Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2005-024 April 8, 2005
AIM-2005-011

Construction by Robot Swarms Using Extended Stigmergy

Justin Werfel, Yaneer Bar-Yam, Radhika Nagpal

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Abstract

We describe a system in which simple, identical, autonomous robots assemble two-dimensional structures out of identical building blocks. We show that, in a system divided in this way into mobile units and structural units, giving the blocks limited communication abilities enables robots to have sufficient global structural knowledge to rapidly build elaborate pre-designed structures. In this way we extend the principle of stigmergy (storing information in the environment) used by social insects, by increasing the capabilities of the blocks that represent that environmental information. As a result, arbitrary solid structures can be built using a few fixed, local behaviors, without requiring construction to be planned out in detail.

This work was supported by NSF grant EIA-0130391.

1 Introduction

In this paper we describe a system for automated construction, using two kinds of entities, robots (specialized for movement and manipulation) and blocks (specialized for structural qualities). All agents within each class are identical. We present an algorithm by which robots can create close-packed structures if blocks are entirely inert, and then describe advantages that result from adding sophistication to the blocks and removing it from the robots. With no direct communication between robots, a set of simple, fixed, local behaviors is sufficient to generate any user-specified contiguous two-dimensional structure without internal gaps.

The ability to automate construction would be useful particularly in settings where human presence is dangerous or problematic; for instance, robots could be sent ahead of human travelers to underwater or extraterrestrial environments, to create habitats in advance. Swarm approaches, involving larger numbers of simpler robots rather than one or a few with greater sophistication, have advantages for such a goal, in particular with respect to decentralization and robustness. Such systems can typically absorb the loss of many components without a significant impact on task completion; similarly, they tolerate components acting in no exact specified order, which is useful because of the difficulty of preplanning robot behavior in detail in uncertain environments.

Swarms of building robots naturally bring to mind, and can draw inspiration from, social insects such as ants and bees. These insects use stigmergy to guide their building activities: e.g, ants deposit materials in ways influenced by their immediate surroundings, and in turn influence those surroundings by depositing material. In this way the insects communicate implicitly through manipulation of the environment.

Stigmergy is a powerful and simple tool, with limitations: while it can be used to produce structures with given qualitative characteristics [1, 11], it does not easily allow the consistent production of *specific* structures (potentially arbitrary and complicated ones); and no general principle has been described for taking a particular desired structure and extracting a set of low-level behaviors that building agents can follow to produce it. Here we propose giving the building material some ability to communicate, and show that as a result of doing so, the robots no longer need ability to determine local structure geometry, the speed of building scales more favorably with both the number of blocks and the number of robots, and the system naturally extends to allow construction of arbitrary, user-specifiable, finite shapes.

In §2 we review related work, and in §3 give our assumptions. §4 presents an algorithm that robots can use with inert blocks; §5 describes the case where blocks can communicate. §6 compares the performance of the two cases theoretically and through simulation. §7 describes error correction and disassembly of structures.

2 Previous work

Several previous papers address topics related to construction, though not with the goal of producing user-specified building designs. For example, [4, 6, 13] study aspects of the use of communication between multiple robots to increase their effectiveness on specific tasks. [2] discusses issues of mechanical design for robots meant to build

arches, towers, and walls, using extruded foam. Others deal with minimization of capabilities for robots performing construction-related tasks, such as arranging pucks into walls [7] or clearing an area [8]. [14] does describe a limited framework for arranging blocks into arbitrary lines and curves; it considers only inert building blocks and requires more capable robots accordingly.

Our approach is particularly motivated by research in two areas: stigmergy-based construction by insects and insect-inspired robots, and self-reconfigurable modular robotics [12, 15]. Work in this latter area has produced hardware systems with many capabilities we will take advantage of here, including connections that self-adjust so that inexact alignment is corrected, and communication links between physically attached modules. Modular robot algorithms typically require that all modules remain connected at all times, and that individual modules be capable of mobility, neither of which are appropriate for automated construction applications. Very recently, modular robotics researchers have begun to describe hardware systems for automated assembly of structures [3, 10]. These systems are based on inert modules, and have not yet explored issues of specifying distributed robot behavior and creating complex user-specified structures. We believe that results presented here are also applicable to these systems.

3 General problem and assumptions

We consider the problem where mobile robots and caches of building blocks are deployed at random into an obstacle-free workspace, along with a marker indicating the location for the start of construction. The goal is for robots to collect blocks from the caches and arrange them into some desired structure starting at the marker (Fig. 1). The marker and caches send out distinct long-range signals which robots can use to navigate to them. We will take the marker to be an object with the same geometry and attachment mechanisms as a block, and have it serve as a 'seed' to which blocks can first be attached.

We will work with square blocks, to be assembled in the horizontal plane. An important constraint is that a block can be added to the growing structure if and only if the potential attachment site has at least two adjacent sides open; otherwise there is insufficient room to maneuver to add the new block (Fig. 2). Though a system has been demonstrated in which cubical blocks can be slid into spaces like that at (D) in that figure [10], the task of mechanical design will be simplified and the precision with which robots must operate reduced if we maintain this constraint. In particular, if we prevent gaps like that at (D), then situations where a robot needs to maneuver a block down a longer 'tunnel', like that around (C), will also be prevented.

Robots can move in any direction in the plane, alone or while holding a block. They can detect the presence of other robots, for collision avoidance, either by active short-range signals or by passive perception. Once in the immediate vicinity of the growing structure, they can follow its perimeter, and observe the blocks in their immediate neighborhood.

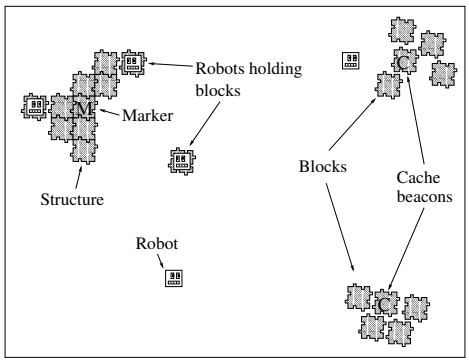

Figure 1: Overview of the system we describe here. As the structure is assembled, it forms a lattice with an implicit coordinate system, of which the marker can be taken to be the origin.

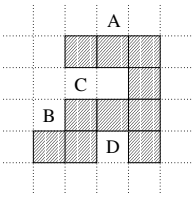

Figure 2: Examples of valid and invalid prospective docking sites for a sample structure (shaded squares represent grid sites already occupied by blocks). A new block can be attached at A or B; C and D are too spatially constrained to allow a block to be maneuvered into position.

4 Inert blocks

One of the primary motivations for separating system elements into mobile robots and passive blocks, rather than having a single class of mobile blocks that undergo self-reconfiguration, is that once a structure is in place, mobility of the building elements is unnecessary. Optimizing the building elements for their structural properties is likely to make them more effective and cheaper. The most extreme version of this strategy is for blocks to be completely inert, mass-producible, and perhaps no more sophisticated than bricks.

4.1 Algorithm for robots

One aspect of the problem is to avoid unwanted gaps. To do so, it is necessary to avoid situations where two blocks end up in the same row, unless they are adjacent. Since robots are assumed to have access only to local information, block placement must

Algorithm 1 Pseudocode procedure for assembly of a structure of inert blocks, by a single robot.

```
loop
  get block from cache
  go to structure perimeter
  passed-left-corner ← false
  while still holding block do
    if at-right-corner or (at-left-corner and passed-left-corner=true) then
      attach block here
    else if at-left-corner then
      passed-left-corner ← true
    follow perimeter counterclockwise
```

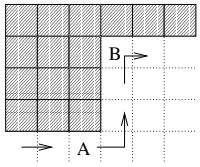

Figure 3: A: A left corner. B: A right corner. Arrows indicate the path a perimeter-following robot would take.

be strongly restricted based only on local configurations of existing blocks, in such a way as to prevent such situations. One strategy to accomplish this is to build up the structure by layers, adding successive new blocks in a row along an edge of the existing structure, always starting at the same end and going in the same direction.

Algorithm 1 outlines a procedure by which a robot can build a close-packed structure out of inert blocks. It can be summarized by the rule that a block can only be placed at a 'right corner', or at a 'left corner' (Fig. 3) which immediately follows another left corner. Intuitively, attaching a block at a right corner corresponds to adding it at the end of an existing incomplete row; attaching at a left corner which follows a left corner corresponds to starting a new row; attaching at a left corner which follows a right corner would result in two nonadjacent blocks in the same row, which is not allowed.

We can prove by induction that this algorithm guarantees a close-packed structure. Consider a robot following the perimeter of a valid structure (no two nonadjacent blocks in the same row). As shown in Fig. 4, the previous corner (P) and the next corner (N) on the robot's tour may each be either a left or right corner:

1. If both P and N are left corners, then attaching a block at N gives a valid structure.

2. If P is a left corner and N is a right corner, then attaching a block at N gives a valid structure.

3. If P is a right corner and N is a left corner, then attaching a block at N gives an

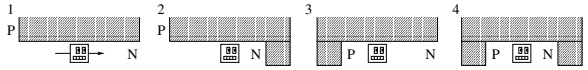

Figure 4: Schematic drawing of a robot following the structure perimeter counterclockwise. N and P are the next and previous corners, respectively, on the robot's tour. Depending on the structure geometry, N and P may each be either a left or right corner, as enumerated here and in the text.

invalid structure.

4. By hypothesis, both cannot be right corners, since that would entail two nonadjacent blocks in that row.

Thus restricting robots to attaching blocks at right corners, or left corners which immediately follow other left corners, will maintain the validity of a structure. Finally, the structure is valid at the earliest stage, when it consists of a single block. This completes the induction.

Figure 5 shows a structure in the process of being built by this algorithm, with ten robots. With multiple robots active simultaneously, situations can arise which require minor extensions to the algorithm. For example, gridlock can occur if robots are crowded too densely around the structure perimeter, such that none can advance nor attach a block where they are. One solution is for robots to give up and leave the structure vicinity temporarily, with probability according to how long they've been unable to move. Another complication is that two robots very close together may decide at almost the same time to attach blocks, such that either attachment would be allowed on its own, but if one block were attached the other should not be. If, when a robot decides to attach, it signals any robots following within two spaces behind it to reset their 'passed-left-corner' flags to false, this problem will be avoided.

This use of inert blocks, like other applications of stigmergy, requires robots to be able to reliably recognize geometric features of the structure. That nontrivial task could be made easier, e.g., by incorporating a rail into the design of the blocks such that when a robot reaches the structure, it hooks on to the rail for its perimeter traversal, explicitly registering the turning of corners. Another limitation is that, while Algorithm 1 can produce infinite sheets without holes, more complex structures are not in general possible in this simplest framework, without, e.g., distinct landmarks or ways of marking particular blocks. Such approaches could be used to extend the class of structures a distributed team of robots could assemble out of inert blocks. However, it is fruitful instead to transfer some of the planning power into the blocks themselves.

5 Communicative blocks

With computation as inexpensive as it is and is becoming [5], we can put limited sophistication into the blocks. Robots have to communicate through external signaling, which can be unreliable, subject to multipath effects and other complications, and may

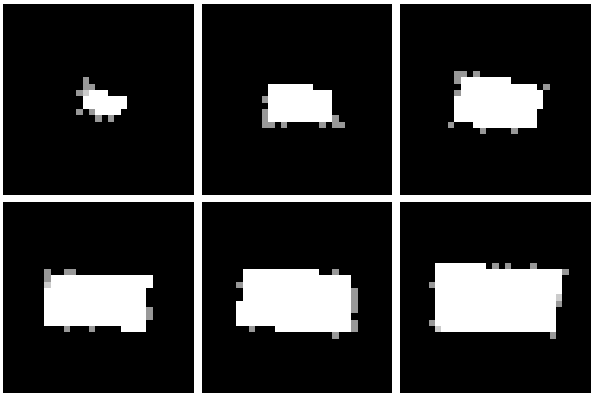

Figure 5: Construction of a sample rectangular structure, showing successive snapshots during the process of construction (left to right, top to bottom). White: blocks; gray: robots; light gray: robots in the process of attaching blocks; black: empty cells.

not scale up well with large numbers of robots. Blocks, however, have a direct physical connection to each other once they are connected to the structure; that link can be the basis for reliable, unambiguous, rapid communication. Modular robots typically use such data connections along with their physical links.

A block attached to the structure can immediately obtain from its neighbors its position in a common coordinate system, along with information about the current and desired final structure. Blocks along the perimeter of the structure and robots traversing the perimeter can also communicate, with lower bandwidth; this communication is very short-range, avoiding problems associated with signaling over distances, interference when many robots are active, and ambiguity in which agents are signaling. If the block design includes a physical rail to facilitate perimeter-following, as mentioned as a possibility above, this communication could be conducted via that attachment.

5.1 Algorithms for blocks and robots

This communication makes global information about the structure available to robots anywhere along its exterior. As a result, robots need keep no record of the geometry they've passed during their perimeter traversal; instead, the structure itself can designate sites available or not for block attachment.

The basic idea for how to guarantee a close-packed structure remains the same as for inert blocks, but more sites are now available for potential attachment. The first block added to a given row can be located anywhere along its length, not just at one end. Further construction in that row can proceed in both directions from the first block, not just in the one direction possible with inert blocks.

Algorithm 2 outlines procedures to be followed by both robots and blocks to build a close-packed structure. Robots simply circle until they find a site that gives them permission to attach. The structure ensures that no two nonadjacent blocks will be

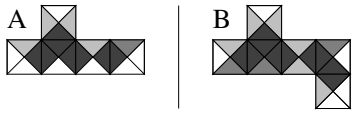

Figure 6: States for block sides in Algorithm 2 are shown here as white (open), light gray (corner), medium gray (closed), dark gray (done). A: Block states for a sample structure. The south sides of blocks in the lower row, in particular, are 'open', since no blocks have been attached in the next row down. B: When a block is attached in the next row down, the south side of the adjacent block is marked 'done'; that of the next block is marked 'corner', allowing attachment; and those of other blocks in the row are marked 'closed', preventing two blocks from being attached in the same row unless adjacent. The newly attached block has its north side marked 'done' (since that side is directly attached to the structure), its east side marked 'open' (since the east side of the block at north is 'open'), its west side marked 'corner' (since the west side of the block at north is 'done'), and its south side marked 'open' (since that side is neither attached to a block nor adjacent to a side attached to a block).

attached in a single row by having each block maintain a state for each of its sides, associated with where blocks have already been attached in the row the side borders (Fig. 6). Before any blocks have been attached in a given row, blocks in the adjoining row will have their corresponding side marked 'open'; attaching a block will send a message down the adjoining row, marking sides 'closed' to lock out further attachment in that row, except adjacent to existing blocks in the row, where sides are marked 'corner'. Sides bordering an actual attachment are in state 'done'.

As with Algorithm 1 for inert blocks, certain extensions are necessary to ensure correct operation when multiple robots are active at the same time. Most importantly, if messages propagate through the structure with finite speed, the algorithm needs to ensure that only one robot can receive permission to attach a block in an 'open' row at a time. This can be done, for instance, by sending a preliminary message down the row to make sure that all blocks in the row know that attachment at a site is being considered, and waiting until all blocks in the row have confirmed that they will not grant other robots permission to attach, before extending permission to the first one. This consultation with the row is only necessary for the first block attached in an 'open' row; thereafter the previously 'open' sides in the row take on other states, and can allow or forbid robots to attach without reference to any other blocks.

5.2 Compound structures

Unlike with inert blocks, the algorithm readily extends to structures of finite extent and arbitrary shape. We still forbid internal gaps, and require that any spaces between blocks in the desired final structure be at least two grid cells wide, so that perimeter-following can continue unimpeded. Essentially the same procedure as Algorithm 2 allows for deliberate infoldings of the perimeter. The difference is that two blocks may now be attached in the same row without being adjacent, provided that they are

Algorithm 2 Pseudocode procedure for assembly of a close-packed structure of communicative blocks, by a single robot. Each side of each block maintains a state, from the set {open, closed, corner, done}. The function f is defined as f(done) = corner, f(open) = open, f(anything else) = closed.

A: Blocks

```
if just finished getting attached to structure then
  for all sides S do {see Fig. 6}
    if S is directly attached to structure then
      state(S) ← done
    else if S is adjacent to a side S′ attached to the structure then
      B ← block attached at S′
      state(S) ← f(side of B with same orientation as S)
    else
      state(S) ← open
else if already part of structure then
  if robot asks to attach a block to side S then
    if state(S) = closed then
      answer no
    else if state(S) = corner then
      answer yes
    else {state(S) is open}
      answer yes
      send message 'close(S)' to blocks on both sides adjacent to S
      {that message will propagate along the row}
  if receive message 'close(S)' then
    state(S) ← closed
    send message 'close(S)' to block on side opposite that from which message
    came
  if a new block was just attached to side S then
    state(S) ← done
    send message 'attached(S)' to blocks on both sides adjacent to S
  if receive message 'attached(S)' then
    state(S) ← corner
    {mark newly formed right corners to allow attachment}
```

B: Robots

```
while structure not complete do
  get block from cache
  go to structure perimeter
  while still holding block do
    ask any adjacent structure blocks if block can be attached here
    if all structure blocks answer yes then
      attach block here
    else
      follow perimeter counterclockwise
```

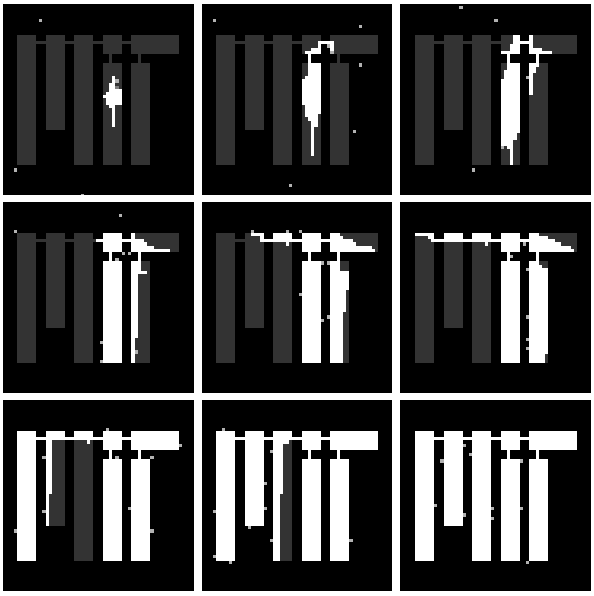

Figure 7: Construction of a sample compound structure, showing successive snapshots during the process of construction (left to right, top to bottom). Gray: robots; white: blocks; dark gray: regions where blocks will eventually be placed; black: regions where blocks will never be placed.

separated by space meant to be left empty in the final structure.

With this change, the user can easily specify a structure of any shape meeting the above criteria. The shape can be represented as a binary occupancy matrix, or more compactly as a collection of rectangles whose superposition gives the desired structure [9]. This information can be stored with the marker prior to construction, and thereafter disseminated along with a shared coordinate system to successive blocks as they are added to the structure. Fig. 7 shows an example of building a structure in the shape of the MIT logo (with letters attached, for a single contiguous structure). In individual runs, the details of which parts of the structure are assembled at which times may vary, but the final shape will reliably be that specified in advance.

6 Performance comparison

We now compare construction with inert vs. communicative blocks using these two algorithms, in terms of the time required to build structures. We present theoretical and experimental results on time behavior in terms of both (A) the number of blocks in the structure and (B) the number of robots. Simulations take place on a grid; a cell can be empty or contain a block, a robot, or a robot carrying a block.

Table 1: Summarized results from §6.1. (1) $N = 1$; (2) N arbitrarily large.

	Inert blocks	Communicative blocks
$t(n)$ (1)	$O(n(L + D + A))$	$O(n(L + D + A))$
$t(n)$ (2)	$O(nA)$	$O(\sqrt{n}A)$
$D(n)$ (1)	$O(n)$	$O(n^{0.6})$

6.1 Effect of structure size

We first consider how the time required to build a structure scales with its size. In this section we treat the two extremes of a single robot and an unlimited number of robots.

In general, the time required to build a structure of n blocks will be a function of several factors: N, the number of robots working on assembly; L, the time it takes to retrieve a block and bring it to the structure perimeter; D, the average distance a robot must travel along the perimeter before it finds a site where it can attach a block; A, the time required to attach a block to the structure. We will measure time in units such that it takes one time step to travel the length of one cell of the structure grid.

By definition, a single robot will require time $L + D + A$ on average to fetch and attach a block. L depends on factors like the locations of caches in the workspace, and A on the mechanical design of the hardware implementation; D, by contrast, mainly depends on the algorithm. Worst-case analyses of the scaling behavior of $D(n)$ can be performed for both inert and communicative blocks; of more interest, however, is the typical case, which is more easily explored by simulation.

We measure $D(n)$ experimentally, by repeatedly starting one robot at random on the circumference of a large circle centered on the marker, letting it move toward the marker until it reaches the structure perimeter, and recording how many cells it needs to move before it finds a valid site to attach a block. For inert blocks, D turns out to grow roughly linearly with n (Fig. 8), reflecting the tendency for structures built in this way to be highly elongated; with a square structure, a robot would typically have to travel on the order of $\sqrt{n}$ to find a valid site. For communicative blocks, D grows approximately as $n^{0.6}$. Thus we find that with one robot, the performance for the time required to build a structure of n inert blocks will be $O(n^2 + n(L + A))$; with communicative blocks, $O(n^{1.6} + n(L + A))$.

In the limit where N is very large, if we ignore interference between robots, some robot will always be present and available to attach a block at a site as soon as it becomes permissible to do so. Fig. 9 shows that with inert blocks, there will be four allowed sites at each step, regardless of the size of the structure, as a result of the requirement that robots turn a left corner before attaching a block at a left corner. The time required to attach n blocks is then $O(nA)$. With communicative blocks, the number of allowed sites increases linearly with the step number (along with the perimeter); at step s, then, $O(s)$ blocks will be attached, for a cumulative total of $O(s^2)$, and the time to build a structure of n inert blocks will accordingly scale like $O(\sqrt{n}A)$.

Table 1 summarizes these results.

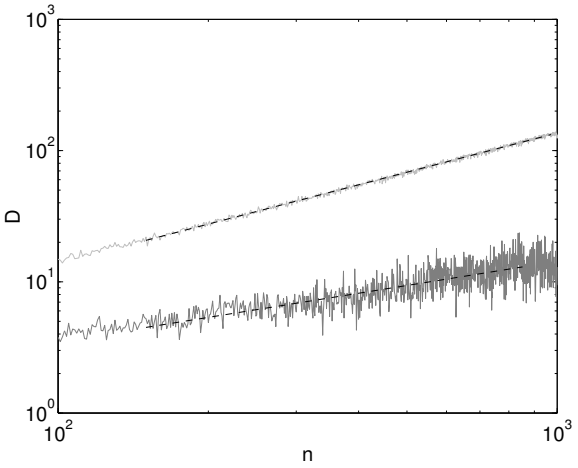

Figure 8: Mean distance D a robot reaching the structure must travel to find a site where it can attach its block, as a function of the number of blocks n already part of the structure, averaged over 100 runs. Light gray, inert blocks; dark gray, communicative blocks; dashed lines give the least-squares best fit on this log-log scale, obtaining $D(n) \sim O(n^{0.994 \pm 0.002})$ for inert blocks, $D(n) \sim O(n^{0.61 \pm 0.02})$ for communicative ones.

6.2 Effect of number of robots

We next investigate how the time required to build a fixed-size structure varies with the number of robots, exploring the space between the $N = 1$ and $N \to \infty$ limits. We do this experimentally with simulations of N robots building a square 21 blocks on a side.[1] In these simulations, $L = 0$ and $A = 1$; robots approach the structure from a circle as in the $D(n)$ experiments of §6.1; if blocked by another robot in front of it, a robot waits for the first to move on before continuing.

Fig. 10 shows how the time for structure completion scales with the number of robots N, with both inert and communicative blocks, averaged over 100 independent runs.[2] Construction with communicative blocks finishes much more quickly than construction with inert blocks, by more than a factor of 2 on average, consistently for all numbers of robots tested. In both cases, up to 10 robots can work on a structure of this size at once without significant interference effects, in the sense that N robots finish the structure in $1/N$th the time it takes a single robot.

By 20 robots, interference effects begin to appear. With inert blocks, the absolute time required to complete the structure increases as N increases further. Robots need to survey the structure firsthand to determine if a left corner is a valid site, rather

[1] Robots using Alg. 1 for inert blocks would continue building the structure larger; here we prevent blocks from being attached outside that square, for purposes of comparison.

[2] Omitted from the averages are a total of 5 runs with inert blocks and 84 or 100 robots, in which gridlock was so extreme that no block was attached within 12000 time steps.

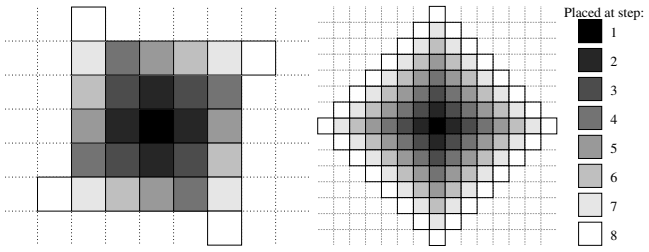

Figure 9: If the number of robots N is arbitrarily large, then at the end of each step of length A, we can assume that blocks will be attached at every site where it is legal to do so. In these diagrams, blocks attached at successive steps are shown successively lighter. Left, inert blocks; right, communicative blocks.

than being able to rely solely on local information; thus gridlock can occur with $N \gtrsim$ the length of the structure perimeter, with robots able neither to move nor to attach blocks at their current location. With communicative blocks, the absolute time required continues to decrease as robots are added, though with diminishing returns; when the structure can tell robots at once whether or not a site is valid for attachment, the total number of valid sites at any given time may be a limiting factor, but gridlock will not occur under the conditions of this simulation.

Fig. 11 shows the number of sites at which block placement would in theory be permissible, over the course of construction, for both types of blocks and for both 5 and 50 robots. Communicative blocks allow many more sites to be open for construction at once than do inert blocks. With communicative blocks, increasing the number of robots increases the peak number of valid sites; the system is able to take advantage of the additional robots so that construction goes on at once in more places around the perimeter (as with the large-N scaling in §6.1). By contrast, with inert blocks, only a few sites are ever available at a time for block attachment; and increasing the number of robots only leads to traffic jams, slowing the construction process, without opening up more simultaneous places along the perimeter where blocks can be attached.

We do not separately model robot interference elsewhere in the workspace, for two reasons: first, our goal is to explore the use of the algorithms, and what they mean for robot behavior around the structure; second, robots are likely to be most densely packed around the structure perimeter, so that if interference occurs anywhere, it will occur around the structure.

7 Additional construction procedures

We briefly consider how two related procedures, disassembly and error correction, could be accomplished with such a system. Both are easier to accomplish with communicative blocks than with inert ones, in terms of robot capabilities required.

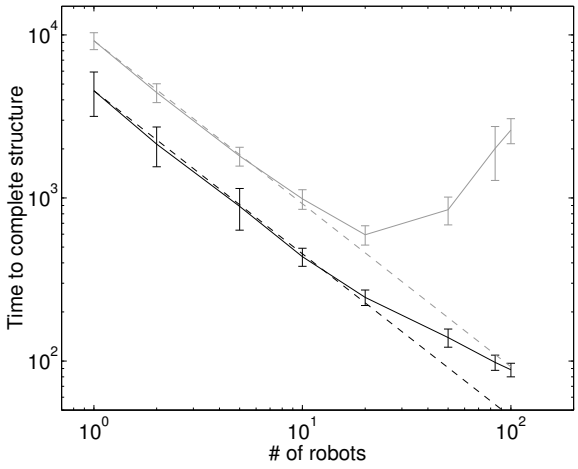

Figure 10: Mean time to completion for a 21×21-block structure, as a function of number of robots. Gray lines correspond to inert blocks, black lines to communicative ones. The dashed lines show how quickly completion would occur if there were no interference and N robots took $1/N$th the time taken by one robot.

7.1 Disassembly

Many applications will involve structures intended to be temporary; the modular nature of this system is especially suited to such cases, allowing a structure to be dismantled following use and its parts reused for another structure. Disassembly can be accomplished by having robots follow the structure perimeter, removing any blocks they find that satisfy the following criteria:

1. It has at most two neighbors, bordering adjacent sides; i.e., if that site were unoccupied, it could physically accomodate a block, as in Fig. 2.

2. If it does have two neighbors along adjacent sides, then the other site which those two neighbors border is occupied by a block (see Fig. 12).

3. It is not attached at the marker location, unless no other blocks remain; the block at the marker must be the last one removed.

The second and third criteria ensure that throughout disassembly, the structure remains one contiguous piece, which a robot homing in on the marker signal will be able to find. Violating either of those rules can result in part or all of the remaining structure becoming isolated, and thereby lost to robots no longer able to reach it by the simple procedure of following the marker signal to the structure and following the perimeter thereafter. If the structure is physically tethered to some anchor point via the first block attached (as would likely be the case, e.g., in space-based applications), then violating those criteria would mean that blocks could drift off and be lost in a more literal sense.

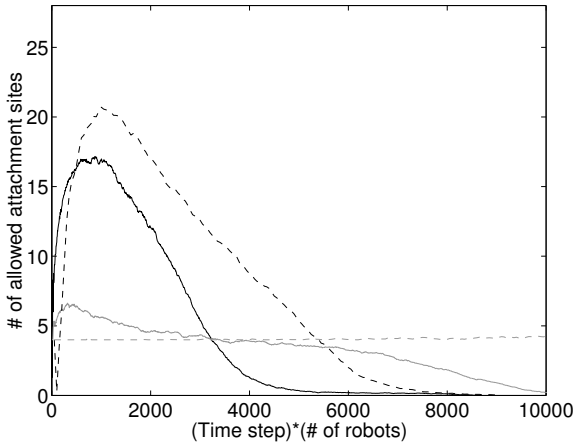

Figure 11: Mean number of available sites for attachment of communicative (black) or inert (gray) blocks, with 5 (solid) or 50 (dashed) robots. The time axis is scaled by the number of robots, so that all curves appear on a single plot. The curves peak and decline as the boundaries of the 21×21 building area are reached and the growth of the structure is restricted.

Care must also be taken with multiple robots that two blocks not be chosen for removal at the same time, such that each block considered alone satisfies the three criteria, but would violate criterion 2 if the other block were removed. Blocks 1 and 3 in Fig. 12 illustrate such a pair.

With inert blocks, robots thus need to be capable of determining the occupied/unoccupied status of all eight cells surrounding the block under consideration for removal. With communicative blocks, that information can be provided by the blocks themselves.

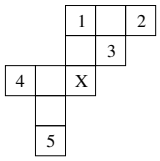

Figure 12: A structure in the process of disassembly. Numbered blocks can be removed. The block marked X meets criterion 1 in the text but not 2; if removed, it would split the structure into two parts. The marker may be at the location of any of the unmarked blocks.

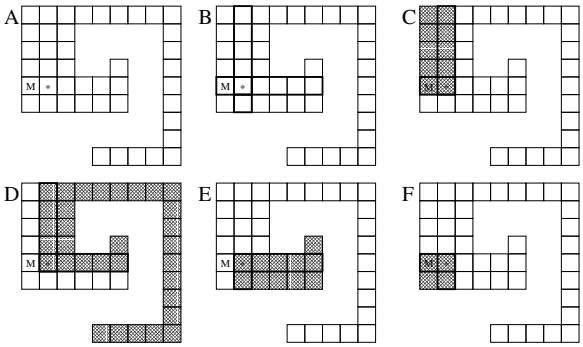

Figure 13: Error correction in a compound structure. The marker is at M, the block to be replaced at *. See text for discussion.

7.2 Error correction

Suppose that a block mistakenly ends up attached at a location where it should not be, or needs to be replaced for whatever reason. It may be some time before this error arises or is detected; we will thus consider the problem of replacing any arbitrary block in the structure. We outline the high-level approach and omit elaboration of an algorithm to be carried out by individual blocks.

Because of our assumption that a block can only be removed from the structure if it is already free on two sides, it is necessary to remove all blocks in an entire 'quadrant' of the structure in order to free up the target block. Fig. 13 demonstrates the demarcation of quadrants. (A) shows a completed compound structure; the marker is at M, and the block to be replaced at *. Extending horizontal and vertical arms from the target block, as in (B), defines the four (overlapping) quadrants, crosshatched in (C)–(F). A quadrant is constructed essentially by flood-filling the area including and to one side of a pair of arms.

Note that in a compound structure, a quadrant may contain blocks whose absolute coordinates go beyond those of the associated arms (as in (D), where a quadrant nominally above the right-hand arm includes blocks that extend below it). Also, any blocks that would otherwise be left isolated from the rest of the structure if a quadrant were removed must be included in that quadrant. This issue arises if an arm lies along an edge of the structure for all or part of its length. In this example, the right-hand arm lies along such an edge, at the top side of the second and third blocks from the target block; any blocks above and further along that arm must therefore be included in the quadrant nominally below it, as shown in (E).

For efficiency, the quadrant containing the fewest blocks should be chosen for removal. As with disassembly, the block attached at the marker must remain intact, and so any quadrants containing that marker are omitted from consideration in this choice. Here the quadrant in (F) contains the fewest blocks, but cannot be removed because it includes the marker (as does the quadrant in (C)). The quadrant chosen for removal will be that in (E).

With communicative blocks, those in the quadrant chosen forbid robots from attaching any further blocks to them, and ask to be removed. Upon receiving such a request, some robots may switch their role temporarily from 'assembly' to 'disassembly'. Once the target block has been removed, that section of the structure can be restored by construction as usual. Construction may also progress as usual in other parts of the structure the whole time this correction is taking place. With inert blocks, there is no straightforward way to implement error correction as described here.

8 Conclusions

In this paper we have outlined an early system for automated construction by a swarm of agents, and described simple algorithms by which robots could assemble gap-free structures out of blocks. We have shown that an active form of stigmergy, in which information stored in the environment can also be passed between objects in the environment, has advantages over a passive form (in which objects are purely inert) in several respects. Robots need not be able to recognize geometric features of the structure. Construction proceeds much more rapidly in absolute terms, and the time required to build a given structure scales more favorably with the size of the structure. The parallelism of the swarm can better be exploited, in that construction can proceed in many more locations around the structure at a time. Crippling interference with many robots is drastically reduced. The class of buildable structures easily and naturally extends to those of arbitrary perimeter, without requiring more complicated robot abilities or behavior.

Since particular robots are never assigned particular critical tasks, nor do the algorithms depend on actions being executed in any particular order, the system is highly robust to the temporary or permanent loss of robots, so long as some robots remain. Individual blocks, similarly, are not crucial for task completion. With communicative blocks, it is undesirable for them to fail after their attachment to the structure; but an error correction procedure can recover from that eventuality. The marker and cache beacons are failure points in the current formulation, in that if they fail, the system may be unable to complete its assigned task. One possible safeguard against their loss would be to build into robots or blocks the potential to replace them [14].

Various systems for automated construction are presently in early stages of design, specifying inert (but specialized) building materials [3, 10]. Our results suggest that incorporating a capability for communication into those materials could be of great utility.

Future work will consider building with heterogenous blocks, extending the algorithms to handle structures with internal gaps, and constructing in three dimensions.

Acknowledgments

We would like to thank Daniela Rus and Jason Redi for useful discussions.

References

[1] Bonabeau, E., Dorigo, M. & Théraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press Inc.

[2] Bowyer, A. (2000). *Automated Construction using Co-operating Biomimetic Robots* {Technical Report, University of Bath Department of Mechanical Engineering}. Bath, UK.

[3] Everist, J., et al. (2004). A system for in-space assembly. In *Proc. IROS 2004*, Sendai, Japan: 2356–2361.

[4] Gerkey, B. & Matarić, M. (2002). Pusher-watcher: an approach to fault-tolerant tightly-coupled robot coordination. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Washington, D.C., USA: 464–469.

[5] Hill, J., et al. (2000). System architecture directions for networked sensors. In *ASPLOS-IX*, Cambridge, MA.

[6] Jones, C. & Matarić, M. (2004). Automatic synthesis of communication-based coordinated multi-robot systems. In *Proc. IROS 2004*, Sendai, Japan: 381–387.

[7] Melhuish, C., Welsby, J. & Edwards, C. (1999). Using templates for defensive wall building with autonomous mobile ant-like robots. In *Proc. Towards Intelligent Autonomous Mobile Robots 99*, Manchester, UK.

[8] Parker, C., Zhang, H. & Kube, R. (2003). Blind bulldozing: multiple robot nest construction. In *Proc. IROS 2003*, Las Vegas, USA.

[9] Støy, K. & Nagpal, R. (2004). Self-reconfiguration using directed growth. In *Proc. DARS 2004*, Toulouse, France.

[10] Terada, Y. & Murata, S. (2004). Automatic assembly system for a large-scale modular structure: hardware design of module and assembler robot. In *Proc. IROS 2004*, Sendai, Japan: 2349–2355.

[11] Theraulaz, G. & Bonabeau, E. (1995). Coordination in distributed building. *Science* **269**: 686–688.

[12] Vona, M. & Rus, D. (2001). Crystalline robots: self-reconfiguration with compressible unit modules. *Autonomous Robots* **10**(1): 107–124.

[13] Wawerla, J., Sukhatme, G. & Matarić, M. (2002). Collective construction with multiple robots. In *Proc. 2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Lausanne, Switzerland.

[14] Werfel, J. (2004). Building blocks for multi-agent construction. In *Proc. DARS 2004*, Toulouse, France.

[15] Yim, M., Zhang, Y., & Duff, D. (2002). Modular robots. *IEEE Spectrum* **39**(2): 30–34.