Global-to-Local Rule Generation for Self-Assembly and Self-Repair by Robot Swarms

Daniel Arbuckle and Aristides A. G. Requicha

Department of Computer Science
University of Southern California
Los Angeles, California, USA
daniel.arbuckle@usc.edu, requicha@usc.edu

Abstract

We present a pair of techniques, one a reactive self-assembly technique intended for use in swarms of simple robots, the other a distributed method by which a swarm can determine a set of such self-assembly rules given that the swarm has been assembled by external forces into the goal shape. The self-assembly technique recovers from the partial destruction of the assembled structure, both during and after construction, and has a compact representation allowing for the construction of large structures. The rule generation technique is able to produce local rules for a broad class of global shapes without recourse to any central knowledge or processing, and does so without requiring any more writable memory than that needed to store the complete rule set.

1. Introduction

Many of the objects that people use are defined primarily by their form. We are interested in the possibility that swarms of general-purpose self-assembling robots might be used to replace many such objects in daily life. These robots would need to be as simple as possible, re-usable and easy for non-technical people to use. They would also need to be able to reliably construct the desired forms, and if possible repair those forms when they are damaged.

Guided by these goals, we have developed a software architecture for self-assembly robots. Robots running our architecture are able to self-assemble into complex non-symmetric shapes, tolerate the failure of one or many of the robots making up a structure, repair the structure even after most of the constituent robots are damaged or destroyed, construct instances of a form that have been scaled to either a larger or smaller size, and learn the local rules needed to construct a global shape given an example of that shape built by external forces. All of these capabilities can be implemented in an extremely minimal robot with little processing power or memory. The work described here is related to the techniques described in [1], [2], [3] and their references. We assume that all robots are identical, square-shaped, and have exactly the same program (set of rules). Initially, they move randomly in the plane. They exchange messages only with other robots that come into contact with them, and may attach or detach themselves from adjacent robots.

2. Self-Assembly and Self-Repair

Self-assembly rules in our system take the form of reactive rules defining messages that a robot should transmit in response to the receipt of other messages. These rules implicitly define the shape of the desired structure, on the principle that a message cannot be successfully transmitted to a neighbor that does not exist. Following up on this idea, robot programs can be arranged so that any attempt to transmit a message to a particular neighbor implies that there should be a neighbor there to receive it. This allows the use of a simple actuation scheme that attempts to acquire or retain neighbors for each message being sent. Figure 1 illustrates this concept with an attached pair of robots (squares with a pair of "V" shapes on each edge). Arrows represent attempted transmission of messages. Since this pair is attempting to transmit messages to two positions in which there is no robot to receive them, the actuation will attempt to bind robots to the structure in those positions.

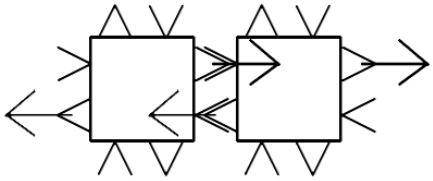

Fig. 1. A pair of robots attempting to bind other robots on two edges

This differs from a simple crystallization-analogue mechanism in that the need for neighbors can be time-varying as the flow of messages through the structure progresses, allowing the construction of temporary support structures, as well as ordered multistage construction. Another important feature of the technique is that the use of messages allows most of the robots to operate based on a few generic rules, confining the use of shape-specific rules to a tiny fraction of the total number of robots in a structure. This provides a dramatic decrease in the amount of memory required to represent a shape — in particular, the robots that are in the interior of the shape, and those that are part of a line segment between vertices of the shape, all execute only generic rules. Finally, having the robots respond reactively to messages instead of storing state facilitates self-repair, since it allows the robots to always act correctly, even when part of the structure has been destroyed or rendered inoperative. This fault tolerance is unfortunately not available during global-to-local learning of a shape, due to the storage of state in individual robots during that process.

3. Global-to-Local Rule Generation

The global-to-local process operates in several steps. It is assumed that a group of robots have previously been formed into an example of the desired solid two-dimensional polygonal shape, and that one or more robots on the boundary of the shape receive an external signal to initiate the learning process.

The first step of the global-to-local process is a simple message that propagates to all connected robots in the structure. On reception of this message, a robot records that it has a neighbor connected to it in the direction from which it received the message. Then, if it is aware of only that single neighbor, it retransmits the original message toward its other potential neighbors and sends an acknowledgment back to the known neighbor. The known neighbor, on receiving the acknowledgment, records that it has a neighbor connected in the direction from which it received the acknowledgment.

This first pair of messages has the effect of informing each connected robot of the topology of its local neighborhood. If the robots have the capacity of directly sensing connectedness, those messages are not necessary. Connectedness information is used by the robots to deduce whether they are on an edge of the structure.

After local topologies and edge status have been discovered, a probe message is sent. Probe messages propagate from edge robot to edge robot though a parameterized number of hops. Probe messages have fields for position of the currently receiving robot relative to the robot that originally sent the message, measured in terms of how many robots the message has passed through in each axis. Each robot updates these hop count and relative position fields as it sends messages on to its neighbors. When a probe message is received with a hop count equal to 0, the receiving robot sends an echo message in the other direction. Probe messages are propagated around corners, as indeed they must be since their purpose is to identify which corners should be considered vertices of the shape. Figure 2 illustrates the behavior triggered by a probe message. Each message is represented in the figure with its name, parameters, direction of transmission, and the time at which it was transmitted. At time 0, the leftmost robot is transmitting a probe message with a hop count of 3, and a relative x, y position of the receiver of (1, 0). Similar messages are sent at times 1, 2 and 3, each with a decremented hop count and an incremented receiver position. When the probe message with a hop count of 0 is received, it causes the echo message to be transmitted at time 4.

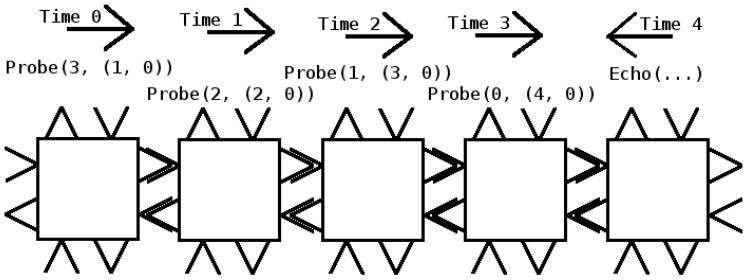

Fig. 2. A probe message being sent to the right, and after 4 hops triggering an echo message in the other direction.

An echo message contains relative position fields similar to those stored in a probe message, but it also contains the relative position of the robot that received a probe message and emitted an echo message, hereafter called the distant robot. Each robot, on receiving an echo message, checks the relative positions stored in fields of the message to see if they are consistent with a line passing through the robot with

relative position (0,0), which is the robot that initiated the probe. If the Manhattan distance from the position of the potential vertex robot to the line between the initiator and the distant robot is greater than two, the robot determines that it is at a vertex of the structure, in which case it initiates negotiation with the robot that sent the original probe message to determine the parameters of the line between them, and sends a new probe message back in the direction from which it received the echo message.

If no robot finds itself off the line between origin and endpoint during the propagation of an echo message, the message continues to propagate until it reaches the robot that originated the probe message or until it reaches a hop count threshold (not represented in the example), whichever comes first. In either case a new probe message is sent, with a new and higher hop count threshold. In this way the probe messages reach further and further from a known vertex until they have reached far enough to prove that another robot is a vertex. Figure 3 illustrates the behavior triggered by an echo message. At time 0, the distant robot has received a probe message with hop count 0 and is emitting an echo message. At time 1 the echo message propagates. At time 2, the vertex robot has the information it needs to prove that it is a vertex, and so it begins the process of negotiating the line parameters between itself and the initiator while at the same time initiating a new probe.

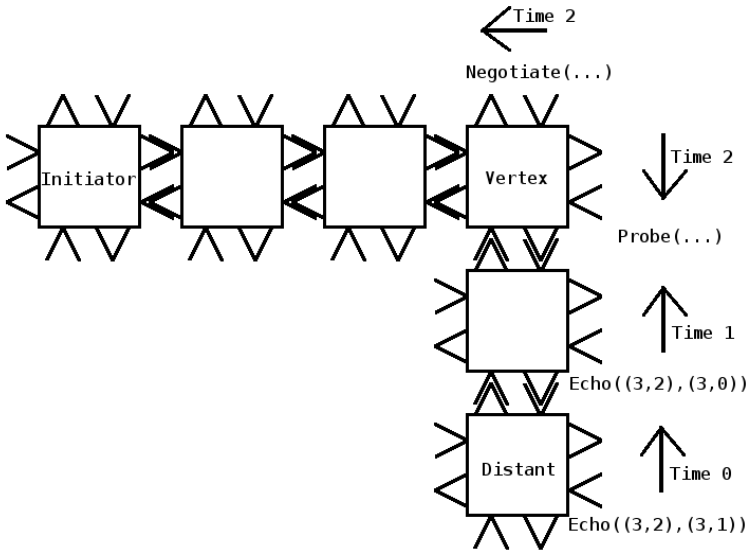

Fig. 3. The propagation of an echo message, and the probe and negotiation messages it triggers

The hop count threshold on the echo messages represents a tradeoff between precision and speed: so long as it is less than infinity, this process can misidentify a robot as occupying a vertex position, because the true vertex was beyond the hop threshold. Having a finite hop count threshold puts a linear bound on the amount of time it takes to probe a line of robots. In our experience, the result of the system is not

strongly dependant on the choice of hop count threshold, since even thresholds of only ten hops or so can err only in the case of very shallow angles.

Once neighboring vertex robots have finished negotiating the parameters of the edge between them, they begin continuously updating each other, each with the rules that the other knows. In this way the full set of rules is shared among all of the robots.

Once this has been achieved, the robots are able to construct further copies of the shape, including rescaled copies, and to repair those constructed shapes if they are damaged either during construction or after they are completed.

4. Verification

The construction technique mentioned here, and further described in [3], has been mathematically shown to correctly construct structures that approximate a polygon, given a set of vertices and edges describing the polygon. The proof for this hinges on showing that line segment approximations are correctly constructed, which can in turn be proved by showing that the constructed line segments are equivalent to those that would be drawn by Bresenham's algorithm were the line segment rasterized to a video display. This has been done. The global-to-local rule generation has not been fully mathematically analyzed as yet. The technique has been tested extensively in simulation, and seen to reliably produce qualitatively good approximations of the input shape, as in Fig 4. The architecture is presently being ported for use on the Jasmine swarm robot [4], after which physically embodied demonstrations are anticipated.

(a) (b) (c)

Fig. 4. (a) A hand-created structure, (b) The same structure after the learning process, with the vertex robots marked, (c) The self-assembled structure created by executing the learned rules

References

[1] D. J. Arbuckle and A. A. G. Requicha, "Active self-assembly", IEEE Int'l Conf. on Robotics and Automation, New Orleans, LA, pp. 896-901, April 25-30, 2004.

[1] D. Arbuckle and A. A. G. Requicha, "Shape restoration by active self-assembly", Applied Bionics and Biomechanics 2005, vol. 2, no. 2 pp. 125-130, Woodhead Publishing, Cambridge, England.

[3] D. J. Arbuckle and A. A. G. Requicha, "Self-repairing self-assembled structures," Proc. IEEE Intl Conf. on Robotics and Automation (ICRA 06), Orlando, FL, May 15-19 2006

[4] http://swarmrobot.org/