Autonomous multi-agent construction

Justin Werfel
6.978

December 10, 2002

Abstract

Swarm intelligence, the idea of teams of autonomous low-level agents operating in collaboration to achieve some high-level goal without centralized control, is an area of growing research interest with a wide range of potential applications. Here I describe the design and construction in simulation of a system of mobile agents that arrange blocks into desired two-dimensional shapes on a grid. Such a construction team could be used as the first stage in a system for remote building of structures, laying out the floor plan that a more sophisticated system could extend upwards.

1 Introduction

The traditional approach to an engineering project is to develop a system constructed to possess a desired set of capabilities by centralized design and control. Another approach to solving a problem is via the use of many simpler agents with simpler individual behavior, from whose interactions the desired high-level function emerges. This approach, found more typically in natural than man-made systems, generically gives greater flexibility and robustness, but is not yet understood to the necessary extent as to allow the use of a set of standard design tools. Because of its many advantages, demonstrated throughout biological systems, this study of so-called swarm intelligence is one that has been receiving increasing research interest in the recent past[1].

One family of applications for such systems has to do with construction. The possible uses for structure-building teams of robots are many and far-ranging, from automating the production of low-cost housing to allowing construction in settings where human presence is dangerous or problematic. This latter class in turn ranges from uses in disaster areas, to the construction of first-stage

[1] It's also the basis of Michael Crichton's latest technothriller.

bases of operations to await the arrival of pioneers in, for example, underwater or extraterrestrial environments.

The natural inspiration for such a system is the colonies of social insects that, despite their small size and limited complexity, collaborate to build enormous and complicated structures in a variety of hostile environments. Their nests can contain a variety of specialized structures (for ventilation, food storage, gardening, brood chambers, etc.), and are organized based in part on need, the presence of obstacles, etc. in addition to higher-level considerations.

2 Previous work

Earlier work on construction by teams of mobile autonomous robots includes that at the Interaction Lab at USC. Among their recent results is demonstration of the construction of a linear wall made out of bricks held together with Velcro of alternating polarity. This was accomplished in hardware with a single robot; when, in simulation, they used multiple robots, the robots primarily interfered with each other; limited communication between robots permitted the reduction of that interference.

The University of Bristol is likewise performing experiments involving robots moving building units on a two-dimensional surface. Their primary concern is minimalism, finding the simplest possible set of rules that still results in the construction of some organized structure. The techniques they develop with that constraint are probabilistic and very slow, and require frequent disassembly of incorrectly built parts.

A research group at the University of Bath is pursuing a very ambitious project whose goal is to create robots that build full three-dimensional walls and arches at human scale. Their primary focus is on the mechanical engineering concerns associated with building the robots, particularly due to the necessity that they be able to climb on the structures they build; they are less concerned with the coordination of multiple robots.

Eric Bonabeau, one of the pioneers of research in swarm intelligence, and his colleagues have taken a different approach to the problem, simulating mobile construction agents in three dimensions and studying the kinds of structures that result when the agents are given different rules directing their behavior. This approach may lead to greater eventual understanding of how higher-level results emerge from low-level behaviors, but is unlikely to be immediately useful in the design of systems with given desired high-level behavior.

3 The 6.978 system

The goal of the project for this class was the development of a system of simple, identically programmed, autonomous agents able to build two-dimensional structures of arbitrary design by rearranging blocks of building material into desired shapes. Design considerations for the system were consistently shaped by this goal and by the desire for the agents to remain as *simple* and as *local* as possible. Simplicity considerations included trying to minimize the amount of memory that agents required, the list of their capabilities, etc. Locality considerations restricted them to information only about their immediate surroundings.

The simulation was written in Swarm built on top of Objective C, restricting objects to occupying one space at a time of a two-dimensional cellular grid. In that grid world are found bricks, passive objects which can be picked up and moved, initially scattered at random across the surface; there are the agents, which perform the construction in ways to be described; there is also a beacon, an active though immobile object that serves as the reference point around which all agent activity occurs. The idea is to be able to scatter a handful of generic agents in the vicinity of sufficient building material, place a beacon programmed with the desired building design, and let construction proceed without further intervention.

The beacon sends out a signal, which agents can pick up and, based on its strength and direction, use to determine the distance and direction to the beacon. The beacon can communicate in a more sophisticated way with agents when they get close enough; this communication is primarily concerned with the building design, which is preprogrammed into the beacon (allowing the same agents to be involved in any construction project without needing to be reprogrammed). The design takes the form of a list of corners for the desired building; each specifies its distance from the beacon, the angle to the next corner, and whether the wall between the two is to be straight or curved. Such a list completely specifies the building geometry, though not its orientation.

3.1 Agent capabilities

Agents have the abilities to pick up and put down single bricks; to move in any of the four cardinal directions, if nothing blocks them (or, if applicable, the brick they carry) from doing so; to measure the strength of the beacon signal within a local area of radius r,[2] and evaluate its gradient based on those measurements; to perceive bricks and other agents within that distance; and to communicate with agents within that distance, exchanging information and commands.

For the sake of convenience and because of the limitation of the cell-based framework, agents also have access to the global coordinate system (which, because of the shared fixed reference of

[2] In the simulations whose results are shown, the value of r used was 4; this was the smallest value for which the corresponding neighborhood on the cellular grid appeared reasonably circular to visual inspection.

the beacon, is equivalent to giving each a perfect motion integrator and a perfect compass). This nonlocal and nonindependent ability would be absent in a physical instantiation of the system; each agent would establish its own coordinate system, use motion integration to keep track of its position, keep its axes calibrated by reference to frequently encountered fixed markers (the beacon and, once established, the corners of the structure), and match coordinate systems with other agents when they exchange information based on a transformation according to mutual reference to the same markers. The transition to independent coordinate systems for each agent would doubtlessly not be a trivial one and would require the solution of additional engineering problems, but should not be a substantial obstacle to the realization of this system without a global coordinate system.

3.2 Agent state

Agents have the following internal variables.

1. *mode*: one of {clear, doneclearing, bepoint, collect, seal, off}. These are described in the discussion of agent behavior below.

2. *goal*: a pointer to an object, either nil, the agent itself, the brick it's carrying (if applicable), or the beacon; used primarily as an auxiliary mode indicator.

3. *held*: a pointer to the brick the agent is holding, or nil if none.

4. *thresh*: an integer controlling the size of the area cleared for construction, based on the size of the desired structure as specified in the building plan; it could be recalculated each time it's needed, if memory is at that much of a premium.

5. *frust*: an integer incremented if the agent is unable to move in its desired direction during a time step, and reset to zero when the agent moves. If it reaches too large a value, the agent changes its behavior, in ways depending on the mode.

6. *X, Y; destX, destY; beacX, beacY*: positions of the agent, its current destination, and the beacon, in global coordinates.

7. *plist*: the building plan, expressed as an array of corners, as described above.

8. *pnum*: the number of the corner the agent either embodies or is currently working on building a wall originating from.

9. Other internal variables: fewer than a dozen, used for various lower-level purposes.

3.3 Agent behavior

The algorithm the agents follow is essentially as follows. Start in *clearing* mode: follow the signal gradient to the beacon (establishing its position), then circle outwards. If a brick is encountered on the way, pick it up and carry it directly outward until the signal strength from the beacon passes some threshold (*thresh*), then go back to the beacon and repeat. If the threshold is reached without any bricks encountered, or if an agent in any mode later than *clearing* is encountered, enter *doneclearing* mode: circle back inwards, in the opposite direction to increase the number of *clearing* agents encountered, to bring the entire agent population onward to the same mode and avoid the problem of having some agents working on building the structure while others clear it away just as quickly. After completing several such circles, follow the gradient to the beacon to receive an assignment.

The beacon has the design for an N-corner structure, as previously described. The first N agents that come to it in this mode are assigned to be successive numbered corners, and enter *bepoint* mode. The first of these follows the gradient outward to the appropriate radius, and immobilizes itself there. Each succeeding corner is specified in relation to the previous; the corresponding agent circles at that radius until it finds that previous corner or another agent that knows its location; it then calculates its destination location on that basis, and goes and immobilizes itself there.

Agents after the first N that reach the beacon receive the building design, choose a pair of successive corners at random to build a wall between, and enter *collect* mode: first they must know the locations of their selected corners, which they find either by seeking them out personally or by being told their locations by agents they encounter which already know; during this stage they circle in the opposite direction to other agents, again to increase the rate of unique encounters. Next, they go out beyond the outskirts of the cleared area to find a brick, go to the first of their two corners, and follow the line between the two (straight or curved as appropriate) until they find a valid place to put their brick that isn't already occupied. They do this by calculating the location of the nearest point to themselves on the desired wall, i.e., the perpendicular to the line or arc connecting the two corners, based on the known positions of those corners and the type of wall desired. They then look to see if that cell is occupied (which requires being within sensor range). In the current version, agents do not distinguish between occupation by carried bricks, placed bricks, or other agents; this may lead to temporary bypassing of locations that would have opened up a few time steps later when the blocking agent moved on, but it also helps avoid traffic jams (the agent in the way may in turn be waiting for the first agent to get out of its way so it can leave the area), and if more than a few agents are at work, another will be able to fill in the gap during a later pass; also, distinguishing between carried and placed bricks would require a more sophisticated identification system in a hardware implementation of this system. If the cell in question is unoccupied, the agent goes on to try to place the brick there, setting its variable *goal* to indicate that to itself in future time steps; otherwise, it moves along the desired wall, and will

check the new perpendicular location on the next time step.

Agents repeat this process until they reach the second corner without finding a place that needs a brick, at which point they enter *seal* mode; they return back along the wall to the first corner, making sure there are no gaps they missed the first time. A more complete system, with agents that had the capability to spray some sealant over the bricks for airtightness or otherwise ensure the wall was fully unbroken, would also exercise that ability during this stage. Upon reaching the first corner again, the agent chooses another pair of corners to work on the wall between, until in the end it has verified that all are completed. At that point, the agent enters *off* mode: it follows the gradient to the outskirts of the simulation environment, to avoid interfering with any other construction that may still be ongoing, and ceases to be active. In more complicated situations, where other construction tasks on other parts of a more complicated building still remain, or the structure is subject to damage and requires constant maintenance, etc., the appropriate behavior would be to continue to *collect* rather than turning *off*.

There are a large number of additions to this basic algorithm. For instance, if an agent wants to place a brick somewhere but is prevented from doing so for too long, it will give up and move on. An agent unable to move at all for too long will send out a signal to all agents within range, on receipt of which agents will shuffle around at random for several time steps, in the hopes of breaking up a traffic jam if that was causing the problem (as frequently occurs when more than a few agents are at work). The variable *frust* determines these behaviors, as described in the section on agent state above. An agent heading to claim a brick which another picks up before the first reaches it will return to its previous goal and continue to search.

Other modifications are a necessity resulting from the grid-based framework; in a physical instantiation where positions and angles could take on arbitrary real values, the problems they were introduced to solve would not exist (though others would surely take their place). This is an issue endemic to simulations. To begin with, the movement primitives given to the agents are only single steps in the four cardinal directions. Thus if an agent wants to travel in any other direction, its actual movement each time step will still be either horizontal or vertical, with probability of each according to the desired angle of net travel. If an agent in *collect* or *seal* mode finds a place it wants to deposit a brick, it first has to line itself up with that space either horizontally or vertically, then move in in the other direction to place it, requiring an extra step and an extra sub-mode specification. Because agents could move in any direction but were not given the ability to rotate in place, if an agent ended up being one space away from the place it wanted to deposit a brick but facing in the wrong direction to do so, it had to move one space away from its real destination first before turning and moving back in for the deposition, again requiring the handling of an extra step. Similarly, if an agent ended up in the location it wanted the brick to be, it had to move horizontally or vertically away from the center for two time steps, then return to place the brick. In these three last cases, when choosing a direction to move in for these intermediate steps, the agent checked to make sure the necessary path was clear for itself and its brick (in some cases checking the x-

or y-direction first, based on its position, such that it would be more likely to choose the direction closer to radial—another complication that would be removed if movement in arbitrary directions were possible). In the last two cases, the movement was organized based on a counter that was decremented with each successive step, so that if something prevented it from moving as planned during any of those steps, it would end up placing the brick in an inappropriate place. *Seal* mode for circular walls, due to an early design choice (later relaxed) that required that bricks be adjacent horizontally or vertically at all points for 'airtightness' of walls (so that a gap was defined to exist between diagonally placed bricks), involved a considerably more complex procedure for searching for and filling in breaks in walls within the local neighborhood than that described above[3]. These last two issues may have contributed to the problems that remain at this point with occasional misplacement of bricks, as can be seen in the figures.

The granularity of the grid led to other types of problems as well. For instance, in some circumstances, the agent tried to place a brick in an empty space at (x, y), was prevented from doing so for too long for whatever reason, and moved one space further along the desired wall to look for another candidate spot; but the rounded-off integer coordinates of the closest perpendicular location the agent calculated based on its new position were again (x, y), so that the agent returned to that previous location, was again prevented from reaching its goal, and in this way became trapped in a loop, unable to move on effectively. To avoid this problem, each time an agent tried to place a brick and failed, a counter was incremented, and it moved along the wall for that many time steps before it next looked for a viable spot to place its brick; when it placed a brick successfully, that counter was reset to 0.

4 Discussion

Figure 1 shows an example of a structure created with this system. If the goal of the project was to create a system capable of constructing arbitrary two-dimensional shapes, then it was largely successful: the simpler shapes it creates are sealed, or very nearly so, though some problems remain particularly with the placement of bricks around corners; and the range of possible shapes is fairly arbitrary, though more complicated ones are executed less successfully. Figure 2 shows a variety of structures created with the system, illustrating these points.

One of the common goals in swarm intelligence is the exploitation of stigmergy, determining agent actions by external state rather than elaborate internal routines. This is an approach I tried to use to as great an extent as possible, giving agents two main variables indicating mode, each of which had a small number of states, and having them use those values plus the state of the local environment to determine behavior at each time step. The problem that gave rise to the

[3]This complication to the main algorithm now strictly remains only for historical reasons and could be simplified; I mention it for completeness, because it is a part of the system whose results are shown in this paper.

bulk of my debugging necessity was that as I tried to expand agent capabilities, something about the state of the environment would occasionally lead to the utilization of an unintended behavior at an inappropriate time; with a large number of agents operating for an extended period, these inappropriate behaviors ended up occurring with frequency sufficient to significantly interfere with the overall task, especially when misplaced bricks blocked freedom of movement where it was necessary and so on. Adding successive new behaviors to existing ones is, as is well known, an approach that can lead to great difficulties in scaling up to increasingly complex systems, as behaviors interact with each other in unexpected ways; however, I have yet to be introduced to an effective approach that avoids this problem.

I'm not certain, of the things I learned while engineering this system, which are generalizable to other similar systems. Certainly I have a clearer idea of what specific approaches prove more successful and what pitfalls there are to avoid, so that if I were to begin rebuilding the system from scratch at this point, the result would be more concise and elegant than the one I have now and easier to refine further. However, many of the less obvious problems I found it necessary to overcome were essentially artifacts of the simulation environment (a problem which is, again, characteristic of simulations in general), and an attempt to take the next major step and realize this system in hardware would obviate many of those pitfalls and introduce an entirely new, unfamiliar set.

There are certain higher-level qualitative statements I might legitimately be able to make, having to do with the use of multiple agents and the issue of their coordination. To zeroth order, if N agents are all working on a task, we might expect it to be completed in 1/Nth the time it takes a single agent to complete it. This incremental advantage is diminished as the number of agents increases, as they start getting in one another's way (as USC's Interaction Lab found). Communication between agents can help reduce this interference. In my system, I found the following things useful to communicate: First, the locations of corners; all agents needed this information, whether to determine where to place themselves as successive corners or to specify the endpoints of walls, and they obtained it much more quickly through their distributed search than they would have on their own. Second, the building mode: agents which were still engaged in the task of clearing a space to work, that encountered agents in later stages, were instructed to move on to later stages; so doing kept the team in phase and prevented some agents from working on building a wall while others worked just as hard at carrying those same bricks away. Third, logjams: when an agent was unable to move for too long, it signaled all those around it to shuffle their positions, allowing impasses to be broken; in this way, the interference that often resulted from the presence of several agents working on the same task in the same area was reduced. Note that of the aforestated effects of multiple agents, the advantages (aside from the overall increase in task completion speed due simply to the greater total amount of resources available) are all related to the use of communication, to exchange information and commands; and the disadvantages (primarily, increasing interference) are not. One further advantage that comes about from the involvement of

more agents, but does not rely on communication, can be an increased degree of completion of the overall building task; for various reasons, agents can cease to make useful progress during the course of construction, and a greater supply of agents increases the probability of the task being completed in the end. An additional factor that increases the speed of task completion when a greater number of agents work in parallel is more significant for more complicated structures with many corners: when one wall is completed and an agent switches to work on another, there is some switching time (perhaps not very significant) required as it may have to circle the structure to reach the other wall; when all walls are under construction by different agents simultaneously, this repeated switching time is avoided. Finally, this discussion has assumed that a single agent can complete the entire task by itself, as is the case in this scenario[4]; there are other tasks one could construct that are impossible without more explicit coordination of multiple agents' effort, such as moving large obstacles.

5 Future directions

Ultimately, to be useful, this system would have to be capable of constructing fully three-dimensional structures. The task will be to extend upwards the foundation that the system currently lays out, in such a way that it remains stable against collapse throughout the process.

The next step toward attaining this goal could take place likewise in a two-dimensional system, but one vertically oriented rather than horizontally; in a physical instantiation, the workspace would be bounded by two vertical parallel plates, and agents could move to arbitrary positions in the workspace by using these plates for support. Agent movement could thus be similar to that in the system described in this paper, but the structure they build would additionally have to meet the constraints of gravity (thus this step would require the writing of a physical simulator, no small task in itself). The advantage of such a framework is that it avoids the problem of the structure having to support the agents as they work on higher sections.

Further steps could include first restricting the agents to be in contact with the structure or the ground at all times, thus requiring their building strategy to work in the absence of external support but leaving the agents effectively weightless; then additionally requiring the structure to support the weight of the agents. At such a point, the further extension to the full three-dimensional system might be reasonably straightforward.

Another direction to pursue, which should be fairly simple based on the current system, would

[4] As currently implemented, there must be as many additional agents as there are corners, since those that mark the corners don't go on to build the walls. This requirement could be eliminated with 'virtual' corners, marked only by coordinates in agents' memories and not by stationary agents. However, the resulting lack of a reliable reference could lead to substantial loss of accuracy in a system with no global coordinate system, as agents' local coordinate systems drift with respect to the world over time.

be the construction of multiple-room structures. A different beacon could be used to specify the location and design of each room; agents could temporarily store each room's design as they work on it, then move on to another and get its design from the beacon there when they finish with the first; doors between rooms could be as simple as leaving a gap when close enough to the line running between the two beacons. Or the specification of the building design could be made more sophisticated, so as not to require more than a single beacon; the simplest way to do this would be to specify corners and walls separately, so that walls could be associated with arbitrary pairs of corners.

With the extensions to three dimensions and to multiple rooms both in place, the system should not be far from the capability to build multi-story, multi-room structures.

The final step will be to implement the system in hardware. The engineering problem of designing and building a physical version of this system with agents strong enough (or building materials light enough) to carry building materials to arbitrary locations, but light enough (or assembled building materials strong enough) to be supported by the structure in progress without disrupting it by their presence, will be a hugely formidable one. However, this process of simulation will both demonstrate the ultimate feasibility of a construction system of multiple autonomous agents, and point out important considerations for the design and control of the physically instantiated system.

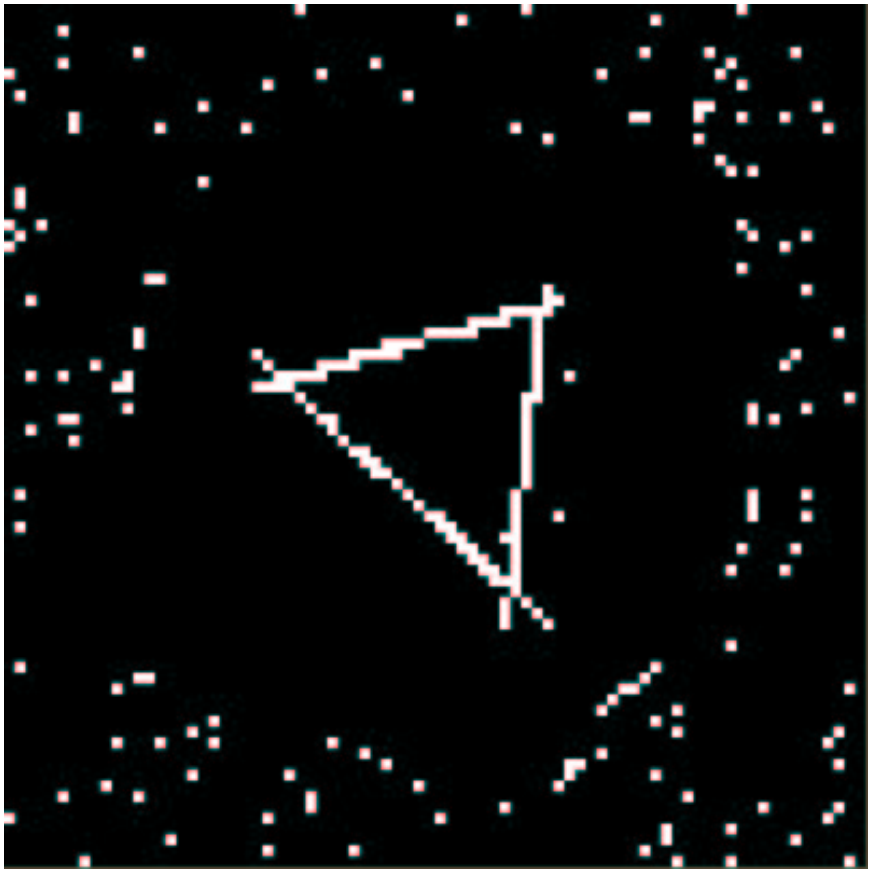

Figure 1: The result of programming the beacon with the design for a structure of three corners, each at equal radius, with 120 degrees between successive corners and straight walls between them. Only bricks and agents embodying corners are shown.

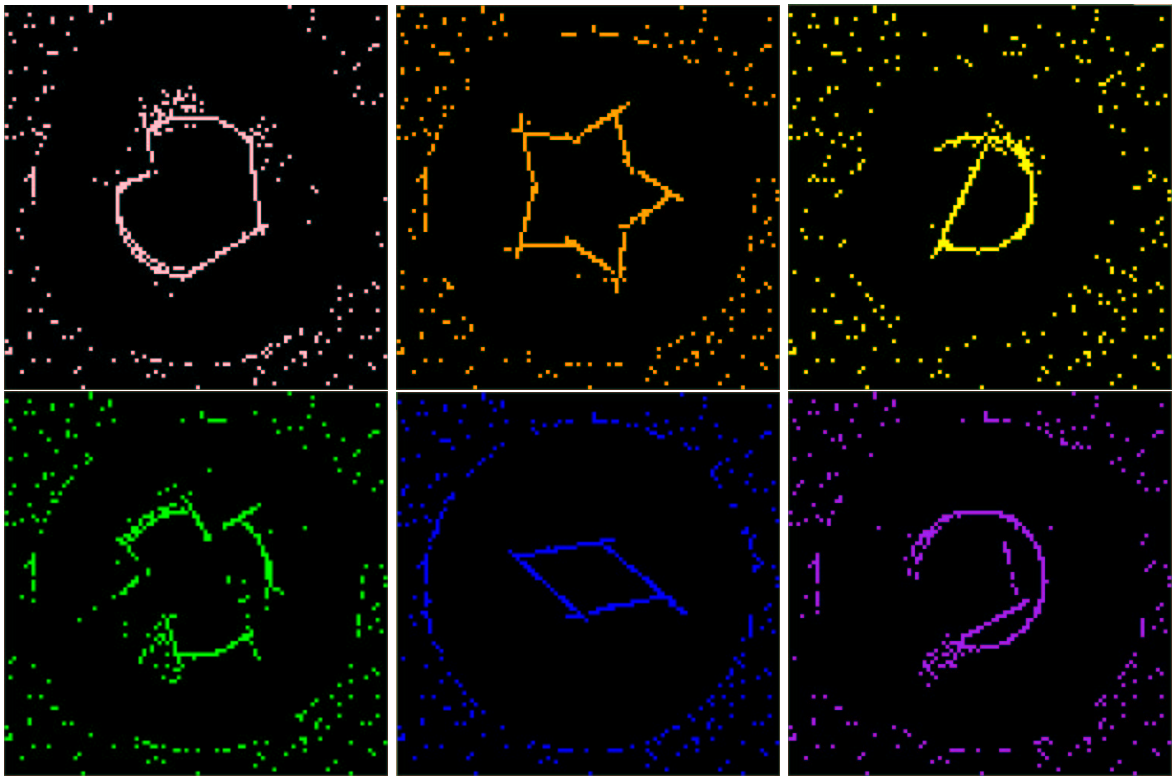

Figure 2: Several different structures constructed by the system for different building programs, demonstrating what are intended to be a pink heart, orange star, yellow moon, green clover, blue diamond, and purple horseshoe (apologies to General Mills). Again, only bricks and agents acting as corners are shown; the brick color is chosen according to the structure and has no other meaning. Simpler structures are completed more successfully, in particular because of remaining difficulties associated with the placement of bricks near corners. Difficulties with the horseshoe resulted in part because the algorithm assumes that successive corners are located with monotonically increasing angle; having several corners ordered in the opposite direction when the shape doubled back on itself caused parts of the algorithm to fail, in particular those associated with ensuring the agent is on the appropriate side of the relevant corners when entering and leaving *collect* and *seal* modes. A further significant revision of the program would take this issue into account.