AI-Driven Generative Design: Evolutionary Optimization of Residential Floor Plans

Zuxing Wu
a1816653

Supervisor: Adel Nikfarjam

Master of Computer Science

November 4, 2024

Contents

1 Introduction

1.1 Aim

Residential floor plan design is a complex and challenging task that requires careful consideration of various factors, such as room sizes, adjacencies, privacy, convenience, and orientations. Traditional methods of floor plan design are often time-consuming and labor-intensive, leading to suboptimal solutions. Evolutionary algorithms have been proposed as a promising alternative for optimizing floor plans, as they can efficiently explore the design space and generate high-quality solutions. This project aims to develop a new method for optimizing residential floor plans using evolutionary algorithms and address some limitations of previous research. The proposed method will involve the application of evolutionary algorithms to generate and evolve floor plans based on a novel representation scheme. The fitness of each floor plan will be evaluated based on several criteria (e.g. privacy, comfort, practicality, convenience). The performance of the proposed method will be evaluated using a set of benchmark problems and compared with existing approaches. The results of this project will contribute to the field of residential floor plan design and provide valuable insights into the application of evolutionary algorithms to architectural design problems.

1.2 Motivation

Residential floor plan design is a critical aspect of architectural design that involves the layout of rooms, corridors, and other spaces within a building. The design of a floor plan can have a significant impact on the functionality, convenience, comfort, ventilation and energy efficiency of a building. Traditional methods of floor plan design are often based on manual sketches or computer-aided design (CAD) tools, which can be time-consuming and labor-intensive. Moreover, these methods may not always produce optimal solutions, as they rely on the intuition and experience of the designer.

Evolutionary algorithms have been proposed as a promising alternative for optimizing floor plans, as they can efficiently explore the design space and generate high-quality solutions. Evolutionary algorithms are a class of optimization algorithms inspired by the process of natural selection. They work by maintaining a population of candidate solutions (individuals) and iteratively applying genetic operators (e.g. crossover, mutation) to generate new solutions. The fitness of each solution is evaluated based on a predefined fitness functions, which measure how well the solution satisfies the objectives of the optimization problems.

2 Literature Review

Previous research has explored the application of evolutionary algorithms to optimize residential floor plans. Brintrup et al. [1] compared three interactive

genetic algorithms (i.e., sequential IGA, multi-objective IGA, parallel IGA) on a multi-objective floor planning task, and found that the multi-objective IGA provides more diverse results and faster convergence for optimizing floor plans. They developed interactive evolutionary algorithms that allow designers to incorporate their preferences and constraints into the optimization process. This method has shown promising results in generating floor plans that meet both functional and aesthetic requirements.

It was found that proportional roulette wheel selection is the best parent selection method for the mating pool, and k-point crossover is the most effective for fitness evolutionary improvement [3]. Combining evolutionary algorithms with greedy-like algorithms can help find near-optimal solutions in Automated Floor Plan Generation (AFPG), though it is a simplified model of multi-objective optimization by linear composition of the partial evaluation functions [4]. Subramanian et al. [5] used a genetic algorithm with KD tree models in a web application to generate floor plans for even non-expert users.

Wang and Duan [7] tried to optimize floor plan design by focusing on energy consumption and consumer satisfaction, proving that the preferences of different types of consumers differed significantly. Therefore, different evaluation criteria are needed for satisfying different family types [7]. The quality and efficiency of residential floor plan design can be improved by combining Monte Carlo tree search algorithm (MCTS) and particle swarm optimization (PSO) [8]. The MCTS algorithm takes human experience into consideration so that it can compress the search space and improve the efficiency of the search process [8]. The PSO algorithm can handle continuous variables and is suitable for optimizing the size of rooms due to parallel processing [8].

Enengy consumption, probable uniformity (PU), and spatial useful daylight illuminance (sUDI) are three objectives that are considered in the optimization process using NSGA-II algorithm, and the results show that the NSGA-II algorithm can provide a set of more sustainable floor plans that requires less computational power and time [2]. Reliable metrics for evaluating the amount of light and the uniformity of light are tested in the optimization process, and can avoid unwanted convergence because they restrict each other [2].

Furthermore, the integration of machine learning techniques with evolutionary algorithms has gained attention in recent decades. For example, Wang et al. [6] proposed a hybrid approach that combines a genetic algorithm with a neural network to predict the fitness of candidate solutions. This method significantly reduces the computational cost of evaluating large populations and accelerates the optimization process.

Overall, the literature indicates that evolutionary algorithms, particularly when combined with other optimization techniques and machine learning methods, offer a powerful tool for optimizing residential floor plans. These approaches not only improve the efficiency and quality of the designs but also provide flexibility in accommodating various design preferences and constraints.

3 Methodology

The methodology of this project involves the application of evolutionary algorithms to optimize residential floor plans. The process will be divided into several key stages: initialization of the population, representation of the floor plan, design of the evolutionary optimization framework, and performance evaluation. These stages are outlined below.

3.1 Population Initialization

The first step in the evolutionary process is to generate an initial population of residential floor plans. An efficient initialization strategy will be employed to ensure diversity within the population while adhering to basic architectural constraints, such as room adjacency and size. Specifically, the initialization process will involve generating a set of random rooms and arranging them into a floor plan layout. The rooms will be assigned random sizes and positions within the layout, with constraints on room adjacencies and widths. The initialization strategy will be designed to produce a diverse set of floor plans that can serve as a starting point for the evolutionary optimization process.

3.2 Representation of Floor Plans

Residential floor plans will be represented using a Python class, that encodes the spatial information of rooms, such as adjacency and dimension. This representation will be designed to allow for easy manipulation by evolutionary operators (e.g. crossover or mutation) while preserving architectural feasibility and obeying relevant constraints. Most importantly, the representation will be flexible enough to accommodate different types of rooms, such as bedrooms, living rooms, kitchens, and bathrooms. The representation will also include the presence of doors and windows, and other architectural features. This representation will be used throughout the optimization process to encode, decode, and evaluate floor plans.

3.3 Evolutionary Optimization Framework

The evolutionary optimization framework currently will be based on particle swarm optmization (PSO) algorithm, which is suitable for optimizing the size of rooms due to parallel processing [8]. The PSO algorithm will be used to evolve the population of floor plans over multiple generations. The framework will involve the application of genetic operators, such as crossover and mutation, to generate new floor plans from the existing population. The fitness of each floor plan will be evaluated based on 4 types of criteria (e.g. privacy, comfort, practicality, convenience). The optimization process will be repeated for a specified number of generations or until a termination criterion is met.

If possible, more than one evolutionary algorithms will be used in the process of optimizing the floor plan. For example, Monte Carlo tree search algorithm

(MCTS) handles discrete variables (e.g. room position) and particle swarm optmization (PSO) deals with continuous variables [8].

3.4 Fitness Function Design

The fitness function will evaluate each floor plan based on several criteria: privacy, comfort, practicality, convenience. These criteria can be evaluated by a group of human evaluators, shown as Table 1.

Evaluation indicators of floor plans of MURBs.

First-level indicator	Second-level indicator	Type	Reference
C1 Privacy	C11 dis (MBR, BR)	Positive	Gao et al. (2013)
	C12 dis (MBR, BA)	Positive	Kong (2013)
C2 Comfort	C21 Orientation of living room (0: north; 1: south)	Positive	Gao et al. (2013)
	C22 Light in dining room (0: dark; 1: bright)	Positive	Gao et al. (2013)
	C23 Ventilation (Ratio of width to depth of floor plan)	Positive	Gao et al. (2013)
	C24% of south-facing rooms	Positive	Gao et al. (2013)
C3 Practicality	C31% of hall	Negative	Kong (2013)
	C32% of balcony	Positive	Kong (2013)
	C33 Efficiency rate of a house	Positive	He (2020)
C4 Convenience	C41 DIS (BR, BA)	Negative	Gao et al. (2013)
	C42 DIS (BA, BAL)	Negative	Gao et al. (2013)

Note: "dis (m, n)" means the straight distance between the geometric centers of room m and n.
"% of R" means the percentage of area of R to the interior area of the floor plan.
"Efficiency rate of a house" means the percentage of interior area to the total area of the floor plan.
"DIS (m, n)" means the walking distance along 0- or 90-degree lines between the geometric centers of room m and n.

Table 1: Evaluation indicators. Table 2 of Wang and Duan [7]

The straight distance between the geometric centers of room m and n is defined as:

$$\text{dis}(m, n) = \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2}$$

The percentage of area of R to the interior area of the floor plan is defined as:

$$\%\text{ of } R = \left(\frac{\text{Area of } R}{\text{Interior Area of Floor Plan}} \right) \times 100\%$$

The efficiency rate of a house is defined as the percentage of interior area to

the total area of the floor plan:

$$\text{Efficiency Rate} = \left(\frac{\text{Interior Area}}{\text{Total Area}} \right) \times 100\%$$

The walking distance along 0- or 90-degree lines between the geometric centers of room m and n is defined as:

$$\text{DIS}_{\text{door}}(m, n) = \text{DIS}(m, \text{door}_m) + \text{DIS}(\text{door}_m, \text{door}_n) + \text{DIS}(\text{door}_n, n)$$

The proposed method will be implemented in Python. The optimization process will be carried out in two stages: initialization and mutation. The initialization stage will involve generating an initial population of floor plans using a novel strategy. The mutation stage will consist of iteratively applying genetic operators to the population to produce new generations. The fitness of each individual will be evaluated using multiple fitness functions that considers various factors such as the position of rooms, room sizes, and room adjacencies. The optimization process will be repeated for a specified number of generations or until a termination condition is met. The performance of the proposed method will be evaluated using a set of benchmark problems and compared with existing approaches.

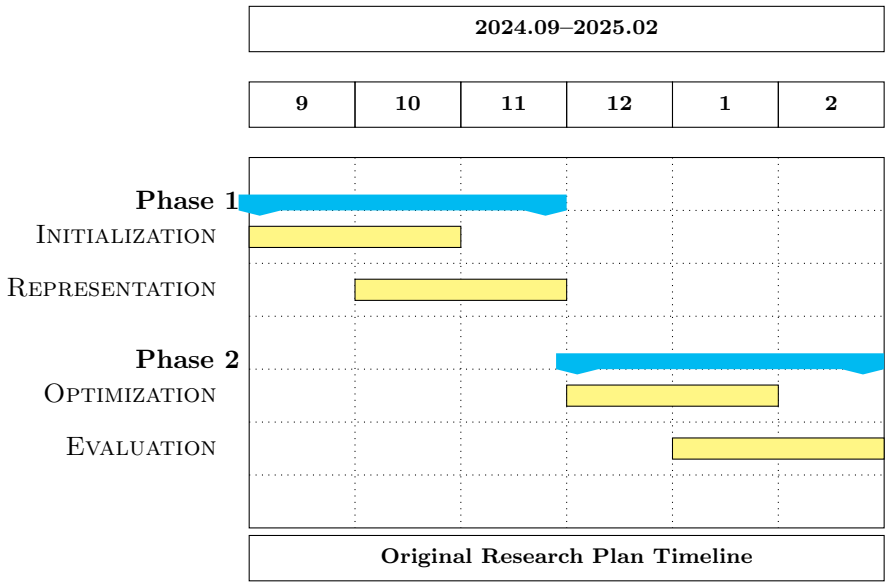
4 Plan and Progress

4.1 Research Plan

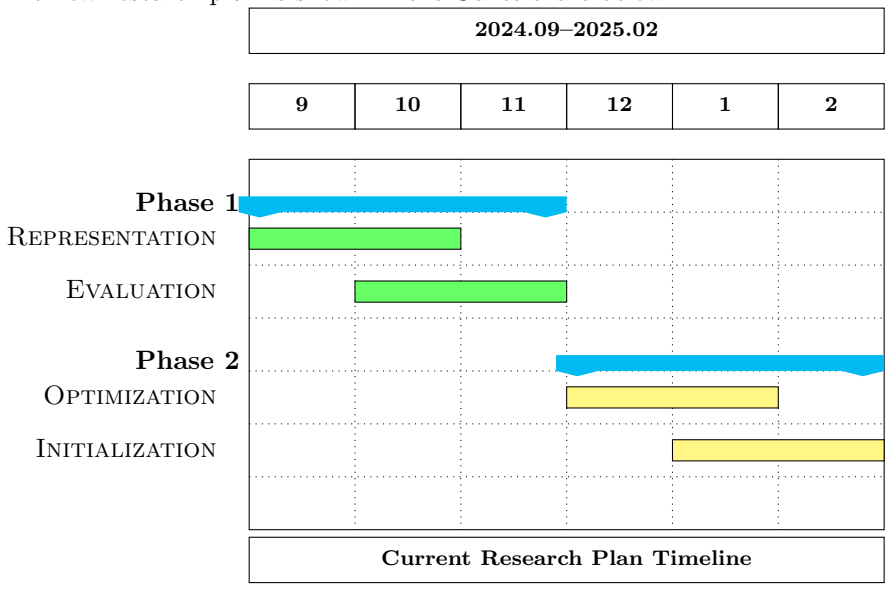
My research plan consists of four main phases at the beginning:

- Phase 1
    1. Population initialization
    2. Solution representation
- Phase 2
    1. Evolutionary optimization
    2. Solution evaluation

Each phase involves specific tasks and activities that will be carried out over a period of 6 months. The timeline for the research plan is shown in the Gantt chart below.



However, the research plan has been adjusted according to the progress of the research. My supervisor and I both agree that the research plan should be adjusted to focus on the solution representation and solution evaluation first, and then move on to the evolutionary optimization and population initialization. The new research plan is shown in the Gantt chart below.



4.2 Progress

It can be seen from the Gantt chart above that the research plan has been adjusted to focus on the solution representation and solution evaluation first, and then move on to the evolutionary optimization and population initialization. It makes sense to focus on the solution representation and solution evaluation first, as solution representation is the prerequisite of solution evaluation, and the solution evaluation is the key to the success of the research. The evolutionary optimization and population initialization can be adjusted according to the progress of the research.

From the start of the project, I have completed the following tasks: 1. Solution representation, 2. Most of the solution evaluation functions. The solution representation is based on the Python class, which encodes the spatial information of rooms, such as vertexes, doors, windows, and adjacent rooms. The solution evaluation is based on the evaluation indicators shown in Table 1. The fitness function is designed to assess each floor plan based on several criteria: privacy, comfort, practicality, and convenience. It includes 11 evaluation indicators from Wang et al. [7] research and 1 self-made indicator that are used to calculate the fitness of each floor plan.

All fitness values calculated from these indicators are normalized to the range of [0, 1] and then combined to form the final fitness value. The Min-Max normalization method is used to normalize the fitness values. The Min-Max normalization equation is defined as:

$$X_{\text{normalized}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

where X is the original value, $X_{\min}$ and $X_{\max}$ are the minimum and maximum values of the evaluation indicator, respectively.

5 Experimental Results

I have not conducted any experiments yet. The experiments will be conducted after the completion of the evolutionary optimization and population initialization. For now, I can only show the results of the solution representation and solution evaluation. A floor plan solution is encoded by different classes based on the element level, such as Vertex, Door, Window, Room, and House.

5.1 Vertex

A vertex is defined by its x and y coordinates. The class Vertex is shown in Listing 1 (all methods body are omitted).

```
1 class Vertex:
2     def __init__(self, x, y):
3         self.x = x
4         self.y = y
```

Listing 1: Vertex class

5.2 Door

A door is defined by its name, two vertices, length. The class Door is shown in Listing 2 (all methods body are omitted).

```
class Door:
  def __init__(self, name, vertex1, vertex2, length=2.0):
      self.name = name
      # door's position
      self.vertex1 = vertex1
      self.vertex2 = vertex2
      # door's length
      self.length = max(length, 2.0)
      # door's width
      self.width = ((vertex1.x-vertex2.x)**2 + (vertex1.y-
          vertex2.y)**2)**0.5
       # Adjust vertices if width is less than min_width 0.9
      if self.width < 0.9:
          self.adjust_vertices(0.9)
```

Listing 2: Door class

5.3 Window

A window is defined by its name, two vertices, length, height. The class Window is shown in Listing 3 (all methods body are omitted).

```
class Window:
  def __init__(self, name, vertex1, vertex2, length=0.4,
      height=0.0):
      self.name = name
      # window's position
      self.vertex1 = vertex1
      self.vertex2 = vertex2
      # height is the distance from floor to the bottom of
          the window
      self.height = height
      # window's width
      self.width = ((vertex1.x-vertex2.x)**2 + (vertex1.y-
          vertex2.y)**2)**0.5
      # window's length
      self.length = length if length > 0.4 else 0.4

      # Adjust vertices if width is less than min_width 0.4
      if self.width < 0.4:
```

```
            self.adjust_vertices(0.4)
```

Listing 3: Window class

5.4 Room

A room is defined by its name, type, vertices, doors, windows. The class Room is shown in Listing 4 (all methods body are omitted).

```
class Room:
  def __init__(self, name, type, vertices=[], doors=[],
      windows=[]):
      self.name = name
      # room's vertices, clockwise order
      self.vertices = vertices
      # room's walls
      self.walls = []
      # room's doors
      self.doors = doors
      # room's windows
      self.windows = windows
      # room's type
      self.type = type
      # room's area
      self.area = self.get_area()
      # room's centroid
      self.centroid = self.get_centroid()
      # ajacent rooms
      self.adjacent_rooms = []
      # number of doors
      self.num_doors = len(self.doors)
      # minimum width/length of the room
      self.min_width = 0.9
```

Listing 4: Room class

5.5 House

A house is defined by its rooms list. It can also calculate its adjacent matrix, adjacent list, area, interior area, width, depth. The class House is shown in Listing 5 (all methods body are omitted).

```
class House:
  def __init__(self, rooms=[]):
      self.rooms = rooms
      self.num_rooms = len(self.rooms)
      self.adjacent_matrix = self.get_adjacent_matrix()
      self.adjacent_list = self.get_adjacent_list()
      self.area = self.get_area()
```

```
        self.interior_area = self.get_interior_area()
        # width of the house
        self.width = self.get_width()
        # depth of the house
        self.depth = self.get_depth()
```

Listing 5: House class

5.6 Floor Plan Visualizations

A function is implemented with the help of the matplotlib library to visualize the floor plan. The function is shown in Listing 6.

```
import matplotlib.pyplot as plt

def draw_wall(vertices):
    x, y = [vertex.x for vertex in vertices], [vertex.y for
        vertex in vertices]
    x = list(x) + [x[0]]
    y = list(y) + [y[0]]
    plt.plot(x, y)

# Function to draw all rooms of the house
def draw_house(house):
    for room in house.rooms:
        draw_wall(room.vertices)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.gca().spines['right'].set_visible(False)
    plt.gca().spines['top'].set_visible(False)
    plt.show()
```

Listing 6: Visualize floor plan

A simple example of a floor plan is shown in Figure 1.

5.7 Feasibility Check

There are many necessary feasibility checks that need to be done before the project can be completed. The following are some of the most important ones: window feasibility check, door feasibility check, room feasibility check. Take room feasibility check as an example, 2 random rooms should not overlap. The room's overlap check function is shown in Listing 7.

```
  def overlap_check(self):
        for i in range(len(self.rooms)):
            for j in range(i + 1, len(self.rooms)):
                room1 = self.rooms[i]
                room2 = self.rooms[j]

                # Create polygons for the rooms
```

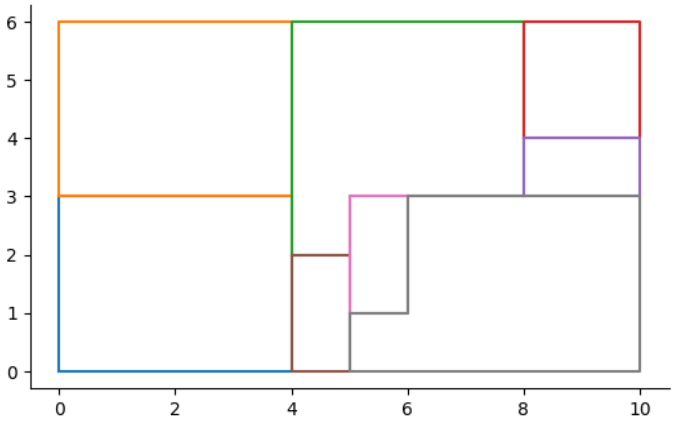


Figure 1: A simple floor plan example

```
                polygon1 = Polygon([(vertex.x, vertex.y) for
                    vertex in room1.vertices])
                polygon2 = Polygon([(vertex.x, vertex.y) for
                    vertex in room2.vertices])

                # Check for overlap
                if polygon1.overlaps(polygon2):
                    print(f'{room1.name} and {room2.name}
                        overlap')
                    return True
        return False
```

Listing 7: Room overlap check

5.8 Normalization of Fitness Values

The fitness values calculated from the evaluation indicators are normalized to the range of [0, 1] using the Min-Max normalization method. The key idea of Min-Max normalization is to scale the original values so that they fall within a specific range. It is critical to find the correct minimum and maximum values for each evaluation indicator to ensure that the fitness values are normalized correctly. For example, the minimum and maximum values for the evaluation indicator "DIS (BR, BA)" are the minimal width of a room and the sum of the width and length of the house, respectively. Listing 8 below is an example of how the fitness values are normalized using the Min-Max normalization method:

```
# calculate the walking distance
distance = (abs(BR_center.x - BR_door_center.x) + abs(
    BR_center.y - BR_door_center.y) +
```

```
            abs(BR_door_center.x - BA_door_center.x) + abs(
                BR_door_center.y - BA_door_center.y) +
            abs(BA_door_center.x - BA_center.x) + abs(
                BA_door_center.y - BA_center.y))
print(f'Distance: {distance}')

# normalize the distance
normalized_distance = (distance - BR.min_width) / (house.
    width+house.depth - BR.min_width)
return normalized_distance

print(f'Normalised Walking distance bedroom-bathroom: {
    DIS_BR_BA(house1)}')
```

Listing 8: Normalization of fitness values

The output of the normalized walking distance between the bedroom and bathroom is shown below:

```
  Distance: 3.5
  Normalised Walking distance bedroom-bathroom:
      0.17218543046357618
```

As shown in the output, the walking distance between the bedroom and bathroom is 3.5, and the normalized value is 0.17218543046357618, which falls within the range of [0, 1]. This normalization process ensures that the fitness values are comparable and can be combined to form the final fitness value.

6 Conclusion and Future Work

In this project, Particle Swarm Optimization (PSO) is applied to the entire evolutionary optimization process for floor plan design. This approach is aimed at developing an efficient, user-friendly workflow that optimizes spatial layouts while meeting functional and aesthetic requirements. With PSO as the foundation, the goal is to complete the floor plan optimization process by implementing an adaptive, automated pipeline capable of producing high-quality layouts that align with user needs.

To ensure a smooth workflow, each component of the system will be tested rigorously for both efficiency and usability. Initial testing will focus on the system's ability to generate optimized layouts within a feasible time frame. Any bottlenecks identified during testing will prompt optimizations to increase processing speed, ensuring the final system performs efficiently in a real-world setting. This will involve streamlining the PSO configuration and refining the interaction between components, as well as incorporating caching or faster algorithms where necessary to handle larger floor plans.

One additional aspect of this project is to generate vivid, detailed floor plans, including features such as furniture placement, texture suggestions, and

enhanced visualizations to assist both designers and end-users in better understanding and interacting with the layouts. By incorporating these visual elements, the system aims to produce floor plans that are both functionally optimized and visually appealing, which can be valuable for designers as well as clients.

Moreover, the project will explore integration with AI models to further enhance the system's capabilities. By using AI, the floor plan generation process can be augmented to suggest or adapt layouts based on user preferences or trends in design, making it a more versatile tool for various applications. This integration would also involve machine learning techniques that learn from user feedback to iteratively improve the generated layouts, aligning them more closely with client needs over time.

7 Plagiarism Declaration

I hereby declare that this submission is my own work and to the best of my knowledge, it contains no material previously published or written by another person, except where due to acknowledgment is made. Furthermore, I believe that it contains no material which has been accepted for the award of other degree or diploma in any university or other tertiary institutions.

8 References

[1] A. M. Brintrup, H. Takagi, and J. Ramsden. Evaluation of sequential, multi-objective, and parallel interactive genetic algorithms for multi-objective floor plan optimisation. In F. Rothlauf, J. Branke, S. Cagnoni, E. Costa, C. Cotta, R. Drechsler, E. Lutton, P. Machado, J. H. Moore, J. Romero, G. D. Smith, G. Squillero, and H. Takagi, editors, *Applications of Evolutionary Computing*, pages 586–598, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[2] B. Chaichi and A. Andaji Garmaroodi. Multi-objective optimization of a residential zone by proposing appropriate comfort factors using none dominated sorting genetic algorithm. *Journal of Building Engineering*, 86:108842, 2024.

[3] A. de Almeida, B. Taborda, F. Santos, K. Kwiecinski, and S. Eloy. A genetic algorithm application for automatic layout design of modular residential homes. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002774–002778, 2016.

[4] M. Nisztuk and P. B. Myszkowski. Hybrid evolutionary algorithm applied to automated floor plan generation. *International Journal of Architectural Computing*, 17(3):260–283, 2019.

[5] R. Subramanian, T. DheenaDayalan, T. Badhrirajan, C. Dhinakaran, C. Devakirubai, and R. R. Sudharsan. An automated house plan generator lever-

aging genetic algorithms. In *2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, pages 1–6, 2021.

[6] L. Wang. A hybrid genetic algorithm–neural network strategy for simulation optimization. *Applied Mathematics and Computation*, 170(2):1329–1343, 2005.

[7] T.-K. Wang and W. Duan. Generative design of floor plans of multi-unit residential buildings based on consumer satisfaction and energy performance. *Developments in the Built Environment*, 16:100238, 2023.

[8] S. Yan and N. Liu. Computational design of residential units' floor layout: A heuristic algorithm. *Journal of Building Engineering*, 96:110546, 2024.