Shaper-GA: Automatic Shape Generation for Modular House Design

Extended Abstract[†]

Bruno Taborda
CISUC
Instituto Universitário de Lisboa (ISCTE-IUL)
Portugal
Bruno_Taborda@iscte-iul.pt

Ana de Almeida
ISTAR-IUL & CISUC
Instituto Universitário de Lisboa (ISCTE-IUL)
Portugal
Ana.Almeida@iscte-iul.pt

Filipe Santos
ISTAR-IUL
Instituto Universitário de Lisboa (ISCTE-IUL)
Portugal
Filipe.Santos@iscte-iul.pt

Sara Eloy
ISTAR-IUL
Instituto Universitário de Lisboa (ISCTE-IUL)
Portugal
Sara.Eloy@iscte-iul.pt

Krystian Kwiecinski
Warsaw University of Technology
Poland
Krystian.Kwiecinski@pw.edu.pl

ABSTRACT

This work presents a Genetic Algorithm (GA) approach to produce automatic designs for modular houses: Shaper-GA. A set of architectural design rules defining a language of design is incorporated into the GA fitness function. When possible genetic drift or local convergence might be occurring, the method starts an adaptive mutation rate to overcome fitness stagnation. The GA tool efficiently produces several layout solutions obeying the design rules and further placement constraints. Such a tool can be integrated into an appropriate user interface allowing future house owners to customize their own house or construction companies to answer client's' requirements while complying with a language of design.

KEYWORDS
Genetic Algorithms, Automatic Layout Design, Cutting and Packing, Language of Design

1 INTRODUCTION

Mass fabrication of houses seems to have started in the mid-19th century with the establishment of colonial settlements and, since then, prefabricated houses have been successfully used [1]. However, modular construction, i.e., to build using modules, emerged with the Fuller experimentation, in last century's 20s and 30s, and the Dymaxion House, which incorporated prefabricated bathroom modules. In fact, house modules, more than individual elements (like doors or walls) but rather self-enclosed dwelling spaces, are often fabricated off-site [2]. With recent advances in mass construction and the urge of vast developing urban dwellings, mass construction has been making the headlines, like with the 461 Dean Street modular skyscraper in Brooklyn, NY, USA. However, architectural evolution resulted mainly in the development of design processes in which the end user influence on the design has been very limited. Kwiecinski and Slyk presented a formal language of design for the development of a mass customized system allowing for Polish costumers to participate in the design of their homes [3]. Two different approaches have been taken to meet the automation of this language of design and providing a good technical solution satisfying users' needs: one based in shape grammars [4] supplemented with processes and the other based in genetic algorithms.

This work introduces the latter, Shaper-GA, which is an automated shaper floor planning application, implemented via a genetic algorithm (GA) and able to generate rectangular houses obeying a predefined architectural language of design.

The remainder of the paper is organized as follows. In section 2 the problem formulation can be found. A summary of the most relevant literature review is presented on section 3. Section 4 describes the proposed genetic algorithm, Shaper-GA. Results and are discussed in section 5, where a summary of the experiments is also presented. Section 6 presents conclusions and possible directions for future work.

2 PROBLEM FORMULATION

A house layout can be viewed as a set of positions that represent the spatial relations between rooms – a floor plan. Each house has a predetermined number of rooms, total width and depth, and a central axis that equally divides the width in two, allowing for a central corridor to access all the rooms.

The rooms (smaller rectangles) must be placed such that all the architectural design rules [3] are fulfilled and without rotating any of the rectangles. An optimal solution, or proper layout, is one that obeys all the rules and further positioning constraints derived from the transposition of the design, and is free of room overlaps. Thus, the rooms should be assigned into the floor (larger enclosing rectangle) such that the positioning constraints are met. The problem to be solved may be looked at as a Two-Dimensional Single Large Object Placement Problem (2SLOPP) [5], where the overall dimensions are both fixed and there are further positioning restrictions to be obeyed.

The number of possible arrangements for the rooms' placement in the given floor is combinatorial. Almeida et al. present a small example, with 6 different rooms of fixed dimensions and a fixed entrance module [6], having 2×10^{12} possible combinations for the rooms positioning, that is, different floor plans. Naturally, the fact that some of the rooms have relative positioning design rules to fulfill restricts the total number of free possibilities. However, the combinations remain combinatorial in nature. If we add the possibility of dynamic adjustments of rooms depth, the search space dimension for a $d \times w$ rectangular house with R rooms, each with dimensions $d_i \times w_i$ and $m_{h_i} \leq d_i \leq M_{h_i}$, $(d_i, w_i, m_{h_i}, M_{h_i} \in \mathbb{N}$, $= 1,2 \ldots, R)$ is given by

$$\prod_{i=1}^{R}(w - w_i)(d - d_i)(M_{h_i} - m_{h_i}) \qquad (1)$$

In fact, if a variable dimension is allowed, the problem may be considered as a Two-Dimensional Strip Packing Problem (2D-SPP) [7][8], where the objects have fixed width and variable depth. Either way, both mathematical formulations involved in the problem of finding the optimal house layout are NP-hard combinatorial problem formulations [9][7][8], establishing Genetic Algorithms as a suitable method to tackle the layout design search. Moreover, the open possibility of emergence of design layouts from the architectural shape grammar through evolutionary strategies is especially appealing, further motivating this explorative study.

3 LITERATURE REVIEW

Cutting & Packing (C&P), general field that encompasses 2SLOPP and 2D-SPP, refers to combinatorial optimization problems with diverse real applications. Several methods and approaches have already been proposed to tackle the computational complexity usually involved [10]–[13]. Applications using evolutionary algorithms have also been proposed to address C&P problems [14]–[17]. According to Wäscher et al. [5], 2SLOPP is a particularization of a C&P problem where the main goal is to place smaller objects on larger ones, leaving as little free space as possible. 2D-SPP is a specific case of Strip Packing where a set of rectangular objects should be inserted into one container without overlaps, in such a way that the strip is minimized [9][10]. In this case, the rectangles have fixed width but variable depth. Although being a slightly more recent variant of the C&P problem, a handful of publications can also be found [7], [11]–[15]. To solve a 2-D Bin Packing Problem (2BPP) of polygonal shapes on a rectangular canvas, a genetic algorithm whose main feature is the definition of each figure based on an orthogonal axis was implemented [17]. The orthogonal axis has as parameters x, y, and θ (Euclidian coordinates and rotation angle of the figure). Also for the Bin Packing Problem (BPP), a GA method has been proposed to brand polygonal figures in a rectangular piece [18].

In terms of applications for architecture and buildings, GA strategies are recently appearing as an optimization tool of interest, due to its robustness and simplicity [19]. GAs have been employed to find the best exterior building architectural solutions and to solve complex architectural problems [20]. A GA approach aiming at the generation of automatic designs for modular houses production, G-Shaper [6], was the first attempt to find an optimal house layout incorporating a language of design. This version considered both rooms' dimensions as previously fixed.

4 METHODOLOGY

4.1 Classic genetic algorithm

Over 30 years ago, John Holland proposed Genetic Algorithms (GA) as a paradigmatic method to tackle computationally complex search spaces [21]. The classic approach for genetic algorithms [22] can be described by Algorithm 1.

Algorithm 1: Classic genetic algorithm

```
1. Randomly generate an initial population
2. Select $N$ best fitted individuals
3. While stop criterion not met:
4.     Select parents for reproduction
5.     Crossover
6.     Mutation
7.     Select $N$ individuals for a new generation
```

Defining the right quantity of individuals (*N*) for the evolutionary population is an important GA parameter. If there is a huge number of individuals in the population, the generational chromosomic diversity tends to be large and consequently the exploration of the search space is higher. The lack of diversity may hinder a broader exploration and the algorithm might easily get stuck in a local solution.

4.2 Shaper-GA

This section describes an evolution of the previously referred G-Shaper algorithm [6], Shaper-GA. In the previous version, only one final optimal solution based on the rules of Kwiecinski and Slyk [3] is found and the rooms' side dimensions, width and depth, are both fixed. Shaper-GA primal difference is that rooms' depths are not fixed: the algorithm may adjust depths within a pre-defined range. Another difference is that several proper house layouts should be found before the algorithm terminates its evolution. Finally, when possible genetic drift or local convergence might be occurring, the method starts an adaptive mutation rate to overcome fitness stagnation. The next subsections describe the operators used in Shaper-GA in more detail.

4.2.1 Chromosome and Gene representation. The encoding for a chromosome representing a house layout is an array X of R genes (where R is the predefined number of rooms). Each gene represents a room X_i, whose position in the layout can be represented by assigning a 1 to each of $d_i \times w_i$ sequential cells in a $d \times w$ rectangular binary grid representing the house, where d (depth) and w (width) are the ceiling or integer values for the given house side dimensions.

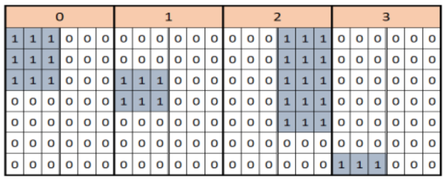

Figure 1 : Multiple gene representation visualization

The encoding of one room is described by:

Table 1: Room (gene) encoding

Parameter	Description
Name	Name of room i (e.g.: living room)
x	x=1 or x=$w/2+1$
y	$y \in [1, d - d_i]$
Width	w_i (fixed)
Depth	$d_i \in [m_i, M_i]$ (predefined range)

In Table 1, x and y are the grid cell coordinates (row and column) of the up-left vertex of a rectangular shape representing the positioning of the room in the grid. Since the house divisions must be placed on either one of the sides of a central axis that divides the weight of the layout into two equal subrectangles, the vertical is either 1 or w/2+1.

4.2.2 *Fitness.* Embodiment of the rules of design. The architectural rules to be obeyed by the optimal house layout chromosome are the following:

1. The vestibule must be placed next to the garage;
2. The toilet and technical room must be placed next to the garage;
3. The kitchen must be placed either at the front of the house or next to the living room;
4. The dining room must be placed at front or next to the kitchen;
5. The living room must be placed next to the kitchen or dining room;
6. The single bedroom must be placed next to another single bedroom or a bathroom;
7. The bathroom is placed next to a single bedroom;
8. The double bedroom must be placed at the back.

The rooms are to be positioned into a compact rectangular floor plan according to the rules, having only the garage sticking out of the rectangular area at a predetermined position [4] (Figure 2). The fitness function penalizes non-compliance with the rules in the chromosome's layout representation. Harder penalties are assigned for overlaps found between rooms. While non-compliance implies a penalization of 100 per rule, overlaps penalization is $100 \times n_o$, where n_o is the number of cells that overlay. Considering $F(i)$ as the fitness of individual i and $P(i)$ as the sum of all penalties for i, F is calculated by Equation 2:

$$F(i) = \begin{cases} 1, & P(i) = 0 \\ -\frac{1}{P(i)}, & P(i) \neq 0 \end{cases} \qquad (2)$$

The higher the fitness the most adapted the house is. A fitness value of 1 implies that the house represented by the chromosome is an optimal solution (obeys all the rules and there are no overlaps).

4.2.3 *Population.* An initial population is generated with M different individuals, each representing a house layout. Note that unfeasible layouts, that is, layouts with overlapping room's positions may be generated. The rationale behind is that a non-expected layout might emerge, defining an alternative placement of the rooms not foreseen by the architect but still satisfying the design rules. The individuals are evaluated and sorted in decreasing order of fitness value. The $M \times 0{,}1$ individuals with higher fitness values are selected for the first evolutionary population (that will evolve until the stop criterion is met).

4.2.4 *Selection.* There are two different selection moments: parent selection and next generation selection. The selection of the individuals (parents) that will be allowed to reproduce generating offspring was implemented using the Roulette Wheel Selection (RWS). This method is the most efficient operator for recombination within this domain of application as shown in the extensive study developed for the previous version [6]. The probability of an individual i being selected for crossover in a population with N individuals having $F(i)$ as the fitness function is described in Equation 3:

$$P_{select}(i) = F(i) / \sum_{j=0}^{N} F(j), i \in \{1,2,..,N\} \qquad (3)$$

In the case of the selection of individuals[1] that will survive for the next generation, three different methods are tested:

- Elitism: half of the most adapted offspring joins with half of the most adapted individuals from the actual evolutionary population (parents) to create the new evolutionary population.
- Ranking: selection of the N most fitted individuals between the current evolutionary population and its offspring.

[1] Note that the size of the evolutionary population is fixed (Section 4.2.3).

- Elitism plus ranking: the best 10% of the current population transfer to the new evolutionary population, which is completed by the most fitted individuals between the remaining parental population and offspring.

The reason not to choose only one selection mechanism comes from the fact the test results have not distinguished any operator (Section 5).

4.2.5 *Crossover and Mutation*. The crossover method used in Shaper-GA is Random Respectful Crossover (RRC) since it has shown the best results for the previous G-Shaper version [6]. If both parents share the same gene (i.e., the same room's position), it will be copied for the offspring, otherwise a new gene (new positioning) is randomly created.

The mutation operator may modify one room for a new offspring chromosome. The probability for a mutation to occur is of 2%. This operator was implemented in Shaper-GA using the two different approaches presented in Section 4.3 that tackle the issue of depth dimension variability, and, in fact, are a first strategy to enable the possible emergence of solutions.

4.3 Variability and Emergence of solutions

Shaper-GA as a house layout automatic generator has the following objectives: produce several proper (optimal) layout solutions and optimize performance. Within the several solutions, the architectural main challenge is that of visualizing layouts that are somehow unexpected.

4.3.1 Optimization. The evolution of a GA may slower down due to genetic drift. Trying to overcome this issue, an adjustable mutation rate is explored. Considering G as the current population, if the average generational fitness doesn't increase at least 0.5% between generation $G-1$ and G, the probability of mutation duplicates. An upper limit of 20% was defined for the maximal probability of mutation. Once the average fitness unlocks the situation above, the mutation probability goes back to 2%.

4.3.2 Production of several solutions. Shaper-GA aims to provide different optimal solutions, i.e. different room arrangements obeying all the positioning constraints. Even when the internal structure of the house remains the same, rooms vary its depth, making the arrangements define different houses (Erro! A origem da referência não foi encontrada.2).

The question of parametric depth was implemented using a resize function (Algorithm 2) that allows for a mutation in a gene to affect only the depth of the room. When a resize mutation is decided, the room may either increase or decrease its depth. Otherwise, a completely new random positioning is generated to fulfill the mutation decision.

Resizing can also occur in the crossover operator. Since we are using RRC recombination, when parents have different genes (positions) for a given chromosome index, the new room for the offspring has equal probability of being assigned to a new random position or keeping the position and being resized.

Algorithm 2 - Resize function

```
1. Randomly generate r ∈ [0, 1];
2. If r < 0,5
3.    If $d_i > m_i$ then $d_i \leftarrow d_i - 1$;
4. Else
5.    If $d_i < M_i$ then $d_i \leftarrow d_i + 1$;
```

Shaper-GA implements and tests two different GA versions: Standard and Resizable from Beginning (RfB) version. While in the Standard version, for the initial population, each house is generated with all the rooms having a given (standard) depth (Table 3), RfB version generates each random room with depth in the allowed range for each type of division (Table 2). Thus, in RfB the initial population has individuals with the same type of rooms but with different depths. More explicitly, each gene X_i will be generated with a (random) depth $d_i \in \{m_i, M_i\}$ for given m_i and M_i.

In either of the versions, when a chromosome representing a proper house is selected for gene mutation, it can only suffer a resize. In case the respective layout is not optimal, a room mutation has equal probability of generating a new position for the room or resize it (Algorithm 3).

Algorithm 3 - Mutation operator

```
1. Randomly generate a value $c \in [0, 1]$;
2. If $0.001 \le c \le P$ ($P \in [0.02, 0.2]$)
3.    Pick a random room, $d$;
4.    If fitness of $d$ = 1.0
5.       Resize $d$;
6.    Else
7.       Randomly generate $r \in [0, 1]$;
8.       If $r < 0,5$
9.          Generate new position for $d$;
10.      Else:
11.         Resize $d$;
```

5 Results and discussion

Both Shaper-GA versions – Standard and RfB - were implemented using Java.

The modular layout design rules for a house are the ones previously referred to (subsection 4.2.2) and described on previous works [6][4]: a detached one-store house, with a garage sticking out of the rectangular perimeter. The house's width and depth are 14×17 grid units (usually a grid unit stands for one square meter ($1m \times 1m$) area). There is also a central axis dividing the floor plan into identical left and right sides. This imposition allows for the insertion of a central communication area – a corridor.

Table 2: Depth (in grid units) for each room

Room i	Minimum depth: m_i	Maximum depth: M_i
Kitchen	2	8
Living room	4	9
Dining room	4	6
Double bedroom	5	7
Single bedroom	4	7
Bathroom	3	5

Table 3: Standard depth (in grid units) for each room

Room	Standard depth
Kitchen	2
Living room	5
Dining room	4
Double bedroom	6
Single bedroom	4
Bathroom	3

Four of the rooms (vestibule, toilet, garage, and tech room) positions are previously fixed by the design and, thus, only the remaining 6 rooms are to be positioned (bathroom, kitchen, living room, double bedroom, 2 single bedrooms and dining room). The rooms' side dimensions are measured in grid units (Table 2 and Table 3). Although the final rooms' width might have to be adjusted at the end of the evolution to insert the corridor, during the evolution the width is fixed to 7 grid units, i.e. w/2. This implies that each one of the rooms can be placed either at the left side (the correspondent left column coordinate is $x = 1$) of the axis, or at the right side ($x =$ w/2+1).

Shaper-GA evolution for this example stops when a minimum of five different optimal solutions are found. Two houses are different if there is, at least, one room that differs between the two houses. As an example, the solutions shown in Figure 2 have either different depths or arrangements. In the layout at the top-left, the living room is 4 × 7 grid units. On the solution at the top-right, the living room area is larger (5 × 7). In contrast, this kitchen's area is smaller (in fact, it stands out as a kitchenette). Different layout arrangements may also be found, like the one at the bottom of Figure 2.

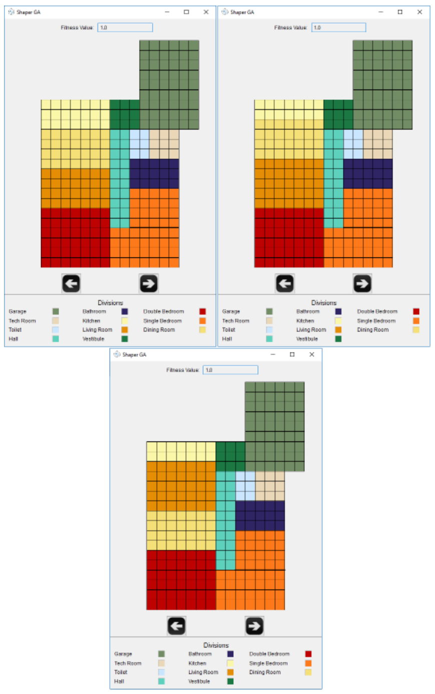

Figure 2: Three different solutions from Shaper-GA

The following figures (Figure 3 and Figure 4) show averaged results of 30 runs with 1 million generations each and 100 individuals in the evolutionary population. While for the standard version the selective schemas performance is rather similar, the same cannot be said for the RfB version, where the Elitism & Ranking schema clearly outperforms the other two. In general, Elitism & Ranking produces the best average results for both versions. Interestingly, comparing the average fitness values in the population for both versions it is visible that, for the same amount of generations, the Standard version reaches higher averaged values, 0.203 (approx.) against 0.139 (approx.) with the RfB version, showing the latter to evolve rather slower than the former.

6 Conclusions

Shaper-GA employs a classic genetic algorithm approach that returns at least five different house layouts compliant with the language of design proposed by Kwiecinski et al. [4] in the form of a shape grammar.

Related to a previous and less dynamical version, Shaper-GA encloses several differences. A new encoding for the gene (room) has been used allowing for a more efficient search in terms of running time. To prevent larger generation drift, a new mechanism for spatial search space exploration increase was introduced, that adjusts the mutation probability in function of the average of the evolutionary population fitness values variation. The crossover operator, RCC, also has been modified to allow for a recombination that incorporates the possibility of a random gene (room) depth resize, instead of using only random gene generation. Finally, this is the first GA approach for this specific problem that deals with variable room side dimensions and can return several different solutions.

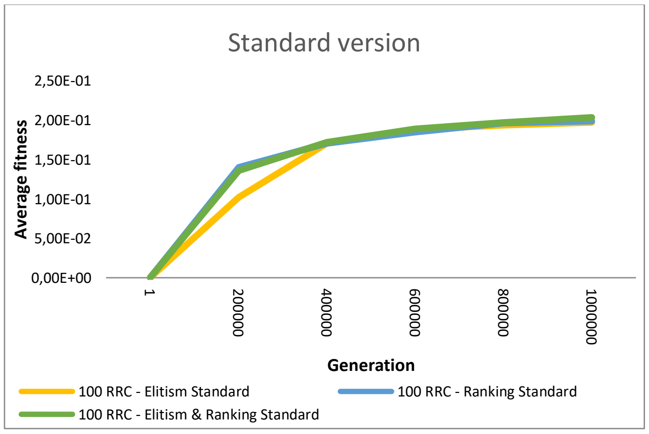

Figure 3: Average fitness values for the Standard version

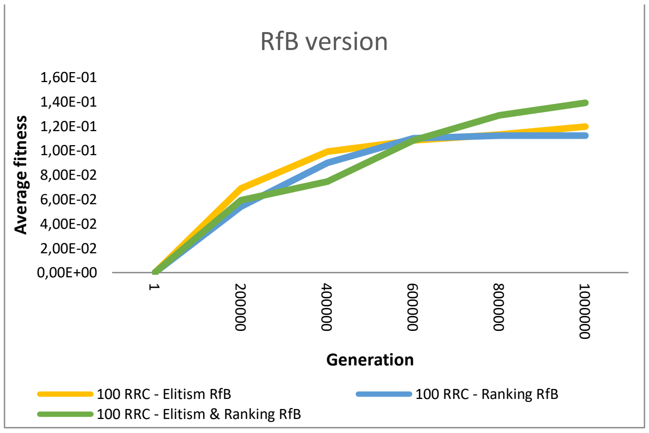

Figure 4: Average fitness values for the RfB version

A deeper exploration of this method is still needed to produce an operational software prototype to be tested by real users: a) to reduce the number of generations required to obtain several different solutions; b) to work with more complex layouts. Namely, the mutation operator can be further adjusted so that, when the average population fitness starts to smooth its increase, it favors specially fitted alterations that effectively reduce the penalizations, thus increasing fitness values.

Another interesting challenge is that of evolving communication elements like doors and windows, which is crucial to produce a fully automated design system. Such a tool can be integrated into an appropriate user interface allowing future owners to customize their own house or construction companies to answer client's' requirements while complying with a language of design.

As a final note, this study in object placement constrained by relative positioning rules could be used for the development of an evolutionary approach for chip layout design.

REFERENCES

[1] A. Vogler, M. Eekhout, and IOS Press., *The house as a product*. .
[2] K. Mrkonjic, "Environmental Aspects of Use of Aluminium for Prefabricated Lightweight Houses: Dymaxion House Case Study," *J. Green Build.*, vol. 2, no. 4, pp. 130–136, Nov. 2007.
[3] K. Kwiecinski and J. Slyk, "System for customer participation in the design process of mass-customized houses," *Fusion Data Integr. its Best, Vol 2*, vol. 2, pp. 207–215, 2014.
[4] K. Kwiecinski, F. Santos, A. De Almeida, B. Taborda, and S. Eloy, "Wood Mass-Customized Housing A dual computer implementation design strategy," *Complex. Simplicity - Proc. 34th eCAADe Conf.*, vol. 2, pp. 349–358, 2016.
[5] H. S. Gerhard Wäscher, Heike Haußner, "An improved typology of cutting and packing problems," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1109–1130, 2007.
[6] A. De Almeida, B. Taborda, F. Santos, K. Kwiecinski, and S. Eloy, "A genetic algorithm application for automatic layout design of modular residential homes," *Proc. 2016 IEEE Int. Conf. Syst. Man Cybern.*, pp. 2774–2778, 2016.
[7] A. Bortfeldt, "A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces," *Eur. J. Oper. Res.*, vol. 172, no. 3, pp. 814–837, 2006.
[8] J. Thomas, "Advances in Computational Intelligence," vol. 7902, no. November, 2013.
[9] S. M. Maxence Delorme, Manuel Iori, "Bin packing and cutting stock problems: Mathematical models and exact algorithms," *Eur. J. Oper. Res.*, vol. 255, no. 1, pp. 1–20, 2016.
[10] A. Lodi, S. Martello, and M. Monaci, "Two-dimensional packing problems: A survey," *Eur. J. Oper. Res.*, vol. 141, no. 2, pp. 241–252, 2002.
[11] $ María, C. Riff, X. Bonnaire, and B. Neveu, "A revision of recent approaches for two-dimensional strip-packing problems."
[12] J. F. Oliveira *et al.*, "A SURVEY ON HEURISTICS FOR THE TWO-DIMENSIONAL RECTANGULAR STRIP PACKING PROBLEM," *Pesqui. Operacional*, vol. 36, no. 2, pp. 197–226, Aug. 2016.
[13] E. Hopper and B. C. H. Turton, "A Review of the Application of Meta-Heuristic Algorithms to 2D Strip Packing Problems," *Artif. Intell. Rev.*, vol. 16, no. 4, pp. 257–300, 2001.
[14] C. Salto, G. Leguizamón, and E. Alba, "Parallel ACO algorithms for 2D Strip Packing."
[15] V. Southern African Institute for Industrial Engineering, T. I. SPARC (Organization), and T. Hua, *A genetic algorithm for two dimensional strip packing problems*, vol. 20, no. 2. The Southern African Institute for Industrial Engineering, 2009.
[16] H. Gharsellaoui and H. Hasni, "An hybrid genetic algorithm for two-dimensional cutting problems using guillotine cuts AND LITERATURE RE-," no. February 2014, 2012.
[17] V. Ayala-ramirez, A. Ponce-Pérez, A. Perez-Garcia, A. Pérez-Garcia, A. Ponce-p, and P. Arturo, "Bin-Packing Using Genetic Algorithms," *Electron. Commun. Comput. Int. Conf.*, vol. 0, no. Conielecomp, pp. 311–314, 2005.
[18] S. Jakobs, "On genetic algorithms for the packing of polygons," *Eur. J. Oper. Res.*, vol. 88, no. 1, pp. 165–181, 1996.
[19] A. D. S. Curriculum, "High-rise Building Optimization," vol. 1, pp. 305–314, 2012.
[20] L. Li, "The optimization of architectural shape based on Genetic Algorithm," *Front. Archit. Res.*, vol. 1, no. 4, pp. 392–399, 2012.
[21] J.H. Holland, *Hidden Order: How Adaptation Builds Complexity*. 1995.
[22] H.-G. Beyer, H.-G. Beyer, H.-P. Schwefel, and H.-P. Schwefel, "Evolution strategies – A comprehensive introduction," *Nat. Comput.*, vol. 1, no. 1, pp. 3–52, 2002.