



Supplementary Materials for

Designing Collective Behavior in a Termite-Inspired Robot Construction Team

Justin Werfel,* Kirstin Petersen, Radhika Nagpal

*Corresponding author. E-mail: justin.werfel@wyss.harvard.edu

Published 14 February 2014, *Science* **343**, 754 (2014)
DOI: 10.1126/science.1245842

This PDF file includes:

Materials and Methods

Figs. S1 to S12

Table S1

Captions for movies S1 to S5

References

Other supplementary material for this manuscript includes the following:

Movies S1 to S5

Materials and Methods

In the main paper we have presented a decentralized multi-agent system that automatically builds user-specified structures. Here we provide details of algorithms, proofs of correctness, and hardware implementation.

Section 1 presents an algorithm for the offline compiler that automatically generates the structpath representation of the desired target structure from the higher-level one provided by the user. Section 2 provides an algorithm that allows an arbitrary number of independent robots to build the target structure, and a proof that that algorithm will succeed in correctly completing that structure. Section 3 presents algorithmic details for the examples shown in Fig. 2D and 2E of processes with variable outcomes. Section 4 presents implementation details of the hardware prototype.

1 Structpath compiler

If robots are permitted to move freely over the structure in any way physically possible, many conflicts and traffic jams can result. The purpose of the structpath is to provide traffic rules that direct the flow of robot movement and organize their building activity so that the local information available to them is sufficient for provably correct construction. Because many different sets of such rules can be possible for a given structure, and because the process of finding one can be too computationally expensive for the simple robots we envision, a common structpath is pre-generated by an offline compiler (described in this section) and supplied to all robots at the start of construction. The output of the compiler should not be confused with a traditional planning approach: a structpath does not pre-specify a detailed series of agent actions, but instead provides only the equivalent of traffic rules, which will work for any number of robots and accommodate flexibility in the order and timing of their actions.

The structpath (Fig. S1) is a static representation of the user-specified target structure. It consists of the 2D projection of the target structure in the ground plane, annotated with the height of the desired stack of bricks at each site, and specifying a travel direction between each adjacent pair of sites (Fig. S1A,B). The direction specification thus defines a directed graph. There is a unique starting site, with desired height 1, corresponding to the seed brick.

The compiler’s task is to translate the desired target structure from whatever format is provided by the user into this structpath representation. The nontrivial part of this task is in the assignment of travel directions between adjacent pairs of sites. Restrictions on the system hardware and capabilities require certain constraints on this assignment:

- Robots are physically capable of climbing steps at most one brick high. Every site in the structpath must be reachable by robots. “Reachable” implies that there exists a path that starts at the seed, passes through the site in question, and ends at an “exit” site. An exit site is any site on the external perimeter of the final structure with desired height 1. Every step along that path must be “traversable”, i.e., the final desired heights of the stacks of

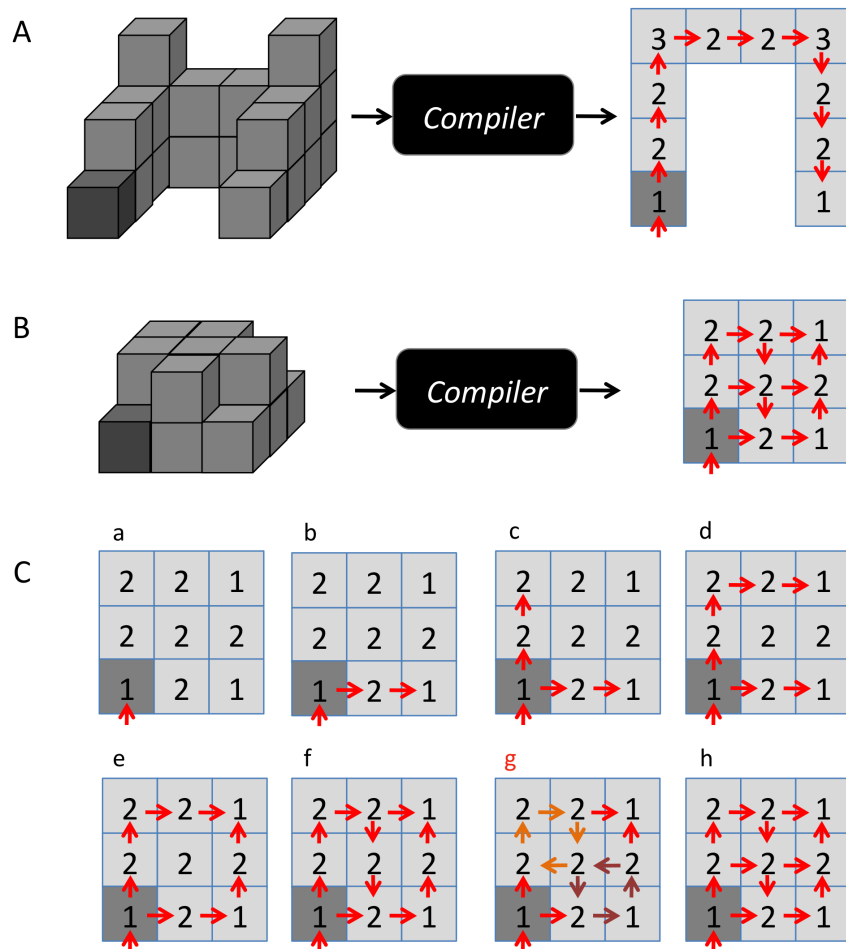


Figure S1: Structpath representation. (A, B) Example target structures and corresponding structpath representations. The seed brick is shaded; numbers give the height of the stack of bricks at each site. (A) The structure has a unique structpath. (B) The structure has many possible structpaths; one is shown. (C) A sample compilation process generating the structpath shown in (B). In step (g), an untenable labeling is rejected due to the presence of cycles (high-lighted), and the compiler backs up to the previous tenable labeling (f) and chooses a different site to continue with.

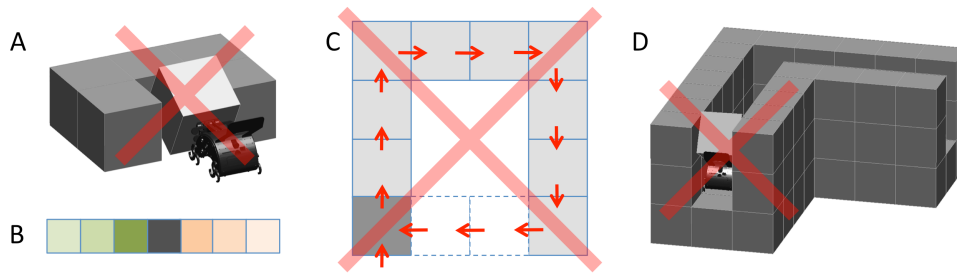


Figure S2: Restrictions on the hardware and structpath. (A) Forcing a brick into a one-brick-wide space directly between two others is a mechanically difficult operation requiring high precision, which we do not require robots to perform. (B) The result is the row rule, which requires that any contiguous row of bricks be added in some sequence building outward from an arbitrary initiation point. Here the grey brick is the first attached; bricks to each side must be attached in order of increasingly lighter tint, with no restriction on the relative ordering of bricks to left and right (accordingly shown as different colors). (C) Cycles in a structpath graph would typically correspond to violations of the row rule. In this example, adding bricks in order following the cycle means a violation of the row rule when the brick in the lower right corner is attached; continuing to add bricks in the bottom row will eventually result in a robot needing to force a brick into a site directly between two others. (D) A robot may be unable to carry a brick down a tall narrow passage only one brick wide, especially if turns are involved.

bricks at the two neighboring sites differ by at most 1. Note that there can be more than one such path through a given site, implying that there are different routes a robot may take to build at that site.

- Robots are assumed not to be able to perform the mechanically difficult task of forcing a brick into a site directly between two others (Fig. S2A). Therefore when building any contiguous row of bricks, the first brick placed can be at any site in the row, but the rest of the row must grow outward in sequence from that initiation point (the *row rule*, Fig. S2B).

In the interest of permitting a simple, consistent robot algorithm, we choose to satisfy the row rule by specifying that for a contiguous row of bricks at the same height, the first to be attached is the one closest to the seed (the root of the structpath graph), with further attachment proceeding strictly away from the root. As a result, we prohibit cycles in the graph; in most cases cycles would correspond to violations of the row rule (Fig. S2C), and the exceptions would require special-case rules for robots to handle. Note that even with this restriction, there are many possible building orders for a given structure (Fig. 2C, Movie S2).

These considerations thus motivate the structpath compiler (Alg. 1). User input is provided in any form that specifies which sites in 3D space are intended to be occupied by bricks. We consider only contiguous structures (discontinuous ones can be treated as multiple separate

Algorithm 1 Pseudocode routine for structpath compiler, either returning a valid labeling of travel directions (arrows) between neighboring sites, or identifying that no such labeling exists for the given seed site.

- 1: initialize: seed site has incoming arrow from off structure where robots enter; all other edges unlabeled
 - 2: **while** any sites have unlabeled edges **do**
 - 3: choose a site with at least one labeled and at least one unlabeled edge
 - 4: choose one unlabeled edge of that site
 - 5: add arrows in a straight line from initial site to end of contiguous row
 - 6: **if** current graph has cycles
 or any site has all edges labeled without at least one being a traversable incoming arrow and (unless an exit site) at least one being a traversable outgoing arrow **then**
 - 7: this labeling is untenable: eliminate it from the search tree
 - 8: **else**
 - 9: recurse
-

structures to be built separately). The preprocessing step for the compiler takes the 3D user input and converts it to a 2D projection into the ground plane, where each site specifies the height of the stack of bricks intended there. The seed site can be designated by the user or selected automatically from the set of height-1 sites bordering on the outside perimeter. Next the task of assigning travel directions for pairs of neighboring sites begins (Fig. S1C):

Recursively, the compiler chooses a site in the 2D projection that does not yet have directions assigned to all sides that border on neighboring sites; it chooses one such side and assigns a direction from that site to its neighbor, from that neighbor to its neighbor, and so on in a straight line to the end of that row, thus enforcing the row rule. If at any point the labeling so far has produced a cycle in the graph, or resulted in the isolation of a site (i.e., a site has had directions assigned to all four sides, and does not have at least one incoming traversable edge and at least one outgoing traversable one), then the labeling is untenable. In that case that branch of the search tree is pruned, and the search continues from the graph state before the last set of labels was added (Fig. S1C,f–h). This type of depth-first search is guaranteed to check all possible graph labelings in finite time (28). Note that for any given structure, there may be more than one viable structpath (Fig. S1B) or there may be no viable structpath at all. The compiler routine will either return the first viable structpath found, or demonstrate that no viable structpath exists for the requested structure.

The efficiency of the compiler can be improved with several heuristics and tools (28). For instance, sites can be chosen for direction assignment according to minimum distance from the root rather than at random, which can speed discovery of feasible solutions. Pruning large regions of the search space is often possible. Recording labelings that have been shown to be infeasible avoids repetition when the same labeling can be reached multiple times by different orderings of site choice. Some structures impossible to compile can be identified as such in a preprocessing step, e.g., checking for sets of sites mutually traversable among themselves but

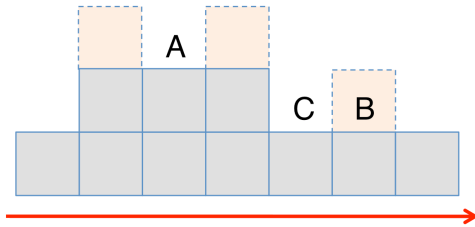


Figure S3: Dangers of attaching bricks indiscriminately (side view of linear structure). Gray squares represent bricks already present; orange ones are sites where bricks are not yet present but could physically be attached. If a robot at A adds a brick to its left, it creates an unclimbable cliff; if it adds a brick to its right, it creates an undescendable cliff. If a robot adds a brick at B, it becomes impossible to add another at C.

not reachable by robots starting from the seed. (Similarly, structures that can be compiled but are impossible for a given hardware implementation to build can be pre-identified and rejected: in particular, those with narrow corridors that robots cannot carry a brick through (Fig. S2D).) The overall difficulty of compilation depends on nontrivial properties of a structure. A brute-force approach could in general solve the problem, in the worst case, in time exponential in the number of sites (since each site can have one of two directions assigned to each of four sides); however, constraints between arrows in the same row, and characteristics that are found in typical structures of interest (e.g., thin walls), reduce the space of possibilities—like many such problems, typical-case performance appears much faster than worst-case expectations.

2 Robot algorithm

All robots are programmed with a single common behavioral algorithm (Alg. 2) always used regardless of what structure they are building. The algorithm refers to the structpath to determine allowed movement and brick attachment; to build a different structure, robots use a different structpath, without changing their behavioral algorithm.

The core of the robot algorithm lies in the determination of allowed brick attachment. Our material attachment model is purely additive, appropriate for bricks that, e.g., are mortared into place as they are added. Thus robots need to add bricks in such a way that (1) they are only ever attached at sites that the structpath specifies should be occupied by bricks in the final complete structure; (2) their attachment ordering satisfies constraints on future possible robot actions, in particular not preventing robots from moving over the structure or adding bricks at any sites meant to be occupied later. A robot that attached bricks indiscriminately at any sites ultimately meant to be occupied could easily make future progress on the structure impossible (Fig. S3).

Robots thus need to be able to gather two kinds of potentially nonlocal information before attaching a brick: (1) their location in some global coordinate system, so that all building activity is spatially coordinated; and (2) information about where bricks are currently attached and

Algorithm 2 Robot control loop for predefined structures. H_i is the desired final height of the stack of bricks at site i , h_i the current height of that stack; the robot’s current location is designated site 0. “Parent” and “child” designations refer to the directed graph provided by the structpath; “next” sites are children with $|H_i - H_0| \leq 1$ (i.e., the step to reach them is traversable) (Fig. S7A). The key checks in line 8 are illustrated in Fig. S4.

```

1: loop
2:   get new brick from supply
3:   go to structure
4:   follow perimeter counterclockwise until entry point found
5:   climb onto structure
6:   while on structure do
7:     move to any “next” site
8:     if holding brick
       and  $h_0 < H_0$ 
       and for all parent sites  $i$ : ( $h_i > h_0$  or  $h_i = H_i$ )
       and for all child sites  $i$ : ( $h_i = h_0$  or  $|H_i - H_0| > 1$ ) then
9:       move to any “next” site
10:      attach brick at site just vacated
11:  interrupt: if robot close ahead then
12:    perform collision avoidance

```



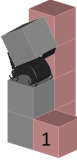

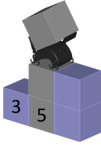
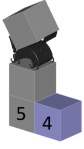
Condition	Satisfied ✓	Violated ✗
The structpath must specify a brick is intended here ($h_0 < H_0$)		
For all parent sites: either the stack of bricks must be higher than that at the attachment site ($h_i > h_0$), or the stack must be complete ($h_i = H_i$)		
For all child sites: either the stack must be the same height as that at the attachment site ($h_i = h_0$), or the intended final difference in heights between the two sites must be greater than the single step a robot can handle ($ H_i - H_0 > 1$)		

Figure S4: Three checks in line 8 of Alg. 2, and example cases where each check is satisfied or violated. Each case shows only a fragment, not a complete structure. A number at the base of a stack of bricks gives the desired final height H of that stack. Bricks at the candidate attachment site are gray; at parent sites, red; at child sites, blue.

where they are absent, sufficient to prevent additions that will be problematic for continued construction. In keeping with the stigmergy approach, robots can obtain both kinds of information by local examination of the current structure. For (1), the seed provides a unique landmark, and robots can keep track of their movement thereafter by counting sites passed as they move along the structure, which by being made of square bricks embodies a physical coordinate system. For (2), the rules about permitted brick attachment, and the fact that all robots are following those same rules, make it possible for a robot to look at just the neighboring sites and infer from that local observation where more distant bricks have and have not been attached.

Alg. 2 formally describes the behavior of a single robot and specifies the conditions (line 8; Fig. S4) evaluated by a robot when it is determining whether or not to attach a brick. Figs. S5 and S6 show how building according to this algorithm proceeds for two simple examples. As we show next, the algorithm is sufficient to allow any number of independent robots to simultaneously be building the structure, and allow the structure to emerge in many different ways, but with the guarantee that the final structure will be the one desired.

Theorem: Robots following Alg. 2 will correctly complete a target structure, working with a structpath generated by Alg. 1.

Proof: We will show four properties: (1) Robots will never build configurations of bricks that prevent their physical progress along any part of the structpath. (2) Robots will never build configurations in which they cannot physically attach bricks at any sites where bricks are desired. (3) Robots will never build configurations in which they could physically attach additional desired bricks but are prohibited everywhere from doing so purely by the (logical, not physical) restrictions of Alg. 2. (4) Different independent robots will not attach bricks at mutually conflicting sites. Together, these four properties imply freedom from deadlock: the construction process can never get stalled at any intermediate state. Along with the condition that robots only ever attach bricks at sites where bricks are ultimately intended (line 8 of Alg. 2, second condition), this result demonstrates the correctness of the algorithm, in accordance with a common strategy for analyzing the behavior of distributed algorithms (29).

Notation: h_i is the current height of the stack of bricks at site i ; H_i is the desired final height of that stack. The directed graph provided by the structpath defines “parent” and “child” sites, i.e., a site’s parents are those from which structpath arrows point in, and its children are those to which arrows point out. A “next” site of i is a child site j for which $|H_i - H_j| \leq 1$, i.e., a robot is able to move from i to j on the final structure (Fig. S7A).

(1) *Robots will never build configurations of bricks that prevent their physical progress along the structpath.* By assumption, the only configurations that prevent robot movement are unclimbable and undescendable cliffs, i.e., a change in height of 2 or more ($|h_a - h_b| > 1$) between neighboring sites a and b . If the difference between the *final* desired heights of two neighboring sites is greater than 1 ($|H_a - H_b| > 1$), the step between them is not considered traversable, and robots will never attempt to move directly between them (lines 7 and 9 of Alg. 2). Therefore robots need to ensure that they do not create situations with $|h_a - h_b| > 1$ only for neighboring pairs of sites where $|H_a - H_b| \leq 1$.

Robots always attach a brick at the parent site on the structpath from which they came (lines

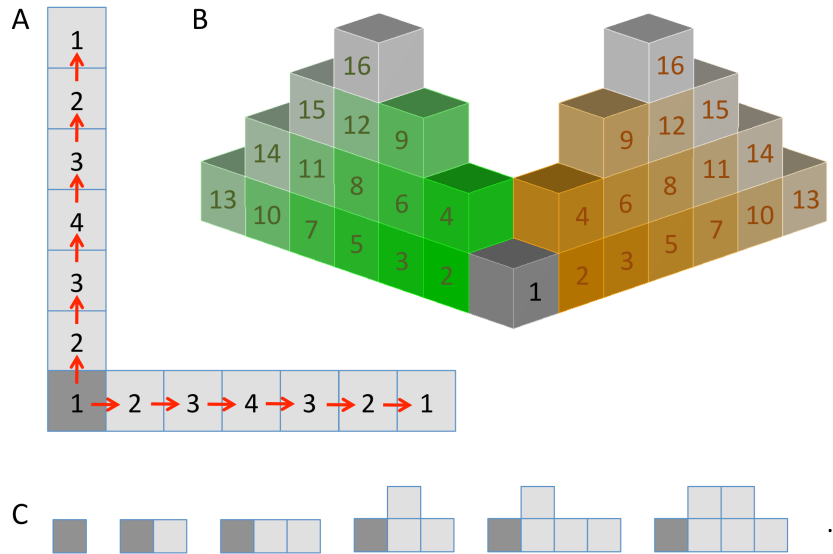


Figure S5: Building order resulting from Alg. 2 for a sample structure. (A) The structpath. The number at each site gives the desired final height of that stack of bricks. (B) The final structure. The number at each site gives the order in which each of the bricks in that wing will be attached by robots using Alg. 2. No ordering constraints exist between the bricks in one wing and those in the other. The colors of the two wings are distinct to indicate this independence; within each wing, bricks are colored with decreasing saturation to indicate ordering. Note that at each height, each contiguous row of bricks is added in a sequence starting nearest the seed and proceeding away, satisfying the row rule. (C) The partial structure for one wing after each of the first few steps in its building sequence. Bricks are added forming a growing staircase permitting robot movement at all stages.

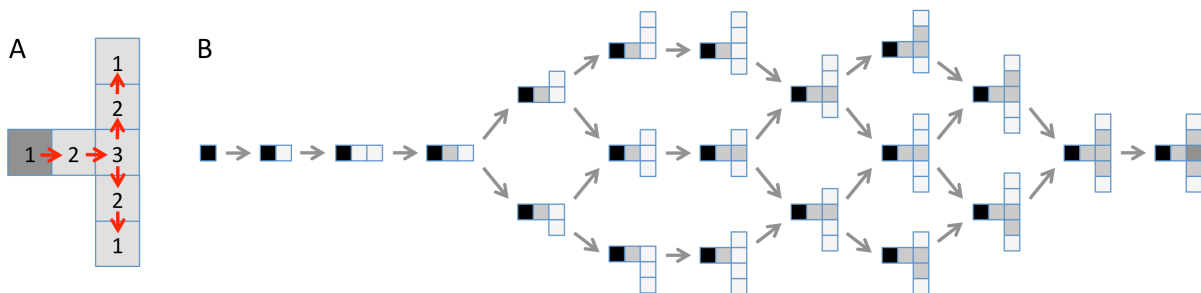


Figure S6: All possible building orders resulting from Alg. 2 for a sample structure. (A) The structpath; the number at each site gives the desired final height of the stack of bricks there. (B) All possible building sequences, moving from left (seed only) to right (completed structure). The seed is black; other sites are colored light to dark according to the height of the stack there.

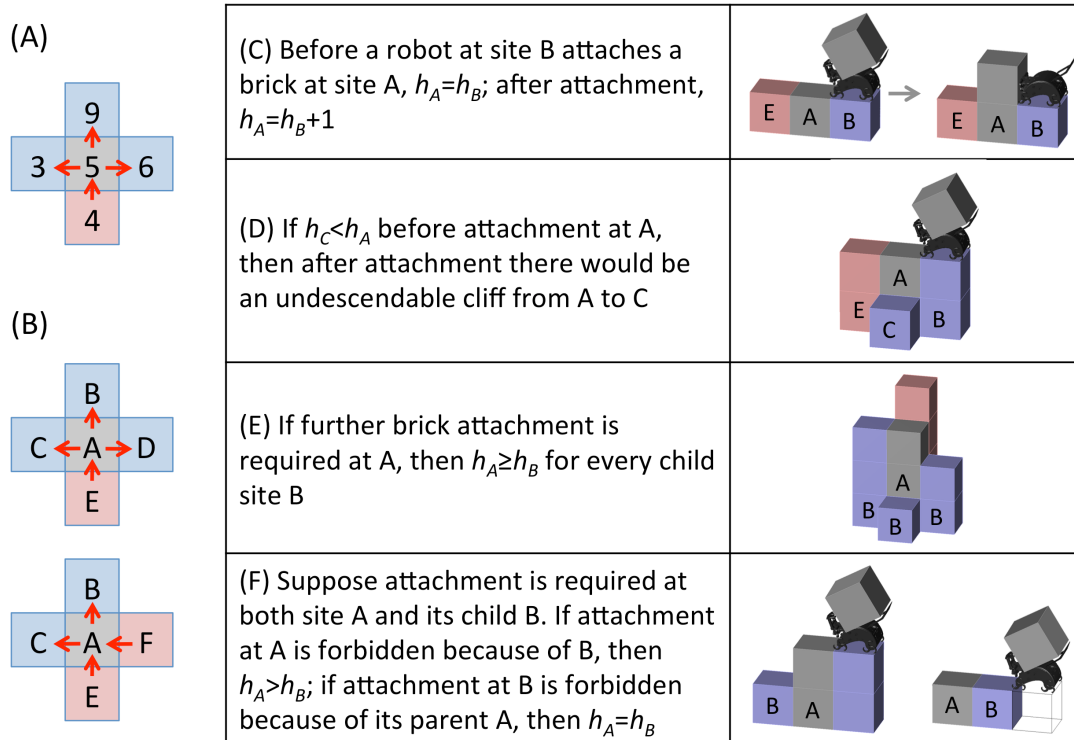


Figure S7: Auxiliary diagrams to help illustrate the proof of correctness of Alg. 2. Each panel shows only fragments, not complete structpaths or structures. Colors are defined in panel (A). (A) Parent, child, and “next” sites. In this fragment of a structpath, the central site (gray, with desired height 5) has one parent site (tinted red; desired height 4) and three child sites (tinted blue; desired heights 3, 6, and 9). Of the three child sites, only the one with desired height 6 is a “next” site, since the difference in desired heights between the central site and its other children exceeds 1. (B) Nomenclature for sites in a local neighborhood. A given site A must have at least one and may have at most three child sites B, C, D, and must have at least one and may have at most two parent sites E, F. (C–F) Sketches of situations mentioned in the proof (see text).

9 and 10 of Alg. 2). Let A be a site where a robot is about to add a brick, and B the child site the robot has moved to in order to attach at A; up to two additional child sites C and D, and up to two parent sites E and F, could neighbor A (Fig. S7B). The structpath specification guarantees that some B exists, with $|H_A - H_B| \leq 1$. Since by assumption the robot is attaching a brick at A, all conditions in line 8 of Alg. 2 must be satisfied; in particular, $h_A = h_B$ before the attachment. After the attachment, $h_A = h_B + 1$ (Fig. S7C). Thus the transition between A and B cannot pose any problems for robot movement; if the attachment causes problems, they must be between A and one of its other neighbors C–F.

For the child sites C and D, if $h_C < h_A$ or $h_D < h_A$ before attachment at A, then after attachment there would be an undescendable cliff between A and that child site (Fig. S7D). But for attachment to have been allowed at A by Alg. 2, it must be true that $|H_C - H_A| > 1$ or $|H_D - H_A| > 1$, in which case the corresponding edge was not traversable and robots would never move directly between those sites in any case. Similarly for the parent sites E and F, if $h_E < h_A$ or $h_F < h_A$ before the attachment, then attaching a brick at A would result in an unclimbable cliff; but Alg. 2 forbids attachment unless $h_E \geq h_A, h_F \geq h_A$. Thus building according to Alg. 2 will not interfere with robot movement.

(2) *Robots will never build configurations in which they are physically unable to attach bricks at sites where bricks are desired.* First note that whenever a robot attaches a brick, it does so at a parent site at the same level as itself:

Lemma 1: When a robot attaches a brick at a site A, it does so from a child site B with $h_A = h_B$.

Proof: Lines 9 and 10 of Alg. 2 specify that a robot attaches at A after moving to a “next” site B. The checks in line 8 require that either $h_A = h_B$ or $|H_A - H_B| > 1$. But in order for B to be a “next” site of A, it must be that $|H_A - H_B| \leq 1$. Therefore $h_A = h_B$.

Thus if a robot is unable to attach a brick at a site A, it could be because (a) it could not travel to any “next” site from A, or because the stack of bricks at site A is at that time (b) lower or (c) higher than that at site B.

Case (a) does not occur because the structpath compiler guarantees that every site will have at least one traversable outgoing path, and property (1) above guarantees that robots’ movement will never be restricted.

Case (b) does not occur as a result of the following lemma:

Lemma 2: If construction so far has proceeded according to Alg. 2 and further brick attachment is required at site A ($h_A < H_A$), then $h_A \geq h_B$ for every child site B (Fig. S7E).

Proof: If $h_A < h_B$, then the brick at B that makes $h_B > h_A$ could not have been legally attached: when that earlier attachment at B was being considered, the third check at line 8 of Alg. 2 would have failed, because $h_A \not\geq h_B$ for B’s parent site A, and by assumption $h_A \neq H_A$.

Case (c) can occur, but only temporarily. By assumption, $h_A > h_B$, B is a “next” site of A, and a brick is called for at A ($h_A < H_A$); therefore a brick must be called for at B, since otherwise the edge between A and B would not be traversable. Once that attachment at B occurs, we have $h_A = h_B$ and attachment at A can proceed. Attachment at B can also be temporarily impossible, if the height at B is greater than that at one or more of its own child sites; however,

for a structure of finite height, this recursion must terminate, and bricks will eventually be added to allow attachment at B and then A.

(3) *Robots will never build configurations in which they could physically attach additional bricks but are prohibited from doing so purely by the restrictions of Alg. 2.* The proof is by contradiction. Suppose that there exist sites A for which $h_A < H_A$ (i.e., construction is incomplete) but at no such site is attachment allowed according to Alg. 2. That is, for each such site, at least one of the following must be true: (a) $h_E \leq h_A$ and $h_E \neq H_E$ for at least one parent site E; (b) $h_B \neq h_A$ and $|H_B - H_A| \leq 1$ for at least one child site B.

Lemma 3 (Fig. S7F): If attachment is required at both site A and its child B, and attachment at A is forbidden because of B (i.e., the check in line 8 of Alg. 2 fails for that child site), then $h_A > h_B$. If attachment at B is forbidden due to A (i.e., the check fails for that parent site of B), then $h_A = h_B$.

Proof: In the first case, the check for attachment at A that fails because of B is ($h_A = h_B$ or $|H_A - H_B| > 1$); thus both of those two conditions must be false, and $h_A \neq h_B$. By Lemma 2, $h_A \geq h_B$. Therefore it must be that $h_A > h_B$. Similarly, in the second case, the check for attachment at B that fails due to A is ($h_A > h_B$ or $h_A = H_A$); thus it must be that $h_A \leq h_B$. Again by Lemma 2, $h_A \geq h_B$. Therefore it must be that $h_A = h_B$.

Corollary 1: For any pair of parent and child sites A and B, it cannot be the case both that attachment at A is forbidden due to B and that attachment at B is forbidden due to A.

It must be, then, that for each site A where attachment is required but currently forbidden, the forbidding is due to either a child or parent site A' where attachment is required but forbidden due to its own child or parent site A'' distinct from A. A chain of forbidden sites can then be followed from any starting point, with no backtracking. Because the structpath graph contains no cycles, this chain must eventually terminate. At that final site, attachment must be permitted, due to Corollary 1 (the forbidding cannot be due to the previous site in the chain), contradicting the original assumption that attachment is forbidden everywhere. Therefore there is no way to build a configuration of bricks consistent with Alg. 2 where all further attachment is prohibited by that algorithm.

(4) *Different independent robots will not attach bricks at mutually conflicting sites.* The checks robots perform in line 8 of Alg. 2, and the effects of the actions they take by adding bricks, are purely local, depending on and affecting the heights of the stacks of bricks only at their own location and immediately neighboring sites. Robots maintain a distance from each other along the structpath sufficient to keep their local neighborhoods from overlapping (i.e., more than one unoccupied site intervening; if two robots come together where branches of the structpath merge, one gets priority to move into the merge site and the other waits until the first has moved on). The proofs of properties (1)–(3) above apply to any structure that has been built in accordance with Alg. 2; if a distant robot adds a brick while following those rules, far from the specific sites A–F considered in the proofs and beyond the sensing range of a robot at one of those sites, the structure remains one built in accordance with those rules and the proofs still hold. Thus robots can follow Alg. 2 independently without possibility of conflict.

To prevent traffic deadlock as well as to reduce interference between robots, a robot not on

the structure whose wait time exceeds a threshold leaves the active workspace and returns after a random interval. This behavior temporarily reduces the working density of robots when their numbers exceed the amount of available work, and provides a decentralized mechanism for the collective to adjust its size to match the task (4).

We emphasize again that the structpath specifies a unique final structure but not (in general) a unique construction process. Since robots may take many different paths through a structure, the same final structure will emerge in different ways in different runs (Fig. 2C, Movie 2). The compiler and robot algorithm together ensure that the building process will obey the strong constraints on brick placement that allow unhindered robot movement and guarantee convergence of the building process to the correct structure. This is in contrast to previous algorithmic work in collective construction and self-assembly that ignores constraints such as gravity (30) or limits on robot movement and material placement (3, 31), relies on active building elements that are self-mobile and/or can communicate information (32), or uses genetic algorithms to search for agent rules without guarantees of achieving a desired target structure or proofs of correctness (33, 34). While these other frameworks may simplify the coordination requirements, they significantly increase the difficulty of translating such algorithms to practice. We demonstrate the feasibility of our algorithmic approach with a full implementation of the algorithm on multiple independent robots with purely on-board sensing.

3 Algorithms for variable outcomes

Fig. 2 includes in panels D and E examples of structures that are not rigidly predetermined like that in panel C, but nevertheless have key desired regularities. This section provides the algorithms used in those and similar examples.

3.1 Ramifying paths

Alg. 3 lets robots build a connected set of one-brick-high paths from an initial seed (Fig. 2D). A robot reaching the end of any straight path may stochastically turn to continue the path’s progress at a 90° angle; a later visitor can extend the original path beyond the junction in its original direction. A path terminates once it is extended to within a given distance from other paths, leaving enough space so that robots can follow the perimeter of the structure everywhere. To evaluate the latter condition, robots following this algorithm need to be able to detect the presence of bricks within a larger sensing range (Fig. S8), which is not necessary for other algorithms presented in this work. The main parameter affecting the shape of the resulting structure is J , the probability of starting a new side branch at the end of an existing branch.

The correctness of the algorithm, in the sense that construction following it will continually progress without encountering deadlock, is a result of the assumption that two perimeter-following robots heading opposite directions in an aisle can pass each other as long as the aisle is at least two bricks wide. The check they perform to make sure an area is empty before attaching

Algorithm 3 Robot control loop to create stochastically determined structures like those of Fig. 2D. A robot is at an “end of path” if no bricks continue straight ahead, and at a “junction” if bricks lead off to either side.

```
1: loop
2:   if not holding brick then
3:     get new brick from supply
4:   go to structure
5:   follow perimeter counterclockwise until entry point found
6:   climb onto structure
7:   while on structure do
8:     advance straight ahead to end of path or next junction, whichever comes first
9:     if junction reached then
10:      choose any path to continue down
11:     if at current end of path and more than 2 sites beyond last junction visited then
12:       with probability  $J$ , turn  $90^\circ$  left or right at random
13:     move one site forward // may step off structure
14:     if all sites in range shown in Fig. S8 are empty then
15:       attach brick at current site
16:   interrupt: if robot close ahead then
17:     perform collision avoidance
```

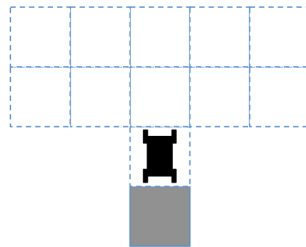


Figure S8: A robot using Alg. 3, having just stepped off the structure (grey), checks the sites marked with dotted lines. If none are occupied, it can proceed to attach a brick at its current location.

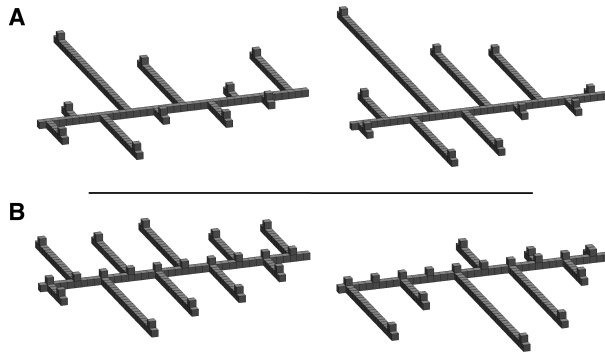


Figure S9: Sample structures built with Algs. 4 (A) and 5 (B).

bricks (Fig. S8) ensures that this minimum space is maintained. If a hardware implementation dictates that robots need wider aisles, a correspondingly larger sensing range would be required.

3.2 Straight paths with alternating side paths

The example in the previous case is simple and lightweight but, with its reliance on direct observation as the only mechanism preventing conflicting brick placements, difficult to extend to accommodate more complex structures. An alternative approach is to adopt rules that ensure at a higher level that separate branches remain sufficiently separated for the user's purposes.

A simple way to do so starts with Alg. 4, which lets robots build a single straight path with alternating side paths of varying and stochastically determined lengths (Fig. S9A), spaced at known intervals. Robots can turn off the main path at predetermined locations, and can lengthen side paths or stop their growth at their current length. Bricks are again used as physical markers to coordinate robot activity: paths are normally laid out one brick high, and a brick stacked on top of another on a side path indicates the termination of that path, meaning that no additional bricks should be added to it.

The efficiency of the system can be improved by giving the robots a small amount of dynamic memory, and extending the above use of stacked bricks to indicate completed side paths (Alg. 5; Fig. S9B). If a robot reaches the end of a side path marked with a stacked brick, it can remember the identity of that side path, circle back around to where that path leaves the main path, and stack another brick at that intersection, indicating that the side path is already complete. Other robots encountering that raised intersection then decide not to turn down the side path there, improving efficiency by avoiding unnecessary trips. Similarly, with some probability a stacked brick along the main path can be placed before construction of the corresponding side path even begins, eliminating any side path at that location altogether and increasing the variability of the structures the system produces. The length of the main path can be taken to be fixed (as it is in Algs. 4 and 5 as written) or could be determined stochastically as with the lengths of the side branches.

Algorithm 4 Robot control loop to create stochastically determined structures like those of Fig. S9A. x is the robot’s distance traveled along the main path, which can be determined by counting sites traveled from the seed; S is the desired spacing between side paths; L is the desired length of the main path. An “intersection” site is one on the main path where $x \bmod S = 0$. A “blocker” is a brick on top of another (i.e., a stack of height 2), and indicates a completed side path.

```
1: loop
2:   if not holding brick then
3:     get new brick from supply
4:   go to structure
5:   follow perimeter counterclockwise until entry point found
6:   climb onto structure
7:   while on structure do
8:     if on main path then
9:       advance along main path
10:      if robot steps off current end of main path and  $x \leq L$  then
11:        attach brick to extend main path
12:      else if at intersection site then
13:        make random choice to stay on main path or turn down side path
14:      else // on side path
15:        advance along side path
16:        if blocker encountered then
17:          advance to end of path and leave structure
18:        else if at end of path (atop final brick before stepping off structure) then
19:          with probability  $K$ , create blocker by stacking brick atop previous site along path,
          and continue off structure;
          otherwise, step off structure and attach brick to extend side path
20: interrupt: if robot close ahead then
21:   perform collision avoidance
```

Algorithm 5 Refinement of Alg. 4, using blockers on main path to (1) improve efficiency by preventing robot travel down completed side paths and (2) eliminate some side paths altogether, creating stochastically determined structures like those of Fig. S9B. x is the robot's distance traveled along the main path, which can be determined by counting sites traveled from the seed.

```

1:  $b \leftarrow 0$  //  $b$  can store the location of a completed side path to shut off
2: loop
3:   build according to Alg. 4 until intersection site reached
4:   if blocker here then // side path shut off
5:     stay on main path
6:     if  $b = x$  then // had been intending to shut off this side path, but another robot got
       there first
7:        $b \leftarrow 0$ 
8:     else if  $b \neq 0$  then // en route to shut off a side path
9:       if  $b = x$  then
10:        create blocker by stacking brick at intersection
11:         $b \leftarrow 0$ 
12:       else // have not yet reached target intersection
13:         advance along main path
14:       else // free to take either path
15:         if main path continues and side path has not yet been started then // may shut off side
           path before it begins
16:           with probability  $P$ , create blocker by stacking brick at intersection site;
           otherwise, make random choice to stay on main path or turn down side path
17:         else
18:           make random choice to stay on main path or turn down side path
19:         if on side path and blocker encountered then
20:            $b \leftarrow x$ 

```

Key parameters affecting the layout of the resulting structures are: L , the desired length of the main path; S , the spacing between side paths; K , the probability per step of ending a side path in progress; and P , related to the probability of shutting off a potential side path before it begins.

The correctness of the algorithm follows from the restricted shape of the structure (one main central path with linear side branches at known spacing) together with the unidirectional traffic flow (straight up the main path, outward along side branches, counterclockwise around the perimeter). Deadlock cannot occur since the growth of each branch proceeds straight outward from a central source, and the lack of turning in side branches prevents collisions.

Algorithm 6 Robot control loop to create hybrid structures (stochastically determined backbone plus predetermined terminal structures). Each of N possible terminal structures has an equal chance of being chosen at the end of a given side path. Fig. 2E shows an example structure built using this routine, with three possible terminal structures: (1) a small fort, (2) a small tower, and (3) a linear set of two small staircases. Note in that figure that each terminal structure is preceded along its side path by a two-brick-high segment of the corresponding length.

```

1: loop
2:   build according to Alg. 5 until reaching a blocker on a side path
3:   advance to the end of the two-brick-high segment, counting its length  $n$ 
4:   if side path extends exactly one brick beyond that segment then
5:     attach brick to extend the side path (Fig. S10A)
6:   else if side path extends exactly two bricks beyond that segment then
7:     with probability  $1/(N - n + 1)$ , extend the side path; otherwise, extend the two-brick-
       high segment (Fig. S10B)
8:   else
9:     switch to building terminal structure  $n$  according to Alg. 2
10:  if terminal structure  $n$  has only one possible robot path and at end of path still carrying
      brick then
11:     $b \leftarrow x$  // terminal structure has been completed; shut off side path

```

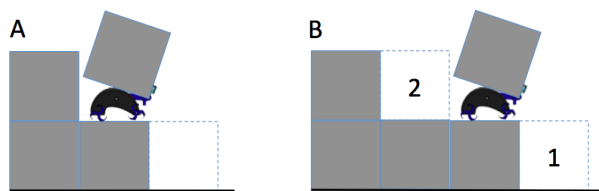


Figure S10: Local configurations of bricks referred to in Alg. 6. (A) If a path of bricks extends exactly one site beyond a stack two bricks tall (line 4), then add a brick to extend the path (dotted lines, line 5). (B) If a path of bricks extends exactly two sites beyond a stack two bricks tall (line 6), then add a brick either to extend the path (site 1) or to extend the two-brick-high segment (site 2).

3.3 Hybrid structures

The previous example can be extended to add predetermined structures at the ends of the side paths (Fig. 2E). These structures can be chosen randomly from a set of options provided by the user; the robots use different configurations of bricks to coordinate on the choice of structure to be built.

Alg. 6 outlines how the two earlier algorithms 2 and {4 or 5} can be combined. Robots build according to the stochastic Alg. 4/5 to begin with; when they reach a stack indicating the termination of a side path, they extend the 2-brick-high segment to a length corresponding to the terminal structure to be built—note how each of the three terminal structure types in Fig. 2E is preceded along the side path by one, two, or three raised bricks. The terminal structure is then built according to Alg. 2 just beyond that identifying section of the side path.

Spacing between side paths needs to be large enough to accommodate the largest terminal structure requested, in order to avoid collisions between different branches. If terminal structures have only one possible path that robots can take along the corresponding structure path, then as in Alg. 5, once they are complete a second-tier brick can be placed along the main path at the intersection to indicate completion and avoid unnecessary trips down that side path thereafter. If multiple robot paths exist within a terminal structure, it is less straightforward for a single robot to verify that the structure has been completed.

In addition to the parameters L, S, K, P discussed in the previous section, factors affecting the ultimate form of the complete structure include N , the number of available terminal structures provided; the relative probabilities for choosing among them; and the layouts of the available terminal structures themselves.

The correctness of the algorithm follows from the correctness of its two component parts (Alg. 4/5 for the stochastic backbone, and Alg. 2 for the terminal structures). So long as the spacing between side paths is specified to be large enough for the largest terminal structures, collisions between construction of structures along different branches cannot occur.

4 Hardware details

The system comprises custom-fabricated robots and bricks (Fig. S11), with 3 robots and 48 bricks available. A philosophy of simplicity motivated our design; we limited robots to a small number of actuators and sensors. Designing the robots and bricks together made it possible to incorporate passive mechanical features into both to make tasks easier (“mechanical intelligence”).

Robots ($17.5 \times 11.0 \times 12.0$ cm, 800 g) are made from 3D-printed structural parts and standard off-the-shelf components. Two actuators control differential steering (one motor drives the left two wheels, one motor the right two). A third actuator rotates the gripper to raise and lower bricks. Torsion springs hold the side prongs on the gripper closed when raised; lowering the gripper forces the prongs open, so that bricks are held securely when raised and released when lowered. Four pushbuttons on the gripper give tactile feedback regarding its position and the

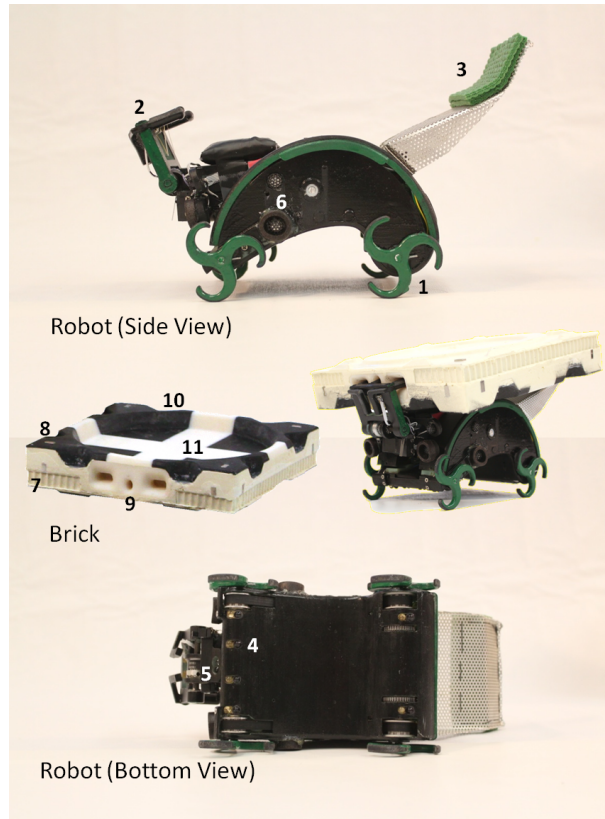


Figure S11: Robot and brick hardware (see text for details). (1) Whogs; (2) gripper; (3) shelf to support held brick; (4) IR sensors (used for ground markings); (5) IR sensor (used for cliffs); (6) ultrasound emitter/receiver; (7) rippled lower edge; (8) magnets; (9) handle; (10) notch in upper edge; (11) indentation in upper face.

presence or absence of a brick. Six infrared transceivers on the robot's underside detect black or white areas below it; a seventh identifies downward steps too high for the robot to descend, acting as a cliff sensor preventing it from falling off an edge of the structure. An accelerometer senses tilt angle as the robot climbs or descends. Four ultrasound transmitter/receivers on front and sides are used to evaluate distance to the structure while perimeter-following, and, along with a fifth emitter at rear, signal presence to other robots. A full battery charge lasts for approximately 45–90 minutes of operation (depending on the operations required).

Bricks ($21.5 \times 21.5 \times 4.5$ cm, 200 g) are molded of urethane foam. The lower outside edge is rippled to improve robots' ultrasound readings when at an angle to the edge. A total of 16 neodymium magnets are embedded in the six faces to help with alignment and attachment. A 3D-printed handle, matching the robot's gripper, is embedded in one side of each brick. The magnets and handle are incorporated into the brick during a one-step casting process. Notches in the upper edges act as intermediate steps to help robots climb (and passively align while

climbing), and a circular indentation in the upper face helps keep robots in place while turning atop a brick; matching features on the bottom of each brick help with alignment and attachment. The top surface of each brick is coated with a thin layer of impact-resistant paint, with a black-and-white cross pattern which robots can use to help keep track of their movement.

A specialized assembly of three bricks acts as the seed for a new construction project, as well as the cache from which robots retrieve new bricks. A white line on the ground guides a perimeter-following robot to climb onto the center brick of the assembly. The brick to its left serves as the cache; it lacks magnets in its upper face, so that bricks placed atop it are not held in place. The cache is manually reloaded with a new brick each time a robot takes the previous one. The third brick is the start of the structure.

With these features, robots can execute the necessary primitive operations:

- To pick up a brick from the cache, a robot faces it, backs up slightly to leave room to lower the gripper, moves forward so that the gripper is inserted into the handle, and raises the gripper.
- To attach a brick on top of another, a robot stands on top of the adjacent brick, faces the target brick, lowers the gripper and gripped brick, turns back and forth slightly to ensure alignment, backs up slightly to withdraw the gripper from the brick, and raises the gripper.
- To attach a brick at ground level, a robot stands on top of the old brick to which the new one is to be attached, turns 180°, backs up off the structure to a distance of just over one brick-length away from the perimeter (using its ultrasound sensors to adjust distance and angle), lowers the gripper and brick, moves forward so that the brick it holds collides with the structure, turns back and forth slightly so that the magnets lock the new brick into place, backs up to withdraw the gripper from the brick, raises the gripper, and turns to the right in preparation for perimeter-following.
- To follow the perimeter of a structure, a robot turns in place until the sonar unit on its left side detects an object, then moves forward, in a curving path according to the distance of the detected object: bearing left if the object is more than 15 cm away, right if the distance is less than 10 cm, straight otherwise.
- Each robot emits 42 kHz sonar pulses from its five emitters once per second, and can detect such pulses from other robots to its front or sides with its four receivers. A pulse detected within 2.5 ms of emitting one is taken to be an echo from a nearby object, and a signal from another robot otherwise.
- To keep track of its movement on the structure, a robot observes different patterns of black and white with its six main IR sensors as it translates or rotates; the accelerometer lets it determine when it starts and finishes a climb or descent.

Using these capabilities, the robots can fully implement the complete reactive construction algorithm Alg. 2. This system constitutes significant advances over a previous single-robot

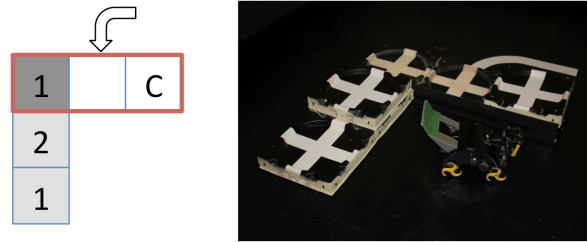


Figure S12: Test structure for hardware performance experiment. At left, a schematic showing the seed assembly (outlined in red; from right to left, the three sites correspond to the brick cache, entry point, and start of the structpath) and structpath (with three bricks to be added by the robot). At right, the completed structure with robot.

prototype that demonstrated basic capabilities of climbing and brick manipulation while atop a structure (23). Robots in the system presented here have the critical additional abilities of navigating away from a structure as well as on top of it, following its perimeter, and identifying the seed brick; adding bricks to sites around the perimeter of a structure, not only atop previously placed bricks; and detecting the presence of other robots nearby and responding to them appropriately.

The supplementary movies demonstrate the three robots carrying out Alg. 2 to build different structures highlighting different features. Movie S3 shows robots demonstrating all basic operations: following a structure perimeter using ultrasound ranging, finding the marked seed, picking up a brick, moving over a complicated structure in progress (both with and without holding a brick, and including climbing up and down steps), detecting one another's presence and responding by waiting for the one further ahead in the loop to move on, and attaching bricks to the structure both atop existing parts of the structure and on the ground. Movie S4 shows a case where the structpath includes a branching point, where robots take multiple paths, and leave the structure at different points. Movie S5 demonstrates the reactive nature of the algorithm. A robot acts with no awareness of what has previously been built (by itself or by other agents). Accordingly, addition and removal of bricks by an external agent, in ways consistent with the building rules, requires no special response from the robot and has no effect on the eventual successful completion of the structure.

To evaluate the performance of these research prototype robots, we conducted an experiment in which one robot repeatedly built a small test structure. The robot constructed a three-brick staircase (Fig. S12) ten times in succession in an unbroken four-hour period. Table S1 shows the number of times different primitive actions were performed during this sequence, and how many errors occurred. Minor errors (e.g., an alignment operation the robot was unable to complete on its own) were corrected by hand and the robot allowed to proceed; in six cases, errors that would interfere with the robot's ability to continue were handled by returning the robot to the ground, resetting its state without a brick, and continuing the experiment from that point. These

Task	Number of attempts	Successes	Failures
Pick up brick	36	36	0
Attach brick:			
On ground	20	18	2
Atop structure	10	9	1
While on ground:			
Follow perimeter to entry point	41	41	0
Detect entry point	41	36	5*
While atop structure:			
Align atop brick	342	341	1
Align midway between bricks	70	69	1
90° turn atop brick	240	236	4
Detect brick edge	36	34	2

Table S1: Results of hardware performance experiment, described in text and Fig. S12. *: If the robot misses the marking indicating the entry point, it makes an unnecessary additional circle around the structure, not requiring intervention.

six errors were due to (1) failure to report a low battery, (2) dust accumulated on a sensor, (3) internal I2C bus communication failure, (4) alignment error leading to the robot falling off the structure, (5–6) two edge detection errors where the robot mistakenly identified a brick as the end of the existing structure. The time to complete the structure was 20 ± 5 minutes, with a total distance traveled of more than 80 meters, requiring slightly less than three full battery charges. When moving (i.e., including alignment and turning actions performed while advancing, but excluding picking up or attaching bricks or turning to do so), the average robot speed was 0.76 meters per minute while traveling on the structure and 0.74 meters per minute while off it. Picking up a brick took 40 ± 47 seconds, and attaching one took 46 ± 18 seconds atop the structure and 27 ± 7 seconds on the ground.

In the clip shown in Movie S3, robots pick up and attach bricks 9 times, perform alignment and turning operations atop bricks 278 times, and travel a total distance of over 30 meters (including 52 ascents and descents) in 23 minutes, all with no errors. In the clip shown in Movie S4, robots pick up and attach bricks 16 times, perform alignment and turning operations atop bricks 145 times, and travel a total distance of over 20 meters in 31 minutes, all with no errors.

Bridging the gap from this research prototype to a full production system that builds large-scale structures with complete autonomy would involve several issues. An industrial redesign process would loosen legacy constraints and provide other improvements (e.g., replacing our 3D-printed prototype with more mechanically robust materials to reduce wear and tear, which currently results in a slow loss of accuracy over testing and debugging cycles) that would increase reliability over that of the current system. Additional robot capabilities, both in sensing

and actuation, would be needed in order to deal autonomously with rare but serious failure modes (e.g., complete robot breakdown, misaligned bricks) which would inevitably occur in a sufficiently large system even if the probability of any such event is very low. Robots would need to be able to recharge on their own (in the current system, a robot that detects a low battery state leaves the workspace to indicate to the supervising human that its battery pack needs replacing). The cache would need to provide a sufficient supply of bricks without requiring manual reloading. While addressing these challenges will require considerable additional work, our research prototype demonstrates the feasibility of implementing autonomous decentralized multi-robot construction, presenting a system in which multiple independent climbing robots use onboard sensing with no centralized vision or position reference to flexibly build user-specified structures.

Movie S1: Robots in simulation build the tower of Fig. 1D. Robots are independently controlled, and aside from the static structpath, have access only to local information, with no knowledge about the overall current structure state, number of robots (here, 20), actions taken by others, etc.

Movie S2: A predetermined final structure can be built via any of many possible paths. Two panels show different orderings in which bricks are added in independent simulation runs, both ultimately producing the structure of Fig. 2C; robots are not shown.

Movie S3: Hardware demonstration of three robots working on a castle-like structure, starting from a partially complete state (Fig. 4A). Robots are autonomous, independently controlled, and have strictly onboard, short-range sensing. Total elapsed time in this clip is 23 minutes.

Movie S4: Hardware demonstration of three robots building a branching structure (Fig. 4B). Multiple different paths along the structure exist for the robots to follow from entry to exits. Total elapsed time in this clip is 31 minutes.

Movie S5: Hardware demonstration of reactive behavior. Robots respond to local situations they encounter; changing the structure (in ways consistent with the structpath rules) while they work requires no special response and has no effect on the eventual successful completion of the structure. Total elapsed time in this clip is 23 minutes.

References

1. J. S. Turner, A superorganism's fuzzy boundaries. *Nat. Hist.* **111**, 62–67 (2002).
2. S. Camazine *et al.*, *Self-Organization in Biological Systems* (Princeton Univ. Press, Princeton, NJ, 2001).
3. G. Theraulaz, E. Bonabeau, Coordination in distributed building. *Science* **269**, 686–688 (1995). [doi:10.1126/science.269.5224.686](https://doi.org/10.1126/science.269.5224.686) [Medline](#)
4. J. Werfel, thesis, Massachusetts Institute of Technology (2006).
5. P. W. Anderson, More is different. *Science* **177**, 393–396 (1972). [doi:10.1126/science.177.4047.393](https://doi.org/10.1126/science.177.4047.393) [Medline](#)
6. Y. Bar-Yam, *Dynamics of Complex Systems* (Westview, Boulder, CO, 1997).
7. M. Mitchell, *Complexity: A Guided Tour* (Oxford Univ. Press, New York, 2009).
8. B. Khoshnevis, Automated construction by contour crafting—related robotics and information technologies. *Autom. Constr.* **13**, 5–19 (2004). [doi:10.1016/j.autcon.2003.08.012](https://doi.org/10.1016/j.autcon.2003.08.012)
9. S. Wismer, G. Hitz, M. Bonani, A. Gribovski, S. Magnenat, Autonomous construction of a roofed structure: Synthesizing planning and stigmergy on a mobile robot. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (7–12 October 2012, Vilamoura, Algarve, Portugal), pp. 5436–5437. [doi:10.1109/IROS.2012.6386278](https://doi.org/10.1109/IROS.2012.6386278)
10. F. Nigl, S. Li, J. E. Blum, H. Lipson, Autonomous truss reconfiguration and manipulation. *IEEE Robot. Autom. Mag.* **20**, 60–71 (2013). [doi:10.1109/MRA.2012.2201579](https://doi.org/10.1109/MRA.2012.2201579)
11. Q. Lindsey, D. Mellinger, V. Kumar, Construction of cubic structures with quadrotor teams. In *Robotics: Science and Systems VII*, H. Durrant-Whyte, N. Roy, P. Abbeel, Eds. (MIT Press, Cambridge, MA, 2012), pp. 177–184.
12. J. Willmann, F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio, M. Kohler, Aerial robotic construction: Towards a new field of architectural research. *Int. J. Archit. Comput.* **10**, 439–460 (2012). [doi:10.1260/1478-0771.10.3.439](https://doi.org/10.1260/1478-0771.10.3.439)

13. A. Bolger *et al.*, Experiments in decentralized robot construction with tool delivery and assembly robots. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (18–22 October 2010, Taipei), pp. 5085–5092.
[doi:10.1109/IROS.2010.5651495](https://doi.org/10.1109/IROS.2010.5651495)
14. J. Worcester, J. Rogoff, M. A. Hsieh, Constrained task partitioning for distributed assembly. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (25–30 September 2011, San Francisco), pp. 4790–4796.
[doi:10.1109/IROS.2011.6095046](https://doi.org/10.1109/IROS.2011.6095046)
15. R. A. Brooks, A. M. Flynn, Fast, cheap and out of control: A robot invasion of the solar system. *J. Br. Interplanet. Soc.* **42**, 478–485 (1989).
16. P.-P. Grassé, La reconstruction du nid et les coordinations inter-individuelles chez *Bellicositermes natalensis* et *Cubitermes* sp. La théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes Soc.* **6**, 41–81 (1959). [doi:10.1007/BF02223791](https://doi.org/10.1007/BF02223791)
17. Y. Shoham, M. Tennenholtz, On social laws for artificial agent societies: Off-line design. *Artif. Intell.* **73**, 231–252 (1995). [doi:10.1016/0004-3702\(94\)00007-N](https://doi.org/10.1016/0004-3702(94)00007-N)
18. J. Maynard Smith, The theory of games and the evolution of animal conflicts. *J. Theor. Biol.* **47**, 209–221 (1974). [doi:10.1016/0022-5193\(74\)90110-6](https://doi.org/10.1016/0022-5193(74)90110-6) [Medline](#)
19. See supplementary materials on *Science* Online.
20. R. Brooks, A robust layered control system for a mobile robot. *IEEE J. Robot. Autom.* **2**, 14–23 (1986). [doi:10.1109/JRA.1986.1087032](https://doi.org/10.1109/JRA.1986.1087032)
21. R. Arkin, *Behavior-Based Robotics* (MIT Press, Cambridge, MA, 1998).
22. R. D. Quinn *et al.*, Insect designs for improved robot mobility. In *Proceedings of the 4th International Conference on Climbing and Walking Robots* (24–26 September 2001, Karlsruhe, Germany), pp. 69–76.
23. K. Petersen, R. Nagpal, J. Werfel, TERMES: An autonomous robotic system for three-dimensional collective construction. In *Robotics: Science and Systems VII*, H. Durrant-Whyte, N. Roy, P. Abbeel, Eds. (MIT Press, Cambridge, MA, 2012), pp. 257–264.

24. S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics* (MIT Press, Cambridge, MA, 2005).
25. E. Bonabeau, M. Dorigo, G. Théraulaz, *Swarm Intelligence: From Natural to Artificial Systems* (Oxford Univ. Press, New York, 1999).
26. A. J. Ijspeert, A. Martinoli, A. Billard, L. M. Gambardella, Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Auton. Robots* **11**, 149–171 (2001). [doi:10.1023/A:1011227210047](https://doi.org/10.1023/A:1011227210047)
27. M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: A review from the swarm engineering perspective. *Swarm Intell.* **7**, 1–41 (2013). [doi:10.1007/s11721-012-0075-2](https://doi.org/10.1007/s11721-012-0075-2)
28. S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach* (Prentice Hall, Upper Saddle River, NJ, 2009).
29. N. Lynch, *Distributed Algorithms* (Morgan Kaufmann, San Francisco, 2009).
30. J. Werfel, R. Nagpal, Three-dimensional construction with mobile robots and modular blocks. *Int. J. Robot. Res.* **27**, 463–479 (2008). [doi:10.1177/0278364907084984](https://doi.org/10.1177/0278364907084984)
31. S. Yun, M. Schwager, D. Rus, Coordinating construction of truss structures using distributed equal-mass partitioning. In *Proceedings of the 14th International Symposium on Robotics Research* (31 August–3 September 2009, Lucerne, Switzerland), pp. 607–623. [doi:10.1007/978-3-642-19457-3_36](https://doi.org/10.1007/978-3-642-19457-3_36)
32. A. Grushin, J. A. Reggia, Automated design of distributed control rules for the self-assembly of prespecified artificial structures. *Robot. Auton. Syst.* **56**, 334–359 (2008). [doi:10.1016/j.robot.2007.08.006](https://doi.org/10.1016/j.robot.2007.08.006)
33. E. Bonabeau, S. Guérin, D. Snyers, P. Kuntz, G. Theraulaz, Three-dimensional architectures grown by simple ‘stigmergic’ agents. *Biosystems* **56**, 13–32 (2000). [doi:10.1016/S0303-2647\(00\)00067-8](https://doi.org/10.1016/S0303-2647(00)00067-8) [Medline](#)
34. S. von Mammen, C. Jacob, G. Kókai, Evolving swarms that build 3D structures. In *Proceedings of the IEEE Congress on Evolutionary Computation* (2–5 September 2005, Edinburgh), pp. 1434–1441. [doi:10.1109/CEC.2005.1554858](https://doi.org/10.1109/CEC.2005.1554858)