Distributed Multi-Robot Algorithms for the TERMES 3D Collective Construction System

Justin Werfel, Kirstin Petersen, and Radhika Nagpal

***Abstract*— The research goal of collective construction is to develop systems in which large numbers of autonomous robots build large-scale structures according to desired specifications. We present algorithms for TERMES, a multi-robot construction system inspired by the building activities of termites. The system takes as input a high-level representation of a desired structure, and provides rules for an arbitrary number of simple climbing robots to build that structure, using passive solid building blocks under conditions of gravity. These rules are decentralized, rely on local information and implicit coordination, and provably guarantee correct completion of the target structure. Robots build staircases of blocks (potentially removable as temporary scaffolds) that they can climb to build structures much larger than themselves.**

I. Introduction

In nature, there are many examples where relatively simple and limited individuals coordinate to self-assemble large-scale structures. A classic example is termite mound construction [1], [8]. Millimeter-scale insects build meter-scale mounds, with complicated architecture including features such as specialized nest chambers, fungus gardens, and self-regulating ventilation systems. Termite colonies achieve tremendous complexity, parallelism and robustness, with individuals that are simple, decentralized, and expendable. Engineering can draw inspiration from these natural systems with the research area of collective construction, whose goal is to develop robot swarm construction systems in which large numbers of autonomous robots build large-scale structures according to desired specifications. Such artificial construction systems have potential for application in many settings difficult or dangerous for humans, e.g., construction of levees, structural support elements, or temporary shelters in disaster areas; or construction of underwater or extraterrestrial habitats. Eventually such systems could increase automation in the construction industry and reduce accidents, as well as enable automated long-term repair and maintenance in dynamic environments.

A key challenge to the realization of collective construction systems is algorithmic: how do robots coordinate to construct a large-scale structure correctly, while retaining a high level of parallelism and simplicity at the single robot level? Another challenge is physical: how do we design robots and modular building materials such that robots can construct structures much larger than themselves in the presence of gravity? These challenges are not entirely separable, e.g., physical constraints have to be taken into account by the decentralized algorithms.

This work was supported by the Wyss Institute for Biologically Inspired Engineering.

J. Werfel is with the Wyss Institute for Biologically Inspired Engineering, Harvard University, Cambridge, MA 02138, USA justin.werfel@wyss.harvard.edu

K. Petersen is with the School of Engineering and Applied Sciences and Wyss Institute, Harvard University, Cambridge, MA 02138, USA kirstin@eecs.harvard.edu

R. Nagpal is with the School of Engineering and Applied Sciences and Wyss Institute, Harvard University, Cambridge, MA 02138, USA rad@eecs.harvard.edu

In this paper we describe an algorithmic approach to 3D collective construction, as part of a project called TERMES, which is inspired by the building activities of termites and other social insects. In this system, autonomous robots build structures using passive modular building blocks, climbing over structures that they themselves build. Robots operate under conditions of gravity, constructing staircases of blocks as scaffolds to allow them to reach heights and build structures larger than themselves. Elsewhere we introduced a hardware system that implements a climbing robot and blocks it can manipulate to build large structures [5]. Here we focus on the high-level algorithmic approach, by which an arbitrary number of such robots can build 3D structures, using decentralized control and implicit coordination. We show that this system can autonomously build arbitrary user-specified structures from a large class of possibilities. We prove the correctness of the algorithm, and show how robots can use simple rules to avoid the construction of intermediate deadlock structures or structures that can no longer be traversed by the robots.

II. Related Work

Algorithmically, the problem of collective construction is closely related to lattice-based self-reconfiguring modular robotic systems and programmed self-assembly [2], [6], [12]. Typically in such systems all modules are intelligent, communicating, and mobile. Collective construction can be thought of as an example of a bipartite self-assembling modular robot system, where there are two types of “elements”: robots (self-mobile) and building blocks (passive, non-mobile, designed for attachment). This bipartite separation lets blocks be optimized for structural properties and low expense, and robots be specialized for mobility and reused for other building projects.

Unfortunately, this separation also increases the complexity of algorithm design beyond that for traditional modular robots. Robot movement constraints can be quite complex, especially when carrying blocks, and the use of passive blocks without embedded electronics implies that information needed for coordination in the self-assembly process is more difficult to propagate through the system. Further, if

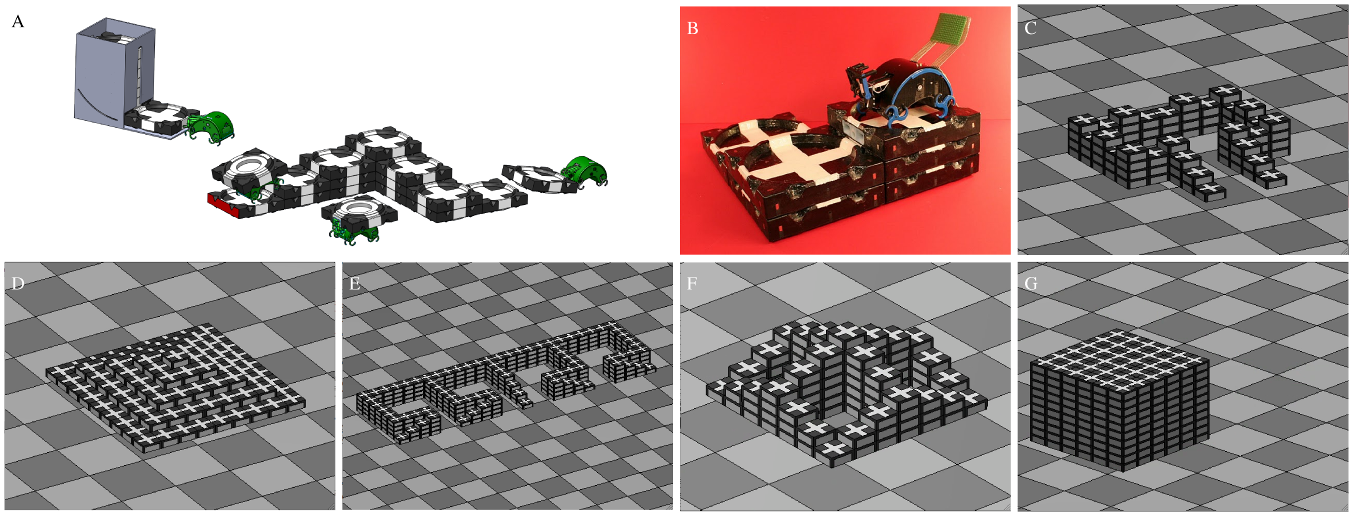

Fig. 1. (A) Overview of proposed system. Robots collect blocks from a cache (at left) and use them to build a desired structure starting from a marker block (with red face). (B) Hardware implementation. (C–G) Examples of structures buildable by the system, demonstrating single-path additive structures (C,D), splitting (E,F) and merging (F) paths, and a structure requiring a temporary staircase as scaffold (G).

the ultimate goal is to develop human-relevant construction, then the process must take into account gravity, which places constraints on final structures as well as intermediate structures during the building process.

Several groups in the modular robotics community have worked on decentralized and local algorithmic approaches to 3D collective construction, e.g., [7], [11], [13]. In each of these systems, there have been algorithms and compilers designed for creating complex 3D structures. These algorithms generally involve extensive communication, often among active structural elements. Difficulties with the requirements of physical operation in terrestrial gravity, and other mechanical issues, have severely limited the ability to translate such systems into practice.

One notable work deals with gravity by using flying robots, quad-rotor helicopters capable of building 3D truss structures, including physical implementation [4]. This system relies on precise global localization of its robots at all times (using Vicon), and on centralized control. Our interest is in decentralized algorithms for independently controlled robots with only local information and on-board sensing.

Some algorithmic work in the area of self-assembling modular robots [3] accounts for gravity by adding a constraint that (self-mobile) blocks cannot ”climb” more than 1 block height; this constraint is handled with the use of staircases to reach higher levels. This is an effective approach to dealing with gravity, and our TERMES work builds on this basic concept. However, the algorithmic approach in [3] depends heavily on inter-block communication for coordination, which can not be easily translated to collective construction with passive blocks. Instead we take advantage of implicit stigmergic principles, following the approach of our previous work on 2D structures [10] that showed how coordination can be achieved with passive blocks and robots that use memory and short-range pattern recognition. We show that even with such a simple coordination mechanism and hardware requirement for robots and blocks, it is possible for a system like TERMES to construct complex structures and coordinate in a robust and provable manner.

III. Model

The design of the TERMES system has been driven by considerations of robot implementation, cheaply manufacturable blocks, and the need to operate in gravity. The system (Fig. 1) comprises two main types of components: mobile robots to perform construction, and specialized blocks for them to use as building material. A unique marker block (identical to the other blocks in shape but recognizably distinct from them) is placed in the environment as a landmark to indicate where construction should begin. Free blocks are located in a cache, far enough from the structure so as not to interfere with its progress. Beacons let robots navigate between the structure and cache.

Robots can move freely in the plane, unloaded or while carrying one block; they can follow the perimeter of the structure in progress, and recognize the marker block when found. A robot can attach a block at an adjacent site at the same level as itself (e.g., a robot atop a two-block stack can add a block atop an adjacent two-block stack); it can climb onto a block one step higher, descend to one one step lower, or move from one to another at the same level. While traveling atop a structure, robots can keep track of their movement and location relative to the structure, using local sensors (infrared and tilt) to detect features on blocks and recognize height changes. We assume no global information such as GPS, and for coordination of robot actions rely primarily on implicit communication, through joint manipulation of a common environment.

Blocks are square boxes in shape, with footprint larger than that of the robots. Markings provide visual feedback to help robots align with block axes and keep track of movement from one block to the next. Self-aligning connectors (complementary physical shapes and magnets) achieve precise alignment and secure attachment. Besides these passive

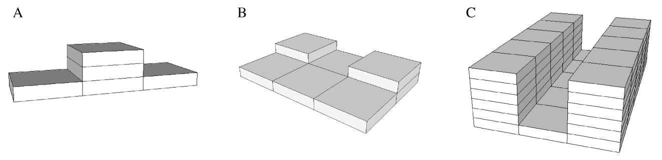

Fig. 2. Problematic situations for robots that we want to prevent from occurring. (A) A robot cannot climb onto or down from the central "cliff". (B) Attaching a block at the rear center site would be difficult because of the blocks flanking it. (C) A robot may have trouble carrying a block down a narrow tunnel like this one.

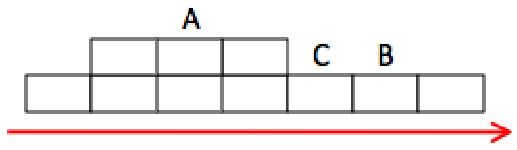

Fig. 3. Dangers of attaching blocks indiscriminately (side view of linear structure). If a robot at A adds a block to its left, it creates an unclimbable cliff; if it adds a block to its right, it creates an undescendable cliff. If a robot adds a block at B, it becomes impossible to add another at C.

physical features, blocks play no active role in coordinating robot activities or transferring information.

The desired target structure is specified by the user as a high-level representation such as an occupancy grid, indicating which sites (relative to the marker block) are ultimately intended to be occupied by blocks.

IV. Single-Path Additive Structures

In this section we present methods (first proposed but only briefly discussed in [5]) by which an arbitrary number of robots can provably build structures from a given class.

A. Admissible Structures

Due to considerations of physical implementation, the TERMES hardware, both blocks and robots, restricts the class of structures the system can build. Block shape and stacking require the structure to be representable as a 3D occupancy grid (no curved surfaces, no overlaps like bricks, etc.). Blocks cannot hang unsupported: every block must be supported by the ground or a stack of other blocks.

Limitations on how robots can move and manipulate blocks affect not only the final structures but also the intermediate configurations that can be allowed to occur during the construction process. Robots cannot climb up or down more than the height of one block at a time (Fig. 2A). Simple mobile robots will find it difficult or impossible to fit one solid block directly between two others (Fig. 2B). Similarly, it may be very difficult for robots to carry blocks down narrow corridors (Fig. 2C), particularly if turning corners is required.

Inappropriate intermediate configurations can lead to deadlock. Preventing deadlock in 3D collective construction requires considerable nonlocal information [11], and information gathered by robots can easily become stale as other robots modify the structure. Some coordination mechanism is therefore required.

Algorithm 1 Robot routine for single-path additive structure.

```
loop
    get new block from cache
    go to structure
    follow perimeter clockwise until entry point found
    climb onto structure
    while on structure do
        follow structpath
        if holding block
        and plan specifies block here
        and next site along path is at same level
        and (just descended from higher level or previous
        site is at same level and supposed to be empty) then
            move to next site along structpath
            attach block at site just vacated
```

One approach that can provide effective coordination is to specify a single-file path that all robots can and must follow while adding blocks to the structure. Doing so brings several benefits—giving a clear priority scheme that makes it easy to resolve conflicts, making it easily predictable where blocks are already present, reducing interference between robots—without greatly limiting the number of robots that can be present on the structure at once, and so preserving considerable opportunity for parallelism.

In this section we limit admissible structures to those for which such a path can be found. Let Π be the 2D projection of the desired structure in the ground plane, with each site marked with the height of the stack of blocks at that site (as in Fig. 5A). The class of admissible structures, then, is those for which a nonbranching path P can be drawn such that: (1) its entry and exit point lie along the structure perimeter; (2) it visits each site in Π exactly once; (3) the stack height changes by not more than 1 from one site to the next along P; (4) no site on P is flanked on opposite sides by two previously visited sites; (5) no site on P is flanked on opposite sides by two sites whose desired stack heights are more than 2 blocks higher than its own. Conditions 4 and 5 avoid the conflicts described in Fig. 2.

A simple offline compiler takes an input structure specified as a 3D occupancy grid, and finds a path P satisfying these constraints or determines that no such path exists. It is sufficient to consider in turn each height-1 site along the structure perimeter as a potential starting point, and perform a depth-first search for a P that satisfies the requirements above. The compiler returns a "structpath" representation, consisting of P plus the desired stack height at each site, which is used by the algorithm presented next.

B. Algorithm

Attaching blocks without restriction at any sites eventually meant to be occupied will quickly lead to problems (Fig. 3). A partial ordering on block attachment must be imposed in order to prevent deadlock.

Theorem: Alg. 1 [5] guarantees completion of a target structure, working from a structpath as described above.

Proof: Start by assuming an unfolded path (no turns) for clarity. For Alg. 1 to guarantee producing the target structure, several things must be shown: (1) robots will not create unclimbable or undescendable cliffs (Fig. 3A); (2) robots will not create unfillable gaps (Fig. 3B); (3) deadlocks cannot occur, where physically reachable sites remain where blocks should be attached but the rules forbid attachment; (4) two robots cannot attach blocks at mutually conflicting sites.

(1) Unclimbable/undescendable towers are not possible because that would involve attaching a block at a location where the next or previous site along the path is at a lower level, forbidden by Alg. 1.

(2) Unfillable gaps will not occur because that would involve attaching a block at a site following an empty site meant to be occupied, not permitted by Alg. 1.

(3) Proof by contradiction. Assume that deadlock has occurred in a structure built legally according to Alg. 1. Then for each remaining site[1] where attachment is desired, one of the following must be true in order for Alg. 1 to forbid it: (a) a block is present at (X+1,Z)—impossible because Alg. 1 would not have permitted that block to be attached, since (X,Z) is supposed to be occupied but is empty; (b) no block is present at (X+1,Z-1); (c) no block is present at (X-1,Z-1); (d) (X-1,Z-1) is occupied, and (X-1,Z) is empty and supposed to be occupied, in which case we apply the whole argument recursively to (X-1,Z) instead (must terminate for a finite structure). Cases (b) and (c) break down into subcases:

(3b) Either there is supposed to be no block at (X+1,Z-1), in which case the target structure would be illegal (involving an undescendable cliff); or there is supposed to be a block there and it is legal to attach it there, a contradiction (this is not deadlock after all); or there is supposed to be a block there and it is illegal to attach it, in which case we apply this whole argument recursively to that site instead (the argument must terminate since the structure must reach the ground eventually). Case (3c) resolves analogously.

(4) Multiple robots cannot cause problems: the conditions of Alg. 1 are local, so the actions of more distant robots (beyond the safe following distance they maintain along the single-file path, see below) can have no effect on whether those local checks are satisfied.

Folding up the linear path does not affect this argument. The only potential new difficulties it could introduce are the need to put a block directly between two others at right and left, or the need to carry blocks down narrow tunnels, which are prevented by conditions 4 and 5 on P above. □

This approach only involves adding blocks to the structure; once attached, they never need to be removed. Robots bring blocks to the structure, climb onto the marker block at the entry point of the path, and follow the path until climbing back off the structure, attaching the block they carry at the first available opportunity. Robots are not required to directly communicate with other robots at any time.

This routine leads to the growth of the structure as a rising and falling staircase along the specified path. Fig. 4 shows the order in which blocks will be added for a sample structure.

[1]Identify this site by its coordinates (X,Z), where the x-axis runs along the path direction and the z-axis runs vertically.

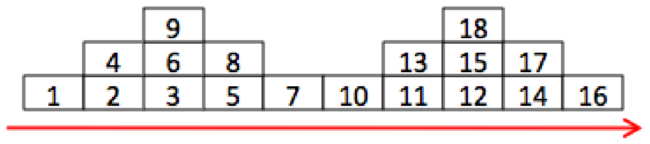

Fig. 4. Side view of a linear structure in which blocks will be added in the order shown as robots enter from the left and leave to the right.

C. Resolving Conflicts with Multiple Robots

Multi-robot systems need to be able to handle encounters when two robots meet, resolve conflicts when two want to take conflicting actions, and so on. In the construction scenario, another concern is indirect conflicts that result in deadlock when two robots attach blocks at mutually conflicting sites.

The use of an effectively 1D path makes it straightforward to resolve many of these issues, by allowing unambiguous priorities associated with robots. Robots follow a single-file line on and around the structure; in general, the farther along the sequence of [fetch block]-[return to structure]-[follow perimeter]-[follow structpath] a robot is, the higher its priority. In an encounter, a lower-priority robot waits for the higher-priority one to move on and make room. For instance, a robot approaching the perimeter, encountering one already following the perimeter, should wait until the latter has passed.

In this way, in principle any number of robots can follow Alg. 1 and expect to successfully complete the target structure, always yielding to robots ahead of them. The 1D path provides effective serialization of attachments, and the algorithm prevents indirect conflicts.

D. Time to Completion

We can characterize the time required to build a structure of N blocks. Depending on the shape of the structure, the corresponding path will be of length $\lceil 2\sqrt{N}-1 \rceil$ at minimum and N at maximum. Suppose it takes time L to fetch a new block and return to the structure with it, and one time step to move one space along the path or attach a block. Then one robot acting alone will take time between $O(N(\sqrt{N}+L))$ and $O(N(N+L))$ to build the full structure.

Multiple robots can take advantage of parallelism to decrease construction time. In the extreme case—N robots well-behaved enough to avoid all interference—a continuous line of robots with blocks will move along the path, constantly adding blocks to the current end of the structure. Such a case will require only $O(N+L)$ time to complete the structure.

In general, a case between the one-robot and N-noninterfering-robots extremes is expected, with some robots fetching blocks from the cache while others add them to the structure, thus achieving some speedup through this parallelism.

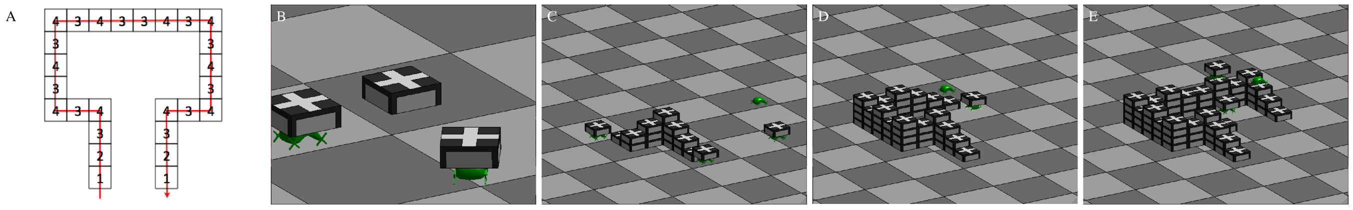
Fig. 5. (A) Compiled path for the structure shown in Fig. 1C. (B–E) Snapshots from a simulation of five robots autonomously building the structure.

V. Branching and Merging Paths

Not all structures can be described with single paths. By allowing a path to split and rejoin itself, we can build a larger class of structures (e.g., Figs. 1E,F and 6A,B).

When a robot reaches a site where the path splits, it picks a branch at random to continue down. We allow multiple exit points. When two branches merge, they must enter the merge site from orthogonal directions, not from opposite sides; otherwise the merge site would require adding a block directly between two others.

The conditions in the inner **if** statement in Alg. 1 must be changed slightly to handle branching paths. When attaching a block at a split point, the "next site along path..." condition must apply to all outgoing branches; similarly, when attaching at a merge point, the "just descended... **or** previous site..." check must apply to the previous site from all incoming branches. Additionally, two robots approaching a merge point from different branches will need to establish right-of-way (with explicit communication or implicit conventions, like drivers at intersections), since there is no longer an unambiguous line-leader once the path is not 1D.

The offline structpath compiler must be modified non-trivially to generate branching paths. In addition to the search becoming more complicated because of the greater space of possible paths, it is possible for construction along adjacent parallel branches to lead to a block needing to be attached directly between two others, if the order of robot actions happens to be unlucky (Fig. 6C). The compiler should recognize such possibilities and reject them if an alternative exists. Compiler design is left to future work; in this paper we have designed structpaths for branching structures by hand.

VI. Temporary Staircases

Another way to extend the class of buildable structures arises if robots can remove blocks as well as adding them. In this case, robots can build structures that have sections unreachable by any path in the final structure (e.g., tall towers with no way of climbing to the top), by building auxiliary staircases as scaffolds that give access to the necessary level and are then removed when the structure is completed [3]. Figure 7 gives an example.

Staircases are built in adjacent pairs forming a two-lane highway, so that robots can go up one lane and down the other (Fig. 7B). Robots first build the full structure with staircase. Once the structure is finished, they switch to following a new path that includes only the staircase and not the completed structure, and use a new algorithm (Alg. 2) that gradually removes the staircase. The staircase

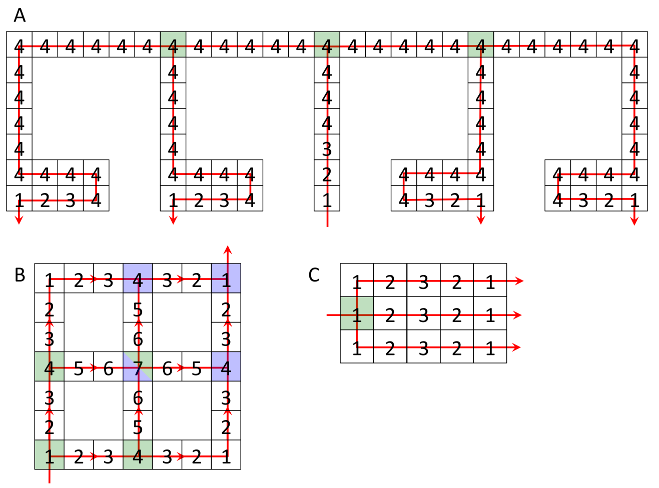
Fig. 6. Structpaths for branching structures. Split sites are marked in green, merge sites in blue. (A,B) Structpaths for the structures shown in Fig. 1E and 1F. (C) Potential problem with adjacent parallel branches: if the top and bottom rows happen to be built first, completing the middle row later will require attaching blocks directly between two others.

is removed layer by layer from the top down, with each layer being peeled off from the end of the path back toward the start. A robot switches from additive construction to staircase removal when it reaches the end of the completed structure while still holding a block, or when it reaches the point where the staircase-only path splits from the main structure path and finds an unclimbable cliff in the latter direction (indicating that other robots have already started disassembly).

The user input specifies which part of the target structure is permanent and which is the staircase to be removed. The compiler then outputs two structpaths: (1) a single unbroken path that runs up one staircase, visits every site in the target structure exactly once, and descends a second staircase running immediately adjacent to the first; and (2) a staircase-only path that goes up the entry staircase and immediately down the exit one (Fig. 7B). A future version of the compiler could take only the final target structure as input, and automatically find places to add staircases if otherwise unbuildable [3].

VII. Conclusion and Future Work

In this paper we have presented an algorithmic approach to collective construction with climbing robots building with solid blocks in the presence of gravity. We have shown that the system can autonomously and provably build user-specified structures from a large class of possibilities, using a physically-motivated model based on a hardware

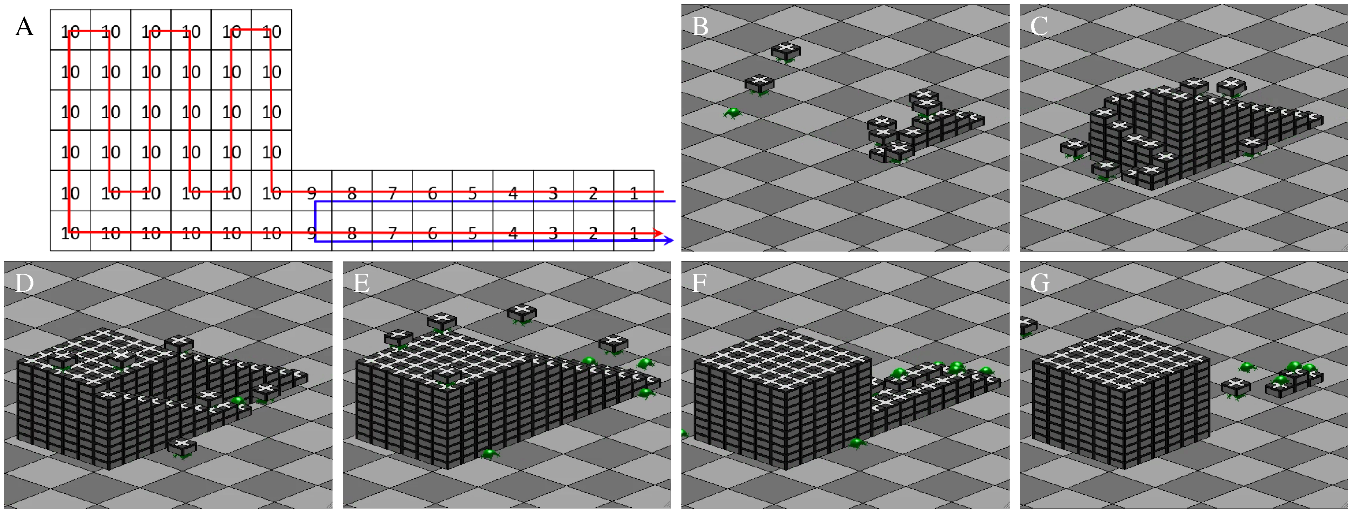

Fig. 7. (A) Compiled paths for the structure shown in Fig. 1C: full path for structure plus staircase in red, staircase-only path in blue. (B–G) Snapshots of ten robots building the structure and removing the staircase after the tower is complete.

Algorithm 2 Robot algorithm for removing a temporary staircase.

```
start by following Alg. 1 (construction)
if reach end of full path without having attached block
or reach site where the full path and staircase-only path
split, and encounter an unclimbable cliff in the direction
of the structure path then
  leave structure and discard current block
  while staircase not entirely removed do
    go to structure
    follow perimeter clockwise until entry point found
    climb onto structure
    while on staircase do
      go to next site along staircase-only path
      if just descended step and not carrying block then
        turn and pick up the block just descended from
    discard block
```

implementation we have presented elsewhere [5]. Robots are independently controlled, and coordinate their actions implicitly through manipulation of a shared environment.

One direction for future work is the further development of the offline compiler, extending it to search for branching paths and to automatically add temporary staircases to otherwise unbuildable structures. Extending the approach to multiple construction stages [3], each involving a temporary staircase for a different substructure, could increase the set of buildable structures still further. We are also investigating the use of stochastic rules to build structures not fully specified in advance, potentially adapting to preexisting environmental elements [9].

While the current system is capable of building a large class of structures, it is limited by the need for each block to be supported by a stack of others. The use of heterogenous block shapes [3] (e.g., short beams or larger plates, which could be realized for instance by creating "unfolding" blocks that robots can carry in a compressed state) could enable features like short roofs and overhangs, thus dramatically increasing the space of interesting structures the system can create.

References

[1] Pierre-Paul Grassé. La reconstruction du nid et les coordinations inter-individuelles chez Bellicositermes natalensis et Cubitermes sp. La théorie de la stigmergie: Essai d'interpretation du comportement des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1959.

[2] Roderich Groß and Marco Dorigo. Self-assembly at the macroscopic scale. *Proc. IEEE*, 96(9):1490–1508, 2008.

[3] Alexander Grushin and James A. Reggia. Automated design of distributed control rules for the self-assembly of prespecified artificial structures. *Robotics and Autonomous Systems*, 56(4):334–359, 2008.

[4] Quentin Lindsey, Daniel Mellinger, and Vijay Kumar. Construction of cubic structures with quadrotor teams. In *Proc. Robotics: Science & Systems VII*, 2011.

[5] Kirstin Petersen, Radhika Nagpal, and Justin Werfel. TERMES: An autonomous robotic system for three-dimensional collective construction. In *Proc. Robotics: Science & Systems VII*, 2011.

[6] Kasper Støy, David Brandt, and David J. Christensen. *Self-Reconfigurable Robots: An Introduction*. MIT Press, 2010.

[7] Yuzuru Terada and Satoshi Murata. Automatic modular assembly system and its distributed control. *International Journal of Robotics Research*, 27(3–4):445–462, 2008.

[8] J. Scott Turner. A superorganism's fuzzy boundaries. *Natural History*, 111(6):62–67, July-August 2002.

[9] Justin Werfel, Donald Ingber, and Radhika Nagpal. Collective construction of environmentally-adaptive structures. In *Proc. 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[10] Justin Werfel and Radhika Nagpal. Extended stigmergy in collective construction. *IEEE Intelligent Systems*, 21(2):20–28, 2006.

[11] Justin Werfel and Radhika Nagpal. Three-dimensional construction with mobile robots and modular blocks. *International Journal of Robotics Research*, 27(3–4):463–479, 2008.

[12] Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S. Chirikjian. Modular self-reconfigurable robot systems: Challenges and opportunities for the future. *IEEE Robotics and Automation Magazine*, 14(1):43–52, 2007.

[13] Seung-kook Yun, Mac Schwager, and Daniela Rus. Coordinating construction of truss structures using distributed equal-mass partitioning. In *Proc. 14th International Symposium on Robotics Research*, 2009.