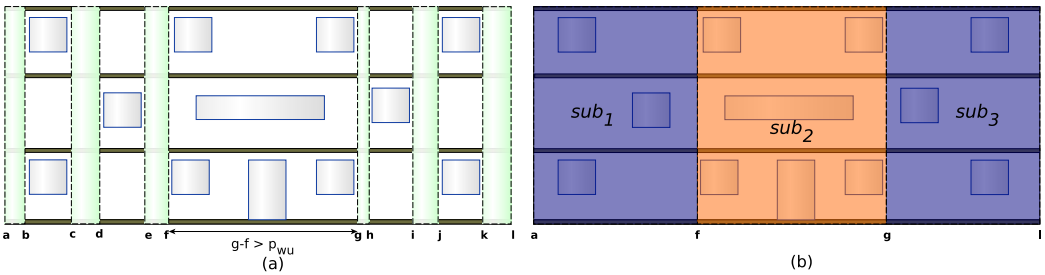


Figure 3.38 – Definition of subfaçades according to ZoFCos.

Step 3: For each subfaçade try failure detection. If its width is less than p_{wl}, then; a) merge it with an adjacent subfaçade already marked as `NotVertical` (if any) or; b) mark the subfaçade as `NotVertical`. For instance, the width of the subfaçade 1 in Figure 3.39.a is smaller that the panel's width lower bound p_{wl}, then, the subfaçade must be marked as horizontal and merged with an adjacent subfaçade as is shown in Figure 3.39.b.

Step 4: For any subfaçade marked as `NotVertical`, try to design the subfaçade envelope using horizontal orientation.

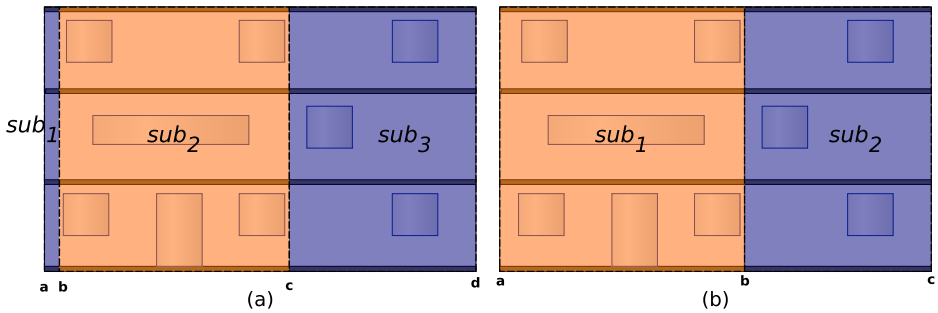


Figure 3.39 – Subfaçades failure detection and solution.

At the end of the process, every subfaçade has been marked at least one time.

Phase 2: Defining columns and rows

After the phase 1, the second phase is executed for each subfaçade. In this phase, the columns (respectively rows) for the vertically (respectively horizontally) designed envelope must be defined. It is at these columns (rows) where panels are going to be placed, the edges of the panels matching the edges of the columns. For convenience, let us continue our solution description using vertical panels. The first task is to determine where the left border and the right border of columns will be placed. To do so, the phase 2 uses the ZoFCos intervals that have been found in the previous phase as they are free of frames conflicts. Then, it suffices to select a point within the ZoFCos to define the columns. As an invariant, given that the entire subfaçade must be covered (no holes in the envelope) and considering the column definition from the left to the right of the subfaçade, the end of the column i must be equal to the start of column $i+1$.

Taking into account that envelopes should be composed of the minimum number of panels, the definition of the columns, for instance, is made using the upper bounds for panels width (respectively height). The idea is to place the left and right edge of the column over one or two ZoFCo, in such a way that the width is maximal (see Figure 3.40.a). If using the width upper bound makes the column enter in a frame conflict, meaning that the point in which the width is maximal does not match a ZoFCo, then the upper bound cannot be used for the current column width (see Figure 3.40.b). In consequence, the width of the column is reduced, as less as possible, while solving the conflict (see Figure 3.41.a). Potential inconsistent sizes are handle as well. The algorithm for the phase 2 is applied to every subfaçade *independently* and behaves as follows.

Let x_1 and x_2 define the start and end of the column i. Both variables are used to iterate until the end of the subfaçade. Then do:

Step 1: Let x_1 be equal to the origin (left border) of the subfaçade.

Step 2: Let $x_2 = x_1 + p_{wu}$.

Step 3: If the point x_2 does not belong to any ZoFCo, then move x_2 to be the end (right edge) of the previous ZoFCo (the previous at the left). As an illustration, the second column

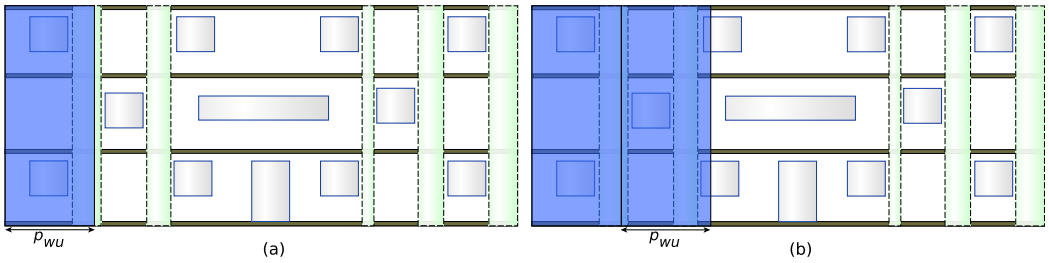


Figure 3.40 – Guillotine cut for columns definition.

in Figure 3.40.b is redefined to match the first ZoFCo on the left, as shown in Figure 3.41.a.

Step 4: Try failure detection. If $x_2 - x_1 < p_{wl}$, then reduce width of previous column $i-1$ by $p_{wl} - (x_2 - x_1)$ and update x_1 and x_2. This process must be done iteratively as a width reduction in any previous column may generate new size conflicts. Lastly, if there is no previous column then fail and exit.

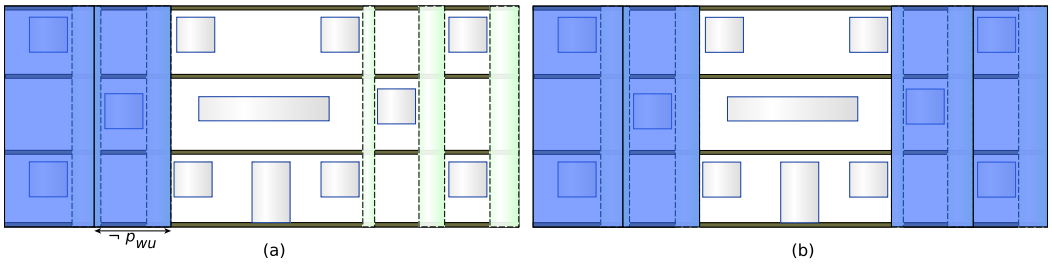


Figure 3.41 – Guillotine cut for columns definition (continuation).

Step 5: Define the column i from x_1 to x_2.

Step 6: Set $x_1 = x_2$ and iterate again from Step 2 until all the subfaçade has been processed.

At the end of the process, every subfaçade has been divided in columns (respectively rows) where panels borders will be located. The last process then sets the final position and size of panels over these columns.

Phase 3: Panels packing

The packing of panel is executed for each of the columns and rows generated in the phase 2. As commented before, this last phase has as objective to set the final position and size of panels. Additionally, this phase handles potential conflicts with frames by executing guillotine cuts in non-conflictive zones. Likewise the previous phases, the packing starts by an extreme of the subfaçade until its end. In the case of an envelope designed using vertical panels, the packing process starts in the bottom of each column by extending the panel in its maximum allowed height (see Figure 3.42.a). If a frame is blocking the horizontal cut, then the panel height must be reduced to the first place in which no conflict exists (see Figure 3.42.b).

The packed panels in a given column will have the width of the column and the height defined by the horizontal guillotine cut. Again, a similar process is carried on rows when designing envelopes with horizontally oriented panels. The algorithmic solution of the third phase, applied to each column *independently*, behaves as follows.

Let y_1 and y_2 define the bottom edge and top edge of the panel i. Both variables are used to iterate until the end of the column. Then do:

Step 1: Let y_1 equals the bottom of the column

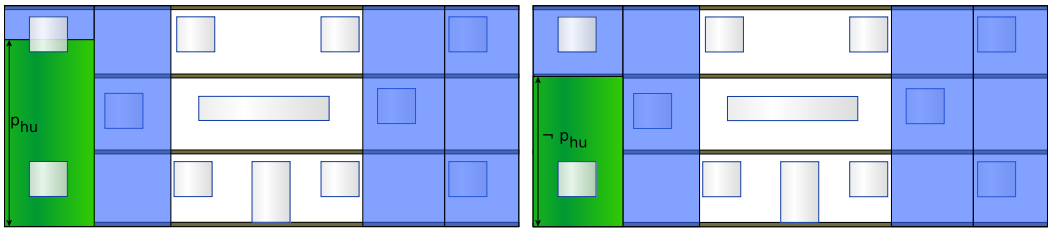


Figure 3.42 – Guillotine cuts for packing panels.

Step 2: Let $y_2 = y_1 + p_{hu}$

Step 3: If horizontal guillotine cut in y_2 has conflicts with frames, then move y_2 to be under conflictive frames (as illustrated in Figure 3.42.b).

Step 4: Check if top corners are included in supporting areas. If this is not the case, reduce the panel's height until the first supporting areas are found (as presented in Figure 3.43).

Step 5: Using the current position and size, compute the panel's weight and check if it is successfully supported in the supporting areas at the bottom corners. If the weight cannot be supported then reduce the panel size in 10% and iterate again from Step 3.

Step 6: Try failure detection. If the $y_2 - y_1 < p_{hl}$, then reduce height of previous panel $i-1$ by $p_{hl} - (y_2 - y_1)$ and update y_1 and y_2. If there is no previous panel then fail and exit.

Step 7: Define a panel from y_1 to y_2 in the current column.

Step 8: Set $y_1 = y_2$ and iterate from Step 2 until the column has been processed.

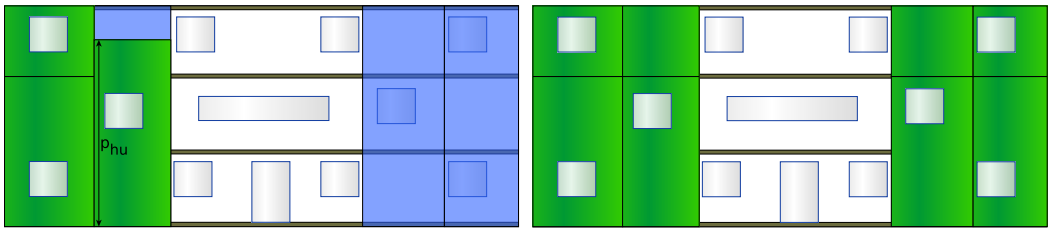


Figure 3.43 – Guillotine cuts for packing panels.

It may be the case that the defined columns defined in the previous phase do not traverse supporting areas, in which case the envelope cannot be designed with the CaSyE algorithm using the current panel size bounds. Otherwise, at the end of the this phase, each column (respectively row) has been covered by panels (as presented in Figure 3.43.b). The resulting insulating envelope for the façade is the union of all panels of every subfaçade (if any). Now, let us discuss how these phases are merged together to assist architects design.

3.3.3 Implementation

The phases 1, 2 and 3 are executed sequentially to generate a given envelope. This sequential process is executed twice in order to generate two different solutions (if the façade geometry allows it): One setting the ratio $p_w > p_h$ (horizontally oriented panels) and one setting the ratio $p_w < p_h$ (vertically oriented panels). To understand how the phases work together, we present to figures of algorithms. The first one, in Figure 3.44 on the next page called *Guillotine Design Given Orientation*, executes all the phases using a given orientation. To do so, starting from

the original façade, it finds the ZoFCos, then finds the subfaçades (marking them accordingly), creates the columns/rows and then allocates the panels according to the orientation. Every subfaçade marked with the selected orientation, for instanceVertical, is processed by the phase 2 and phase 3 whereas the other subfaçades are stored in a list of *unknown* subfaçades. Panels successfully designed in the execution are stored in a list called *solution*.

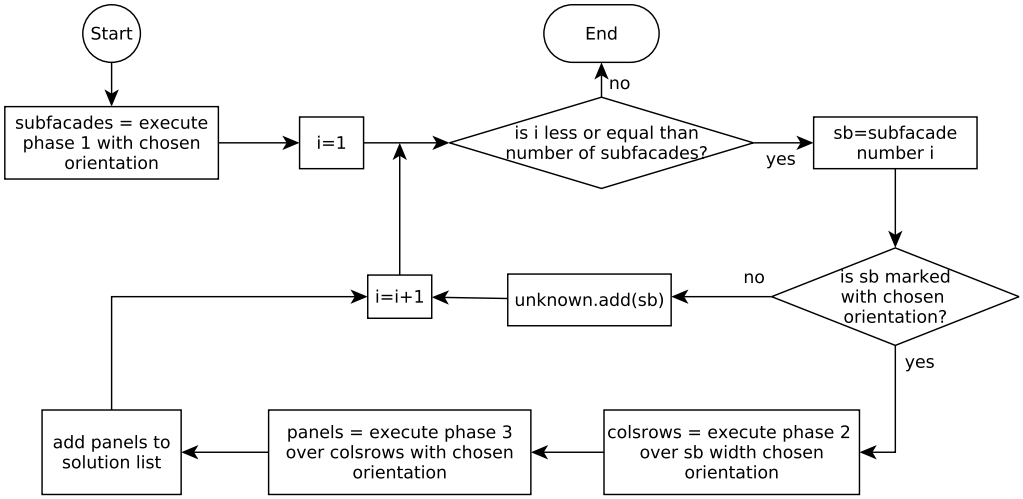


Figure 3.44 – Algorithm 9: Guillotine design given orientation.
***Inputs*:** Panel size bounds $(p_{wl}, p_{wu}, p_{hl}, p_{hu})$, list of frames (F), list of supporting areas (sa), minimum distance between panel borders and frames borders (d), orientation (p_o), façade size $(fac_w,\ fac_h)$, subfaçades list unknown=[], list solution=[].
***Output*:** Designed panels in *solution* list and new subfaçades in *unknown* list.

Algorithm 9 is used in the Algorithm 10 named CaSyE, presented in Figure 3.45 on the facing page, which represents the full solution.

It begins by setting the list of subfaçades that have not yet been processed, i.e., the original façade. Then, it executes Algorithm 9 attempting to design the envelope using vertical orientation. This process may dynamically create more subfaçades to be latter processed by the same Algorithm 9 but using horizontal orientation. In any moment, if the list of not processed subfaçades is empty then the algorithm ends. Also, after executing the design with vertical panels, every subfaçade is checked for two marks, NotVertical and NotHorizontal, to early detection of failure. In an attempt to overcome the no solution situation, the algorithm is executed twice: One starting vertically designed envelopes and another starting with horizontally designed envelopes. Indeed, the orientation in which starts an envelope design has an influence in the kind of subfaçades found and consequently in the final output. In other words, swapping the execution of vertical and horizontal orientation has an impact in the resulting solution. Thus, a given façade has, potentially, two different solutions.

3.3.4 Evaluation Cases

In this section are presented some scalability tests and examples of envelopes generated by the cutting solution CaSyE.

Scalability

As stated in Section 2.3 on page 43, panel size upper bounds are 10 meter for one dimension and 3 meters for the other one. Consequently, the CaSyE algorithm is executed twice, one with

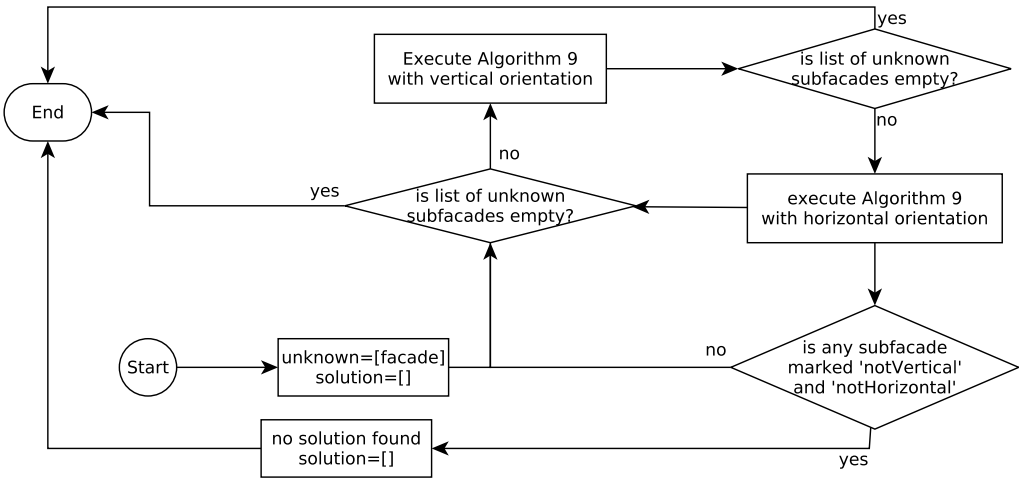


Figure 3.45 – Algorithm 10: CaSyE - Cutting Algorithm for Symmetrical Envelopes
Inputs: Panel size bounds $(p_{wl}, p_{wu}, p_{hl}, p_{hu})$, list of frames (F), list of supporting areas (sa), minimum distance between panel borders and frames borders (d), façade size $(fac_w,\ fac_h)$, solution list=[].
Output: Designed panels in *solution* list (possibly empty).

vertically oriented panels assured with $p_{wu} = 3$ meters and $p_{hu} = 10$ meters, and a second time with horizontally oriented panels assured with $p_{wu} = 10$ meters and $p_{hu} = 3$ meters.

Figure 3.46 shows the time expended by the CaSyE algorithm to find the first solution (independently of the number of panels and orientation) whereas Figure 3.47 shows the number of panels of the best solution (w.r.t. number of panels) on the envelopes versus façade area.

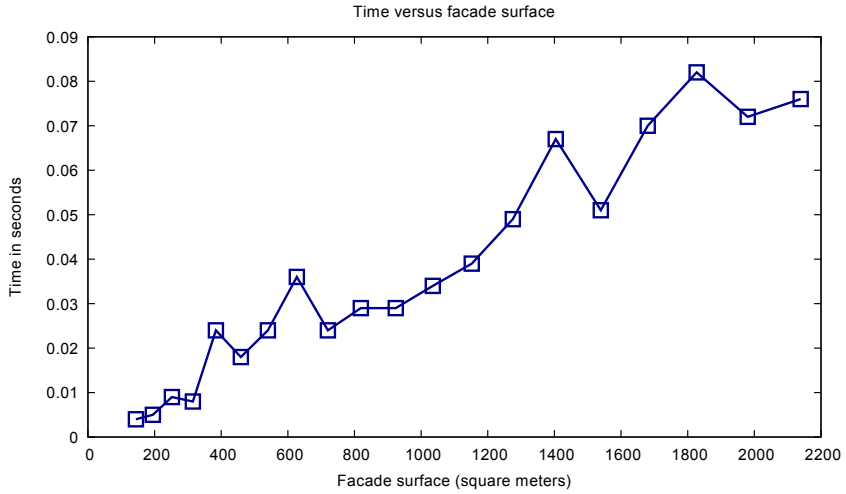


Figure 3.46 – Time versus façade area.

According to the graphic, the cutting solution solution runs in competitive computational time, generating a envelope in less than a second for a façade with area 2000 m^2. As commented previously, the complexity of the solution is in relation with the position and number of frames over the façade surface; the more frames the more comparisons the algorithm does and consequently the relation façade area-time is almost linear.

In the same way, the number of panels used in a given envelope grows with the façade area, number and position of frames (see Figure 3.47). This holds true only if the panel size bounds are the same in every instance, as is the case in our experiments. Here, however, no aesthetics measurement is done. In fact, although an evaluation of symmetry may be built with respect to the junctions alignment and panels sizes, these are not a sufficient condition to consider an envelope aesthetically pleasant. The aesthetics evaluation must be still done visually. The next subsection will then present some insulating envelopes generated by the algorithm and their aesthetics aspects will be discussed.

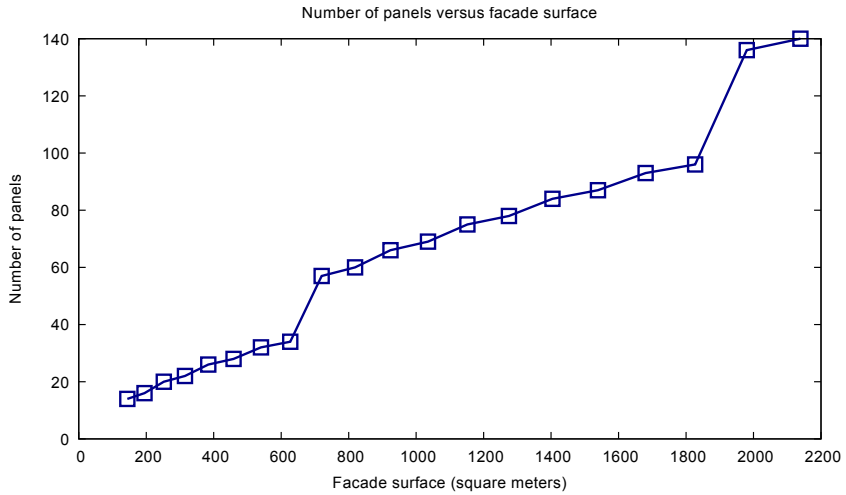


Figure 3.47 – Number of panels versus façade area.

Façades illustrations

The façades presented in Figures 2.15 on page 44, 2.14 on page 44 and 2.16 on page 44 in Section 2.4 on page 45, are used to illustrate the results generated by the algorithm. Similarly as the GaLaS illustrations, every literal (a) of each figure presents the original façade whereas literals (b) and (c) present, respectively, insulating envelopes with vertical and horizontal panels.

— Figure 3.48 presents the original façade number 1.

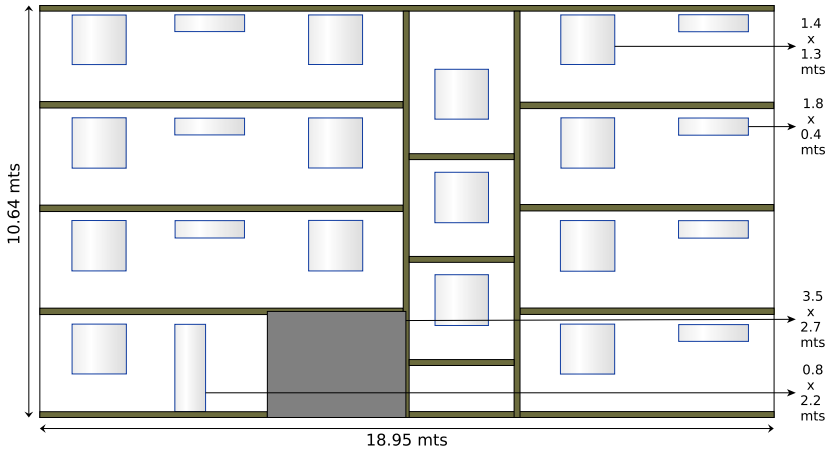


Figure 3.48 – Façade instance 1.

— Figure 3.49 shows an envelope thrown by the CaSyE algorithm, using $p_{wu} = 3$ meters as width upper bound and $p_{hu} = 10$ meters as height upper bound, i.e., vertically designed envelope. Note that the already-defined panel (black rectangle) has a width bigger than the panels width upper bound $p_{wu} = 3$. In consequence, the area above it has been divided in two columns and lastly two different panels. The envelope is composed of 14 and its length of junctions is 108.2 meters.

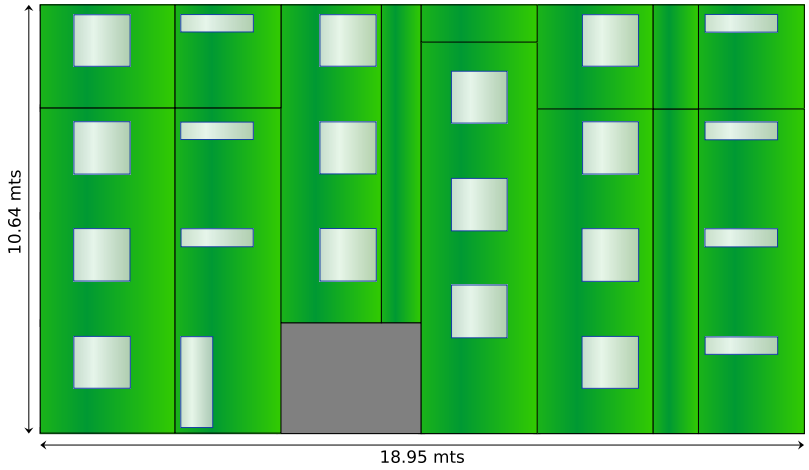


Figure 3.49 – Solution with $p_{wu} = 3$ meters and $p_{hu} = 10$ meters.

— Figure 3.50 shows the result when executing the CaSyE algorithm using $p_{wu} = 10$ meters as width upper bound and $p_{hu} = 3$ meters as height upper bound, i.e., horizontally designed envelope.

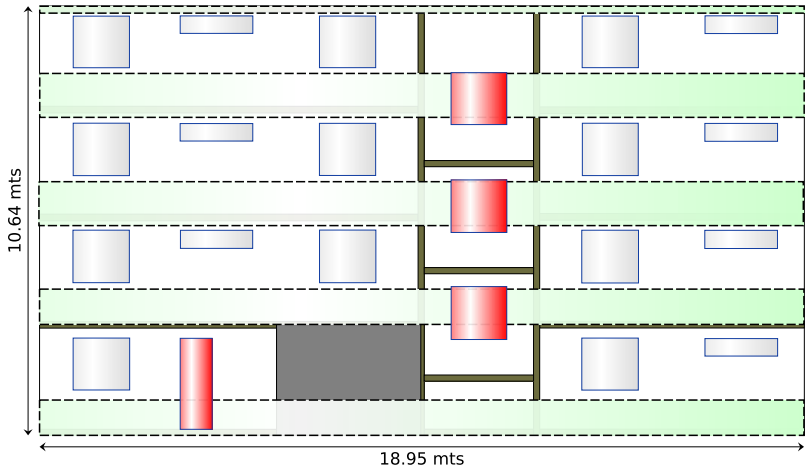


Figure 3.50 – Solution with $p_{wu} = 10$ meters and $p_{wu} = 3$ meters.

Simply stated, the envelope for this façade cannot be designed with the CaSyE algorithm using horizontally oriented panels. The reason for this is evident; it is not possible to generate the respective ZoFCos to attach panels' corner due to the position of the frames (the figure illustrates the impossibility over a set of these frames). In consequence, the algorithm would try to design the envelope in the other orientation, given as result the same envelope of Figure 3.49. To overcome this situation, the technique of non-guillotine

cuts seems a good strategic direction of work. This consideration will be discussed as future work (Section 6.2 on page 141).

— Figure 3.51 presents the original façade number 2.

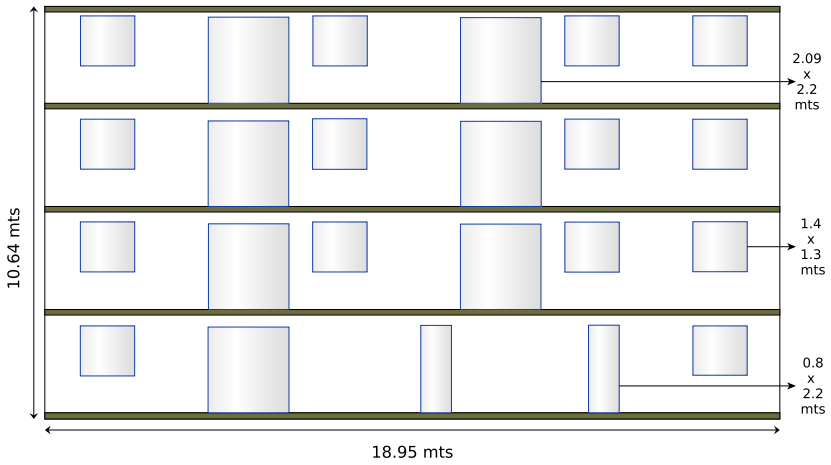


Figure 3.51 – Façade instance 2.

— Figure 3.52 shows a envelope solution made out with $p_{wu} = 3$ meters as width upper bound and $p_{hu} = 10$ meters as height upper bound, i.e., vertically designed envelope. The envelope is composed of 16 panels and its length of junctions is 123.02 meters.

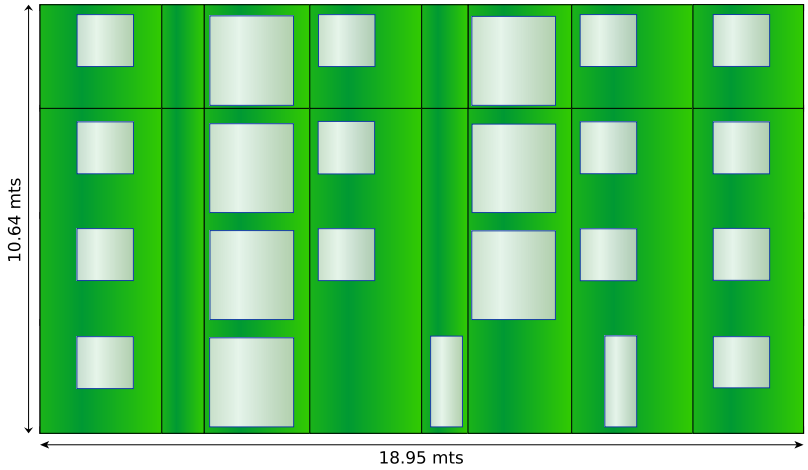


Figure 3.52 – Solution with $p_{wu} = 3$ meters and $p_{hu} = 10$ meters.

— Figure 3.53 on the next page shows a envelope solution made out with $p_{wu} = 10$ meters as width upper bound and $p_{hu} = 3$ meters as height upper bound, i.e., horizontally designed envelope. The envelope is composed of 8 panels and its length of junctions is 97.0 meters. Note that this solution is almost the same as the one generated by the GaLaS algorithm (cf. Figure 3.29) differentiable only because the joints in the CaSyE solutions are aligned.

— Figure 3.54 on the facing page presents the original façade number 3.

— Figure 3.55 on the next page illustrates an envelope solution made out with $p_{wu} = 3$ meters as width upper bound and $p_{hu} = 10$ meters as height upper bound, i.e., vertically designed envelope. Here, although the frame in the middle of the façade is blocking the definition

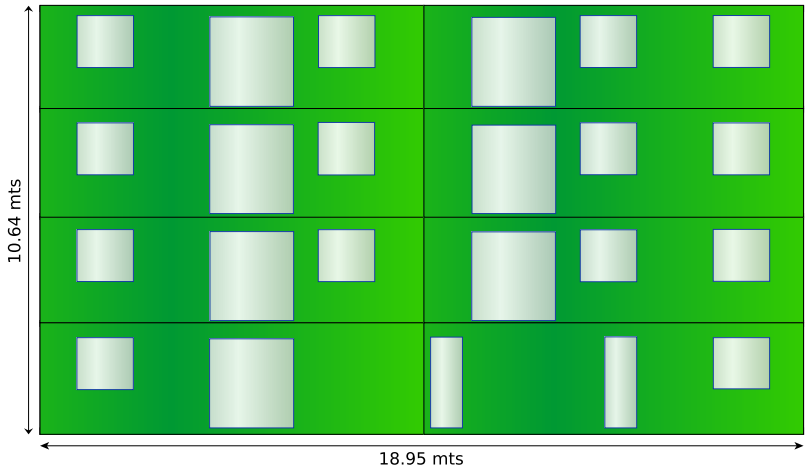


Figure 3.53 – Solution with $p_{wu} = 10$ meters and $p_{hu} = 3$ meters.

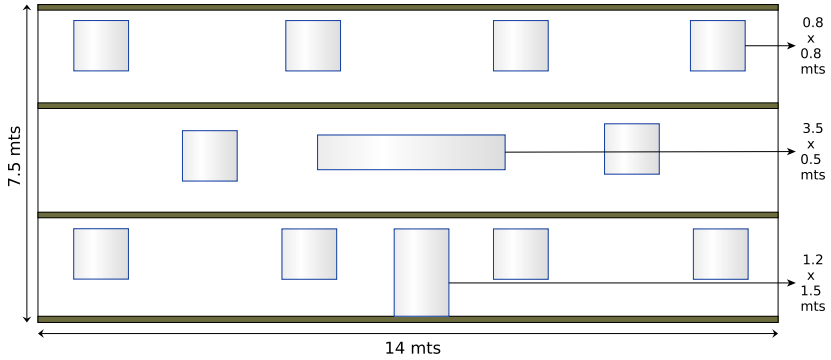


Figure 3.54 – Façade instance 3.

of vertical panels, the insulating envelope still has its vertical junctions aligned (symmetric appearance). Further, in the middle only 3 panels were designed thus preventing the length of junctions to increase with small panels. The envelope is compose of 7 panels and its length of junctions is 65.5 meters.

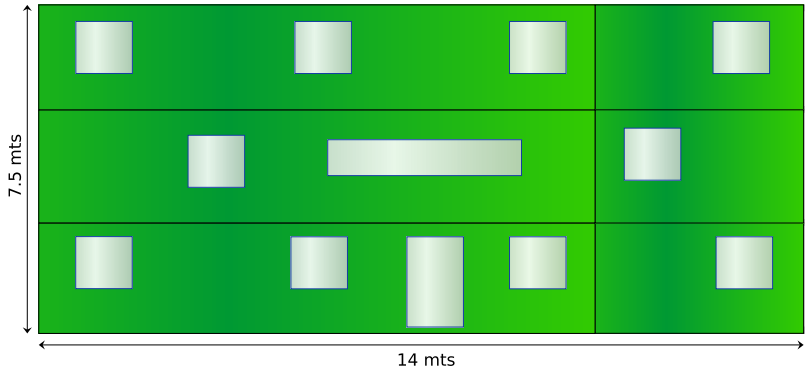


Figure 3.55 – Solution with $p_{wu} = 10$ meters and $p_{hu} = 3$ meters.

— Figure 3.56 illustrates an envelope solution made out with $p_{wu} = 10$ meters as width upper bound and $p_{hu} = 3$ meters as height upper bound, i.e., horizontally designed envelope. The envelope is compose of 6 panels and its length of junctions is 57.0 meters.

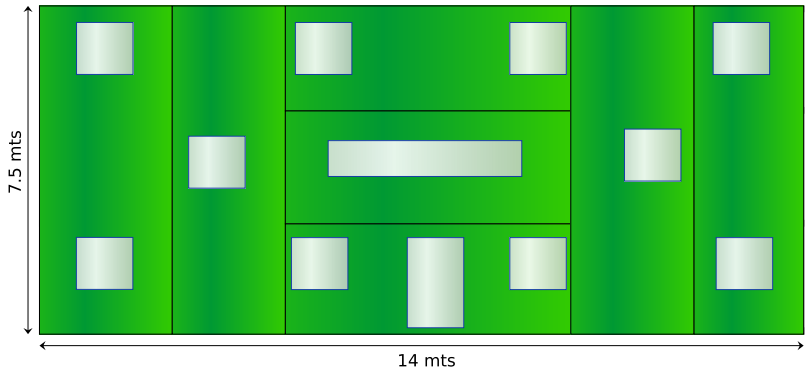


Figure 3.56 – Solution with $p_{wu} = 3$ meters and $p_{hu} = 10$ meters.

3.3.5 Discussion

In this section we have proposed a constraint-based heuristic approach, named CaSyE, that exploits façade geometrical structure and takes inspiration from the human aesthetic concept of symmetry. The motivation behind the cutting approach is the need for designing aesthetics insulating envelopes for façades in an automated fashion. Given that no standard definition of aesthetics exists, we have considered the junctions alignment (symmetry) as a good criterion. The heuristic is based on the well known technique of guillotines cuts and is able to generate up-to two envelopes. Then, it tries to artificially behave as a human being would behave in the sense that conflicts with frames are avoided while attempting to provide a pleasant appearance. The guillotine cuts approach allows to:

— Avoid constraint conflicts with frames as cuts traverses the façade only in zones free of frames.

— Given that cuts are made from one extreme of the façade to the other, panels junctions are aligned thus given a symmetric appearance.

— Once defined the columns or rows to place panels, it sets the final position and size according to the columns/rows size and the panel size upper bounds. This decision is made in order to minimize the number of used panels.

As well the GaLaS solution, the architect may use a generated insulating envelope as a good basis for designing a new solution: The relevant envelope can be therefore adapted and tuned by adding some tacit knowledge (really difficult to formalize), such as his/her own artistic flair, in order to turn the renovation more pleasing to the eye or give it an architectural expression. The heuristic uses the preferred orientation to design an envelope as a soft constraint and a hard constraint. It does so by starting the design with the preferred orientation making partitions of the façade (subfaçades) and changing the orientation when the portions of the envelope cannot be designed using the preferred orientation. Then, any partition on the façade has all its composing panels in a given orientation (hard constraint) but the complete envelope may have panels (partitions) in any orientation (soft constraint). This solution, unlike the greedy GaLaS one, considers the geometry and structure of the façade to execute guillotine cuts in non-conflictive areas and thus it is not "blind". Note, however, that the CaSyE algorithm follows the same local optimal decisions as the GaLaS one: The phase 2, columns/rows generation, is defined with respect to the panel size upper bounds and is limited only by the presence of

ZoFCos. Further, the phase 3, panels allocation, sets the final size of panels, again, with the maximum allowed panel size and is limited by the presence of frames and absence of supporting areas. Thus, the same principles that guide the GaLaS algorithm ("blind") are useful as well as for the junctions alignment (symmetrical) goal of the CaSyE algorithm (not "blind"). As the heuristic includes too the bottom-left preference, bigger panels are, potentially, designed at the bottom of the façade which is coherent with the on-site assembly process requirements.

The CaSyE algorithm, as well as the GaLaS one, seeks to minimize the number of panels by always trying to set their size in their maximum allowed. Our cutting heuristic provided more aesthetic solution than GaLaS and is a source for strong design aiding tool for the following reasons:

Reason 1: As well as the previous approaches (InDiE and GaLaS) it does not assume a fixed number of panels to create insulating envelopes but rather considers the geometry of the façade to design them.

Reason 2: Although no solution diversity is generated, it does provide aesthetics envelopes considered as junctions alignment (symmetric).

Reason 3: According to our scalability tests, the algorithm performs in competitive computational time.

We acknowledge that this approach, although not *"blind"*, can only generate up-to two insulating envelopes with the current scheme. Nonetheless, it is a fast, even when considering the façade geometry, and generates "pleasant" symmetrical insulating envelopes not generated by the GaLaS solution.

To conclude, we shall now compare the GaLaS and CaSyE algorithms in a more detailed way. Figure 3.57 presents the time spent to reach a first solution for each algorithm against the façade area. Due to the high number of solutions found by the GaLaS algorithm, the execution time is limited to 30 seconds. The results show that both algorithms behave in a competitive computational time as solutions for big instances are reached in less than 0.10 seconds.

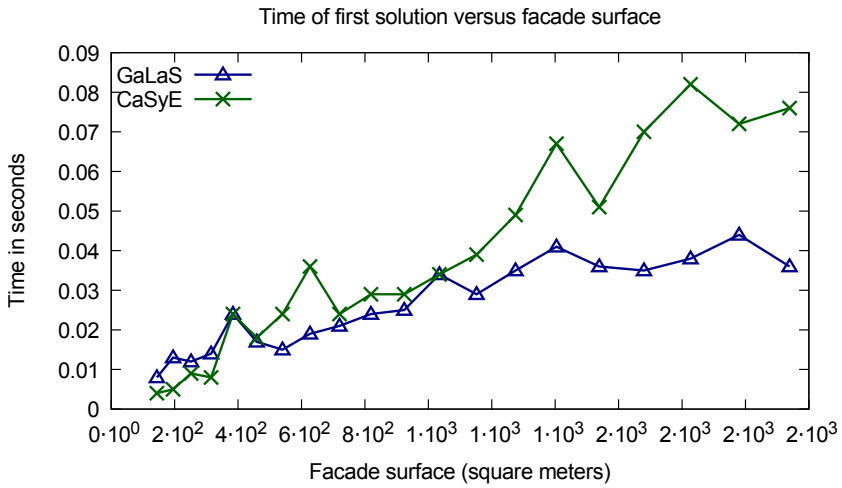


Figure 3.57 – Time to reach first solution versus façade area.

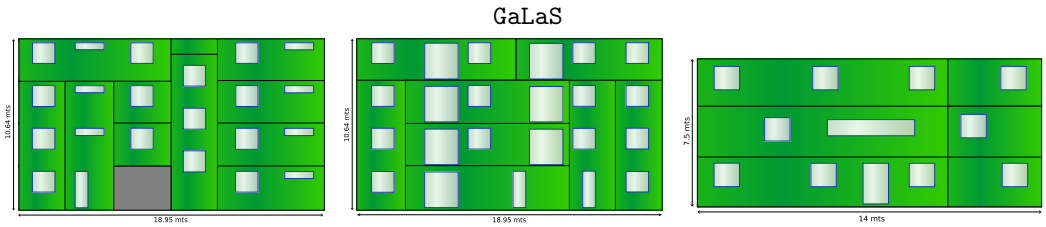
We do not show the comparison of number of solutions found by each algorithm as the CaSyE solution can only generate up-to two insulating envelopes. In other words, the GaLaS solution generates solution diversity whereas the CaSyE one does not.

Finally, let us compare the insulating envelopes thrown by GaLaS algorithm presented in Section 3.2 on page 56 with the CaSyE solution. Table 3.2 on the next page presents the results of both algorithms over the three façade instances. For convenience, we present in Figure 3.58 on the following page the best solutions generated by each algorithm.

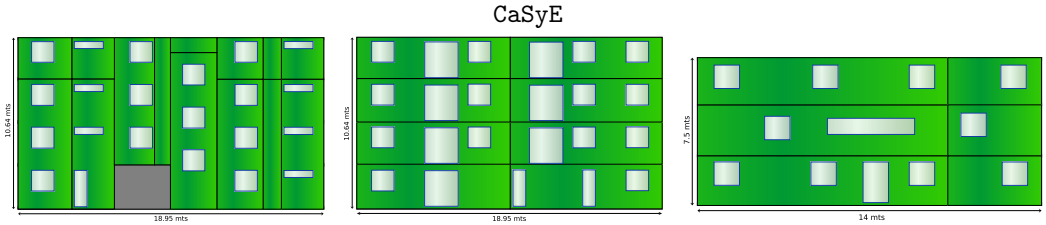
As the table shows, the greedy GaLaS algorithm is more robust than the CaSyE algorithm; it is faster and finds the best (close to optimal) solutions as it is able of changing panel orientation (soft constraint). The solutions of GaLaS, nonetheless, are not aesthetically pleasant as junctions alignment is not part of its capabilities and thus do not generate envelopes with aligned junctions. The CaSyE algorithm, on the other hand, generates envelopes with the highest number of panels, ergo largest length of junctions) but in return, it does generates aesthetics envelopes if junctions alignment (symmetry) is considered and it does so intentionally independent of the façade geometry. The algorithm, by contrast, is restricted in number of solutions.

Table 3.2 – Results comparison for GaLaS and CaSyE solutions.

		GaLaS			CaSyE	
		Panels orientation			Panels orientation	
		Vertical	Horizontal	V & H	Vertical	Horizontal
Façade #1	N	15	13	**11**	14	-
	Loj	113.2	106.9	**100.5**	108.2	-
Façade #2	N	13	**8**	**8**	16	**8**
	Loj	115.2	97.0	**95.6**	123.0	97.0
Façade #3	N	8	**6**	**6**	7	**6**
	Loj	63.8	57.0	**57.0**	65.5	**57.0**



(a) 11 panels (b) 8 panels (c) 6 panels



(d) 14 panels (e) 8 panels (f) 6 panels

Figure 3.58 – Best solutions for the three façades for GaLaS and CaSyE.

3.4 Digest

In this chapter, we have shown one manual interactive approach InDiE and two constraint-based heuristics, GaLaS and CaSyE, used to assist the design of insulating envelopes.

In a first step, we have introduced solution named InDiE for the a manual interactive design of insulating envelopes. The design process is performed by drawing one panel at a time over a GUI presenting a given façade. For each panel, the system executes several functions to validate each of the constraints in the model. If an ill-designed panel is drawn, i.e., a panel with constraint conflicts, the system visually informs by setting the panel's color to red. On the contrary, if a well-designed panel is drawn, the color is set to green. The architect then is guided by the system reactions that are performed in less than 100 milliseconds (real-time). Now, given that the InDiE implements constraint validation and not solving, drawn panels may be re-dimensioned and re-allocated while the system continues to inform about conflicts.

In a second step, we have presented our first automatic solution, called GaLaS. This constraint-based heuristic uses the knowledge extracted from the industrial case to make local decisions in the aim to arrive to close to optimal envelope designs. We have discussed the underlying scheme of the solution and explained our algorithmic design choices. In particular, we have highlighted the local (optimal) decisions for choosing the origin point (first available point at the bottom-left of the façade) and for choosing the size assigned to a given panel. Conversely, we have presented and discussed the internal of the algorithms in charge of solving constraint conflicts, which are bases as well in local (optimal) decisions. The solution treats the orientation of panels in two different setups. First, it tries to build an envelope design using only a given orientation but changes it, or performs backtrack, if the panel cannot be well-designed. In a second setup, we let the GaLaS algorithm to mix panel orientation intentionally, by using search trees, which has proven a good choice for designing envelopes with the lowest number of panels. The binary tree allows as well a great solution diversity.

In a third step, we have discussed the second automatic solution, named CaSyE, conceived to generate aesthetics insulating envelope designs. The constraint-based heuristic is built on the technique of guillotine cuts. It studies the geometrical composition of the façade to execute orthogonal cuts from one extreme of the façade to the other. We have studied the phases in which the solution is divided and discussed how similar local decisions as those ones used by GaLaS guide the design process in the three phases. In particular, the assignment of position and size for panels is done with respect to zones free of constraint conflicts, found by the guillotine cuts, and the maximum allowed size for panels. The preference for panel orientation is taken into account but, if the façade conditions blocks the panel orientation, the algorithm makes partitions of the subfaçade and design those partitions by changing orientation. This makes the solution slightly more robust. The envelope designs thrown by the CaSyE have a better aesthetic appearance as the junctions between panels are aligned.

In three years, Cyberdyne will become the largest supplier of military computer systems. All stealth bombers are upgraded with Cyberdyne computers, becoming fully unmanned. Afterwards, they fly with a perfect operational record. The Skynet Funding Bill is passed. The system goes online on August 4th, 1997. Human decisions are removed from strategic defense. Skynet begins to learn at a geometric rate. It becomes self-aware 2:14 AM, Eastern time, August 29th. In a panic, they try to pull the plug.

Terminator 2: Judgment Day
T-800 Terminator, 1991

4 Filtering-Based Design

Contents

In this chapter two filtering-based solutions for the insulating envelopes design are proposed. The motivation to develop two more solutions lies on the particular benefits provided by each one of them. The first filtering solution, named SkEdE, is in fact the second manual design solution. Here, the filtering and exploration capabilities of the constraint solver are used to solve a declarative model generated from a sketch specification manually drawn by the architect. The second filtering-based solution, called OpackS, is in fact the third automatic design solution. Here, filtering algorithms and a dedicated search heuristic have been developed to address the design problem as a constraint optimization problem. In order to understand the proposed solutions, a brief description of the constraint programming paradigm, filtering and search, introduces the chapter.

4.1 A Word on Constraint Programming

Constraint satisfaction problems, as discusses in Section 1.2.1 on page 5, may be tackled with different methods as linear programming, genetic algorithms or local search and may be adapted to be solved by SAT (abbreviation of Boolean Satisfiability Problem) and ASP (Answer Set Programming) methods, among others. A particular well-suited method for solving CSP, the one used by the solutions in this chapter, is Constraint Programming (CP).

Constraint programming has been recognized as a powerful computer-programming paradigm for solving combinatorial problems. It encompasses a wide body of techniques from computer science, artificial intelligence, graph theory, operational research and more (Rossi et al., 2006). Although it has its origins on image processing applications, on the very first paper dedicated to constraint programming (Montanari, 1974), it has been used on almost all branches of the industry, from organizing social events (Brailsford et al., 1996, Dotú and Van Hentenryck, 2005) and machine scheduling (Baptiste et al., 2012, Lombardi and Milano, 2012) to academic papers assignment (Aranda et al., 2005) and bio-informatics (Barahona et al., 2011).

Usually, constraint programming is divided in three major notions; constraint variables and domains, filtering and search. These programming notions are often encapsulated in a programming environment or toolkit that allows a fast application development. Some of these environments implement a mixture of mathematical models, such as `Mozart-Oz` and `IBM CP optimizer`, and many more solely dedicated to CP, such as `Gecode`, `Choco` and `ECLiPSe` among others. These CP environments are referred to as constraint *solvers* or constraint *engines* (Schulte, 2002). Constraint solvers used to tackle a wide spectrum of problems (i.e., independent of the any problem particularities) are known as *general purpose solvers*. These solvers provide abstractions for declaring constraint variables, posting constraints over those variables and applying search procedures. Given that, this dissertation exploits the capabilities of a constraint solver with search capabilities, namely `Choco` version 3 (Prud'homme et al., 2014), the three notions are further discussed in what follows.

4.1.1 Constraint variables & domains

Decision variables in a constraint solver are known as constraint variables and represent the objects of the problem. Domains are the values to be, potentially, assigned to the variables. These domains may take different types of representation whose most emblematic are discrete domains (or integer domains), continuous domains (or real domains), set domains and boolean domains. Variables and domains are then closely tied in such a way that every variable is declared with a given domain type. Further modification of the domain type is not allowed due to the declarative view of the paradigm. The ultimate goal of solving a constraint model is to assign to each variable a *single value* from its domain. Solvers do so by repeatedly applying filtering of incompatible values and search.

4.1.2 Filtering

At the core of a constraint engine is a constraint. A constraint expresses partial information among the variables the constraint holds on. For instance, the first order formula $T_1 \geq T2 + 2$ states a constraint over the domains of T_1 and T_2. Semantically, it could express "the team T_1 must have at least 2 more members than team T_2" or maybe "the task T_1 should start 2 time units after the task T_2". Constraints are implemented by domain-dependent filtering algorithms which aim at removing values from variable domains that are not compatible with the constraint semantics. These algorithms implement one or several notions of consistency: The consistency indicates how well it filters values without loosing valid solutions whilst removing values as much as possible. The filtering process is known as *constraint propagation*, for short *propagation*.

Typically, general purpose solvers implement two types of constraints (propagators) that are used in a wide range of applications. The first ones are the conventional constraints, or atomic

constraints, such as $<, \leq, =, \neq, >, \geq$. The second ones are called *global constraints* and are seen as a composition of conventional constraints:

> "A global constraint is a constraint that captures a relation between a non-fixed number of variables. An example is the constraint `alldifferent`$(x_1,...,x_n)$, which specifies that the values assigned to the variables $x_1,...,x_n$ must be pairwise distinct. Typically, a global constraint is semantically redundant in the sense that the same relation can be expressed as the conjunction of several simpler constraints. Having shorthands for frequently recurring patterns clearly simplifies the programming task. What may be less obvious is that global constraints also facilitate the work of the constraint solver by providing it with a better view of the structure of the problem." (cf. van Hoeve and Katriel, 2006, p. 205)

The dissertation makes uses of several global constraints from `Choco`, such as arithmetic constraints of the form $x + y < z$ and ordering constraints (as discussed in ordering constraint—2.12 on page 41 in Section 2.2.3 on page 38). A representative global constraint used to tackle `Cutting & Packing` problems, and used by the solutions in this chapter, is the non-overlapping constraint, called in most CP environments as `DiffN`, which states that variables are pairwise **different** in at least one of **N** dimensions, with $N \geq 1$.

Other kinds of global constraints used in the present chapter builds on the notion of *reification* (Beldiceanu et al., 2013). Reification of a constraint works as follows. Suppose a constraint C stating that $x \leq z + 2$. Also, suppose a boolean decision variable B with domain $[0,1]$. A reification may be used to express the fact that posting the constraint C is dependent on the value of B ($B \rightarrow x \leq z + 2$). Then, if $B = 1$ the constraint C is posted. The reification may work in the other sense, meaning that the true value of B is dependent on the satisfaction of C ($x \leq z + 2 \rightarrow B$). Then, if the constraint is satisfied, the variable B will be assigned the value 1. Using this reification notions is possible to define *logical constraints* such as `and`, `or`, `if` and `ifThen`. These logical constraints, implemented by `Choco`, enhance the expressive power of the language (Jefferson et al., 2010).

Finally, CP solvers users may be given the option to implement new constraints for specific applications. They do so by defining the filtering algorithm and clearly stating when a constraint is redundant in the problem (entailed) and thus may be discarded. In Section 4.3 on page 106 we take our `Cutting & Packing` problem structure to implement specific propagators for some constraint in our model.

4.1.3 Search

At the other end of the spectrum, solvers provide well-known search procedures (such as Large Neighborhood Search (Pisinger and Ropke, 2010), First Fail (Haralick and Elliott, 1980, Beck et al., 2005) and Depth-First Search (Cormen et al., 2009b)) that are interleaved with filtering in order to arrive rapidly to solutions. Search procedures are used because, habitually, the filtering is not powerful enough to find CSP solutions (Barták, 1999). Combining filtering and search is then necessary to solve most problem instances. In essence, a search procedure traverses the solution space by dividing the problem into subproblems thus generating a search tree. The way the search traverses the tree depends on the speculation on decision variables. One of the most basic search heuristics considers a binary search tree to assure a complete search. This means that the problem is divided into two subproblems. This process is often called branching. The following description depicts one branching possibility by considering a single constraint variable and a single domain value. Figure 4.1 and Table 4.2 present, respectively, an illustration of the search tree and the variable domain after propagation (node 4 is a solution).

1. Execute constraint propagation to filter incompatible domain values,
2. choose one variable cst_v from the set of unassigned variables,
3. choose a value val_v from the values in the current domain of cst_v,
4. create two subproblems with:

— the variable assigned to the value, i.e., $cst_v = val_v$,

— the variable different to the value, i.e., $cst_v \neq val_v$,

5. for each of the subproblems, start from Step 1.

As expected, creating two subproblems (assigning a value and its negation) makes the search complete, i.e., no solutions are lost in search although it is time consuming.

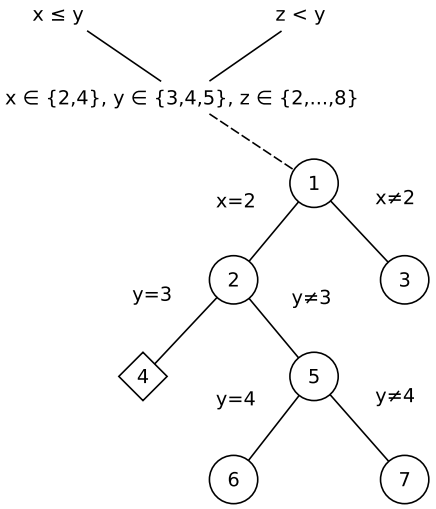


Figure 4.1 – Binary branching.

Node	x	y	z
1	$\{2,4\}$	$\{3,4,5\}$	$\{2,...,4\}$
2	2	$\{3,4,5\}$	$\{2,...,4\}$
3	4	$\{4,5\}$	$\{2,...,4\}$
4	2	3	2
5	2	$\{4,5\}$	$\{2,...,4\}$
6	2	4	$\{2,3\}$
7	2	5	$\{2,...,4\}$

Table 4.2 – States of branching.

As a final note, it is worth mentioning that a CP environment may be conceived, or used, not to find solutions but rather to perform filtering. In fact, most solvers are capable of executing filtering without executing search; it depends on the methods provided by the solver. To the best of our knowledge, there exists only one environment solely dedicated to perform filtering; the filtering engine CoFiADe (Vareilles et al., 2012). It only performs filtering because it is conceived for the interactive support of decision-making processes and not to solve combinatorial problems. Then, one of the main applications of this filtering engine is interactive configuration.

4.2 Sketching Design: SkEdE

This section presents a filtering-based solution for the manual design of insulating envelopes.

4.2.1 Motivation

One of the earlies stages of design is that of sketching in which architects draw their initial ideas while making improvements in the process: *"Normally during the conceptual phase, designers quickly represent as many as possible different solutions in a short time. These are evaluated visually before exploring further possibilities"* (cf. Stuyver and Hennessey, 1995, p. 236).

Thus, sketching is one of the design key elements as it improves the quality of novel ideas (McKoy et al., 2001) as well as their quantity (Hernandez et al., 2014). This design phase has been subject to studies in the last 20 years or so. Consequences of such efforts are the improvement of techniques and methods within computer-aided design systems. This element is in particular useful to architects as their communication language is mostly visual. Architects in charge of designing insulating envelopes use these sketches as a departure solution point. They do so by considering the façade structure to design aesthetics patterns while respecting the set of constraint imposed by the façade and industrial conditions.

We have previously presented the first alternative for manual envelopes design that implements an interactive behavior (Section 3.1 on page 48). The second alternative for this manual design, presented in this section, is to generate an envelope from a hand-made architect sketch. This approach contrasts with the first one in two aspects. First, no interactive design is achieved during sketching but rather an envelope design as similar as the architect's sketch. And second, it uses an underlying constraint solver to tune the drawn panels by removing incompatible values from variable domains (constraint propagation) to reach well-designed panels as well as to generate compliant envelopes by applying search. Although the interactive design is lost, the design through complete sketch allows the architects' ideas to be free because constraints in this setup may be violated (panels larger than size upper bounds, frames partial overlapping, panels overlapping and holes). Further, the solution is fully declarative in the sense that no specialized filtering algorithms or dedicated search heuristics are needed but only a constraint solver application programming interface (API).

The input of the algorithmic solution is a computer-based sketch, possibly with ill-designed panels, and its output is an insulating envelope, with well-designed panels, with smaller or equal size of those of the original sketch, and possibly holes. The filtering-based sketching approach is illustrated in Figure 4.2 on the following page.

4.2.2 Scheme

To understand the constraint sketching solution, it is important to know the traditional architects design process of sketching. The traditional sketching process is as follows:

Step 1: The architect makes a rough drawing of the façade on a paper sheet with a pencil.

Step 2: The architect analyses the façade and divides it into vertical and horizontal areas where the panels will be located.He does so in regard to the façade geometry.

Step 3: They make use of the vertical and horizontal areas previously drawn, to design panels by setting their exact position and size.

Step 4: Iterates steps 2 and 3 until satisfaction.

As the reader can remark, the process finishes when the architect is *satisfied* with the sketch, or if he/she wants to start over, meaning that panels in the sketch do not need to cover all the façade surface: An exploratory idea may involve few panels and holes between them. Once the architect has finished her/his sketch, she/he and a structural engineer must make sure that all constraints are satisfied. In particular, they must adjust the panels size in order to respect the manufacturers size limitations and the maximum load bearing capabilities of the façade.

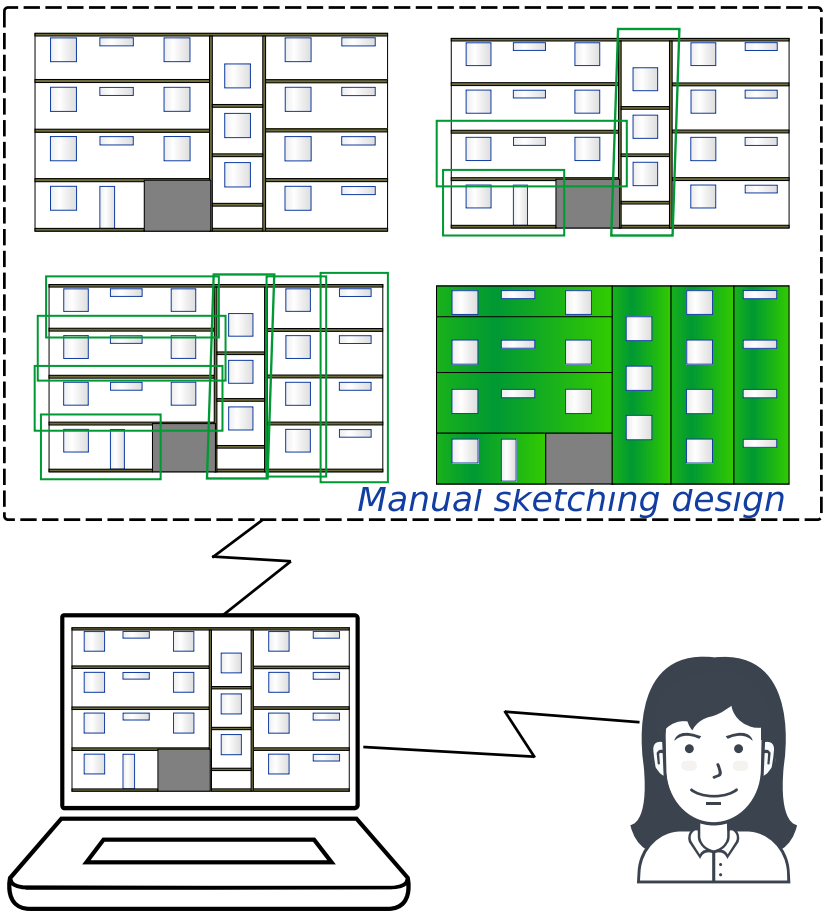


Figure 4.2 – Concept: Filtering-based sketching.

After tuning the sketch, the length of junctions is computed. Finally, a numerical model of the insulating envelope must be mapped from the sketch: The model of the insulating envelope needed by the manufacturer. Conventionally, the sketching task has been made using pen-and-paper (Tversky, 2012). However, CAD systems have replaced this paper sketching in different industry sectors (Suwa and Tversky, 1996). Our purpose is to support the design of insulation envelopes by replacing the paper sketching with a (constraint) computer-based sketch.

Considering the conditions when using a GUI and a computer mouse or drawing pens, steps 1, 2 and 3 of the traditional sketching process are changed. First, as the system already has a numerical model of the façade product of the Stage 1 of renovation process, the façade is no longer drawn by the architect but by the system. And second, the definition of the sketch is no longer achieved by vertical and horizontal lines but rather by precise rectangular panels drawn with the cursor. The tasks of the architect is then to draw panels over the façade presented by the GUI. Consequently, the input for the filtering-based solution of sketching is a) the façade and panels related information (sizes, bounds, minimum distances, etc.) and the position and size of each of the panels, possibly ill-designed, on the architect sketch. The output of the system is a set of well-designed panels that resemble as much as possible to those of the original sketch. As an invariant, and given that filtering is a monotonic process, the resulting well-designed panels are always smaller or equal than the drawn panels.

Before going any further, let us analyze the behavior of the filtering-based sketching for one panel sketch. Here, let us assume that the size constraint (2.4 on page 38) is the only one

taking part in the design process. Figure 4.3.a shows the façade and a panel drawn by the architect. A CP representation of this (potentially) ill-designed panel is created in the filtering-based sketching. Then, as a CP representation, the position (p_{x0}, p_{y0}) and size (p_w, p_h) are constraint variables with given domains defined by the current position (a,c) and size (b-a,d-c) of the drawn panel. Simply stated, the panel is only defined by bounded decision variables.

At this point the panel may be smaller (in one or both dimensions) than the panels' lower bounds or bigger (in one or both dimensions) than the panel's upper bounds, i.e., an ill-designed panel w.r.t. size constraint. If at least one dimension of the panel is smaller than the lower bounds in that dimension, then the panel has no correct design under filtering-based sketching. This is because the declarative view of CP prevents to add values to variable domains (constraint filtering is monotonically decreasing) and thus the generated panel cannot be bigger than the drawn panel. For all other cases, it is the responsibility of the constraint solver to prevent the size constraint conflicts by applying filtering over the size of the panel.

Figure 4.3.b shows the panel representation after applying the size constraint. Here, however, after applying the constraint, the panel is not yet assigned as it has not a fixed position and neither a fixed size: The panels' size bounds are consistent with the size constraint ($p_w = [p_{wl}, min(b-a, p_{wu})]$, $p_w = [p_{hl}, min(d-c, p_{hu})]$) [1] but different sizes, as many as its domain size, can be assigned to it and consequently different positions ($p_{x0} in [a, b-p_{wl}]$, $p_{y0} in [c, d-p_{hl}]$). Figure 4.3.c shows two possible panel designs that are completely consistent w.r.t. the size constraint. Each of them, and many more, is reached by applying search over the decision variables. Now, from a renovation point of view, the large panel is preferred over the small one and thus the search heuristic should explore the solution space by considering first the panels' size upper bounds that will, plausibly, generate the biggest panel area. We will discuss later this exploration preference.

According to the above description, we have divided the filtering-based sketching in three phases: (a) drawing of panels, (b) constraint posting and filtering and, (c) solving by applying a search procedure. To further understand these phases, we study again one panel sketch but including the rest of the constraints of the model (except for the non-overlapping constraint, for evident reasons).

Phase a) Drawing one panel. In the phase of drawing, none of the constraints needs to be respected so to not limit the aesthetic flair of the architect (although it is limited by the graphical user interface capabilities). In Figure 4.4.a, for instance, the architect has drawn a panel that does not fulfill any of the requirements: The panel is out of the façade area, it partially overlaps windows and it is as big as the architect wants regardless of the panel size limitations. All these constraint conflicts are latter solved by the filtering process in the phase b.

Phase b) Constraint posting and filtering over one panel. In the filtering phase all constraint conflicts, except that of panels overlapping, are solved by exploiting global constraints of the underlying constraint solver. In Figure 4.4.b, for example, the width of the panel has been reduced in such a way that p_{x0} and p_{y0} remain inside the façade surface and in such a way that the windows are no longer partially overlapped. The same occurs in the vertical axis. Note here that the panel keeps its relative position to the façade as it was originally conceived. In other words, instead of moving the value of p_{x0} while keeping its original size, the domain of p_w has been reduced. Note also that more panel designs are possible because, as discussed before, the panel is not yet assigned.

Phase c) Solving over one panel. Finally, the solving phase takes into account the CP representation of all drawn panels by using the search procedure to assigns the final value to decision variables. A set of panel designs may be obtained, for instance, by assigning to the panel's width any value between its current lower bound and upper bound. These different panel designs are very similar but, at the practical level, are in fact different insulating panel designs.

Before discussing the details of the solution space exploration, let us now analyze the filtering-based sketching over several panels sketch. In this case, the drawing and filtering phase remain

1. Note that the upper bounds for the panels size must be consistent with the size of the drawn panel and with the size constraint and thus the minimum of these values is chosen.

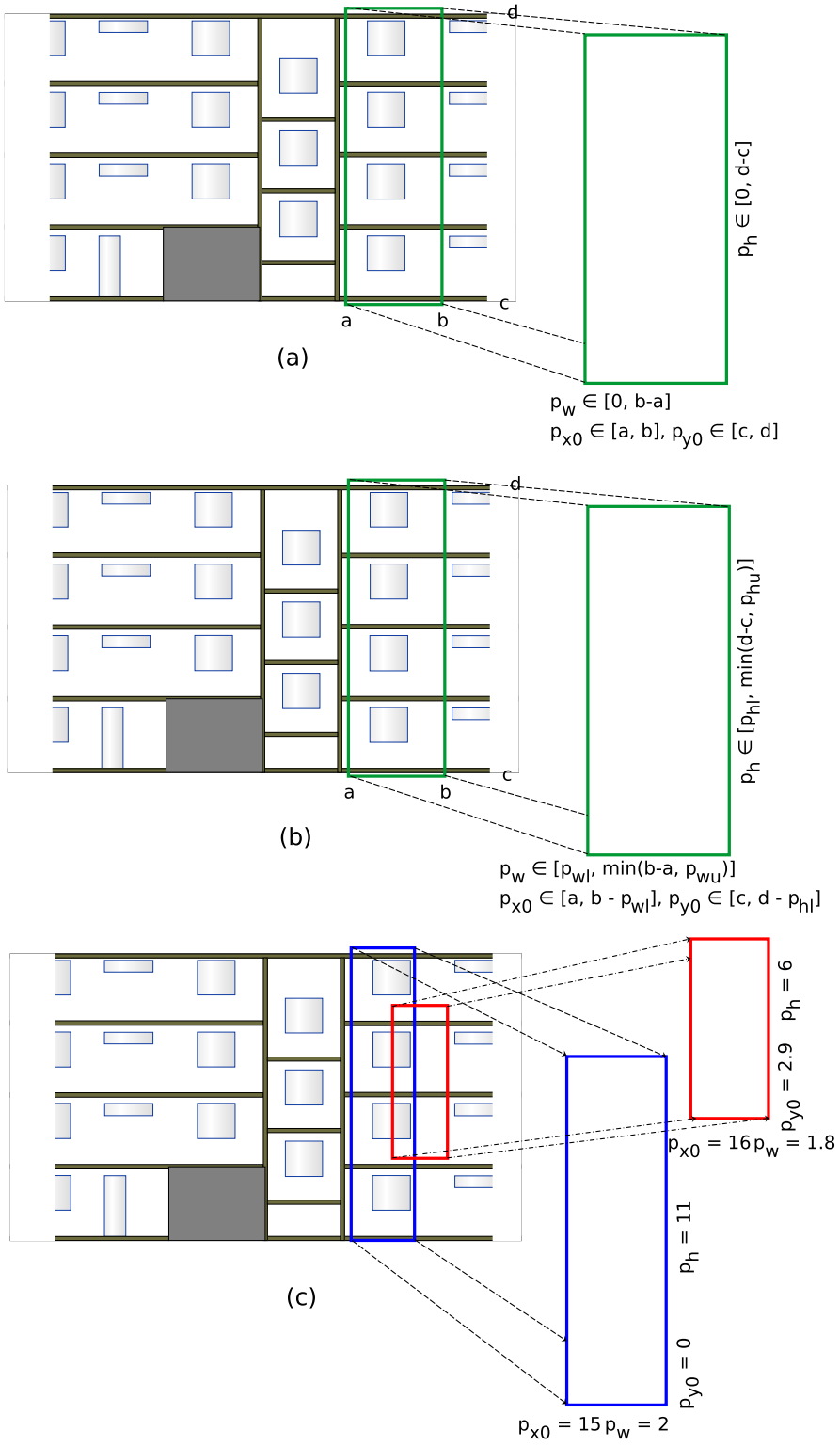
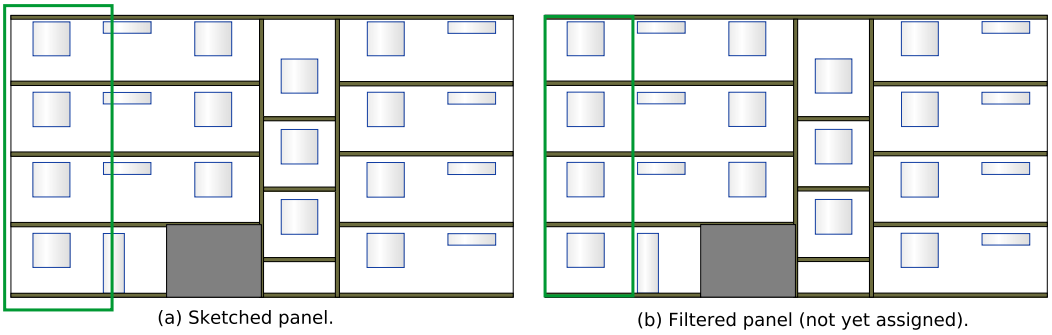


Figure 4.3 – CP representation of architects' drawn panel.

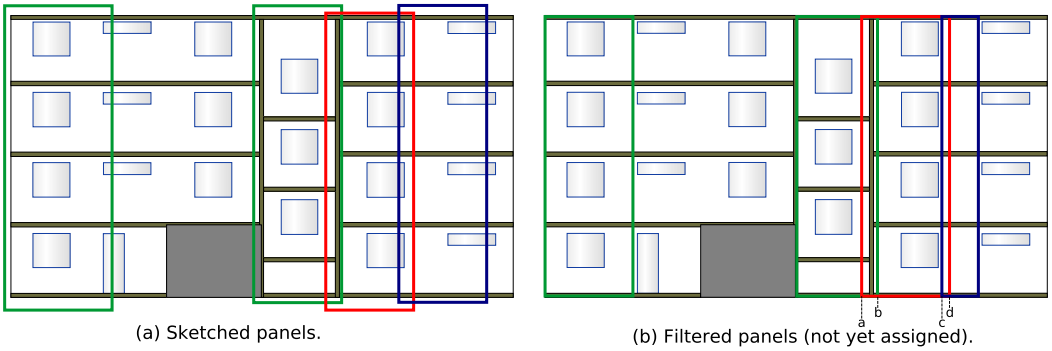


(a) Sketched panel. (b) Filtered panel (not yet assigned).

Figure 4.4 – a) Drawing phase and b) filtering phase for one panel sketch.

the same although it now includes one additional constraint, namely, non-overlapping (Beldiceanu et al., 2011).

Figure 4.5 presents an illustration when sketching four panels.



(a) Sketched panels. (b) Filtered panels (not yet assigned).

Figure 4.5 – a) Drawing phase and b) filtering phase for four panels sketch.

Filtering of incompatible values (phase b) makes each panel be inside the façade surface, avoid frames partial overlapping and force corners to be included in supporting areas (see Figure 4.5.b). Nonetheless, as the panels are not yet assigned, the filtering of the non-overlapping constraint only removes values from domains that are incompatible with the current not assigned position. It may finish its work only when the panels begin to be assigned one by one; once a panel is assigned the non-overlapping filtering algorithm can prune domains of adjacent panels. Then, the final position avoiding panels overlapping is achieved by applying a search procedure (phase c). The intersection between two panels may be used to present the user a set of compliant solutions: Each solution generated by iterating over an intersected interval. We, by contrast, are only interested in the first solution which, as we will see in what follows, is the one with the largest panels areas.

The search heuristic, that assigns the final panels sizes and positions, cannot make use of the objective of minimizing number of panels as it is a fixed value in this setup. So, to reach a solution as similar to the original sketch, we introduces an intermediary constraint variable for the panel area: $p_{area} = p_w \times p_h$. This variable is used in the search strategy by choosing as branching value the upper bound of the variable domain (strategy implemented in most CP environments). In consequence, the first solution, where all panels have a fixed position and size, has panels as large as possible and thus resembles as much as possible as those designed by the architect.

Given that constraint posting is a monotonic operation, panels may be drawn in any order

causing no conflicts. However, the order in which panels are drawn may affect the final output. Be aware that panels are stored, in standard data structures (e.g., lists), in the order they were drawn. When the search procedure is applied to a list of variables, for instance the areas, it selects one variable and one value of its domain to create two subproblems (as explained in Section 4.1 on page 94). The chosen variable is always the first unassigned variable of the list. This means that the first panel to be assigned a position and a size, by filtering and search, is the first panel drawn by the architect; a policy "first-drawn first-served". Now, an envelope sketch may be designed with different panel order, for instance, one design starts with panels drawn at the left of the façade and a second design starts with panels drawn at the right of the façade. Ergo, due to the first-drawn first-served policy, drawing panels in different order will, potentially, generate different envelopes. This behavior may be changed by ordering the panels with respect to the position of p_{x0}, for example. However, we consider the first-drawn first-served a good policy to allow more diversity in the resulting insulating envelopes and to respect architects choices.

4.2.3 Implementation

In this section, we illustrate how to support architects sketching by exploiting the declarative nature of CP and show that practitioners without any expertise on CP can implement a CAD with a CP environment and its application programming interface (API). Simply put, the implementation may use any general purpose CP environment, for example `Choco` and `Gecode`, as black-box solver. Given the reduced global constraints in the continuous domain module of most CP environments, the real representation may be mapped into integer representation by simple arithmetic operations. The implementation is as follows.

Size constraint. Constraint variables declaration in a given domain. Each variable is instantiated according to its semantics. For example, the domain of panels width may be instantiated with $p_w = [0.5, 13]$ meters. Domains may contain holes, which is a capability provided by several solvers (such as `Choco`, `Gecode` (Schulte et al., 2010) and `Mozart-Oz` (Schulte et al., 1998)). In addition, using reified constraints, we can restrict the possible combinations between the panels width and height. In essence, the conditions state that if one dimension takes the larger possible value ($max^{\uparrow}$), then the other dimension should be constrained to the smaller domain ($max^{\downarrow}$).

Non-overlapping constraint. Ensured with the `DiffN` global constraint (Beldiceanu et al., 2011). Although this is one of the most complex relations in the model, the filtering is very "ligth" as most of the panels have conflicts only with the adjacent ones. This means that from the beginning of the filtering the constraint is entailed for the majority of pairs p^i-p^j.

Installation and weight constraint. This constraint can be implemented using logical constraints. For instance, fixing a panel's p_{x0} one may constraint the corresponding p_{y0} as illustrated in Figure 4.6 on the next page. The implementation then constrains each rectangle against each supporting area. As a final step, the matching is forced by telling the solver that either p_{x0} or p_{y0} is in a supporting area.

Frames and interference constraint. Conversely, the mandatory overlapping of frames is achieved using a logical constraint. In this case, the condition states that if the panel is partially overlapping a frame then one of its dimensions (width or height) must be reduced in order to solve the conflict.

Area constraint. This constraint is ignored in the sketching model. Likewise the interactive solution presented in Section 3.1 on page 48, when doing manual design the architects may left

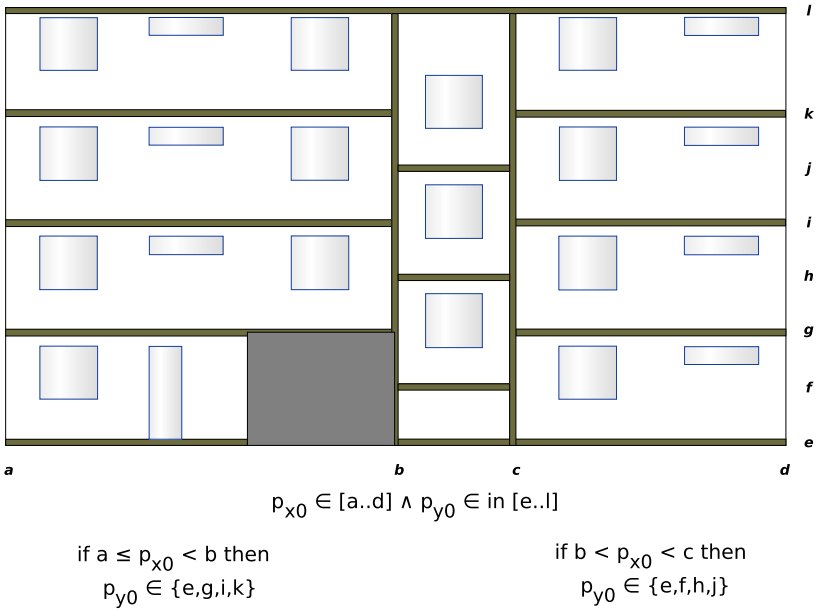


Figure 4.6 – Illustrative example of matching panel corners with logical constraints.

holes in the envelope as it may be beneficial for the artistic process. Adding the constraint, however, may be a good choice to differentiate between fully compliant solutions and not compliant ones.

4.2.4 Evaluation Case

In this section we present one example of envelope design using manual architects sketching. Figure 4.7 on the next page presents an illustrative sketch over the façade presented in 2.14 on page 44. Panels size upper bounds have been set in $[1,13]$ meters for one dimension and $[1,3.5]$ meters for the other dimension. In the sketch, the seven panels have been drawn from the left of the façade to the right and from the bottom to the top.

As the reader can see, constraint may be violated in the sketch. The underlying constraint handling of the `Choco` solver version 3 will resolve conflicts in the filtering phase. Recall that the solving process tries to set panels as large as possible following the order the user draws them. After asking the solver to find all solutions, the system throws several results, among them the following:

Figure 4.8 on the next page shows the first solution thrown by the support system. In it, the panels are designed in their maximum allowed size while respecting the constraints. The length of junctions of the envelopes is 76.80 meters.

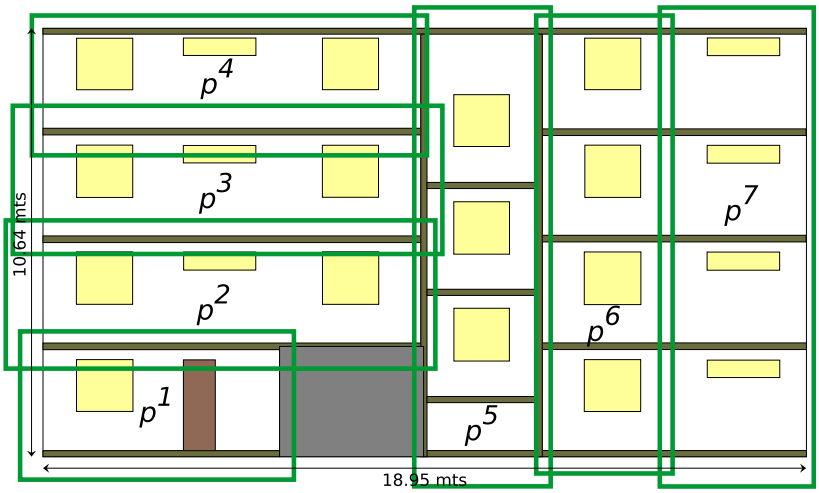


Figure 4.7 – Architects sketching.

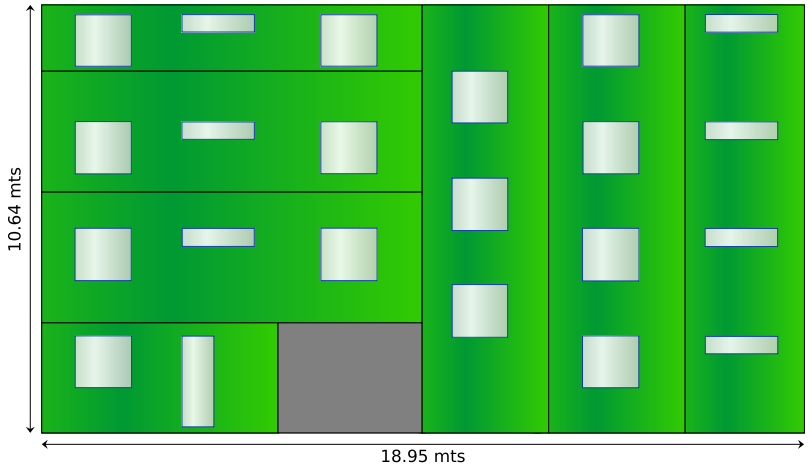


Figure 4.8 – First resulting compliant insulating envelope from sketch.

4.2.5 Discussion

In this section we have described our experience on constructing a CAD system for the design problem of designing façades insulating envelopes. The system uses declarative nature of CP to assist architects conception phase of sketching then improving the design process. Then we have given the motivation behind a CAD for supporting the architects sketching of insulating envelopes, described the traditional pen-and-paper sketching that most architects follow, illustrated a computer-based sketching that is supported by the filtering capabilities of a constraint solver and we have shown how the CAD can be constructed having a constraint solver and its respective API.

The `SkEdE` solution, as well as the `InDiE` one, seeks to assist the manual design of panels (respectively envelopes) and exploits the capabilities of a GUI to achieve the goal. Then, both approaches promotes the architects creativity by let him/her to design panels following its own artistic flair. In addition, the architect's hand-made sketch may contain holes given the opportunity, as well as the interactive design `InDiE`, to generate partially designed envelopes and send

them to automatic solutions for completion. Nevertheless, they present significant differences.

In contra position with the first manual solution InDiE that implements functions for each of the constraints in the model, the sketching design is a fully declarative solution of the CSP that is supported by the filtering algorithms and search strategies of a constraint solver (Choco). Secondly, whereas the interactive design of InDiE works in a panel-by-panel basis and informs for each panel the constraint conflicts (in different colors), the SkEdE solution allows the drawing of multiple panels and solves constraint conflicts for all of them in a given execution. Finally, the envelopes generated with the filtering-based sketch resemble as much as possible, from the point of view of size, to those ones drawn by the architects.

The use of CP technology and its declarative model is in such extend intuitive that the development of the CAD system focused mostly on the panel CP representation and the GUI capabilities and no adaptations were needed for the filtering and search phases. Our proposed computer-based sketching have been implemented with the Choco solver by referencing only its API. Then. we consider that, from the techniques offered by OR and AI, CP is the most intuitive out-of-the-box for modeling and solving. This highlights the importance of the great flexibility, expressive power and simplicity of the declarative model of CP.

4.3 Open Packing Design: OpackS

This section presents a filtering-based solution for the automatic design of insulating envelopes. A previous version of this section has been presented in Barco et al. (2015c).

4.3.1 Motivation

The essence of this solution is to generate optimal insulating envelopes with respect to the number of panels. It do so by applying the constraint programming approach of filtering and search. Here, however, the preferred orientation of panels is considered as a hard constraint, meaning that all panels in a given insulating envelope will have the same orientation. Then, this solution contrasts to the other automatic approaches (GaLaS and CaSyE), not only because it takes into account future states by applying a filtering to the panel variables but because it is more limited as it takes the orientation as a hard constraint. Nevertheless, it traverses the solution space (as explained in Section 4.1 on page 94) and thus performs a complete search under the orientation condition. An illustration of the concept is presented in Figure 4.9.

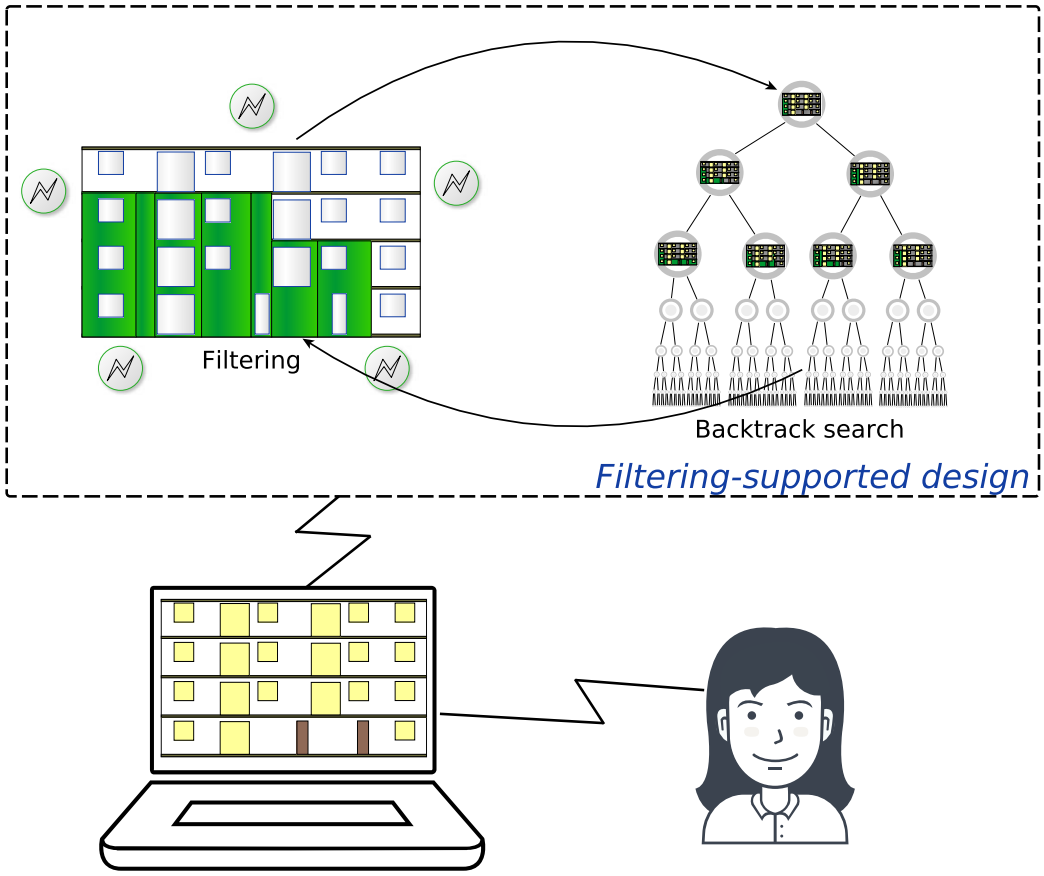


Figure 4.9 – Concept: Filtering-based design using backtrack search.

Now, unlike the sketching design, where the underlying solver Choco version 3 handles a defined set of decision variables coming from the envelope sketch, the filtering-based design does not count in advance with the set of panels (decision variables) to perform filtering and search. Not having a predefined number of panels becomes a drawback given that the great majority of CP environments implement global constraints and search strategies with a fixed set of input variables. To overcome this situation, it is then necessary to apply special notions as

dynamic creation of variables and constraints (Barták, 2003) or allow variables to be optional (Laborie, 2009). We have chosen to rely on the latter notion. Our approach then targets the problem of unfixed number of variables by speculating about their number and enabling them to be optional (as does the scheduling environment of IBM CP optimizer (Schutt et al., 2013) in one dimension).

We have implemented filtering algorithms (propagators in Choco) for several constraints in the model that allow us to reflect the fact that panels are optional in the solution. In addition, we have developed a dedicated search heuristic exploiting the problem structure to arrive rapidly to solutions. The search heuristic is based on the same greedy principle that guide the other automatic solutions: Extending the panel sizes in its maximum allowed size and starting the packing from the bottom-left corner. We show that our heuristic performs, under the orientation limitation, equal or better than the GaLaS and CaSyE solutions. Also, we show that our heuristic outperforms predefined search heuristics natively implemented in the underlying constraint solving, such as first-fail and activity search.

4.3.2 Scheme

In order to exploit the capabilities of a general purpose constraint solver, the problem of unfixed number of variables (panels) must be settled. Our work is inspired from van Hoeve and Régin (2006), where the authors introduce *open* global constraints. An open global constraint is an extension of an existing global constraint that includes a set variable (or an array of binary variables) to indicate the subset of decision variables the former constraint holds on. In other words, some decision variables of the problem become optional (see Laborie (2009), Schutt et al. (2013) and Section 4.4.16 in Schulte et al. (2010) for further information).

Optional panels

At first glance, we know that decision variables are related to panels (i.e., panels). But, to link variables to panels at a low level it is first necessary to speculate about the number of panels to be used in an insulation envelope. Thus, we first heuristically bound this number by setting a minimum and maximum number of panels that may be in a façade envelope. Computing the minimum and maximum number of panels that can be packed over a given façade is straightforward; it is a relation between the panels size bounds and the façade size. Let min denote an estimate of the minimum number of panels to cover the façade. Conversely, let max denote an estimate of the maximum number of panels to cover the façade. Given the lower and upper bounds for panels' size and the façade size, we consider:

$$min = \left\lfloor \frac{fac_w \times fac_h}{max^{\downarrow} \times max^{\uparrow}} \right\rfloor \qquad (4.1) \qquad\qquad max = \left\lceil \frac{fac_w \times fac_h}{min^{\downarrow} \times min^{\uparrow}} \right\rceil \qquad (4.2)$$

We then create a set of max *optional* panel variables, each one referring to a panel that may or may not belong to the solution. Using the maximum number of panels to create the set of potentially used panels represents the worst case scenario, i.e., each panel is bounded to the length lower bound in each dimension. As we will see later, the worst case scenario is never reached in the experiments and is not likely to be reached in real-life façades. Then, apart from the variables describing their position and size, each panel $1 \leq p^i \leq max$ is also described by its presence on the solution:

— $p^i_b \in \{0,1\}$ indicates whether or not the panel p^i is used in the solution.

Note that, as a panel is already defined by an array of integer variables (its coordinates and size), it is more natural to extend it with a fifth binary variable representing its presence in the solution than introducing a set variable to represent all present panels (van Hoeve and Régin, 2006). The total number of used panels is then given by

$$N = \sum_{i=1}^{max} p_b^i \tag{4.3}$$

Constraints Semantics (for optional panels)

In regard to optional panels setup, some of the constraints of the CSP model presented in Section 2.2.3 on page 38 are modified to include the boolean variable semantics. Also, an additional constraint for symmetry breaking of unused panels is added.

Size compatibility *This compatibility constraint links the width and the height of panels. For the constraint model of `OpackS`, we set this as a hard constraint, meaning that for a given execution the panel has an orientation that cannot be changed on run time.*

Area *Used panels area equals to façade area. It modifies constraint 2.6 on page 38 to reflect the fact that only used panels are relevant to cover the façade area.*

$$\sum_{i=1}^{max} \left(p_b^i \times p_w^i \times p_h^i\right) = fac_w \times fac_h \tag{4.4}$$

Non-Overlapping *Any pair of used panels p^u and p^v must not overlap in at least one dimension. It modifies constraint 2.7 on page 39*

$$\begin{aligned} \forall_{p^u,q^v} \mid p_b^u = p_b^v = 1,\ & p_{x0}^u \geq p_{x0}^v + p_w^v \vee p_{x0}^v \geq p_{x0}^u + p_w^u \\ \vee\ & p_{y0}^u \geq p_{y0}^v + p_h^v \vee p_{h0}^v \geq p_{h0}^u + p_h^u \end{aligned} \tag{4.5}$$

This corresponds to the *Open* variant (van Hoeve and Régin, 2006) of the `DiffN` (Beldiceanu et al., 2011) constraint, i.e. a generalization of `DiffN` to handle optional panels.

Ordering *Priority is given to use the first panels and used panels are ordered geometrically. It modifies constraint 2.12 on page 41.*

$$\texttt{LexChainLessEq}(\{\{(1-p_b^i), p_{x0}^i, p_{y0}^i\} | 1 \leq p^i < max\}) \tag{4.6}$$

As explained in Section 2.2.3 on page 38 for constraint 2.12 on page 41, the constraint imposes an order to panels in the aim of avoiding the generation of symmetrical solutions. In the open version, the boolean variable states that used panels come first than unused ones:

$$\begin{array}{ccccccccc} 1-(p_b^i=1) & \leq & 1-(p_b^i=1) & \leq & 1-(p_b^i=0) & \leq & \ldots & \leq & 1-(p_b^{max}=0) \\ \downarrow & & \downarrow & & \downarrow & & \ldots & & \downarrow \\ 0 & \leq & 0 & \leq & 1 & \leq & \ldots & \leq & 1 \end{array}$$

Unused *Unused panels are arbitrarily fixed. In order to avoid wasting time on unused panels, we may fix their origin variables to the first possible attachment point as well as its size variables to their minimum values.*

$$\forall p^i, 1 \leq p^i \leq max, \quad b_p^i = 0 \Rightarrow (p_{x0}^i = p_{x0}^i.LB \wedge p_{y0}^i = p_{y0}^i.LB \wedge p_w^i = p_w l \wedge p_h^i = p_h l) \tag{4.7}$$

Where LB stands for lower bound of the variable domain. This means that if a given panel is not going to be used $(p_b^i = 0)$, the position and size will no affect the resulting insulating envelope and thus its variables may be set deterministically.

Optimization

A COP may be constructed using the aforementioned constraint model and a given objective function. Thus, the result variable N may be turned into a decision variable and thus utilize it as optimization feature. For the problem at hand, the minimization of the number of panels has been chosen as it entails the length of junctions minimization. The number of panels is bounded by

$$min \leq N \leq max \tag{4.8}$$

The optimization then is $minimize(N)$.

4.3.3 Implementation

The previous stated scheme has been implemented with the constraint solver `Choco`. Some other constraints like the open variant of `DiffN` must be implemented by using the API of the solver. As well, to exploit the structure of the problem we have developed a dedicated search heuristic. It is worth mentioning that we use only integer representation and thus, the façade size and panel bounds are mapped from real to integer. For instance, a panel size upper bound of 8.5 meters is mapped into 850 centimeters.

Filtering

This section provides details on the model implementation. Basically, our solution follows the approach in van Hoeve and Régin (2006) to handle decision variables that are potentially in the solution. In our work, however, as panels are already represented of several integer decision variables, we found more natural to use an extra binary variable per panel instead of a set variable. Intuitively, an open constraint with boolean variables may be implemented following traditional filtering algorithms and may be enhanced by targeting the structure of the problem (van Hoeve and Régin, 2006).

An open constraint for panel non-overlapping As can be seen in the literature, the `OpenDiffN` constraint has already been implemented (see `No-Overlap` with optional panels in Section 4.4.16 in Schulte et al. (2010) for instance) but we consider necessary to provide a brief description of its behavior. The filtering algorithm of the `OpenDiffN` checks whether two panels that are part of the solution, i.e., whose b_i is equal 1, overlap and proceeds to domain filtering to prevent overlaps, as traditional `DiffN` propagators do. Conversely, if the overlapping of two panels is unavoidable, then domain filtering on the boolean variables ensures that at least one of the two panels is not used ($b_i = 0$). The overall filtering is strengthened by a constructive disjunction algorithm (Würtz and Müller, 1996, Hentenryck et al., 1998) that computes an attaching point for the bottom left corner of each panel, that is valid (from the packing point of view only) with respect to already fixed panels. In essence, it ensures that p_{x0} lower bound is feasible when branching on it and, if it is not the case then the problem comes from an earlier node in the search tree.

A logical (reified) constraint for installing panels In order to attach panels onto the façade, it is necessary that the four panel corners match supporting areas.

This constraint is implemented using logical constraints in the same way that it is done in the constraint-based sketching. Using the Figure 4.6 on page 103 in Section 4.2 on page 97 as example, suppose that $p_{x0} = a$ (left edge of façade), then possible locations for p_{y0} must be constrained to the horizontal points $\{e, g, i, k\}$ as it is in these points where supporting areas are located. If, by contrast, $b < p_{x0} < c$ then the value of p_{y0} must be constrained to $\{e, f, h, j\}$.

The implementation then constrains each panel against each supporting area. As a final step, the matching is forced by telling the solver that all corners are in supporting areas. This can

be done using the membership constraint that forces a decision variable to be part of a given domain.

A constraint dedicated to frame covering The constraint 2.8 on page 39, that ensures frames covered by one and only one panel, is propagated using a dedicated approach. The constraint has not been modified by the notion of optional panels but it is not natively implemented within the `Choco` solver. The filtering algorithm is pretty simple and works as follows: For every frame, two *support* panels (i.e. panels the frame fits in) are computed (illustration in Figure 4.10). In case no support panel is found then the solver fails, i.e., there is no more panels that may cover the frame. In case only a single panel is found, then a filtering procedure is applied to enforce it to cover the frame. Finally, in case two panels have been found, then no propagation is triggered because, at that time, we do not know which panel will be used to cover the frame.

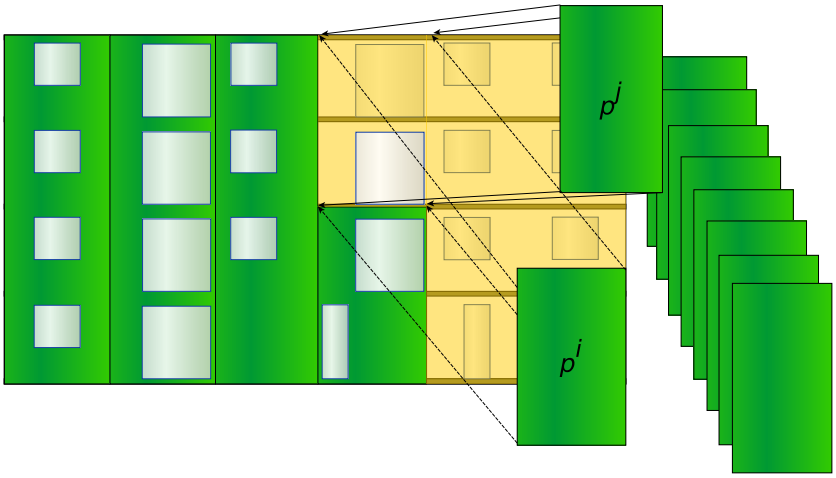


Figure 4.10 – Compute two panels to potentially cover a frame.

Embedding symmetry-breaking The lexicographic constraint (van Hoeve and Hooker, 2009) has a strong influence on the model. It enables to output *different* solutions, to reduce the size of the search tree but that is not all: It is possible to speed up the other global constraints by taking that information into account while filtering. For instance, any `for` loop seeking all used panels ($b_i = 1$) can stop as soon as one undetermined panel ($b_i = \{0, 1\}$) has been found because further panels are either undetermined or unused. Thus, it is possible to exploit the problem structure to improve the implemented constraints.

Search heuristic

The search heuristic is responsible of assigning panel's decision variables when propagation cannot infer more information. Our heuristic is described in Figure 4.11 on the facing page Algorithm 11. It is a constructive approach that considers each panel one by one and uses the following variable selection priority:

— b_i presence of the panel in the solution,

— p_{x0} position of left edge over the façade,

— p_{y0}, position of bottom edge over the façade

— p_w width of the panel and,

— p_h height of the panel.

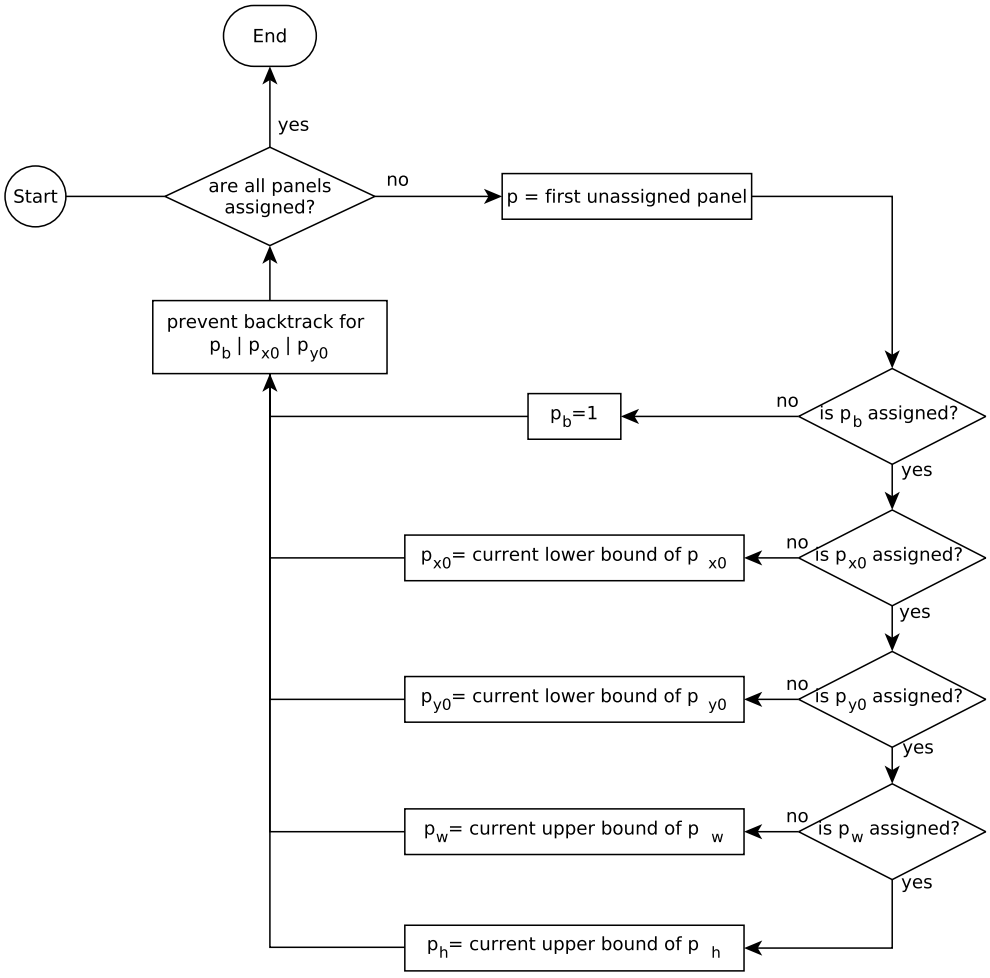


Figure 4.11 – Algorithm 11: Dedicated search heuristic.
***Inputs*:** Set of optional panels.
***Outputs*:** Variable to branch, and value to be assigned, a given node of the search tree.

We apply a traditional binary branching scheme over stated variables (Smith, 2005). It means that, instead of iterating over domain values, the heuristic assigns a value to a variable and removes that value from the variable domain upon backtracking. The heuristic then selects, for each node of Figure 4.12 on the next page, the pair variable-value with which the branching is performed.

The originality of our method is that some decisions cannot be negated, i.e., the negation are not alternatives for backtracking. The "prevent backtrack for" box in Algorithm 11 tells the solver not to try different values on failure for p_b, p_{x0} and p_{y0}. For instance, if $p_{x0} = 1$ and the node fails, it will not try to propagate $p_{x0} \neq 1$ and compute a new decision. Instead, it will backtrack once more (to the decision associated with the size of the previous panel). We do this because not setting p_{x0} in its lower bound means that the panel would not be adjacent to the already-placed panels which would result in a hole in the envelope. Also, given that panels are indistinguishable from each other, backtracking on $p_b = 0$ is not allowed: If a panel p^j is not going to be used, then the remaining potentially used panels ($p_b = [0, 1]$) are not going to used neither.

The heuristic implements the following key design choices:

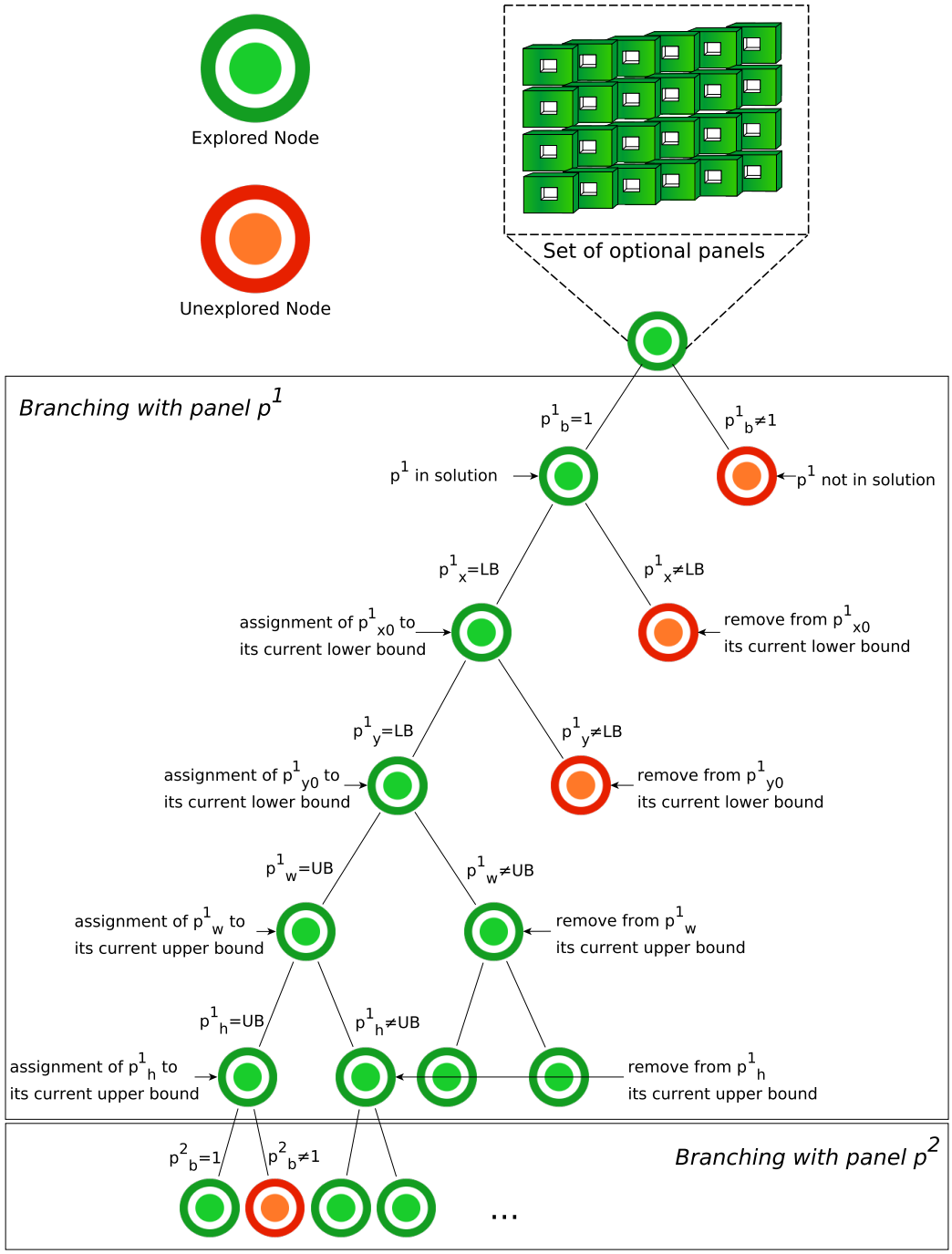


Figure 4.12 – Concept: Dedicated search heuristic.

1. We set the p_b variables to 1 in order to arrive rapidly to solutions.
2. The position variables p_{x0} and p_{y0} are fixed to their lower bounds in order to leave no uncovered places between the considered panel and previously placed panels. In short, the real decision variables are p_b, p_w and p_h. But p_{x0} and p_{y0} should be set in a deterministic way without backtracking. As panels are ordered, trying a larger value would lead to a hole on the façade, which is forbidden. Note that this is only possible because the filtering

is strong enough: The lower bound is indeed a valid support from the packing point of view.

3. The size variables p_w and p_h are set to their upper bounds in order to try panels as large as possible and thus cover the largest possible area. This enables to get a first solution that is very close to the optimal value (greedy decision).

Finally, as we explained, the backtrack is only allowed for the width and height. Then, referencing the Figure 4.12 on the facing page, the assignment $p_w = upperBound$ is applied to one node and $p_w \neq upperBound$ is applied to a second node. The variable in the latter node is not yet assigned as it has as domain many different values from which to choose. From the application perspective, the process over this variable has basically only removed 1cm from its domain. Further exploration will assign the variable to different size values.

4.3.4 Statistics & Examples

In this section are presented some scalability tests and examples of envelopes generated by the filtering-based solution `OpackS`. The underlying solver used in the tests is `Choco` (Prud'homme and Fages, 2013).

Scalability

2-step approach. In a first experiment we want to evaluate whether or not the maximum number of used panels is a good approximation of the optimum. Figure 4.13 on the next page presents the number of used panels and the number of optional panels for every instance. The maximum number of panels, which represents the worst case scenario where panels' size lower bounds are used, is never reached. Further, this maximum number is an upper bound far to high: For a façade of size 60×31 meters, the solver handles 2640 optional panels to compute a first solution that uses only 66 panels. This means that we create too many useless variables that will slow down the solving process. Therefore, in order to boost performance, we set up a 2-step approach:

Step 1: In a first step, we execute the model asking the solver to find one optimal solution. In this executing no time-limit is imposed.

Step 2: In a second step, we create a new model in which the maximum number of optional panels is replaced by the number of panels composing the envelope found in the previous step. Then, we enumerate all optimal solutions within a 30 seconds time window.

Impact of symmetry breaking. In a second experiment, we measure the impact of adding symmetry-breaking constraints. More precisely we compare the time to find a first solution and the number of computed solutions with and without symmetry-breaking constraints. Due to the huge amount of solutions, we use a time limit of five minutes.

Figure 4.14 on the following page shows the time to reach the first solution, with and without symmetry breaking constraints, when constraining the y-coordinate to match supporting areas.

As we can see it on Figure 4.14 on the next page, symmetry-breaking constraints speed up the search. Moreover, it enables to skip solutions that are identical for the end user. Using symmetry breaking constraints the execution time and used memory improves as the number of nodes and failures decrease.

Finally, Figure 4.15 on page 115 shows the number of solutions found in 30 seconds without and with symmetry breaking constraints. Note that the number of symmetrical solutions found is huge in comparison with the non-symmetrical solutions. Also, note that it seems like the number of solutions decreases when the façade area increases: this is due to the time limit. As the problem gets bigger, the solving process gets slower and enumerates less solutions in a given time window.

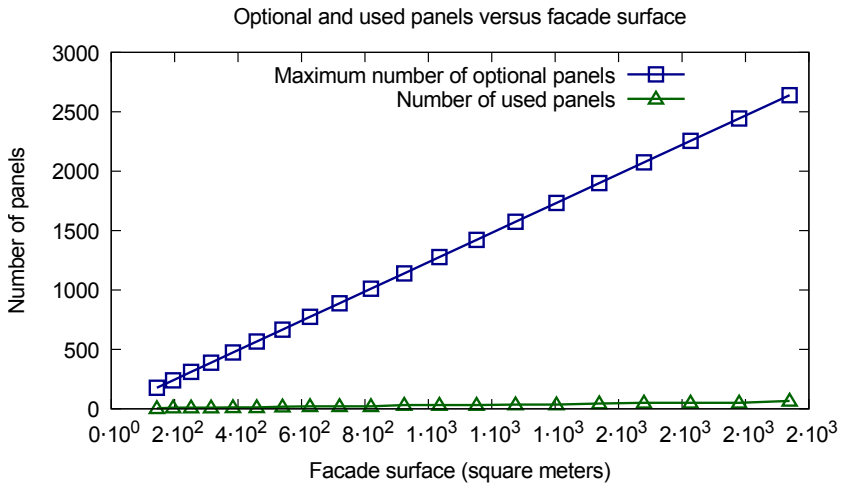


Figure 4.13 – Maximum number of optional panels and number of used panels in the first solution, for every instance.

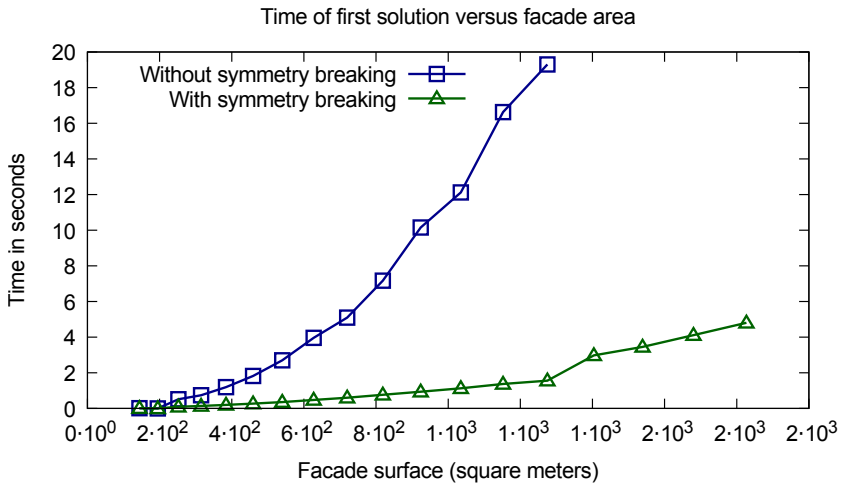


Figure 4.14 – Time to reach a first solution with and without symmetry breaking constraints.

Search comparison

In regard to different search heuristics, we have not found any well-suited for addressing the problem. Actually, well-known black-box search strategies such as domOverWDeg (Boussemart et al., 2004), impact-based search (Refalo, 2004) or activity-based search (Michel and Van Hentenryck, 2012), do not perform well given the problem particularities. These heuristics are designed to solve problems in a blind way, when we have no expert knowledge of the problem. In our case, we mix very different kind of variables (booleans, positions, sizes) that we are able to group by panels and order. Introducing randomness on either the variable selection or the value selection may be disastrous. In particular, using arbitrary values for p_{x0} and p_{y0} makes a huge amount of possibilities for uncovered places.

Given that the traditional search heuristics do not find solutions for the smallest façade (15×13 meters) with which the `GaLaS`, `CaSyE` and `OpackS` algorithms have been tested, we changed the representation to pixels in order to reach at least one solution with `Choco` predefined search heuristics. Also, we have increased the computing time from 30 seconds to 3 minutes. We have

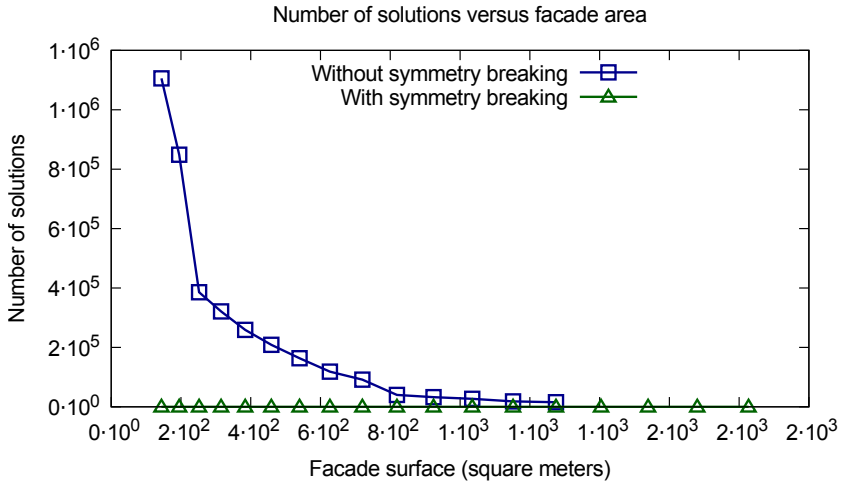


Figure 4.15 – Number of solutions found without and with symmetry breaking constraints.

tested 16 predefined heuristics from `Choco` on a small façade (400×100) with panel size bounds of $[20, 150]$ pixels in both dimensions. In consequence, whereas the initial façade representation has as width 1500 centimeters, domain taken by p_{x0}, the new representation has only 400 pixels. We present the results for those ones that threw at least one solution over a time window of 180 seconds. These strategies are:

— `domOverWDeg` which selects the variable with the smallest ratio $\frac{|d(x)|}{w(x)}$, where $|d(x)|$ denotes the domain size of a variable x and $w(x)$ its weighted degree;

— `lexico_LB` which chooses the first non-instantiated variable, and assigns it to its lower bound;

— `lexico_Split` which chooses the first non-instantiated variable, and removes the second half of its domain;

— `maxReg_LB` which chooses the non-instantiated variable with the largest difference between the two smallest values in its domain, and assigns it to its lower bound;

— `minDom_LB` which chooses the first non-instantiated variable with the smallest domain size, and assigns it to its lower bound and;

— `minDom_MidValue` which chooses the first non-instantiated variable with the smallest domain size, and assigns it to the value closest to the middle of its domain.

No results are shown without symmetry breaking constraints because predefined search heuristic does not generate solution diversity: They always enumerate the same symmetrical solution whereas our heuristic finds 31 different solutions from the total number of solutions. Using symmetry breaking constraints, by contrast, predefined heuristics find the same number of solution as our dedicated search heuristic.

Tables 4.1 on the next page and 4.2 on the following page respectively provide the results for a 400×100 and 400×200 instance. The last entry is our own search heuristic. Although some predefined heuristics have a good performance on the first (small) instance, none of them scales. In fact, no predefined search heuristic finds a solution for a façade with size 400×200 in reasonable computational time whereas our dedicated heuristic already finds 726 different solutions in 180 seconds. Our heuristic clearly outperforms the others.

Table 4.1 – Heuristic comparison on a 400×100 (pixels) façade.

Strategy	First solution time (s)	Total time (s)	#nodes	#solutions
domOverWDeg	18.77	19.94	1412897	66
lexico_LB	**0.03**	0.22	2380	66
lexico_Split	**0.03**	**0.16**	**441**	66
maxReg_LB	**0.03**	0.22	2380	66
minDom_LB	0.74	19.96	1411183	66
minDom_MidValue	43.43	47.32	4755206	66
dedicated	**0.03**	0.85	10978	66

Table 4.2 – Heuristic comparison on a 400×200 (pixels) façade with a 3−minute time limit

Strategy	First solution time (s)	#nodes	#solutions
domOverWDeg	-	7286594	0
lexico_LB	-	5772501	0
lexico_Split	-	4966920	0
maxReg_LB	-	5490088	0
minDom_LB	-	11961712	0
minDom_MidValue	-	11157755	0
dedicated	**0.039**	**3499527**	**726**

Façades illustrations

The façades presented in Figures 2.14 on page 44, 2.15 on page 44 and 2.16 on page 44 in Section 2.4 on page 45, are used to illustrate the results generated by the algorithm. As we did before, the original façade is shown in the literal (a) of each figure and insulating envelopes in literals (b) and (c).

— Figure 4.16 presents the original façade number 1.

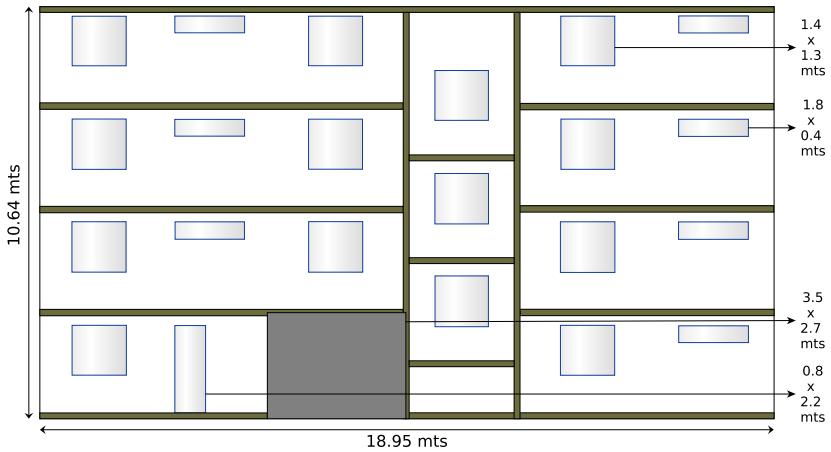


Figure 4.16 – Façade instance 1.

— Figure 4.17 on the facing page shows the first envelope solution thrown by the constraint solver using 3 meters as width upper bound and 10 meters as height upper bound. The envelope is composed of 14 panels and its length of junctions is 108.25 meters. The `OpackS` generates more solutions with the same number of panels and same length of junctions. In fact, it generates hundreds of solutions with small, even tiny, panel differences and thus

are meaningless solutions from the renovation perspective.

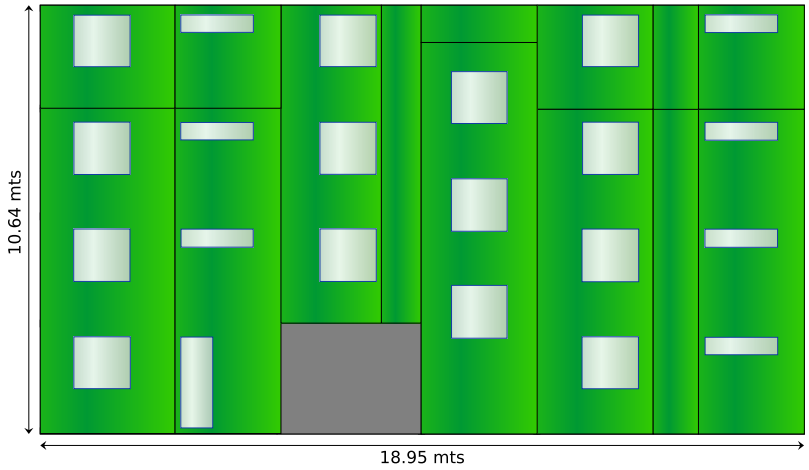


Figure 4.17 – Packing with vertical optional panels over façade 1.

— Figure 4.18 shows the first envelope solution thrown by the constraint solver using 10 meters as width upper bound and 3 meters as height upper bound. The envelope is composed of 12 panels and its length of junctions is 106.2 meters. Note that this solution is has the same design as those ones generated by the GaLaS and CaSyE algorithms. The reason for this is the decisions of taking the first available bottom-left point of the façade and the maximum allowed size for with and height. Note as well that the OpackS solution cannot do better than this envelope without changing the orientation.

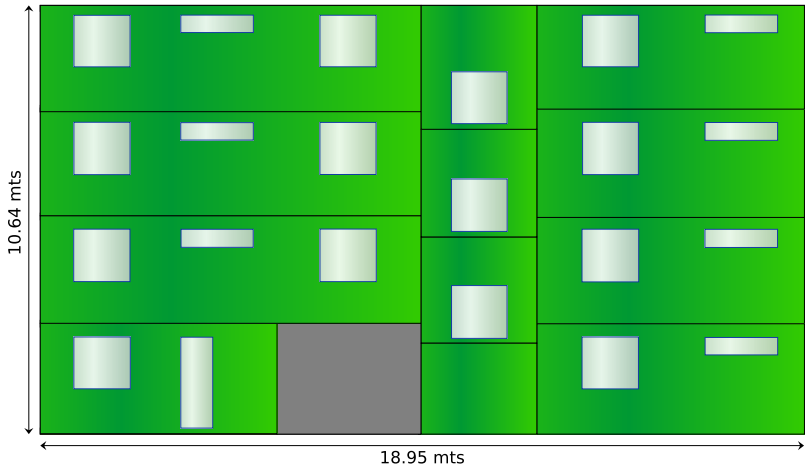


Figure 4.18 – Packing with horizontal optional panels over façade 1.

— Figure 4.19 on the following page presents the original façade number 2.

— Figure 4.20 on the next page shows the first envelope solution thrown by the constraint solver using 3 meters as width upper bounds and 10 meters as height upper bound. The envelope is composed of 15 panels and its length of junctions is 120.36 meters. In this insulating envelope, the limitation of having the panel orientation as a hard constraint

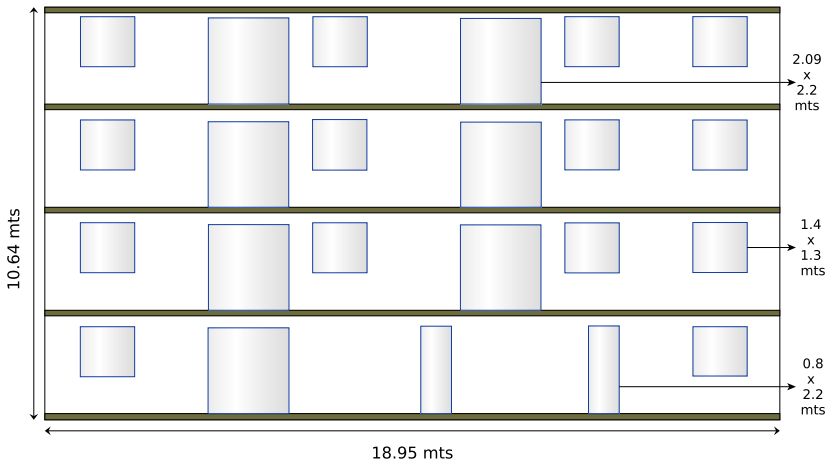


Figure 4.19 – Façade instance 2.

is more noticeable: In comparison to the GaLaS design, the OpackS cannot design a horizontal panel at top-right of the façade and thus it has to use several small ones to cover the remaining area.

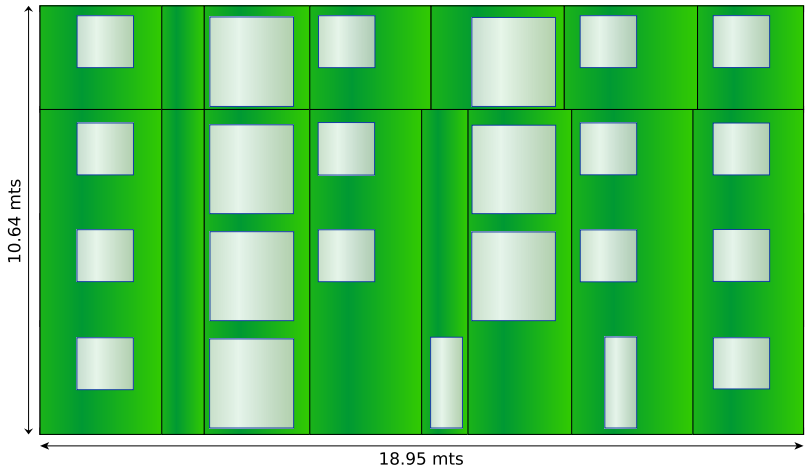


Figure 4.20 – Packing with vertical optional panels over façade 2.

— Figure 4.21 on the facing page shows the first envelope solution thrown by the constraint solver using 10 meters as width upper bound and 3 meters as height upper bound. The envelope is composed of 8 panels and its length of junctions is 97.08 meters.

— Figure 4.22 on the next page presents the original façade number 3.

— Figure 4.23 on the facing page shows the partial designed envelope reach by OpackS when using 3 meters as width upper bound and 10 meters as height upper bound. Note that, as the panel orientation is a hard constraint, the middle region of the envelope cannot be designed due to the size of the frames. The façade under the stated conditions has no solution with OpackS algorithm.

— Figure 4.24 on page 120 shows the first envelope solution thrown by the constraint solver using 10 meters as width upper bound and 3 meters as height upper bound. The envelope

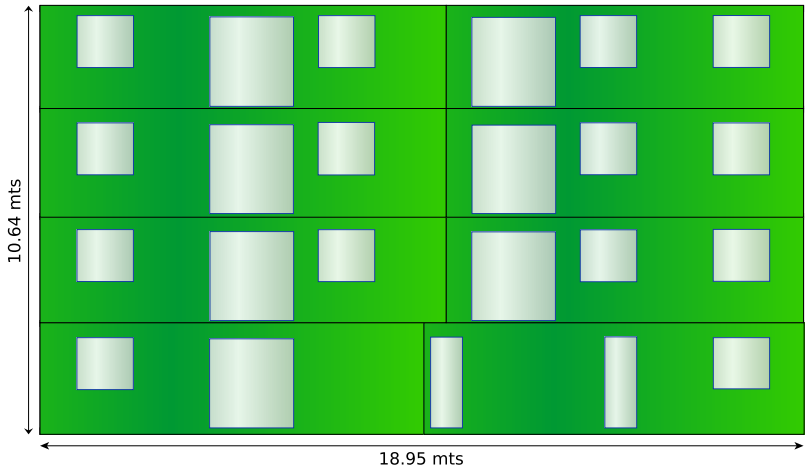


Figure 4.21 – Packing with horizontal optional panels over façade 2.

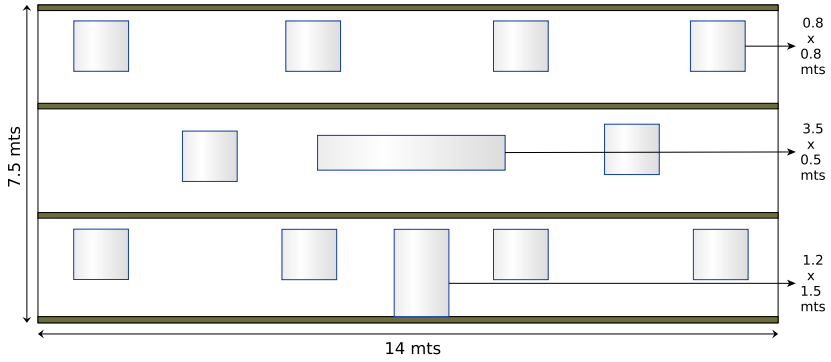


Figure 4.22 – Façade instance 3.

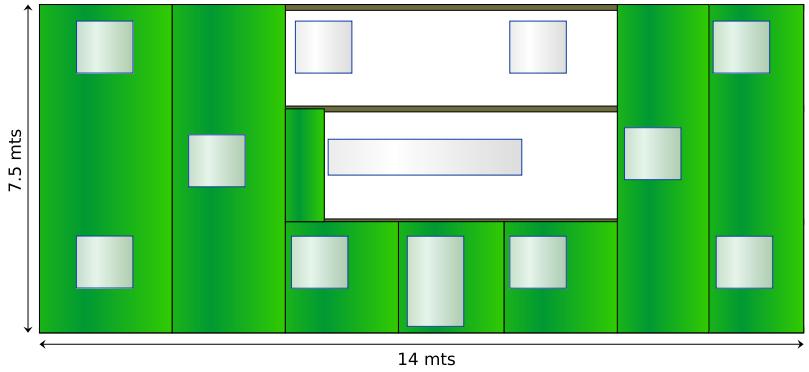


Figure 4.23 – Packing with vertical optional panels over façade 3.

is composed of 6 panels and its length of junctions is 57 meters. The same solution have been generated by the other approaches.

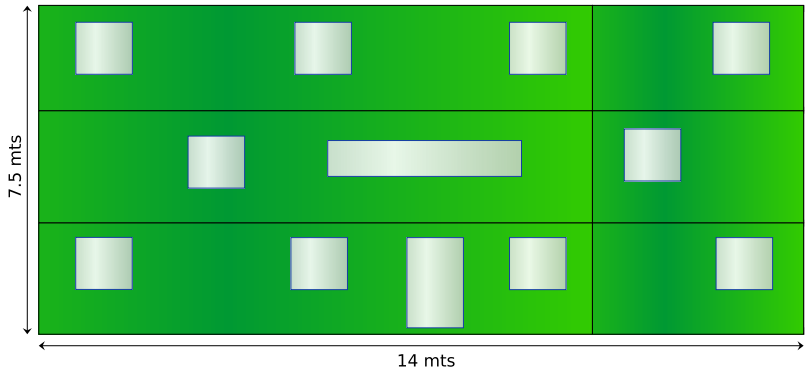


Figure 4.24 – Packing with horizontal optional panels over façade 3.

4.3.5 Discussion

This section presented the third automatic solution, called `OpackS`, to the façade-layout synthesis problem. This is also the second solution that uses as underlying mechanism the constraint solver `Choco` that enhances the development of the support system thanks to all the pre-defined constraints, search and provided abstractions. The filtering-based automatic solution allows to:

— Design correct panels and envelopes by applying filtering algorithms to the variable domains. The solution then minds the future states by removing values incompatible with the constraints in the model.

— Define a dedicated search heuristic that may exploit the structure of the problem in order to boots performance.

— To efficiently traverse the solution space by means of backtrack search provided by the solver.

— Find optimal solutions, w.r.t. the number of panels and under the conditions of only one panel orientation, by exploring good candidate nodes while avoiding bad ones (capabilities of the backtrack search).

Additionally, and likewise the `GaLaS` and `CaSyE` solutions, the architect may adapt the envelopes generated by `OpackS` in order to improve quality w.r.t. to number of panels or aesthetic tunning. The `OpackS` solution contrast with the `GaLaS` (Section 3.2 on page 56) and `CaSyE` (Section 3.3 on page 75) in three key points. Firstly, it assumes a minimum and maximum number of optional panels part of an insulating envelope and works with all of them in the solving process whereas the `GaLaS` and `CaSyE` pack panels on-the-fly. Secondly, the constraint solving is achieved by dedicated filtering algorithms to bootstrap the performance. Finally, it uses the optimality issues of the problem domain to look for valid optimal solutions. On this regard, the `OpackS` design only generates optimal solution for a given orientation, meaning that it is a hard constraint. This is due to the hard constraint of panel orientation that we have consider in the `OpackS` model. We have found that allowing the panel to have any orientation, i.e., the disjunction $p_w > p_h \wedge p_w \leq p_h$, increases the combinatorics and the solving time to unpractical limits (no solution after one computing hour). So, the greedy `GaLaS` algorithm is more robust as it accepts to swap orientation in run time. Further development of filtering algorithms and/or the search strategy may help to solve this gap. Lastly, the generation of envelopes does not takes into account the junctions alignment and thus no aesthetics envelopes are generated.

The `OpackS` solution, nonetheless, implements a dedicated search heuristic that uses the same decisions of that we have highlighted along the thesis; it assigns to the origin point of panels the first bottom-left available point of the façade and assigns the panels' size in their

upper bounds. Once more, the biggest panels on the first envelope are generated at the bottom of the façade under this setup (congruent with the on-site assembly requirements). The search mechanism then combines two different principles, a greedy constructive approach that is efficient but limited and a customized backtracking algorithm to explore alternatives. The customized search heuristic prevents backtracking for p_b, p_{x0} and p_{y0} in order to avoid holes in the resulting envelopes.

Considering the orientation as a hard constraint, the OpackS solution minimizes the number of panels by setting panels size in their maximum allowed value while using backtrack search to leave no valid node unexplored. Our approach provides good quality solutions and is a source for strong design aiding tool for the following reasons:

Reason 1: The implementation of the OpackS, enhanced by the use of constraint solver toolkit (decision variables, constraints and search heuristics), addresses particularity of unknown number of panel by assuming a big set of optional panels. This optionality notion is transparently implemented with boolean constraint variables.

Reason 2: The solving computing time, for realistic façades, is competitive for the industrial scenario (even when the other algorithmic solutions performs better), as it generates optimal envelopes with a given panel orientation.

Reason 3: The different tests that have been carried out show the scalability of our approach as well as evaluation over different real-life French façades.

Reason 4: The model is easier to extend than the previous approaches (GaLaS and CaSyE) as the addition of variables and constraints is transparent in the declarative nature of CP. Then, after the respective modelling of aesthetics aspects such as symmetry, for instance, the knowledge needed to be added to the constraint-based model effortless.

The utilization of the constraint solver in this automatic solution differs from the one in the sketching solution SkEdE presented in the previous Section 4.2 on page 97. Essentially, the sketching solution uses the Choco API to apply filtering and predefined search heuristics over an architects hand-made sketch. Here, in opposition, we had to develop our own propagators for specific constraint in our model and adapt traditional constraints, as DiffN, to give them an "open" view. The adaptation is based on the manipulation of optional panels represented by an extra (boolean) decision variable. Then, unlike the sketching solution, a certain level of expertise is required for the development of the OpackS one.

To conclude, we shall now compare the three automatic algorithms (GaLaS, CaSyE and OpackS) in a more detailed way. Figure 4.25 on the next page presents the time spent to reach a first solution for each algorithm against the façade area. One more time, and as presented in Section 3.2.5 on page 73, the execution time is limited to 30 seconds. The results show that for small façade instance, the algorithms behave in a competitive computational time. Nevertheless, the filtering-based algorithm OpackS increases the computational time from façade size of 45×23 meters. This is due to the exponential combinatorics within the problem and the solution space exploration of the OpackS algorithm.

Figure 4.26 on the next page presents the number of solutions found in a time window of 30 seconds, for each algorithm against the façade area. According to the figure, the GaLaS algorithm along with the binary tree implementation generates more solution diversity than those of CaSyE and OpackS. The number of generated envelopes exceeds in several times of magnitude the other approaches. The CaSyE algorithm as it is currently conceived only generates up-to two envelopes but with symmetrical aspect. On the other hand, the OpackS has the limitation, and advantage, of generating optimal non-symmetrical solutions (with orientation as a hard constraint), although in a high computational time, and thus it is not able to enumerate as much solutions as the GaLaS solutions.

Finally, let us compare the insulating envelopes thrown by GaLaS, CaSyE and OpackS for the three façade instance presented in Section 2.3 on page 43. Table 4.3 on page 123 presents the results. For convenience, we present in Figure 4.27 on page 124 the best solutions generated by each algorithm.

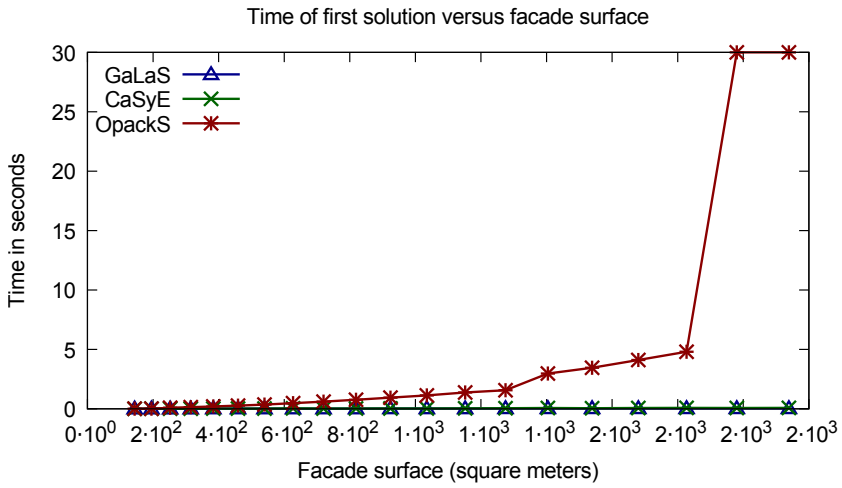


Figure 4.25 – Time to reach first solution versus façade area.

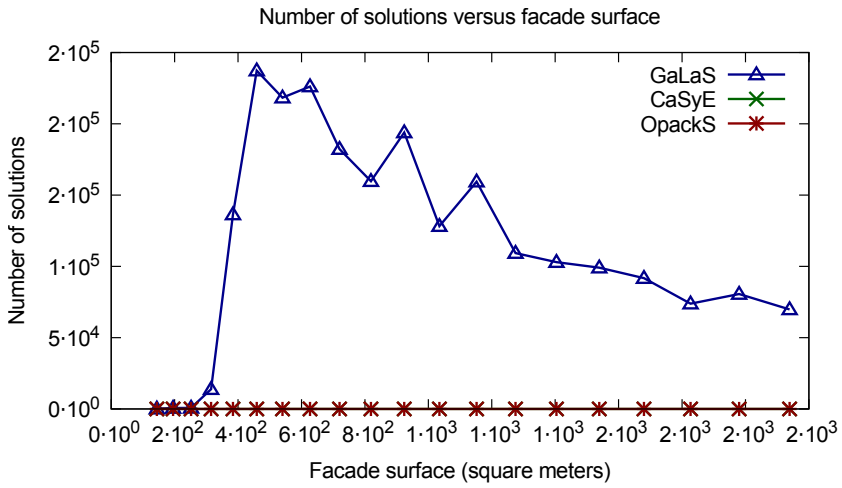


Figure 4.26 – Number of solutions in 30 seconds time window versus façade area.

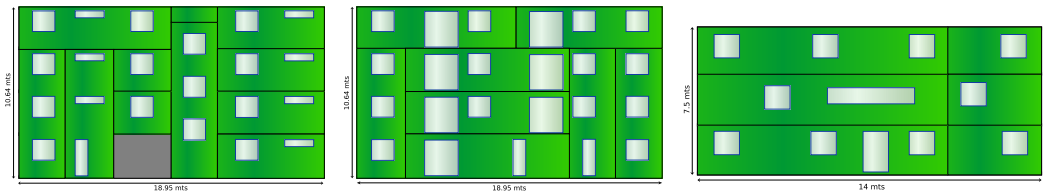
As the table shows, the greedy GaLaS algorithm is the more robust solution to the façade-layout synthesis problem; it is fast and finds the best (close to optimal) solutions as it is able of changing panel orientation. The solutions of GaLaS, nonetheless, are not aesthetically pleasant as junctions alignment is not part of its capabilities and thus do not generate symmetrical envelopes. Additionally, when asking only envelopes composed of only one orientation, we have found that the OpackS performs better or equal than the other solutions due to the way it traverses the solution space (filtering + backtrack search). As we commented in Section 3.2.5 on page 73, the vertical envelope for the second façade generated by the GaLaS algorithm, has one horizontal panel (cf. Figure 3.28 on page 70) and thus presents a better solution than OpackS which has the orientation as a hard constraint. One would assume that, overcoming the hard constraint setup, the OpackS would outperform the GaLaS solution (discussed as future work in Section 6.2 on page 141). Likewise the GaLaS algorithm, the OpackS does not generate symmetrical envelopes intentionally but rather "by change" if the façade geometry leads to it. Lastly, the CaSyE algorithm does generate envelopes with the biggest length of junctions (bigger number of panels) but, in return, it generates aesthetics envelopes if symmetry is considered

Table 4.3 – Results comparison for three automatic solutions.

		GaLaS			CaSyE		OpackS	
		Panels orientation			Panels orientation		Panels orientation	
		Vertical	Horizontal	V & H	Vertical	Horizontal	Vertical	Horizontal
Façade #1	N	15	13	**11**	14	-	14	12
	Loj	113.2	106.9	**100.5**	108.2	-	108.2	104.2
Façade #2	N	13	**8**	**8**	16	**8**	15	**8**
	Loj	115.2	97.0	**95.6**	123.0	97.0	120.3	97.0
Façade #3	N	8	**6**	**6**	7	**6**	-	**6**
	Loj	63.8	57.0	**57.0**	65.5	**57.0**	-	**57.0**

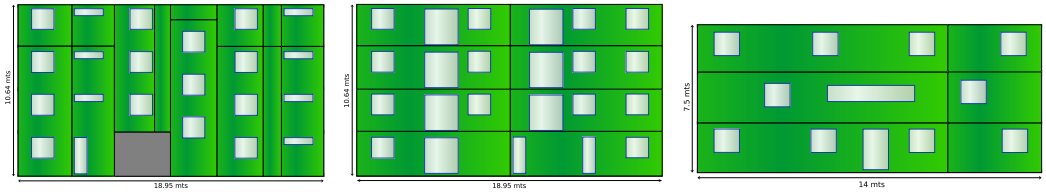
and it does so intentionally independent of the façade geometry. The algorithm, by contrast, is restricted in number of solutions and may fail as the OpackS solution does.

GaLaS



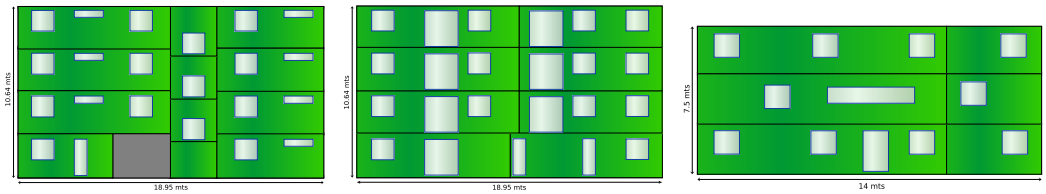
(a) 11 panels (b) 8 panels (c) 6 panels

CaSyE



(d) 14 panels (e) 8 panels 6 panels(f)

OpackS



(g) 12 panels (h) 8 panles (i) 6 panels

Figure 4.27 – Best solutions for the three façades for GaLaS, CaSyE and OpackS.

4.4 Digest

In this chapter, we have shown how to use constraint programming technology, mixing filtering and solving, to assist the design of insulating envelopes. We have presented two design approaches, SkEdE and OpackS, that rely on the constraint solver Choco to perform the design.

In the first step, we have presented a brief background of constraint programming technology. We have discussed the constraint programming notions of variables and domains, filtering algorithms including global constraints and logical constraints, and we have discussed the details in which backtrack search is based.

In a second step, we discussed a manual filtering-based design approach SkEdE that generates well-defined panels taking as input an architect hand-made sketch. Benefits from this approach are, on the one hand, that it braces up architects creative ideas by admitting as input ill-designed panels. Second, and unlike the manual solution of InDiE (Section 3.1 on page 48), it is not limited by the one-by-one panel setup as it may solve constraint conflicts and throw solutions for an undetermined number of panels. Finally, it is a fully declarative solution that exploits all capabilities of the constraint solver without specialized adaptation.

In a third step, we have presented an automatic filtering-based design approach OpackS that uses the power of global constraints to efficiently remove inconsistent values for panel variables. To do so, we have embraced the notion of optional decision variables to reflect the fact that panels are optional in an insulating envelopes. Consequently, constraints in the CSP model have been adapted, as well as filtering algorithms on the constraint solver, to reflect the optional panels setup. In addition, the approach uses a customized backtrack search, in which some decisions cannot be negated, that implements the same optimality choices of the GaLaS and CaSyE solution to arrive rapidly to solutions. As a major condition in this setup, the panel orientation is a hard constraint in the model. This is a limitation given that the geometrical composition of some façades, as illustrated in Figure 4.23 on page 119, cannot be designed using only panels in a given orientation. Nonetheless, in comparison to the other automatic approaches, it performs better or equal w.r.t. the number of panels. Thus, assumingly, a workaround to overcome the hard constraint should probably provide better results that the other solutions.

The good thing about science is that it's true whether or not you believe in it.
Neil deGrasse Tyson, 2011

5 Decision Support System

Contents

The chapter presents the system architecture, the configuration tasks within it and, the input and output specifications. Screen-shots of the support system usage are presented in Appendix A on page 171. Different versions of this chapter have been presented in Barco et al. (2015b) and in Barco et al. (2016b).

5.1 Motivation

Computer support for the thermal renovation process, and its conception by means of decision support system, is motivated by the following facts. First, the diversity of existing façades is very large as a result of history, building's technical aspects, or locations. Second the expectations of the architects or building owners about the façade style to renovate are also numerous. These two points lead to an extreme diversity of façade renovation problems. Traditional methods of course are able to handle them but for a high cost and they require expertise and are time consuming. The challenge is to be able to achieve the façade renovation problem with all its diversity at a much lower cost within industrialized approaches and methods.

The support system here presented implements a web-oriented architecture that divides different configuration tasks into different web-services. The system allows:

1. To efficiently introduce the renovation specifications (geometrical, structural, etc.),
2. to generate different valid configuration of panels and envelopes,
3. to rank the different solutions based on several criteria (number of panels, length of junctions, etc) and,
4. to present them to the person in charge of the renovation, such as an architect.

The solutions generated by the architects using the support system are numerical models of insulating envelopes to be send to factories for manufacturing. As such, the system, that uses constraint satisfaction as underlying model and implements the algorithmic solutions (`InDiE`, `GaLaS`, `CaSyE`, `SkEdE` and `OpackS`) proposed in this dissertation, is classified both as a computer-aiding design (CAD) system as well as a product configuration software (PCS).

5.2 Configuration Tasks

The support system here introduced is a web-oriented application where a web-service architecture allows to solve all configuration tasks within a three-step renovation process, presented in Figure 5.1.

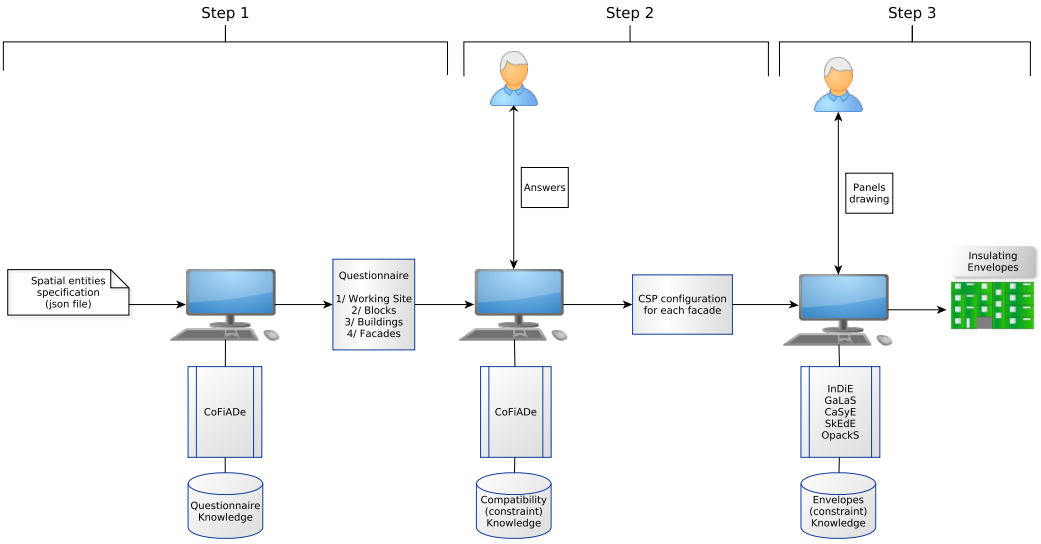


Figure 5.1 – Configuration process overview.

To understand the configuration tasks, let us first consider the information flow from the user's perspective, between the user and the system within a three step process.

Step 1: The user uploads a json file, as shown in Figure 5.2 on the facing page, containing the geometry and structural specification of spatial entities (working site, buildings and blocks and façade). This file comes from **Stage 1** and **Stage 2** of the renovation industrialization process, presented in Section 2.1.2. Information is stored in a data base.

⇒ The system generates a questionnaire for each of the spatial entities (from working site to façade) in the input file. The aim of the questionnaire is to determine for each

Type	id	ref	x	z	width	height	load
faạde	fac3		0.00	0.00	12.598	10.907	0
slab nose	sln1	fac3	0.00	0.00	12.598	0.2	2
shear wall	shw1	fac3	9.373	0.16	0.14	10.97	3
window	win1	fac3	9.614	9.177	1.398	1.444	0
door	doo1	fac3	9.614	0.326	2.419	2.036	0
panel	pan1	fac3	0.00	0.00	2.980	2.881	0

Figure 5.2 – Json Input File Example

spatial entity the limits for panels' size and weight. This questionnarie is built thanks to generic CSP instances, one for each spatial entity.

Step 2: The user has to answer the questions as much as possible (leaving blanks for the questions he cannot answer). Using the information about spatial entities (database) and their manufacturing/installation/environmental conditions (user's answers), the system deduces panels' lower and upper bounds for width, height and weight for each façade (parametrization of each CSP). The user has the possibility to overwrite these bounds within the deduced limits.

⇒ The system creates a constraint satisfaction model for each façade using the information in the input file and the deduced limits for panels' size (deduced by the questionnaire, Step 3). Here, each constraint satisfaction model instance is parameterized according to the façade information and the particular deduced panels' limits.

Step 3: The user invokes the second service for panels/envelopes configuration: he/she sends the parameterization of each façade, including panels' limits, and gets support (manual and automatic) for insulating envelopes design. She/he is able to draw panels, sketch an envelope, ask the system to find a solution in an automatic way and to tune any solution.

⇒ Several insulating envelopes are designed in a manual, automatic or semi-automatic way, depending on the user's expectations. For each of the insulating envelopes, the system provides the number of used panels and the length of junctions. The user is therefore able to select, for each façade, the best insulating envelope with respect to his/her needs: the one which contains the less panels or the one which is the more aesthetic.

5.2.1 Questionnaire Configuration

Once the questionnaire generated from the json input file, the user can start answering the information which has an impact on the panels'size and weight, Step 2 of the configuration process presented in Section 5.2 and illustrated in Figure 5.3. The questionnaire inquires the following information.

Working site. This is the bigger spatial entity of the renovation.

1. Values provided by the user are:
   — Is the working site in a windy region? {yes, no}
   — When does the on-site work take place? {summer, fall, winter, spring}
   — What is the cost target ? Euros.
   — What is the performance target ? $w.m^{-2}.k^{-1}$
   — Is there some obstacles ? {yes, no}

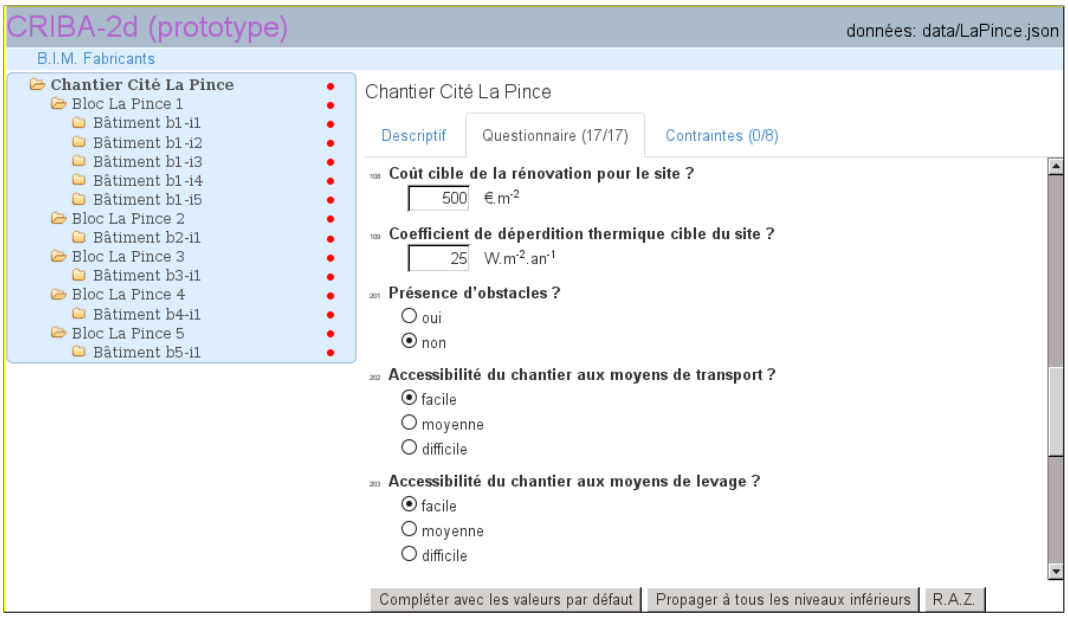


Figure 5.3 – Questionnaire screen-shot: Working site questions.

— How is the accessibility ? {easy, medium, hard}

2. Parameters deduced by the first filtering service which can be overwritten by the user:
   — Panels' width (wp_{wl}) and height (wp_{hl}) lower bound.
   — Panels' width (wp_{wu}) and height (wp_{hu}) upper bound.

Block. A block is a set of buildings which are usually attached by a common wall.

1. Values provided by the user are:
   — Is there some obstacles ? {yes, no}
   — How is the accessibility ? {easy, medium, hard}
2. Parameters deduced by the first filtering service which can be overwritten by the user:
   — Panels' width (bp_{wl}) and height (bp_{hl}) lower bound.
   $bp_{wl} \in [wp_{wl}, wp_{wu}]$ and $bp_{hl} \in [wp_{hl}, wp_{hu}]$
   — Panels' width (bp_{wu}) and height (bp_{hu}) upper bound.
   $bp_{wu} \in [wp_{wl}, wp_{wu}]$ and $bp_{hu} \in [wp_{hl}, wp_{hu}]$

Building. A building is the spatial entity where apartments are arranged and is the host of several façades.

1. Values provided by the user are:
   — Is there some obstacles ? {yes, no}
   — How is the accessibility ? {easy, medium, hard}
2. Parameters deduced by the first filtering service which can be overwritten by the user:
   — Panels' width (bgp_{wl}) and height (bgp_{hl}) lower bound.
   $bgp_{wl} \in [bp_{wl}, bp_{wu}]$ and $bgp_{hl} \in [bp_{hl}, bp_{hu}]$
   — Panels' width (bgp_{wu}) and height (bgp_{hu}) upper bound.
   $bgp_{wu} \in [bp_{wl}, bp_{wu}]$ and $bgp_{hu} \in [bp_{hl}, bp_{hu}]$

Façade. A façade is a composition of apartments along with frames.

1. Values provided by the user are:
   — Is there some obstacles ? {yes, no}

— How is the accessibility ? {easy, medium, hard}

2. Parameters deduced by the first filtering service which can be overwritten by the user:
   — Panels' width (fp_{wl}) and height (fp_{hl}) lower bound.
   $fp_{wl} \in [bp_{wl}, bp_{wu}]$ and $fp_{hl} \in [bp_{hl}, bp_{hu}]$
   — Panels' width (fp_{wu}) and height (fp_{hu}) upper bound.
   $fp_{wu} \in [bp_{wl}, bp_{wu}]$ and $fp_{hu} \in [bp_{hl}, bp_{hu}]$

This information collection has two specific goals: (1) to provide details about renovation aspects, such as the accessibility conditions, that are needed in the configuration process and (2) to provide upper bound for panels' size and weight for each level of spatial entities. Note then that panel's limits may be overwritten due to limitations of each spatial entity.

5.2.2 CSP-based Façade Configuration Problem

The support system relies on a configured constraint satisfaction model for each façade to renovate (generated at the end of Step 2 of the configuration process presented in Section 5.2 and illustrated in Figure 5.3). Each façade has (potentially) different size, number of windows, supporting areas, etc. It is important from the point of view of the configuration to assign valid range of values to panels, for instance, possible positions for panels must lie between zero and the façade width and height. Simply put, each façade has its own configuration parameters used in the constraint satisfaction model and in the envelope configuration process. Also, each façade may have different accessibility conditions, obstacles or even user preferences.

When configuring these CSP instances (Step 3 of the configuration process presented in Section 5.2 on page 128), it is important to conserve downwards consistency. Downwards consistency refers to the fact that information on higher level of the spatial entities hierarchy is propagated to the inferior levels, i.e., working site → blocks → buildings → façades, but it cannot be propagated upwards. As an example consider only accessibility conditions, obstacles presence and panels' size limits, for the specification in Figure 5.4.

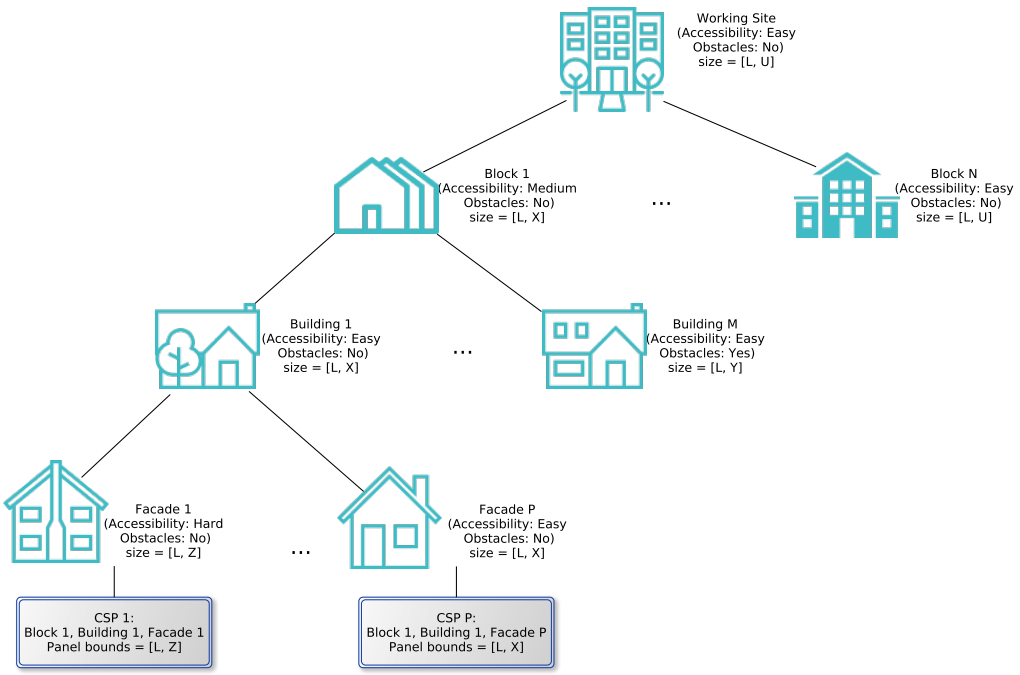


Figure 5.4 – Downwards consistency among entities.

Note that lower spatial entities on the hierarchy inherit values from upper levels. In Figure 5.4 on the previous page, for instance, façade 1 has an accessibility condition valuated to *hard* and thus the upper bound for panels' size is reduced to a given Z. This upper bound is not propagated upwards to the building 1; it conserves its inherited value X. Consequently, façade 2 inherits the value of X from the building 1 as no further reduction is needed for their panels' configuration. Naturally, it is the case that $Z < X < U$. Using this information a CSP is configured for each façade to be renovated. Note that monotonic properties of constraint satisfaction framework make transparent this configuration process.

5.3 Implementation

As commented before, different configuration tasks are assigned to different web-services. The information inputted into the questionnaire is processed by the first service called *Pre-processing Service*. Once the first service has pre-processed the information, the user invokes the second one to provide compliant envelope configurations. The second service is then called *Envelopes Design Service*. The architecture of the on-line support system is presented in Figure 5.5.

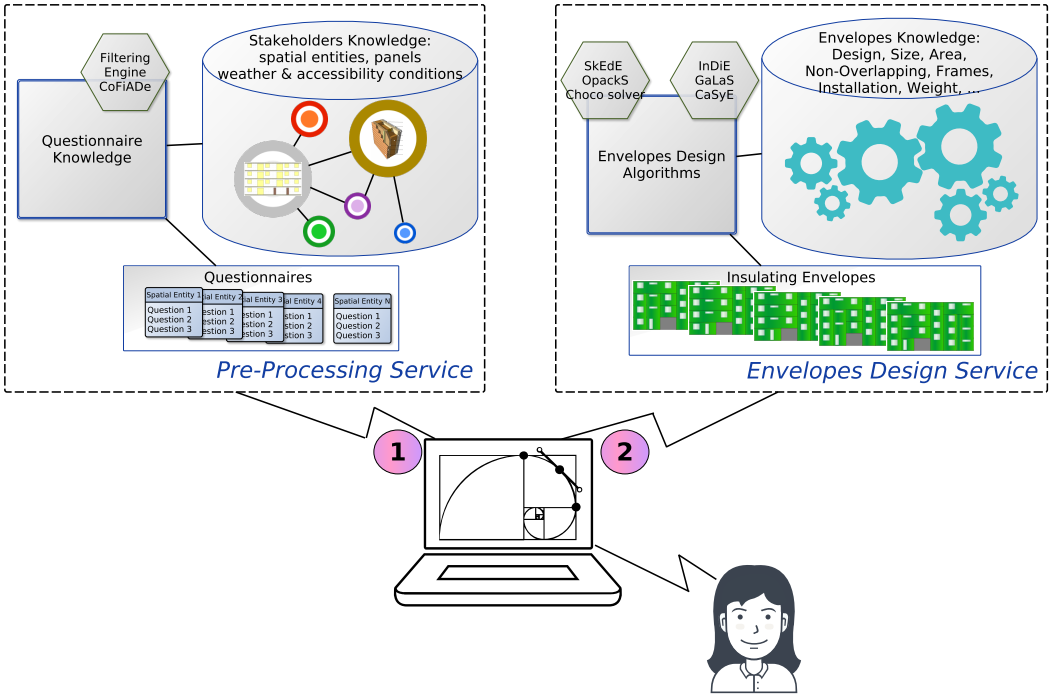


Figure 5.5 – Service-based architecture for on-line configurator.

Now, let us describe its INPUT and OUTPUT in a formal way. For each of the services the INPUT is a tuple of the form $\langle \mathcal{SPEC}, \mathcal{V}, \mathcal{D}(\mathcal{V}), \mathcal{C}(V) \rangle$ with $|\mathcal{V}| = |\mathcal{D}(\mathcal{V})|$ and

— $\mathcal{SPEC} = \langle \mathcal{WS}, \mathcal{BK}, \mathcal{BG}, \mathcal{FAC} \rangle$; $\mathcal{WS}$ variables describing the working site, $\mathcal{BK}$ variables describing blocks, $\mathcal{BG}$ variables describing buildings and $\mathcal{FAC}$ variables describing façades.

— $\mathcal{V} = \langle \mathcal{P}, \mathcal{FA} \rangle$; $\mathcal{P}$ variables describing a generic panel and $\mathcal{FA}$ variables describing a generic fastener.

— $\mathcal{D}(\mathcal{V}) = \langle \mathcal{D}(\mathcal{P}), \mathcal{D}(\mathcal{FA}), \rangle$; domain for each one of the variables in $\mathcal{V}$.

— $\mathcal{C}(\mathcal{V})$ a set of constraints over variables in $\mathcal{V}$.

Information in $\mathcal{SPEC}$ describes only properties of the spatial entities such as the number of façades, sizes, frames positions, etc. It corresponds to the parameters presented in Section 2.2.1 on page 37. Variables in $\mathcal{V}$ and $\mathcal{D}(\mathcal{V})$ are manufacturer dependent and include the generic panels' domains which depends on the manufacturing process. These variables in $\mathcal{V}$ correspond to the decision variables presented in Section 2.2.2 on page 37. Constraints in $\mathcal{C}(\mathcal{V})$ correspond to those from the problem domain by stakeholders, as presented in section 2.2.3 on page 38.

5.3.1 Pre-Processing Service

In this section, details of the pre-processing service in charge of the questionnaire configuration are introduced. The pre-processing service exploits the filtering engine CoFiADe (Vareilles et al., 2012). We only exploit the CoFiADe compatibility tables filtering in our implemetantion in order to deduce the relevant bounds for panels'size and weight. Compatibility tables are constraints expressed by enumerating all possible valid compatibility of values (these kinds of constraints can also be found, for instance, in the finite domain constraint module of `SWI-Prolog` (Wielemaker et al., 2010, Triska, 2012)).

Mapping

The pre-processing service is in charge of removing values from elements in $\mathcal{D}(\mathcal{V})$ that are not allowed by the established constraints. Here, constraints $\mathcal{C}(\mathcal{V})$ describe valid combination among different arguments in $\mathcal{SPEC}$ and variables in $\mathcal{V}$. We denote this set of constraints $\mathcal{C}_f(\mathcal{V})$ to distinguish them from the ones used in the envelopes design service. These constraints are formalized as compatibility tables. Formally, the filtering is a mapping $\mathcal{M}$ from variables and domains to domains

$$\mathcal{M}(\mathcal{SPEC}, \mathcal{V}, \mathcal{D}(\mathcal{V}), \mathcal{C}_f(\mathcal{V})) \rightarrow \mathcal{D}'(\mathcal{V}) \tag{5.1}$$

The result $\mathcal{D}'(\mathcal{V})$ contains the new domain of the decision variables describing the generic panels and fasteners, where $\mathcal{D}'(\mathcal{V}) \subseteq \mathcal{D}(\mathcal{V})$.

As stated previously in Steps 1 and 2 of the configuration process presented in Section 5.2 on page 128, the goal of pre-processing is to set domains for configurable components using spatial entities information and constraints to do so. In our on-line support system we use the filtering engine CoFiADe to perform this filtering. Several reasons support our choice. First, the system is already on-line, making it usable in no time. Second, it is well conceived for supporting decision-making processes. And third, it uses efficient compatibility tables for domain pruning; applying a given compatibility table is made in constant time $\mathcal{O}(1)$.

Stakeholders Knowledge

Configurable components of the renovation are panels. Panels are configurable by fixing their width, height and position over the façade.

The following material describes the compatibility tables, presented in Table 5.1 on the next page, which state the allowed combinations between the user's input values and configurable components values. Now, to deepen the details of the spatial entities conditions, we have further divided the conditions presented in Section 2.1.2 on page 26 into wind speed presence (more than 80 km/h), season (i.e., summer, spring, fall, winter), obstacles presence (e.g., trees, water source, etc) and accessibility conditions (e.g. roads).

C1: Relation between the wind speed conditions of spatial entities and panels' size, where $envp_{wu}$ and $envp_{hu}$ are upper-bounds for panels' width and height, respectively, when constrained by environmental conditions

C2: Relation between the season in which the on-site work will take place and panels' size, where $seap_{wu}$ and $seap_{hu}$ are upper-bounds for panels' width and height, respectively, when constrained by the season.

C3: Relation between obstacles in spatial entities and panels' size, where $obsp_{wu}$ and $obsp_{hu}$ are upper-bounds for panels' width and height, respectively, when constrained by the presence of obstacles.

C4: Relation between accessibility of spatial entities and panels' size, where $accp1_{wu}$ and $accp1_{hu}$ are upper-bounds for panels' width and height, respectively, when constrained by *medium* level accessibility conditions, and $accp2_{wu}$ and $accp2_{hu}$ are upper-bounds for panels' width and height, respectively, when constrained by *hard* level accessibility conditions, with $accp2_{wu} \leq accp1_{wu}$ and $accp2_{hu} \leq accp1_{hu}$.

It is worth highlighting that the different upper bounds for panels coming from wind, season, obstacles and accessibility conditions, along with the user's preferred bounds, are coherently used to set panels' limits. For instance, we consider the final panels' width upper bound for a given façade as **min(**$envp_{wu}, seap_{wu}, obsp_{wu}, accp2_{wu}, userp_{wu}$**)**, where $userp_{wu}$ is the upper bound imposed by the user.

Table 5.1 – Compatibility tables on pre-processing service.

C_1

Wind	Panels' size
yes	$(w_p \leq envp_{wu}) \wedge (h_p \leq envp : hu)$
no	$\emptyset$

C_2

Season	Panels' size
summer ∨ spring	$\emptyset$
fall ∨ winter	$(w_p \leq seap_{wu}) \wedge (h_p \leq seap_{hu})$

C_3

Obstacles	Panels' size
yes	$(w_p \leq obsp_{wu}) \wedge (h_p \leq obsp_{hu})$
no	$\emptyset$

C_4

Accessibility	Panels' dimensions
easy	$\emptyset$
medium	$(w_p \leq accp1_{wu}) \wedge (h_p \leq accp1_{hu})$
hard	$(w_p \leq accp2_{wu}) \wedge (h_p \leq accp2_{hu})$

5.3.2 Envelopes Design Service

As stated in Step 3 of the configuration process presented in Section 5.2 on page 128, the envelopes design service uses the proposed algorithms `InDiE`, `GaLaS`, `CaSyE`, `SkEdE` and `OpackS` to help the user to design compliant envelopes. As presented in Chapter 3 on page 47, when the envelopes design service uses the `InDiE`, `GaLaS` or `CaSyE` algorithms, it exploits either functional programming or heuristic approaches. As explained in Chapter 4 on page 93, when the envelopes design service uses `SkEdE` and `OpackS` solutions, it exploits CP techniques, and more precisely, the constraint solver `Choco`. `Choco` implementation is based on filtering algorithms as studied in Section 4.1 on page 94 (these kinds of constraints can be found, for instance, in the constraint programming environments `Gecode` (Schulte et al., 2010) and `ECLiPSe` (Schimpf and Shen, 2012)).

Envelopes Knowledge

The envelopes design service is in charge of insulating envelope designs. The system uses the algorithms presented in this dissertation to help the user's drawing to generate insulating envelope designs. Now, while information of $\mathcal{SPEC}$ and $\mathcal{V}$ are the same as the pre-processing services, it is not the case for domains and constraints. To differentiate them let's call the input domains $\mathcal{D}_s(\mathcal{V})$ and the constraints $\mathcal{C}_s(\mathcal{V})$. Intuitively, variable domains $\mathcal{D}_s(\mathcal{V})$ are provided by the mapping of the pre-processing service, i.e.,

$$\mathcal{M}(\mathcal{SPEC}, \mathcal{V}, \mathcal{D}(\mathcal{V}), \mathcal{C}_f(\mathcal{V})) = \mathcal{D}'(\mathcal{V}) = \mathcal{D}_s(\mathcal{V}) \tag{5.2}$$

where $\mathcal{D}(\mathcal{V})$ is the initial domain for panels' size and weight. Constraints in $\mathcal{C}_s(\mathcal{V})$ are stated as first order formulas and express, not compatibility among elements but, requirements for valid envelopes (see Section 2.2 on page 37). The output of the solving service is a set of insulating envelopes, each one characterized by the number of panels and length of junctions. Formally, the envelopes design service is a function of the form

$$\mathcal{F}(\mathcal{SPEC},\mathcal{V},\mathcal{D}_s(\mathcal{V}),\mathcal{C}_s(\mathcal{V}),\mathcal{H}) = \langle \mathcal{PX},\mathcal{PY},\mathcal{PW},\mathcal{PH},\mathcal{FX},\mathcal{FY}\rangle \tag{5.3}$$

where $\mathcal{PX}$ and $\mathcal{PY}$ represent the origin of coordinates for each panel in the solution, $\mathcal{PW}$ and $\mathcal{PH}$ the width and height, respectively, for each panel in the solution, and $\mathcal{FX}$ and $\mathcal{FY}$ represent the position of each frame on the envelope. Additionally, the function is parameterized by an heuristic $\mathcal{H}$ stating which algorithm is meant to be used.

5.4 Personalization and Recommendation

In order to improve the configuration capabilities of the system, it is useful to allow the personalization of the insulating envelopes. In this section we present the personalization (preferences) set by the user, the features that are crucial for the configuration and the features that may be excluded or completed by the system (our analysis is based on Falkner et al. (2011)).

5.4.1 User's Preferences

As indicated in Section 2.1.2 on page 26, in **Stage 3** of the renovation industrialization process, the user has the possibility to overwrite panels' limits within those deduced by the system, as presented in Figure 5.6. For instance, the user can enforce panels to be square ($p_w = p_h$) in order to give the façade a specific look. This first feature for personalization allows the user to generate more solution diversity (defined as different solutions (Schreiber, 2010)).

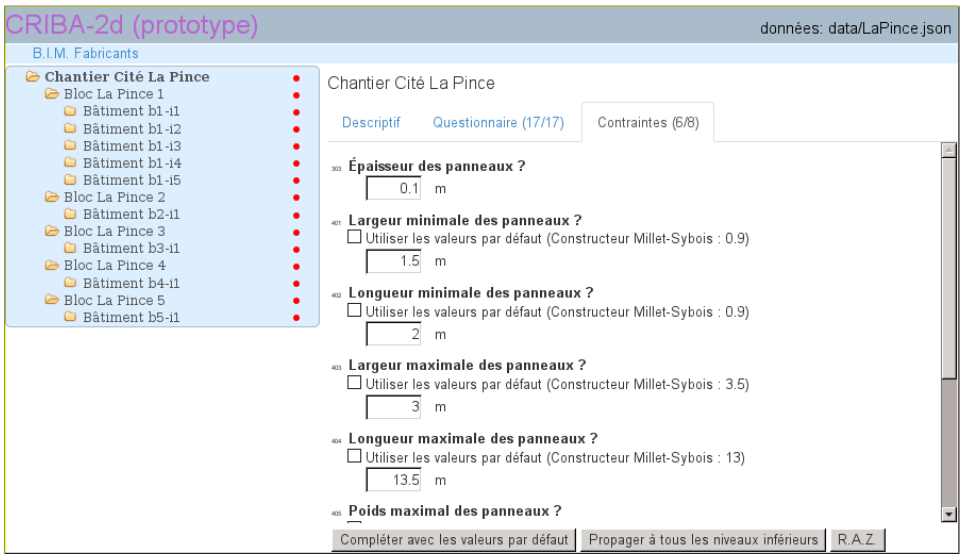


Figure 5.6 – Questionnaire screen-shot: Overwriting of panels' limits.

The second feature for personalization of the configuration depends on panels' orientation. Panels' orientation is only a relation between panels' width p_w and height p_h, being horizontally oriented when $p_w > p_h$ and vertically oriented otherwise. This is an important personalization feature as it affects the aesthetics properties of the envelopes, which is a hard constrain in the `OpackS` solution. Nonetheless, the preferred panels' orientation may be not respected due

to conflicts in the constraint model (such as in GaLaS; it is therefore considered as a soft-constraint. When the preferred orientation cannot be satisfied, the system recommends to the user the solutions composed of the maximum number of panels in the preferred orientation; an application of the *nearer-is-better* similarity function (Falkner et al., 2011). Choosing a minimum number of panels to be vertical (respectively horizontal) is a personalization feature considered as future work (Section 6.2 on page 141).

5.4.2 Features Ranking and Exclusion

To simplify the process from the user's viewpoint, we have selected a set of key features to the configuration as well as the optional ones which can be left to the system. The idea behind this ranking is to avoid overwhelming the final user with the process (Tiihonen and Felfernig, 2010). Three of the features are key for the configuration, meaning that they are the most highly ranked (Falkner et al., 2011):

— Number of panels: It is important to present to the user first the envelopes composed of the minimum number of panels. Among the set of solutions with minimum number of panels, it presents first those with minimum length of junctions.

— Orientation: It is important to achieve a pleasant aesthetic envelope. If this feature is not set, the system recommends envelopes configured using only one panel's orientation (hard constraint, OpackS). Then, it tries to configure envelopes using only vertically or only horizontally oriented panels.

— Panels' size bounds: The limits for the panels' size are mandatory and have to take into account all the renovation industrialization process limitations. Forgetting one of them (manufacturing, shipping or installing limitations) leads to envelopes configuration which cannot be either manufactured, shipped or installed onto the façades.

At the other end of the spectrum are the optional features which can be left in blank or be filled by the support system. In the renovation process, these features concern mainly information that has an impact on the configuration:

— Working site in windy region and season of on-site work: Default values are *no wind* in the region and *summer* for on-site work which have no impact on panels' limits.

— Obstacles presence and accessibility conditions: These properties are inherited from the working site information and propagated downwards. If no information is available, default values are *no obstacles* and *easy accessibility* which have no impact on panels' limits.

— Fasteners location: The location of fasteners (bottom, lateral edges, top) is another preference (soft constraint). This is however bypassed by the system if the size of panels demands it. For instance, the system recommends *bottom* fasteners for horizontal panels and *lateral* fasteners for vertical ones.

5.5 Discussion

A brief discussion about the division of task into different services, and its communication behavior, closes the chapter.

5.5.1 Underlying Handling

The division of configuration tasks and its handling by different services is supported by the underlying declarative model of constraint satisfaction. Indeed, the monotonic properties of constraint-based systems make it possible to collect the requirements from the problem domain, state them as constraint and map them into constraint-based implementations in a simple intuitive way.

Now, note that the user's answers to the questionnaire are sent to the pre-processing service which replies with the panels' size bounds for a particular façade. This means that the clients' browser is the one capturing the panels' bounds. Then, upon user's request, the system sends the façades specifications and particular panels' bounds to the envelopes design service which replies with different envelopes solutions (when automatic mode is needed by the user). Therefore, there is no need for any intermediary procedures to map information from one service to the other; only values (answer to questions) are sent to the pre-processing service and only values (panels' bounds) are returned from it whereas only values (façade specification and panels' bounds) are sent to the envelopes design service and only values (envelopes numerical model for manufacturing) are returned from it.

5.5.2 Advantages

Benefits for configuration tasks division into web-services are rather simple. On the one hand we use a modular design for the architecture. In our on-line system the modular design allows to add or remove variables, domains and questions in the pre-processing service, i.e., by means of adding or removing compatibility tables. In addition, as we use CoFiADe, we may mix different variable representations as integer domains, continuous domains and symbolic domains whereas in most constraint systems mixing variable domains is not allowed or is not efficient enough. For instance, given the reduced number of constraints for continuous domains in `Choco`, the representation has to be changed to integer domains.

On the other hand, as a benefit of tasks division, we improve performance by avoiding the use of binary equalities and binary inequalities constraints whose computational time is $\mathcal{O}(n * m)$, where n and m are the number of values in the domain of the two variables involved in the constraint. Thus, at the moment of finding solutions, the underlying `Cutting & Packing` algorithms, focus on the envelopes generation rather than deducing panels limits or other time consuming responsibilities. In particular, the underlying constraint solver `Choco`, used to explore the solution space for optimal solutions, propagates and applies search using only those constraints defining insulating envelopes and not compatibilities between configurable components.

Regarding the performance, the two configuration tasks must be studied separately. As commented before, applying a given compatibility table in the filtering service is made in constant time. Thus, the time involved in the pre-processing service depends on the questionnaire and on the number of spatial entities (buildings, façades and so on). On the envelopes design service, by contrast, the performance depends on the underlying `Cutting & Packing` algorithm. Execution over façades with size 40×10 meters, 50×12 meters and 60×15 meters lasts between one and two seconds. The use of a dedicated heuristic that exploits the problem structure allows to reach such good performance.

And yet the working part of this population of 2,500 persons was producing as much real wealth for society, as, less than half a century before, it would have required the working part of a population of 600,000 to create. I asked myself what became of the difference between the wealth consumed by 2,500 persons and that which would have been consumed by 600,000?

The Revolution in the Mind and Practice of the Human Race
Robert Owen, 1849

6 Concluding Remarks

We conclude this dissertation with a synthesis of the thesis context, contributions of our work and possible directions for future research.

6.1 Synthesis

The problem of façade-layout synthesis consists in designing façades insulating envelopes using rectangular configurable panels. The problem raises in a multi-partner project called CRIBA which aims at the industrialization of buildings thermal renovation. Our work within the project focused on assisting architects in the manual and automatic design of envelopes that, according to stakeholders, should have the minimum number of panels, and within those minimum length of junctions, and the largest panels should be located at the bottom of the façade.

This design problem is the main scientific problem addressed in the dissertation. The problem, that we have called *façade-layout synthesis problem*, is subject to the following particularities.

1. Firstly, the number of rectangular panels to design an envelope and their size are not known in advance though panels sizes are bounded.
2. Secondly, rectangular windows and doors over the façade must be covered by panels with the condition that no partial overlapping is allowed.
3. Thirdly, panels are attached by their corners in specific rectangular areas over the façade.
4. Fourthly, no overlapping between panels, and no holes in the insulating envelopes, are allowed.
5. Lastly, envelopes should be composed of the minimum number of panels and, among those, the preferred one have minimum length of junctions.

On regard to the particularities of the problem and the rectangular geometry of elements (façades, windows/doors, supporting areas and panels), the problem was treated as an orthogonal two-dimensional `Cutting & Packing` problem. In this dissertation we have studied the modelling capabilities of constraint satisfaction problems as well as solving techniques, from artificial intelligence and operational research, to solve this two-dimensional `Cutting & Packing` problem. Under a scientific viewpoint, we have focused our attention on the characteristic of unknown number of panels as it is a particularity overlooked in the literature. Further, to the best of our knowledge, there is no other work tacking simultaneously the five particularities of the problem and thus we have proposed a new `Cutting & Packing` classification that expresses

the traditional problems while including the the façade-layout synthesis one (see Figure 2.1 on page 25). The results of our research have been condensed in a (mock-up) decision support system dedicated to assist architects in the insulating envelopes design. As such, the dissertation was built upon the fields of constraint satisfaction, `Cutting & Packing`, product configuration and layout-synthesis. The remaining of this section is devoted to outline the major contributions of the dissertation.

The first contribution of our research is the *description* of this previously unaddressed problem and its *modelling* by constraint satisfaction problems (CSPs). We have contributed to the fields of `Cutting & Packing` and layout synthesis by describing the particularities and novelties of the façade-layout synthesis problem. To do so, we have presented, in Section 2.1 on page 23, the renovation process, spatial entities, façade and configurable panel details along with the renovation requirements that guides our modelling decisions. Conversely, we have contributed to the fields of constraint satisfaction and `Cutting & Packing` by developing a constraint-based model of the problem. We have then used the knowledge extracted by stakeholders to represent the envelopes design problem as a CSP. To do so, we have presented, Section 2.2 on page 37, the parameters customizing the CSP, decision variables and constraint defining well-designed panels as well as well-designed insulating envelopes along with optimality issues. More precisely, we have presented and discussed nine constraints (size, size compatibility, area, panels non-overlapping, frames overlapping, installation, weight, interference and symmetry breaking constraints) and one design objective (minimization the number of panels in a given envelope).

On a second step, we have contributed with the fields of `Cutting & Packing`, layout synthesis and product configuration by developing five algorithmic solutions (two manual and three automatic) to the design problem and implementing them in a decision support system. We will now discuss each of these solutions in regard to their benefits and limitations.

The first manual solution, called `InDiE` (Section 3.1 on page 48), implements an *interactive* behavior as it provides *real-time* feedback to the user. This interactive approach presents a major benefit; it allows to design panels one-by-one while informing, visually, about constraint conflicts. Then, the user is capable to iteratively construct the envelope by re-dimensioning and re-allocating panels when constraint conflicts exists. As a condition, and given the hand-eye coordination demanded in this setup, the visual responses of the `InDiE` design is made in less than 100 milliseconds.

The second manual solution, called `SkEdE` (Section 4.2 on page 97), supports architects creative *sketching* by exploiting the filtering and search capabilities of a constraint solver. This solution does not implements an interactive behavior, unlike the `InDiE` one, and it does not inform the architects about constraint conflicts during the sketching. Thus, the iterative construction process of envelopes is lost in this setup. However, this solution presents other advantages. First, the design based on sketch is not limited to one panel at the time design; the architect may draw any number of panels over the façade, even ill-designed ones. Second, although it does not inform about constraint conflicts, it does solve conflicts when they are present by executing filtering algorithms from an underlying constraint solver. This behavior is independent of the number of panels as global constraints may be applied to any number of variables. Finally, it performs search over the panel representation (decision) variables in order to present a consistent envelope design. Here, the `SkEdE` presents panels design as similar as possible to those drawn in the sketch by setting their size in their maximum allowed value.

The first automatic solution, called `GaLaS` (Section 3.2 on page 56), makes local (optimal) decisions in the aim to arrive fast to a close to optimal solution. This solution implements the well-known *greedy* algorithmic notion. It iteratively designs envelopes by placing one panel at the time and solving constraint conflicts locally. The key local decisions involve the positioning of the panel in the first bottom-left point of the façade and the sizing on their maximum allowed. These local decisions are "blind" in the sense that no future states of the design, neither the façade geometry, is considered which is a limitation it has. Due to this local decision-making, the resulting envelopes have not their junctions aligned or, if they are, it is only accidentally

(aesthetics aspects). The GaLaS heuristic allows the design under the soft constraint of preferred panel orientation. Then, when assigning size and solving conflicts it will respect, as much as possible, the chosen orientation. Two handy decisions allow to overcome no-solutions setup when the design of a panel fails due to constraint conflicts: First, it changes the orientation of the panels and, if it still does not work, it performs backtracking to change the position and/or size of the previously designed panels. These two decision are significant benefits of the solution. Lastly, we have allowed the solution to overpass the orientation preference and thus combine panel orientation indistinctly. The combination provides two major benefits: First, it allows to reach solution with the lowest number panels as it mixes vertical and horizontal orientation to design the envelopes and, second, it allows to generate different solutions in a short computational time (achieved with binary trees).

The second automatic solution, called CaSyE (Section 3.3 on page 75), executes cuts over the façade in order to design panels in areas free of constraint conflicts. This solution implements the guillotine cuts technique that executes orthogonal cuts from one extreme of the façade to the other. This solution, unlike the GaLaS one, is not "blind" as it takes into account the geometrical composition of the façade to execute the guillotine cuts. Executing orthogonal guillotine cuts engenders the main advantage of the solution; it generates solutions in which the panels junctions are aligned, i.e., more aesthetic envelopes. In addition, the heuristic adopts the same local decisions as the GaLaS algorithm as it sets panels origin point in the first available bottom-left point of the façade and sets panels size according to the maximum allowed and the position of the guillotine cuts. The construction mechanism of the algorithm forces portions of the envelope to be designed only with one orientation (hard constraint). But, the resulting envelopes have a mixture of vertical and horizontal panels (soft constraint) as the final envelopes are the union of different portions. The solution has three major drawbacks. On the one hand, it generates up-to two envelopes for a given façade, i.e., low solution diversity. Second, the generated envelope has more panels than those generated by the GaLaS one (except for symmetrical façade where both approaches throw analogous results). Lastly, the solution will throw no result for façade in which no guillotine cuts can be made and thus is not as robust as the GaLaS solution.

The third automatic solution, called OpackS (Section 4.3 on page 106), is supported by filtering and search capabilities of a constraint solver in the aim to generate optimal solutions w.r.t. the number of panels. As a condition and limitation, the orientation of panels is a hard constraint in this solution, meaning that envelopes are composed of panels with only one orientation and this orientation cannot be changed on run-time. Considering this limitation, i.e., design with panels in only one orientation, we have shown that the OpackS solution generates optimal insulating envelopes. Moreover, the OpackS behaves better or equal than the GaLaS and CaSyE solutions under the hard constraint of panel orientation. This performance have been reached by adapting the CSP model to reflect the fact that panels are optional in a given envelope and adapting some filtering algorithms to exploit this fact. Then, the solution is not "blind" as the GaLaS solution because, although it does not consider the geometry of the façade as does the CaSyE one, it does take into account future states by applying filtering over the panel variables. Further, an additional and borrowed benefit of the solution is the backtrack search "skills" provided by the underlying constraint solver (complete search although limited by time).

As a concluding note, the dissertation has presented an (mock-up) on-line product configuration software (specialized version of a decision support system) in which the model and algorithmic solutions have been implemented. The novelty of this support system is the division of configuration (consequently design) tasks within the renovation process (Chapter 5 on page 127).

6.2 Future Work

The study presented in this dissertation raises new questions and challenges. The following are some of the future directions contemplated by the author. Now, before entering in future directions for each algorithmic solution, we discuss three issues directions for the improvement

of all of the solutions, mainly the automatic ones.

To begin, it is appealing to put efforts on the improvement of the solutions, or development of new ones, in such a way that aesthetics aspects, other than junctions alignment, are taken into account. To our understanding, the symmetry consideration on the envelopes look, alignment of junctions, is a good criterion for generating "pleasant" solutions (as shown by the `CaSyE` solution). Nevertheless, no other criteria aesthetics have been studied. An attractive starting point of research, is the inclusion of patterns in the design knowledge. Here, however, the goal of minimizing the number of panels may compromised, as proven by the `CaSyE` algorithm that generates envelopes with the highest number of panels. Aesthetics of envelopes is a property highly appreciated by architects and is a fascinating challenge for us.

Next, we consider future research for the `Cutting & Packing` of shapes that are not rectangles. This is one of the most interesting directions as it would remove limitations of the algorithms and would strengthen the aesthetics of envelopes as new design patterns may be conceived. Then, the algorithms would be able to cope with different convex shapes of panels, façades and frames (see illustration in Figure 6.1).

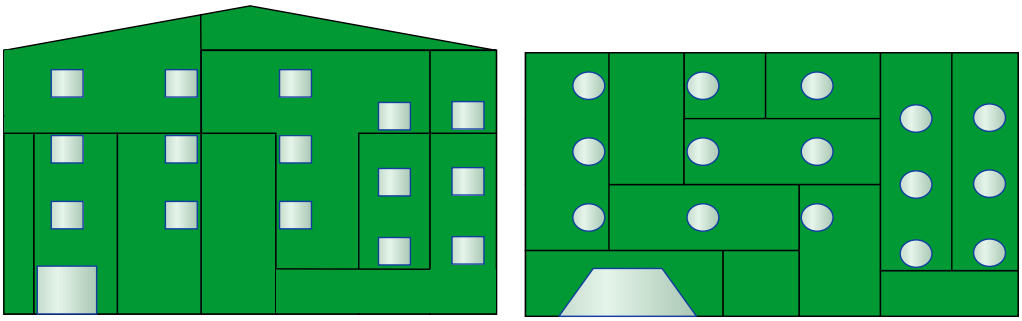


Figure 6.1 – Triangular gable, convex panel shapes and circular frames shapes.

Lastly, we consider post-processing capabilities to be, potentially, a powerful aiding mechanism for improving the quality of the results. These capabilities may come in two forms. First, a manual post-processing in which, for instance, the architect re-dimensions one panel and the system automatically re-dimension the remaining panels to be consistent with the new panel design and constraints. Here is the architect who guides the post-processing. The second form is to have a fully automated post-processing. Once an insulating envelope has been thrown, a procedure may re-dimension the panels in order to align the junctions between them, or make them more similar in size, for instance, and thus achieve a more pleasant envelope. The automated re-dimension of panels is tricky due to the diverse set of constraints that must be respected.

In what follows, we discuss future work for each algorithmic solution.

1. Interactive manual design `InDiE` (Section 3.1 on page 48). The interactive design may be promoted to a semi-recommender design. Up-to-now, when the user manually draws panels over the GUI, it receives a feedback about the ill- or well-design of the panel. When the panel is ill-designed, the system only informs about conflicts existence but does not suggest, or recommend, any alternative direction. An interactive design under a recommender setup would visually show two or three alternatives for a well-designed panel. The guided design would be further strengthened.
2. Greedy automatic design `GaLaS` (Section 3.2 on page 56). The following strategic directions have been identified. First, inclide pre-processing capabilities to the algorithm. Intuitively, a human design takes advantages of the façade dimensions and positions of frames and supporting areas to start the design. Thus, it is a good choice to perform a geometrical analysis of the façade and use the results to guide the design. In particular, an

analysis may look for symmetries in the façade, distances between windows and between supporting areas to define better local decisions when assigning the panels' sizes (currently in its maximum allowed size). Figure 6.2 presents an illustration of such case. Note in the figure that the panels in literal (b) are almost of the same size whereas the size difference of the panels in literal (a) is noticeable. Having panels with similar sizes may not only improve the envelope aesthetics but also it is also a good choice for the renovation logistic as it would required fewer types of trucks and installation machinery.

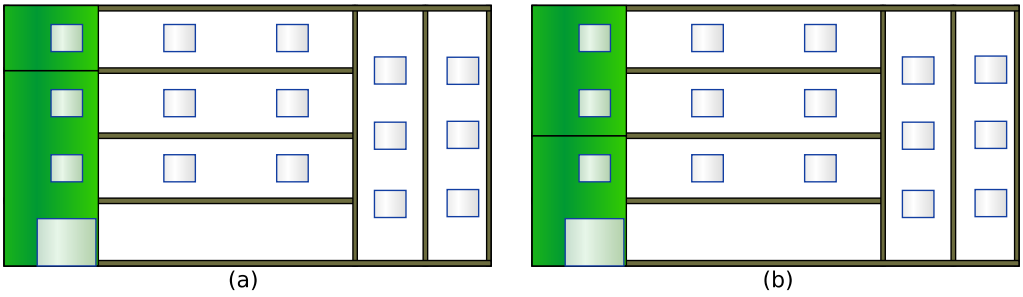


Figure 6.2 – Preprocessing stuff.

The second direction of the solution is to enhance the search with better exploration capabilities. For instance, using a *Branch and Bound* strategy we can avoid exploring nodes that, given their partial design and its respective evaluation, do not satisfy the user criteria (e.g., number of panels, length of junctions, etc.). Thus, at the end of the process would only remain those envelopes that the architect has requested.

3. Cutting automatic design `CaSyE` (Section 3.3 on page 75). Some attractive improvements to this solution may be achieved with the following research. A strategic direction for the `CaSyE` algorithm is to enhance it to generate more than two solutions. A tentative alternative is to exploit non-guillotine cuts techniques as well as the combination of horizontal and vertical cuts. Indeed, allowing the algorithm to make partial orthogonal cuts over the façade, in both dimension, will generate different results on the envelopes and, potentially, better results w.r.t. number of panels and length of junctions. More important, executing non-guillotine cuts in both dimensions may allow the algorithm to find solutions where it currently fails. For instance, the façade in Figure 6.3.a, that have been already studied in Section 3.3.4 on page 84, may be solved with this approach as shown Figure 6.3.b.

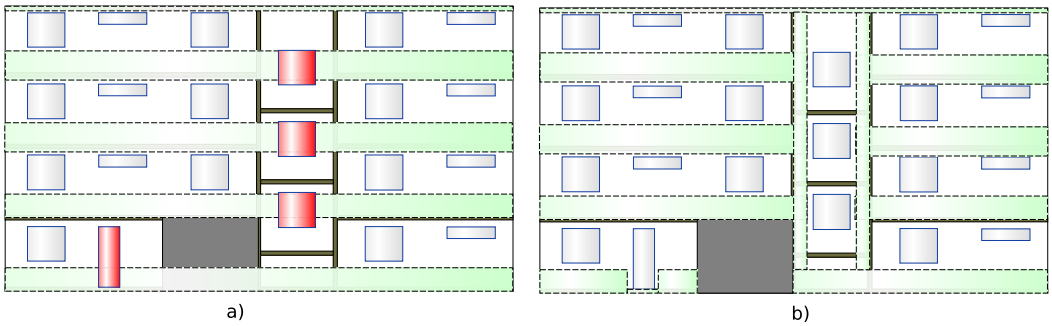


Figure 6.3 – Non-guillotine cuts in both dimensions.

4. Filtering-based manual design `SkEdE` (Section 4.2 on page 97). Even when we have already proposed an interactive design of envelopes (`InDiE` design), research on how to promote the `SkEdE` design into an interactive one is interesting. In fact, the current implementation

of the sketching design may be adapted to an interactive one: Every drawn panel is sent to the constraint solver to execute filtering without executing search. Here, nonetheless, the response time may not satisfy the real-time requirements ($<$ 100 milliseconds).

5. Filtering-based automatic design `OpackS` (Section 4.3 on page 106). This solution can be further explored with the following strategic directions. First, the relaxation of the hard constraint for panels' orientation. This is, to be sure, one of the most interesting directions as it would improve the quality of the results, w.r.t. the minimum number of panels, that will potentially be better than the other automatic solutions. This is, however, not a simple task due to the combinatoric within the problem. Also, the current maximum number of optional panels is too high and thus the two-step approach that we have used. Further analysis must stress the calculation of this maximum with regard to the façade composition rather than its size. Finally, the proposed dedicated heuristic follows the same greedy approach when assigning the origin point (currently bottom-left) and size of panels (currently upper bound). Choosing different values may lead to different solutions ergo different appearance. Randomness, although may produce holes, is an interesting direction to generate different geometrical patterns.

Dans ce singulier pays, où les hommes ne sont certainement pas à la hauteur des institutions, tout se fait carrément, les villes, les maisons et les sottises.

Le Tour du monde en quatre-vingts jours
Jules Verne, 1873

7 Résumé Long en Français

Les travaux de recherche présentés dans cette thèse se situent dans une problématique d'aide à la conception d'enveloppes isolantes pour la rénovation thermique de bâtiments résidentiels collectifs. Ces enveloppes isolantes sont composées de panneaux multifonctionels rectangulaires, configurables et préfabriqués en usine. Leur conception repose sur les cinq caractéristiques suivantes. Premièrement, le nombre de panneaux nécessaires pour concevoir une enveloppe ainsi que leur taille respective ne sont pas connus au début de la rénovation (mais leur taille est cependant bornée). Deuxièmement, en raison des contraintes de fabrication, chaque fenêtre et chaque porte présentes sur la façade à rénover doivent être insérées dans un et un seul panneau. Troisièmement, les panneaux sont fixés à des endroits spécifiques de la façade, assez résistants pour supporter leur poids, nommés zones d'accroche. Quatrièmement, ni trous (zone non couverte), ni chevauchements entre panneaux ne sont autorisés. Cinquièmement, afin de garantir une isolation thermique performante tout en minimisant son coût, les enveloppes doivent être composées d'un nombre minimal de panneaux. Au vue de la complexité de ce problème, nous restreignons nos travaux de recherche aux façades rectangulaires portant des menuiseries et des zones d'accroche rectangulaires.

Compte tenu des cinq caractéristiques énoncées et de l'hypothèse de forme rectangulaire des éléments traités (panneaux, façades, menuiseries, zones d'accroche), la conception des enveloppes est à la fois un problème de découpe et de conditionnement à deux dimensions et un problème de configuration. Ce problème est formalisé et traité comme un problème de satisfaction de contraintes et a pour but d'aider la conception dédites enveloppes isolantes.

Ce mémoire est divisé en six chapitres. Dans ce résumé étendu, nous présentons les idées principales de nos travaux et nous nous concentrons sur ses contributions majeures.

7.1 Introduction

Le problème de découpe et de conditionnement à deux dimensions consiste à positionner des composants à l'intérieur d'un ensemble de contenant (éventuellement singleton) de telle sorte que les composants soient complètement contenus dans les contenants sans se chevaucher. Les caractéristiques du problème peuvent varier en fonction du secteur d'application visé, tels que la transformation du bois, la découpe de métal, de verre ou de cuir, la conception de pages Web, de microcircuits ou d'appartement.

Étant donné que les problèmes de découpe et de conditionnement à deux dimensions sont NP-difficiles, plusieurs travaux de recherche ont mis au point des solutions algorithmiques et

informatiques pour soutenir la prise de décision dans les différents secteurs de l'industrie. En particulier, les techniques d'intelligence artificielle et de recherche opérationnelle ont montré leur robustesse face à ces problèmes. Par exemple, le problème de découpe de pièces minimisant les chutes, un cas particulier des problèmes de de découpe et de conditionnement souvent présents dans les industries de l'acier et de l'aluminium, a été formalisé comme un modèle de programmation linéaire, un modèle de programmation dynamique et un modèle évolutif. Ces types de solutions, cependant, sont soit trop généraux ou trop spécifiques, pour résoudre tous les problèmes de découpe et de conditionnement qui proviennent de l'industrie. Par-dessus tout, il est difficile de créer des heuristiques génériques qui exploitent efficacement la connaissance et l'expertise de chaque problème spécifique.

Un cas particulier de problème de découpe et de conditionnement se pose dans notre contexte de rénovation thermique de bâtiments par l'extérieur. Ce cas particulier, aussi appelé problème de calepinage de façades, concerne l'allocation de panneaux rectangulaires configurables sur des façades rectangulaires. L'ensemble des panneaux couvrant une façade constitue son enveloppe isolante qui réduit le transfert thermique entre l'intérieur et l'extérieur du bâtiment. La motivation derrière cette isolation est de parvenir à une réduction de la consommation énergétique des bâtiments qui dépasse actuellement le secteur des transports et de l'industrie. Cinq caractéristiques rendent notre problème original et intéressant :

1. Premièrement, le nombre de panneaux constituant une enveloppe, ainsi que leur taille, ne sont pas connus *a priori*. La taille des panneaux est cependant bornée par un intervalle donné dépendant des contraintes de fabrication et de transport sur site.
2. Deuxièmement, les menuiseries des façades (fenêtres et portes existantes) doivent être complètement recouvertes par les panneaux, chacune devant être couverte par un et un seul panneau.
3. Troisièmement, les panneaux sont fixés dans des zones spécifiques qui sont assez résistantes pour supporter leur poids supplémentaire (mur de refend, nez-de-dalle, etc), appelées zones d'accroche.
4. Quatrièmement, afin de garantir une isolation extérieure performante, aucun trou, ni aucun chevauchement entre panneaux ne sont admis dans les enveloppes.
5. Cinquièmement, afin de garantir une isolation extérieure performante tout en minimisant le coût global de la rénovation, les enveloppes doivent être composées d'un nombre minimal de panneaux.

L'un des problèmes clés adressés dans ces travaux est de permettre aux architectes de concevoir rapidement plusieurs solutions de calepinage de façade respectant l'ensemble des contraintes et besoins exprimés. Nos travaux de recherche trouvent donc leur origine dans ce besoin industriel de conception d'enveloppes isolantes pour la rénovation thermique de bâtiments résidentiels collectifs.

Dans le cadre de nos travaux de thèse, nous posons la restriction suivante sur la géométrie des éléments considérés : les façades, les menuiseries (portes et fenêtres), les zones d'accroche ainsi que les panneaux sont rectangulaires. Cette limitation nous permet de considérer le problème de conception de solutions de calepinage comme un problème de découpe et de conditionnement à deux dimensions. Il est à noter que chacune des façades d'un bâtiment a ses propres dimensions, porte un nombre de menuiseries potentiellement différent et positionnées distinctement des autres façades du bâtiment, et possède des zones d'accroche avec une résistance spécifique. Clairement, la conception d'une enveloppe isolante dépend d'une part de la géométrie de la façade (position des menuiseries et des zones d'accroche) et de sa résistance, et d'autre part, des préférences de l'architecte (rendu esthétique du bâtiment).

7.2 Contexte Scientifique

Tout problème de découpe et de conditionnement possède une structure commune qui permet de l'identifier comme tel. Cette structure concerne deux aspects. En premier lieu, deux éléments

participent au problème de découpe et de conditionnement :

— Un ensemble, peut-être singleton, de grands objets ou contenants : Ces objets sont les entités spatiales dans lesquelles le conditionnement a lieu.

— Un ensemble de petites entités ou articles : Ce sont les entités spatiales à conditionner dans les grands objets.

En second lieu, dans le cadre de la structure du problème de découpe et de conditionnement, chaque problème doit satisfaire :

— Le non-chevauchement des petites entités,

— Le conditionnement de toutes les entités ou d'un sous-ensemble dans le grand objet.

Ces éléments sont spécifiquement définis dans une ou plusieurs dimensions. En outre, les éléments peuvent avoir des géométries régulières ou irrégulières : La plupart des travaux scientifiques portent sur des éléments à géométrie régulière (comme des rectangles ou cercles), alors que des éléments à géométrie irrégulière apparaissent souvent dans les contextes industriels. Quelle que soit la géométrie du contenant et des entités, le conditionnement doit satisfaire un ensemble des exigences héritées du domaine d'application. Ces exigences sont généralement connues sous le nom contraintes supplémentaires.

Une sous-catégorie de problèmes de découpe et de conditionnement correspond aux problèmes à deux dimensions. Parmi ces derniers, un cas particulier de problèmes est appelé le problème de découpe et de conditionnement rectangulaire. Dans ce cas, les petites entités et les contenants sont de géométrie rectangulaire. Il existe deux possibilités pour conditionner le petites entités dans les contenants : celles-ci sont autorisées à pivoter ou non. Dans ce dernier cas, nous parlons de problèmes de découpe et de conditionnement orthogonaux : les bords des entités sont parallèles à ceux des contenants. Nous retrouvons cette propriété d'orthogonalité dans ces travaux de thèse.

En ce qui concerne la complexité, l'ensemble des problème de de découpe et de conditionnement sont classés comme NP-difficile. Cela signifie qu'il n'y existe aucun algorithme permettant de les résoudre en temps polynomial (à condition que $P \neq NP$). En d'autres termes, l'augmentation du nombre de petites entités rend le problème insoluble en temps fini.

En raison de la complexité des problèmes de découpe et de conditionnement à deux dimensions, classé comme NP-difficile, les scientifiques ont mis au point des solutions algorithmiques appuyant la prise de décision dans différents secteurs de l'industrie. En particulier, les techniques d'intelligence artificielle et de recherche opérationnelle ont montré leur robustesse face à ces problèmes.

Par exemple, le problème de découpe de pièces minimisant les chutes souvent présent dans les secteurs de l'acier et de l'aluminium et se formalise comme un modèle de programmation linéaire, modèle de programmation dynamique ou modèle évolutif. Ces types de solutions, cependant, sont soit trop généraux ou trop spécifiques, pour résoudre tous les problèmes de découpe et de conditionnement qui proviennent de l'industrie. Par dessus tout, il est difficile de créer des heuristiques génériques qui exploitent efficacement la connaissance et l'expertise de chaque problème spécifique.

7.3 Contexte Industriel

Un cas particulier de problèmes de découpe et de conditionnement à deux dimensions se pose dans le contexte de rénovation thermique de bâtiments par l'extérieur du projet CRIBA. Ce cas particulier, aussi appelé problème de calepinage de façades, concerne à la fois le dimensionnement des panneaux et leur positionnement sur la façade. L'ensemble des panneaux dimensionnés et positionnés couvrant une façade constitue son enveloppe isolante. Cette enveloppe réduit le transfert thermique entre l'intérieur et l'extérieur du bâtiment. La motivation derrière cette isolation est de parvenir à une réduction de la consommation énergétique des bâtiments qui dépasse actuellement les secteurs des transport et de l'industrie.

La conception des solutions de calepinage repose d'une part, sur une description complète, précise et exacte de chacune des façades en termes de géométrie et de résistance et, d'autre part, sur une description générique des panneaux configurables. Cette connaissance, enrichie par les attentes de l'architecte (orientation des panneaux, tailles imposées), conduit à des solutions de calepinage tenant compte des exigences de toutes les parties prenantes (architectes, fabricants de panneaux, etc).

La description des façades est essentiellement acquise auprès d'un géomètre pour leur géométrie (taille et position des menuiseries) et d'un bureau d'étude structure pour leurs propriétés structurelles (position et résistance des zones d'accroche). Cette description donne lieu à la réalisation d'une maquette numérique du bâtiment sur laquelle les éléments clés sont conservés : les menuiseries telles que les fenêtres, les portes et entrées de garage, ainsi que les zones d'accroche, telles que les murs de refend ou les nez-de-dalle. Tout le reste, comme les gouttières, balcons ou lampadaires directement attachés au bâtiment, sont enlevés et ne sont donc pas considérés dans le problème. En outre, le pignon du bâtiment (partie triangulaire d'un mur entre les bords de l'intersection des pentes de toit) est considéré comme étant hors périmètre de nos travaux car il a besoin de panneaux à géométrie spécifique (triangulaire) qui seront dimensionnés et positionnés manuellement par l'architecte. Enfin, à la demande des architectes et pour plusieurs raisons principalement géométriques, des zones de la façade peuvent être retirées du processus de calepinage, appelées « Zone hors Configuration » ou ZoC. La géométrie de ces zones doit être rectangulaire, afin de ne pas interférer avec le processus de découpe et de conditionnement et traiter ces zones comme des panneaux déjà définis.

Les panneaux sont des rectangles non-déformables avec leurs côtés parallèles à ceux de la façade. Le problème adressé dans cette thèse est donc une instance d'un problème de découpe et de conditionnement orthogonal à deux dimensions. La description des panneaux isolants est le deuxième point clé dans la rénovation des bâtiments. Leur description doit être précise en termes de tailles admissibles, des poids admissibles et de leurs propriétés thermiques. En effet, chaque panneau est configurable et possède une plage de tailles différentes (largeur et hauteur), un nombre potentiel de menuiseries incluses, une plage de poids différents (fonction de leur taille, de leurs caractéristiques d'isolation, des menuiseries qu'ils contiennent et du revêtement choisi) et une performance thermique (fonction du type d'isolation, de son épaisseur et les menuiseries incluses). Nous soulignons le fait que chaque panneau est individuellement configuré (taille, poids, isolation, etc) puis fabriqué avec la structure approprié contenant les nouvelles menuiseries. Il est à préciser qu'aucun raccord n'est possible sur le chantier, rendant la précision exigée au millimètre.

Les caractéristiques des panneaux sont essentiellement recueillies auprès des fabricants de panneaux. Doivent être spécifiés la plage de largeurs possibles p_w (bornes inférieures et supérieures), les hauteurs possibles p_h (bornes inférieures et supérieures), les contraintes reliant la largeur p_w et la hauteur p_h des panneaux (par exemple, la combinaison des deux bornes supérieures n'est pas autorisée), l'épaisseur d'isolation possible I_th (bornes inférieures et supérieures), les types d'isolation I_ty, les contraintes reliant l'épaisseur I_th et le type d'isolation I_ty, et l'ensemble des types de revêtement externe C_lt qui sont autorisés (gamme de couleurs, matériaux et textures, comme le bois, le vinyle, la pierre ou l'aluminium).

Les panneaux sont fabriqués juste avant leur expédition, sans stock. Le type de transport choisi (convois exceptionnels ou plus petits camions) a un impact sur leur taille : l'utilisation de convois exceptionnels pour livrer les panneaux sur site n'a aucun impact sur leurs tailles admissibles (toutes les combinaisons autorisées), alors que l'utilisation de plus petits camions impose aux panneaux des dimensions qui leur sont adaptées.

Les panneaux ont une orientation (horizontale ou verticale) fonction du rapport entre leur largeur p_w et leur hauteur p_h. Si le rapport $\frac{p_w}{p_h}$ est inférieur à 1, le panneau est vertical, sinon, horizontal. Cette information a un impact fort sur la structure interne du panneau et son sens de pose sur la façade. Ceci est d'autant plus pertinent si le panneau contient des menuiseries. Quand un panneau contient des menuiseries, compte tenu de sa structure interne, une distance minimale d doit être respectée entre les bords du panneau et ceux des menuiseries. Les panneaux

sont également caractérisés par une performance thermique T_p et un poids W_p, fonction de sa taille (p_w, p_h), de l'épaisseur de l'isolant $(I_t h)$, du type d'isolant $(I_t y)$, des menuiseries qu'il contient et du revêtement $C_l t$, uniquement utilisé dans le calcul de poids W_p.

Les panneaux peuvent être fixés sur une façade de différentes manières : « suspendus », « posés » ou « agrafés » le long de leur périmètre. La façon de fixer les panneaux sur la façade a un impact sur la manière de répartir leur poids sur les zones d'accroche et par répercussion sur le calepinage de l'enveloppe isolante. Les panneaux sont fixés par des attaches métalliques, appelées fixations. Chaque fixation est composée de deux parties : l'une fixée directement sur la façade (patte de support) et l'autre installée directement sur le panneau à l'usine (support du panneau). La position exacte des fixations dépend du calepinage de la façade et de la position des panneaux. Son poids est directement inclus dans celui des panneaux. Le poids de chaque panneau est divisé en deux et réparti sur deux zones d'accroche : soit sur les coins du bas à droite et à gauche, si le panneau est « posé », soit sur les deux coins du haut à droite et à gauche, si le panneau est « suspendu », soit au milieu de ses cotés verticaux si le panneau est « agrafé ». Chaque fixation peut supporter le poids de deux panneaux, car le mode d'accroche « suspendu », « posé » et « agrafé » est choisi pour l'ensemble de la façade à rénover. Lors du positionnement d'un panneau, nous vérifions donc que les zones d'accroche impactées puissent effectivement encaisser chacune, la moitié de son poids.

Dans le problème abordé ici, nous ne considérons que deux caractéristiques principales des panneaux : leur taille et leur poids. En effet, le type d'isolant et son épaisseur sont déterminés pour toute la rénovation par des simulations de performance énergétique au début du processus de rénovation. Il devient donc inutile de calculer la performance de chaque panneau et de chaque calepinage, mais il devient important d'estimer les fuites thermiques pour différencier les différentes solutions de calepinage. Ces fuites thermiques apparaissent principalement à la jonction entre les panneaux. Notre objectif de minimisation du nombre de panneaux est donc renforcé. Concernant le poids des panneaux, le revêtement extérieur est choisi en amont pour toute la rénovation, de manière similaire au type et à l'épaisseur d'isolant.

7.4 Méthode & Objectifs

Parmi l'ensemble des techniques d'intelligence artificielle et de recherche opérationnelle, les problèmes de satisfaction de contraintes ou *CSP* se sont avérés remarquablement robustes pour aborder les problèmes de découpe et de conditionnement orthogonaux à deux dimensions. De plus, il a été prouvé que les *CSP* permettent de modéliser aisément à la fois les problèmes de calepinage et de configuration. En particulier, leur caractère déclaratif permet une représentation claire des connaissances comportant des variables de décision et de leurs relations, aussi nommées contraintes. Ce modèle de connaissances est indépendant de la solution mise en œuvre pour trouver une ou plusieurs solutions au problème ainsi que du langage de programmation sous-jacent. En effet, le modèle de connaissances, qui est le plus critique pour la résolution d'un problème, peut être résolu en utilisant différentes techniques telles que la programmation linéaire en nombres entiers mixtes et la programmation par contraintes ou optimisé par l'utilisation de méta-heuristiques telles que algorithmes génétiques.

Dans nos travaux, nous nous concentrons sur la modélisation et la résolution de problèmes de découpe et de conditionnement orthogonaux à deux dimensions pour aider à la décision en rénovation de bâtiments. La question scientifique principale abordée dans ce mémoire se résume à « Comment générer des solutions optimales, en terme de nombre minimal de panneaux, pour le problème de calepinage de façades, en tenant compte des attentes de l'ensemble des parties prenantes et des limites industrielles ? » Les travaux de cette thèse tendent à répondre aux deux questions suivantes :

1. Comment modéliser le problème de découpe et de conditionnement orthogonal à deux dimensions comme un problème de satisfaction de contraintes lorsque le nombre d'entités configurables et leur taille sont inconnus ?

2. Comment couper / paquer / couvrir une surface rectangulaire avec un nombre non fixé d'entités rectangulaires configurables ?

7.5 Contributions

Parmi l'ensemble des techniques issues de l'intelligence artificielle et de la recherche opérationnelle, nos travaux de thèse s'appuient sur les approches de satisfaction de contraintes pour modéliser et résoudre le problème de calepinage de façades. Ce problème est mis en oeuvre avec différentes solutions algorithmiques en utilisant un langage de programmation orientée objet.

L'ensemble des solutions algorithmiques présentées ici aborde le problème de calepinage de façades lorsque le nombre d'entités rectangulaires (panneaux) et leur taille sont inconnus. Ces travaux de thèse présentent deux contributions majeures :

Contribution 1 : En raison des caractéristiques originales du problème de calepinage de façades, sa description et sa formalisation comme un problème de satisfaction de contraintes est la première contribution de ces travaux de thèse. En particulier, la description du processus de rénovation industriel, la conception des enveloppes perçue comme un problème de découpe et de conditionnement et le système d'aide à la décision pour les architectes sont discutés. L'ensemble des exigences, des limites et des connaissances « métier » recueillies auprès des parties prenantes et des architectes a été intégré dans un seul modèle de satisfaction de contraintes et sert de base au système d'aide à la décision. Dans ce modèle, les variables de décision sont liées à la position et la taille des panneaux pour chaque enveloppe isolante. Supposons que N représente le nombre de panneaux constituant une enveloppe isolante donnée. Chaque panneau p^i où $i \in [1, N]$ est décrit par l'origine de son coin en bas à gauche et sa taille :

- $p^i_{x0} \in [0, fac_w]$ est l'origine du rectangle p^i selon l'axe horizontal.
- $p^i_{y0} \in [0, fac_h]$ est l'origine du rectangle p^i selon l'axe verticale.
- $p^i_w \in [p_{wl}, p_{wu}]$ est la longueur du rectangle p^i selon l'axe horizontal.
- $p^i_h \in [p_{hl}, p_{hu}]$ est la longueur du rectangle p^i selon l'axe vertical.

Les contraintes régissant notre modèle sont les suivantes :

- `Taille` *Les panneaux ont une plage de largeurs $p^i_w \in [p_{wl}, p_{wu}]$ et une plage de hauteurs $p^i_h \in [p_{hl}, p_{hu}]$, avec $p_{wl} = p_{hl} = 0.5m$. Certaines contraintes peuvent lier la largeur et la hauteur des panneaux pour exprimer, par exemple, des contraintes de fabrication. Dans notre cas, les deux dimensions d'un panneau ne peuvent pas être mises en même temps à leur borne supérieure : Si $p_{wu} > p_{hu}$ alors $p_w = p_h = p_{wu}$ n'est pas autorisé et, si $p_{wu} < p_{hu}$ alors $p_w = p_h = p_{hu}$ n'est pas autorisé (contrainte de fabrication).*
- `Area` *L'ensemble de la façade doit être recouvert de panneaux. Cette contrainte implique qu'aucun trou n'est autorisé dans l'enveloppe, et que la somme des surfaces des panneaux doit être égale à la surface de la façade (contrainte de métier).*
- `Non-chevauchement` *Le chevauchement entre panneaux est interdit. Cela signifie que pour deux panneaux donnés u et v il y a au moins une dimension (verticale ou horizontale) où leurs projections ne s'intersectent pas (contrainte d'installation).*
- `Menuiserie` *Chaque menuiserie de la façade doit être complètement recouverte par un et un seul panneau. De plus, chaque bord des menuiseries et chaque bord des panneaux les incluant doivent être séparés par une distance minimale notée d (contrainte de fabrication).*
- `Installation` *Les panneaux doivent être fixés par leurs quatre coins sur des zones d'accroche assez résistantes pour supporter leur poids (contrainte d'installation).*

— `Poids` *En supposant qu'une fixation peut supporter deux panneaux contigus, nous vérifions pour chaque panneau, que la moitié de son poids* $\frac{p_{we}}{2}$ *peut être supportée par un mètre carré de la zone d'accroche* sa_l^k *pour chacun de ses coins inférieurs (contrainte d'installation).*

— `Interférence` *Afin de couvrir toute la façade, un panneau donné* p^i *doit être adjacent au bord de la façade ou veiller à ce que suffisamment d'espace soit laissé pour positionner un autre panneau. Cet espace minimum dépend des bornes inférieures des panneaux dans une dimension donnée (respectivement* p_{wl} *pour le largeur des panneaux et* p_{hl} *pour la hauteur) (contrainte de métier).*

— `Ordre` *Les panneaux étant équivalents entre eux, nous les ordonnons géométriquement en imposant un ordre sur* p_{x0} *et* p_{y0}*. Cette contrainte lexicographique assure que la priorité est donnée aux panneaux rectangles pour la constitution des enveloppes. Cet ordre géométrique permet d'éviter la génération de solutions symétriques en interdisant la permutation des panneaux (contrainte de résolution).*

Contribution 2 : Deuxièmement, les solutions algorithmiques basées sur les contraintes constituent notre seconde contribution. En particulier, ces travaux de thèse présentent deux solutions manuelles (i, iv) et trois automatiques (ii, ii, v) pour le problème de calepinage :

i Premièrement, une solution interactive manuelle pour guider pas à pas la conception des enveloppes, nommée `InDiE`. Cette solution utilise les capacités d'une interface graphique avec des algorithmes de validation des panneaux, afin de permettre une configuration interactive de chaque panneau pris un à un, sur la façade considérée. La conception est faite visuellement en présentant différentes couleurs pour les panneaux bien configurés (respectant l'ensemble des contraintes évoqué) et les mal configurés (violant au moins une contrainte). Cette conception manuelle permet à l'architecte d'exprimer sa créativité lors de la conception de chaque panneau et de générer des solutions esthétiques issues de ses désirs et de sa vision du bâtiment après rénovation. Cette conception manuelle est constructive et est réalisée panneau par panneau. L'objectif du système est d'informer l'architecte de la présence d'un conflit lors de la conception de l'enveloppe isolante. Les interactions entre le système et l'utilisateur se doivent d'être en temps réel pour permettre la libre expression de l'architecte. La solution algorithmique `InDiE` peut être mise en œuvre dans tous les langages fonctionnels sans s'appuyer sur des outils complexes de type boîte noire, tels que les solveurs de contraintes, les bibliothèques de programmation linéaire ou méta-heuristiques. En outre, nous proposons une implémentation Java script orientée Web qui nous offre la possibilité d'avoir une interaction en temps réel avec l'utilisateur évitant le trafic potentiel du réseau et les retards.

ii Deuxièmement, une solution automatique pour générer plusieurs solutions de calepinage à partir d'un algorithme glouton et d'une heuristique de recherche, nommée `GaLaS`. Cette solution suit une approche constructive en prenant des décisions locales lors du positionnement et du dimensionnement des panneaux, et en résolvant les conflits (violation de contraintes) localement.

L'idée derrière la solution `GaLaS` est d'être aussi simple et rapide que possible et repose sur un algorithme glouton. Par définition, un algorithme glouton fonctionne en divisant le problème en étapes. À chaque étape, l'algorithme fait un choix qui est supposé être le meilleur dans l'état actuel de la résolution : il prend la meilleure décision locale. Puis, en cumulant les meilleures décisions locales, l'algorithme tend à converger vers une solution optimale. Cependant, de manière générale, les algorithmes gloutons n'atteignent pas la solution optimale mais des solutions proches de l'optimal. En conséquence, la solution `GaLaS` peut être qualifiée d' « aveugle » dans le sens où elle ne tient pas compte ni de la géométrie de la façade, ni de sa structure, ni des états futurs (à l'exception pour la mise en oeuvre de la contrainte d'interférence `Interférence`). Néanmoins, l'un des avantages des algorithmes gloutons consiste

à générer plusieurs solutions dans un temps de calcul raisonnable, satisfaisant ainsi l'exigence sur le temps de réponse du système, exigence identifiée dans le projet CRIBA.

Notre algorithme suit une approche constructive intuitive, à savoir, placer un par un de façon optimale les panneaux, sans tenir compte de ceux à venir. L'utilisation des algorithmes gloutons dans les problèmes de découpe et de conditionnement est également connu sous le nom de conditionnement en ligne. Le conditionnement de chaque article est réalisé à la volée, durant la phase d'exécution. Cela n'est pas le cas dans notre solution `GaLaS`, car il n'existe pas d'ensemble prédéfini de panneaux à positionner en façade à priori. La solution `GaLaS` crée durant la résolution du problème de calepinage, autant de panneaux que nécessaire pour couvrir la façade.

L'une des sous-étapes clés du problème de calepinage de façade est la définition du panneau optimal. En raison de notre objectif de minimisation du nombre de panneaux présents dans la solution, un panneau optimal est celui qui couvre la plus grande surface de la façade tout en respectant l'ensemble de contraintes du problème. Les conflits sont donc résolus à l'aide de décisions locales et optimales qui portent uniquement sur le dimensionnement des panneaux. La solution `GaLaS` exécute un retour-arrière si la décision locale ne peut être appliquée au regards des contraintes du problème.

iii Troisièmement, une solution automatique pour générer plusieurs solutions de calepinage à partir d'un algorithme de type « guillotine », nommée `CaSyE`. Cette solution exploite la géométrie et la structure de la façade pour réaliser des coupes transversales verticales et horizontales. Une partition de la façades en zones sans conflits est ainsi réalisée. Une fois la façade partitionnée, la solution `CaSyE` suit une approche gloutonne afin de positionner et dimensionner les panneaux dans chacune des zones sans conflits. Cette solution limite ainsi les retour-arrières en évitant la violation des contraintes de `Menuiserie` et d'`Installation`. Dans cette solution, la notion de symétrie des solutions de calepinage est considérée comme critère esthétique. Cette seconde solution algorithmique automatique `CaSyE` contraste avec l'approche gloutonne `GaLaS` car chaque panneau est conçu en tenant compte de la géométrie et de la structure de la façade. Elle se base, en effet, sur une approche de recherche de solutions moins aveugle qu'un algorithme glouton et génère ainsi des solutions de calepinage plus esthétiques et harmonieuses.

iv Quatrièmement, une solution manuelle pour esquisser une solution de calepinage par les architectes, nommée `SkEdE`. Cette solution repose sur un modèle de contraintes instancié à partir de l'esquisse réalisée puis résolu par le solver de contraintes `Choco`. Cette seconde solution algorithmique manuelle `SkEdE` contraste avec l'approche interactive `InDiE` sur deux aspects. Premièrement, l'esquisse de l'enveloppe est complètement dessinée avant d'être confrontée aux contraintes, et résolue en cas de conflits. Deuxièmement, cette solution exploite le solveur de contraintes `Choco` pour redimensionner et réduire la taille des panneaux mal-conçus (ne respectant pas au moins une contrainte du problème) et aboutir à une solution de calepinage valide.

Bien que le dialogue avec l'architecte soit perdu lors de la construction des solutions de calepinage avec la solution `SkEdE`, celle-ci permet à l'architecte de réaliser un croquis complet de la façade librement. Chaque panneau esquissé ne respecte pas obligatoirement l'ensemble des contraintes du problème : il peut être plus grand qu'autorisé par la contrainte `Taille` chevaucher d'autres panneaux esquissés (contrainte `Non-chevauchement`) ou des menuiseries (contrainte `Menuiserie`), etc. Lorsque l'esquisse est terminée, le *CSP* est généré et résolu : chaque panneau est redimensionné si besoin et la solution de calepinage est définie.

v Cinquièmement, une solution automatique pour générer plusieurs solutions de calepinage à partir d'un algorithme exploitant le filtrage des contraintes et une heuristique de recherche dédiée, nommée `OpackS`. Cette solution est une solution constructive,

comme la solution gloutonne GaLaS, mais elle présuppose un ensemble maximal de panneaux optionnels entrant potentiellement dans la constitution d'une enveloppe isolante. Tant que la surface de la façade n'est pas totalement recouverte de panneaux, la solution OpackS sélectionne un panneau optionnel, selon un ordre lexicographique (contrainte Ordre), le positionne et le dimensionne en respectant l'ensemble des contraintes du problème. Quand un panneau est placé sur la façade, un filtrage des valeurs inconsistantes est effectuée pour les panneaux optionnels restants. Si la surface de la façade est complètement recouverte alors les panneaux optionnels restants ne sont plus considérés et sont forcés à « non-utilisés ».

L'essence de cette solution consiste à générer des enveloppes isolantes optimales en nombre de panneaux par l'utilisation de la programmation par contraintes et plus spécifiquement, des algorithmes de filtrage et de résolution. La solution OpackS contraste avec les autres approches automatiques GaLaS et CaSyE non seulement car elle traverse de manière exhaustive l'espace de solution, mais aussi parce qu'elle tient compte des états futurs en retirant les valeurs incompatibles des panneaux optionnels avec ceux composant l'enveloppe en construction. De plus, contrairement à la solution SkEdE, où le solveur Choco est sous-jacent et résout un problème borné en nombre de panneaux, la solution OpackS doit composer avec un nombre de panneaux nécessaires à la rénovation qui est inconnu au début de la résolution et variable dans le temps. Ne pas connaître le nombre de panneaux entrant dans une solution est un inconvénient majeur pour les approches par contraintes. En effet, la grande majorité des environnements de programmation par contraintes mettent en œuvre des contraintes globales et des stratégies de recherche portant sur un ensemble fini et fixe de variables d'entrée. Pour remédier à cette situation, il est alors nécessaire d'appliquer des notions particulières comme la création dynamique de variables et de contraintes ou de permettre à des variables d'être optionnelles. Dans nos travaux de recherche, nous avons choisi d'utiliser cette notion d'optionalité de variables. Pour ce faire, le nombre de panneaux maximum est estimé (surface de la façade divisée par la surface minimale autorisée des panneaux) et l'ensemble des panneaux optionnels est généré (comme le fait l'environnement d'ordonnancement d'IBM CP optimiseur dans une dimension). Les panneaux sont activés les uns après les autres jusqu'à ce que la surface de la façade soit totalement recouverte ou qu'il soit avéré qu'il n'existe aucune solution de calepinage.

De plus, une maquette logicielle du système d'aide à la décision pour concevoir différentes solutions de calepinage, intégrant l'ensemble de nos solutions a été développée. Le développement de cette maquette est motivé par deux points majeurs. Tout d'abord, il existe une grande diversité de façades à rénover, en termes de géométrie et de propriétés structurelles, liée à l'histoire, aux techniques et aux lieux de construction. Puis, les besoins et exigences des architectes et/ou des maîtres d'ouvrage sur le style des façades après rénovation sont également nombreux et subjectifs. Les deux points soulevés ici conduisent à une grande diversité des problèmes de rénovation de façades.

Bien entendu, les méthodes traditionnelles de rénovation sont en mesure de proposer des solutions de rénovation, mais à un coût élevé. Le défi de nos travaux de recherche est d'être en mesure de proposer des solutions à l'ensemble des problèmes de rénovation de façades par l'extérieur, avec toute leur diversité, à un coût moins élevé par l'utilisation d'approches et de méthodes industrialisées (tel que défini dans le cadre du projet CRIBA).

Le problème de rénovation de façades par l'extérieur revêt toutes les particularités des problèmes de personnalisation de masse : la définition d'une solution spécifique (ou sur-mesure) à partir d'un ensemble de composants standards et paramétrables, au prix d'une solution standardisée et produite en masse. En outre, en plus de la diversité des façades et des attentes de rénovation, est associée celle liée à la diversité des solutions : pour une façade et un ensemble d'attentes de rénovation, de nombreuses solutions peuvent être identifiées, chacun d'entre elles ayant une performance thermique et un coût donnés (liés principalement au nombre de panneaux

présents dans celle-ci).

La maquette du système d'aide à la décision présentée ici met en œuvre une architecture orientée service ou *SOA* qui associe les différentes tâches de configuration à différents services. Le système permet de :

1. introduire efficacement les spécifications de rénovation, tels que les informations géométriques et structurelles des façades, les limitations sur la taille des panneaux et leur poids admissibles pour chacune d'entre elles,
2. produire différentes solutions de calepinage valides pour chaque façade, de manière manuelle, automatique et semi-automatique,
3. classer les différentes solutions en fonction de plusieurs critères, tels que leur nombre de panneaux et leur longueur de joints,
4. présenter chacune de ces solutions en deux dimensions, à la personne en charge de la rénovation, comme un architecte.

Les solutions générées par les architectes qui utilisent notre maquette d'aide à la décision sont des modèles numériques d'enveloppes isolantes, dont la nomenclature peut être envoyée aux usines de fabrication. En tant que tel, notre maquette, qui utilise la satisfaction de contraintes comme modèle sous-jacent et met en œuvre les solutions algorithmiques proposées dans cette thèse, peut être classée à la fois comme un système d'aide à la conception par ordinateur ou *CAO* et comme un logiciel de configuration de produit ou *PCS*.

7.6 Perspectives

Les perspectives de nos travaux de thèse sont nombreuses mais nous ne présenterons que les trois principales à nos yeux :

Esthétique des solutions de calepinage Pour commencer, un effort conséquent doit être mené pour améliorer nos solutions algorithmiques automatiques ou en proposer de nouvelles pour générer des solutions de calepinage plus esthétiques. En effet, dans nos solutions algorithmiques, seul le critère d'alignement des joints entre panneaux, est retenu comme critère esthétique. D'autres critères esthétiques pourraient être pris en compte dans le futur. Par exemple, un point de départ prometteur pour la génération de solutions de calepinage esthétique est l'identification et la prise en compte de motifs lors de la génération des solutions de calepinage. Ce critère, comme tous ceux permettant la génération de solutions esthétiques, pourrait faire perdre le caractère optimal des solutions en nombre de panneaux.

Calepinage avec des polygones simples Une autre piste intéressante se porte sur la géométrie des panneaux à prendre en compte. Actuellement, dans nos travaux, seuls les panneaux rectangulaires non-déformables sont considérés. Cette limitation nous conduit, par exemple, à exclure les pignons de la surface de la façade à calepiner. La définition et la modélisation de panneaux de type polygone simple, déformable ou non, permettraient de traiter l'ensemble de la surface des façades avec toutes leurs spécificités géométriques et structurelles, mais aussi de générer d'autres types de solutions et peut être plus esthétiques.

Post-traitement des solutions de calepinage Enfin, nous considérons deux possibilités de post-traitement des solutions de calepinage afin de les améliorer. La première possibilité est manuelle et permettrait, par exemple, à un architecte, de retoucher une solution simplement. Pour cela, les panneaux pourraient être un à un repris, redimensionnés par exemple, et le système automatiquement répercuterait cette modification à l'ensemble de la solution de calepinage et garantirait sa consistance. La seconde possibilité de post-processing est automatique. Suite à la génération d'une solution de calepinage, le système pourrait automatiquement redimensionner certains panneaux pour aligner leurs joints, par exemple, ou les rendre plus similaires en dimension. Ce post-traitement automatique est délicat à mettre au point au vu de la nature et de la complexité des contraintes à prendre en compte.

Bibliographie

M. Ågren, N. Beldiceanu, M. Carlsson, M. Sbihi, C. Truchet, and S. Zampelli. Six ways of integrating symmetries within non-overlapping constraints. In W. J. van Hoeve and J. N. Hooker, editors, *CPAIOR*, volume 5547 of *Lecture Notes in Computer Science*, pages 11–25. Springer, 2009. ISBN 978-3-642-01928-9.

O. Akin, B. Dave, and S. Pithavadian. Heuristic generation of layouts (hegel) : based on a paradigm for problem structuring. *Environment and Planning B : Planning and Design*, 19(1) :pp. 33 – 59, 1992. doi: 10.1068/b190033. URL http://www.envplan.com/abstract.cgi?id=b190033.

M. Aldanondo, M. Veron, and H. Fargier. Configuration in manufacturing industry requirements, problems and definitions. In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, volume 6, pages 1009–1014 vol.6, 1999. doi: 10.1109/ICSMC.1999.816691.

M. Aldanondo, A. Barco-Santa, E. Vareilles, M. Falcon, P. Gaborit, and L. Zhang. Towards a bim approach for a high performance renovation of apartment buildings. In S. Fukuda, A. Bernard, B. Gurumoorthy, and A. Bouras, editors, *Product Lifecycle Management for a Global Market*, volume 442 of *IFIP Advances in Information and Communication Technology*, pages 21–30. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-45936-2. doi: 10.1007/978-3-662-45937-9_3. URL http://dx.doi.org/10.1007/978-3-662-45937-9_3.

R. Alvarez-Valdes, F. Parreño, and J. Tamarit. A tabu search algorithm for a two-dimensional non-guillotine cutting problem. *European Journal of Operational Research*, 183(3) :1167 – 1182, 2007. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2005.11.068. URL http://www.sciencedirect.com/science/article/pii/S0377221706002979.

A. E. Ansary and M. Shalaby. Evolutionary optimization technique for site layout planning. *Sustainable Cities and Society*, 11(0) :48 – 55, 2014. ISSN 2210-6707. doi: http://dx.doi.org/10.1016/j.scs.2013.11.008. URL http://www.sciencedirect.com/science/article/pii/S2210670713000760.

B. Aranda, F. Díaz, and V. Ortíz. The problem of assigning evaluators to the articles submitted in an academic event : A practical solution incorporating constraint programming and heuristics. In P. Van Roy, editor, *Multiparadigm Programming in Mozart/Oz*, volume 3389 of *Lecture Notes in Computer Science*, pages 305–316. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25079-1. doi: 10.1007/978-3-540-31845-3_25. URL http://dx.doi.org/10.1007/978-3-540-31845-3_25.

R. Baldacci and M. A. Boschetti. A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem. *European Journal of Operational Research*, 183(3) :1136 – 1149, 2007. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2005.11.060. URL http://www.sciencedirect.com/science/article/pii/S0377221706002943.

P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-based scheduling : applying constraint programming to scheduling problems*, volume 39. Springer Science & Business Media, 2012.

P. Barahona, L. Krippahl, and O. Perriquet. Bioinformatics : A challenge to constraint programming. In P. van Hentenryck and M. Milano, editors, *Hybrid Optimization*, volume 45 of *Springer Optimization and Its Applications*, pages 463–487. Springer New York, 2011. ISBN 978-1-4419-1643-3. doi: 10.1007/978-1-4419-1644-0_14. URL http://dx.doi.org/10.1007/978-1-4419-1644-0_14.

A. Barco, E. Vareilles, M. Aldanondo, and P. Gaborit. Building thermal renovation overview. In F. Esposito, O. Pivert, M.-S. Hacid, Z. Raś, and S. Ferilli, editors, *Foundations of Intelligent Systems*, volume 9384 of *Lecture Notes in Computer Science*, pages 379–385. Springer International Publishing, 2015a. ISBN 978-3-319-25251-3.

A. Barco, E. Vareilles, P. Gaborit, and M. Aldanondo. Industrialized building renovation : Manufacturing through a constraint-based on-line support system. In *Industrial Engineering and Engineering Management (IEEM), 2015 IEEE International Conference on*, pages 947–951, Dec 2015b. doi: 10.1109/IEEM.2015.7385789.

A. Barco, M. Aldanondo, E. Vareilles, and P. Gaborit. External buildings retrofit : Employing guillotine cuts for aesthetic envelopes. In *Industrial Engineering and Engineering Management (IEEM), 2016 IEEE International Conference on*, page to be presented, Dec 2016a.

A. Barco, E. Vareilles, P. Gaborit, and M. Aldanondo. Building renovation adopts mass customization configuring insulating envelopes. *Journal of Intelligent Information Systems*, 2016b. doi: 10.1007/s10844-016-0431-6.

A. F. Barco, E. Vareilles, M. Aldanondo, and P. Gaborit. A recursive algorithm for building renovation in smart cities. In *Foundations of Intelligent Systems*, volume 8502 of *Lecture Notes in Computer Science*, pages 144–153. Springer International Publishing, 2014. ISBN 978-3-319-08325-4.

A. F. Barco, J.-G. Fages, E. Vareilles, M. Aldanondo, and P. Gaborit. Open packing for facade-layout synthesis under a general purpose solver. In G. Pesant, editor, *Principles and Practice of Constraint Programming*, volume 9255 of *Lecture Notes in Computer Science*, pages 508–523. Springer International Publishing, 2015c. ISBN 978-3-319-23218-8.

R. Barták. Constraint Programming : In Pursuit of the Holy Grail. In *Proceedings of the Week of Doctoral Students (WDS)*, June 1999.

R. Barták, M. A. Salido, and F. Rossi. Constraint satisfaction techniques in planning and scheduling. *Journal of Intelligent Manufacturing*, 21(1) :5–15, 2008. ISSN 1572-8145. doi: 10.1007/s10845-008-0203-4. URL http://dx.doi.org/10.1007/s10845-008-0203-4.

R. Barták. Dynamic global constraints in backtracking based environments. *Annals of Operations Research*, 118 (1-4) :101–119, 2003. ISSN 0254-5330. doi: 10.1023/A:1021805623454.

S. Battiato, A. Milone, and G. Puglisi. Artificial mosaic generation with gradient vector flow and tile cutting. *JECE*, 2013 :8 :8–8 :8, Jan. 2013. ISSN 2090-0147. doi: 10.1155/2013/908905. URL http://dx.doi.org/10.1155/2013/908905.

M. Bauerly and Y. Liu. Effects of symmetry and number of compositional elements on interface and design aesthetics. *International Journal of Human-Computer Interaction*, 24(3) :275–287, 2008. doi: 10.1080/10447310801920508. URL http://dx.doi.org/10.1080/10447310801920508.

C. A. Baykan and M. S. Fox. Intelligent cad systems iii : Practical experience and evaluation. In P. J. W. Ten Hagen and P. J. Veerkamp, editors, *Intelligent CAD Systems III : Practical Experience and Evaluation*, chapter Constraint Satisfaction Techniques for Spatial Planning, pages 187–204. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991. ISBN 978-3-642-84392-1. doi: 10.1007/978-3-642-84392-1_13. URL http://dx.doi.org/10.1007/978-3-642-84392-1_13.

C. A. Baykan and M. S. Fox. Artificial intelligence in engineering design (Volume I). In C. Tong and D. Sriram, editors, *Artificial Intelligence in Engineering Design, Vol. 1, Design Representation and Models of Routine Design*, chapter WRIGHT : A Constraint Based Spatial Layout System, pages 395–432. Academic Press Professional, Inc., San Diego, CA, USA, 1992. ISBN 0-12-660561-0. URL http://dl.acm.org/citation.cfm?id=140017.140037.

C. A. Baykan and M. S. Fox. Spatial synthesis by disjunctive constraint satisfaction. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 11 :245–262, 9 1997. ISSN 1469-1760. doi: 10.1017/S0890060400003206. URL http://journals.cambridge.org/article_S0890060400003206.

J. C. Beck, P. Prosser, and R. J. Wallace. Trying again to fail-first. In B. V. Faltings, A. Petcu, F. Fages, and F. Rossi, editors, *Recent Advances in Constraints : Joint ERCIM/CoLogNet International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2004, Lausanne, Switzerland, June 23-25, 2004, Revised Selected and Invited Papers*, pages 41–55. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-32252-8. doi: 10.1007/11402763_4. URL http://dx.doi.org/10.1007/11402763_4.

N. Beldiceanu, M. Carlsson, E. Poder, R. Sadek, and C. Truchet. A generic geometrical constraint kernel in space and time for handling polymorphic k-dimensional objects. In C. Bessière, editor, *Principles and Practice of Constraint Programming – CP 2007*, volume 4741 of *Lecture Notes in Computer Science*, pages 180–194. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74969-1. doi: 10.1007/978-3-540-74970-7_15. URL http://dx.doi.org/10.1007/978-3-540-74970-7_15.

N. Beldiceanu, M. Carlsson, S. Demassey, and E. Poder. New filtering for the cumulative constraint in the context of non-overlapping rectangles. *Annals of Operations Research*, 184(1) :27–50, 2011. ISSN 0254-5330. doi: 10.1007/s10479-010-0731-0. URL http://dx.doi.org/10.1007/s10479-010-0731-0.

N. Beldiceanu, M. Carlsson, P. Flener, and J. Pearson. On the reification of global constraints. *Constraints*, 18(1) :1–6, 2013. ISSN 1383-7133. doi: 10.1007/s10601-012-9132-0. URL http://dx.doi.org/10.1007/s10601-012-9132-0.

H. Ben Amor and J. Valério de Carvalho. Cutting stock problems. In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, pages 131–161. Springer US, 2005. ISBN 978-0-387-25485-2. doi: 10.1007/0-387-25486-2_5. URL http://dx.doi.org/10.1007/0-387-25486-2_5.

J. A. Bennell, J. F. Oliveira, and G. Wäscher. Cutting and packing. *International Journal of Production Economics*, 145(2) :449–450, Oct. 2013. ISSN 09255273. doi: 10.1016/j.ijpe.2013.06.021. URL http://www.sciencedirect.com/science/article/pii/S0925527313002910.

A. Bortfeldt and G. Wäscher. Constraints in container loading – a state-of-the-art review. *European Journal of Operational Research*, 229(1) :1 – 20, 2013. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2012.12.006. URL http://www.sciencedirect.com/science/article/pii/S037722171200937X.

F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting systematic search by weighting constraints. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 146–150, 2004.

S. Brailsford, P. Hubbard, B. Smith, and H. Williams. Organizing a social event—a difficult problem of combinatorial optimization. *Computers & Operations Research*, 23(9) :845 – 856, 1996. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/0305-0548(96)00001-9. URL http://www.sciencedirect.com/science/article/pii/0305054896000019.

S. C. Brailsford, C. N. Potts, and B. M. Smith. Constraint satisfaction problems : Algorithms and applications. *European Journal of Operational Research*, 119(3) :pp. 557 – 581, 1999. ISSN 0377-2217.

K. Brown. CSPLib problem 008 : Vessel loading. http://www.csplib.org/Problems/prob008, 2014.

M. Brusco and S. Stahl. Using quadratic assignment methods to generate initial permutations for least-squares unidimensional scaling of symmetric proximity matrices. *Journal of Classification*, 17(2) :197–223, 2000. ISSN 0176-4268. doi: 10.1007/s003570000019. URL http://dx.doi.org/10.1007/s003570000019.

P. Charman. Solving space planning problems using constraint technology, 1993.

N. Christofides and E. Hadjiconstantinou. An exact algorithm for orthogonal 2-d cutting problems using guillotine cuts. *European Journal of Operational Research*, 83(1) :21 – 38, 1995. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/0377-2217(93)E0277-5. URL http://www.sciencedirect.com/science/article/pii/0377221793E02775.

G. Cintra, F. Miyazawa, Y. Wakabayashi, and E. Xavier. Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *European Journal of Operational Research*, 191(1) :61 – 85, 2008. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/j.ejor.2007.08.007. URL http://www.sciencedirect.com/science/article/pii/S0377221707008831.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Greedy Algorithms*. Chapter 16 of Introduction to Algorithms, Third Edition. The MIT Press, 2009a. ISBN 0262033844, 9780262033848.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Elementary Graph Algorithms*. Chapter 22 of Introduction to Algorithms, Third Edition. The MIT Press, 2009b. ISBN 0262033844, 9780262033848.

J. Csirik and G. J. Woeginger. On-line packing and covering problems. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms : The State of the Art*, pages 147–177. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-68311-7. doi: 10.1007/BFb0029568. URL http://dx.doi.org/10.1007/BFb0029568.

Y. Cui, Y. Yang, X. Cheng, and P. Song. A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem. *Computers & Operations Research*, 35(4) :1281 – 1291, 2008. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2006.08.011. URL http://www.sciencedirect.com/science/article/pii/S0305054806001985.

G. Da Silveira, D. Borenstein, and F. S. Fogliatto. Mass customization : Literature review and research directions. *International Journal of Production Economics*, 72(1) :1–13, June 2001. ISSN 09255273. doi: 10.1016/S0925-5273(00)00079-7. URL http://www.sciencedirect.com/science/article/pii/S0925527300000797.

M. Dinar, J. Shah, J. Cagan, L. Leifer, J. Linsey, S. Smith, and N. Hernandez. Empirical studies of designer thinking : Past, present, and future. *Journal of Mechanical Design, Transactions of the ASME*, 137(2), 2 2015. ISSN 1050-0472. doi: 10.1115/1.4029025.

A. Dix, J. Finlay, G. Abowd, and R. Beale. *Human-Computer Interaction (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003. ISBN 0130461091.

I. Dotú and P. Van Hentenryck. Scheduling social golfers locally. In R. Barták and M. Milano, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3524 of *Lecture Notes in Computer Science*, pages 155–167. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-26152-0. doi: 10.1007/11493853_13. URL http://dx.doi.org/10.1007/11493853_13.

A. Drira, H. Pierreval, and S. Hajri-Gabouj. Facility layout problems : A survey. *Annual Reviews in Control*, 31(2) :255 – 267, 2007. ISSN 1367-5788. doi: http://dx.doi.org/10.1016/j.arcontrol.2007.04.001. URL http://www.sciencedirect.com/science/article/pii/S1367578807000417.

K. Dutta and S. Sarthak. Architectural space planning using evolutionary computing approaches : a review. *Artificial Intelligence Review*, 36(4) :311–321, 2011. ISSN 1573-7462. doi: 10.1007/s10462-011-9217-y. URL http://dx.doi.org/10.1007/s10462-011-9217-y.

H. Dyckhoff. Cutting and packing a typology of cutting and packing problems. *European Journal of Operational Research*, 44(2) :145 – 159, 1990. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/0377-2217(90)90350-K. URL http://www.sciencedirect.com/science/article/pii/037722179090350K.

A. N. Elshafei. Hospital layout as a quadratic assignment problem. *Operational Research Quarterly*, 28(1) :pp. 167–179, 1977.

A. Falkner and H. Schreiner. Chapter 16 - siemens : Configuration and reconfiguration in industry. In A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, editors, *Knowledge-Based Configuration*, pages 199 – 210. Morgan Kaufmann, Boston, 2014. ISBN 978-0-12-415817-7. doi: http://dx.doi.org/10.1016/B978-0-12-415817-7.00016-5. URL http://www.sciencedirect.com/science/article/pii/B9780124158177000165.

A. Falkner, A. Felfernig, and A. Haag. Recommendation technologies for configurable products. *AI Magazine*, 32 :99–108, 2011.

L. Fan, P. Musialski, L. Liu, and P. Wonka. Structure completion for facade layouts. *ACM Trans. Graph.*, 33(6) : 210 :1–210 :11, Nov. 2014. ISSN 0730-0301. doi: 10.1145/2661229.2661265. URL http://doi.acm.org/10.1145/2661229.2661265.

Z. Fan, J. Wang, and E. Goodman. An evolutionary approach for robust layout synthesis of mems. In *Advanced Intelligent Mechatronics. Proceedings, 2005 IEEE/ASME International Conference on*, pages 1186–1191, July 2005. doi: 10.1109/AIM.2005.1511171.

I. Feinerer. Efficient large-scale configuration via integer linear programming. *Artif. Intell. Eng. Des. Anal. Manuf.*, 27(1) :37–49, Jan. 2013. ISSN 0890-0604. doi: 10.1017/S0890060412000376. URL http://dx.doi.org/10.1017/S0890060412000376.

A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. Developing constraint-based recommenders. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 187–215. Springer, 2011. ISBN 978-0-387-85819-7. doi: 10.1007/978-0-387-85820-3_6. URL http://dx.doi.org/10.1007/978-0-387-85820-3_6.

A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen. *Knowledge-based Configuration : From Research to Business Cases*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1 edition, 2014a. ISBN 012415817X, 9780124158177.

A. Felfernig, S. Reiterer, F. Reinfrank, G. Ninaus, and M. Jeran. Chapter 7 - conflict detection and diagnosis in configuration. In A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, editors, *Knowledge-Based Configuration*, pages 73 – 87. Morgan Kaufmann, Boston, 2014b. ISBN 978-0-12-415817-7. doi: http://dx.doi.org/10.1016/B978-0-12-415817-7.00007-4. URL http://www.sciencedirect.com/science/article/pii/B9780124158177000074.

U. Flemming. Knowledge representation and acquisition in the LOOS system. *Building and Environment*, 25 (3) :209 – 219, 1990. ISSN 0360-1323. doi: http://dx.doi.org/10.1016/0360-1323(90)90047-U. URL http://www.sciencedirect.com/science/article/pii/0360132390090047U.

U. Flemming and R. Woodbury. Software environment to support early phases in building design (seed) : Overview. *Journal of Architectural Engineering*, 1(4) :147–152, 1995. doi: 10.1061/(ASCE)1076-0431(1995)1:4(147). URL http://dx.doi.org/10.1061/(ASCE)1076-0431(1995)1:4(147).

U. Flemming, C. Baykan, R. Coyne, and M. Fox. Hierarchical generate-and-test vs constraint-directed search. In J. Gero and F. Sudweeks, editors, *Artificial Intelligence in Design '92*, pages 817–838. Springer Netherlands, 1992. ISBN 978-94-010-5238-2. URL http://dx.doi.org/10.1007/978-94-011-2787-5_41.

J. D. Frutos, E. R. Santos, and D. Borenstein. Decision support system for product configuration in mass customization environments. *Concurrent Engineering*, 12(2) :131–144, 2004. doi: 10.1177/1063293X04044382. URL http://cer.sagepub.com/content/12/2/131.abstract.

L. Garcá-Hernández, J. M. Palomo-Romero, L. Salas-Morera, A. Arauzo-Azofra, and H. Pierreval. A novel hybrid evolutionary approach for capturing decision maker knowledge into the unequal area facility layout problem. *Expert Systems with Applications*, 42(10) :4697 – 4708, 2015. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2015.01.037. URL http://www.sciencedirect.com/science/article/pii/S0957417415000524.

Gecode Team. Gecode : Generic constraint development environment, 2006. Available from http://www.gecode.org.

E. Gelle and R. Weigel. Interactive configuration using constraint satisfaction techniques. In *In Second International Conference on Practical Application of Constraint Technology, PACT-96*, pages 37–44. Menlo Park, AAAI Press, 1996.

J. A. George, J. M. George, and B. W. Lamar. Packing different-sized circles into a rectangular container. *European Journal of Operational Research*, 84(3) :693 – 712, 1995. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/0377-2217(95)00032-L. URL http://www.sciencedirect.com/science/article/pii/037722179500032L. Cutting and Packing.

S. Gholizadeh-Tayyar, J. Lamothe, and L. Dupont. *Integration of Supply Chain Planning with Time and Resource Constrained Project Scheduling Problems for Building's Thermal Renovation Projects*, pages 566–577. Springer International Publishing, Cham, 2015. ISBN 978-3-319-24141-8. doi: 10.1007/978-3-319-24141-8_53. URL http://dx.doi.org/10.1007/978-3-319-24141-8_53.

D. Glover, D. Miller, D. Averis, and V. Door. The interactive whiteboard : a literature survey. *Technology, Pedagogy and Education*, 14(2) :155–170, 2005. doi: 10.1080/14759390500200199. URL http://dx.doi.org/10.1080/14759390500200199.

V. Godin. Interactive scheduling : Historical survey and state of the art. *A I I E Transactions*, 10(3) :331–337, 1978. doi: 10.1080/05695557808975222. URL http://dx.doi.org/10.1080/05695557808975222.

M. Goetschalckx. An interactive layout heuristic based on hexagonal adjacency graphs. *European Journal of Operational Research*, 63(2) :pp. 304 – 321, 1992.

L. Greco and M. Cascia. Saliency based aesthetic cut of digital images. In A. Petrosino, editor, *Image Analysis and Processing – ICIAP 2013 : 17th International Conference, Naples, Italy, September 9-13, 2013, Proceedings, Part II*, pages 151–160. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-41184-7. doi: 10.1007/978-3-642-41184-7_16. URL http://dx.doi.org/10.1007/978-3-642-41184-7_16.

A. Haag. Sales configuration in business processes. *IEEE Intelligent Systems and their Applications*, 13(4) :78–85, Jul 1998. ISSN 1094-7167. doi: 10.1109/5254.708436.

R. M. Haralick and G. L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3) :263 – 313, 1980. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/0004-3702(80)90051-X. URL http://www.sciencedirect.com/science/article/pii/000437028090051X.

S. Helber, D. Böhme, F. Oucherif, S. Lagershausen, and S. Kasper. A hierarchical facility layout planning approach for large and complex hospitals. *Flexible Services and Manufacturing Journal*, pages 1–25, 2015. ISSN 1936-6582. doi: 10.1007/s10696-015-9214-6. URL http://dx.doi.org/10.1007/s10696-015-9214-6.

P. V. Hentenryck, V. Saraswat, and Y. Deville. Design, implementation, and evaluation of the constraint language cc(fd). *The Journal of Logic Programming*, 37(1–3) :139 – 164, 1998. ISSN 0743-1066. doi: http://dx.doi.org/10.1016/S0743-1066(98)10006-7. URL http://www.sciencedirect.com/science/article/pii/S0743106698100067.

N. V. Hernandez, L. C. Schmidt, G. O. Kremer, and C.-Y. Lin. An empirical study of the effectiveness of selected cognitive aids on multiple design tasks. In S. J. Gero, editor, *Design Computing and Cognition '12*, pages 227–246. Springer Netherlands, Dordrecht, 2014. ISBN 978-94-017-9112-0. doi: 10.1007/978-94-017-9112-0_13. URL http://dx.doi.org/10.1007/978-94-017-9112-0_13.

M. Hifi and R. Ouafi. A best-first branch-and-bound algorithm for orthogonal rectangular packing problems. *International Transactions in Operational Research*, 5(5) :345 – 356, 1998. ISSN 0969-6016. doi: http://dx.doi.org/10.1016/S0969-6016(98)00026-4. URL http://www.sciencedirect.com/science/article/pii/S0969601698000264.

E. Hopper and B. Turton. A review of the application of meta-heuristic algorithms to 2d strip packing problems. *Artificial Intelligence Review*, 16(4) :257–300, 2001a. ISSN 0269-2821. doi: 10.1023/A:1012590107280.

E. Hopper and B. C. H. Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*, 128 :34–57, 2000.

E. Hopper and B. C. H. Turton. A review of the application of meta-heuristic algorithms to 2d strip packing problems. *Artificial Intelligence Review*, 16(4) :257–300, 2001b. ISSN 1573-7462. doi: 10.1023/A:1012590107280. URL http://dx.doi.org/10.1023/A:1012590107280.

E. Huang and R. E. Korf. New improvements in optimal rectangle packing. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 511–516, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1661445.1661527.

E. Huang and R. E. Korf. Optimal rectangle packing : An absolute placement approach. *CoRR*, abs/1402.0557, 2014. URL http://arxiv.org/abs/1402.0557.

W. Huang and D. Chen. *An Efficient Quasi-Human Heuristic Algorithm for Solving the Rectangle-Packing Problem*. Simulated Annealing, Cher Ming Tan, InTech, 2008. ISBN 978-953-7619-07-7. doi: 10.5772/5562. URL http://www.intechopen.com/books/simulated_annealing/an_efficient_quasi-human_heuristic_algorithm_for_solving_the__rectangle-packing_problem.

F. Hutter, H. Hoos, and K. Leyton-Brown. Automated configuration of mixed integer programming solvers. In A. Lodi, M. Milano, and P. Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140 of *Lecture Notes in Computer Science*, pages 186–202. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13519-4. doi: 10.1007/978-3-642-13520-0_23. URL http://dx.doi.org/10.1007/978-3-642-13520-0_23.

B. Höfling. Chapter 18 - encoway : From erp-based to sales-oriented configuration. In A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, editors, *Knowledge-Based Configuration*, pages 219 – 227. Morgan Kaufmann, Boston, 2014. ISBN 978-0-12-415817-7. doi: http://dx.doi.org/10.1016/B978-0-12-415817-7.00018-9. URL http://www.sciencedirect.com/science/article/pii/B9780124158177000189.

S. Imahori, M. Yagiura, and H. Nagamochi. *Practical algorithms for two-dimensional packing*. Chapter 36 of Handbook of Approximation Algorithms and Metaheuristics (Chapman & Hall/Crc Computer & Information Science Series), 2007. ISBN 1584885505.

C. Iris and D. Pacino. A survey on the ship loading problem. In F. Corman, S. Voß, and R. R. Negenborn, editors, *Computational Logistics : 6th International Conference, ICCL 2015, Delft, The Netherlands, September 23-25, 2015, Proceedings*, pages 238–251. Springer International Publishing, Cham, 2015. ISBN 978-3-319-24264-4. doi: 10.1007/978-3-319-24264-4_17. URL http://dx.doi.org/10.1007/978-3-319-24264-4_17.

N. Izadinia, K. Eshghi, and M. H. Salmani. A robust model for multi-floor layout problem. *Computers and Industrial Engineering*, 78(0) :127 – 134, 2014. ISSN 0360-8352. doi: http://dx.doi.org/10.1016/j.cie.2014.09.023. URL http://www.sciencedirect.com/science/article/pii/S0360835214002903.

T. Jacobsen and L. Höfel. Descriptive and evaluative judgment processes : Behavioral and electrophysiological indices of processing symmetry and aesthetics. *Cognitive, Affective, & Behavioral Neuroscience*, 3(4) :289–299, 2003. ISSN 1531-135X. doi: 10.3758/CABN.3.4.289. URL http://dx.doi.org/10.3758/CABN.3.4.289.

J. Jankowski and M. Hachet. A Survey of Interaction Techniques for Interactive 3D Environments. In *Eurographics 2013 - STAR*, Girona, Spain, May 2013. URL https://hal.inria.fr/hal-00789413.

C. Jefferson, N. C. Moore, P. Nightingale, and K. E. Petrie. Implementing logical connectives in constraint programming. *Artificial Intelligence*, 174(16–17) :1407 – 1429, 2010. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/j.artint.2010.07.001. URL http://www.sciencedirect.com/science/article/pii/S0004370210000974.

K. A. D. Julia A. Bennell. Hybridising tabu search with optimisation techniques for irregular stock cutting. *Management Science*, 47(8) :1160–1172, 2001. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/822601.

U. Junker. *Configuration*. Chapter 24 of Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc., New York, NY, USA, 2006. ISBN 0444527265.

J. Jylänki. A thousand ways to pack the bin - a practical approach to two-dimensional rectangle bin packing, 2010. URL http://clb.demon.fi/files/RectangleBinPack.pdf. Research report. Available at : http://clb.demon.fi/files/RectangleBinPack.pdf.

T. Kokeny. A new arc consistency algorithm for csps with hierarchical domains. In *Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on*, pages 439–445, Nov 1994. doi: 10.1109/TAI.1994.346459.

R. E. Korf. Optimal rectangle packing : Initial results. In E. Giunchiglia, N. Muscettola, and D. S. Nau, editors, *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003), June 9-13, 2003, Trento, Italy*, pages 287–295. AAAI, 2003. ISBN 1-57735-187-8. URL http://www.aaai.org/Library/ICAPS/2003/icaps03-029.php.

M. Krishnan, T. Karthikeyan, T. Chinnusamy, and A. Murugesan. An evolutionary hybrid algorithm for layout planning in flexible manufacturing system. *Advanced Materials Research*, 984-985(0) :444–451, 2014. ISSN 1662-8985. doi: 10.4028/www.scientific.net/AMR.984-985.444. URL www.scientific.net/AMR.984-985.444.

S. Kumar and V. Bansal. Review of information technology-based tools in construction site layout planning. In *Proceedings of the 7th International Conference on Engineering Mechanics, Structures, Engineering Geology (EMESEG '14)*, pages 196–201. WSEAS Press, 2014.

P. Laborie. IBM ILOG CP optimizer for detailed scheduling illustrated on three problems. In W.-J. van Hoeve and J. Hooker, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 5547 of *Lecture Notes in Computer Science*, pages 148–162. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01928-9. doi: 10.1007/978-3-642-01929-6_12. URL http://dx.doi.org/10.1007/978-3-642-01929-6_12.

A. Letort, M. Carlsson, and N. Beldiceanu. Synchronized sweep algorithms for scalable scheduling constraints. *Constraints*, 20(2) :183–234, 2014. ISSN 1572-9354. doi: 10.1007/s10601-014-9172-8. URL http://dx.doi.org/10.1007/s10601-014-9172-8.

S. C. Leung, X. Zhou, D. Zhang, and J. Zheng. Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, 38(1) :205 – 215, 2011. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2010.04.013. URL http://www.sciencedirect.com/science/article/pii/S0305054810000973. Project Management and Scheduling.

L. Y. Liang and W. C. Chao. The strategies of tabu search technique for facility layout optimization. *Automation in Construction*, 17(6) :657 – 669, 2008. ISSN 0926-5805. doi: http://dx.doi.org/10.1016/j.autcon.2008.01.001. URL http://www.sciencedirect.com/science/article/pii/S0926580508000034.

R. S. Liggett. Automated facilities layout : past, present and future. *Automation in Construction*, 9(2) : 197 – 215, 2000. ISSN 0926-5805. doi: http://dx.doi.org/10.1016/S0926-5805(99)00005-9. URL http://www.sciencedirect.com/science/article/pii/S0926580599000059.

A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems : A survey. *European Journal of Operational Research*, 141(2) :241 – 252, 2002. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/S0377-2217(02)00123-6. URL http://www.sciencedirect.com/science/article/pii/S0377221702001236.

M. Lombardi and M. Milano. Optimal methods for resource allocation and scheduling : a cross-disciplinary survey. *Constraints*, 17(1) :51–85, 2012. ISSN 1383-7133. doi: 10.1007/s10601-011-9115-6. URL http://dx.doi.org/10.1007/s10601-011-9115-6.

R. Marler and J. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6) :369–395, 2004. ISSN 1615-1488. doi: 10.1007/s00158-003-0368-6. URL http://dx.doi.org/10.1007/s00158-003-0368-6.

S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip-packing problem. *INFORMS J. on Computing*, 15(3) :310–319, July 2003. ISSN 1526-5528. doi: 10.1287/ijoc.15.3.310.16082. URL http://dx.doi.org/10.1287/ijoc.15.3.310.16082.

F. McKoy, N. Vargas-Hernández, J. Summers, and J. Shah. Influence of design representation on effectiveness of idea generation. In *Proceedings of the ASME Design Engineering Technical Conference*, volume 4, pages 39–48. Elsevier Science, 12 2001.

I. C. McMANUS. Symmetry and asymmetry in aesthetics and the arts. *European Review*, 13 :157–180, 10 2005. ISSN 1474-0575. doi: 10.1017/S1062798705000736. URL http://journals.cambridge.org/article_S1062798705000736.

B. Medjdoub and B. Yannou. Separating topology and geometry in space planning. *Computer-Aided Design*, 32 (1) :39 – 61, 2000. ISSN 0010-4485. doi: http://dx.doi.org/10.1016/S0010-4485(99)00084-6.

B. Medjdoub and B. Yannou. Dynamic space ordering at a topological level in space planning. *Artificial Intelligence in Engineering*, 15(1) :47 – 60, 2001. ISSN 0954-1810. doi: http://dx.doi.org/10.1016/S0954-1810(00)00027-3.

J. Michalek and P. Papalambros. Interactive design optimization of architectural layouts. *Engineering Optimization*, 34(5) :485–501, 2002. doi: 10.1080/03052150214021. URL http://dx.doi.org/10.1080/03052150214021.

L. Michel and P. Van Hentenryck. *Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems : 9th International Conference, CPAIOR 2012, Nantes, France, May 28 – June1, 2012. Proceedings*, chapter Activity-Based Search for Black-Box Constraint Programming Solvers, pages 228–243. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-29828-8. doi: 10.1007/978-3-642-29828-8_15. URL http://dx.doi.org/10.1007/978-3-642-29828-8_15.

I. G. C. J. I. Miguel. Minion : A fast, scalable, constraint solver. In *The European Conference on Artificial Intelligence*, Trentino Italy, 2006. IOS Press.

S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In *AAAI*, pages 25–32, 1990.

U. Montanari. Networks of constraints : Fundamental properties and applications to picture processing. *Information Sciences*, 7(0) :95 – 132, 1974. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/0020-0255(74)90008-5. URL http://www.sciencedirect.com/science/article/pii/0020025574900085.

T. Müller. *Constraint Propagation in Mozart*. Doctoral dissertation, Universität des Saarlandes, Naturwissenschaftlich-Technische Fakultät I, Fachrichtung Informatik, Saarbrücken, Germany, 2001.

S. Myung and S. Han. Knowledge-based parametric design of mechanical products based on configuration design method. *Expert Systems with Applications*, 21(2) :99 – 107, 2001. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/S0957-4174(01)00030-6. URL http://www.sciencedirect.com/science/article/pii/S0957417401000306.

J. Neliben. How to use structural constraints to compute an upper bound for the pallet loading problem. *European Journal of Operational Research*, 84(3) :662 – 680, 1995. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/0377-2217(95)00030-T. URL http://www.sciencedirect.com/science/article/pii/037722179500030T. Cutting and Packing.

I. Nica, F. Wotawa, R. Ochenbauer, C. Schober, H. Hofbauer, and S. Boltek. Chapter 19 - kapsch : Reconfiguration of mobile phone networks. In A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, editors, *Knowledge-Based Configuration*, pages 229 – 240. Morgan Kaufmann, Boston, 2014. ISBN 978-0-12-415817-7. doi: http://dx.doi.org/10.1016/B978-0-12-415817-7.00019-0. URL http://www.sciencedirect.com/science/article/pii/B9780124158177000190.

N. Ntene and J. van Vuuren. A survey and comparison of guillotine heuristics for the 2d oriented offline strip packing problem. *Discrete Optimization*, 6(2) :174 – 188, 2009. ISSN 1572-5286. doi: http://dx.doi.org/10.1016/j.disopt.2008.11.002. URL http://www.sciencedirect.com/science/article/pii/S1572528608000844.

OpenRules, Inc. Constraint programming solvers catalog, 2013. Available from `http://openjvm.jvmhost.net/CPSolvers/`.

K. Orsvärn and M. H. Bennick. Chapter 17 - tacton : Use of tacton configurator at {FLSmidth}. In A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, editors, *Knowledge-Based Configuration*, pages 211 – 218. Morgan Kaufmann, Boston, 2014. ISBN 978-0-12-415817-7. doi: http://dx.doi.org/10.1016/B978-0-12-415817-7.00017-7. URL http://www.sciencedirect.com/science/article/pii/B9780124158177000177.

M. Pan and Y. Rao. An integrated knowledge based system for sheet metal cutting–punching combination processing. *Knowledge-Based Systems*, 22(5) :368–375, July 2009. ISSN 09507051. doi: 10.1016/j.knosys.2009.02.008. URL http://www.sciencedirect.com/science/article/pii/S0950705109000434.

O. Pantalé, J.-L. Bacaria, O. Dalverny, R. Rakotomalala, and S. Caperaa. 2D and 3D numerical models of metal cutting with damage effects. *Computer Methods in Applied Mechanics and Engineering*, 193(39-41) : 4383–4399, Oct. 2004. ISSN 00457825. doi: 10.1016/j.cma.2003.12.062. URL http://www.sciencedirect.com/science/article/pii/S0045782504002269.

D. Pisinger and S. Ropke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 399–419. Springer US, Boston, MA, 2010. ISBN 978-1-4419-1665-5. doi: 10.1007/978-1-4419-1665-5_13. URL http://dx.doi.org/10.1007/978-1-4419-1665-5_13.

C. Prud'homme and J. Fages. An introduction to choco 3.0 an open source java constraint programming library. In *CP Solvers : Modeling, Applications, Integration, and Standardization. International workshop.*, Uppsala Sweden, 2013.

C. Prud'homme, J.-G. Fages, and X. Lorca. *Choco3 Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2014. URL http://www.choco-solver.org.

L. Pérez-Lombard, J. Ortiz, and C. Pout. A review on buildings energy consumption information. *Energy and Buildings*, 40(3) :394 – 398, 2008. ISSN 0378-7788.

R. Rabiser, M. Vierhauser, M. Lehofer, P. Grünbacher, and T. Männistö. Chapter 20 - configuring and generating technical documents. In A. Felfernig, L. Hotz, C. Bagley, and J. Tiihonen, editors, *Knowledge-Based Configuration*, pages 241 – 250. Morgan Kaufmann, Boston, 2014. ISBN 978-0-12-415817-7. doi: http://dx.doi.org/10.1016/B978-0-12-415817-7.00020-7. URL http://www.sciencedirect.com/science/article/pii/B9780124158177000207.

P. Refalo. *Principles and Practice of Constraint Programming – CP 2004 : 10th International Conference, CP 2004, Toronto, Canada, September 27 -October 1, 2004. Proceedings*, chapter Impact-Based Search Strategies for Constraint Programming, pages 557–571. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-30201-8. doi: 10.1007/978-3-540-30201-8_41. URL http://dx.doi.org/10.1007/978-3-540-30201-8_41.

E. Rodrigues, A. R. Gaspar, and Álvaro Gomes. Automated approach for design generation and thermal assessment of alternative floor plans. *Energy and Buildings*, 81(0) :170 – 181, 2014. ISSN 0378-7788. doi: http://dx.doi.org/10.1016/j.enbuild.2014.06.016.

F. Rossi, P. v. Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006. ISBN 0444527265.

D. Sabin and E. Freuder. Configuration as composite constraint satisfaction. In *in Proc. Artificial Intelligence and Manufacturing. Research Planning Workshop*, pages 153–161. AAAI Press, 1996.

D. Sabin and R. Weigel. Product configuration frameworks-a survey. *IEEE Intelligent Systems*, 13(4) :42–49, July 1998. ISSN 1541-1672. doi: 10.1109/5254.708432. URL http://dx.doi.org/10.1109/5254.708432.

G. Schenner, A. A. Falkner, A. Ryabokon, and G. Friedrich. Solving object-oriented configuration scenarios with ASP. In M. Aldanondo and A. A. Falkner, editors, *Proceedings of the 15th International Configuration Workshop, Vienna, Austria, August 29-30, 2013.*, volume 1128 of *CEUR Workshop Proceedings*, pages 55–62. CEUR-WS.org, 2013. URL http://ceur-ws.org/Vol-1128/paper8.pdf.

J. Schimpf and K. Shen. Eclipse - from lp to clp. *CoRR*, abs/1012.4240, 2010.

J. Schimpf and K. Shen. Eclipse-from lp to clp. *Theory Pract. Log. Program.*, 12(1-2) :127–156, Jan. 2012. ISSN 1471-0684. doi: 10.1017/S1471068411000469.

Y. Schreiber. Value-ordering heuristics : Search performance vs. solution diversity. In D. Cohen, editor, *Principles and Practice of Constraint Programming – CP 2010 : 16th International Conference, CP 2010, St. Andrews, Scotland, September 6-10, 2010. Proceedings*, pages 429–444. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-15396-9. doi: 10.1007/978-3-642-15396-9_35. URL http://dx.doi.org/10.1007/978-3-642-15396-9_35.

C. Schulte. *Programming Constraint Services*, volume 2302 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2002. URL http://link.springer.de/link/service/series/0558/tocs/t2302.htm.

C. Schulte and M. Carlsson. Chapter 14 - finite domain constraint programming systems. In P. v. B. Francesca Rossi and T. Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 495 – 526. Elsevier, 2006. doi: http://dx.doi.org/10.1016/S1574-6526(06)80018-0. URL http://www.sciencedirect.com/science/article/pii/S1574652606800180.

C. Schulte, G. Smolka, and J. Würtz. Finite domain constraint programming in oz – a tutorial. In *Programming Systems Lab, German Research Center for Artificial Intelligence (DFKI)*, 1998.

C. Schulte, G. Tack, and M. Lagerkvist. Modeling and programming with Gecode, 2010. URL http://www.gecode.org/doc-latest/MPG.pdf.

A. Schutt, T. Feydy, and P. Stuckey. Scheduling optional tasks with explanation. In C. Schulte, editor, *Principles and Practice of Constraint Programming*, volume 8124 of *Lecture Notes in Computer Science*, pages 628–644. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40626-3. doi: 10.1007/978-3-642-40627-0_47. URL http://dx.doi.org/10.1007/978-3-642-40627-0_47.

F. Senel, B. Cetisli, S. Guvenc, and I. Tasci. Optimum placement of the cutting patterns on the leather with image processing and optimization. In *Signal Processing and Communications Applications Conference (SIU), 2015 23th*, pages 1216–1219, May 2015. doi: 10.1109/SIU.2015.7130056.

S. Shikder, A. Price, and M. Mourshed. Interactive constraint-based space layout planning. In *Barrett, P. et al. (eds). Proceedings of CIB World Building Congress 2010 : Building a Better World*. CIB Publishers, 2010. ISBN 9781905732906. URL https://dspace.lboro.ac.uk/2134/9768.

W. Shin and A. Ravindran. Interactive multiple objective optimization : Survey i—continuous case. *Computers & Operations Research*, 18(1) :97 – 114, 1991. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/0305-0548(91)90046-T. URL http://www.sciencedirect.com/science/article/pii/030505489190046T.

B. M. Smith. Modelling for constraint programming, 2005. URL http://www.math.unipd.it/~frossi/cp-school/new-barbara-tutorial.pdf.

T. Soininen, J. Tiihonen, T. Männistö, and R. Sulonen. Towards a general ontology of configuration. *Artif. Intell. Eng. Des. Anal. Manuf.*, 12(4) :357–372, Sept. 1998. ISSN 0890-0604. doi: 10.1017/S0890060498124083. URL http://dx.doi.org/10.1017/S0890060498124083.

M. Solimanpur and A. Jafari. Optimal solution for the two-dimensional facility layout problem using a branch-and-bound algorithm. *Computers and Industrial Engineering*, 55(3) :606 – 619, 2008. ISSN 0360-8352. doi: http://dx.doi.org/10.1016/j.cie.2008.01.018. URL http://www.sciencedirect.com/science/article/pii/S0360835208000387.

M. Solimanpur, P. Vrat, and R. Shankar. Ant colony optimization algorithm to the inter-cell layout problem in cellular manufacturing. *European Journal of Operational Research*, 157(3) :592 – 606, 2004. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/S0377-2217(03)00248-0. URL http://www.sciencedirect.com/science/article/pii/S0377221703002480.

T. Strecker and L. Hennig. Automatic layouting of personalized newspaper pages. In B. Fleischmann, K.-H. Borgwardt, R. Klein, and A. Tuma, editors, *Operations Research Proceedings 2008 : Selected Papers of the Annual International Conference of the German Operations Research Society (GOR) University of Augsburg, September 3-5, 2008*, pages 469–474. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-00142-0. doi: 10.1007/978-3-642-00142-0_76. URL http://dx.doi.org/10.1007/978-3-642-00142-0_76.

R. Stuyver and J. Hennessey. A support tool for the conceptual phase of design. In M. A. R. Kirby, A. J. Dix, and J. Finlay, editors, *People and Computers X, Proceedings of HCI '95, Huddersfield, August 1995*, pages 235–245. Cambridge University Press, 1995. ISBN 0-521-56729-7.

M. Suwa and B. Tversky. What architects see in their sketches : Implications for design tools. In *Conference Companion on Human Factors in Computing Systems*, CHI '96, pages 191–192, New York, NY, USA, 1996. ACM. ISBN 0-89791-832-0. doi: 10.1145/257089.257255. URL http://doi.acm.org/10.1145/257089.257255.

O. Teboul, L. Simon, P. Koutsourakis, and N. Paragios. Segmentation of building facades using procedural shape priors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3105–3112, June 2010. doi: 10.1109/CVPR.2010.5540068.

J. Tiihonen and A. Felfernig. Towards recommending configurable offerings. *International Journal of Mass Customisation*, 3(4) :389–406, 2010. doi: 10.1504/IJMASSC.2010.037652. URL http://www.inderscienceonline.com/doi/abs/10.1504/IJMASSC.2010.037652.

M. Triska. The finite domain constraint solver of swi-prolog. In T. Schrijvers and P. Thiemann, editors, *Functional and Logic Programming*, volume 7294 of *Lecture Notes in Computer Science*, pages 307–316. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29821-9. doi: 10.1007/978-3-642-29822-6_24.

A. N. Tuch, J. A. Bargas-Avila, and K. Opwis. Symmetry and aesthetics in website design : It's a man's business. *Computers in Human Behavior*, 26(6) :1831 – 1837, 2010. ISSN 0747-5632. doi: http://dx.doi.org/10.1016/j.chb.2010.07.016. URL http://www.sciencedirect.com/science/article/pii/S0747563210002128. Online Interactivity : Role of Technology in Behavior Change.

B. Tversky. What do sketches say about thinking ?, 2012. In Proceedings of AAAI Spring Symposium on Sketch Understanding, pp. 148–151.

W. van Hoeve and I. Katriel. *Global Constraints*. Chapter 7 of Handbook of Constraint Programming (Foundations of Artificial Intelligence). Elsevier Science Inc., New York, NY, USA, 2006. ISBN 0444527265.

W. J. van Hoeve and J. N. Hooker, editors. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 6th International Conference, CPAIOR 2009, Pittsburgh, PA, USA, May 27-31, 2009, Proceedings*, volume 5547 of *Lecture Notes in Computer Science*, 2009. Springer. ISBN 978-3-642-01928-9. doi: 10.1007/978-3-642-01929-6. URL http://dx.doi.org/10.1007/978-3-642-01929-6.

W.-J. van Hoeve and J.-C. Régin. Open constraints in a closed world. In J. Beck and B. Smith, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3990 of *Lecture Notes in Computer Science*, pages 244–257. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-34306-6. doi: 10.1007/11757375_20. URL http://dx.doi.org/10.1007/11757375_20.

E. Vareilles, P. Gaborit, M. Aldanondo, S. Carbonnel, and L. Steffan. CoFiADe Constraints Filtering for Aiding Design. In *JFPC 2012*, page paper9, Toulouse, France, May 2012. URL https://hal.archives-ouvertes.fr/hal-00811512. 6 pages.

E. Vareilles, A. Barco Santa, M. Falcon, M. Aldanondo, and P. Gaborit. Configuration of high performance apartment buildings renovation : A constraint based approach. In *Industrial Engineering and Engineering Management (IEEM), 2013 IEEE International Conference on*, pages 684–688, Dec 2013. doi: 10.1109/IEEM.2013.6962498.

T. Walsh. Symmetry breaking using value precedence. *CoRR*, abs/0903.1136, 2009. URL http://arxiv.org/abs/0903.1136.

G. Wäscher, H. Hauß ner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3) :1109–1130, Dec. 2007. ISSN 03772217. doi: 10.1016/j.ejor.2005.12.047. URL http://www.sciencedirect.com/science/article/pii/S037722170600292X.

J. Wielemaker, T. Schrijvers, M. Triska, and T. Lager. Swi-prolog. *Computing Research Repository*, abs/1011.5332, 2010.

J. Wielemaker, T. Schrijvers, M. Triska, and T. Lager. SWI-Prolog. *Theory and Practice of Logic Programming*, 12(1-2) :67–96, 2012. ISSN 1471-0684.

F. Wu, D.-M. Yan, W. Dong, X. Zhang, and P. Wonka. Inverse procedural modeling of facade layouts. *ACM Trans. Graph.*, 33(4) :121 :1–121 :10, July 2014. ISSN 0730-0301. doi: 10.1145/2601097.2601162. URL http://doi.acm.org/10.1145/2601097.2601162.

J. Würtz and T. Müller. Constructive disjunction revisited. In *Proceedings of the 20th Annual German Conference on Artificial Intelligence : Advances in Artificial Intelligence*, KI '96, pages 377–386, London, UK, UK, 1996. Springer-Verlag. ISBN 3-540-61708-6. URL http://dl.acm.org/citation.cfm?id=647616.731575.

H. Xie, P. Henderson, and M. Kernahan. Modelling and solving engineering product configuration problems by constraint satisfaction. *International Journal of Production Research*, 43(20) :4455–4469, 2005. doi: 10.1080/00207540500142381.

H. H. Yanasse, A. S. I. Zinober, and R. G. Harris. Two-dimensional cutting stock with multiple stock sizes. *The Journal of the Operational Research Society*, 42(8) :pp. 673–683, 1991. ISSN 01605682. URL http://www.jstor.org/stable/2583786.

D. Yang, M. Dong, and R. Miao. Development of a product configuration system with an ontology-based approach. *Computer-Aided Design*, 40(8) :863 – 878, 2008. ISSN 0010-4485. doi: http://dx.doi.org/10.1016/j.cad.2008.05.004. URL http://www.sciencedirect.com/science/article/pii/S0010448508001061.

D. Yang, M. Dong, and X. Chang. A dynamic constraint satisfaction approach for configuring structural products under mass customization. *Engineering Applications of Artificial Intelligence*, 25(8) :1723 – 1737, 2012. ISSN 0952-1976.

K. Yue, C. Hickerson, and R. Krishnamurti. Determining the interior layout of buildings describable by shape grammars. In *Proceedings of the 13th international conference on Computer aided architectural design research in Asia [CAADRIA '08]*. CUMINCAD, 2008. URL http://cumincad.scix.net/cgi-bin/works/Show?_id=caadria2008_14_session2a_117.

M. Zawidzki, K. Tateyama, and I. Nishikawa. The constraints satisfaction problem approach in the design of an architectural functional layout. *Engineering Optimization*, 43(9) :943–966, 2011. doi: 10.1080/0305215X.2010.527005. URL http://dx.doi.org/10.1080/0305215X.2010.527005.

A. Ziesemer, R. Reis, M. Moreira, M. Arendt, and N. Calazans. Automatic layout synthesis with astran applied to asynchronous cells. In *Circuits and Systems (LASCAS), 2014 IEEE 5th Latin American Symposium on*, pages 1–4, Feb 2014. doi: 10.1109/LASCAS.2014.6820314.

Appendices

A

Support System Demonstration

The remaining of the section presents the execution of the system over real façades on France.

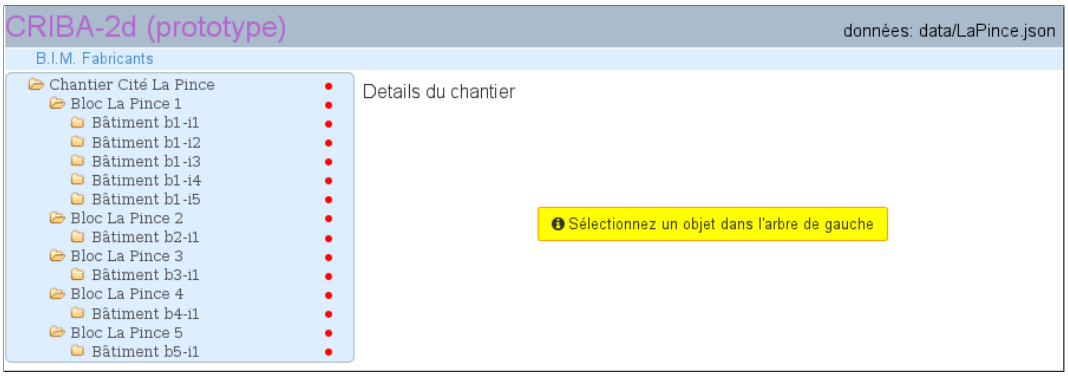


FIGURE A.1 – After opening a working site specification, every block, building and façade is parsed and shown in the interface.

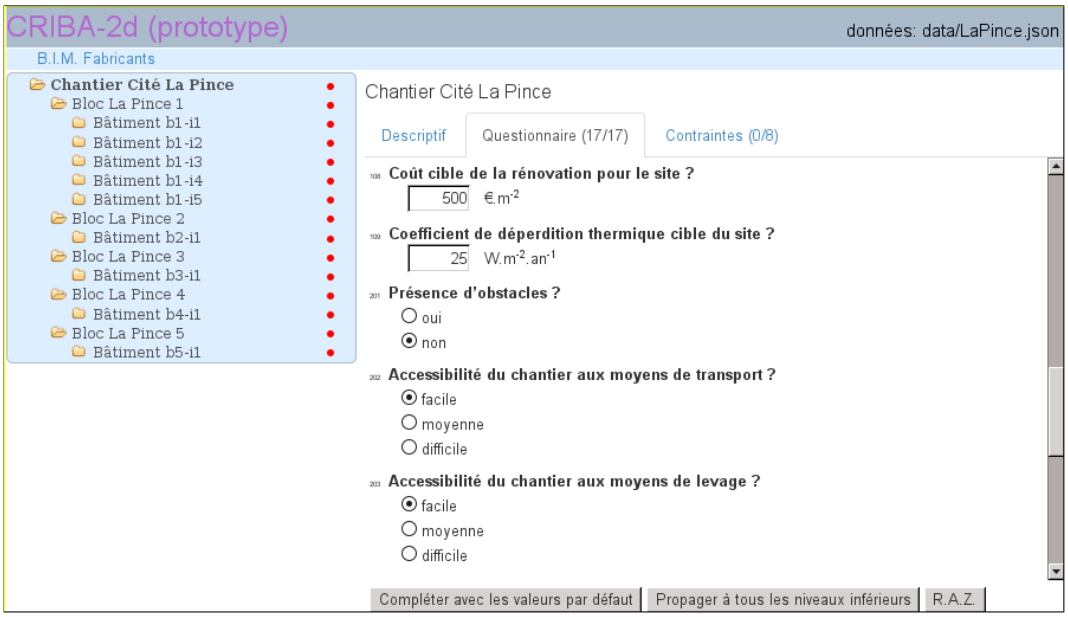


FIGURE A.2 – For every spatial entity the system presents a questionnaire about its accesibility conditions.

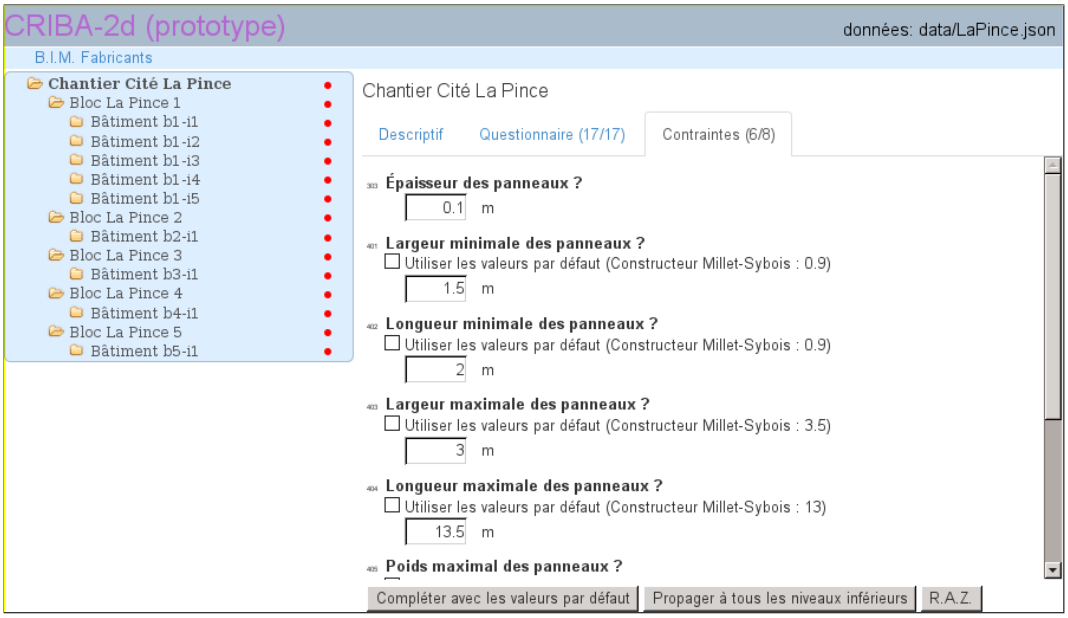


FIGURE A.3 – For every spatial entity the system presents a questionnaire about the panels' constraints.

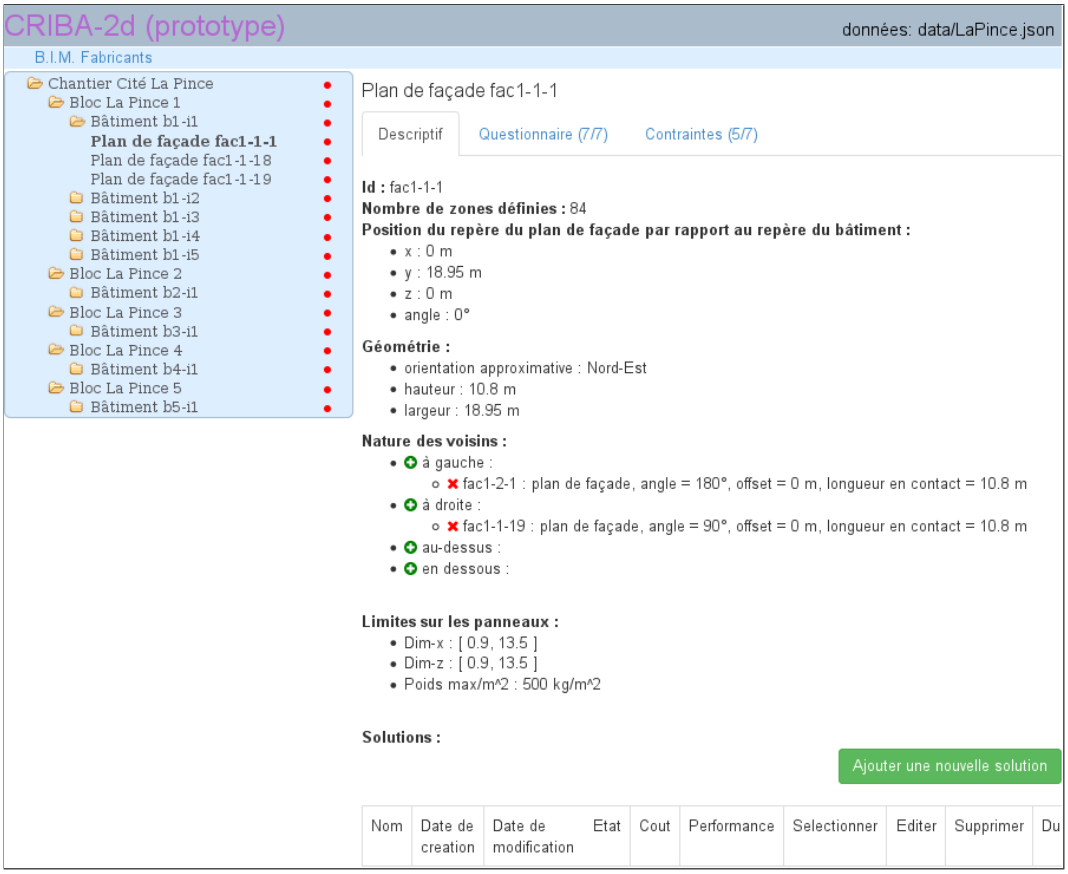


FIGURE A.4 – When selecting a façade, the geometrical properties and the panels' bounds are shown. The design button is enabled.

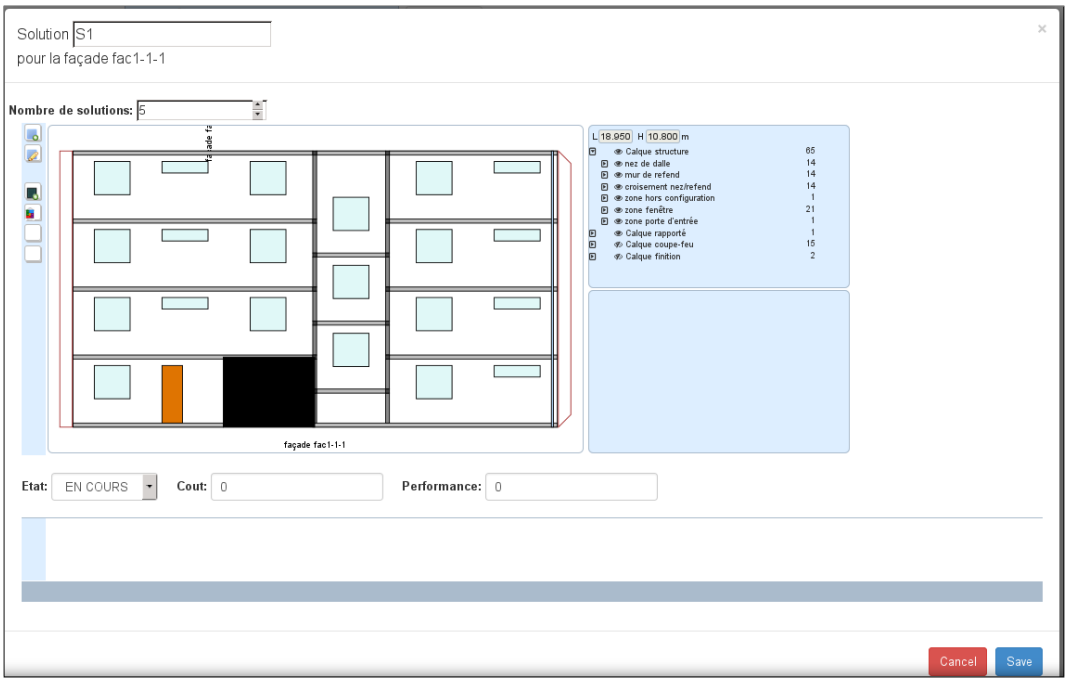


FIGURE A.5 – At pressing the design button, a two-dimensional view of the façade is shown, along with the input/output information and the design toolbar.

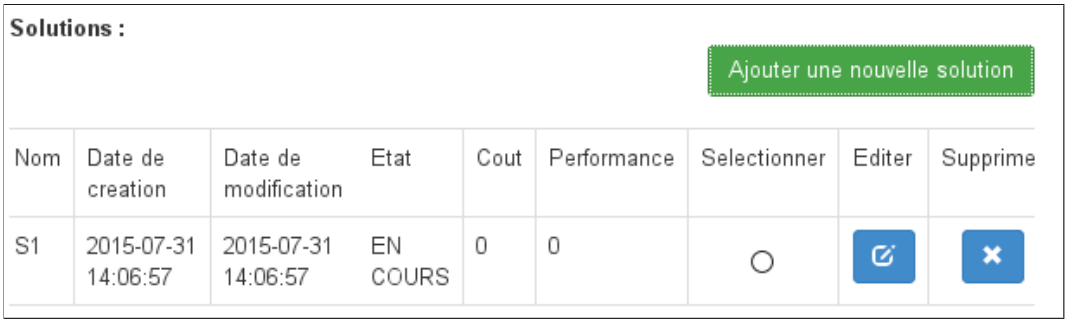


FIGURE A.6 – Adding a new solution and its respective information.

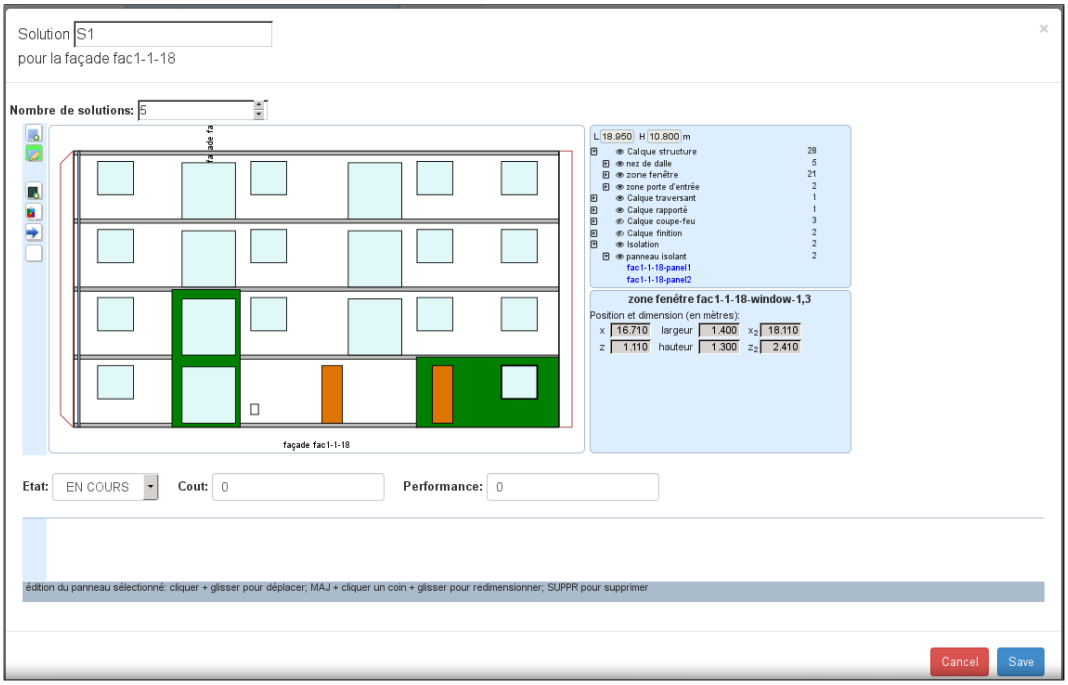


FIGURE A.7 – Over the two-dimensional view the architects draws panels. The interactive design visually informs of well and ill defined panels.

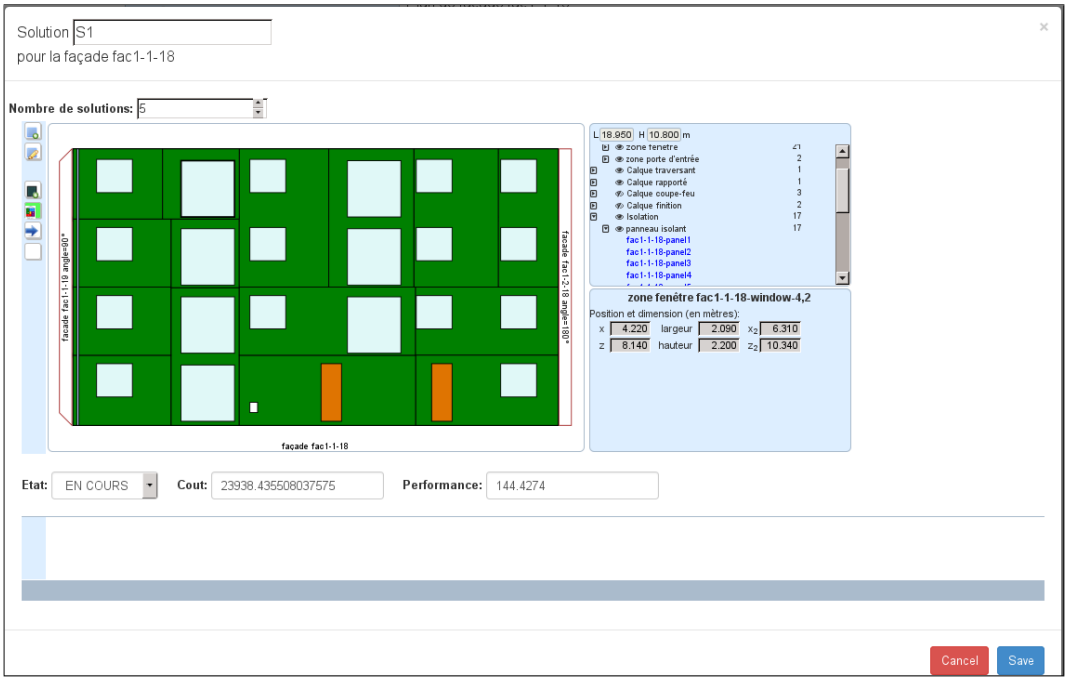


FIGURE A.8 – When asking for a solution, the system throws compliant insulating envelopes that includes the architect-defined panels. The cost and length of junctions is shown for every solution.

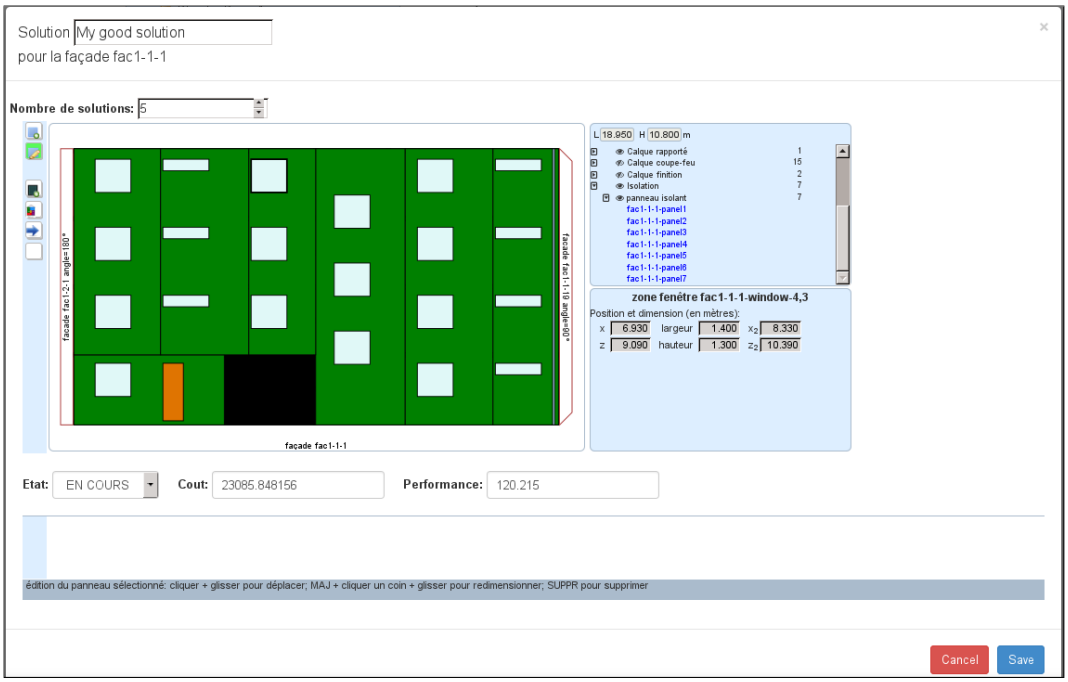


FIGURE A.9 – Fully automatic solutions are also possible.

Conception sous Contraintes : Configuration de Panneaux Isolants à Deux Dimensions

Les travaux de recherche présentés dans cette thèse se situent dans une problématique d'aide à la conception d'enveloppes isolantes pour la rénovation thermique de bâtiments résidentiels collectifs. Ces enveloppes isolantes sont composées de panneaux multifonctionels rectangulaires, configurables et préfabriqués en usine. Leur conception repose sur les cinq caractéristiques suivantes. Premièrement, le nombre de panneaux nécessaires pour concevoir une enveloppe ainsi que leur taille respective ne sont pas connus au début de la rénovation (mais leur taille est cependant bornée). Deuxièmement, en raison des contraintes de fabrication, chaque fenêtre et chaque porte présentes sur la façade à rénover doivent être insérées dans un et un seul panneau. Troisièmement, les panneaux sont fixés à des endroits spécifiques de la façade, assez résistants pour supporter leur poids, nommés zones d'accroche. Quatrièmement, ni trous (zone non couverte), ni chevauchements entre panneaux ne sont autorisés. Cinquièmement, afin de garantir une isolation thermique performante tout en minimisant son coût, les enveloppes doivent être composées d'un nombre minimal de panneaux. Aux vues de la complexité de ce problème, nous restreignons nos travaux de recherche aux façades rectangulaires portant des menuiseries et des zones d'accroche rectangulaires.

Compte tenu des cinq caractéristiques énoncées et de l'hypothèse de forme rectangulaire des éléments traités (panneaux, façades, menuiseries, zones d'accroche), la conception des enveloppes est à la fois un problème de découpe et de conditionnement à deux dimensions et un problème de configuration. Ce problème est formalisé et traité comme un problème de satisfaction de contraintes et a pour but d'aider la conception dédites enveloppes isolantes. En tant que tel, les travaux de cette thèse présentent deux contributions majeures. En raison des caractéristiques originales du problème de calepinage de façades, sa description et sa formalisation comme un problème de satisfaction de contraintes constituent la première contribution de ces travaux de thèse. Deuxièmement, les solutions algorithmiques basées sur les contraintes constituent notre seconde contribution. En particulier, ces travaux de thèse présentent deux solutions manuelles et trois automatiques pour le problème de conception d'enveloppes isolantes.

Mots-clés : Problème de satisfaction de contrainte, Configuration, Calepinage, Aide à la décision, Rénovation du bâtiment, Génie Civil

---

Constraint-Based Design : Two-Dimensional Insulating Panels Configuration

The research presented in this thesis falls within the problem of supporting the design of thermal insulating envelopes for the renovation of collective residential buildings. These insulating envelopes are composed of rectangular multi-functional panels, configurable and prefabricated in the factory. Their design is based on the following five characteristics. First, the number of panels needed to design an envelope and their size are not known at the beginning of the renovation (but their size is however bounded). Second, because of manufacturing constraints, every window and every door present on the façade to be renovated must be inserted into one and only one panel. Third, panels are attached to specific areas of the façade strong enough to support their weight, called supporting areas. Fourth, neither holes (uncovered area) or overlapping between panels are allowed. Fifth, to ensure efficient thermal insulation while minimizing cost, envelopes should be composed of a minimum number of panels. In view of the complexity of this problem, we restrict our research to rectangular façades with rectangular joinery and supporting areas.

Given the five stated characteristics and the assumption of rectangular elements (panels, façades, joinery, supporting areas), the envelopes design is both a two-dimensional `Cutting & Packing` problem as well as a configuration one. This problem is formalized and treated as a constraint satisfaction problem and aims to support the design of such insulating structures. As such, the thesis presents two major contributions. Given the original features of the building renovation problem, its description and its formalization as a constraint satisfaction problem are the first contribution of the work. Second, constraint-based algorithmic solution's are our second contribution. In particular, the thesis presents two manual and three automatic solutions for the design problem of insulating envelopes.

Keywords : Constraint satisfaction problem, Configuration, Layout synthesis, Decision support system, Building renovation, Civil engineering