International Journal of Architectural Engineering & Urban Planning, Volume 34, Number 4, 2024
DOI: 10.22068/ijaup.688

Research Paper

Automated Floor Plan Generation by a Combination of Evolutionary Algorithm (Genetics) and Machine Learning (K-Nearest Neighbors and K-Means Clustering)

Reza Babakhani [*], Mahsa Safarnejad

Department of Architecture, Science and Research Branch, Islamic Azad University, Tehran, Iran

Received: January 2023, **Revised:** July 2023, **Accepted:** May 2024, **Publish Online:** November 2024

Abstract
Design is a fundamental, problem-oriented, purposeful, and comprehensive activity. Despite the widespread use of computers in architecture, more than three decades after their introduction, the design process is still predominantly carried out by humans, starting with hand-drawn sketches which are later translated into digital formats via software. This is due to the fact that computers lack inherent design intuition, which remains a significant challenge in automating the architectural design process. This study aims to explore a novel approach that integrates artificial intelligence (AI) algorithms for the automatic generation of architectural plans. The goal is to develop a system capable of producing designs that meet user requirements while adhering to established rules, regulations, and design standards. The central hypothesis of this research posits that by combining evolutionary algorithms with machine learning techniques, it is possible to create a process that allows machines to approximate a form of design intuition. The methodology of this research includes a combination of literature review, documentation analysis, and quantitative data analysis. The study employs genetic algorithms, supervised learning algorithms, and Python libraries. The findings indicate that using feature vectors for supervised learning can facilitate the identification of optimal designs, thereby introducing a degree of "relative intuition" into machines. Additionally, the application of genetic algorithms for exploring the design space and optimizing plans based on the dimensions of the user's land proves to be effective. Finally, by storing design process experiences through algorithms, it is possible to create a foundation for reinforcement learning, which improves the system's performance over time. In conclusion, the study presents the Automated Design Intelligence (ADI) Theory as a new theoretical framework for automating architectural design, offering a potential shift in how design processes can be approached through AI and machine learning.

Keywords: *Machine learning, Genetic algorithms, Automated design intelligence.*

INTRODUCTION

Computer-aided design (CAD) software has had a dramatic impact on architectural practice since the emergence of computers in academia in the 1950s, and especially since the introduction of personal computing in the 1980s. Although early researchers envisioned a wide-ranging future interaction between computers and human designers (Negroponte, 1969), the first computer tools to be widely adopted by architectural designers were computerized versions of traditional drafting and rendering tools. While they allowed designers to produce content much faster than with traditional methods, they did not fundamentally change the process of design (Nagy, 2017).

The concept of generative design, as described in this paper, addresses this limitation by tasking a computer to explore a design space semi-autonomously, and then report back to the designer which options it considers promising for further analysis. Because a computer can process information much quicker than a human, such a system allows a much deeper exploration of complex design spaces. Traditionally, such an approach has been used to

[*] *Corresponding author: reza.babakhani@srbiau.ac.ir*

optimize a given model to achieve maximum possible performance based on concrete objectives (Marler, 2004).

Computer does not have any inherent intuition about design and the human designer must explicitly describe to the computer how to determine which designs perform better than others. The model needs to be connected to a search algorithm that can control the input parameters of the model, get feedback from the metrics, and intelligently tune the parameters to find high-performing designs while also exploring the full possibilities of the design space. One of the most promising of these algorithms is the multi-objective genetic algorithm (MOGA) which uses principles of evolution to create sequential generations of designs and evolve them to contain higher-performing designs over time (Murata, 1995).

The quantification of spatial experience has also been explored by a variety of authors. (Hillier, 1976). proposed a variety of analytical tools for studying spatial configurations which they called 'space syntax' (Peponis, 1998). extended this work by proposing a universal method for understanding plan topology through linear representation (Turner, 2001). Design is a fundamental, purposeful, pervasive, and ubiquitous activity and can be defined as the process of creating new structures characterized by new parameters, aimed at satisfying predefined technical requirements. It consists of several phases, which differ in details such as the depth of design, kind of input data, design strategy, procedures, methodology, and results (Renner, 2003).

Goldberg presents an idealized framework for conceptual design in four components: problem, designer, alternative designs and design competition, and shows how evolutionary techniques (specifically genetic algorithms) can be thought of as a lower bound on the performance of a designer that uses recombinative and selective processes (Goldberg, 1991). Rosenman has explored evolutionary models for non-routine designs (Rosenman, 1997) and has investigated the generation of creative house plans (later referred to as floorplans in this paper) using genetic algorithms (Rosenman, 1997).

The creation of floorplans has also been investigated by Gero and Schnier as an evolving representation problem that restructures the search space (Gero, 1995) by co-evolution of design and solution spaces (Poon, 1997) and using case-based reasoning by De Silva Garza and Maher (De Silva, 2000). At the same time, collaborative systems have been the focus of studies into creativity and computer-supported cooperative work (Wilson, 1991) since the early 90s. There has been a paradigm shift from computer-aided design systems to computer-supported collaborative design systems (Peng, 2001). It has been argued that much of our intelligence and creativity results from interspaces. We present a collaborative interactive genetic algorithm implementation for our model to evolve floorplans and widget layout/style design, as a user-interface development tool (Banerjee, 2008). Most approaches probe possible placements in a design space (Preas, 1979). implemented an exhaustive algorithm to select the rectangular arrangements satisfying constraints among all the possible generations. However, due to computational restrictions, the method could only handle layouts with up to ten rooms (Schneider, 2011).

Evolutionary algorithms have been applied to search layout possibilities by treating design variants via crossovers and mutation operators in these approaches. Evolutionary strategy is used to fit rooms into target envelopes while improving appropriately designed fitness functions. Furthermore, mutations are allowed during such evolution, for example, to switch rooms (crossover) in order to optimize connectivity (Knecht, 2010). Recently, an automated layout generation has been proposed to sample and efficiently explore the layout search space (Merrell, 2010). The method, however, is not designed to support interactive design refinements. In a related attempt, (Harada, 1995). designed a system that allows users to interactively drag rooms, but the possible layouts are predefined. Specifically, the algorithm searches for a matched state that best reflects user intents from a set of constructed transformations for mapping states. Moreover, only limited sets of constraints are considered, and the method does not generalize to handle manufacturing constraints. More recently, physical and manufacturing considerations have also been explored in the context of geometric form finding (Umetani, 2012). Similarly, in this work, we focus on pre-cast concrete-based constructions and consider their implications in design and layout problems.

The most common genre of FLP involves a finite number of rectangular building blocks or modules Mi ($i = 1, 2 \ldots N$), representing various activities or functional units such as departments, machines, rooms, cells, activities, or spaces. The objective is to minimize the cost of inter-module flow by placing all the modules on the packing space without overlaps, in such a way that the edges of Mi are parallel to the x and y axes respectively. It is a well-known NP-complete problem; thus, a verifiably optimal solution cannot be known even for modest-size problems (Ahmad A. R., 2005). The purpose is usually to minimize costs, time, or distance in the flow of material and occupants through different departments (Brotchie, 1971).

The methodologies vary from heuristic search methods (Liggett, 1981) and genetic algorithms (Lee, 2005) to mixed-integer programming (Irohara, 2007) and threshold-accepting algorithms (Huang, 2010). These automated design techniques caught the attention of researchers from the field of architecture as they could potentially solve the problem of large spaces associated with hospitals and schools (Liggett, 2000).

Doulgerakis (Doulgerakis, 2007) used genetic programming with an agent-based approach to assign activities (space functions) to geometric rooms. In this case, the vertical circulation, and their consequent implications were ignored (Flack, 2011). Dolgrakis tested two methods of evolutionary computation, genetic algorithm, and genetic programming, and in multi-story buildings, proposed solutions to place the staircase as a fixed space in the problem, which is repeated at each level.

There was no need to change the form and move the floors and this element was drawn separately in the plan. Beyond the issue of floor connections, Zimmerman used a rectangular partitioning method in which restrictions on several specific levels were applied, such as vertical wall alignments or recessed space constraints on walls and walls (Zimmermann, 2005). This method was also considered for a long time in order to complete the previous methods.

But over time, the goal of productive and computer-aided design is to plan living space based on the placement of objects in regular or irregular shapes by software in a specific architectural design by a designer, and this has become one of the most popular design methods. This is because of the widespread use in people's daily lives, such as placing books on bookshelves (Crasto D. , 2005), Placing cars in the car park, as discussed by (Lee J. Y., 2006), and arranging objects or pictures in PowerPoint slides share similar principles. Solving this problem is one of the most fascinating and challenging tasks researchers are trying to address (Keckeisen, 2004).

The production of architectural designs requires spatial planning and the goal is to find practical places and dimensions for a set of interconnected objects that meet all design requirements and have the maximum design quality in terms of design preferences (Michalek, 2002). This is the need of architects and society because they have to come up with acceptable designs instead of non-optimal solutions. Computer technology was used in the mid-1960s for the structural implementation of architectural designs (Levin, 1964). Now, with advances in computational capabilities over the years and algorithm modifications, it seeks to solve such design problems and find practical solutions to minimize human interference in the design process, along with the dual challenge of addressing constraints. Topological and dimensional properties of spaces (Verma, 2010).

It is going back and forth the topological constraints control a set of spaces and the relationship between them and make them responsive to each other in order of spatial arrangement, while the dimensional constraints applied to space are dimensions that are possible for a particular space. Different researchers and architects have given different preferences to two sets of constraints and have considered different priorities in the automated design process (Thakur, 2010).

In fact, by examining the theoretical foundations of the previous research, the problem extracted is that the process of designing architectural plans is not a task that can only be done by machines and algorithms and also all the effective points and data can't be calculated and applied by the designer, rather, a method must be innovated that combines the original design by the designer with the optimization, plotting, and application of rules based on user needs by algorithms.

The main objective of this research is to find a solution that can minimize the human interventions in the optimization process and achieve an ideal design, the primary design is somehow done from human studies and activities and ideas, and not like putting a book on a shelf, in fact, the goal is to achieve an appropriate method of automatic architectural plans production by artificial intelligence with the ability to identify the appropriate design based on user needs. At the same time, it can provide rules, regulations, and design standards based on the dimensions and sizes of the designed land.

The research hypothesis now is that a process can be created by using evolutionary algorithms (genetics) and machine learning algorithms (k-means clustering) and (k-nearest neighbors) interactively and simultaneously so that this objective (automatic architectural plans production by artificial intelligence) is realized.

BACKGROUND RESEARCH

Table 1 summarizes key developments in the synthesis and optimization of architectural plans. Mitchell (1976) introduced the first theory, achieving sixteen possible positions for squares and rectangles. Two decades later, Jo and Gero (1996) simulated architectural plans using genetic algorithms. In 1997, researchers—Rosenman, Schnier, Gero, Kazakov, and Jagielski—advanced this approach by generating sample plans with genetic algorithms and genetic programming.

From 1998 to 1999, Garza and Maher, Bentley, Elezkurtaj and Franck, in a study using genetic algorithms and evolutionary strategy algorithms, were able to create architectural spatial combinations. Following research from 2001 to 2002, for the first time, they were able to generate architectural plans through genetic algorithms, sequential quadratic programming, and reproduction of simulations, in addition, from 2003 to 2008, attempts were made to generate architectural plans with genetic algorithms and genetic programming, led by Makris, Virirakis, Makris, Bausys and Pankrasovait, Homayouni, Doulgerakis, Banerjee et al., and Serag et al.

From 2009 to 2011, research in this paradigm shifted from using single genetic algorithms or programming to combining genetic algorithms with other emerging computational techniques. Researchers such as Inoue and Takagi, Wong and Chan, and Benjamin Dillenburger integrated genetic algorithms with other methods to develop innovative architectural plans. Additionally, Thakur et al., de la Barrera Poblete, Knecht, and Flack introduced alternative generative approaches, expanding the possibilities for architectural design. Between 2011 and 2012, Ricardo Lopes et al. and Reinhard Koenig advanced this work further by employing hierarchical algorithm methods to generate new spatial divisions for architectural plans.

The method column of Table 1 highlights research conducted over the past decade, emphasizing a growing trend among researchers toward the combination of algorithms. The integration of hybrid algorithms has shown promising results in generating architectural plans that account for multi-factor design considerations. The hypotheses of these studies suggest that combining diverse artificial intelligence algorithms—such as genetic algorithms, optical hill climbing, and swarm intelligence—not only accelerates the plan generation process but also enables the inclusion of various environmental data, energy parameters, and design standards into the process while maintaining adequate quality.

However, in prior studies, algorithm combinations have primarily occurred within the same category, such as combining genetic algorithms with genetic programming or hill climbing with genetic programming. These attempts focused on leveraging evolutionary algorithms to explore the design space and propose suitable solutions. Despite extensive testing, these approaches faced significant limitations. For example, they failed to consistently produce more than ten spatial elements without encountering detrimental mutations during the design process. These mutations, inherent to the structure of evolutionary algorithms (analogous to genetic mutations in nature), could be either beneficial or harmful. Additionally, evolutionary algorithms are time-intensive, requiring significant computational resources to explore the design space effectively.

Table 1. Background of Plan Generation Research (Rodrigues, 2013).

Researcher	Method	sA	oO	aB	bB	eF	fL	S	iD	eW	eD	wD	oF	Year
Thakur et al	GA/DA	•							•		•		gts	2010
de la Barrera Poblete	GA+VD							•					g	2010
Knecht	GA/ES+K-D				•			•					gt	2010
Flack	GA/GP	•			•		•	•					gt	2011
Ricardo Lopes et al	AH													2011
Reinhard Koenig	AH													2012
Rodrigues	ES+SHC	•	•	•	•			•	•	•	•	•	gt	2012
Victor Calixto	GA/GP	•			•		•	•			•	•		2015
Stanislas Chaillou	GAN	•			•		•	•						2019
RUIZHEN HU	Graph2Plan	•			•		•	•			•			2020
Maciej Nisztuk	GA	•			•		•	•			•	•		2020

GA: Graph algorithm, SO: Synthesis and Optimization, GA: Genetic Algorithm, GP: Genetic Programming, ES: Evolutionary Strategy, SA: Simulated Annealing, SQP: Sequential Quadratic Programming, L: Lindenmayer System, VD: Vernoy Diagram, DA: Digestra Algorithm, SHC: Search of Hill Climbing, AH: Hierarchical Algorithm, Graph2Plan: Pixel to Pixel, t: Topological, h: Heating, c: Cooling, l: Lighting, s: Walking Distance, oF: Objective Function, wD: Wall Dimensions, eD: External Door, eW: Window, iD: Internal Door, S: Spaces, fL: Floors, st: Stairs, eL: Elevators, eF: Equipment- Furniture, bB: Building Area, aB: Adjacent Buildings, oO: Openings orientation, sL: Location, sA: Adjacent spaces.

The second group of studies involves deep learning algorithms, which often operate without incorporating evolutionary algorithms. Generative Adversarial Networks (GANs) have gained popularity in recent years for generating architectural plans. These algorithms rely on pixel-based representations of plans, simulating spatial arrangements by manipulating color pixels to reflect user concepts. However, this method has notable limitations, including a lack of scalability, as the architectural design inherently operates on scales measured in meters and centimeters rather than pixels. Furthermore, these pixel-based approaches cannot accommodate optimization processes or interpret regulatory frameworks, such as National Building Regulations. Consequently, GAN-based methods are often restricted to generating preliminary designs or conceptual ideas, leaving significant room for subjectivity and variability in outcomes.

A review of prior research highlights two major distinctions in the present study. Firstly, this research introduces a novel approach that combines an evolutionary algorithm (genetic algorithm) with supervised learning techniques (e.g., k-means clustering) to improve plan generation. Secondly, it introduces the concept of utilizing user input as a numerical vector to guide the algorithm in understanding user requirements and generating designs aligned with environmental standards and conditions.

The findings underscore that architectural design cannot be fully automated by removing the architect from the process. Instead, the architect's role is essential as a guide or trainer, providing the machine with curated architectural data and examples of optimized plans (as illustrated in Figure 1). The algorithm, trained on this data, can then adapt to climatic conditions, user preferences, and urban regulations to generate a design. Finally, the design can be plotted in architectural software and delivered to the user, ensuring a balance between automation and professional oversight.

METHODOLOGY

The proposed method is based on evolutionary algorithms (genetics) and machine learning (KNN and k-means clustering) to generate architectural plans called MLGAFPG, which is proposed to allocate space in a two-dimensional level for common urban lands. The MLGAFPG consists of two techniques, the first based on machine learning and the second on genetic algorithm. In the machine learning algorithm, the input information is entered by the user, and samples of architectural plans based on machine-supervised learning have already been learned.

The algorithm proposes designing a considered plan among several thousand learned models based on the user's need and land data. Here, these samples are collected based on the most common dimensions of urban lands and completed according to architectural standards. Then, through the second algorithm, as genetic evolutionary algorithms and according to the specialized and professional preferences and limitations of architecture, by an evolutionary strategy (Genetic), the production process and matching the floor plans of the architectural floors begin with the dimensions of the user's mainland.

This is a special two-step approach that can be used to produce architectural plans. In fact, in the first stage, architectural plans are trained to the machine learning algorithm as learnable examples, and then in the next stage, with the help of GA, in the process of several repetitions, it changes the geometric form and brings to the main coordinates of designing land. The aim of the GA stage is a local search for the most suitable position of the architectural plan spaces in the selected land based on common urban examples for design. With the help of GA, the best state is maintained in the stage of searching algorithm generation space, which is equal to the coordinate data of the spaces. GA is designed to search the areas among the sample spaces generated in the machine learning algorithm and the main dimensions are entered as land data.

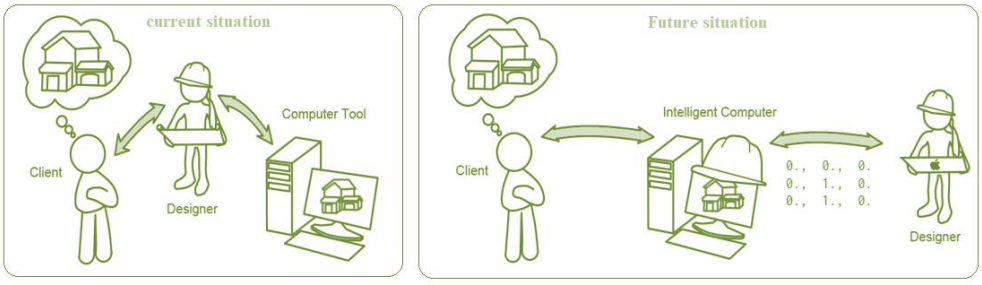

Fig 1. Changing the position and role of the architect in the design process

The proposed method of automated production of architectural plans generates a process in which a set of objects (spaces/rooms, exterior doors and windows, and interior doors) are drawn on a two-dimensional plane, a method in which topological relations, both geometric constraints and user requirements are satisfied. In fact, each floor map design produced is a set of plans learned by the machine learning algorithm and combined with user input information in a hierarchical manner. Each plan floor is made up of different spaces and elements, including the exterior windows, exterior and interior doors, and the floor of stages. This method can solve the main problem of mutation in the single evolutionary algorithms that result in the production of random spaces in more than 8 or 10 spaces.

Figure 2 illustrates a segment of the process for selecting and resizing architectural plans to align with the user-specified land dimensions. Initially, the machine learning algorithm selects the most appropriate design sample from the database of pre-trained primary plans based on the user-defined features, as outlined in Table 2. Following this, the dimensions of the selected sample are adjusted to match the dimensions of the user's land using a genetic algorithm.

Additionally, the system is designed to incorporate various supplementary conditions, including compliance with design regulations, energy optimization, economic cost analysis, integration of solar energy production systems, material modulation, and other advanced functionalities. These enhancements can be applied to the initial design plan to meet broader requirements. However, the current study focuses exclusively on the selection and dimensional adjustment of design samples. The methodologies for implementing the additional capabilities will be addressed in future research.

DISCUSSION

The need for interaction between machine learning algorithms and genetics algorithms is to build the automatic design intelligence of residential plans in Tehran city based on user needs and architectural criteria. In this research, the method of this interaction is analyzed and examined. To achieve this goal, steps must first be taken and the architectural data, which is qualitative and quantitative, turned into numbers.

In Figure 3, we can see an example of the spatial relationships of the architectural plan, which has been converted to zero and one codes, and the arrangement of these zeros and ones can be input for the algorithm to learn the spatial relationships in the plans numerically.

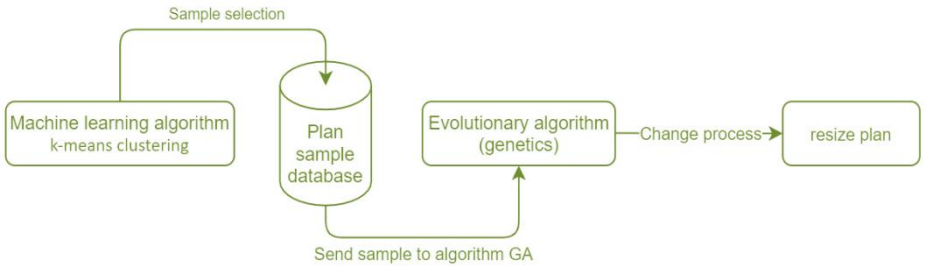

Fig 2. The Process of Changing the Dimensions of the Sample Plan for the Land Proposed by the User

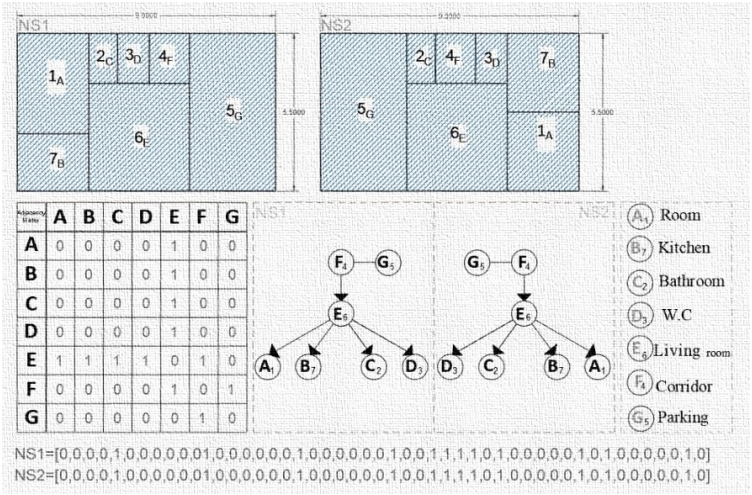

Fig 3. Construction of a Numerical Vector of the Spatial Relationships of Architectural Plans

In the following, Figure 4 shows how the genetic algorithm based on the search structure of the problem space can share the added area to the dimensions of the land among the available spaces vertically and horizontally and align the proposed plan for the user with it. In fact, in this algorithm, the goal is to achieve a plan where the user wants to receive a map based on the dimensions of his land and submit it to legal authorities, so the plan must comply with all designing rules and regulations and land dimensions.

Figure 5 shows how the algorithm directs each space in the form of a square and rectangle with central control through rectangular points (x, y) in vertical and horizontal directions to match the added space to the dimensions of the trained plan by the machine to the main dimensions of the land in a two-way manner. Over time, this method classifies plans that have been generated with new dimensions, by non-supervised machine learning algorithms and adds in the memory of the algorithm's automatic design intelligence as a new experience. These experiences will continue to be the basis for generating better plans by automated design intelligence algorithms.

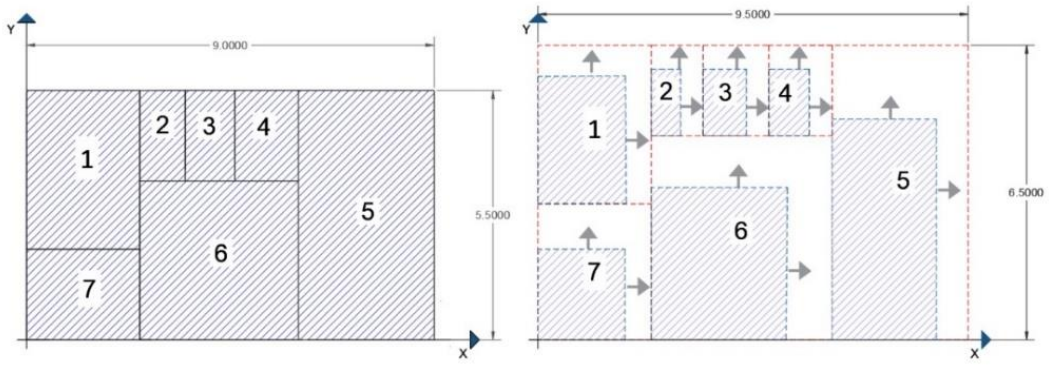

Fig 4. Matching the Dimensions of Plan Spaces by Genetic Algorithm with the Dimensions of the Main Ground

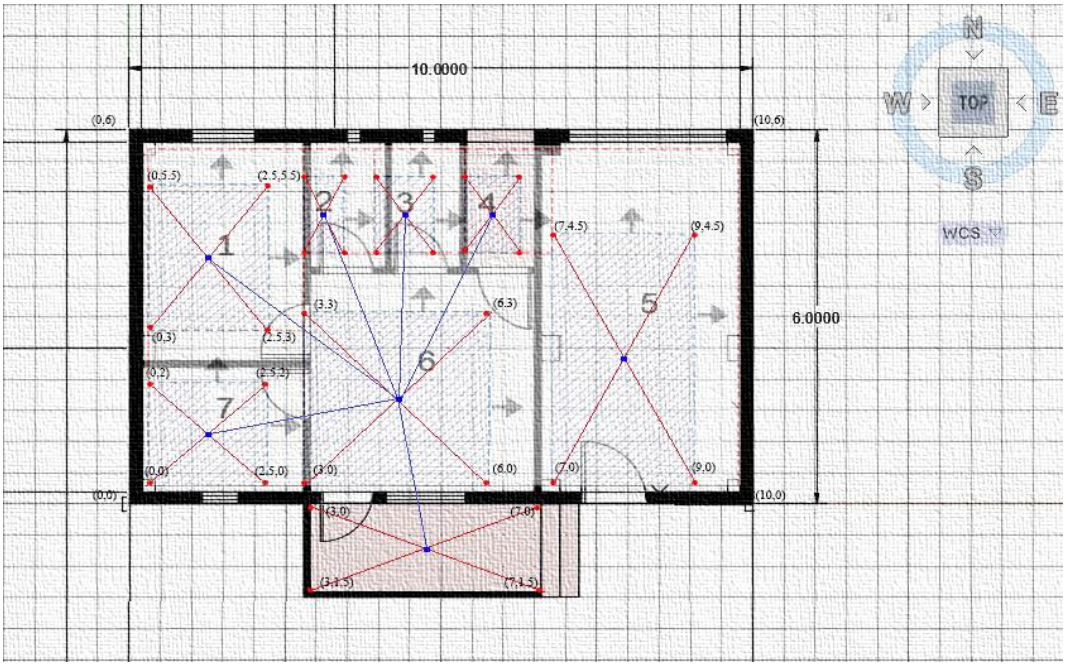

Fig 5. Searching the Design Problem Space Using a Genetic Algorithm

As mentioned before, in the former automated designing methods, only evolutionary algorithms have been used and all spaces, openings, and communication spaces have been found and applied by the search method in problem space; but in this research, for the first time, a combination of two machine learning algorithm and the genetic algorithm will be the basis for designing architectural plans. To achieve the automatic design intelligence of architectural plans, the features that make up an architectural plan are first extracted and categorized by data analysis algorithms, as illustrated in Table 2. These are some of the features that differentiate architectural plans from one another.

Here, in number 1, the user determines the design location, which is one of the cities, and in the second input, he enters the minimum and maximum width allowed for residential land, which is based on the most common dimensions built in the last 50 years.

Additionally, the third input is the virtual length that is selectable for the land; the fourth input is the land type which is grouped into four parts: north, south, east, and west; the fifth input is the type of building in terms of typical villa, multi-story villa and apartment; the sixth input is the type of windows that are classified according to the personality and interest of people into large, small, medium and floor to ceiling; the seventh input is the number of floors from one to five, which is adjusted based on the number of common floors; the eighth input is the number of units that the user can specify; the ninth input is the number of rooms that can be specified as input data from 1 to 5 rooms.

The tenth input is the number of resident population per unit, which is directly related to energy problems; the eleventh input is the parking lot orientation, which is on the right or left side of the building for left-hand and right-hand people; the twelfth input determines the location of the bedroom (Some people are interested in the bedrooms being in different directions and experiencing different lights).

Thirteenth input is the kitchen orientation, which is determined by the user based on people's interests and tastes, the fourteenth input is the kitchen model from the ordinary type to an island one; the fifteenth input is the bathroom model, which has several models; the sixteenth input is a corridor that allows people to have a corridor in the house or need uniform spaces without a partition. Finally, the seventeenth to twentieth inputs are based on national regulations. These cases can determine the type of plan that automatic design intelligence will design for the user and draw in AutoCAD software.

To make it easier to write functions and function calls in the problem statement process, subjects and inputs are abbreviated, and according to Table 3, each of the variables and input data is equivalent to a short abbreviation of the full subject. These data are then implemented as numerical data based on Figure 6 so that the machine can perceive and analyze them.

Table 2. Information on Architectural Plan Features

Row	User input information	User selection range
1	Location of the city	Tehran
2	Width of the ground	7 to 20
3	Length of the ground	10 to 25
4	Land type	North, South, Eastern, Western
5	Building type	Home, House, Apartment
6	Window type	Small, Medium, Large, Floor to ceiling window
7	Number of floors	1 to 5
8	Number of units	1 to 3
9	Number of rooms	1 to 5
10	Number of population	1 to 5
11	Parking places	Right or Left
12	Bedroom location	North, South, East, West
13	Kitchen locations	North, South, East, West
14	Kitchen model	Closed, Open, Kitchen Island
15	Bathroom model	Bathroom master, Bathroom and toilet, Bathroom
16	Corridor	Yes or No
17	Stairs	According to the regulations
18	Elevator	According to the regulations
19	Lightwell	According to the regulations
20	Columns	According to the regulations

Table 3. Nomenclature

1	USR	User selection range
2	LC	Location of the city
3	WG	Width of the ground
4	LG	Length of the ground
5	LT	Land type
6	BT	Building type
7	WT	Window type
8	NF	Number of floors
9	NU	Number of units
10	NR	Number of rooms
11	NP	Number of population
12	PP	Parking place
13	BL	Bedroom location
14	KL	Kitchen location
15	KM	Kitchen model
16	BM	Bathroom model
17	COR	Corridor
18	ST	Stairs
19	EL	Elevator
20	LW	Lightwell
21	COL	Columns
22	N	North
23	S	South
24	E	Eastern
25	W	Western
26	Cal	Construction area limit
27	Ewth	Exterior wall thickness
28	HOM	Home
29	HOU	House
30	AP	Apartment
31	SM	Small
32	ME	Medium
33	LA	Large
34	FCW	Floor-to-ceiling window
35	Ri	Right
36	Le	Left
37	CLO	Closed
38	OPE	Open
39	KI	Kitchen Island
40	BMAS	Bathroom master
41	BTO	Bathroom and toilet
42	BAT	Bathroom
43	UIIV	User input information vector
44	COREX	Exterior Corridor
45	SMat	Space matrix
46	SPdir	Space direction
47	Midd	Minimum door dimensions
48	Miwd	Minimum window dimensions
49	MifH	Minimum floor height
50	MisL	Minimum space length
51	MisW	Minimum space width
52	MiA	Minimum minimum area
53	Gal	Gross area limit
54	Iwth	Interior wall thickness

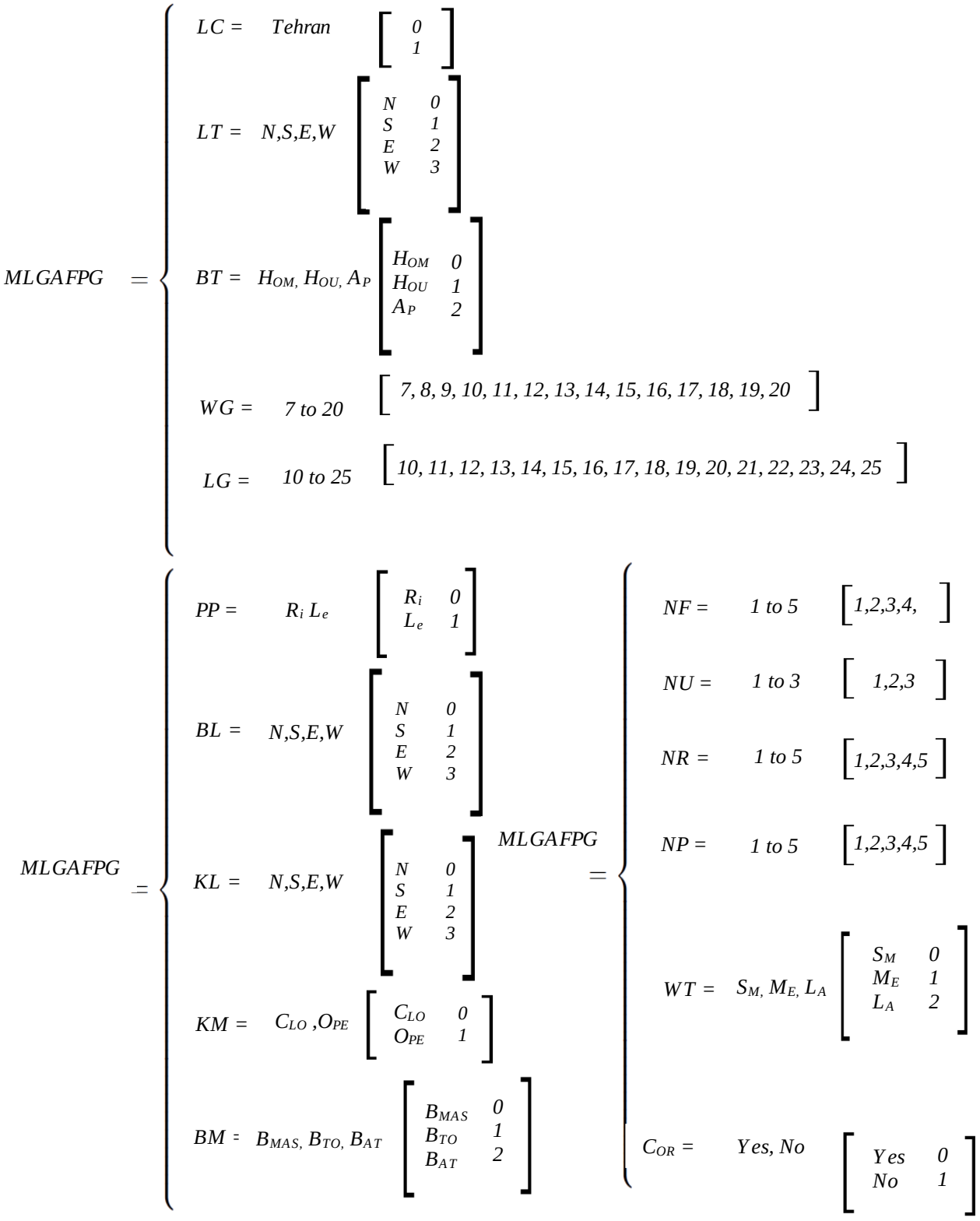

Fig 6. View of the Input Detection Algorithm and the Design Goal in Automated Design Intelligence

Based on the MLGAFPG algorithm, which is the automatic design intelligence of plans, it receives input data from the user and then converts it into encrypted data, which will contain various numbers. It then analyzes those numbers and converts encrypted data into a single cryptographic vector like Figure 7. Then, the resultant automatic design suits the user's needs, calculates it using the mathematical relation in Figure 8, and provides the best plan design state based on instructions, learning, and experience to the user.

$$U_{IIV} \begin{Bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 2 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 2 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 1 \end{Bmatrix} \qquad U_{IIV} = \begin{bmatrix} 0, 1, 0, 1, 0, 0, 1, 0, 0, 2, 1, 1, 1, 2, 2, 1 \end{bmatrix}$$

Fig 7. Converting Numerical Cryptographers to Machine Unit Vectors

Based on Figure 6, the calculation of the similarity of the two data input vectors by the user and the data in the memory of automatic design intelligence learning begins here to reduce the complexity of the sample vector to two, the numbers 2, 3, 4, 2 and 1, −2, 1, 3 are converted and then based on the above relation, the analogy operations of vector A and vector B are applied by the relation xi-yi, which can be observed according to Figure 6 as (2-1)2 +(3 +2)2 + (4-1)2 +(2-3)2, the product of xi-yi subtraction and their exponentiation is the equation √ (1 + 25 + 9 +1) that if we apply the addition operators, we will encounter the number √ (36) and then if we get it out from the radical, we will reach 6, which is the similarity of the two input vectors to the above formula.

Based on the above formula of design intelligence, after recognizing the user's needs based on Figures 6, 7, and 8 and perceiving the spaces and the way they are placed in the architectural plan, one should be able to adjust it based on national standards, rules, and regulations and optimized by energy topics and climatic data must also be applied by automatic design intelligence in the designing process. Elaboration on each case is beyond the scope of this article, so in future research, rules and regulations and spatial layout will be further discussed.

According to Table 4, the minimum dimensions and sizes that must be observed in the design of architectural spaces have been extracted. It is stated that each space and element involved in architectural plans should have what dimensions, size, and orientation to be approved. It is also trained as shown in Figures 6, 7, and 8 and as vectors and numerical data to the automatic design intelligence of the plan, so that it can apply them in the designing process. Python programming language with its special libraries has been used to learn this data by the automatic design intelligence of architectural plans.

$$\text{d (X.Y)} = \sqrt{\sum_{i=1}^{\vec{n}} d(xi - yi)}$$

$$\text{d (X.Y)} = \sum_{i=1}^{\vec{p}} |xi - yi|$$

Determine the Euclidean distance between $\vec{u}$ = (2,3,4,2) and $\vec{v}$ =(1,−2,1,3).

$D(\vec{u},\vec{v}) = \|\vec{u} - \vec{v}\| = \sqrt{(2-1)^2 + (3-2)^2 + (4-1)^2 + (2-3)^2}$

$D(\vec{u},\vec{v}) = \|\vec{u} - \vec{v}\| = \sqrt{1 + 25 + 9 + 1}$

$D(\vec{u},\vec{v}) = \|\vec{u} - \vec{v}\| = \sqrt{36}$

$D(\vec{u},\vec{v}) = \|\vec{u} - \vec{v}\| = 6$

Fig 8. The Formula for Calculating Each Cryptographic Vector with a Similar Vector in Cartesian Coordinates

Table 4. Minimum Rules and Regulations to be Observed in the Design of Plans

Level	Name Space	SMat	SPdir	Midd	Miwd	MifH	MisL	MisW	MiA
L1	Corridor Entrance	0	-	1 m	-	3/24	-	1/40	-
	Corridor	0	-	1 m	-	3/24	-	1/10	-
	bedroom	1	East	90 cm	1/8	3/24	-	2/70	12
	bedroom	1	East	90 cm	1/8	3/24	-	2/5	6/5
	Living room	1	South	1 m	1/8	3/24	-	2/70	12
	Kitchen	2	North	1 m	1/8	3/24	-	2/15	7/5
	Parking	0	-	3 m	-	2/88	5	2/5	12/5
	Bathroom	2	West	80 cm	1/8	3/24	-	1/20	-
	Balcony	1	East	80 cm	-	1/10	1/30	1/20	1/56
	W.C	2	West	80 cm	1/8	3/24	1/20	1/10	-

After learning the standards and minimum criteria for design by automated design intelligence, the spatial relationships of the plans and neighborhoods should be created by the spatial relations matrix of Figure 10, and this section should be simultaneous with the application of the standards along with equalization of the dimensions of the userland input that its infrastructure and build density, which has already been done in the data analysis section.

This starts the designing process. In fact, after receiving input information and converting it into cryptographs, and recognizing the user's needs automatic design intelligence tries to analyze information such as dimension, size, occupancy level, density, climate, and other basic variables. After analyzing, classifying, and converting them to cryptographic vectors, it designs and locates by referring to the training learned from different types of plans.

This process is similar to reading cryptographic vectors; Climate, criteria, and other design variables will continue, and at the same time, with the help of genetic algorithms, the dimensions of learning plans or automated design intelligence experiences will be closer to the dimensions of the standard occupancy level, to the extent that input dimensions and machine drawing dimensions to be the same. The whole process takes 50 to 100 seconds from the time the information is received by the algorithm to the time it is analyzed, grouped, and drawn in AutoCAD software and later stored and sent to the user according to Figure 9.

Based on Figure 10, In order for the automatic design intelligence to be able to draw the plans required by the user accurately and with the correct spatial relations, it is necessary to learn the spatial layout of the machine in the language of zero-one codes. To do so, first, a matrix from spatial layout is defined in the form of zero-one codes that if there is a connection between two spaces, the number one is used. If there is no number zero, and if there is a space inside another space with a partition door, the number two is used.

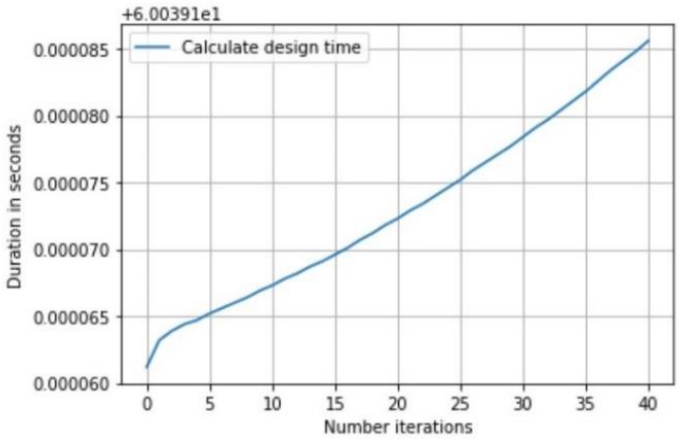

Fig 9. Calculation of Analysis Time and Application of Design Process by Automated Design Intelligence

$$\text{Design 1}=\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \\ S_8 \end{array}\begin{array}{c}\begin{array}{cccccccc} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 \end{array}\\ \begin{pmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&1&1&0&1&0&1&0\\ 0&0&0&0&0&1&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix}\end{array}$$

$$\text{Design 2}=\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \\ S_8 \end{array}\begin{array}{c}\begin{array}{cccccccc} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 \end{array}\\ \begin{pmatrix} 0&1&0&0&0&1&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0\\ 1&0&1&1&0&0&1&0\\ 0&0&0&0&0&1&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix}\end{array}$$

$$\text{Design 3}=\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \\ S_8 \\ S_9 \end{array}\begin{array}{c}\begin{array}{ccccccccc} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 & S_9 \end{array}\\ \begin{pmatrix} 0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1&0\\ 0&0&1&0&0&0&0&1&0\\ 0&0&0&1&1&1&1&0&1\\ 0&0&0&0&0&0&0&1&0 \end{pmatrix}\end{array}$$

$$\text{Design 4}=\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \\ S_8 \end{array}\begin{array}{c}\begin{array}{cccccccc} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 \end{array}\\ \begin{pmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&1&0\\ 0&1&1&1&1&1&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix}\end{array}$$

$$\text{Design 5}=\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \\ S_8 \\ S_9 \end{array}\begin{array}{c}\begin{array}{ccccccccc} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 & S_9 \end{array}\\ \begin{pmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&1&1&0\\ 0&0&0&0&0&1&0&0&0\\ 0&1&0&0&0&0&0&1&0\\ 0&0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&1\\ 0&0&0&0&0&0&0&1&0\\ 0&0&0&1&0&0&0&1&0 \end{pmatrix}\end{array}$$

$$\text{Design 6}=\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \\ S_8 \\ S_9 \end{array}\begin{array}{c}\begin{array}{ccccccccc} S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 & S_9 \end{array}\\ \begin{pmatrix} 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0&0\\ 0&1&0&0&0&0&1&0&0\\ 1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&1\\ 0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0\\ 1&0&1&1&1&0&0&1&0 \end{pmatrix}\end{array}$$

Fig 10. A matrix of Spatial Relationships and Its Transformation into a Plan by Automated Design Intelligence

Notably, this method is used for columns and positioning doors and windows and their orientation, and after the matrices are prepared, it is written in Python programming language as special codes in combination with the library of artificial intelligence. Therefore, it can draw its automatic design intelligence as in Figures 11 to 16 in AutoCAD software.

This is an example of plans that automatic design intelligence has calculated, designed, and drawn the occupancy level after entering data based on user interest and applying designing criteria and national building standards in a typical floor with spaces such as bedroom, kitchen, living room, parking, bathroom and toilet for a 10×20 land.

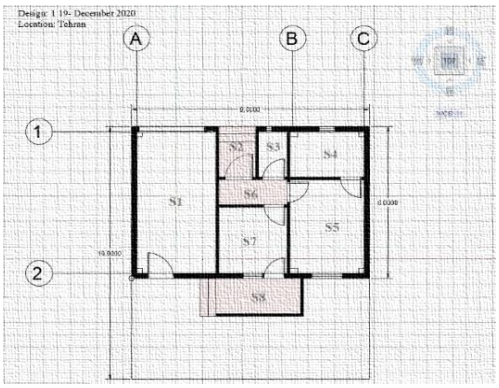

Fig 11. Sample Plan Drawn by Automated Design Intelligence

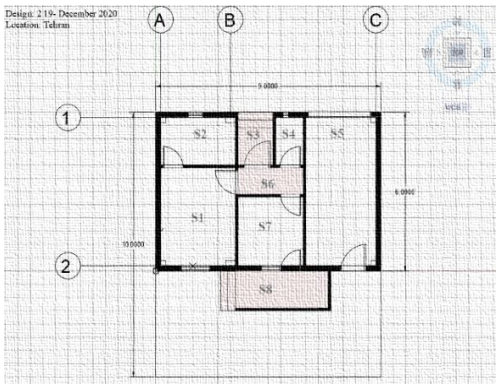

Fig 12. Plan Example Drawn by Automated Design Intelligence

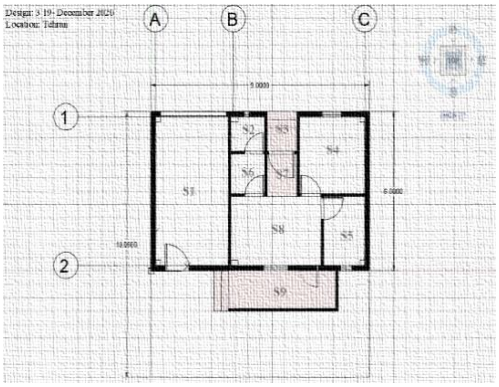

Fig 13. Sample Plan Drawn by Automated Design Intelligence

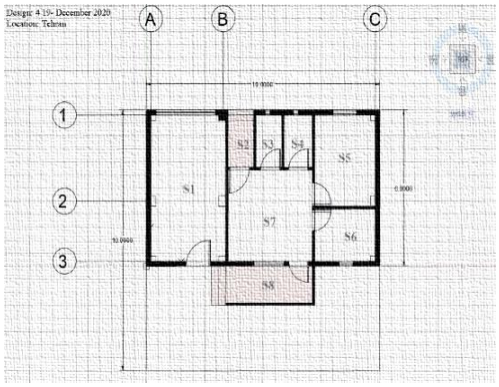

Fig 14. Plan Example Drawn by Automated Design Intelligence

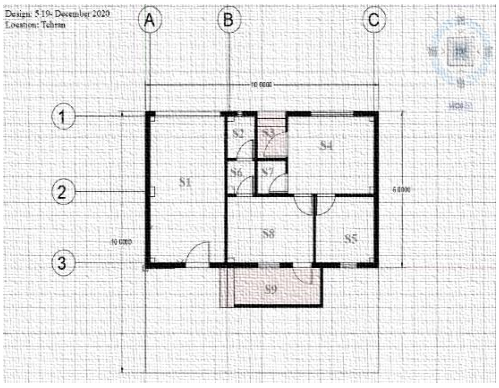

Fig 15. Sample Plan Drawn by Automated Design Intelligence

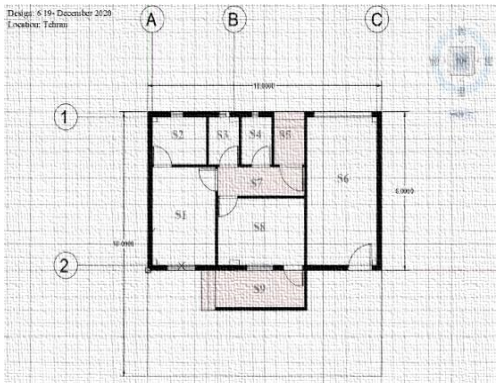

Fig 16. Plan Example Drawn by Automated Design Intelligence

To evaluate the algorithm's learning accuracy and validate the findings of this study, the Cyclin library and its learning and testing modules were utilized. A stratified approach was implemented, where 70% of the database—comprising dimensional and spatial feature vectors of residential spaces—was allocated as training data. The remaining 30% was reserved as test data to assess the algorithm's performance after the learning phase.

This training and testing process was repeated twice to ensure robustness and consistency. As depicted in Figures 17 and 18, the algorithm's accuracy improved significantly, increasing from 70% in the initial iteration to 90% after the second repetition. This demonstrates the effectiveness of the training process and the refinement of the algorithm's predictive capabilities over successive iterations.

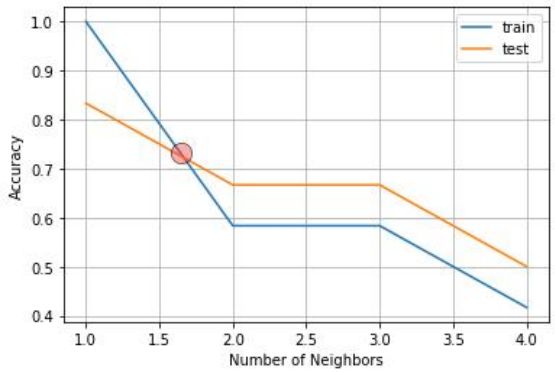

Fig 17. Algorithm Learning Rate with 75% Accuracy

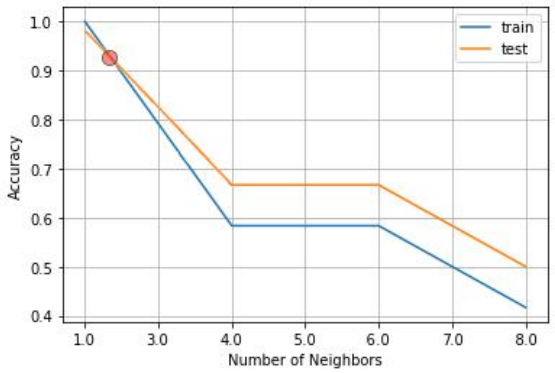

Fig 18. Algorithm Learning Rate with 93% Accuracy

Figure 19 illustrates the workflow of the proposed design process using machine learning and evolutionary algorithms. Traditional approaches often relied exclusively on evolutionary algorithms or deep learning techniques for plan generation. However, these methods lacked the capability to incorporate user input during the design process. This research is novel in enabling users to actively specify their requirements during the plan generation process facilitated by artificial intelligence.

The proposed workflow begins with the user inputting the required architectural features via a user panel. This information is transmitted to the server, which translates it into a vector of numerical attributes and sends it to the algorithm's core, specifically leveraging k-means clustering. The algorithm initiates the design process by automatically launching AutoCAD software and opening a new workspace.

The vectorized user input is processed by a machine learning algorithm, which compares it with a database of trained architectural plan features using Euclidean distance calculations. Based on these comparisons, the system assigns labels to the database entries, ranking them in ascending order from 0 to infinity. These labels represent the relevance of each trained plan to the user's specified requirements.

The system identifies the optimal plan label using the K-nearest neighbor (K-NN) algorithm. The corresponding feature vector is then utilized in the plan generation process, initiated through Python scripts and specialized libraries developed for this study. These scripts automate the plotting of architectural plans in AutoCAD software.

Simultaneously, the genetic algorithm ensures that the dimensions of the plan align with the user's specified land parameters. The genetic algorithm also optimizes the dimensions of various plan elements, such as openings, using advanced computational methods, including standards from ASHRAE and Delft University of Technology. Details on this optimization process are addressed in a separate study.

Upon completion of the design process, the final architectural plans, including energy calculations and other outputs, are saved in AutoCAD and additional file formats. These are packaged into a zip file and delivered to the user via the server. Furthermore, the system stores the design experience as a CSV file, including details such as dimensions, space syntax, room coordinates, and other relevant calculations. These saved datasets enable future development of the artificial intelligence system, enhancing the diversity and quality of generated architectural plans.

RESEARCH LIMITATIONS

This research faces several limitations. First, implementing the server and running multiple simultaneous algorithms require robust and advanced computational infrastructure. Second, as this work explores cutting-edge concepts in artificial intelligence and architecture, time and cost constraints pose significant challenges. Finally, the algorithm is currently tailored for residential villa designs, and its application to other typologies, such as commercial, office, or medical facilities, would necessitate the collection of domain-specific data and additional training of the AI model.

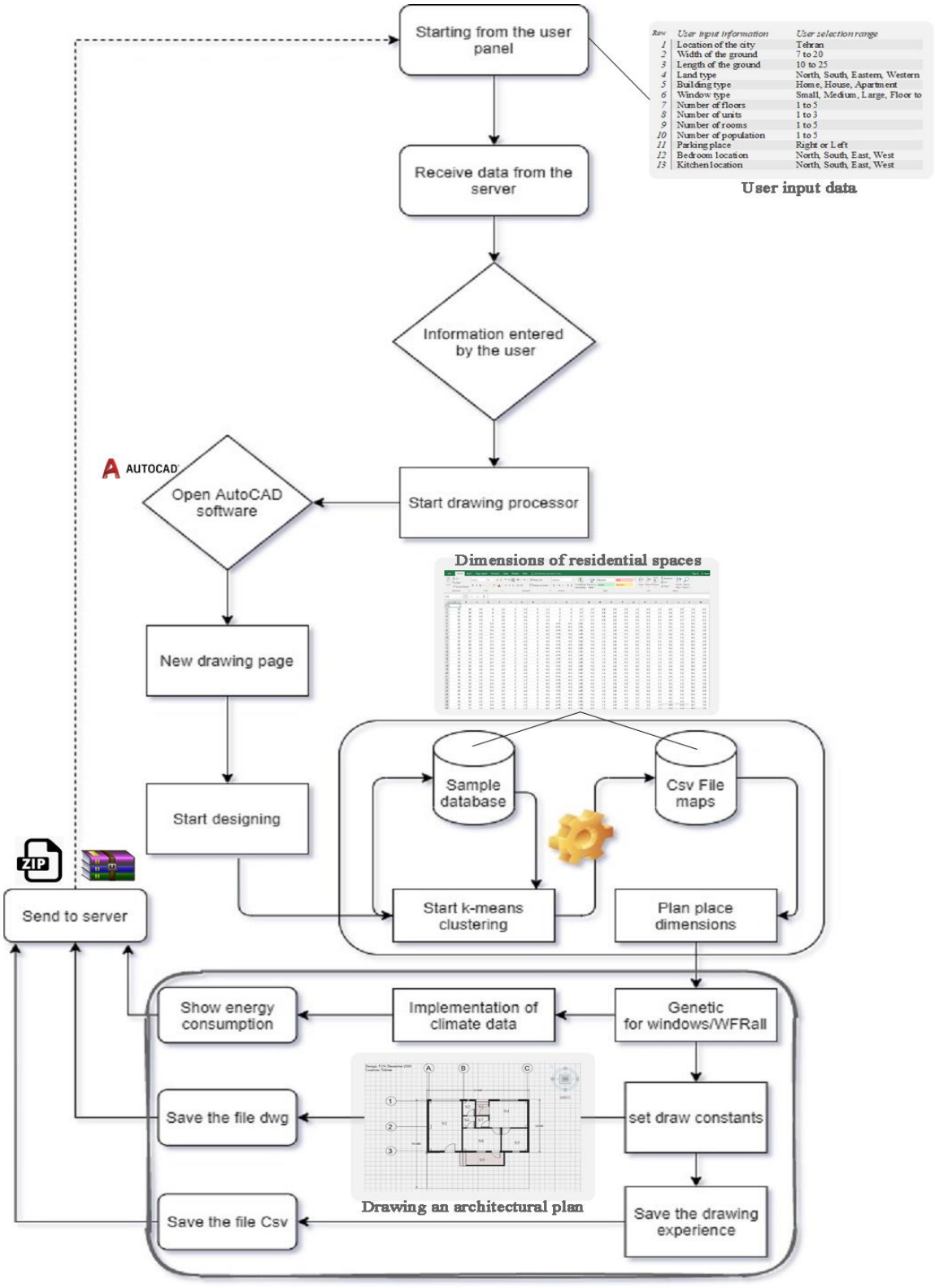

Fig 19. Part of the Process of Producing a Plan with Artificial Intelligence

CONCLUSION

The findings of this study demonstrate that employing feature vectors as part of supervised learning, combined with k-nearest neighbor and k-means clustering algorithms, effectively facilitates the identification of optimal designs. This approach enhances supervised machine awareness, enabling more informed decision-making during the design process. Additionally, the application of genetic

algorithms for searching the design space and evaluating plans based on the constraints of the designable land proves to be a robust solution.

Notably, the algorithm significantly reduces the time required for the plan design process, achieving execution times ranging from 50 to 100 seconds for a single request sent to the server. This represents a substantial improvement in efficiency compared to traditional methods.

Furthermore, the research highlights the utility of Python libraries in preserving the machine's design experiences as CSV files, enabling structured documentation of generated plans. These saved experiences are suggested for integration into reinforcement learning algorithms, offering the potential to generate more diverse and sophisticated design solutions in future iterations.

In conclusion, this paper presents the Automated Design Intelligence (ADI) (Reza, 2023). Theory as a new theoretical framework for automating architectural design offers a potential shift in how design processes can be approached through artificial intelligence and machine learning.

A.1. MATHEMATICAL MODEL

$$
\text{MLGAFPG} = \begin{cases}
\text{LC} = \text{Tehran} \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{(A.1)} \\
\text{LT} = \text{N,S,E,W} \begin{bmatrix} \text{N} & 0 \\ \text{S} & 1 \\ \text{E} & 2 \\ \text{W} & 3 \end{bmatrix} & \text{(A.2)} \\
\text{BT} = H_{OM}, H_{OU}, A_P \begin{bmatrix} H_{OM} & 0 \\ H_{OU} & 1 \\ A_P & 2 \end{bmatrix} & \text{(A.3)} \\
\text{WG} = \text{7 to 20} \quad [\ 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, \ ] & \text{(A.4)} \\
\text{LG} = \text{10 to 25} \quad [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25] & \text{(A.5)}
\end{cases}
$$

$$
\text{MLGAFPG} = \begin{cases}
\text{PP} = R_i\, L_e \begin{bmatrix} R_i & 0 \\ L_e & 1 \end{bmatrix} & \text{(A.6)} \\
\text{BL} = \text{N,S,E,W} \begin{bmatrix} \text{N} & 0 \\ \text{S} & 1 \\ \text{E} & 2 \\ \text{W} & 3 \end{bmatrix} & \text{(A.7)} \\
\text{KL} = \text{N,S,E,W} \begin{bmatrix} \text{N} & 0 \\ \text{S} & 1 \\ \text{E} & 2 \\ \text{W} & 3 \end{bmatrix} & \\
\text{KM} = C_{LO}, O_{PE} \begin{bmatrix} C_{LO} & 0 \\ O_{PE} & 1 \end{bmatrix} & \text{(A.9)} \\
\text{BM} = B_{MAS}, B_{TO}, B_{AT} \begin{bmatrix} B_{MAS} & 0 \\ B_{TO} & 1 \\ B_{AT} & 2 \end{bmatrix} & \text{(A.10)}
\end{cases}
\quad \text{MLGAFPG} \quad \text{(A.8)} \quad
\begin{cases}
\text{NF} = \text{1 to 5} \quad [1,2,3,4,5] & \text{(A.11)} \\
\text{NU} = \text{1 to 3} \quad [1,2,3\ ] & \text{(A.12)} \\
\text{NR} = \text{1 to 5} \quad [1,2,3,4,5] & \text{(A.13)} \\
\text{NP} = \text{1 to 5} \quad [1,2,3,4,5] & \text{(A.14)} \\
\text{WT} = S_M, M_E, L_A \begin{bmatrix} S_M & 0 \\ M_E & 1 \\ L_A & 2 \end{bmatrix} & \text{(A.15)} \\
C_{OR} = \text{Yes, No} \begin{bmatrix} \text{Yes} & 0 \\ \text{No} & 1 \end{bmatrix} & \text{(A.16)}
\end{cases}
$$

Table. Nomenclature

(A.1)	User selection range	(A.9)	Number of units
(A.2)	Location of the city	(A.10)	Number of rooms
(A.3)	Width of the ground	(A.11)	Number of population
(A.4)	Length of the ground	(A.12)	Parking place
(A.5)	Land type	(A.13)	Bedroom location
(A.6)	Building type	(A.14)	Kitchen location
(A.7)	Window type	(A.15)	Kitchen model
(A.8)	Number of floors	(A.16)	Bathroom model

$$U_{IIV} \begin{Bmatrix} 0\ (A.1) \\ 1\ (A.2) \\ 0\ (A.3) \\ 1\ (A.4) \\ 0\ (A.5) \\ 0\ (A.6) \\ 1\ (A.7) \\ 0\ (A.8) \\ 0\ (A.9) \\ 2\ (A.10) \\ 1\ (A.11) \\ 1\ (A.12) \\ 1\ (A.13) \\ 2\ (A.14) \\ 2\ (A.15) \\ 1\ (A.16) \end{Bmatrix}_{V} = \begin{bmatrix} 0, 1, 0, 1, 0, 2, 0, 0, 1, 0, 0, 2, 1, 1, 1, 2, 2, 1 \end{bmatrix}$$

$D_{IIt} = \begin{bmatrix} 0, 1, 0, 1, 0, 2, 0, 0, 1, 0, 0, 2, 1, 1, 1, 2, 2, 1 \end{bmatrix}$ (Vector plan information in the database)

$U_{IIV} = \begin{bmatrix} 0, 1, 0, 1, 0, 2, 0, 0, 1, 0, 0, 2, 1, 1, 1, 2, 2, 1 \end{bmatrix}$ (Vector plan information from user input data)

$$d(X.Y) = \sqrt{\sum_{i=1}^{\vec{n}} d(xi - yi)} = d(X.Y) = \sum_{i=1}^{\vec{p}} |xi - yi|$$

D_{IIt} (0, 1, 0, 1, 0, 2, 0, 0, 1, 0, 0, 2, 1, 1, 1, 2, 2, 1) and U_{IIV} (0, 1, 0, 1, 0, 2, 0, 0, 1, 0, 0, 2, 1, 1, 1, 2, 2, 1).

$$d(\vec{u},\vec{v}) = \|\vec{u} - \vec{v}\|$$

$$\sqrt{(0-0)^2 + (1-1)^2 + (0-0)^2 + (1-1)^2 + (0-0)^2 + (2-2)^2 + (0-0)^2 + (0-0)^2 + (1-1)^2 + (0-0)^2}$$

$$+(0-0)^2 + (2-2)^2 + (1-1)^2 + (1-1)^2 + (1-1)^2 + (2-2)^2 + (2-2)^2 + (1-1)^2 =$$

$$d(\vec{u},\vec{v}) = \|\vec{u} - \vec{v}\| = \sqrt{0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0}$$

$$d(\vec{u},\vec{v}) = \|\vec{u} - \vec{v}\| = \sqrt{0}$$

$d(\vec{u},\vec{v}) = \|\vec{u} - \vec{v}\| = 0$ (No differences in input data and learning algorithm data)

A.2. CONNECTIVITY/ADJACENCY

Machine learning genetic floor plan generation (MLGAFPG)

$$\begin{cases} U_1 = (0_{X1},4_{Y1}),(0_{X3},11_{Y3}),(5_{X4},4_{Y4}),(5_{X2},11_{Y2}) \\ U_2 = (5_{X1},7_{Y1}),(5_{X3},11_{Y3}),(7_{X4},7_{Y4}),(7_{X2},11_{Y2}) \\ U_3 = (7_{X1},7_{Y1}),(7_{X3},11_{Y3}),(9_{X4},7_{Y4}),(9_{X2},11_{Y2} \\ U_4 = (9_{X1},7_{Y1}),(9_{X3},11_{Y3}),(12_{X4},7_{Y4}),(12_{X2},11_{Y2}) \\ U_5 = (12_{X1},0_{Y1}),(12_{X3},11_{Y3}),(18_{X4},0_{Y4}),(18_{X2},11_{Y2}) \\ U_6 = (5_{X1},0_{Y1}),(5_{X3},7_{Y3}),(12_{X4},0_{Y4}),(12_{X2},7_{Y2}) \\ U_7 = (0_{X1},0_{Y1}),(0_{X3},4_{Y3}),(5_{X4},0_{Y4}),(5_{X2},4_{Y2}) \end{cases}$$

$U_1 = (0_{X1},4_{Y1}),(0_{X3},11_{Y3}),(5_{X4},4_{Y4}),(5_{X2},11_{Y2})$

$$\begin{cases} m_{d1} = \frac{y2-y1}{x2-x1} = \frac{11-4}{5-0} = \frac{7}{5} \\ m_{d2} = \frac{y4-y3}{x4-x3} = \frac{4-11}{5-0} = \frac{-7}{5} \\ X^2 = A^2 + B^2 \\ X^2 = 7^2 + 5^2 \\ X^2 = 74 \\ X = \sqrt{74} = 8.60 \\ m_{d3} = \frac{y2-y3}{x2-x3} = \frac{11-11}{5-0} = 0 \\ m_{d4} = \frac{y4-y1}{x4-x1} = \frac{4-4}{5-0} = 0 \end{cases}$$

$U_2 = (5_{X1},7_{Y1}),(5_{X3},11_{Y3}),(7_{X4},7_{Y4}),(7_{X2},11_{Y2})$

$$\begin{cases} m_{d1} = \frac{y2-y1}{x2-x1} = \frac{11-7}{7-5} = \frac{4}{2} \\ m_{d2} = \frac{y4-y3}{x4-x3} = \frac{7-11}{7-5} = \frac{-4}{2} \\ X^2 = A^2 + B^2 \\ X^2 = 2^2 + 4^2 \\ X^2 = 20 \\ X = \sqrt{20} = 4.47 \\ m_{d3} = \frac{y2-y3}{x2-x3} = \frac{11-11}{7-5} = 0 \\ m_{d4} = \frac{y4-y1}{x4-x1} = \frac{7-7}{7-5} = 0 \end{cases}$$

U3 = (7X1,7 Y1),(7 X3,11 Y3),(9X4,7 Y4),(9 X2,11 Y2)

$$\begin{cases} m_{d1} = \frac{y2-y1}{x2-x1} = \frac{11-7}{9-7} = \frac{4}{2} \\ m_{d2} = \frac{y4-y3}{x4-x3} = \frac{7-11}{9-7} = \frac{-4}{2} \\ X^2 = A^2 + B^2 \\ X^2 = 2^2 + 4^2 \\ X^2 = 20 \\ X = \sqrt{20} = 4.47 \\ m_{d3} = \frac{y2-y3}{x2-x3} = \frac{11-11}{9-7} = 0 \\ m_{d4} = \frac{y4-y1}{x4-x1} = \frac{7-7}{9-7} = 0 \end{cases}$$

U4 = (9X1,7 Y1),(9 X3,11 Y3),(12X4,7 Y4),(12 X2,11 Y2)

$$\begin{cases} m_{d1}=\frac{y2-y1}{x2-x1}=\frac{11-7}{12-9}=\frac{4}{3} \\ m_{d2}=\frac{y4-y3}{x4-x3}=\frac{7-11}{12-9}=\frac{-4}{3} \\ X^2 = A^2 + B^2 \\ X^2 = 3^2 + 4^2 \\ X^2 = 25 \\ X = \sqrt{25} = 5 \\ m_{d3}=\frac{y2-y3}{x2-x3}=\frac{11-11}{12-9}=0 \\ m_{d4}=\frac{y4-y1}{x4-x1}=\frac{7-7}{12-9}=0 \end{cases}$$

$$(\{L\,.W\}.\{S1 \dots SN\}.\{E1 \dots EN\}.\{ST1 \dots STN\}.\{Le1 \dots LeN\}.\{P1 \dots PN\}.\{W1 \dots WN\})$$

$$(Lei\{R1i \dots RNi\}.Fh(i))$$

$$(R1i\{Fr1 \dots FrN\}.\{Wr1 \dots WrN\}.\{Dr1 \dots DrN\})$$

$$(S1i(x.y\{x.y.w.h\} \dots \{xN.yN.wN.hN\})a.b)$$

$$(E1i(x.y\{x.y.w.h\} \dots \{xN.yN.wN.hN\})a.b)$$

$$Np = (\sum 0\, x.y.w(\sum_{i}^{L} 0 \sum_{i}^{s}(Wr + Dr) + Ns + Ne))$$

$$OLA = \sum_{i}^{s} |(Ol(W(L)\%\, 60) - fa(Np))| = 0))$$

$$F(i) = Fpco(i) + Fpno(i) + Fpmo(i)$$

$$F(i) = <>$$

Table. Nomenclature

S	Space	Fh	Floor height
E	Elevator	Fr	boundary of spaces
W	Width of the ground	Wr	Windows
L	Length of the ground	Dr	Doors
ST	Stairs	Fri	Degrees of freedom
Le	Level	OLA	Final area
P	Parking	Ol	Occupancy level
W	Warehouse	fa	Area of spaces
Fpco	Connecting matrices of spaces	Fpno	Neighborhood matrix
Fpmo	Communication space matrix		

REFERENCE

Ahmad, A. R., Basir, O., Hassanein, K., & Imam, M. H. (2005). A hierarchical placement strategy for generating superior layout decision alternatives. *International Journal of Operations and Quantitative Management*, *11*, 261–280.

Banerjee, A., Quiroz, J. C., & Louis, S. J. (2008). A model of creative design using collaborative interactive genetic algorithms. University of Nevada, Reno, USA. https://doi.org/10.1007/978-1-4020-8728-8_21

Brotchie, J., & Linzey, M. (1971). A model for integrated building design. *Building Science*, *6*(3), 89–96. https://doi.org/10.1016/0007-3628(71)90020-X

Crasto, D., Kale, A., & Jaynes, C. (2005). The smart bookshelf: A study of camera-projector scene augmentation of an everyday environment. *In Proceedings of the 7th IEEE Workshop on Applications of Computer Vision* (WACV'05) (pp. I-8). https://doi.org/10.1109/ACVMOT.2005.116

De Silva, G., & Maher, M. L. (2000). Characterising evolutionary design case adaptation. In *Artificial Intelligence in Design*. Dordrecht: Kluwer Academic. https://doi.org/10.1007/978-1-4020-8728-8_21

Doulgerakis, A. (2007). Genetic programming + unfolding embryology in automated layout planning (Master's thesis). *Bartlett School of Graduate Studies, University College London*. https://discovery.ucl.ac.uk/id/eprint/4981

Flack, R. W. J. (2011). Evolution of architectural floor plans (Tech. Rep). *Brock University*. https://doi.org/10.1007/978-3-642-20520-0_32

Goldberg, D. E., & Rzevski, G. (1991). Genetic algorithms as a computational theory of conceptual design. In *Applications of Artificial Intelligence in Engineering VI* (pp. 3–16). https://doi.org/10.1007/978-94-011-3648-8_1

Gero, J. S., & Schnier, T. (1995). Evolving representation of design cases and their use in creative design. In *Proceedings of the 3rd International Conference on Computational Models of Creative Design*.

Harada, M., Witkin, A., & Baraff, D. (1995). Interactive physically-based manipulation of discrete/continuous models. In *Proceedings of CGIT* (pp. 199–208).

Huang, H. H., May, M. D., Huang, H. M., & Huang, Y. W. (2010). Multiple-floor facilities layout design. In *Proceedings of the 2010 IEEE International Conference on Service Operations and Logistics, and Informatics* (pp. 165–170). IEEE. https://doi.org/10.1109/SOLI.2010.5551588

Hillier, B., & Leaman, A., Stansall, P., Bedford., M. (1976). Space syntax. *Environment and Planning B: Planning and Design*, *3*(2), 147–185.

Irohara, T., & Goetschalckx, M. (2007). Decomposition solution algorithms for the multi-floor facility layout problem with elevators. In *19th International Conference on Production Research (ICPR)*, Valparaiso, Chile, July 29-August 3.

Knecht, K., & Koenig, R. (2010). Generating floor plan layouts with k-d trees and evolutionary algorithms. In *Generative Art Conference* (pp. 238–253).

Keckeisen, M., Feurer, M., & Wacker, M. (2004). Tailor tools for interactive design of clothing in virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (pp. 182–185).

Liggett, R. S., & Mitchell, W. J. (1981). Optimal space planning in practice. *Computer-Aided Design*, 13(5), 277–288. https://doi.org/10.1016/0010-4485(81)90317-1

Lee, K. Y., Roh, M. I., & Jeong, H. S. (2005). An improved genetic algorithm for multi-floor facility layout problems having inner structure walls and passages. *Computers and Operations Research*, 32(4), 879–899. https://doi.org/10.1016/j.cor.2003.09.004

Liggett, R. S. (2000). Automated facilities layout: Past, present and future. *Automation in Construction*, 9(2), 197–215. https://doi.org/10.1016/S0926-5805(99)00005-9

Lee, J. Y., Kim, M. S., & Lee, J. J. (2006). Design of fuzzy controller for car parking problem using evolutionary multi-objective optimization approach. In *Proceedings of the IEEE International Symposium on Industrial Electronics* (pp. 329–334). https://doi.org/10.1109/ISIE.2006.295615

Levin, P. H. (1964). Use of graphs to decide the optimum layout of buildings. *Architect, 14*, 809–815.

Michalek, J. J., Choudhary, R., & Papalambros, P. Y. (2002). Architectural layout design optimization. *Engineering Optimization*, 34(5), 461–484. https://doi.org/10.1080/0305215021000033735

Merrell, P., Schkufza, E., & Koltun, V. (2010). Computer-generated residential building layouts. ACM (p. 181). https://doi.org/10.1145/1882261.1866203

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26, 369–395. https://doi.org/10.1007/s00158-003-0368-6

Murata, T., & Ishibuchi, H. (1995). MOGA: Multi-objective genetic algorithms. In *Evolutionary Computation, IEEE International Conference on* (Vol. 1, p. 289).

Negroponte, N. (1969). Through a humanism through machines. *AD Architectural Design*, 9. https://doi.org/10.5555/3289787.3289794

Nagy, D., Lau, D., Locke, J., & Stoddart, J. (2017). Project Discover: An application of generative design for architectural space planning.

Peng, C. (2001). *Design through digital interaction: Computing communications and collaboration on design.* Intellect Books.

Preas, B., & VanCleemput, W. (1979). Placement algorithms for arbitrarily shaped blocks. In *Proceedings*

of the Design Automation Conference (pp. 474–480). https://doi.org/10.1145/62882.62904

Poon, J., & Maher, M. L. (1997). Co-evolution and emergence in design. *Artificial Intelligence in Engineering*, 11, 319–327. https://doi.org/10.1016/S0954-1810(96)00047-7

Peponis, J., Wineman, J., Bafna, S., Rashid, M., & Kim, S. H. (1998). On the generation of linear representations of spatial configuration. *Environment and Planning B: Planning and Design*, 25, 559–576.

Reza, B. (2023). Automatic Generation of Architectural Plans with Machine Learning. *Technology|Architecture + Design*, 183-191. DOI: 10.1080/24751448. 2023.2245712.

Renner, G., & Ekrárt, A. (2003). Genetic algorithms in computer-aided design. *Computer-Aided Design*, 35, 709–726. https://doi.org/10.1007/978-1-4020-8728-8_21

Rosenman, M. A. (1997). An exploration into evolutionary models for nonroutine design. *Artificial Intelligence in Engineering*, 11, 287–293. https://doi.org/10.1016/ S0954-1810(96)00046-5

Rosenman, M. A. (1997). The generation of form using an evolutionary approach. In *Evolutionary Algorithms in Engineering Applications*. Berlin, Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-662-03423-1_4

Rodrigues, E., Rodrigues Gaspar, A., & Gomes, Á. (2013). An approach to the multi-level space allocation problem in architecture using a hybrid evolutionary technique. *Automation in Construction*. https://doi.org/10.1016/ j.autcon.2013.06.005

Schneider, S., Fischer, J., & König, R. (2011). Rethinking automated layout design: Developing a creative evolutionary design method for the layout problems in architecture and urban design. *Design Computing and Cognition* (pp. 367–386).

Turner, A., Doxa, M., O'Sullivan, D., & Penn, A. (2001). From isovists to visibility graphs: A methodology for the analysis of architectural space. *Environment and Planning B: Planning and Design*, 28, 103–121. https://doi.org/10.1068/b2684

Thakur, M. K., & Kumari, M. (2010). Architectural layout planning using genetic algorithms. *Proceedings of the IEEE International Conference on Computer Science and Information Technology*. https://doi.org/10.1109/ ICCSIT.2010.5565165

Umetani, N., Igarashi, T., & Mitra, N. J. (2012). Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics*, 31(4), 86:1–86:11.

Verma, M., & Thakur, M. K. (2010). Architectural space planning using genetic algorithm. In *Proceedings of the 2nd International Conference on Computer and Automation Engineering* (ICCAE) (Vol. 2, pp. 268–275). https://doi.org/10.1109/ICCAE.2010.5451497

Zimmermann, G. (2005). From floor plan drafting to building simulation—An efficient software-supported process. In Beausoleil-Morrison, I. & Bernier, M. (Eds.), *International IBPSA Conference Building Simulation* (pp. 1441–1448).

Wilson, P. (1991). *Computer-supported cooperative work: An introduction. Dordrecht*: Kluwer Academic.

AUTHOR (S) BIOSKETCHES

R. Babakhani., *Department of Architecture, Science and Research Branch, Islamic Azad University, Tehran, Iran*
Email: *reza.babakhani@srbiau.ac.ir*

M. Safarnejad., *Department of Architecture, Science and Research Branch, Islamic Azad University, Tehran, Iran*
Email: *Mahsa.safarnejad@gmail.com*

HOW TO CITE THIS ARTICLE

Babakhani, R,. Safarnejad, M. (2024). Automated Floor Plan Generation by a Combination of Evolutionary Algorithm (Genetics) and Machine Learning (K-Nearest Neighbors and K-Means Clustering). *Int. J. Architect. Eng. Urban Plan*, 34(4): 1-23, https://dx.doi.org/ijaup.688.

URL: http://ijaup.iust.ac.ir