Multi-agent Space Planning

A literature review (2008-2017)

Pedro Veloso, Jinmo Rhee, and Ramesh Krishnamurti

Carnegie Mellon University,
Pittsburgh, USA
{pveloso, jinmor, ramesh}@andrew.cmu.edu

Abstract. In this paper we review the research on multi-agent space planning (MASP) during the period of 2008-2017. By MASP, we refer to space planning (SP) methods based on online mobile agents that map local perceptions to actions in the environment, generating spatial representation. We group two precedents and sixteen recent MASP prototypes into three categories: (1) agents as moving spatial units, (2) agents that occupy a space, and (3) agents that partition a space. In order to compare the prototypes, we identify the occurrence of features in terms of representation, objectives, and control procedures. Upon analysis of occurrences and correlations of features in the types, we present gaps and challenges for future MASP research. We point to the limits of current systems to solve spatial conflicts and to incorporate architectural knowledge. Finally, we suggest that behavioral learning offers a promising path for robust and autonomous MASP systems in the architectural domain.

Keywords: Space planning; Agent-based modeling; Multi-agent systems; Generative systems.

1 Introduction

Designing spatial arrangements is at the core of architecture and has traditionally been solved by human-centered methods. Since the early 1960s, a main branch of CAAD research focused on automated space planning (SP), comprising methods for allocating and configuring architectural spaces based on computational data-structures and algorithms. While a large part of the space planning literature is grounded in classical Artificial Intelligence (AI) techniques and on optimization methods, an emerging branch of SP, which we refer to as Multi-agent space planning (MASP), addresses the use of agents to augment SP exploration in real-time.

However, it is important to be accurate with respect to the term 'agent'. In AI, the term is used generally as an abstraction to describe autonomous computational systems. In this context, an agent is a construct that, immersed in an environment, uses its program to map percept sequences to actions, in order to solve a certain task rationally [1]. 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 systems (MAS).

Another important reference 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 [2]. ABM may also include the user as an omniscient agent that has access to all agents, environments and can affect the behavior of the system.

MASP can have either the goal-oriented approach of MAS, in order to create agents that try to solve a specific SP task, or it can be closer to the exploratory branch of ABM in the development of models that investigate emerging spatial patterns resulting from the interaction of custom agents. Each agent is an entity with local control, interweaving individual perception and action to satisfy certain spatial objectives. This results in a simulation of agents that decide how space should be shaped and occupied.

Due to its emphasis on a continuous and interactive configuration of spatial patterns, MASP has huge potential for future decision making and design systems. However, it still has a secondary role in the literature. General SP literature review papers [3–7] have none or few examples of MASP, recent specialized literature review papers focus on mainstream techniques, such as evolutionary optimization [8] and a recent review of agent-based models in architectural research [9] addresses collaborative design systems, but does not present MASP examples for building design.

This paper addresses this gap by providing an initial categorization and a literature review of recent MASP work (2008-2017) within the scope of building design. It includes the allocation of human activities in architectural space, such as building parts on a site or rooms inside a building, using diagrams or technical representations. It comprehends agents with different complexity, from basic reflex agents that refine a pre-existing spatial arrangement to intelligent agents that can generates complete spatial patterns. It covers agent-based modeling, techniques adapted to control agents (such as physics simulation) and hybridizations of agents with conventional or manual methods. It excludes applications with cellular automata, which were already reviewed by a previous paper [10], and allocation tasks that are strictly for urban design.

Different databases and online documentations were used to look for MASP prototypes, but the main source is the CumInCAD[1]. In the next three sections, we organize the descriptions of sixteen research prototypes and two precedents according to categories that conciliate algorithmic and spatial characteristics: (1) agents as moving spatial units; (2) agents that occupy discrete spaces; and (3) agents that partition space.

After these descriptive sections, we analyze the prototypes based on their types and on the occurrence of features related to computational representation, evaluation and control procedures. The goal is to characterize the prototypes under the proposed categories and to present trends, patterns and challenges for future research.

[1] CumInCAD is a cumulative index about publications in Computer-Aided Architectural Design (CAAD), which provides access to papers of the conferences organized by the sibling CAAD associations (ACADIA, CAADRIA, eCAADe, SIGraDi, ASCAAD and CAAD futures) and other related conferences.

2 Type 1: Agents as moving spatial units

Each agent of a system of type 1 incorporates an individual representation of space, which is modified to interact with other agents and to satisfy spatial objectives.

2.1 IMAGE (IMG) [2]

System description: IMG is an interactive computer system for multi-constrained synthesis of spatial arrangements [11, 12], and an important precedent for type 1.

System algorithm: Each spatial unit is a custom cuboid that moves in the space. These aim at satisfying (multiple) user-defined objectives and constraints: adjacent, keep out, overlap, relative position, ratio, width / depth / height, x / y / z position, distance, shared wall, enclose, next, on top of, floor, above, align, and visual access. A constraint graph stores cuboid units at the main nodes and relations as intermediate labelled nodes. A Least Mean Squares Fit (LMSF) is applied sequentially to all the spaces, changing the descriptors of a unit (dimension, location and rotation) to achieve the best local improvement – i.e., minimum error regarding all the linear equations of the constraints.

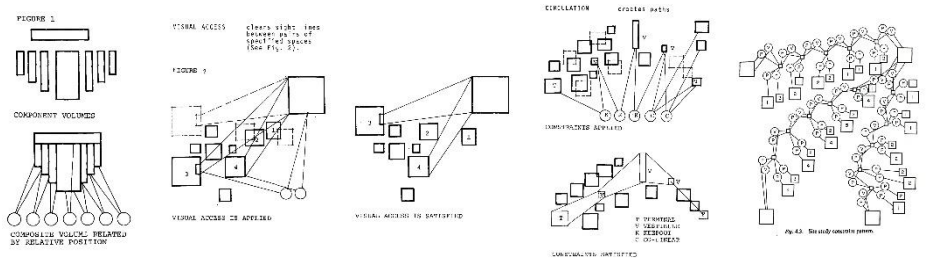


Fig. 1. IMAGE. Left: complex volumes generated by aggregation of cuboids with relative position objective; Left-center and center: visual access objective; center-right: circulation objective; right: constraint graph [11, 12].

2.2 Fuzzy Layout Planner (FLP)

System description: FLP [13] is an editor to create and modify dynamic bubble diagrams for the early-stages of design. It adopts reactive spatial units and focuses on user manipulation.

System algorithm: Each agent is a rectangular bubble, which has minimum agency and is directly manipulated by the designer. As the user changes one of its dimensions, the

[2] We refer to all the prototypes analyzed in this paper by a name and an abbreviation. In cases where the authors defined the name in the paper, we adopt it. Else, we create a simple name based on the paper title, description or characteristics of the prototype. The abbreviation of these names will be used to reference the prototypes in the tables of section 5.

bubble modifies the other dimension automatically to preserve the assigned area. A repulsion force ensures the packing of close bubbles.

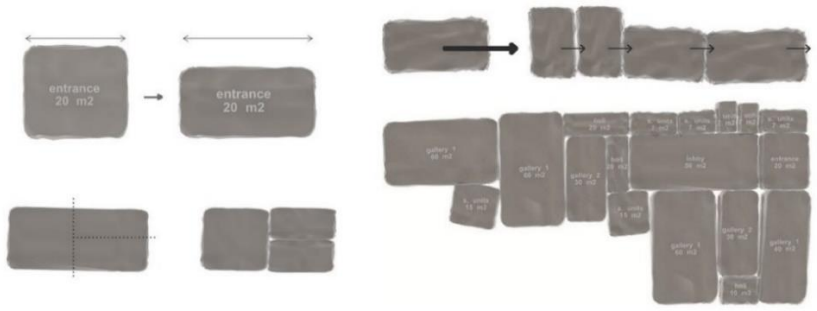


Fig. 2. Fuzzy Layout Planner. Top left: dimensions of bubble adapt to preserve area; bottom-left: dividing bubbles; top-right: repulsion forces; bottom-right: packing bubbles [13].

2.3 Sniffing Space I (SS1)

System description: SS1 [14] is MASP hybrid of type 1 and 2. It consists of a self-organizing bubble diagram combined with loosely packed rectangles [15].

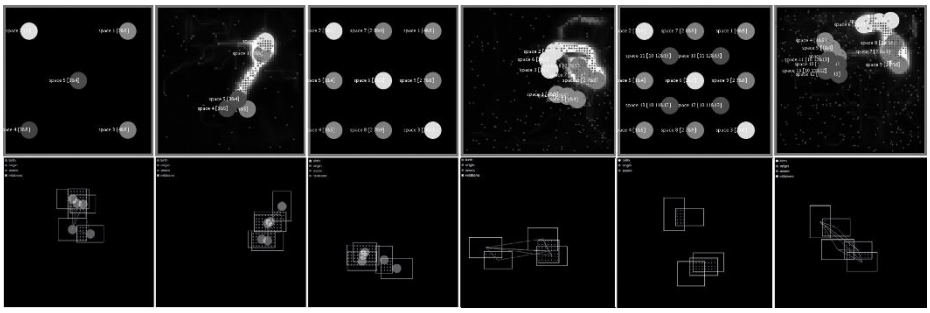


Fig. 3. Sniffing Space. Top row: spatial association generating zones [14]. Bottom row: expansion of the model with rectangular boundaries [15].

System algorithm: SS1 incorporates swarm intelligence – in contrast to the reactive agents of FLP, it uses ant foraging algorithms to define the interaction between two types of agents: (1) ant soldiers; and (2) an array of ant nests. The soldiers are produced by the nests, which are inter-connected by edges indicating association or adversity. The soldiers are points that navigate in the environment, producing and following two different pheromone trails depending on whether they are searching for associated nests or returning to their original nest. Each nest has a fixed circular body and a rectangular boundary with random dimensions x and y that satisfy its required area. The nests gradually follow visiting soldiers that are returning to associated nests, clustering with them and eventually interrupting the production of new soldiers. Additionally, they have three spatial behaviors: adapt x and y when they nestle, move away from

overlapping adversary nests, and overlie with associated nests. It results in a loose packing of rectangles, defined by the connections.

2.4 Actants (ACT)

System description: ACT [16, 17] is a self-organizing system that loosely packs polygons.

System algorithm: It combines swarm intelligence and a physics engine. It is based on dynamic units called Actants, which are represented by soft body polygons composed of internal and boundary nodes that function as sensors and actuators. The nodes maintain the consistency of the shape by attraction-repel forces. The boundary nodes emit and track identity-based pheromone, detect other nodes, and move the Actant.

The system is based on a hunter-prey dynamic between different Actants. The pheromone differences in the environment might attract or repel boundary nodes, keeping the Actant in movement. After a period, if no associated Actant is found by following the pheromone gradients, one of the boundary nodes is selected for a direct search in the local vicinity. When an Actant wants to be found by its associates, it produces pheromone. When it is evading or seeking, it blocks pheromone production, preventing the agglomeration of non-associated Actants.

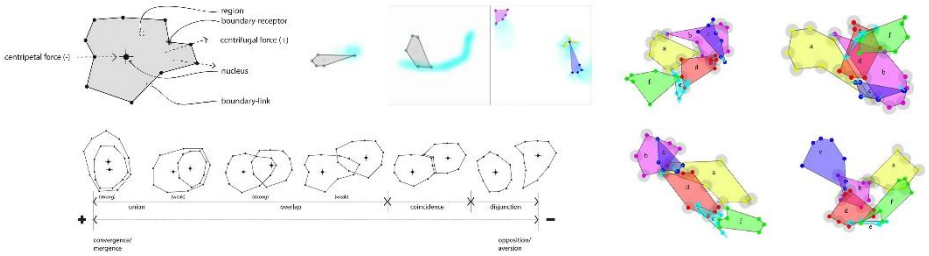


Fig. 4. Actants. Top-left: an Actant and its components; bottom-left: scale of consolidation; top-center: Actants' vertices reacting to pheromone; right: different associations [16].

2.5 Floating Bubbles (FB)

System description: FB [18] is a self-organizing bubble diagram with optional rectangular boundaries.

System algorithm: Physics simulation and heuristics solve the adjacency between bubbles. Agents are represented as circular disks or, optionally, as rectangles, connected by adjacency edges. To solve the diagram, the system assigns two basic forces to the agent: attraction and repulsion. Attraction is proportional to the length of the vector between the boundaries of two connected bubbles. Repulsion is proportional to the overlapping area between two bubbles and pushes each bubble away. However, using only attraction and repulsion forces, the bubbles can get stuck in suboptimal

arrangements. To avoid these situations, a heuristic moves a bubble towards a connected bubble when an adjacent requirement stays unsatisfied for too long. As the heuristic restarts the interactions in a slightly different state, eventually it reaches the equilibrium and satisfy all the adjacencies.

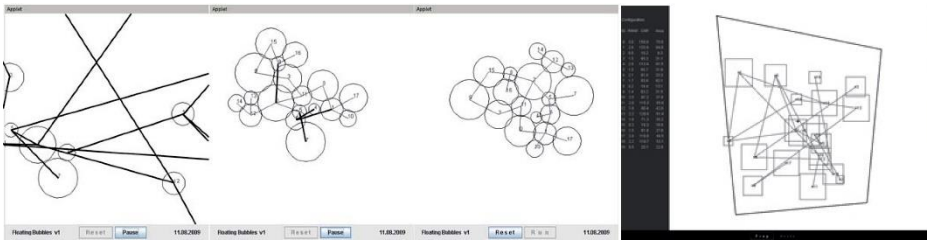


Fig. 5. Floating Bubbles. Left: three stages in the basic floating bubble system; right: one stage of the generative process using squares [18].

2.6 Space-planning with real-time physics (RTF)

System description: RTF [19] is a self-organizing, three-dimensional bubble diagram.

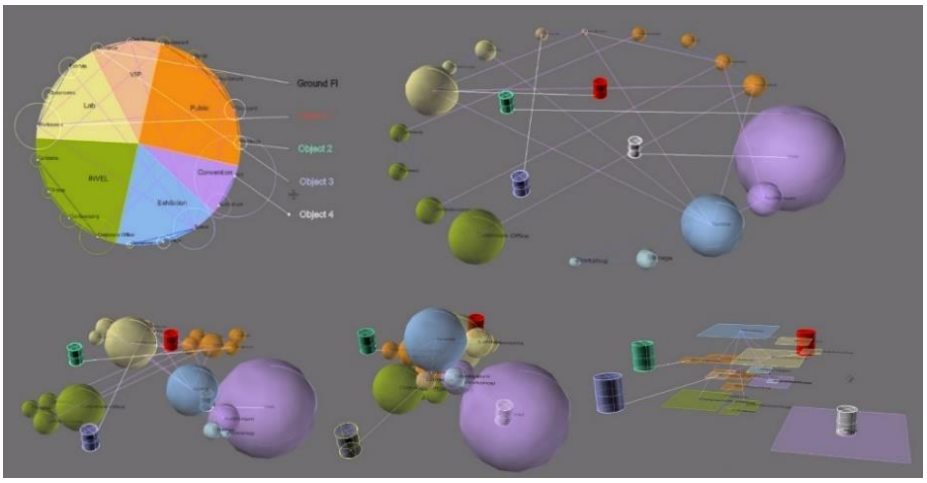


Fig. 6. Space-planning with real-time physics. Top left: abstract diagram. Top right and bottom: physics simulation for bubble diagram.

System algorithm: The user defines an abstract diagram with primary and secondary adjacencies among the elements of the architectural brief. The elements can also be connected to anchor objects, which represent important features of the environment. The specifications of the abstract diagram are translated to a physics simulation, where the elements are spheres, the edges are springs (with different forces for primary, secondary, and anchor connections), and the anchors are cylinders. The resulting bubble diagram is solved by packing the spheres using spring forces. To translate the bubbles

to a floorplan diagram, the spheres are converted to squares in the three-dimensional space, which can be customized by the user.

2.7 Interactive Bubbles (IB)

System description: IB [20] is a self-organizing bubble diagram.

System algorithm: It combines a temperature heuristic with direct application of attraction and repulsion forces to organize circular agents. Attraction and repulsion forces solve the adjacency of connected bubbles and pack them inside a user-defined polygon. The overall behavior of each agent is a result of the sum of the vectors of the forces. A non-destructive heuristic uses varying temperatures to prevent suboptimal arrangements. The temperature is inversely proportional to the satisfaction of the objectives. An unsatisfied agent has higher temperature, reducing its area and its reaction to the surrounding forces. When the agent gets closer to satisfy its objectives, the temperature is reduced, recovering its original area and behavior. The user can manipulate the descriptors of the bubbles in real-time.

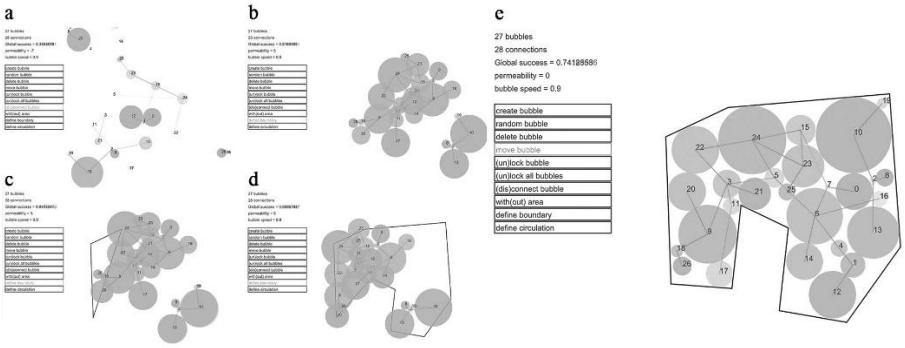


Fig. 7. Interactive bubbles. a-b: simulation of the bubble diagram with changing temperature; c-e: simulation with the definition of a containment area [20].

2.8 Narrative Landscape (NL)

System description: NL [21] is a self-organizing system that packs convex polygons.

System algorithm: It uses a Markov chain combined with physics simulation to create and pack convex spaces. Initially, the user defines a set of spatial types with distinct geometric properties and specifies a transition matrix. The generative process starts by randomly sampling an initial space in that set and sequentially sampling the next spaces using the transition matrix. Different transition matrices and spatial properties will generate different chains of polygons. Each space is a soft body composed of a central particle and surrounded by child particles on the border. The behavior of the bodies is defined by the combination of four types of specialized springs: (1) a Markov spring connects the central particles of neighboring spaces to reinforce adjacency; (2) a set of

collision springs attaches the central particle of a space to close particles of neighboring spaces to reinforce alignment; (3) a minimum distance spring preserves a minimum distance between spaces with different qualities; (4) a within distance spring reinforces adjacency for spaces with the same qualities that are not connected by Markov springs.

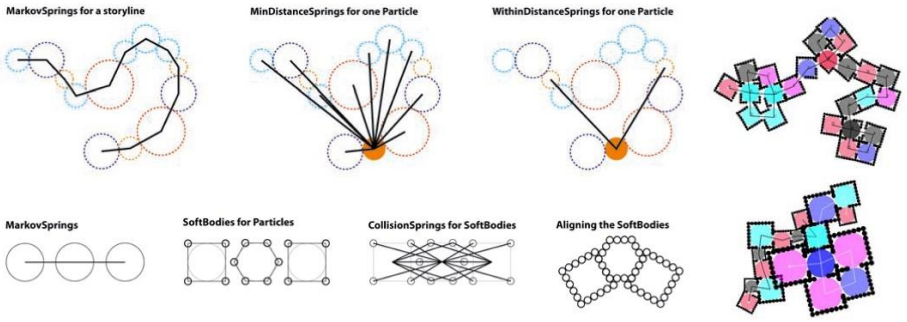


Fig. 8. Narrative landscape. Left: soft bodies and classes for springs connecting particles; right: two layouts [21].

2.9 Multi-agent + Evolutionary (MAE)

System description: MAE [22] is a MASP hybrid of type 1 and 2 that combines a 3D bubble diagram with a grid-based system to allocate activities for architectural layouts.

System algorithm: The simulation in MAE uses two types of agents: spheres that represent regular rooms and capsules that represent corridors, staircases and linear rooms. The agents are placed on the environment and are connected based on adjacency requirements. They interact using rules such as attraction, repulsion, swap and compression. In attraction, each agent is pushed closer to a connected agent by a vector m, defined by the difference between their closest points. Repulsion uses the inverse of m to push overlapping agents apart. For most of the cases, attraction and repulsion only move the target agent. However, they can also affect the orientation and length if the target is a horizontal capsule-like agent. Pairs of connected agents use the swap rule to avoid crossing connections with other pairs. The compression rule is a heuristic that reduces the volume of the building by pushing the agents together along the simulation.

After the simulation, the resulting spaces are assigned to a 3D grid, forming sets of cells with the same room identities and sets of faces between the different rooms. This is the initial state for an optimization. At each iteration, a child solution is generated by mutation – room identities are swapped, and room shapes are changed by moving coplanar faces. The parent and child are evaluated by their room shapes, dimensions, aspect ratios and building shape. An annealing-based function defines the probability of accepting the child as the parent for the next iteration.

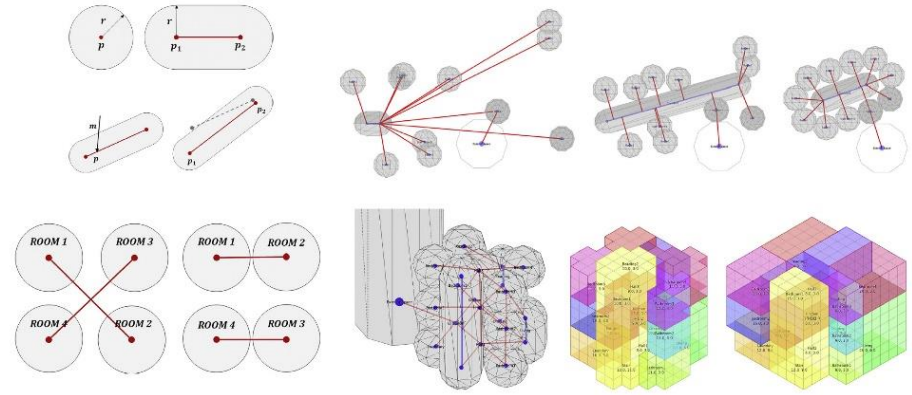


Fig. 9. Multi-agent + Evolutionary. Top-left: spherical agent, capsule-like agent and push operation; bottom-left: swap operation; top-right: agents solving adjacencies; bottom-right: converting agents to a grid representation and optimizing the solution [22].

3 Type 2: Agents that occupy a space

Type 2 agents do not have a custom spatial shape. They navigate and allocate activities in a set of pre-defined spatial units, such as grids.

3.1 Sniffing Space II (SS2)

System description: SS2 [23] is a grid-based system to allocate activities in the routes of an airport.

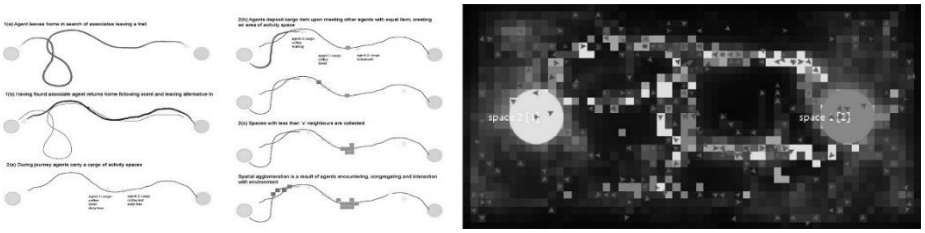


Fig. 10. Sniffing Space II. Left: Routing between destination points; Right: cargo clustering [23].

System algorithm: It combines ant foraging with a cargo placement protocol to allocate activities on a grid. The ant foraging algorithm simulates passenger routing in an airport. Each ant has an agenda with a sequence of destinations or hubs, which represent specific spaces of the airport. Ants not only follow pheromones and navigate between hubs but also respond to other agents sharing similar qualities. These qualities are encoded as random activity cargos that are assigned to each agent. As agents with similar items meet on a trail, they deposit the cargo at their current location. Every time another agent with a similar cargo passes by this cluster of cargos, it also drops it. In

contrast, a passing agent might collect dropped cargos that have less than n neighbors, preventing scattering of activities. The concentration of the cargos defines the activity cells on the routes between destination points.

3.2 Stigmergic Space Adjacency Software (SSAS)

System description: SSAS [24] is a self-organizing system that allocates different activities in a three-dimensional grid.

System algorithm: Agents represent different activities connected by adjacency relationships. Each agent has a specific RGB value and is compatible with agents within a close color range – meaning that they should be adjacent. The agents dispute the territory in a three-dimensional grid of cells. They expand to unoccupied cells with closest pheromone values and, after achieving the desired area, contract. At each node, the agent changes the local environment by diffusing its pheromone value, which attracts agents with compatible values. A regular cell carries a pheromone value and is influenced by the agents. However, the cell can be customized to impose a pre-defined value, stimulating spatial templates over which the agents will dispute the territory – such as circulation systems, boundary areas closer to the light exposure, etc. Also, the user can customize the set of nodes of the three-dimensional grid, creating topographies, a pre-defined *parti* or even voids to preserve views.

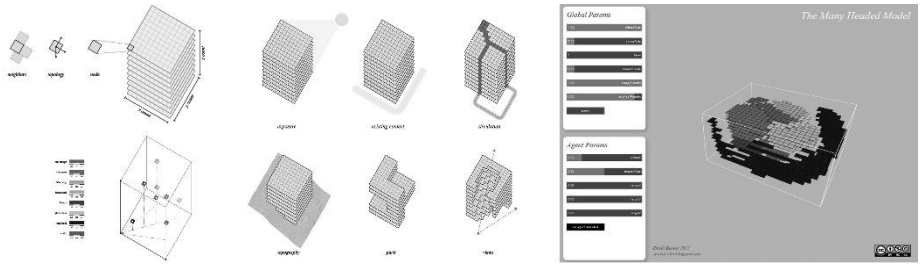


Fig. 11. Stigmergic Space Adjacency Software. Left: grid topology and example of pheromone values; middle: templates for external influence and node masking; right: a solution [24].

3.3 Collective construction (CC)

System description: CC [25] is a multi-agent learning system that allocates different activities in a three-dimensional grid.

System algorithm: Ten color-coded groups containing fifty agents allocate the different activities in the grid using adversarial learning. During the simulation, each agent perceives four features in its cone of vision and can take four actions. The features are: (1) the agent is on the ground level; (2) there is a teammate nearby; (3) there is an opponent nearby; and (4) there is a building block nearby. The actions are: (1) move forward; (2) flock, to align its heading with its teammates; (3) attack, which converts

opponents to teammates or eliminates teammates; and (4) build, which places a building block of the team's color in the cell. Each group uses a neural network with shared weights as its policy. At each time step, the agent uses the four features detected as the input of the network, which outputs the probability of executing the four actions. Then, the agent selects the most probable one. At the end of the simulation, the performance of each team is measured by the number of blocks produced multiplied by the number of surviving team members. The four elite networks remain for the next episode, while the other six are defined by mutation and combination. This experiment is repeated, resulting in different arrangements.

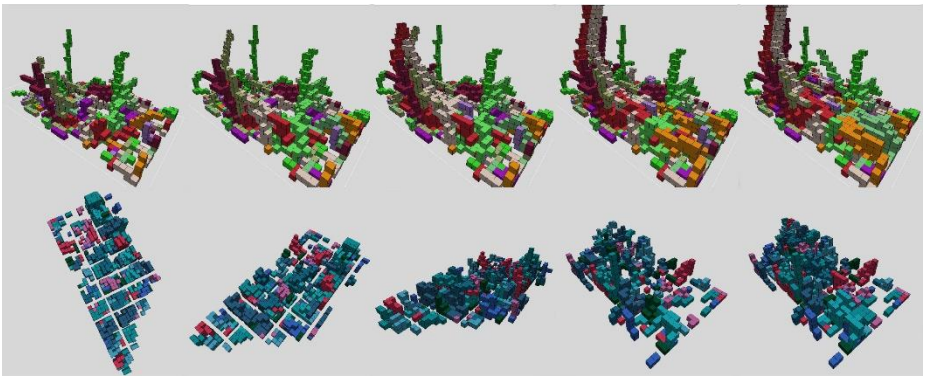


Fig. 12. Collective construction. Results of the experiments [25].

3.4 Artificial Life Space Planning (ALSP)

System description: ALSP [26] is hybrid system of type 1 and 2 that combines activity allocation in a two-dimensional grid with a bubble diagram.

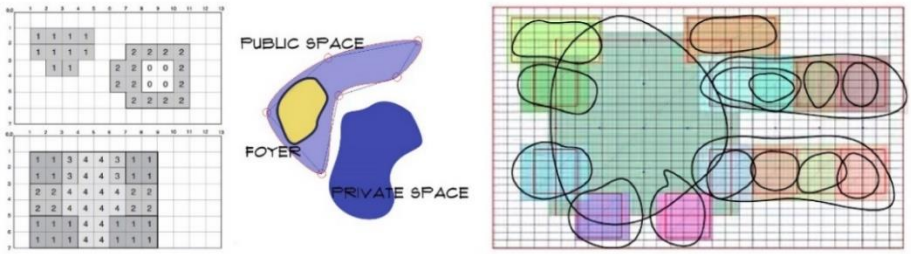


Fig. 13. Artificial Life Space Planning. Left: grid-planning system; center: bubble diagram; right; integration of grid-planning system and bubble diagram [26].

System algorithm: Each bubble is a soft-body defined by a curve interpolated over a series of control points. The user can create, edit, drag and connect these bubbles. Then, a genetic algorithm (GA) specifies rectangles on a grid. The genome is based on five

descriptors: row, col, width, length and state. A fitness function evaluates the area and adjacency of a grid allocation based on the specifications of the bubble diagram (area of the bubbles and connections). Finally, the bubbles will move to the solution specified on the grid, stablishing a design cycle.

4 Type 3: Agents that partition space

In contrast to agents that have an attached spatial representation or agents that occupy a discrete set of cells, type 3 agents navigate in the territory and parametrize the partition of the space.

4.1 YONA (Y)

System description: YONA [27] is a graph-based system to generate residential layout. It serves as an historically important precedent for type 3,

System algorithm: It is based on a three-stage representation with incremental design information: graph embedding, bubble-diagram, and schematic plan. YONA tests the adjacency graph specified by the user to ensure its planarity and, then, uses a top-down heuristic to generate a graph embedding. The user can rearrange the embedding by moving each node to a new position. After a satisfactory arrangement, the program draws an offset boundary and a dual-graph, creating multiple polygonal partitions around the original nodes. Then, it generates b-spline curves inside these partitions, forming a bubble-diagram over which the user can sketch the shape of the rooms.

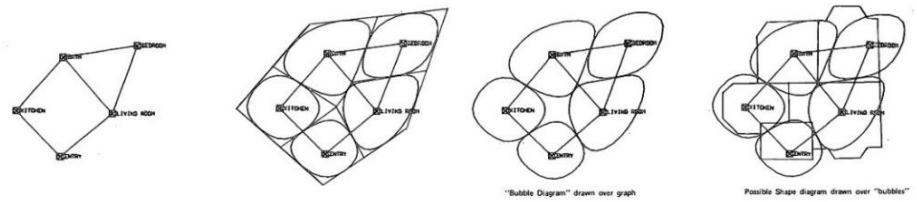


Fig. 14. Stages of YONA: graph embedding, polygons generated by dual-graph connected to offset boundary, bubble-diagram, and plan drawn by user [27].

4.2 Associative Spatial Networks (ASN)

System description: ASN [28] is a system that generates an exhibition hall layout according to the topology of the exhibition and potentially to the users' feedback.

System algorithm: The system is divided in 3 parts. Firstly, adjacency graphs are generated for the exhibitions. Each individual exhibit of an exhibition is described by multiple features and projected to a point in $\mathbb{R}^2$ by a self-organizing map (SOM). A planar graph is created by connecting each point to the neighbors with the closest

distance and by storing these values in the edges. The second stage involves clustering exhibitions that can share similar spatial configuration of the layout. This is solved by a growing neural network that adapts to the varying dimension of the input spectrum for each adjacency graph, clustering them according to similar topologies. The graph with the spectrum closest to the average in each cluster is selected to define the floor plan. In the third stage, the selected average graphs are embedded on a plane and their nodes are distributed in the exhibition hall layout by a repulsion algorithm. Using a Voronoi diagram, each node becomes a polygonal exhibition cell, and the permeability of the wall is defined by the connection between the neighboring nodes (wall for no connection, permeable wall for high edge values and no wall for low edge values).

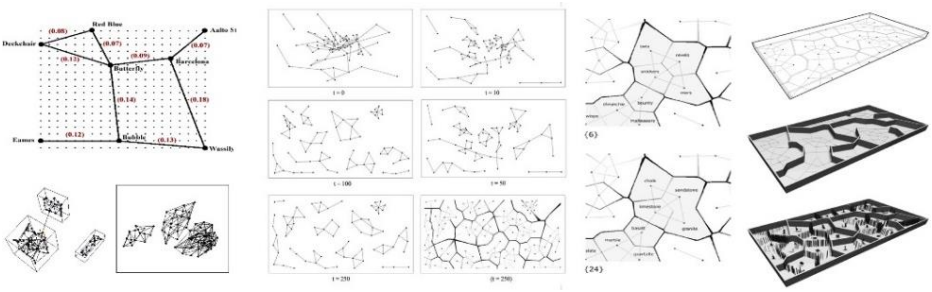


Fig. 15. Associative Spatial Networks. Top-left: adjacency graph for exhibition; bottom-left: growing neural gas mapping; middle: repulsion algorithm distributes graph nodes over the space; right: translating a graph into the layout of the exhibition with a 2d Voronoi diagram [28].

4.3 Dwelling Agglomerator (DA)

System description: DA [29, 30] is a self-organizing settlement of dwelling units connected by public spaces.

System algorithm: It associates a custom flocking model with Voronoi partitioning and an evaluation function to generate settlements in real-time. A flocking simulation moves the agents to form a network with approximated orthogonal connections, following five basic navigation rules. The first two rules are movements that preserve a specific distance range between the agent and its closest neighbors. With this range satisfied, other three rules specify a rotation around the closest neighbor, moving the agent towards a position aligned or perpendicular to the edge connecting the two closest neighbors. The agents are used to partition the space with a Voronoi Diagram, forming the dwelling units and the open space. An evaluation function calculates a score for each unit based on criteria such as area, direct sunlight or accessibility. It defines whether the agent should stop or keep flocking to look for a better position.

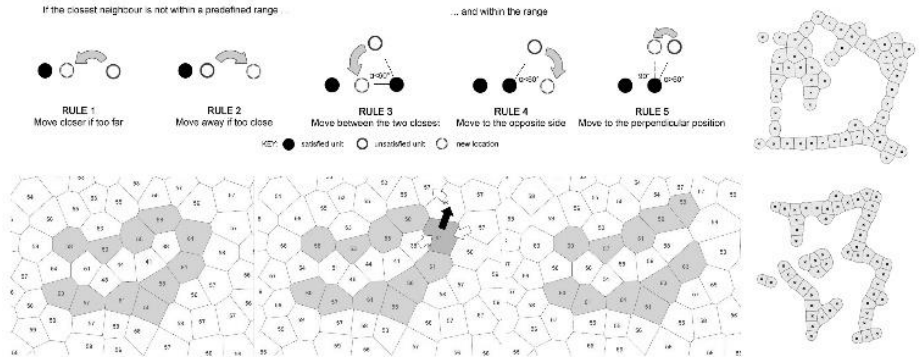


Fig. 16. Dwelling Agglomerator. Top-left: flocking rules; Bottom-left: creation of access routes; right: two layouts [29].

4.4 Responsive Algorithms (RA)

System description: RA[31] is a MASP of type 1 and 3 that generates spatial partitions based on an architectural brief.

System algorithm: RA combines bubble diagrams, graph embeddings and Voronoi partition inside a parametric modelling editor. The user defines a bubble diagram, with disks representing the spatial units with their required area, and edges representing the adjacency. The bubble diagram is treated as a circle packing problem and solved by physics simulation: circular discs are rigid bodies and the edges are springs representing adjacencies. A Voronoi diagram generates a spatial partition using the centers of the disks. This workflow is extensible to the three-dimensional space, where the nodes of the adjacency graph are the centers of packed spheres and a Voronoi partition results in a packing of polyhedral cells.

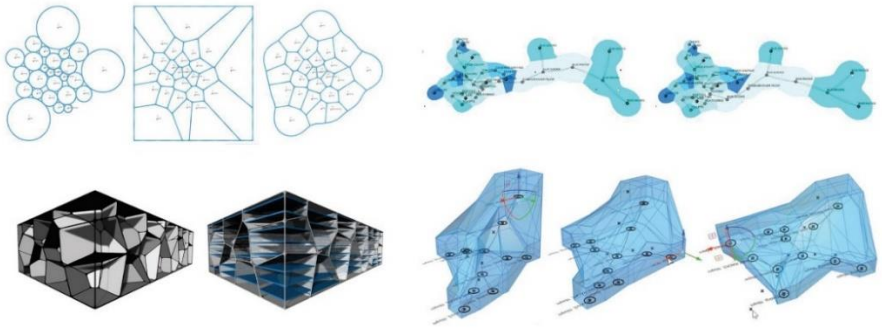


Fig. 17. Responsive Algorithms. Top-left: circle packing, Voronoi diagram and Voronoi diagram with radii; top-right: force-directed graph over Voronoi diagram with radii; bottom-left: 3D Voronoi cells; bottom-right: three-dimensional packing with a skin [31].

4.5 Evolving Floorplans (EF)

System description: EF [32] is MASP of type 1 and 3 that generates floorplans based on topological and flow objectives.

System algorithm: EF uses a general version of Neuroevolution of Augmenting Topologies (NEAT) to evolves a weighted, connected, and undirected graph representing a floorplan. Chronological markers enable crossover and mutation of solutions with different topology and numbers of rooms. The floorplan genome of NEAT is composed of node genes and connection genes. The former store information such as room's size, while the latter assign weights to the connections between pairs of node genes and are added randomly until the graph is connected.

The phenotype (i.e. the floorplan) is generated in four steps. (1) A spectral layout embeds the graph in space, which is the input for (2) a physics simulation that converts the nodes and edges into discs and springs. After the resulting packing, the centers of the discs are extracted and combined with the vertices of an offset concave hull for (3) the Voronoi tessellation. With this combination, the internal Voronoi cells have a boundary like the original circle packing boundary.

Finally, (4) an ant-colony algorithm is applied to the interior edges of the Voronoi mesh to generate a weighted network to satisfy the connection requirements. The edges of the resulting hallways are smoothed and dimensioned according to their flow. The phenotype is evaluated by the NEAT algorithm, enabling the use of objectives such as minimization of traffic between classes, material usage, fire escape paths, and windows.

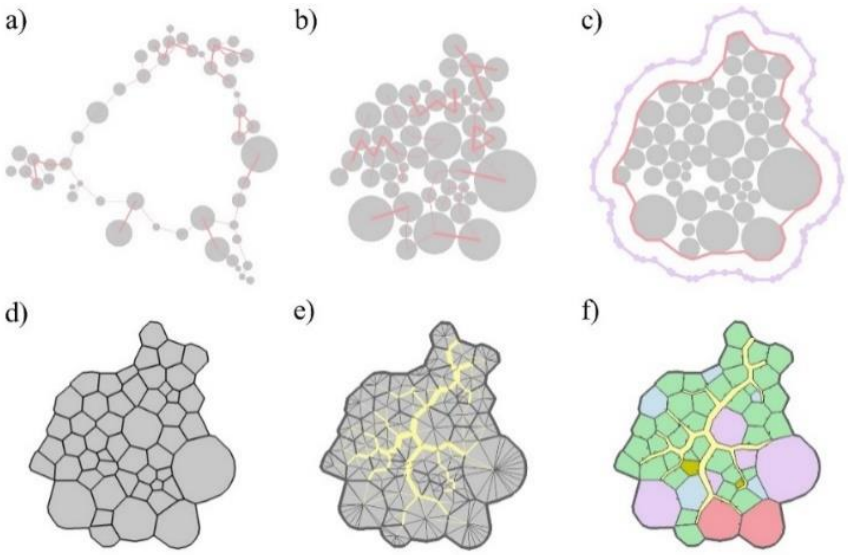


Fig. 18. a) spectral layout; b) physics simulation; c) offset of concave hull; d) Voronoi tessellation; e) generation of circulation using an ant-colony algorithm; f) final floorplan [32].

5 Analysis of features

In this section, we analyze the different MASP prototypes to identify the occurrence of similar features of representation, objectives, and control procedures. Before we advance in the analysis, Table 1 synthesizes the general categorization of types.

Table 1. MASP Types.

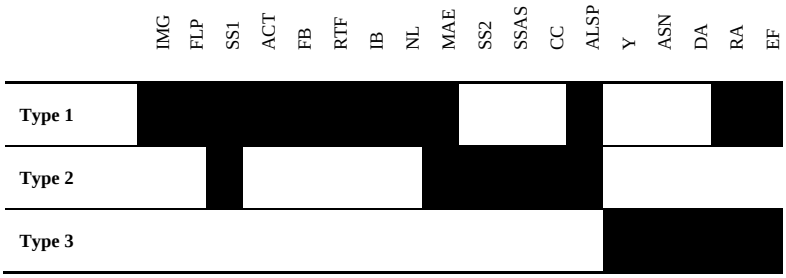
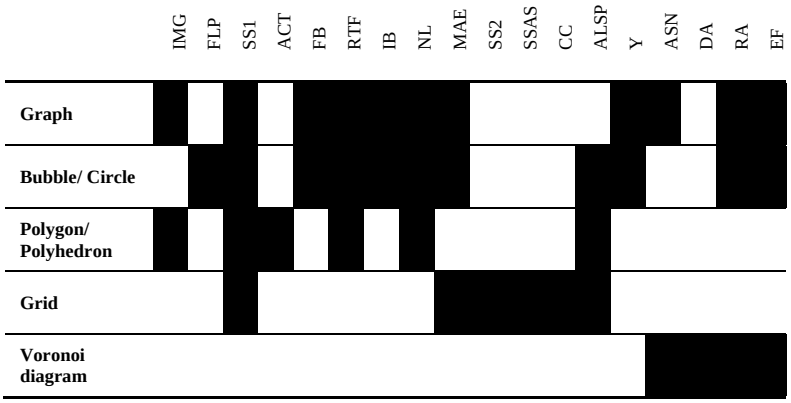


Table 2. Representation of the agents.



In terms of representation, features used in SP for more than four decades [3], such as grids, polygonal representations, and graphs, are still pervasive in MASP. The prototypes also incorporate new features, such as soft shapes (bubbles) and the Voronoi diagram for spatial partition. As shown in Table 2, bubbles and polygons are dominant in type 1. Grids are mostly used in type 2. Voronoi diagrams are dominant in type 3. Graphs are used predominantly in type 1 and 3 to store the connections between spaces. In some prototypes, the diagrammatic solutions were also converted to richer architectural representations with constructive elements (FB and MAE).

Table 3. Spatial objectives.

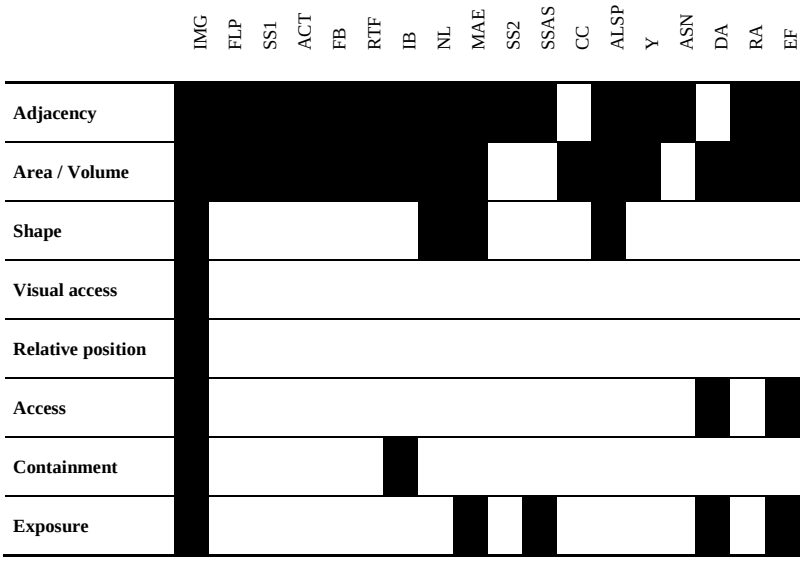


Table 4. Control procedures

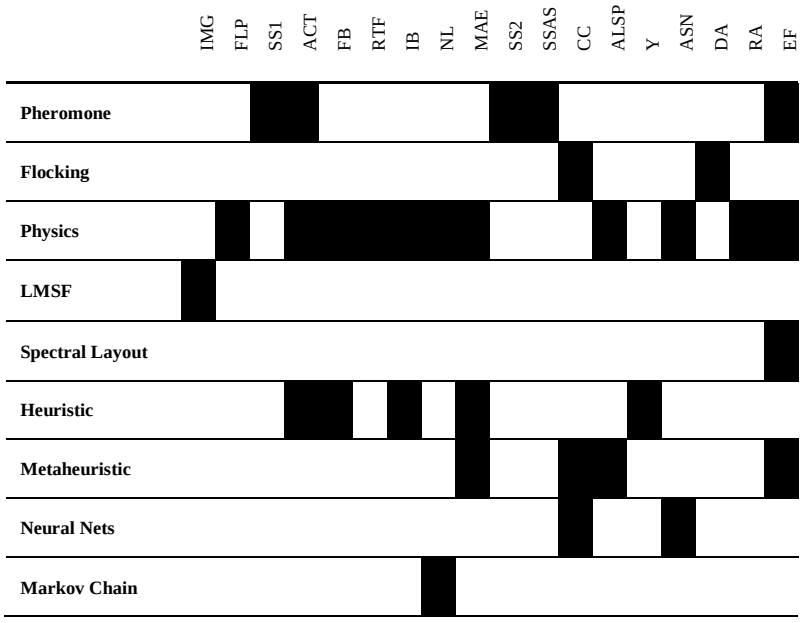


Table 3 shows a contrast between the number of objectives in IMG and in new prototypes. IMG has a list of seventeen objectives, which we simplified to eight features for analysis. Current systems are more abstract and focus on satisfying basic objectives such as adjacency and area/volume. In some of them, relations such as exposure or visual access can be customized by using the tools for adjacency. For example, SSAS uses pheromone markers to promote the exposure or privacy of a part of the activities to the exterior of the building and MAE uses a form of attraction to reduce the external surface of the building. Some systems have an explicit implementation of different objectives, such as the containment of the agents in a custom polygon (IB) or an objective function to evaluate accessibility and exposure (DA).

Table 4 displays the large diversity of control procedures in the prototypes.

Different types of physics simulations are present in most prototypes. They implement a custom version of attraction/repulsion forces (FB, IB and MAE), rigid body simulation (RTF, RA and EF), soft body simulation (ALSP, NL, and ACT), or another unspecified implementation (ASN and FLP).

The second most frequent occurrence is pheromone routing. The algorithms are used to guide large spatial clusters (SS1 and SSAS), to control the vertices of polygonal entities (ACT), to guide agents that carry and drop spatial units (SS2), or to generate a weighted circulation network between the consolidated spatial units (EF).

Flocking has only two occurrences. In DA, the system is a customized Boid model that generates semi-orthogonal networks with spatial agents. In CC, flocking is one of the actions in the policy of agents in a three-dimensional grid.

Hybrid methods are used for all MASP types. They incorporate techniques outside of the scope of agent-based modeling, such as LMSF to guide the agent (IMG), metaheuristics to initialize or refine the solutions (MAE, ALSP and EF), neural networks to generate the adjacency between agents (ASN) or to define the policy of the agents (CC), and Markov Chain to generate new agents (NL).

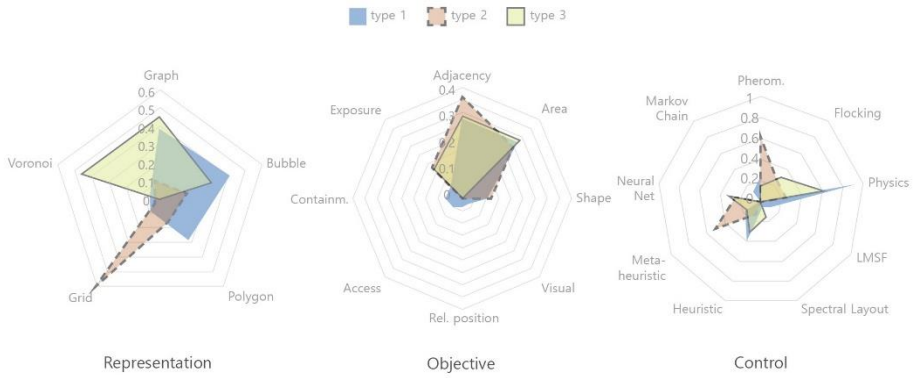


Fig. 19. Radar charts with the relative occurrences of the features for each type. For hybrid prototypes, we divided the occurrence uniformly for each type.

Table 5. Correlation matrix for occurrence of features.

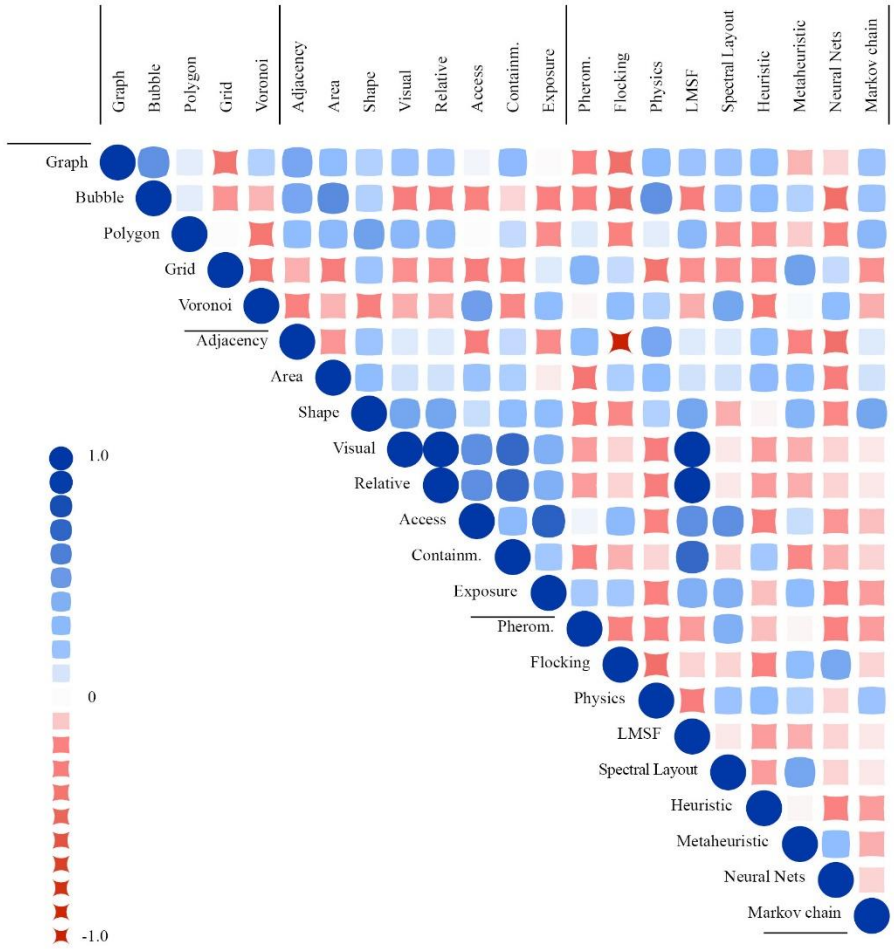
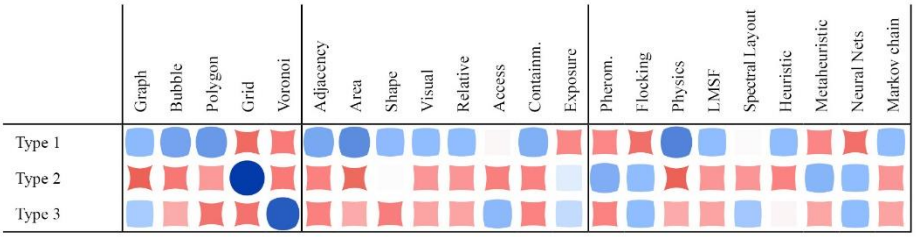


Table 6. Correlation matrix for relations between types and features. Blue is positive and red is negative. For hybrid types, we divided the occurrence uniformly.



Using correlations between features (Table 5**)**, correlations between types and features (Table 6) and the radar charts for the occurrence of features in the types (Fig. 19), it is possible to visualize certain patterns. The occurrences of representational and control features present clear and distinct patterns for each MASP type. In the occurrence of objectives, all types overlap on adjacency and area.

Type 1 has strong occurrence and positive correlation with graphs, bubbles, polygons and physics simulation. Bubbles are used to represent fluid moving shapes, while polygons mostly represent rigid shapes for packing (ACT is an exception and uses polygons for fluid shapes). Other than access and exposure, Type 1 has the strongest positive correlation with all features, even with the two objectives that occur in all types (adjacency and area), which are also strongly correlated to the main representational features of type 1 (graphs, bubbles and polygons). The occurrence of adjacency and area in other types is mostly related to hybridization with type 1.

Type 2 has high occurrence and positive correlation with the grids, which are the representational support for correlated control strategies, such as pheromone routing (SS1, SS2, and SSAS), metaheuristic optimization (MAE and ALSP) or a neural network for multi-agent flocking (CC). Type 2 is slightly positively correlated with exposure, not correlated with shape, and negatively correlated with all other objectives.

Type 3 has a strong correlation with the Voronoi diagrams, which are used to translate a population of points in space to polygonal or polyhedral partitions. It also presents some occurrences of graphs and bubbles due to the hybrid prototypes, but these correlations are respectively weak positive and negative. In terms of control, the only significant positive correlations are related to flocking, spectral layout and neural networks. In terms of objectives, it has the largest correlations with access and exposure among all types.

6 Conclusion

The MASP prototypes analyzed in this paper present novel approaches to SP, with a rich repertoire of techniques for interactive SP exploration. However, MASP is still a broad an open territory for research with big challenges and demands.

There is a scientific challenge. While most papers reference general ideas that inspired their approach – such as Cybernetics, Semiotics, Biology, etc. – and present a high-level description of the algorithms, they do not provide a formal description of the control procedures or share the code. Few papers analyze the performance or expression of the prototypes. One paper provides a brief convergence analysis (FB), some papers present a concrete architectural application (IMG and FB) and one prototype was applied to a real project and used in a workshop (SSAS). To move from local experiments to more robust research, it is necessary to provide mechanisms for the reproduction, comparison and analysis of the different MASP approaches.

There is also two-fold computational challenge: investigate control strategies that incorporate domain knowledge and handle multiple spatial objectives and conflicts. While IMG tried to solve this using LMSF to define the agent's policy, new prototypes

rely on three general trends – bioinspired agents, physics-based agents and hybridization with destructive methods – that have some limitations.

Bioinspired models are generally resilient and adaptable. Still, as architectural problems are not always analogous to biological problems, it is usually hard to produce valid architectural patterns and satisfy architectural objectives.

Physics simulation is a more general model that conciliates simple control with intuitive interaction and can be adapted for different spatial problems and objectives. It has recently been incorporated into real world practices [33]. However, physics-based agents are reactive agents approximately following laws of physics. They do not have any sophisticated policy to manage spatial conflicts.

A valid solution to this issue is to hybridize these agents with metaheuristics that globally improve the solution. Nevertheless, it usually requires a pre-defined metric and imposes discontinuity in the generation procedure.

An alternative to these approaches is to develop MASP system with behavioral learning, which is present in only one the papers (CC). The recent success of deep reinforcement learning (DRL, see [34]) in games, robotics and multi-agent navigation suggests that it is a promising field for inquiry. While the multi-agent setting is challenging for DRL and current neural network architectures are mostly effective for Euclidean data, such as grids, recent researches have been pushing the boundaries both for multi-agent learning [35] and for non-Euclidean spaces, such as graphs and manifolds [36]. With DRL, it is possible to learn cooperative behavior to generate spatial patterns continuously. Agents can learn fully from interaction with the task environment in a simulation or with experts in a supervised setting. Potentially, behavioral learning can support the development of cooperative agents that are robust to variations and defined by domain-specific knowledge.

Acknowledgements. We would like to express our gratitude to the Brazilian National Council for Scientific and Technological Development (CNPq) for granting Pedro Veloso a PhD scholarship (grant #201374/2014-5).

References

1. Russel, Stuart J., Norvig, P.: Artificial intelligence: A modern approach. Prentice Hall, Upper Saddle River (2010)
2. Wilensky, U., Rand, W.: An Introduction to Agent-based Modeling: modeling natural, societal and engineered complex systems with netlogo. The MIT Press, Cambridge (2015)
3. Mitchell, W.J.: Computer-aided architectural design. Mason Charter Pub, New York (1977)
4. Liggett, R.S.: Automated facilities layout: past, present and future. Automation in construction. 9, 197–215 (2000)
5. Homayouni, H.: A Survey of Computational Approaches to Space Layout Planning (1965-2000). Department of Architecture and Urban Planning University of Washington. (2000)
6. Hsu, Y.-C., Krawczyk, R.J.: New generation of computer aided design in space planning methods: A survey and a proposal. In: Keatruangkamala, K. and Nakapan, W. (eds.)

Proceedings of the 8th CAADRIA conference. pp. 101–116. Rangsit University, Bangkok (2003)
7. Lobos, D., Donath, D.: The problem of space layout in architecture: A survey and reflections. Arquitetura Revista. 6, 136–161 (2010)
8. Calixto, V., Celani, G.: A literature review for space planning optimization using an evolutionary algorithm approach: 1992-2014. In: Project Information for Interaction: Proceedings of 19th SIGraDi conference. pp. 662–671. Blucher, São Paulo (2015)
9. Chen, L.: Agent-based modeling in urban and architectural research: A brief literature review. Frontiers of Architectural Research. 1, 166–177 (2012)
10. Herr, C.M., Ford, R.C.: 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. Kyungpook National University, Daegu (2015)
11. Weinzapfel, G., Handel, S.: IMAGE: computer assistant for architectural design. In: Spatial synthesis in computer-aided building design. pp. 61–68. Applied Science Publishers, London (1975)
12. Weinzapfel, G., Johnson, T.E., Perkins, J.: IMAGE: an interactive computer system for multi-constrained spatial synthesis. In: Proceedings of the 8th Design Automation Workshop. pp. 101–108. ACM (1971)
13. Bayraktar, M.E., Çağdaş, G.: Fuzzy Layout Planner: A simple layout planning tool for early stages of design. In: Rudi, S. and Sevil, S. (eds.) Computation and Performance: Proceedings of the 31st eCAADe Conference. pp. 375–381. TUDelft, Delft (2013)
14. Ireland, T.: Emergent Space Diagrams: The application of swarm intelligence to the problem of automatic plan generation. In: Tidafi, T. and Dorta, T. (eds.) Joining Languages, Cultures and Visions: Proceedings of the 13th CAAD Futures conference. pp. 245–258. Les Presses de lUniversité de Montréal, Montréal (2009)
15. Ireland, T.: Stigmergic planning. In: Sprecher, A., Yeshayahu, S., and Lorenzo-Eiroa, P. (eds.) LIFE in:formation, On Responsive Information and Variations in Architecture: Proceedings of the 30th ACADIA conference. pp. 183–189. ACADIA, New York (2010)
16. Ireland, T.: An Artificial Life Approach to Configuring Architectural Space. In: Martens, B., Wurzer, G., Grasi, T., Lorenz, W.E., and Schaffranek, R. (eds.) Real Time: Proceedings of 33rd eCAADe conference. pp. 581–590. eCAADe, Wien (2015)
17. Ireland, T.: A cell inspired model of configuration. In: Combs, L. and Perry, C. (eds.) Computational Ecologies: Design in the Anthropocene: Proceedings of 35th ACADIA conference. pp. 136–147. ACADIA, New York (2015)
18. Hao, H., Ting-Li, J.: Floating bubbles. In: Dave, B., Li, A.I., Gu, N., and Park, H.-J. (eds.) New frontiers: proceedings of the 15th CAADRIA conference. pp. 175–183. CAADRIA, Hong Kong (2010)
19. Syp, M.: Space Planning with Physics, http://www.marcsyp.com/space-planning/
20. Veloso, P.: Exploring the bubble diagram. In: Amen, F.G. (ed.) Design in Freedom: Proceedings of 18th SIGraDi conference. pp. 115–119. Blucher, São Paulo (2014)
21. Christensen, J.T.: The generation of possible space layouts. In: Thompson, E.M. (ed.) Fusion: Data integration at its best.: Proceedings of the 32nd eCAADe conference. pp. 239–246. eCAADe, Newcastle upon Tyne (2014)

22. Guo, Z., Li, B.: Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system. Frontiers of Architectural Research. 6, 53–62 (2017)
23. Ireland, T.: Sniffing Space II: The use of artificial ant colonies to generate circulation patterns in buildings. In: Tidafi, T. and Dorta, T. (eds.) Joining Languages, Cultures and Visions: Proceedings of the 13th CAAD Futures conference. pp. 214–227. Les Presses de lUniversité de Montréal, Montréal (2009)
24. Meyboom, A., Reeves, D.: Stigmergic Space. In: Beesley, P., Stacey, M., and Khan, O. (eds.) Adaptive Architecture: Proceedings of the 33rd ACADIA conference. pp. 200–206. Riverside Architectural Press, Toronto (2013)
25. Narahara, T.: Collective Construction Modeling and Machine Learning: Potential for Architectural Design. In: Fioravanti, A., Cursi, S., Elahmar, S., Gargaro, S., Loffreda, G., Novembri, G., and Trento, A. (eds.) Sharing Computational Knowledge!: Proceedings of the 35th eCAADe Conference. pp. 593–600. eCAADe, Rome (2017)
26. Fernando, R.: Space Planning and Preliminary Design Using Artificial Life. In: Gu, N., Watanabe, S., Erhan, H., Haeusler, M.H., Huang, W., and Sosa, R. (eds.) Rethinking Comprehensive Design: Speculative Counterculture: Proceedings of the 19th CAADRIA conference. pp. 657–666. CAADRIA, Hong Kong (2014)
27. Weinzapfel, G., Negroponte, N.: Architecture-by-yourself: an experiment with computer graphics for house design. In: SIGGRAPH. pp. 74–78 (1976)
28. Harding, J., Derix, C.: Associative spatial networks in architectural design: Artificial cognition of space using neural networks with spectral graph theory. In: Gero, J.S. (ed.) Design Computing and Cognition'10. pp. 305–323. Springer, Dordrecht (2011)
29. Puusepp, R.: Agent-based Models for Computing Circulation. In: Gerber, D., Huang, A., and Sanchez, J. (eds.) Design Agency: Proceedings of the 34th ACADIA conference. pp. 43–52. Riverside Architectural Press, Cambridge (2014)
30. Puusepp, R.: Spatial Agglomerates. In: Gu, N., Watanabe, S., Erhan, H., Haeusler, M.H., and Huang, W. (eds.) Rethinking Comprehensive Design: Speculative Counterculture: Proceedings of the 19th CAADRIA conference. pp. 585–594. CAADRIA, Hong Kong (2014)
31. Bazalo, F., Moleta, T.J.: Responsive Algorithms. In: Ikeda, Y., Herr, C.M., Holzer, D., Kaijima, S., Kim, M.J., and Schnabel, M.A. (eds.) Emerging Experience in Past, Present and Future of Digital Architecture: Proceedings of the 20th CAADRIA Conference. pp. 209–218. CAADRIA, Hong Kong (2015)
32. Simon, J.: Evolving Floorplans, http://www.joelsimon.net/evo_floorplans.html
33. Derix, C., Izaki, A.: Spatial computing for the new organic. AD. 83, 42–47 (2013)
34. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. The MIT Press, Cambridge (2018)
35. Nguyen, T.T., Nguyen, N.D., Nahavandi, S.: Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions and Applications. arXiv:1812.11794 [cs, stat]. (2018)
36. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric Deep Learning: going beyond Euclidean data. IEEE Signal Processing Magazine. 34, 18–42 (2017)