Self-learning Agents for Spatial Synthesis

Pedro Veloso and Ramesh Krishnamurti

Abstract

Over the last decades, a vast repertoire of computational methods has been employed for the synthesis of spatial configuration. Many of these techniques, such as the long-standing black-box optimization or the recent generative adversarial networks, enable a quick exploration of the design solutions based on destructive operations, but encapsulate the generative process, promoting disruptive turn-taking between computer and designer. In contrast, techniques based on agents naturally provide partial design information and enable fine-grained interaction. However, existing agent-based models originate from non-architectural problems, so it is not straightforward to adapt them for spatial design. To address this gap, we present a method to create custom spatial agents that can satisfy architectural requirements. While the method can be adapted to a diversity of representations and goals, we focus on a proof of concept where agents control spatial partitions (represented as polyominoes with no holes) and interact in an environment represented as a grid. The agents learn how to satisfy its individual (shape, area, etc.) and collective goals (adjacency) using multi-agent deep reinforcement learning. In this paper, we focus on the formulation of the environment, agents, and goals and present simulations of trained agents to illustrate possible variations.

Keywords

Space planning • Interactive generative systems • Multi-agent deep reinforcement learning

P. Veloso (✉) · R. Krishnamurti
Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: pedroveloso13@gmail.com

R. Krishnamurti
e-mail: ramesh@andrew.cmu.edu

1 Introduction

Computational generative systems are constructs that, given a proper input of information and a program, can automatically synthesize design solutions. Early computational generative systems were developed to automate the generation of architectural layout, which at the time was called spatial synthesis or space planning [1, 2]. In this context, generative systems were largely influenced by two fields concerned with problem-solving and decision-making—operations research and symbolic artificial intelligence—and their respective methods: optimization and search [1, 3–5].

Following the incorporation of black-box methods, which enabled the solution of general design problems with custom objectives and constraints, optimization became the dominant category for generative systems. For example, genetic algorithms became a widespread technique in the recent debate on generative systems [6–12] and designers today have access to many black-box optimization algorithms encapsulated in plug-ins for CAD software. In this context, the combination of parametric modeling and optimization to explore design alternatives became a mainstream generative workflow [13].

However, the availability of recent computational techniques, such as agent-based modeling and deep learning, provides an opportunity to research and investigate novel approaches to generative systems in order to support different aspects of spatial exploration.

1.1 Interactive Generative Workflows

In design theory, the critical engagement and circular reasoning of designers supported by direct interaction with the design media are considered central elements of design ideation [14–19]. Particularly in the early stages of design, the exploration of design alternatives supports not only the discovery of provisory solutions but also the reformulation

S. Eloy et al. (eds.), *Formal Methods in Architecture*, Advances in Science, Technology & Innovation,
https://doi.org/10.1007/978-3-030-57509-0_24

of the design problem itself [17, 20, 21]. Interaction with the partial states of design representation during the synthesis is important for acquiring knowledge about the problem, imposing design biases, and to support intuitive behaviors such as the leap of insight [22].

However, a large part of the generative techniques relies on strong assumptions about the problem and, consequently, focuses on the process of synthesizing solutions. For example, black-box optimization uses custom strategies to navigate in the parameter space in order to converge to optimal solutions and GANs learn a function that can generate solutions like the ones existing in the dataset.

The consequence of the focus on the final solution is that these techniques often adopt destructive procedures and promote a human-computer interaction based on disruptive turn-taking—i.e., the designer changes the input, then the algorithm executes the synthesis and displays the results as the output. Even in the cases where this is alleviated by some form of interface that partially executes the algorithm, the temporal domain and the repertoire of actions available for the designer are still distinct from the ones of generative procedures.

In this work, we switch the focus of generative systems from automatic generation of design solution to the exploration of design behavior during the generation of partial solutions. The motivation is to incentivize important elements of early and open-ended design exploration by experts, such as the understanding of the consequences of actions over time and the visualization of partial design representations in formation. We intend to enable the designer and the generator to act together on the same design representation over time, which is intrinsic to modeling paradigms that build the solution step-by-step, such as rule-based and agent-based models.

In this research, we focus on agent-based models, because they enable the construction of different parts of the representation in parallel, which results in partial states with more information and, consequently, more opportunities for direct and fine-grained design interactions.

1.2 Multi-agent Space Planning

It is important to be accurate with respect to the term 'agent.' In AI, the term agent is used generally to refer to a construct that, immersed in an environment, uses its program to map percept sequences to actions, in order to solve a certain task rationally [23]. Depending on the task environment, multiple interactive agents can be designed to cooperate, coordinate, and negotiate to achieve a certain goal, forming a multi-agent system (MAS).

Another important referent is agent-based modeling (ABM), where an agent is a unit of representation in the computational modeling of complex systems. Multiple computational agents map percepts to actions, interacting with each other in a shared environment and developing patterns or behaviors that are not necessarily predictable from the perspective of the individual [24]. ABM relies on simulation and fine-grained interactions between agent, environment, and the user to represent complex processes that unfold over time.

We coined the term multi-agent space planning [25] to designate the generative systems based on the simulation of agents that decide how space should be shaped, occupied, or partitioned. In this context, an agent is a spatial entity with local control, interweaving individual perception and action to shape a specific spatial unit. The 'environment' comprehends elements of the space that are independent of the agents. It mediates the relation between the agents (agent-agent interaction), provides global information for the agents (agent-environment interaction), and potentially acts as a game board for designers (meta agent-agent interaction). During a simulation, the agents receive signals from the environment, neighboring agents, or even the designer and must make decisions in order to change the space accordingly.

1.3 Gap in Multi-agent Space Planning

Most of existing agent-based models for spatial synthesis rely on expert knowledge in the form of heuristics or on the adaption of existing models to define how the agent should interact in the environment. This repertoire of methods for interactive spatial synthesis includes swarm algorithms (flocking and pheromone navigation), cellular automata, reaction-diffusion, and physics simulation [25, 26].

Cellular automata (CA) enable the emergence of patterns and have been successfully applied to conceptual form generation, urban morphology [27], and even to building design [28], but the cell-based computation imposes strict restrictions for the satisfaction of architectural requirements. Physics simulation is a more general technique that conciliates simple control with intuitive interaction and can be adapted for different spatial problems and objectives. However, physics-based agents are reactive agents that approximately follow laws of physics. They do not have any sophisticated policy to manage spatial conflicts. Bio-inspired models such as swarm algorithms simulate exogenous phenomena (social navigation) in which the units can move and interact in space. While they enable the incorporation of some architectural requirements, such as area or adjacency, it is usually hard to adapt them to produce conventional architectural forms or satisfy additional requirements.

An alternative to conciliate multiple architectural goals is to hybridize agent-based models with conventional methods,

such as heuristics or black-box optimization, for the initialization or the refinement of the global solution. The downside of this hybridization is that it imposes discontinuity in the generation procedure, limiting the opportunities of direct interaction with the spatial solution.

Overall, the challenge for spatial synthesis with agent-based models is twofold: to develop control strategies that incorporate specific architectural requirements and to preserve the fine granularity of the simulation.

1.4 Spatial Synthesis with Self-learning Agents

Reinforcement learning (RL) is a branch of machine learning that addresses the control problem. It is a viable alternative to support domain-specific agents and to preserve fine-grained interaction in spatial synthesis. The goal in RL is not simply computing good trajectories from a given initial state, which could be addressed by methods such as search or optimization. With a proper RL algorithm, the spatial agents can learn how to build spaces in real-time and from varied states.

In CAAD, RL has been recently applied to experiments related to intelligent adaptive building control [29], autonomous robots [30], fire egress evaluation [31], and machine feedback [32]. In the field of spatial synthesis, RL has been adopted for automatic decision-making with shape grammars [33]. Specifically for multi-agent space planning, our literature review [25] identified one experiment with learning agents. In this example [34], teams of agents try to create large clusters of building blocks on the same grid. While this experiment is motivated by recent achievements in the field of RL, in practice, it uses a natural selection mechanism to improve the policies of each competing group of agents between episodes.

Through the proof of concept in this paper, we describe an instance of our method in which custom spatial agents can learn sophisticated policies to conciliate the satisfaction of architectural requirements and the fine-grained interaction. The agents interact in an environment represented as a square grid by controlling a spatial partition. Both the representation of the space and of the agents' action space and goals are defined to take advantage of the properties of the grid. The designer provides the spatial objectives (adjacency, shape, area, etc.), and the agents learn how to behave using reinforcement learning. Particularly, we use custom RL techniques based on deep learning for the multi-agent setting —which is referred to as multi-agent deep reinforcement learning (MADRL) [35].

This proof of concept is part of the first author's larger doctoral research objective. In this paper, we focus on the formulation of the environment, agents, and goals and we will present simulations of trained agents to illustrate it. The learning algorithms and interaction with designers will be presented in future articles.

2 Development

2.1 Grids as Representation of Space and Information

In this proof of concept, we opted to use the square grid as a diagrammatic representation of architectural space (environment and agents). It is a representation that has been widely adopted in the early days of space planning [1] and in the development of recent agent-based models [34, 36–38]. The grid is a very flexible representation that can:

- Simplify spatial queries (neighborhood, adjacency, distance, etc.)
- Structure data discretely and spatially, supporting efficient computer vision, machine learning, and spatial analysis techniques.
- Provide support for an architecture based on an ensemble of discrete entities that encapsulate contextual design information.
- Be extended to 3D version (voxel grid).
- Enable control over multiple resolutions, providing different levels of complexity for design and the grain resolution of the agent.
- Discretize different geometric entities in a homogeneous representation, which supports both conventional room shapes (rectangles, L-shapes, U-shapes, T-shapes, etc.), but also unconventional and irregular shapes (see Fig. 1).

There are, however, disadvantages to using square grids in that that they are anisotropic—i.e., they bias the representation and operations by the horizontal and vertical axes. Additionally, they present a tradeoff between shape information and the design space. High resolution enables complex forms but results in a very large space for exploration. Low resolution simplifies the design space but imposes a considerable loss of shape information.

2.2 Spatial Representation of Environment and Agents

Information is stored in multidimensional arrays. For instance, the environment is an array of shape ($depth_{env}$, w, h). The width (w) and height (h) define the size of the grid, and $depth_{env}$ defines the number of layers of information. The basic layer of information of the environment contains two sets of cells: $empty_{env}$ and $obstacle_{env}$. Empty cells can be occupied by agents while obstacles cannot (see Fig. 2—left).

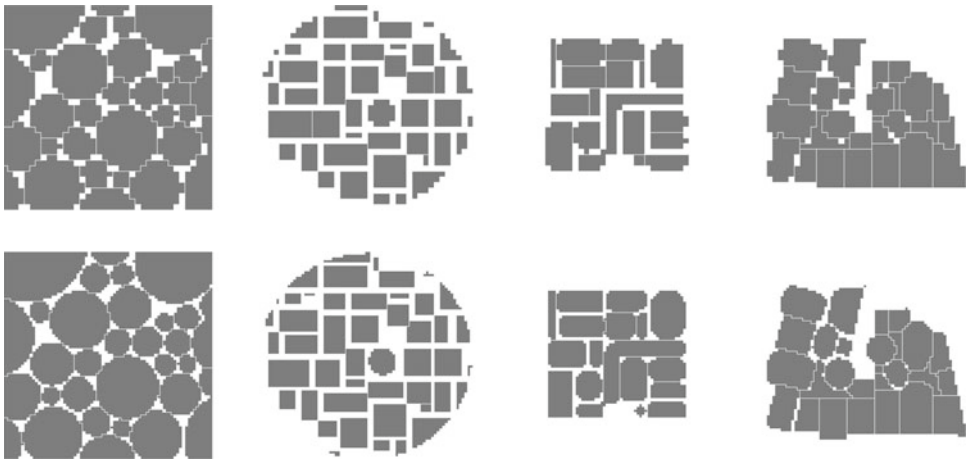


Fig. 1 Examples of real floorplans discretized in grids

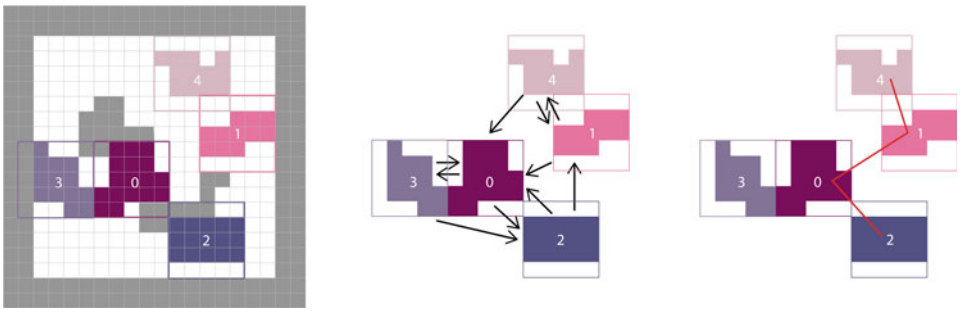


Fig. 2 Left: simplified visualization of agents on the grid with obstacles. Middle: two closest neighbors (m = 2): 0:[3, 2], 1:[4, 0], 2:[0, 1], 3:[0, 2], 4:[1, 0]. Right: adjacency goals (max k = 2): 0:[1, 2], 1:[0, 4], 2:[0], 3:[], 4:[1]

More information can be added to as additional layers, increasing the depth of the environment.

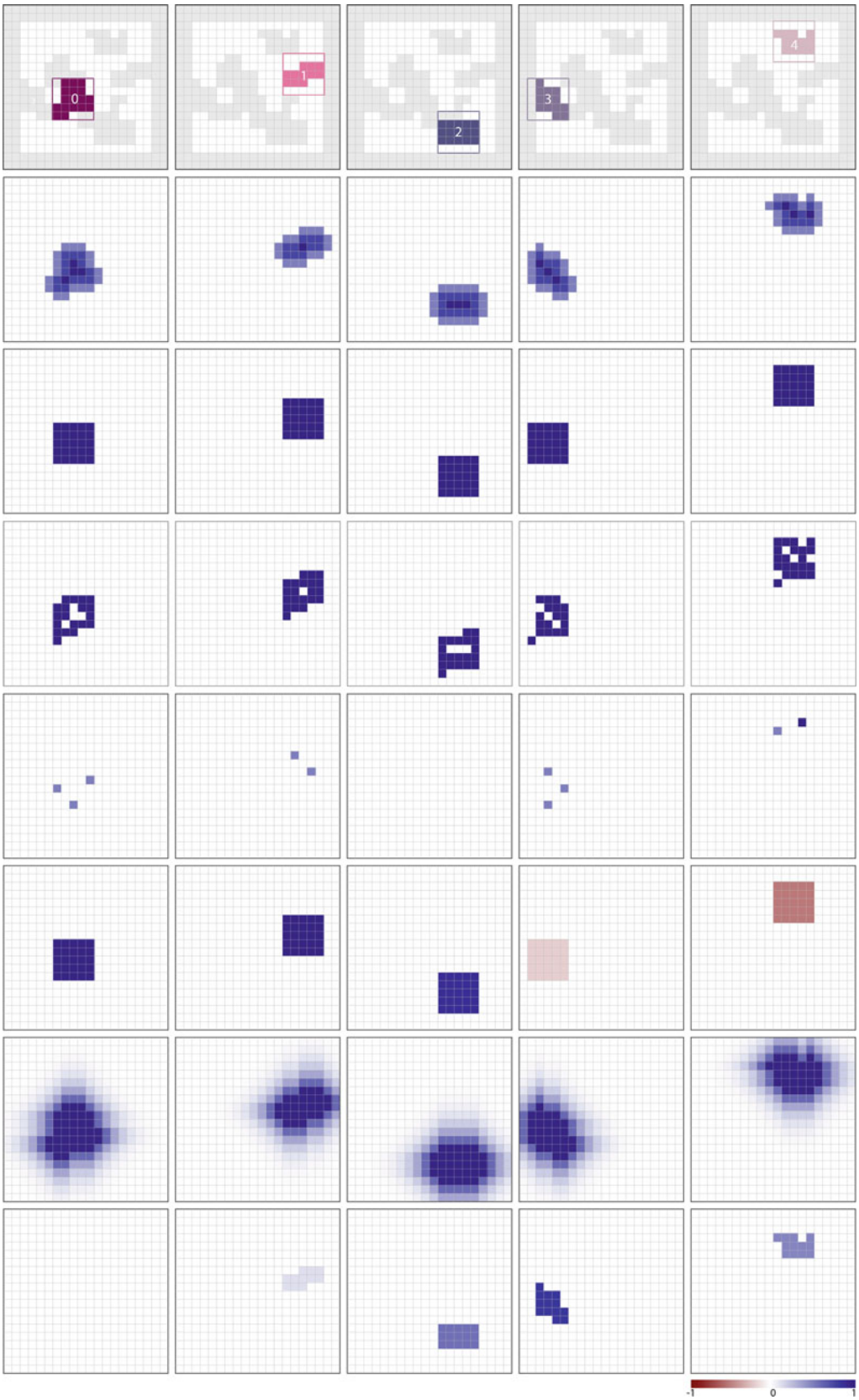
The agent is an array with shape ($depth_{agent}$, w, h), which contains multiple layers of information. Each agent represents a specific spatial partition, such as an activity or room. In this proof of concept, the spatial partition is a polyomino with no holes (PnH). A polyomino is a set of grid cells that share edges. Each PnH_{agent} induces different types of cells (see Fig. 3):

- $adjacent_{agent}$: outside cells in the von Neumann neighborhood [39] of the PnH_{agent}.
- $bridge_{agent}$: adjacent cells that if expanded, create a hole in the PnH_{agent}.
- $non\text{-}adjacent_{agent}$: cells outside of the von Neumann neighborhood of the PnH_{agent}.
- $surface_{agent}$: internal cells that if eliminated will not create a hole in or divide the PnH_{agent}.
- $structure_{agent}$: internal cells that if eliminated will create a hole or divide the PnH_{agent}.
- $mask_{agent}$: square grid placed according to the centroid of the PnH_{agent}.

For the characterization of these cells, we opted for using custom convolution filters to take advantage of parallel computation.

By avoiding the existence of holes, a PnH has certain interesting properties. For instance, it forms a polygon with orthogonal edge boundaries and avoids containment of one agent by another. Therefore, it can form custom diagrams of floorplans, comprehending organizations such as tight packing and loose packing. The shapes can approximate any

Fig. 3 Row 1: simplified representation of agent; row 2: classification of cells: structure (dark blue), surface (medium blue), and adjacent (light blue); row 3: mask with the action space of the agent; row 4: diagram with the legal actions for the current state of the agent; row 5: indication of folding cells: L-folds (light blue), U-folds (dark blue); row 6: representation of ratio area/target area: the agents have the respective (area, target area) pairs: 0: (17, 2), 1: (11, 4), 2: (15, 8), 3: (13, 16), 4: (12, 25); row 7: representation of soft adjacency values; row 8: the current utility of the agent represented in the internal cells of the agent. Color scale for rows 2–8 ▶


0
1
2
3
4
-1
0
1

polygon, such as orthogonal and non-orthogonal rectangles, or free forms such as amoeba-like shapes (see Fig. 1). The resulting polyominoes can be post-processed to depict bubble diagrams or support the development of a parametric model.

Furthermore, an agent can be interpreted as a microenvironment by converting the $\text{PnH}_{\text{agent}}$ to $\text{empty}_{\text{env}}$ and $\text{adjacent}_{\text{agent}}$ to $\text{obstacles}_{\text{env}}$. This enables hierarchical relationships and the representation of different aspects of design, such as zoning, building layout, and floorplan layout.

2.3 Action Space of Agents

The basic actions available for the agents are (1) single-cell expansion, (2) single-cell retraction, and (3) no-action.

Notice that the cells of the action grid are not always available for expansion or retraction. For each possible polyomino in this grid, there will be a set of cells that will preserve the consistency of the PnH (see Fig. 3—row 4). The set of legal cells for the expansion and retraction of an agent is defined as

- legal $\text{expansion}_{\text{agent}}$ = $\text{adjacent}_{\text{agent}}$—$\text{bridge}_{\text{agent}}$—$\text{structure}_{\text{neighbor}}$—$\text{obstacle}_{\text{env}}$—cells outside of $\text{mask}_{\text{agent}}$.
- legal $\text{retraction}_{\text{agent}}$ = $\text{surface}_{\text{agent}}$—cells outside of $\text{mask}_{\text{agent}}$.

In other words, the cells by which an agent can expand and retract are not blocked by an obstacle of the environment, preserve the PnH-ness of all the agents, and are inside the action grid ($\text{mask}_{\text{agent}}$). It is important to notice that the mask or action grid (see Fig. 3—row 3) has shape (w_{action}, h_{action}) < (w, h). It is placed on the environment based on the current centroid of the agent's PnH. Therefore, as an agent expands the PnH in a certain direction, it also moves the centroid and, consequently, the action grid.

The basic actions are building blocks that can be combined to form complex interplays such as blocking, pushing, pulling, or attraction. For example, if the agent eliminates all the cells of its PnH by retraction, it then can jump to any legal cell inside the current action grid, by a single expansion. The agent also has the option of not executing any action this turn. In the grid representation, this is depicted as the cell to the bottom left of the action grid (see Fig. 3—row 4).

2.4 Spatial Objectives

Our proof of concept enables the definition of a variety of objectives. In our first experiments, we focused on simple objectives based on neighborhood information, such as adjacency, or strictly based on local information, such as shape and area. Each of these objectives has an utility (f) and

Table 1 Utility (f) and spatial (g) representations of objectives

Adjacency

$$f^{\text{adj}}(\text{agent}) = \begin{cases} \frac{1}{|K|}\sum_{k\in K} \max\left(g^{\text{adj}}(k)\,\text{PnH}_{\text{agent}}\right) & \text{if } |K| > 0 \\ 1 & \text{otherwise} \end{cases}$$

$$g^{\text{adj}}(\text{agent}, \text{dist}_{\max}, p) = \left(\frac{\text{dist}_{\max} - \text{clip}(\text{DT}(\text{PnH}_{\text{agent}}) - 1, 0, \text{dist}_{\max})}{\text{dist}_{\max}}\right)^{p}$$

K is the set of desired adjacent agents
Clip constrains the values of the grid to a minimum and a maximum
$\text{dist}_{\max}$ is the maximum L1 distance considered for adjacency
DT is a distance transform function based on L1 distance

Area

$$f^{\text{area}}(\text{agent}) = \begin{cases} \frac{\text{area}_{\text{agent}}}{\text{target}_{\text{agent}}} & \text{if } \frac{\text{area}_{\text{agent}}}{\text{target}_{\text{agent}}} \le 1 \\ 1 - \frac{\text{area}_{\text{agent}} - \text{target}_{\text{agent}}}{\text{target}_{\text{agent}}} & \text{if } \frac{\text{area}_{\text{agent}}}{\text{target}_{\text{agent}}} \le 2 \\ 0 & \text{otherwise} \end{cases}$$

$$g^{\text{area}}(\text{agent}) = \text{mask}_{\text{agent}} \min\left(\frac{\text{area}_{\text{agent}}}{\text{target}_{\text{agent}}} - 1,\ 1\right)$$

$\text{area}_{\text{agent}}$ is the number of cells in the PnH of an agent
$\text{target}_{\text{agent}}$ is the target number of cells for the PnH of an agent

Shape (non-folding)

$$f^{\text{fold}}(\text{agent}) = \max\left(1 - \text{sum}(g^{\text{fold}}(\text{agent})), 0\right)$$

$$g^{\text{fold}}(\text{agent}, \text{fold}_{\max}) = \frac{\text{fold}_L(\text{agent})}{\text{fold}_{\max}} + \frac{2\text{fold}_U(\text{agent})}{\text{fold}_{\max}}$$

fold_L is a function that returns a grid with ones in the place of the L-folds
fold_U is a function that returns a grid with ones in the place of the U-folds
$\text{fold}_{\max}$ is a parameter that restricts the number of acceptable folds

a spatial representation (g) (see Fig. 3—rows 5–7 and Table 1).

Each goal function (f) has no restriction to any analytical form; it should return a value in the unit interval representing the performance of the agent in a given state with respect to a goal. Each spatial function (g) returns a layer of information of size (w, h) with the spatial hints for the performance of the agent with respect to the goal, which intends to ease training and enable parameterization by the user. For the current proof of concept, we have functions to indicate area, smooth adjacency, and non-folding metrics (stimulates shapes with few folds, such as rectangle, L, or U).

The utility of the agent state is based on a combination of the selected goals, such as in a weighted average or in the multiplication of the terms. In our experiment, we opted for multiplying the adjacency goal by the average of the other terms. This prevents the agent from getting stuck in local optima based on individual goals, which are generally easier to achieve.

2.5 Learning a Policy

In RL, the goal of the agent is to learn directly or indirectly a policy—i.e., a mechanism that suggests actions in response to its current state in order to maximize the performance of its future trajectory. Learning a policy relies on the interaction of the agent with the environment, which is formalized as a Markov decision process (MDP) [40]. In our proof of concept, the MDP consists of:

- a set of states (S): each state is defined by the agent's percepts, which has multiple layers with the internal and the environmental information.
- a set of actions (A): legal cells for expansion and retraction, induced by the current PnH, plus the extra cell for no-action.
- a set of rewards (R): each reward is a signal with the Δ of the utility between consecutive states s and s' that informs the performance of the agent.
- a function that maps each state-action pair to the resulting reward and state.

At each timestep, an agent can interact with the environment by observing its current state (s) (Fig. 4) and taking an action (a) among the available actions. As a result, the agent moves to the next state (s') and receives an evaluative feedback in the form of a reward (r) from the environment [40] (Fig. 5).

Our setting consists of multiple spatial agents taking actions in the environment. In this case, the architectural spaces are built in parallel, displaying partial information and

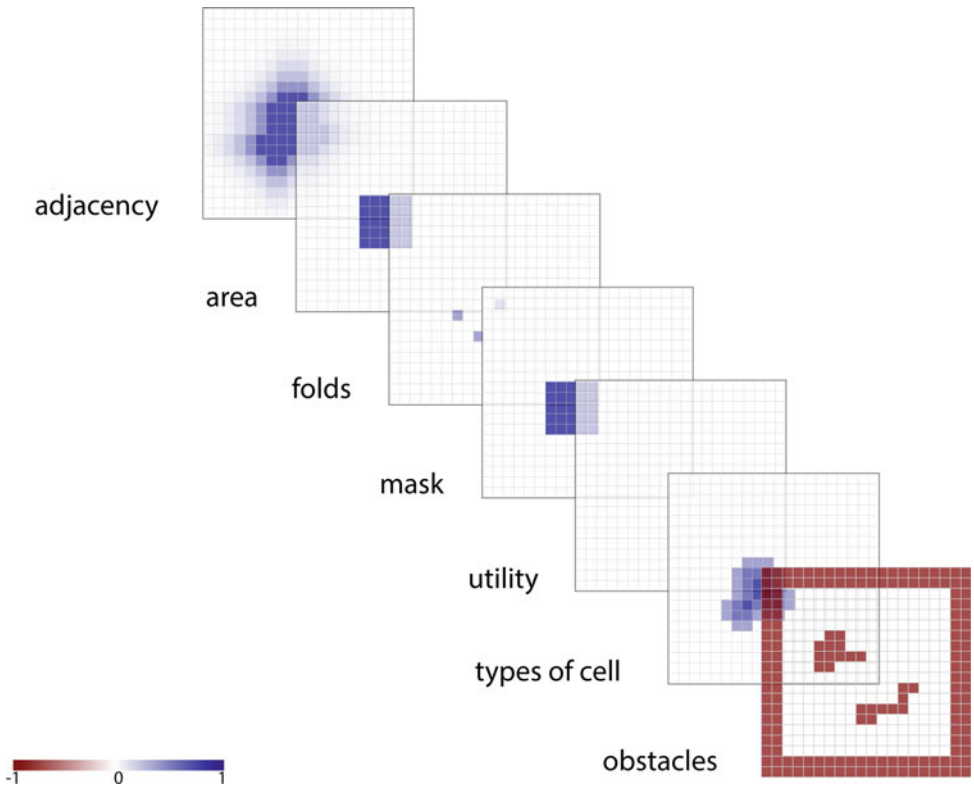


Fig. 4 Example of input state for a single agent: array with shape (7, 20, 20)

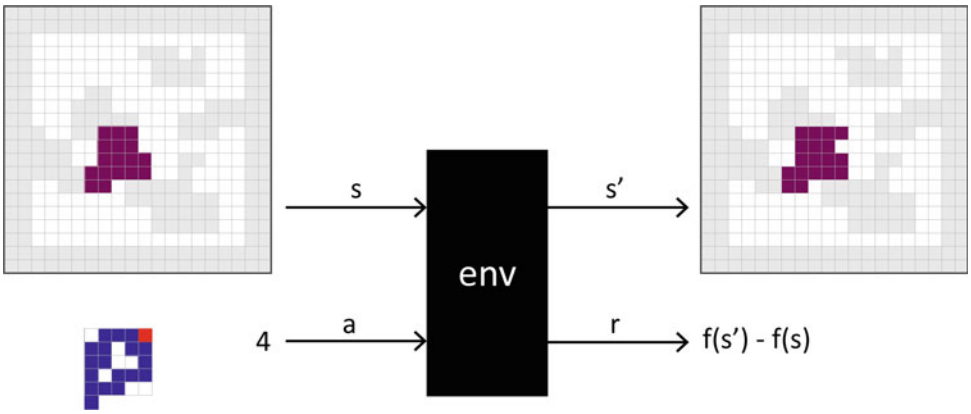


Fig. 5 Interaction between agent and environment. The representation of the states was simplified for visualization

enabling more opportunities for direct, fine-grained interactions. The agents select the actions synchronously, which requires an algorithm to order the execution and solve conflicts. The agents can share information before executing an action, but the possible combination of actions (joint-action space) grows exponentially with the number of agents.

To address this setting, we focus on multi-agent deep reinforcement learning (MADRL), the specific branch of RL that uses models, such as deep neural networks, to estimate the values of state-action pairs (Q-values), or policies for multiple agents. MADRL provides centralized and decentralized strategies to address the problem of multiple interactions and of the exponentially large joint-action space.

On one hand, interactive spatial synthesis presents some non-conventional demands for MADRL, such as varying number of agents, changes in the agent configuration over the time, etc. On the other hand, the formulation of our proof of concept tries to alleviate many of these challenges. For example, to privilege spatial relationships and to reduce the number of parameters in the neural networks, we represented the problem in square grids, which enable the use of convolutional neural networks (CNNs). In contrast to games, where the reward signals are typically sparse, we formulated explicit goals that provide a reward signal for most of the transitions.

To train the agents, we developed a custom version of the Double Deep Q-Networks algorithm (DDQN) [41], a model-free RL method that uses neural networks to estimate the function $Q(s, a)$. This function returns the Q-value—the expected performance of the agent taking action a in state s and, thereafter, following a certain policy. In the training setting, we fix the number of agents and the maximum number of adjacencies per agent. In the execution, the trained model supports a varied number of agents and the input layers provide parametric control.

3 Results

In this initial setting, we trained six agents with a maximum of three adjacencies per agent in an environment with random obstacles. The details of our learning algorithm will be described in a future publication. In this section, we will visually present some of the initial results with agents that mix the learned policy with random actions.

In Fig. 6, we display 20 timesteps of the interaction of the agents in a random environment with the same settings as in the training. The first two rows display the initial 10 timesteps, which show how the trajectory of the agents is easy to grasp visually. In the following two rows, we selected every 100th timestep to show how the behavior of the agent results in local exploration after reaching good configurations. Additional variation can be incentivized by adding more randomness to the policy.

In Fig. 7, we display another simulation with twelve agents. In addition to the fact that the agents will face situations that they were not trained for, the increase in the number of agents and connections as well as the preservation of the same board size makes it significantly more difficult for the agents to satisfy the adjacency and area requirements. Still, the agents find a good configuration and try to improve its performance with local changes, which are constrained by the lack of empty spaces.

Our proof of concept intends to verify whether our formulation and algorithm can result in an effective approximation of the state-action values (Q-values), so that, from any state, the agents can follow a policy that explores valuable configurations. The preliminary results are very promising. The agents learned an effective policy for random environments with obstacles, generalized it for a more challenging setting, and generated variations of the

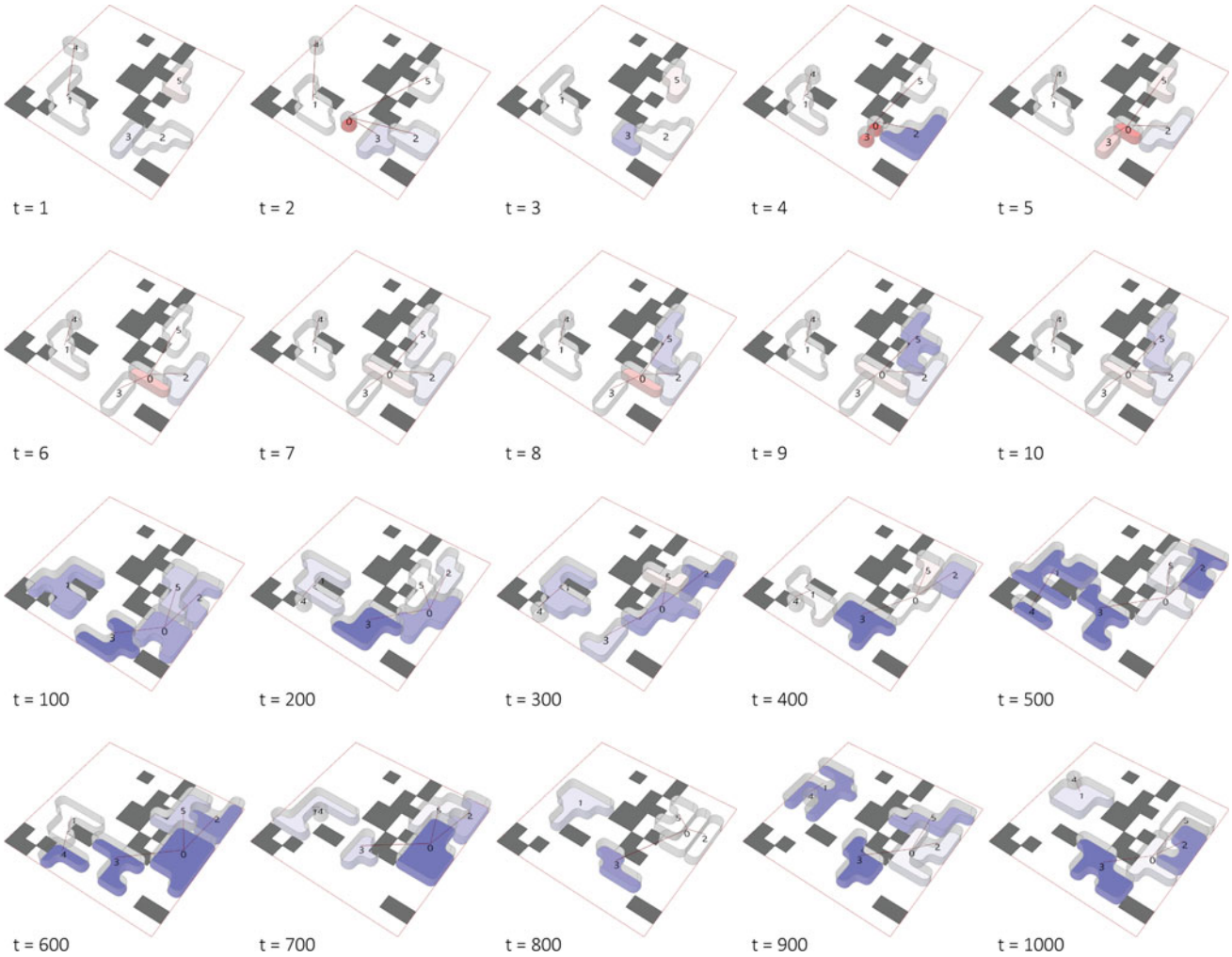


Fig. 6 Simulation of six agents. The goals for the areas are: 0: 5, 1: 7, 2: 4, 3: 3, 4: 1, 5: 5. The goals for adjacencies are: 0: [2, 3, 5], 1: [4], 2: [0], 3: [0], 4: [1], 5: [0]

arrangement in local regions of the design space. Most importantly, they can do all these things by taking increment steps in the environment.

4 Future Steps

In this paper, we have presented a novel framework for interactive space planning with self-learning agents. We describe both the formulation and our proof of concept, which demonstrates that it is possible to teach agents how to behave in large design spaces, addressing specific design goals, and preserving fine-grained interaction.

Future steps of this research will address three topics: learning, representation, and interaction. In terms of learning, we plan to investigate improvements in the learning algorithm, alternative solutions, baselines, and forms of evaluation. The learning algorithm and any improvements will be described in detail in a future publication. We plan to expand this research to the 3D setting and investigate other

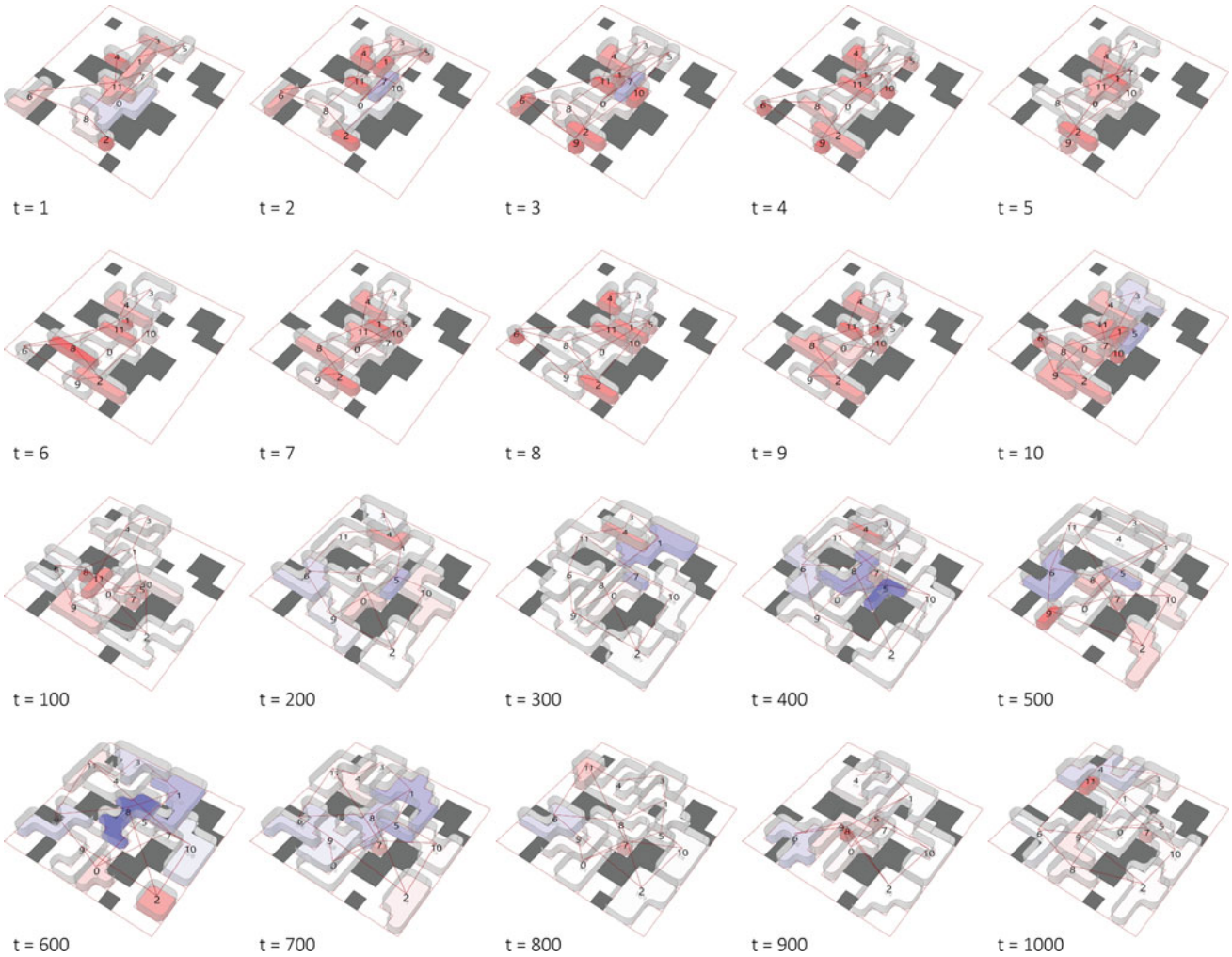


Fig. 7 Simulation of twelve agents. The goals for the areas are [0: 6, 1: 8, 2: 9, 3: 6, 4: 9, 5: 2, 6: 6, 7: 2, 8: 5, 9: 8, 10: 8, 11: 8]. The goals for adjacencies are 0: [5, 7, 9], 1: [3, 5, 8], 2: [8, 9, 10], 3: [1, 4], 4: [3, 11], 5: [0, 1, 10], 6: [8, 9, 11], 7: [0, 10, 11], 8: [1, 2, 6], 9: [0, 2, 6], 10: [2, 5, 7], 11: [4, 6, 7]

forms of design representation that can be integrated into our framework. Finally, our main motivation for developing self-learning agents is to contribute to the development of interactive generative systems that can support observer-dependent and circular processes for custom design generation. We plan to integrate the current proof of concept with a game engine and to explore specific types of interactions and strategies to navigate the design space.

Acknowledgements This research was supported in part by funding from the Carnegie Mellon University Frank-Ratchye Fund for Art @ the Frontier as well as a PhD scholarship granted by the Brazilian National Council for Scientific and Technological Development (CNPq).

References

1. Mitchell, W. (1977). *Computer-aided architectural design*. New York: Mason Charter.
2. Eastman, C. M. (1975). *Spatial Synthesis in Computer-Aided Building Design*. New York: Elsevier Science Inc.
3. Simon, H. A. (1996). The Science of Design: Creating the Artificial. In *The Sciences of the Artificial* (3rd ed.). Cambridge: The MIT Press.
4. Henrion, M. (1978). Automatic space-planning: A postmortem? In: In J. C. Latombe (Ed.), *Artificial Intelligence and Pattern Recognition in Computer Aided Design* (pp. 175–191). New York: North-Holland.
5. Liggett, R. S. (2000). Automated facilities layout: Past, present and future. *Automation in Construction, 9*(2), 197–215.
6. Fischer, T., & Herr, C. M. (2001). Teaching generative design. In: *Proceedings of the 4th Conference on Generative Art*. Milan: Politecnico di Milano.
7. Kolarevic, B. (2005). Digital Morphogenesis. In: B. Kolarevic (Ed.), *Architecture in the digital age: Design and manufacturing* (pp. 12–28). London: Taylor & Francis.
8. Kalay, Y. E. (2004). *Architecture's New Media: Principles, Theories, and Methods of Computer-Aided Design*. Cambridge: The MIT Press.
9. Oxman, R. (2006). Theory and design in the first digital age. *Design Studies, 27*(3), 229–265. https://doi.org/10.1016/j.destud.2005.11.002
10. Gorbman, Y. J., Yezioro, A., & Capeluto, I. G. (2009). Computer-Based Form Generation in Architectural Design—A Critical Review. *International Journal of Architectural Computing, 7*(4), 535–553. https://doi.org/10.1260/1478-0771.7.4.535
11. Vishal, S., & Gu, N. (2012). Towards an integrated generative design framework. *Design Studies, 33*(2), 185–207. https://doi.org/10.1016/j.destud.2011.06.001
12. Henrique, G. C., Bueno, E., Lenz, D., & Sardenberg, V. (2019). Generative Systems: Interwining Physical, Digital and Biological Processes, a case study. In: J. P. Sousa, G. C. Henriques, & J. P. Xavier (Eds.), *Architecture in the age of the 4th Industrial Revolution: Proceedings of the 37th eCAADe and 23rd SIGraDi Conference* (Vol. 1, pp. 25–34). Porto: eCAADe-SIGraDi-FAUP.
13. Nagy, D., Lau, D., Locke, J., Stoddart, J., Villaggi, L., Wang, R., Zhao, D., & Benjamin, D. (2017). Project Discover: An Application of Generative Design for Architectural Space Planning. In: M. Turrin, B. Peters, W. O'Brien, R. Stouffs, T. Dogan (Eds.), *Proceedings of Symposium on Simulation for Architecture and Urban Design* (pp. 59–66). San Diego: Society for Computer Simulation International.
14. Anderson, S. (1966). *Problem-Solving and Problem-Worrying*. Retrieved from http://web.mit.edu/soa/www/downloads/1963-69/TH_AALond-Lect_66.pdf
15. Schön, D. (1983). *The Reflective Practitioner: How Professionals Think In Action*. New York: Basic Books.
16. Corona-Martínez, A. (2003). *The Architectural Project*. (A. Corona-Martínez & M. Quantrill, Trans., M. Quantrill, Ed.). College Station: Texas A&M University Press.
17. Cross, N. (2006). *Designerly ways of knowing*. London: Springer.
18. Lawson, B. (2005). *How designers think: The design process demystified* (4th ed.). Amsterdam: Elsevier.
19. Gänshirt, C. (2007). *Tools for ideas: Introduction to architectural design*. Basel: Birkhäuser.
20. Simon, H. A. (1977). The structure of ill-structured problems. In: *Models of discovery* (pp. 304–325). Dordrecht: Springer.
21. Akin, O., Dave, B., & Pithavadian, S. (1987). Problem structuring in architectural design. *Carnegie Mellon University*. https://doi.org/10.1184/R1/6076109.v1
22. Jones, J. C. (1992). *Design Methods* (2nd ed.). New York: Wiley.
23. Russel, S. J., & Norvig, P. (2010). *Artificial intelligence: A modern approach* (3rd ed.). Upper Saddle River: Prentice Hall.
24. Wilensky, U., & Rand, W. (2015). *An Introduction to Agent-based Modeling: modeling natural, societal, and engineered complex systems with netlogo*. Cambridge: The MIT Press.
25. Veloso, P., Rhee, J., & Krishnamurti, R. (2019). Multi-agent Space Planning: A Literature Review (2008–2017). In J.-H. Lee (Ed.), *Hello, Culture!: Proceedings of 18th CAAD Futures Conference* (pp. 52–74). Daejeon, Korea.
26. Herr, C. M., & Ford, R. C. (2015). Adapting Cellular Automata as Architectural Design Tools. In *Emerging Experience in Past, Present and Future of Digital Architecture: Proceedings of the 20th CAADRIA Conference* (pp. 169–178). Daegu: Kyungpook National University.
27. Koenig, R. (2011). Generating Urban Structures: a Method for Urban Planning Supported by Multi-Agent Systems and Cellular Automata. *Przestrzen I Forma,*(16), 353–376.
28. Araghi, S. K., & Stouffs, R. (2015). Exploring cellular automata for high density residential building form generation. *Automation in Construction, 49,* 152–162.
29. Smith, S. I., & Lasch, C. (2016). Machine Learning Integration for Adaptive Building Envelopes: An Experimental Framework for Intelligent Adaptive Control. In K. Velikov, S. Manninger, M. del Campo, S. Ahlquist, & G. Thün (Eds.), *Proceedings of the 36th ACADIA Conference: Posthumans Frontiers* (pp. 98–105). Ann Arbor: ACADIA.
30. Hosmer, T., & Tigas, P. (2019). Deep Reinforcement Learning for Autonomous Robotic Tensegrity. In: *Ubiquity and Autonomy: Proceedings of the 39th ACADIA Conference* (pp. 16–29). Austin: ACADIA.
31. Jabi, W., Chatzivasileiadi, A., Wardhana, N. M., Lannon, S., & Aish, R. (2019). The synergy of non-manifold topology and reinforcement learning for fire egress. In: *Architecture in the age of the 4th Industrial Revolution: Proceedings of the 37th eCAADE Conference and of the 23rd SIGraDi Conference* (Vol. 2, pp. 85–94). Porto: eCAADe-SIGraDi.
32. Xu, T., Wang, D., Yang, M., You, X., & Huang, W. (2018). An Evolving Built Environment Prototype. In: T. Fukuda, W. Huang, P. Janssen, K. Crolla, & S. Alhadidi (Eds.), *Learning, Adapting and Prototyping: Proceedings of the 23rd CAADRIA Conference* (Vol. 2, pp. 207–215). Beijing: CAADRIA.
33. Ruiz-Montiel, M., Boned, J., Gavilanes, J., Jiménez, E., Mandow, L., & Pérez-de-la-Cruz, J.-L. (2013). Design with shape grammars and reinforcement learning. *Advanced Engineering Informatics, 27*(2), 230–245. https://doi.org/10.1016/j.aei.2012.12.004
34. Narahara, T. (2017). Collective Construction Modeling and Machine Learning: Potential for Architectural Design. In A. Fioravanti, S. Cursi, S. Elahmar, S. Gargaro, G. Loffreda, G. Novembri, & A. Trento (Eds.), *Sharing Computational Knowledge!: Proceedings of the 35th eCAADe Conference*. (pp. 593–600). Rome: eCAADe.
35. Nguyen, T. T., Nguyen, N. D., & Nahavandi, S. (2018). Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions and Applications. arXiv:1812.11794 [cs, stat]. Retrieved from http://arxiv.org/abs/1812.11794
36. Meyboom, A., Reeves, D. (2013). Stigmergic Space. In P. Beesley, M. Stacey, & O. Khan, (Eds.), *Adaptive*

Architecture: Proceedings of the 33rd ACADIA Conference (pp. 200–206). Toronto: Riverside Architectural Press.
37. Fernando, R. (2014). Space Planning and Preliminary Design Using Artificial Life. In: N. Gu, S. Watanabe, H. Erhan, M. H. Haeusler, W. Huang, & R. Sosa, (Eds.), *Rethinking Comprehensive Design: Speculative Counterculture: Proceedings of the 19th CAADRIA Conference* (pp. 657–666). Hong Kong: CAADRIA.
38. Guo, Z., & Li, B. (2017). Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system. *Frontiers of Architectural Research, 6*(1), 53–62. https://doi.org/10.1016/j.foar.2016.11.003
39. Weisstein, E. W. (n.d.). von Neumann Neighborhood. *MathWorld -A Wolfram Web Resource*. Retrieved from https://mathworld.wolfram.com/vonNeumannNeighborhood.html
40. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). Cambridge: The MIT Press.
41. van Hassel, H., Guez, A., & Silver, D. (2016). Deep Reinforcement Learning with Double Q-learning. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (pp. 2094–2100). Phoenix: AAAI.

Pedro Veloso Veloso is a computational designer, educator, and researcher interested in design exploration methods supported by different modes of artificial intelligence, such as classical AI, bio-inspired AI, agent-based modeling, and machine learning. He has completed Bachelor of Architecture and Urbanism from the University of Brasilia. He has completed Master of Architectural Design from the University of Sao Paulo. Currently, he is a PhD candidate in Computational Design at Carnegie Mellon University, investigating generative design and the development of interactive systems with learning techniques.

Ramesh Krishnamurti Krishnamurti is a full professor in the School of Architecture at Carnegie Mellon University. He has a BE (Honors) in Electrical Engineering from the University of Madras, a BA in Computer Science from the University of Canberra, and MASc and PhD in Systems Design from the University of Waterloo. His principal area of research is computational design with emphasis on the formal, semantic, and algorithmic aspects of generative construction and development of design as computation via highly coupled parallel explorations of form and description. He is perhaps best known for his work on computational problems in shape grammar theory and algorithms for spatial patterns.