Collaborating with Agents in Architectural Design and Decision-Making Process: Top-Down and Bottom-Up Case Studies Using Reinforcement Learning

Ozan Yetkin[1](✉), Elif Surer[2], and Arzu Gönenç Sorguç[1]

[1] Department of Architecture, Graduate School of Natural and Applied Sciences, Middle East Technical University, 06800 Ankara, Turkey
oyetkin@metu.edu.tr

[2] Department of Modeling and Simulation, Graduate School of Informatics, Middle East Technical University, 06800 Ankara, Turkey

Abstract. This study focuses on how architectural designers, engineers, and academics can collaborate with computational intelligent agents in a design and decision-making process, which is a great challenge. Focusing on this idea, a novel approach is presented where designers can use intelligent agents to their advantage for exploring possibilities via data generation and data processing. The problem of collaboration is presented in two distinct approaches—top-down and bottom-up. In the top-down approach, a case is selected where the designer intends to solve a housing design problem starting from meeting the general requirements of total area and distribution of housing unit types. In the bottom-up approach, a case is selected where the designer plans for the very same problem by meeting the specific requirements of room area and relations. Both cases are based on a reinforcement learning (RL) approach in which the user is allowed to collaborate with the RL algorithm, and results are compared both with widely used algorithms for similar problems (genetic algorithms) and ground-truth (deterministic solutions by designer). Compared results of top-down and bottom-up approaches have shown that the reinforcement learning approach can be used as an intelligent data system to explore design space to find an optimal set of solutions within the objective space. Finally, both approaches are discussed from a broader perspective of how designers, engineers, and academics can collaborate with agents throughout the design and decision-making processes.

Keywords: Data Processing · Intelligent Decision Support · Intelligent Data Systems · Computational Design · Reinforcement Learning

1 Introduction

Formulating a problem in which a designer, engineer, or academic can collaborate with an intelligent agent throughout the design and decision-making process is a great challenge since there is still no end-to-end solution or algorithm approaching a design problem to provide a creative solution [1]. It is observed that most of the methods aiming to expand the solution space to provide freedom in design are probabilistic approaches [2, 3]. Nevertheless, studies that use probabilistic methods have limitations on narrowing the expanded solution space down to a set of optimal solutions. It is observed that methods aiming to narrow down the solution space to incorporate design knowledge are mainly formulated with deterministic approaches [4, 5]. Here, it should be noted that studies that use deterministic methods have restrictions on expanding the solution space to enable designers to explore a variety of different alternatives. To verify this gap, a survey is conducted in the game artificial intelligence (AI) domain, and it is observed that there are approaches that use data augmentation to balance the trade-off between exploration and exploitation using reinforcement learning (RL). The RL methods are formulated with the environment, agent, and interpreter triad—a common framework in reinforcement learning research. Some of the game AI studies in the RL literature mainly attack the exploration/exploitation problem (i.e., exploring new solutions versus rigorously following an already feasible solution) with a focus on procedural content generation. For example, in Khalifa et al. [6], the proposed method uses reinforcement learning to generate a level for a 2D game (Sokoban) that is playable by seeing the design problem as a sequential task and teaching agents how to take the next action so that the expected final level design quality is maximized. In another example [7], how creative machine learning techniques can be used to interpolate between actual levels from the same game (Super Mario) or even different games (Super Mario and Kid Icarus) to achieve a newly generated level is analyzed.

When similar studies in the literature focusing on the architectural design domain are examined, it is observed that studies are clustered in two main approaches: one group focuses on exploration while giving freedom to the user, but it lacks objectives to satisfy user needs (as ArchiGAN [8] by NVIDIA Research), and the other group focuses on exploitation that satisfies the objectives that user needs yet it lacks to allow freedom to the user (as Spacemaker [9] by Autodesk Research). Even though numerous studies in the architectural design domain focus on either two-dimensional plan layout generation or three-dimensional mass generation, the methods used in the machine learning domain, such as reinforcement learning, where an agent is set loose in an environment where it constantly acts and perceives and only occasionally receives feedback on its behavior in the form of rewards or punishments [10] are prone to define the state-of-the-art and remain unexplored.

This study aims to position itself between two approaches to propose a novel approach that balances the exploration-exploitation trade-off, which is still a gap in the literature. In light of the extensive literature review, the methodology is defined based on mapping reinforcement learning methods to the architectural design domain and acting as an intelligent decision support system. This study uses an approach that has been taken for an architectural design problem which is addressed in two main parts. The first one is an architectural design problem in which the designer starts with a three-dimensional mass study to primarily handle general requirements (such as total area, maximum height, and space distribution), and the second one is an architectural design problem in which the designer starts with a two-dimensional plan to primarily meet specific requirements (like room areas, partition wall locations, and space relations).

For the first case study, the top-down approach is formulated as a mass study where a designer is attempting to meet general requirements such as total floor plan area and count/distribution of plan types. This approach is acknowledged as a combinatorial optimization which consists of finding an optimal object from a finite set of objects, where the set of feasible solutions is discrete [11]—predefined sets of building plans in this case. Therefore, the proposed approach is formulated in a way that allows the designer to search for the design space (mathematical space that encapsulates all possible combinations) via intelligent computational agents to find the set of feasible combinations within the objective space (mathematical space that encapsulates all possible outcomes) using the inputs provided by the user. On the other hand, for the second case study, the bottom-up approach, the problem is formulated as a two-dimensional architectural plan layout generation where the designer is trying to meet specific requirements such as plan layouts, room areas, and spatial relations. The proposed case study is attacking the problem as a procedural layout generation given the boundary conditions, as often used within the game environments to generate different levels. Therefore, data processing and augmentation strategies used within the game design domain are harnessed to be implemented in the design domain to propose an intelligent decision support system.

2 Methods

The proposed methodology of both the top-down and bottom-up approach is mainly based on a reinforcement learning framework consisting of an environment, an agent, and an observer. For the top-down approach, defined as an architectural mass generation problem, the environment is defined as a generative building dataset consisting of available floor plans, area data, and the number of housing types; the agent is defined as a probabilistic combinator. It takes total area and distribution ratios corresponding to housing types input to select among the available set of floor plans with corresponding area data and type number data. The interpreter is defined as a combinatorial optimization algorithm that takes user input to generate rewards and states according to the environment (Fig. 1).

On the other hand, the bottom-up approach is basically a plan layout generation problem; the agent is defined as vectors responsible for determining the size and the geometry of the rooms, the environment is defined as boundary conditions of a building, and the interpreter is defined as the difference between the boundary conditions and the generated room. The proposed framework for the top-down approach is constructed to take the user input data consisting of the desired total area of the building and desired type distribution ratios to try different combinations of housing plans using the floor area and type numbers data iteratively.

This iterative process is projected to converge in order to satisfy both the intended area and type ratios given by the user. On the other hand, to simulate the framework of the bottom-up approach, the OpenAI Gym framework [12] is selected to be used as it has a simple source code that can be easily modified to be used in this context. Therefore, the OpenAI framework was adapted to combine housing plan layouts with different topologies and used as a main framework for generating new variations (Fig. 2).

The dataset that is used for the top-down approach is composed of 70 different three-dimensional (3D) housing plan models that are designed by the authors since available

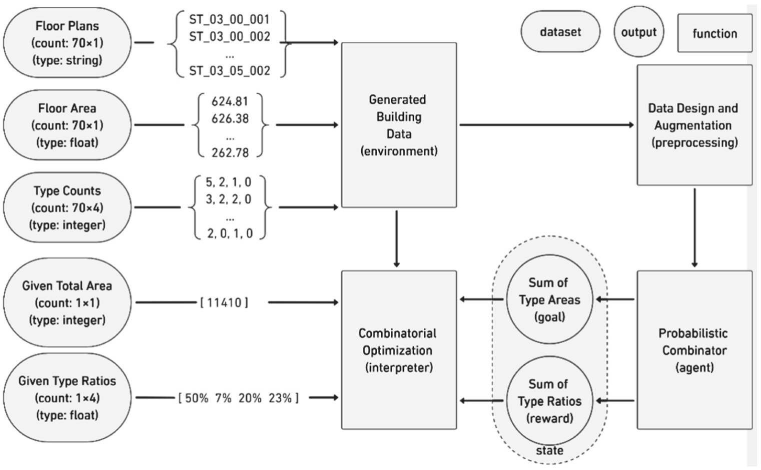
Floor Plans (count: 70×1) (type: string)
ST_03_00_001
ST_03_00_002
...
ST_03_05_002
Floor Area (count: 70×1) (type: float)
624.81
626.38
...
262.78
Type Counts (count: 70×4) (type: integer)
5, 2, 1, 0
3, 2, 2, 0
...
2, 0, 1, 0
Given Total Area (count: 1×1) (type: integer)
[11410]
Given Type Ratios (count: 1×4) (type: float)
[50% 7% 20% 23%]
Generated Building Data (environment)
Combinatorial Optimization (interpreter)
dataset
output
function
Data Design and Augmentation (preprocessing)
Probabilistic Combinator (agent)
Sum of Type Areas (goal)
Sum of Type Ratios (reward)
state

Fig. 1. The RL framework consists of an environment, agent, and interpreter.

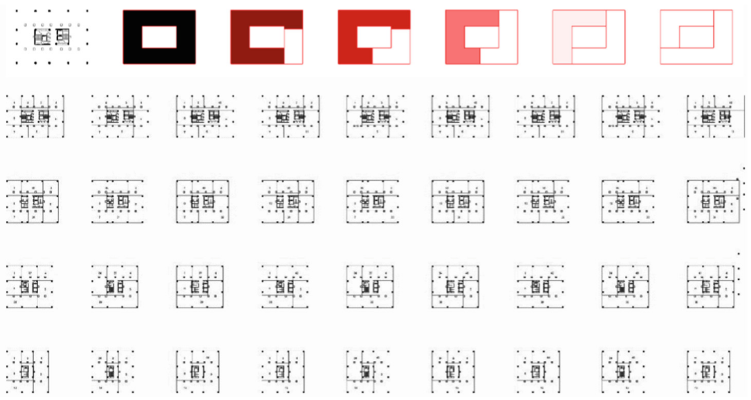

Fig. 2. Different plan layouts are used and combined for the top-down approach.

data for the architectural design domain is still lacking. Since the proposed machine learning method will be reinforcement learning and combinatorial optimization, related features/labels for each 3D housing plan were calculated using the Building Information Modeling (BIM) software and added to the dataset in correspondence. Features/labels that are used as input to calculate and satisfy the objective function consisting of the area of each housing plan and the number of housing types (i.e., 1+1, 2+1, 3+1, 4+1) in each plan. These features/labels are proposed to be used to calculate and satisfy the total area and ratio/distribution of housing types of the total building that the user demands. On the other hand, the data used for the bottom-up approach is derived from the GuessingGame-v0 environment of OpenAI, which aims to guess a number with a 1% deviation rate from the given range using 200-time steps (iterations), and it is modified in a way to adapt architectural plan layout generation for the predefined topology and the intended floor area of each room given by the user (Fig. 3). For the environment, the discrete action space is defined by: no guess submitted yet (0), guess is lower than the target (1), guess is equal to the target (2), guess is higher than the target (3), kept as the same. Yet, rewards changed to 0 if the agent's guess is outside of 5% of the target range and 1 if the agent's guess is inside 5% of the target range; since 1% of the original GuessingGame would increase the computational cost.

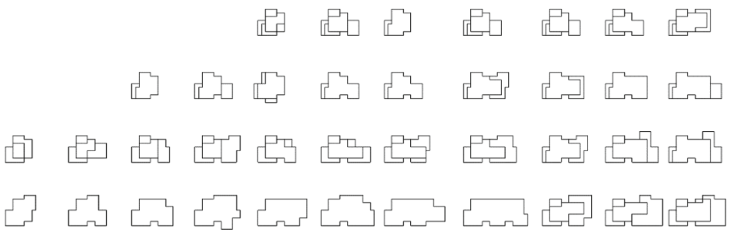

Fig. 3. Predefined housing plan layouts to be used in the bottom-up approach.

The implementation of the top-down approach is based on the reinforcement learning algorithms proposed by Bello et al. [13] for the famous 'traveling salesman problem'. The algorithm uses the TensorFlow framework, which is known to be one of the most robust frameworks within the machine learning domain. Yet, since it is implemented specifically on the traveling salesman problem, it requires further examination and adaptation to be implemented in the proposed problem. Accordingly, an alternative and simpler algorithm is employed, which is also used as a benchmark for the study [13]. The implemented algorithm is a knapsack solver, which is part of OR-Tools developed by Google AI. Even though the already implemented algorithm is not exactly a machine learning model but rather a more traditional artificial intelligence algorithm, it is still implemented to better understand the problem and observe the results. On the other hand, the implementation flow of the bottom-up approach (Fig. 4) is designed to have a graphics engine that is not available in the original GuessingGame environment, so that generated results can also be observed visually. For the graphical part, PyOpenGL is used, a Python binding of the commonly used OpenGL engine, to visualize room layouts generated by the agents. As a benchmark, the ground truth, which is the expected output of the algorithm, is also visualized in the background with a different color. Using the proposed flow, initial tests were conducted with tasks starting from the easiest to the most complex ones, such as guessing the area of a single square, guessing the dimensions of a single rectangle, guessing areas of multiple rectangles with a fixed ratio, and guessing both dimensions and positions of multiple rectangles.

Following the results obtained from the initial test of the framework, it is observed that data design and data augmentation processes are necessary both to transform the proposed framework into a reinforcement learning framework and increase the performance of the converged results (Fig. 5). Therefore, data design is conducted to transform the number of housing-type data derived from BIM into rewards that the agent can use within the reinforcement learning environment. To transfer the housing type numbers data into a reward mechanism, numbers are used to calculate the distribution ratios for

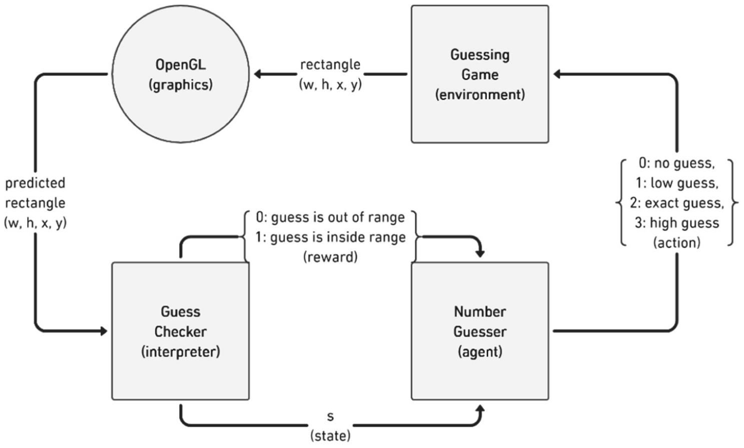

Fig. 4. Implementation workflow including graphics engine and adapted environment.

each plan, and then the absolute total differences between the ratios of each plan and the intended ratios that the user will give are calculated and scaled into a range between 0 and 10. Therefore, it is provided that the agent's probability of choosing a plan with a closer distribution of the intended housing types is aimed to increase since the interpreter will try to maximize the collected rewards. Furthermore, the dataset is augmented using a dataset multiplier calculated based on the ratio of the intended total area to the average area of each plan. This augmentation facilitates the agent's ability to utilize housing plans from the dataset multiple times as the total area increases. Such repetitive usage of suitable housing plans becomes essential, as attempting to find a viable solution by employing each unique plan in the dataset only once would be impractical.

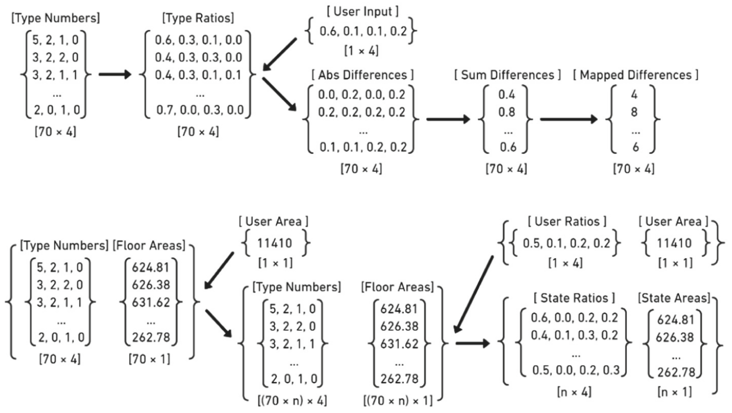

Fig. 5. Data design and data augmentation processes to increase performance.

3 Results and Discussion

Initial outcomes of the top-down approach algorithm are collected on ten different test cases. Initial results have shown a trade-off between total area and housing type ratios demanded by a user. Results show that the intended areas are approximated more precisely and yet often it fails to satisfy the type ratios requested by the user. It was observed that the size or variance of the dataset of 70 different floor plans might not be enough to find an instance that satisfies the type ratio and area objectives efficiently. However, after the integration of the data design and data augmentation process into the proposed workflow, performance is increased both in approximating ratios and the total area desired by the user.

Following the increase in the algorithm's performance, another implementation is also done to compare the proposed workflow with the common approaches in the current literature. Since the most common approach used in generative architectural design systems is genetic algorithms, a genetic algorithm written for a similar knapsack problem is implemented to the same dataset. The dataset is again subjected to the same data preprocessing (design and augmentation) to conform to the comparability of the algorithms and their results (*ceteris paribus* principle). The genetic algorithm is implemented using hyperparameters of crossover rate as 0.8, mutation rate as 0.4, solutions per population as 50, and the number of generations as 250. After the implementation, results are visualized in a line chart for area predictions and bar charts for ratio predictions to compare with the method proposed through the reinforcement learning framework (Fig. 6).

Results have shown that the proposed workflow performs better in finding combinations that have total area values closer to the intended total area by the user. Hence, the genetic algorithm approximated intended type ratios better than the reinforcement learning approach. Yet, not to compare the two methods just by intuition, commonly used regression performance metrics are calculated: mean absolute error (MAE) and root mean square error (RMSE). It is derived that the proposed approach of reinforcement learning outperforms the genetic algorithm in terms of approximating the area objective (with an MAE of 12.8 and an RMSE of 21.8 compared to an MAE of 99.6 and an RMSE of 111.7). In contrast, the genetic algorithm performs slightly better than reinforcement learning in terms of satisfying the ratio objective (with an MAE of 0.138 and RMSE of 0.105 compared to an MAE of 0.141 and RMSE of 0.115) (Table 1). The initial tests for the bottom-up approach are conducted on different levels, as mentioned in the previous section: a single square, a single rectangle, multiple rectangles with a fixed ratio, and multiple rectangles, then results are visualized. The first test conducted with a single square is executed with a single agent responsible for guessing the area of the square given by the user.

The results (Fig. 7) showed that the algorithm converged within 100 time steps when only integers were used in the action space without extensive computational resources. However, to optimize the performance for further tasks, the guessing range of the agent is modified not to include negative numbers since a negative number would not be possible for a floor area. Also, for the algorithm to work with multiple rectangles, the environment is modified to take the area value to be guessed as an input.

Further tests are conducted to implement the use of multiple intelligent agents since the dimensions (width, height) of any rectangle would affect the floor area, and therefore

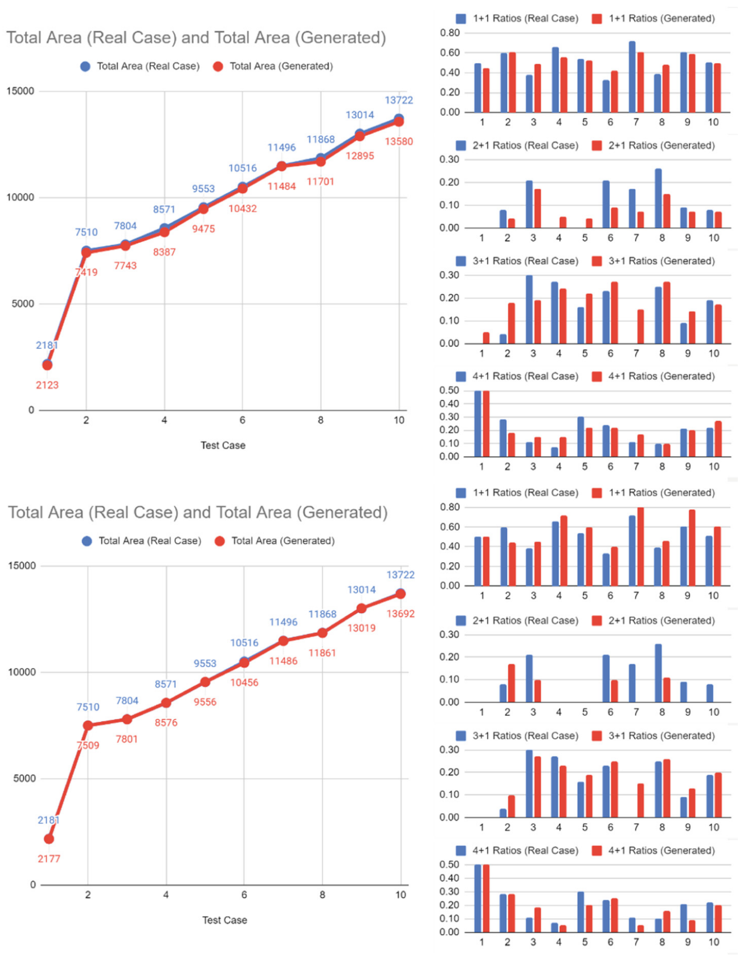

Fig. 6. Visualization of the predicted areas and ratios of reinforcement learning.

Table 1. Comparison of genetic algorithm and reinforcement learning with MAE and RMSE.

Genetic Algorithm		Reinforcement Learning	
Area MAE: 99.600	**Area RMSE:** 111.678	**Area MAE:** 12.800	**Area RMSE:** 21.758
1+1 MAE: 0.076	**1+1 RMSE:** 0.082	**1+1 MAE:** 0.105	**1+1 RMSE:** 0.108
2+1 MAE: 0.088	**2+1 RMSE:** 0.086	**2+1 MAE:** 0.133	**2+1 RMSE:** 0.128
3+1 MAE: 0.168	**3+1 RMSE:** 0.129	**3+1 MAE:** 0.098	**3+1 RMSE:** 0.089
4+1 MAE: 0.220	**4+1 RMSE:** 0.124	**4+1 MAE:** 0.230	**4+1 RMSE:** 0.137
Ratio MAE: 0.138	**Ratio RMSE:** 0.105	**Ratio MAE:** 0.141	**Ratio RMSE:** 0.115

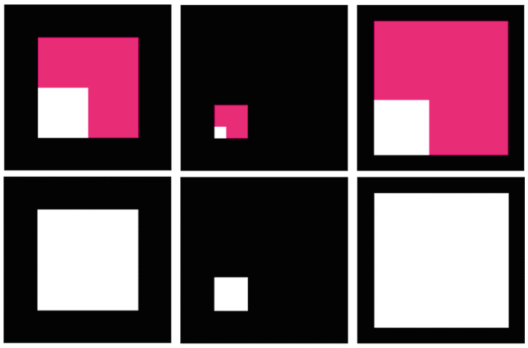

Fig. 7. Results of the first task with an agent (white) to guess an area of a square (pink).

both should be guessed by an agent to fit the common area objective. The second test, conducted after guessing the area of a square, is demonstrated on a simple rectangle where the width and height are tried to be guessed by different agents. Results showed that modifications in the previous test worked successfully in the second case, and the algorithm converged to the intended rectangle, further enhancing the decision-making process. Other tests, which are guessing areas of multiple rectangles with a fixed ratio and guessing both dimensions and positions of multiple rectangles conducted in a similar setting with the addition of more agents and algorithms, are again able to succeed in guessing the given plan layout (Fig. 8).

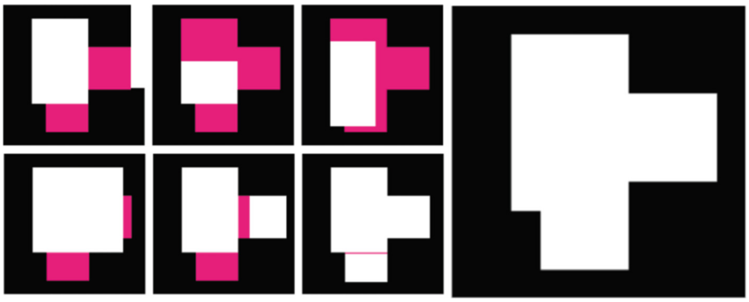

Fig. 8. Results of the complex task which has multiple agents to guess the plan geometry.

Hence, it is observed in this study that it is possible to generate different valid architectural plan layouts based on the relational and hierarchical information retrieved from the user. In addition, since the environment is set to boundary conditions of a building and a reward mechanism is proposed to control the difference between the generated layout and the feasible region of a valid architectural plan, the current study is proven to cover a reliable position between exploration and exploitation.

Both top-down and bottom-up approaches have shown that the reinforcement learning approach can be used to explore the design space to find optimal solutions within the objective space. In addition, to show its performance on extended cases that are not present in the test set, an automation algorithm is implemented to observe how successful the model is, given along with 4572 different inputs (each ratio ranging from 0.0 to 1.0 with an interval of 0.1 while providing a total of 1.0, and total area ranging from 2000 to 20000 with an interval of 1000) from the user. It is observed that the proposed method is still performing well with an MAE of 46.433 and an RMSE of 64.295 for the area objective, an MAE of 0.101, and an RMSE of 0.135 for the ratio objective. Similarly, another test is conducted on an extended dataset consisting of 8421 different housing plans in a different typology than the original dataset (Fig. 9). It is observed that the proposed framework can be adapted to different design problems.

There are also some limitations in the study regarding the following aspects. The first limitation is about the capabilities of the already implemented solver, which is declared to perform better over integers rather than floating numbers. However, since the real data consist of floating numbers for floor areas, it may cause a precision problem. The other limitation is the development process for working in BIM or other 3D environments, which requires further and interoperable development processes to visualize the result of the algorithms. Even though generating data from a correctly modeled 3D model is very easy, generating a model from predicted data back again is not. The last limitation is crucial and related to the problem itself, which is the NP-hardness (non-deterministic polynomial-time hardness) of the planning. Since the selected problem is often regarded as a very complex problem, the algorithm may not converge to a state that can be found manually with human cognition. As can be observed from the visualized results, even if the proposed method is successful in terms of satisfying the total area and the ratio distribution objectives, it fails to converge to compact results as found by human cognition by manual selection (Fig. 10).

However, the architectural design problem is often classified as an NP-hard problem that seems to be the case, considering the total process and subjective aspects. Nevertheless, it is also possible to see some parts, such as plan layout generation and combinatorial optimization of level stacking, as a straightforward engineering problem. Therefore, computational methods and state-of-the-art research in artificial intelligence can aid the architectural design process in that sense, as it has been aiding other disciplines. Hence, since the topologies of the housing plans are predetermined, the suggested approach can hardly be regarded as a creative approach that suggests a different housing plan than the ones already in the dataset. Yet as a future projection, the approach can be extended as a data augmentation and processing strategy, which is processed by agents taking action and passing the information after convergence to the other agent so that the algorithm cannot only guess the given condition with a predetermined topology but also offer different topologies having different relations between rooms.

In light of partial discussions throughout the paper, it can be observed that there is a whole lot more work to be done to achieve an AI agent that is really 'intelligent'. This idea is also clearly presented in the famous Chinese Room argument by Searle [14], which is formulated as a human in a room following a computer program to respond to Chinese characters slipped under the door where the human understands nothing, and yet, following the program for manipulating symbols and numerals, it sends appropriate strings of Chinese characters which leads those outside the door mistakenly suppose that there is a Chinese speaker in the room [15]. Therefore, this hypothesis relies on the idea that intelligence (whether it is human or machine intelligence) is more than just producing outputs that can fool the interrogator (human or machine), and the famous 'Imitation Game' proposed by Turing [16] is not enough to conclude that there is indeed an intelligent behavior. Yet, as machines can process, augment, and produce data that we can put to use as we intended, it provides great potential in collaboration for the future where humans design and machines execute.

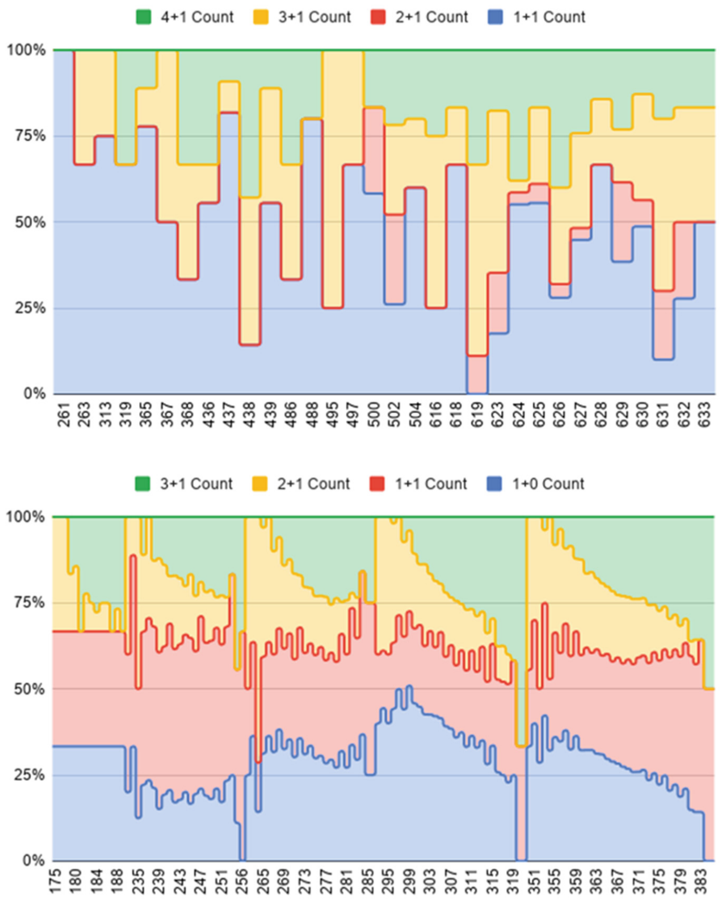

Fig. 9. Visualization of the first dataset of 70 plans and the extended dataset of 8421 plans.

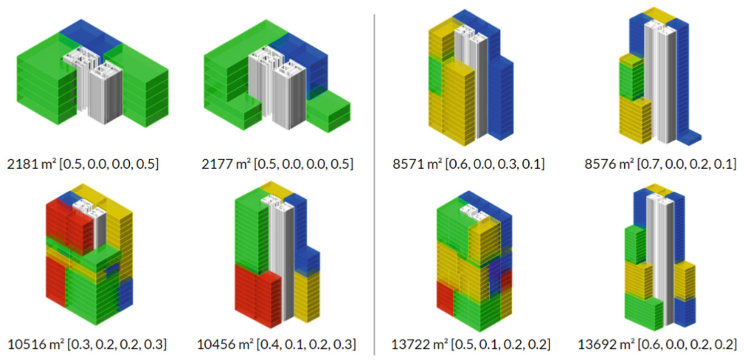
2181 m² [0.5, 0.0, 0.0, 0.5]
2177 m² [0.5, 0.0, 0.0, 0.5]
8571 m² [0.6, 0.0, 0.3, 0.1]
8576 m² [0.7, 0.0, 0.2, 0.1]
10516 m² [0.3, 0.2, 0.2, 0.3]
10456 m² [0.4, 0.1, 0.2, 0.3]
13722 m² [0.5, 0.1, 0.2, 0.2]
13692 m² [0.6, 0.0, 0.2, 0.2]

Fig. 10. 3D models of ground truth and instances generated by reinforcement learning.

4 Conclusion

In conclusion, the presented approach is formulated to demonstrate a novel and interactive way of collaborating with intelligent computational agents in the architectural design context. To demonstrate this way of human-machine interaction, top-down and bottom-up approaches are used, and results of generating 3D mass studies and 2D plan layouts are presented. The comparison results of RL algorithms with genetic algorithms and ground-truth solutions have shown that it is possible both to process and augment the domain-specific data with strategies from another domain which is game design and to use RL agents as a collaboration tool to explore possibilities in the design space. Therefore, it shows a great potential to enhance the decision-making processes in design, as well as other disciplines. Nevertheless, it is difficult to draw a line where creativity starts as elaborated at the end of the results and discussions section. Represented case studies aimed to attack the problem piecewise instead of holistically handling the architectural design problem in one shot. Hence, it should be noted that this approach is not presented to generate either 2D plan schemes or 3D mass studies but rather presented as a new model to combine creative aspects of human designers with the exploration-exploitation capabilities of computational agents. Therefore, it is projected that it has the potential to enhance the capabilities of a designer as a prosthesis of a creative mind in the future and to further expand the use of intelligent computational agents as intelligent decision support systems in various domains such as medical, educational, and societal challenges.

References

1. Dennett, D.C.: The Intentional Stance. MIT Press, Cambridge (1987)
2. Brown, N.C., Mueller, C.T.: Automated performance-based design space simplification for parametric structural design. In: Proceedings of IASS Annual Symposia, vol. 2017, no. 15, pp. 1–10. International Association for Shell and Spatial Structures (IASS) (2017)
3. Nagy, D., Villaggi, L., Zhao, D., Benjamin, D.: Beyond heuristics: a novel design space model for generative space planning in architecture. In: Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), pp. 436–445 (2017)

4. Pan, W., Turrin, M., Louter, C., Sariyildiz, S., Sun, Y.: Integrating multi-functional space and long-span structure in the early design stage of indoor sports arenas by using parametric modelling and multi-objective optimization. J. Build. Eng. **22**, 464–485 (2019)
5. Tsanas, A., Xifara, A.: Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. Energy Build. **49**, 560–567 (2012)
6. Khalifa, A., Bontrager, P., Earle, S., Togelius, J.: PCGRL: procedural content generation via reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 16, no. 1, pp. 95–101 (2020)
7. Sarkar, A., Cooper, S.: Towards game design via creative machine learning (GDCML). In: 2020 IEEE Conference on Games (CoG), pp. 744–751. IEEE (2020)
8. ArchiGAN: a Generative Stack for Apartment Building Design. https://developer.nvidia.com/blog/archigan-generative-stack-apartment-building-design/. Accessed 18 Apr 2023
9. Spacemaker Software. https://www.autodesk.com/products/spacemaker/overview. Accessed 18 Apr 2023
10. Bringsjord, S., Govindarajulu, N.S.: Artificial Intelligence." Stanford Encyclopedia of Philosophy. Stanford University (2018). https://plato.stanford.edu/entries/artificial-intelligence/
11. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency, vol. 24. Springer, Berlin (2003)
12. Brockman, G., et al.: OpenAI Gym. arXiv preprint arXiv:1606.01540 (2016)
13. Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940 (2016)
14. Searle, J.R.: Minds, brains, and programs. Behav. Brain Sci. **3**(3), 417–424 (1980)
15. Cole, D.: The Chinese Room Argument. Stanford Encyclopedia of Philosophy. Stanford University (2020). https://plato.stanford.edu/entries/chinese-room/
16. Turing, A.M.: Computing machinery and intelligence. Mind **59**(236), 433–460 (1950)