Online Dispersion Algorithms for Swarms of Robots

Tien-Ruey Hsiang
Stony Brook University
Stony Brook, NY 11794-3600
trhsiang@ams.sunysb.edu

Esther M. Arkin
Stony Brook University
Stony Brook, NY 11794-3600
estie@ams.sunysb.edu

Michael A. Bender
Stony Brook University
Stony Brook, NY 11794-4400
bender@cs.sunysb.edu

Sándor Fekete
Braunschweig University
D-38106, Germany
sandor.fekete@tu-bs.de

Joseph S. B. Mitchell
Stony Brook University
Stony Brook, NY 11794-3600
jsbm@ams.sunysb.edu

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*geometrical problems and computations, sequencing and scheduling*; I.2.9 [**Artificial Intelligence**]: Robotics—*autonomous vehicles*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*intelligent agents, multiagent systems*

General Terms

Algorithms, Experimentation

Keywords

swarm robotics, approximation algorithms, NP-hardness

1. INTRODUCTION

We illustrate algorithms for dispersing a swarm of primitive robots in a two-dimensional unknown environment R. Each robot in the swarm is equipped with a simple sensor that is able to view neighboring locations to determine the presence of other robots or obstacles. At each time step, based on the sensor readings, a robot may decide to take a step to a neighboring point. The objective is to minimize the *makespan*, that is, the time to *fill* R with robots.

Here, we consider environments that are discrete, composed of unit squares (*pixels*) that are induced by the integer grid within a polygonal domain R. There is at most one robot per pixel and robots move horizontally or vertically at unit speed. Robots enter R by means of $k \geq 1$ *door pixels* on the boundary of R, each of which acts as an infinite source of robots. The domain R is filled when there is a robot in each pixel.

Robots are primitive finite automata, only having local communication, local sensors, and a constant-sized memory. These local autonomous agents are not centrally controlled, yet are expected to perform the global task of dispersing.

In a recent paper [2], we provide a variety of theoretical results, including:

- Algorithms for the single-door case (i.e., $k = 1$), based on leader-follower strategies. Our algorithms are motivated by depth-first and breadth-first search but are applicable to the local nature of the robot model. We prove that our algorithms have optimal makespan $2A-1$, where A is the area (the number of pixels) of R.
- An algorithm for the multi-door case ($k \geq 1$), based on a wall-following version of the leader-follower strategy: We prove that our strategy is $O(\log(k+1))$-competitive, and that this bound is tight for our strategy as well as other natural "simple-minded" strategies that use only strictly local information.

The purpose of this video is to illustrate via animations the results of implementing our dispersion strategies within a simulator. This has been developed as part of a joint project on "pheromone robotics" for world-embedded computations, under DARPA funding, with HRL Laboratories (Payton et al. [3]). The dispersion problem studied here grew out of our discussions with Payton and with Gage, one of the pioneers of swarm robotics [1].

The video shows how even simple local rules lead to complex emergent behaviors of a swarm of robots. These global behaviors are relatively easy to understand visually, while being difficult to convey via traditional text in a paper.

While our results here are given purely in terms of robots moving synchronously on discrete grids, the results apply also to dispersing swarms of robots within an arbitrary connected planar domain, in the ideal setting in which the robots have perfect motion control and synchrony.

2. THE MODEL

We assume that at most one robot can occupy any one pixel at any given time t. Time is discretized into steps $t = 0, 1, 2, 3, \ldots$. At time 0, there is one robot in each door pixel. Each pixel is in one of three states at time t: (1) the pixel is an *obstacle* if it does not lie in R; (2) the pixel is *occupied* if it is in R and a robot occupies it; or (3) the pixel is *unoccupied* if it is in R and no robot occupies it at time t. An unoccupied pixel at time t is classified as either *previously occupied*, if it was occupied by some robot at some time prior to t, or *frontier*, if it has never been occupied. Robots have *sensors* that detect information about the local

SoCG'03, June 8–10, 2003, San Diego, California, USA.
ACM 1-58113-663-3/03/0006.

environment. In particular, if a robot occupies pixel (i, j) at time t, then we assume that it can detect the state of each pixel within a distance r_S of (i, j), where r_S is the (fixed) *sensor radius*. Robots have no global sensor and no knowledge of the coordinates of the pixels they occupy.

Each robot has a small finite memory (independent of the size of R) that allows it to remember the sensor readings of the last $T \geq 1$ time steps; i.e., a robot knows the sensor readings it has taken at time $t-T, t-T+1, \ldots, t-1, t$. In particular, each robot knows which nearby pixels have been occupied recently by other robots.

Each robot has a limited ability to communicate with nearby robots. In particular, at any given time step a robot is able to exchange a constant-size message with any robot that lies within a prescribed *communication radius*, r_C, of the robot. The *communication graph*, $\mathcal{G}(t)$, of the swarm $S(t)$ at time t is defined to be the undirected graph whose nodes are the robots $S(t)$ and whose edges link pairs of robots that communicate. The swarm is *connected* at time t if each connected component of $\mathcal{G}(t)$ contains a door pixel.

A *strategy* is a set of rules by which robots move, basing decisions solely on the constant amount of information they sense and remember. The *makespan* of a strategy is the minimum time, t^*, until the robots have filled R.

3. THE VIDEO SEQUENCE

The video opens with an animation of the filling of a complex room. The rest of the video explains the components of the filling algorithm.

The explanation of the algorithm begins by displaying a polygonal domain with two doors, each of width one, and shows how the robots fill the domain in leader-follower fashion. Then the video demonstrates the movement rules within each chain, where each robot has to wait until the pixel in front of it is vacant before moving.

Next, an animation shows an example of dispersal into a domain through a single door ($k = 1$) via a simple depth-first-search strategy. This is followed by a multiple-door example that illustrates the possibility of unwanted "blocking": The chains of robots on the left reach a dead end and are stopped, leaving only the rightmost chain of robots still active to continue to fill the region; this is wasteful because the strategy fails to use the full width of the door.

A solution to the problem of blocking is *splicing*, where one chain of robots "cuts into" a neighboring chain of robots in order to keep door pixels actively feeding robots into the domain. Splicing must be implemented carefully in order to avoid cycling. A short animation shows how cycles can arise from a straightforward leader-follower algorithm, in which a robot that has no frontier pixel next to it chooses to go to any unoccupied neighboring pixel (splicing into the leader-follower flow passing through this pixel).

Our strategy for multiple doors ($k > 1$) avoids the pitfalls of splicing by allowing splicing to occur in a controlled way: a robot can "cut into" a flow only from one side of the flow (the left side), as illustrated in the next short animation sequence. This asymmetry, coupled with a prioritized movement rule (a robot prefers to go left, then to go forward, then to go right), results in a strategy that effectively makes the robots "hug" the left wall as much as possible. The robots executing this strategy fill the domain in a "laminar flow" behavior, which results in a provable online strategy, with makespan within a factor $O(\log(k+1))$ of optimal [2]. (A figure in the video shows an instance for which this competitive ratio is optimal, assuming a strategy using only local information.)

The next animation shows the simulator with an environment having 3 doors (of widths 2, 3, and 7), with the Laminar Flow Leader-Follower (LFLF) algorithm. The video concludes with animations showing the filling of two classes of "combs" whose teeth have widths that grow exponentially: in one case (Fig. 1), the left hand on wall LFLF strategy works optimally; in the other case (Fig. 2), the strategy requires makespan that is a factor $\Theta(\log k)$ times optimal.

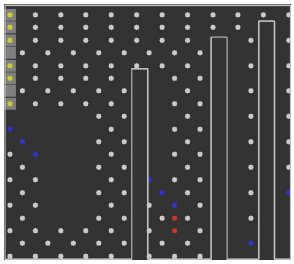

Figure 1: An example shot from the video showing a flow of robots dispersing through a door of width 8: In this environment, the LFLF strategy yields an efficient filling of the comb-shaped room.

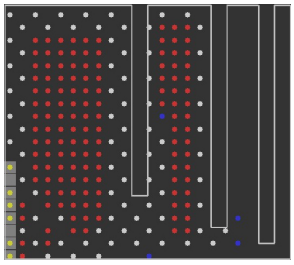

Figure 2: In this (flipped) environment, the LFLF strategy yields an inefficient filling of the comb-shaped room, since blockage leads to only a small number of doors being useful later in the dispersal.

4. ACKNOWLEDGMENTS

E. Arkin and J. Mitchell are partially supported by NSF (CCR-0098172). M. Bender is partially supported by NSF (EIA-0112849, CCR-0208670) and by a grant from Sandia National Labs. J. Mitchell acknowledges support from Honda Research Labs, NASA Ames Research (NAG2-1325), and the U.S.-Israel Binational Science Foundation.

5. REFERENCES

[1] D. Gage. Sensor abstractions to support many-robot systems. In *Proc. of SPIE Mobile Robots VII*, Boston, Vol. 1831, pages 235–246, 1992.

[2] T.-R. Hsiang, E. M. Arkin, M. Bender, S. P. Fekete, and J. S. B. Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. *5th International Workshop on Algorithmic Foundations of Robotics*, 2002.

[3] D. Payton, M. Daily, R. Estkowski, M. Howard, C. Lee. Pheromone Robotics. *Autonomous Robots*, Vol. 11, No. 3, pages 319-324, 2001.