The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

An Empirical Study of the Performance of Active Self-Assembly

Thanaphon Tangchoopong and Aristides A. G. Requicha, *Life Fellow, IEEE*

Abstract—**Several approaches have been proposed recently for building shapes with swarms of self-assembling robots. However, there is a dearth of information about the performance of each approach, and how to compare them. This paper considers the active self-assembly scheme introduced by Arbuckle and Requicha, and investigates its performance through extensive simulations. The difficulties encountered in the evaluation of self-assembly schemes are discussed. Empirical simulation data are presented that show that the time for completion of the boundary of a polygon by the active self-assembly scheme is approximately linear in the size of the polygon, for the range of parameters investigated.**

I. Introduction

Robot swarms are interesting examples of decentralized systems whose global behavior emerges from local rules. Potentially, they can be remarkably robust in the presence of faults, and can adapt to dynamically changing environments. Construction tasks that involve building a specified spatial shape by a swarm of self-assembling robots are especially interesting and challenging. Although these tasks are, in principle, independent of spatial scale, they are likely to be most applicable at the micro and nanoscales. One can conceive of large numbers of identical robots being produced by MEMS (Microelectromechanical Systems) or NEMS (Nanoelectromechanical Systems) techniques, and assembling themselves into shapes that are used for such applications as scaffolds for electronic systems or for organ development *in vivo*. Note that results obtained for macroscopic scales do not necessarily extend to smaller spatial scales because the physics is quite different.

Several algorithms and systems for construction tasks by robot swarms have been proposed in the last few years—see e.g. [1]-[12]. However, the performance of these systems is not well understood, and the systems are difficult to compare. In this paper we will focus on the active self-assembly approach introduced by Arbuckle and Requicha [1]-[6], and investigate its performance, in an attempt to gain some insight on how to evaluate this and similar systems. More specifically, we will consider the latest version of the system, which uses reactive rules and stateless robots [3]-[6].

The remainder of the paper is organized as follow. First we discuss briefly the principles of operation of the active self-assembly system. Next, we address performance criteria and parameter spaces. The bulk of the paper presents the results of extensive simulations, starting with an assessment of the stochastic nature of the system, then looking at the influence of some of the major parameters, and finally showing the relationship between time-to-completion versus input complexity.

T. Tangchoopong is with the Computer Science Department, University of Southern California, Los Angeles, CA 90089-0781 (e-mail: tangchoo@usc.edu).

A. A. G. Requicha is with the Laboratory for Molecular Robotics, University of Southern California, Los Angeles, CA 90089-0781 (phone: 213-740-4502; fax: 213-740-7512; e-mail: requicha@usc.edu).

This work was supported in part by the Okawa Foundation.

II. The Active Self-Assembly Scheme

The robots in this scheme are all identical and identically programmed. They are unit squares aligned with the axes of the Euclidean plane, and move only by translation, maintaining their orientation. Initially, they execute a random walk. When a robot meets another, it may attach to it and exchange messages. Adjacent robots remain attached as long as messages are being sent and received between them; if messaging stops, the strength of the attachment decays with time, and when it reaches zero the robots detach.

An offline compiler processes a boundary representation (i.e., a list of edges) of the goal polygon, and produces a set of purely reactive rules that constitute the program to be executed by each robot. A typical rule specifies an action to be performed when a given message is received; the action usually involves sending other messages.

Two sets of messages circulate in opposite directions along the boundary of a polygon being built. One set instructs the robots to build edges and vertices. The other sends acknowledgements backwards, and is necessary in this scheme to ensure that the process is self-repairing in the presence of robot faults, message corruption and message dropping.

III. Performance Measures

The performance of an algorithm is usually measured in computer science by its execution time as a function of input size. This seems like a reasonable measure for active self-assembly as well. The simulation time for a given shape can be measured easily, but we immediately run into a difficulty with the input size. How do we measure the size of a polygon? Should we use the diameter of the set (largest value of the distance between two points of the set), the length of the boundary, the number of edges, or yet something else? For lack of a convincing answer to these questions, we decided to finesse the problem by using only square objects. For a square, the size can be measured unambiguously as the number of units in an edge, and other possible measures such as diameter or perimeter are proportional to the edge length. This decision has significant consequences, as we will show in the results section: the completion time depends on the shape of the polygon to be constructed.

The active self-assembly algorithm produces filled polygons but takes an asymptotically long time to do it with a high probability. Therefore, we decided to study the completion time for the *boundary* of the polygon. The simulator can keep track of how many robots are on the polygon's boundary at each simulation step. Since we know how many robots are needed to complete the boundary, we can detect boundary completion and measure the corresponding simulation time.

IV. Parameter space

Robotic swarm systems for self-assembly tend to have a large number of parameters. This results in a high-dimensional parameter space, in which it is impractical to search for optimal performance. In this work we leave all of the parameters at their empirically-determined "satisficing" values that were used in the experiments reported in earlier papers [1-6], and change only two.

The simulation starts with an initial set of N robots randomly distributed on the plane. Conceptually, we sample a Gaussian distribution to find the x coordinate of a robot, and sample the same distribution to find its y coordinate. We do this for the N robots to generate the initial conditions. The Gaussian distributions can be centered at some arbitrary point, but for the experiments in this paper we set them at the origin (unless noted otherwise). The standard deviation σ of the Gaussians is another important parameter. N and σ are the two parameters we will vary in this study.

Intuitively, it is clear that the number of robots and their "concentration" or *density* near the object being constructed should have an effect on the boundary completion time T. From elementary probability theory it is easy to derive the number of robots that fall within a square that goes from $-\sigma$ to $+\sigma$, both in x and y. It is proportional to N, and the constant of proportionality is the square of the error function evaluated at $2^{-1/2}$, which is a fixed number, approximately 0.5. Therefore, about half of the robots are initially within the $(-\sigma, +\sigma)$ square. We define robot density as $d = N/4\sigma^2$. From the previous discussion it follows that this density is proportional to the quotient between the number of robots that initially fall in the $(-\sigma, +\sigma)$ square and the area of the square. We can increase the density by increasing the total number of robots, N, or by decreasing the standard deviation, σ. We will see below that both of these actions have a significant impact on the completion time.

The active self-assembly system has a goodly number of additional parameters. Here is a list of most of these (we will not explain what they mean, since that would require a long digression): message dropping rate (5%); signal to noise ratio (10% of messages are 1-bit corrupted); seed growing direction; latch range; bind length; period for checking unbinding; maximum relaxation time; collision buffer distance; send message delay time; flip bit position; pushing direction on collision; and Gaussian distributions sampled to determine the next motion distance, the next event time, and the time of the next decay event. We are quite sure that some of these 14 additional parameters beyond N and σ have a significant influence on the running time T, but we kept them constant for practical reasons. All of the experiments reported in this paper were run with noise, by assuming a 5% probability of dropping a message and a 10% probability of corrupting a message by a random bit flip.

V. Experimental Results

A. Stochastic Variation

The system involves a variety of stochastic behaviors, from the robot initial positions to their random motions, as well as the simulated faults such as message dropping and message corruption. Therefore we would expect that completion times would be random as well. To assess the magnitude of these stochastic effects we ran the same experiment repeatedly, and measured the boundary completion times. Fig. 1 shows the results for a series of 50 runs for a box of size S=40 centered at the origin, with d=1/2, σ=28.8, N=1,600. The mean value is 5,159 and the standard deviation is 613. The standard deviation is on the order of 10% of the average value. We judged this acceptable and did 50 runs for each of the experiments presented below.

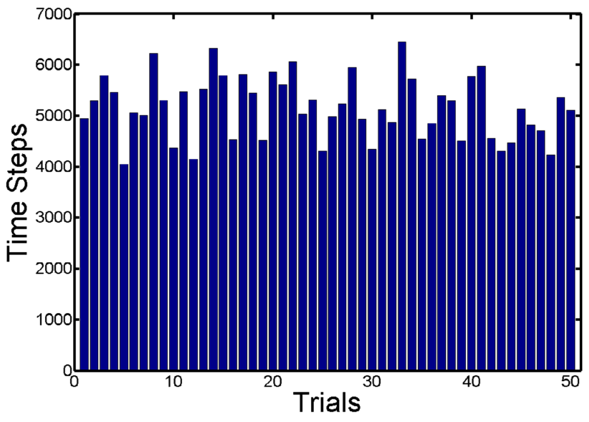

Fig. 1. Times to completion of 50 runs for a box of size 40.

(In all of our experiments we encountered a small number of failures in the first time steps, due to low binding strength of the initial seed. We simply discarded runs in which this happened, rather than modifying the code. Therefore, our results were obtained exactly with the same system discussed in our earlier papers.)

B. Standard Deviation of the Injection Distribution

Here we control the density of robots, as defined above, by changing the standard deviation, σ, of the Gaussian distribution of initial positions, also known as the *injection distribution.*

We assume the robots have a unit edge size, and we measure length in units of robot size. In the series of experiments reported in this section, we use a fixed object, which is a box (i.e., a square) of size S=40 and centered at the origin. We keep the total number of robots fixed at $N=1{,}600=S^2$ and vary σ as follows:

$$\sigma = n^{1/2}.S/2, \text{ for } n = 1, 2, 3, 4, 5.$$

This corresponds to densities d = 1, 1/2, 1/3, 1/4 and 1/5. For each value of σ we run 50 times and calculate the mean and standard deviation of the 50 resulting times for boundary completion. Fig. 2 plots these values as a function of σ. Of course, T is expected to increase when the density decreases, as σ increases and the robots are spread over a larger area. The behavior depicted in Fig. 2 is non-linear. Fig. 3 is a similar plot, but with 1/d in the abscissa. Interestingly, the variation of time T with 1/d is very nearly linear. Note that, with other parameters fixed, 1/d is proportional to σ^2. Therefore, T is linear in the area of the standard deviation square (–σ, +σ)×(–σ, +σ).

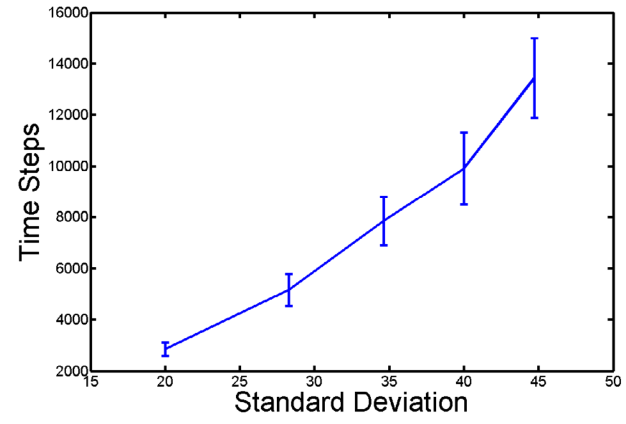

Fig. 2. Mean time of completion of a box of size 40 as a function of σ for fixed N. The bars in this and other figures denote the standard deviations of the results.

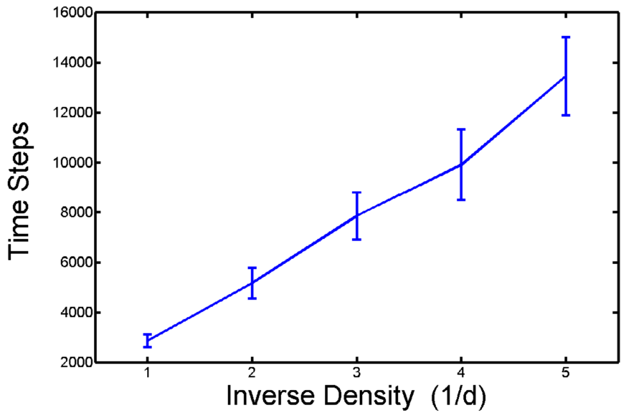

Fig. 3. Mean time of completion of a box of size 40 as a function of the inverse density 1/d for fixed N.

C. Number of Robots

In another series of experiments we control the density through the number of robots, N. We use the same object, a box with S=40, and the same set of densities, d = 1, 1/2, 1/3, 1/4, 1/5. We fix $\sigma = S.2^{-1/2} \sim 28.8$, and let N take the values N = 3,200; 1,600; 1,066; 800; and 640. The results are shown in Fig. 4, with T as a function of N, and in Fig. 5 with T as a function of 1/d. Note that d is proportional to N when all the other parameters are fixed.

Now the behavior is strongly nonlinear. We had hoped that the time to completion would be independent of N and σ taken separately, and would depend only on d. Unfortunately, this is not the case. The number of robots has a stronger impact than the standard deviation of the injection distribution.

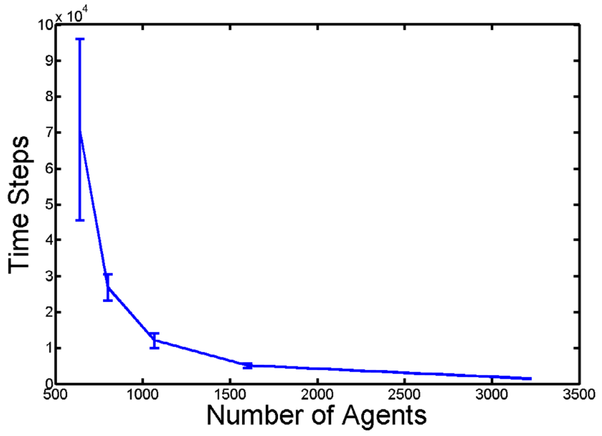

Fig. 4. Mean time of completion of a box of size 40 as a function of N for σ constant.

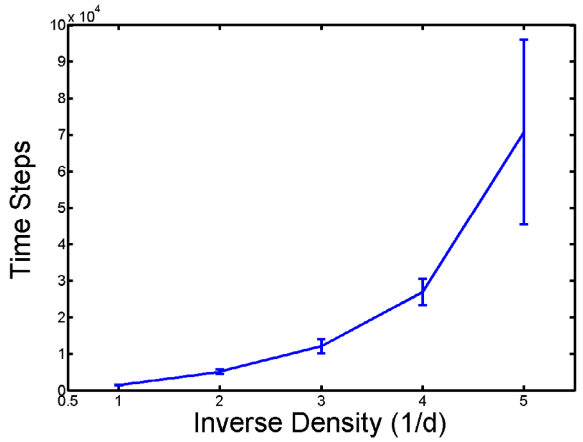

Fig. 5. Mean time of completion of a box of size 40 as a function of 1/d for σ constant.

D. Object Shape

Here we compare the time of completion for 3 different geometries (see Fig. 6): a key-like shape, a box, and an octagonal shape. These last two are centered at the origin.

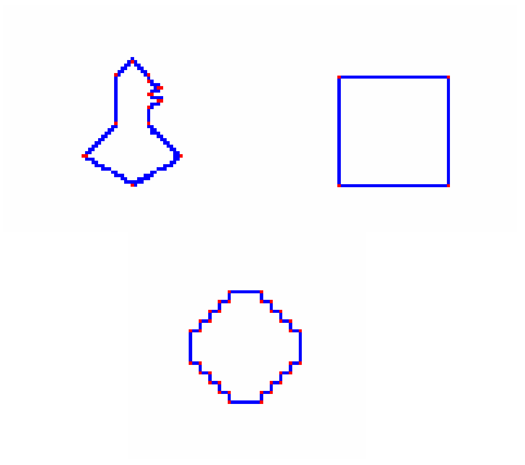

Fig. 6. The three shapes being compared.

The key has an overall size of 34x40 units, while both the box and the octagon have a size of 34x34. The key boundary is composed of 151 robots, whereas the box and the octagon's boundaries both have 136 robots. The Gaussian injection distribution is centered at the origin for the box and octagon, and at the center of gravity of the key, at x=0, y=17. For the 3 shapes we use $\sigma = 2^{1/2}.17=24.04$, and a density d=1/2, which correspond to a number of robots N=1,156. The key has 13 vertices, whereas the box has 4 and the octagonal shape 36. As usual, we perform 50 runs and compute the means and standard deviations for the completion times for each of the 3 shapes. The results are shown in Table 1. It is clear that the running time is significantly impacted by the specific shape being constructed, and not just by its size, or the number of robots on its boundary, or the number of its vertices.

TABLE 1 – MEAN AND STANDARD DEVIATION FOR THE COMPLETION TIME OF 3 COMPARABLE SHAPES

Shape	Mean	Std. Deviation
Key	8,242	1,074
Box 34	4,664	360
Octagon	5,542	327

E. Time vs. Input Size

Finally, we address the major issue we investigate in this paper: the behavior of the boundary completion time T as a function of the input size. The results in the previous sections can be summarized as follows. For shapes with comparable sizes measured by their overall dimensions, or their perimeters, or their number of vertices, T depends on the specific shape. For a fixed shape (a box, in our experiments), T depends approximately linearly on the inverse density 1/d when we keep N fixed and vary σ. However, when we vary N and fix σ the behavior is nonlinear. Given these results, how should one proceed to investigate the time vs. input size dependency?

This question does not have an obvious or unique answer. We decided to proceed as follows. To avoid the dependence on shape, we worked only with squares centered at the origin. We measured the input size by the edge size, S, of the squares. Intuitively, it seems that we should keep the robot density, d, constant for fair comparisons. The results above, plus many other runs not reported here, indicate that the system operates well when the density is d=1/2, and therefore we fixed it at that value in all the experiments reported in this section. But we know from the previous sections that both N and σ impact T, albeit with different strengths. Again on intuitive grounds, the joint configuration of the object and the injection distribution should be held constant. To achieve this, we decided to make the standard deviation of the injection distribution proportional to the object size, and, guided by the results in the previous sections, chose to operate with $\sigma = S.2^{-1/2}$. Since $d=N/4\sigma^2$ and we wanted d=1/2, we used $N=2\sigma^2=S^2$.

Fig. 7 shows the means and standard deviations of 50 runs for each of the squares. The square sizes are S = 20, 40, 60, 80 and 100. The corresponding numbers of robots are N = 400; 1,600; 3,600; 6,400; and 10,000. As noted earlier, σ is calculated by $\sigma = S.2^{-1/2}$ for each object.

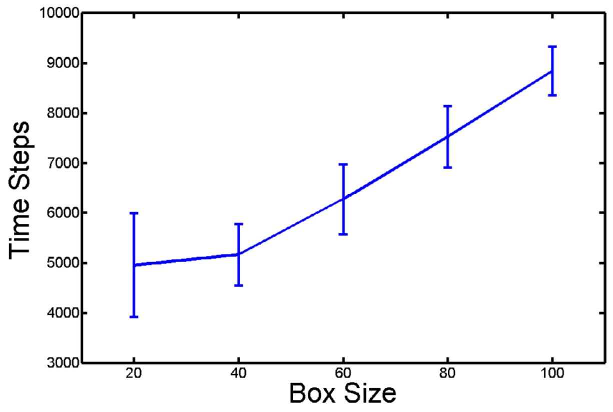

Fig. 7. Mean time of completion as a function of box size, for fixed density.

If we discard the point that corresponds to the box of size 20, Fig. 7 shows that the boundary completion time T is linear on the size S. Note that the standard deviation for the box of size 20 is much larger than the others, and is nearly 20% of the mean value. We generally find that the system behaves somewhat erratically when the object size is relatively small. We speculate that this may be due to the robot size being a non-negligible fraction of the object size.

VI. DISCUSSION

The linear behavior of the boundary completion time with the size of the object for the active self-assembly scheme is a nice result, but one should not read more into it than what is implied by the experiments. We only showed linearity for a specific set of conditions and range of parameters, and extrapolation to other settings may not be justified.

The work reported in this paper shows how difficult it is to assess the performance of a complex self-assembly scheme. Barring any major theoretical insights, which are unlikely, performance must be evaluated experimentally. The systems tend to have a large number of parameters, whose effects on the performance measures are often nonlinear and not necessarily independent, and the simple approach of varying one parameter while others are fixed may not be very useful. Because the systems are stochastic, a large number of runs are necessary to obtain meaningful results. Therefore, a detailed exploration of the parameter space is seldom practical. Some issues are inherently problematic, for example, how does one deal with the influence of object geometry?

Where do we go from here? As a first step, we suggest that the small community interested in construction tasks by robot swarms agree on a set of benchmarks that can be used to test and compare systems. Benchmarks, however imperfect, may be the only feasible approach to deal with the important problem of self-repair, which we ignored in this paper. We believe that self-repair is an essential property of any robotic swarm used for construction tasks, because some of the robots of the swarm will almost surely fail during the task's execution. How should self-repair capabilities be quantitatively assessed?

ACKNOWLEDGMENT

We thank Dan Arbuckle, who originally designed and implemented the active self-assembly system, for advice and help with his system.

REFERENCES

[1] D. J. Arbuckle and A. A. G. Requicha, "Active self-assembly", *Proc. IEEE Int'l Conf. on Robotics & Automation (ICRA '04)*, New Orleans, LA, pp. 896-901, April 25-30, 2004.

[2] D. J. Arbuckle and A. A. G. Requicha, "Shape restoration by active self-assembly", *Applied Bionics and Biomechanics*, Vol. 2, No. 2, pp. 125-130, 2005.

[3] A. A. G. Requicha and D. J. Arbuckle, "CAD/CAM for nanoscale self-assembly", *IEEE Computer Graphics and Applications*, Vol. 26, No. 2, pp. 88-91, March/April 2006.

[4] D. J. Arbuckle and A. A. G. Requicha, "Self-repairing self-assembled structures", *Proc. IEEE Int'l Conf. on Robotics & Automation (ICRA '06)*, Orlando, FL, pp. 4288-4290, May 15-19, 2006.

[5] D. J. Arbuckle and A. A. G. Requicha, "Global-to-local rule generation for self-assembly and self-repair by robot swarms", *Proc. 4th Conf. on Foundations of Nanoscience (FNANO '07)*, Snowbird, UT, pp. 251-255, April 18-21, 2007.

[6] D. J. Arbuckle and A. A. G. Requicha, "Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: algorithms and simulations", Tech. Rept. AR-07, Laboratory for Molecular Robotics, University of Southern California, Los Angeles, CA, 2007, http://www-lmr.usc.edu/~lmr/publications/SwarmRept.pdf .

[7] A. L. Christensen, R. O'Grady and M. Dorigo, "SWARMORPH-script: a language for arbitrary morphology generation in self-assembling robots", *Swarm Intelligence*, Vol. 2, Nos. 2-4, pp. 143-165, December 2008.

[8] C. V. Jones and M. J. Matarić., "From local to global behavior in intelligent self-assembly", *Proc. IEEE Int'l. Conf. on Robotics and Automation (ICRA '03)*, Taipei, Taiwan, pp. 721-726, Sep 14-19, 2003.

[9] E. Klavins, "Directed self-assembly using graph grammars", *Proc. 1st. Conf. on Foundations of Nanoscience: Self Assembled Architectures and Devices*, Snowbird, UT, 2004.

[10] M. Rubenstein and W.-M. Shen, "A scalable and distributed approach for self-assembly and self-healing of a differentiated shape", *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS '08)*, Nice, France, September 22-26, 2008.

[11] J. Werfel, D. Ingber and R. Nagpal, "Collective construction of environmentally-adaptive structures", *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS '07)*, October 2007.

[12] J. Werfel, Y. Bar-Yam and R. Nagpal, "Building patterned structures with robot swarms", *Proc. Int'l. Joint Conf. on Artificial Intelligence (IJCAI '05)*, August 2005.