Cellular ANTomata: Food-Finding and Maze-Threading

Arnold L. Rosenberg
Electrical and Computer Eng'g, Colorado State University, Fort Collins, CO 80523, USA
rsnbrg@engr.colostate.edu

Abstract

A model for realizing ant-inspired algorithms that coordinate robots within a fixed, geographically constrained environment is proposed and illustrated. The model, dubbed Cellular ANTomata*, inverts the relationship between ant-robots and the enviroment that they navigate: intelligence now resides in the environment rather than in the ants. The Cellular ANTomaton model is illustrated via three proof-of-concept problems: having ants "park" in the nearest corner; having ants seek "food items" (both with and without impenetrable obstacles); having a single ant thread a maze. In all cases, "unintelligent" Cellular-ANTomata-based ant-robots accomplish goals provably more efficiently than traditional "intelligent" ant-robots can; indeed, "intelligent" ant-robots cannot park at all! All of the presented algorithms are* scalable*: they provably work within any finite-size environment.*

1. Introduction

As we encounter novel computing environments that offer new opportunities while posing new challenges, it is natural to seek inspiration from natural analogues of these environments. Thus, empowered with technology that enables mobile intercommunicating robotic computers, it is compelling to seek inspiration from social insects—in 2-dimensional settings, mainly ants—when contemplating how to use the computers effectively and efficiently in environments that defy centralized control; many sources (cf. [2, 3, 6]) have done precisely that. After considering the strengths and the weaknesses of the robot-as-ant metaphor in the light of current computer technology, we propose a variant of cellular automata [1, 18]—*Cellular ANTomata*—as a platform for developing the algorithmics of robotic mobile computers within geographically constrained environments. We formalize the proposed model in detail and illustrate it via "proof-of-concept" problems that have ant-robots move and aggregate in various ways.

1.1 Motivating Cellular ANTomata. Our use of cellular automata as a conceptual/algorithmic platform arises from the following considerations. While nature is a valuable source of inspiration in domains where it operates successfully—such as the functioning of social insects—one must not follow nature too literally. Some features and behaviors observed in social insects retain useless remnants of evolutionary cul-de-sacs. Others depend on realities in nature, such as the availability of multitudes of expendable individual agents, that differ sharply from the realities in the world of robotic mobile computers. We therefore strive for a conceptual algorithmic platform that adapts features inspired by the former world to the exigencies of the latter—at least within geographically constrained environments.

1.2 Ant-Robots in a Laboratory. We focus on situations wherein robotic mobile computers (henceforth, *"ants"*) function within a fixed geographically constrained environment. We expect ants to be able to:

- navigate the environment, avoiding all collisions;
- communicate with and sense one another, by "direct contact" (as when real ants meet) and by "time-stamped message passing" (as when real ants deposit volatile pheromones);
- discover goal objects (call them *"food"*) and convey food from one location to another;
- assemble in desired locations and configurations.

A standard approach. A natural approach to achieving artificial ants is to enhance electro-mechanical robots that have both mobility and grasping/conveying capability with additional "machinery" that enables computation and both direct and "time-stamped" communication. Indeed, researchers (cf. [6]) have equipped robots with small transceivers that serve as generators and receptors of "virtual" pheromones. Such an avenue to artificial ants may be inevitable when robots must navigate unbounded (e.g., external) environments. But when robots are to function as automated assistants in a geographically constrained environment such as a laboratory floor, two features of this approach call out for emendation.

1. The likely necessary level of use of each ant's computing and communicating "machinery" will tax any battery technology.

2. The potential for accidents such as unintended impacts will greatly increase the cost of each ant, via either the extra weight needed to insulate electronics or the frequent replacement of incapacitated/damaged ants.

An alternative approach. We revisit the relationship of ants to their environment (henceforth, the *"(laboratory) floor"*). In the "standard" approach, intelligence and initiative reside in the ants; the floor is a brainless, passive environment which is just a physical platform upon which ants sit and move. While such an organization of the "world" is unavoidable when ants operate in unconstrained, uncontrollable environments, it is eminently avoidable on a laboratory floor. We now invert the active-passive relationship between ants and the floor: we tesselate the floor with identical tiles, embedding within each tile a copy of a fixed computer of modest capability. We posit that each computer has:

- a number of I/O ports that is sufficient for necessary communications;
- c registers that record levels of c types of virtual pheromones ([6]) whose volatility is modeled by a scheduled decrementing of the associated register.

In a single "step," each tile/computer is capable of:

1. detecting whether or not it has upon it: • an obstacle or a portion thereof (e.g., a wall covering many tiles); • an ant; • a food item that an ant can pick up and/or move and/or manipulate; • both an ant and a food item.

2. communicating with neighbors—tiles it shares an edge or corner with—by receiving/transmitting one message from/to each in each "step;" sample messages could be: "I DO (NOT) HAVE AN ANT;" "I DO (NOT) HAVE FOOD;" "I DO (NOT) CONTAIN AN OBSTACLE;" "I HAVE LEVEL ℓ_i OF PHEROMONE i."

3. communicating with an ant that resides on the tile, via messages such as: "PICK UP THE FOOD;" "MOVE TO THE NEIGHBORING TILE IN DIRECTION D" (D is a compass direction).

Now, ants: • never collide with obstacles or other ants: they move only by command of the tile they reside on—which communicates constantly with its neighbors; • are now much simpler, containing only the electronics needed to receive commands from the tile they reside on. Consequently, malfunctions/accidents are much rarer and less expensive than with "smart" ants.

1.3 Two Basic Assumptions.

(a) Laboratory floors must be *scalable* in structure and efficiency; e.g., computers may not exploit information about the size of the floor (number of tiles). (b) Our model is "semi-synchronous," in that computers assemble inputs from all neighbors before committing to any action. This does *not* mean that all computers hear the tick of the same clock, but, rather, that variations in clocking between neighboring computers are small (since they come from geographical neighbors). That said, it is easy to implement our algorithms in a *completely distributed manner*, with all coordination among computers being via explicit handshakes, rather than via shared "clock ticks."

1.4 Contributions.

Within this model, we seek algorithms that accomplish three "proof-of-concept" tasks:

1. We have ants "park" in the closest corner of the floor, with ants heading for the same corner configuring themselves as compactly as possible.
2. We have ants seek food items, with one ant per item and one item per ant, as numbers permit. Our algorithm accommodates impenetrable obstacles that block both ants and messages.
3. We have a single ant thread a maze.

When the laboratory floor is $n \times n$:	
Parking:	Time:
Our Algorithm:	$O(n^2)$.
"Intelligent" Ants:	Parking is impossible!
Food-Finding:	Time:
Our Algorithm:	r ants and s food items: $O(\min(nr, n\sqrt{s}))$.
"Intelligent" Ants:	$\Omega(n^2)$, even if $r = s = 1$.
Maze-Threading:	Time:
Our Algorithm:	proportional to length of longest entrance-exit path.
"Intelligent" Ants:	$\Omega(n^2)$, even in presence of a length-$2n$ entrance-exit path.

Our algorithms require coordination among computers to be *distributed and noncentralized.*

Space limits us to high-level sketches of algorithms, analyses, hiding, e.g., how to deal with clock skew.

Related work. Our use of CA to realize ant-inspired algorithms is not original. In [3], CA underlie ant-inspired algorithm for a genre of flow problem; in [2], they are used to implement an ant-inspired clustering algorithm. In [6], virtual pheromones (numerical values) are broadcast by wireless robots (communication is free-space), to plan a gradient-marked path that a designated robot follows to a single goal object. In [11], a CA plans a route for a single robot to a single goal by having the goal greedily broadcast its position. A commercial system is described in [10], that has centrally controlled robots transfer packages between designated stations. All of these sources depart from our goals by positing synchronous models that are not scalable; e.g., the centrally programmable models have global name spaces for tiles/computers (which we prohibit). Our work thus moves CA-based applications in a new direction. The large literature on ant-inspired algorithms is not really related because of quite different foci and groundrules.

2. Cellular ANTomata

CA are natural candidates for intelligent floors, being simple arrays of *finite-state machines* (*FSMs*). Studied since the 1960s [4, 12, 17], CA are still of interest [7, 8, 19], being a model of computers [18, 19] that combines mathematical simplicity with (remarkable) efficiency for a broad range of tasks that require the tight coordination of simple agents [2, 3, 4, 8, 12]. Our adaptation of CA, *Cellular ANTomata*, is tailored to the algorithmics of ants on a laboratory floor.

2.1 Basics. Cellular ANTomata place a copy of a single FSM at each node of a *square mesh*. We present enough detail to facilitate implementing our model ("hardware") and algorithms ("Software").

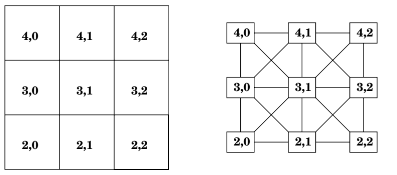

Figure 1. A 3×3 floor and its graph.

Tiled floors and meshes (Fig. 1). We tesselate a square floor with identical tiles[1] and abstract the floor as a *side-n 2-dimensional mesh*, $\mathcal{M}_n$, whose nodes are the tiles, formally, the set[2] $[0, n-1] \times [0, n-1]$. $\mathcal{M}_n$'s *King's-move* arcs (labeled by the compass directions: E, SE, S, SW, W, NW, N, NE) represent tiles' adjacencies; cf. Fig. 2(left). Node $v = \langle i, j\rangle$ is connected by mated in- and out-arcs to its (≤ 8) neighboring nodes. (Corner nodes have 3 neighbors; edge nodes have 5.)

FSMs: Finite-State Machines are given by:

- a finite set Q of *states*. $Q = K \times [0, I_1] \times [0, I_2] \times \cdots \times [0, I_\ell]$: K contains "control" variables; each $I_j \in \mathbb{N}$ is the maximum intensity of pheromone j.
- an *input "alphabet" IN*, which is the union of:
  —the messages that $\mathcal{F}$ can *receive from* neighbors,
  —$\{0, 1\}^3$: indicators of the presence of an ant, an obstacle, food;
- an *output "alphabet" OUT*, which is the union of:
  —the messages that $\mathcal{F}$ can *send to* neighbors,
  —the set C: possible commands to the resident ant (if it exists).
- a state-transition function $\delta : Q \times IN \rightarrow Q \times OUT$, associating a state $\langle k, i_1, \ldots, i_\ell\rangle$ and input $e_{IN} \in IN$ with an output $e_{OUT} \in OUT$ and new state $\langle k', i'_1, \ldots, i'_\ell\rangle$. Each $i'_j - i_j \in \{0, -1, 1\}$: a pheromone level stays stable (0 change) or evaporates (-1 change) or is reinforced ($+1$ change).

[1]We discuss only square tiles; one easily allows hexagonal tiles.

[2]$\mathbb{N}$ is the nonnegative integers. For $i \in \mathbb{N}$ and $j \geq i$, $[i, j] \stackrel{\text{def}}{=} \{i, i+1, \ldots, j\}$.

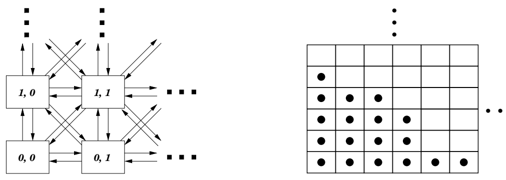

Figure 2. (Left) The 2×2 corner of a mesh. (Right) The 6×6 "prefix" of $\mathcal{Q}_{SW}$, with 18 optimally parked ants (denoted by dots).

2.2 Cellular ANTomata. Given $n \in \mathbb{N}$ and FSM $\mathcal{F}$, the *FSM-array* $\mathbf{F}_n(\mathcal{F})$ is constructed by: (a) *populating $\mathcal{M}_n$ with copies of $\mathcal{F}$.* One assigns each copy of $\mathcal{F}$ a unique *index* $\in [0, n-1] \times [0, n-1]$ and places $\mathcal{F}_{i,j}$ at node $\langle i, j\rangle$ of $\mathcal{M}_n$. (We then speak of "an internal," "a corner," or "an edge" FSM.) (b) *endowing each FSM with sensors for ants, obstacles, and food, and with a unidirectional communication channel to a resident ant.* (c) *endowing each FSM with a bidirectional communication channel to each of its King's-move neighbors.* Thus, IN and OUT for an internal FSM include the set of directional messages $S = \Sigma_N \times \Sigma_{NE} \times \Sigma_E \times \Sigma_{SE} \times \Sigma_S \times \Sigma_{SW} \times \Sigma_W \times \Sigma_{NW}$. S is suitably edited for corner and edge FSMs.

Note. **(a)** *Because we "invert" Nature, messages flow "below the surface" that entities reside on. Hence:* Obstacles atop FSMs do not impede the flow of messages. **(b)** *Choosing between King's- vs. NEWS-move arrays is a matter of cost allocation. The former are architecturally more complicated but algorithmically simpler and more efficient; the latter are architecturally simpler but require a longer state-change cycle.* **(c)** *To achieve* scalability, *we insist that algorithms treat n as an* unknown, *never exploiting its specific value.*

$\mathcal{F}_{0,0}$, the *general* of $\mathbf{F}_n(\mathcal{F})$, is the only FSM that accepts commands from the "outside world."

3. Barrier Synchronization/Activation

CA, hence Cellular ANTomata, can perform various synchronizations that are essential, e.g., when initiating a computation or transitioning from one computation to another. We describe three important such actions.

3.1 Activate All FSMs. The most fundamental synchronization, which precedes any Cellular ANTomaton algorithm, activates all FSMs "simultaneously." The *Firing Squad Synchronization Problem* (*FSSP*) begins with all FSMs in a "sleep" state. $\mathcal{F}_{0,0}$ dispatches messages in such a way that all FSMs enter a designated ACTIVE state at the same step.

Lemma 1 ([8]). $\mathbf{F}_n(\mathcal{F})$ *can simultaneously activate all of its FSMs in* $2n-2$ *synchronous steps, using messages that cross only NEWS arcs.*

3.2 Activate the Corner FSMs. Many activities require $\mathbf{F}_n(\mathcal{F})$'s corner FSMs to be synchronized.

Lemma 2. $\mathbf{F}_n(\mathcal{F})$ *can simultaneously activate its corner FSMs in* $2n-2$ *synchronous steps.*

Sketch. $\mathcal{F}_{0,0}$, sends a WAKE-UP message to its neighbors, $\mathcal{F}_{1,0}$, $\mathcal{F}_{0,1}$, $\mathcal{F}_{1,1}$, repeating the message at every step. The neighbors propagate the messages—$\mathcal{F}_{1,0}$ along its row, $\mathcal{F}_{0,1}$ along its column, and $\mathcal{F}_{1,1}$ along its diagonal—until (after $n-1$ steps) the message reaches the three corner FSMs, $\mathcal{F}_{n-1,0}$, $\mathcal{F}_{0,n-1}$, $\mathcal{F}_{n-1,n-1}$. These FSMs now echo the message along their row, column, and diagonal. When each corner FSM receives a WAKE-UP message from all three of its neighbors in the same step (which, easily, occurs at step $2n-2$), it enters the designated INITIATE-ACTION state. $\square$

3.3 Activate the Center FSM(s). When n is odd, $\mathbf{F}_n(\mathcal{F})$, has a single center FSM, $\mathcal{F}_{\lceil n/2\rceil,\lceil n/2\rceil}$; when n is even, it has four center FSMs: $\mathcal{F}_{n/2-1,n/2-1}$ and $\mathcal{F}_{n/2,n/2}$ along the SW-NE diagonal, and $\mathcal{F}_{n/2-1,n/2}$ and $\mathcal{F}_{n/2-1,n/2}$ along the NW-SE diagonal. See Fig. 3. Many activities are accomplished most efficiently when initiated by the center FSM(s).

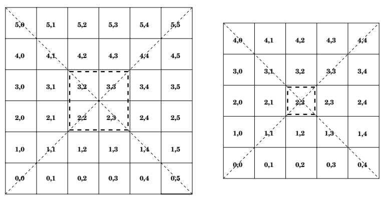

Figure 3. The center cells of meshes.

Lemma 3. $\mathbf{F}_n(\mathcal{F})$ *can activate its center FSM(s) in* $2n-2$ *synchronous steps.*

Sketch. $\mathcal{F}_{0,0}$ sends two instances of the message, ACTIVATE-CENTER: a one-time northward transmission that enlists the aid of $\mathcal{F}_{n-1,0}$; a northeasterly transmission that is repeated at every step, that "semi-activates" all FSMs along the SW-NE diagonal. Once $\mathcal{F}_{n-1,0}$ is activated, it transmits the ACTIVATE-CENTER message repeatedly in the southeasterly direction, to "semi-activate" all FSMs along the NW-SE diagonal.

Say that $\mathcal{F}_{i,j}$ receives the ACTIVATE-CENTER message from both $\mathcal{F}_{i-1,j-1}$ and $\mathcal{F}_{i+1,j-1}$. This means that n is odd and that $\mathcal{F}_{i,j}$ is the center FSM. Alternatively, say that $\mathcal{F}_{i,j}$ receives the message from $\mathcal{F}_{i-1,j-1}$, and it receives the message "I RECEIVED THE ACTIVATE-CENTER MESSAGE FROM MY NORTHWESTERN NEIGHBOR" from $\mathcal{F}_{i,j+1}$. This means that n is even and that $\mathcal{F}_{i,j}$ is the southwestern-center FSM.

The initial messages, to $\mathcal{F}_{n-1,0}$ and along the SW-NE diagonal, occupy $n-1$ steps; the messages along the NW-SE diagonal take an additional $n-1$ steps. $\square$

4. "Parking" the Ants

To put our results in context, we review the main algorithm from [14], which has an FSM-array arrange its ants "compactly" in the array's corners. More specifically, each ant moves as close as possible to the corner cell of $\mathcal{M}_n$ that resides in the same quadrant as does the ant at the moment when it receives the INITIATE_PARKING command.[3] This form of parking is rather easy to accomplish within our "inverted-world" framework of passive ants on an active earth; *it is impossible to accomplish within the standard framework of "intelligent" ants navigating a passive earth.*

Our algorithmic reasoning about parking applies also to collecting gathered "food" in designated places—perhaps a more realistic activity in a laboratory.

4.1 Formalizing the Parking Problem. The *quadrants* of $\mathcal{M}_n$ are special induced subgraphs:

Quadrant	Node-set
southwest	$[0, \lceil n/2\rceil-1] \times [0, \lceil n/2\rceil-1]$
northwest	$[\lceil n/2\rceil, n-1] \times [0, \lceil n/2\rceil-1]$
southeast	$[0, \lceil n/2\rceil-1] \times [\lceil n/2\rceil, n-1]$
northeast	$[\lceil n/2\rceil, n-1] \times [\lceil n/2\rceil, n-1]$

We simplify exposition by developing our quadrant-specific parking algorithm only for $\mathcal{Q}_{SW}$. One easily adapts the algorithm to other quadrants.

The kth *diagonal* of $\mathcal{M}_n$ ($k \in [0, 2n-2]$) is the set

$$\Delta_k \;=\; \{\langle i,j\rangle \in [0,n-1]\times[0,n-1] \mid i+j=k\}.$$

[3]We make each ant's quadrant unambiguous by defining "quadrant" asymmetrically when n is odd.

The *radius-d quarter-sphere* of $\mathcal{Q}_{SW}$ ($d \in [1, 2n-1]$) is the union of diagonals: $\bigcup_{k=0}^{d-1} \Delta_k$.

Formally, the parking problem requires the ants in each quadrant to cluster within the most compact quarter-sphere "centered" at the quadrant's corner node. Focus on $\mathcal{Q}_{SW}$, and consider Fig. 2.

A configuration of ants solves the parking problem for $\mathcal{Q}_{SW}$ iff it minimizes the parking potential function:

$$\Pi(t) \overset{\text{def}}{=} \sum_{k=0}^{2n-2} (k+1) \times \text{(the number of ants on } \Delta_k \text{ at step } t).$$

Thus, if parking begins with $\binom{m}{2} + p$ ants in $\mathcal{Q}_{SW}$ ($p \le m$), then at some step $t^\star$, for all $t > t^\star$, $\Pi(t) = \sum_{i=1}^{m-1} i^2 + mp = \frac{2m-1}{3}\binom{m}{2} + mp$.

4.2 The Parking Process. We begin by partitioning the parking problem into four independent quadrant-specific subproblems (that are solved in parallel). We first let ants know which quadrant they reside in. When $\mathcal{F}_{i,j}$ learns, via the activation of center FSM(s) in Section 3.3, that it is a center FSM, it sends three messages: • "INITIATE_PARKING_NW" to its northern neighbor, $\mathcal{F}_{i+1,j}$; • "INITIATE_PARKING_NE" to its northeastern neighbor, $\mathcal{F}_{i+1,j+1}$; • "INITIATE_PARKING_SE" to its eastern neighbor, $\mathcal{F}_{i,j+1}$. $\mathcal{F}_{i,j}$ then initiates the actual parking algorithm. An FSM that receives the message "INITIATE_PARKING_D" ($D \in \{NW, NE, SE\}$) begins the following process, which we describe for the center FSM that controls $\mathcal{Q}_{SW}$.

1. $\mathcal{F}_{\lceil n/2\rceil-1,\lceil n/2\rceil-1}$ tells its three neighbors in $\mathcal{Q}_{SW}$ to terminate parking-initiation and initiate parking.
2. Each subsequent $\mathcal{F}_{i,j}$ relays the message, to broadcast it throughout $\mathcal{Q}_{SW}$, and begins to participate in the actual parking algorithm.

We sketch the algorithm; details appear in [14]. Table 1 specifies the local ant-moving rules that our algorithm uses. For simplicity, we word the rules as though every FSM were internal; easy modifications accommodate noninternal FSMs. Note that each rule's ant-move may create "holes" that enable further ant-moves.

The body of the process. FSMs that hold ants continually broadcast "I HAVE AN ANT" toward $\langle 0,0\rangle$; antless FSMs continually broadcast "I HAVE A HOLE" away from $\langle 0,0\rangle$. Ant-holding FSMs move their ants closer to the desired configuration *while honoring the priorities implicit in the ordering of moves in Table 1.* Thus:

At each step, Rule SW has precedence over Rule S, which, in turn has precedence over Rule W. In-diagonal rules (NW and SE) destabilize configurations in which no diagonal-lowering move applies.

Termination. Simultaneously with parking: FSMs continually check if their diagonals are full of ants or of holes. When some Δ_k is found to be full of ants (resp., holes), that fact is broadcast to Δ_{k+1} (resp., Δ_{k-1}). Parking terminates when at most one diagonal contains both ants and holes, and no diagonal with holes is closer to $\langle 0,0\rangle$ than any diagonal with ants.

Theorem 1 ([14]). **(a)** *When the algorithm terminates, ants reside in a configuration that minimizes* $\Pi(t)$. **(b)** *Each ant in $\mathcal{Q}_{SW}$ reaches its final parking position in $O(n^2)$ FSM cycles.*

We conjecture that only $O(n)$ steps are required.

Perspective: *"Intelligent" ants (that are FSMs) on an "unintelligent" earth cannot park at all —unless they can "count."*

5. Having Ants Find Food Quickly

In this section, $r \in [1, n^2]$ cells of $\mathcal{M}_n$ each contain a single ant and $s \in [1, n^2]$ cells each contain a single goal-object that we call "food;" a cell can contain both an ant and a food item. Our goal is to ensure that, if $r \le s$, then each ant gets a food item, while if $r \ge s$, then every food item is taken by some ant. Our algorithms work—but may take longer—in arrays whose nonempty cells can contain an ant and/or a food item, or an obstacle—possibly a failed FSM—that preclude both food and ants. Simple modifications of our algorithms allow one to handle *multiple kinds of food*, even *endowed with different priorities*. We present two food-finding algorithms, one of which will be preferred depending on the relative sizes of r and s.

Theorem 2. *Say that there are r ants and s food items on $\mathcal{M}_n$. There is a Cellular-ANTomata algorithm under which each ant gets food when $r \ge s$, and each food item gets taken by an ant when $r \le s$, that operates within $n \cdot \min\{(r+1),\ 4\sqrt{s}\}$ steps.*

Sketch. We trade constant factors for simpler exposition.

$\mathcal{F}_{0,0}$ determines via a synchronization if $\min(r, s) > 0$. If so, it initiates one of the following food-finding procedures, under orders from the outside world. In both algorithms, an FSM that has both food and an ant relays messages but does not initiate them; it moves its ant only when it has its ant switch roles with another ant that wants to "pass through" its cell.

A food-initiated algorithm that operates within $rn + O(n)$ steps. We call the following algorithm *food-initiated*, because it has ants sit passively awaiting messages from the food-possessing FSMs.

Rule SW. A *southwesterly* ant-move: $\mathcal{F}_{i,j}$ sends its ant along arc SW to $\mathcal{F}_{i-1,j-1}$.
Trigger. $\mathcal{F}_{i,j}$ has an ant; $\mathcal{F}_{i-1,j-1}$ has no ant.
Effect. Rule SW decreases the parking potential Π by 2.

↑↑↑↑↑ Rule SW is only for efficiency; it is not needed for correctness. ↑↑↑↑↑

Rule S. A *southerly* ant-move: $\mathcal{F}_{i,j}$ sends its ant along arc S to $\mathcal{F}_{i-1,j}$.
Trigger. $\mathcal{F}_{i,j}$, $\mathcal{F}_{i-1,j-1}$ have ants; $\mathcal{F}_{i-1,j}$ has no ant.
Effect. Rule S decreases Π by 1.

Rule W. A *westerly* ant-move: $\mathcal{F}_{i,j}$ sends its ant along arc W to $\mathcal{F}_{i,j-1}$.
Trigger. $\mathcal{F}_{i,j}$, $\mathcal{F}_{i-1,j-1}$, $\mathcal{F}_{i-1,j}$ have ants; $\mathcal{F}_{i,j-1}$ has no ant.
Effect. Rule W decreases Π by 1.

Rule NW. A *northwesterly* ant-move: $\mathcal{F}_{i,j}$ sends its ant along arc NW to $\mathcal{F}_{i+1,j-1}$.
Trigger. $\mathcal{F}_{i,j}$, $\mathcal{F}_{i-1,j-1}$, $\mathcal{F}_{i,j-1}$, $\mathcal{F}_{i-1,j}$ have ants; $\mathcal{F}_{i+1,j-1}$ has no ant.
Effect. Rule NW does not change Π; it helps search for holes in lower diagonals.

Rule SE. A *southeasterly* ant-move: $\mathcal{F}_{i,j}$ sends its ant along arc SE to $\mathcal{F}_{i-1,j+1}$.
Trigger. $\mathcal{F}_{i,j}$, $\mathcal{F}_{i-1,j-1}$, $\mathcal{F}_{i,j-1}$, $\mathcal{F}_{i-1,j}$, $\mathcal{F}_{i+1,j-1}$ have ants; $\mathcal{F}_{i-1,j+1}$ has no ant.
Effect. Rule SE does not change Π; it helps search for holes in lower diagonals.

Table 1. The local ant-moves for parking. (Application order and limits on clock skew prevent collisions.)

$\mathcal{F}_{i,j}$ sends message ...	to FSM ...
I HAVE FOOD_S	$\mathcal{F}_{i+1,j}$
I HAVE FOOD_SW	$\mathcal{F}_{i+1,j+1}$
I HAVE FOOD_W	$\mathcal{F}_{i,j+1}$
I HAVE FOOD_NW	$\mathcal{F}_{i-1,j+1}$
I HAVE FOOD_N	$\mathcal{F}_{i-1,j}$
I HAVE FOOD_NE	$\mathcal{F}_{i-1,j-1}$
I HAVE FOOD_E	$\mathcal{F}_{i,j-1}$
I HAVE FOOD_SE	$\mathcal{F}_{i,j-1}$

Table 2. Food-announcing messages.

Once $\mathcal{F}_{0,0}$ initiates the food-finding process: Every FSM that possesses food but no ant broadcasts the fact that it has food, with an indication of the direction of the food. In response, every ant-possessing FSM send their ants in the direction specified by an arriving food-announcing message. In more detail, at every step a food-possessing FSM $\mathcal{F}_{i,j}$ sends out the messages in Table 2 to its eight neighbors. The directional part of the message indicates the next step that a food-seeking ant should take in order to access the food. All FSMs relay food-announcing messages, with appropriate changes in direction.

A FSM that has an ant but no food sends analogous "I NEED FOOD" messages. It does not move their ants until it receives a food-announcing message. At that point, it sends its ant in the direction specified in the message.

A food-possessing FSM that is visited by an ant ceases to transmit its own food-announcing message, but it continues to relay those that it receives.

$\mathcal{F}_{0,0}$ broadcasts a procedure-terminating message when it ceases to receive food-announcing messages.

Correctness of the food-initiated algorithm. A message announcing a particular food item continues to be broadcast throughout $\mathcal{M}_n$ until an ant reaches the item. Hence, if $s \geq r$, then every ant will eventually follow a message that ends in its getting food, since its competition diminishes with every match between some ant and some food item. Conversely, foodless ants keep moving in response to food-announcing messages that reach the FSM they reside on. Hence, if $r \geq s$, then some ant will eventually reach every food item.

Termination is guaranteed because: $\mathcal{F}_{0,0}$ initiates the food-finding process, at which point: every FSM containing food but no ant, or an ant but no food, broadcasts that fact at every step. Once $\mathcal{F}_{0,0}$ stops hearing one or both of these messages, it knows that one or both of the unmatched food or ant supplies has been exhausted.

Timing of the food-initiated algorithm. Both food-announcing messages and food-pursuing trajectories follow shortest paths. Hence, some ant will reach some food item within $2n$ steps: n for some food-announcing message to spread throughout $\mathcal{M}_n$, and n for some ant to follow that message. Even if all ants pursue the same food item initially, once the first ant reaches that food, we are in the initial situation, *except that:* (1) we now have $r - 1$ ants and $s - 1$ food items; (2) $\mathcal{M}_n$ is already permeated with food-announcing messages. It follows that all ant-food matching will have occurred within $(r + 1)n$ steps. An additional $O(n)$ steps will suffice for $\mathcal{F}_{0,0}$ to terminate the process.

An active-ant algorithm that operates within $O(n\sqrt{s})$ steps. We call this an *active-ant* algorithm because it has ants spontaneously search for food-announcing messages, rather than passively await them.

Once $\mathcal{F}_{0,0}$ initiates the food-finding process:

Every FSM that possesses food but no ant broadcasts *along its NEWS arcs* the fact that it has food; these messages are repeated at every step (until an ant arrives) and are relayed by all FSMs. Simultaneously, every FSM that possesses an ant but no food dispatches its ants to the perimeter of $\mathcal{Q}_{SW}$. (FSMs on the left and bottom know that they are on the perimeter by the absence of certain neighbors; FSMs on the right and top know it by comparing their quadrant assignment to their neighbors'.) As an ant A reaches the perimeter—which may require it to wait for other ants to pass by—it begins to traverse the perimeter in a (for definiteness) clockwise sense. When A senses a food-announcing message, it follows that message to the food.

- If A reaches food, then it stays there unless/until forced to move away (see the next item).
- If A encounters a food item with an ant, then

  —if there is no other food item on the other side of the current one, then A returns to the perimeter and resumes its traversal;

  —if there is another food item on the other side of the current one, then A switches roles with the ant currently on this food item. A stay with this item unless/until it is forced to move away (by another ant), and the current ant pursues the more distant food item.

Correctness of the active-ant algorithm follows by the same reasoning as the food-initiated algorithm.

Timing of the active-ant algorithm. No matter how the s food items are distributed throughout $\mathcal{Q}_{SW}$, $\Omega(\sqrt{s})$ food-announcing messages will be sent to some edge of the quadrant. Thus, as long as there are enough food-seeking ants traversing the perimeter, $\Omega(\sqrt{s})$ food items are taken on the ants' first circuit of $\mathcal{Q}_{SW}$, $\Omega(\sqrt{s-\sqrt{s}})$ on the second circuit, and so on. If $r \geq s$, then, all food items will have been taken within $O(\sqrt{s})$ circuits, whence the claimed time bound. $\square$

By having messages skirt any obstacles that may exist in $\mathcal{Q}_{SW}$ (which could be failed FSMs): *both of the algorithms of Theorem 2 work when some nonempty cells contain food and/or an ant, while others contain (portions of) obstacles;* see Fig. 4. Of course, unfortunately placed obstacles can force ants to traverse longer distances, so the timing guarantees of the Theorem need not hold in obstacle-laden arrays.

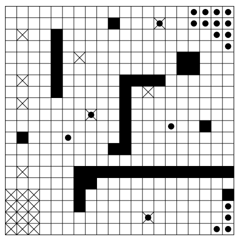

Figure 4. A mesh populated with ants (black dots), food (X-ed cells) and obstacles (blackened cells). Three ants have found food.

> **Perspective**: *Even a single "intelligent" ant (that is a FSM) requires, in the worst case, $\Omega(n^2)$ steps to find a single food item.*

6. Having an Ant Thread a Maze

Consider $\mathcal{M}_n$ with one ant-containing cell along some edge (the *entrance*) and one empty cell along some edge (the *exit*); all other edge cells are either empty or contain obstacles that block the ant. Interior cells of $\mathcal{M}_n$ are either empty or contain an obstacle—but there is at least one path of empty cells connecting the entrance to the exit. The challenge is to have the ant find the exit efficiently; this is essentially the problem of having an ant thread a maze—and that is what we shall call the problem. The following solution was developed by my seminar on Cellular ANTomata at Univ. Massachusetts.

Theorem 3. *There is a Cellular-ANTomata algorithm under which an ant can thread a maze in $\mathcal{M}_n$ in a number of steps that is proportional to the length of the shortest path between the entrance and the exit.*

Sketch. Most of the ideas are similar to the food-initiated algorithm of Section 5. $\mathcal{F}_{0,0}$ activates the FSM, $\mathcal{F}$, at the exit cell. $\mathcal{F}$ then broadcasts its location via messages of the form, "EXIT DIRECTION $= N$" (which $\mathcal{F}$ sends to its southern neighbor, if it exists). Each FSM in an empty cell relays *the first such that it receives* (with the appropriate direction); ties can be broken arbitrarily. When the ant-possessing FSM at the entrance receives the message for the first time, it dispatches the ant along the path of relayed messages to the exit. Because of the relaying regimen, this is a shortest path. $\square$

Our algorithm clearly works in multi-exit mazes also.

It is not hard to construct mazes that require any deterministic intelligent ant to explore $\Omega(n^2)$ cells of $\mathcal{M}_n$, even though there is an entrance-exit path of length $O(n)$. Fig. 5 depicts a family of mazes that stymie any deterministic “intelligent”-ant algorithm. In the figure: the circle depicts the entrance to the maze, and blackened cells contain impenetrable obstacles. X-ed cells represent potential exits: in each instance of this maze, precisely one of these cells is the exit; all others contain obstacles. A deterministic “intelligent” ant (that is a FSM) must thread the maze in search of the actual exit.

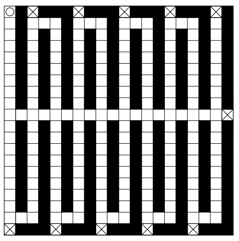

Figure 5. A family of mazes that can require a deterministic “intelligent” ant to follow a lot of useless paths. (Precisely one X-ed cell is the exit.)

Perspective: *A deterministic “intelligent” ant (that is a FSM) requires, in the worst case, $\Omega(n^2)$ steps to thread a single-exit maze.*

7. Conclusion

We have significantly extended the initial work of [14] on a novel paradigm for implementing ant-inspired robotic algorithms in geographically constrained environments, Cellular ANTomata. The hallmark of our paradigm is an inversion of the site of “intelligence,” between the ant-robots and the surface that they traverse. We have added two simple, yet relevant, proof-of-concept problems to the one studied in [14]: food-gathering (Section 5) and maze threading (Section 6). As in [14], we found that algorithms based on Cellular ANTomata significantly outperform traditional approaches. We have just scratched the surface of this challenging and inviting research topic.

References

[1] A.W. Burks (Ed.) (1970): *Essays on Cellular Automata*. Univ. Illinois Press, Urbana-Champaign, IL.

[2] L. Chen, X. Xu, Y. Chen, P. He (2004): A novel ant clustering algorithm based on cellular automata. *IEEE/WIC/ACM Intl. Conf. Intelligent Agent Technology*.

[3] D. Chowdhury, V. Guttal, K. Nishinari, A. Schadschneider (2002): A cellular-automata model of flow in ant trails: non-monotonic variation of speed with density. *J. Phys. A: Math. Gen. 35*, L573–L577.

[4] S.N. Cole (1966): Real-time computation by n-dimensional iterative arrays of finite-state machines. *7th IEEE Symp. on Foundations of Computer Science*, 53–77.

[5] G. Folino, G. Mendicino, A. Senatore, G. Spezzano, S. Straface (2006): A model based on cellular automata for the parallel simulation of 3D unsaturated flow. *Parallel Computing 32*, 357–376.

[6] D. Geer (2005): Small robots team up to tackle large tasks. *IEEE Distributed Systems Online*, vol. 6, no. 12.

[7] E. Goles and S. Martinez (Eds.) (1999): *Cellular Automata and Complex Systems*. Kluwer, Amsterdam.

[8] J. Gruska, S. La Torre, M. Parente (2004): Optimal time and communication solutions of firing squad synchronization problems on square arrays, toruses and rings. In *Developments in Language Theory* (C.S Calude, E. Calude, M.J. Dinneen, Eds.) *Lecture Notes in Computer Science 3340*, Springer-Verlag, Berlin, 200–211.

[9] E.D. Innis (2006): *Tessellated Finite Automata for the Control of Autonomous Agents*. Senior Honors Thesis, Univ. Massachusetts.

[10] http://www.kivasystems.com/

[11] F. Marchese (1996): Cellular automata in robot path planning. *EUROBOT’96*, 116–125.

[12] E.F. Moore (1962): The firing squad synchronization prolem. In *Sequential Machines, Selected Papers* (E.F. Moore, Ed.), Addison-Wesley, Reading, MA, pp. 213–214.

[13] M.O. Rabin and D. Scott (1959): Finite automata and their decision problems. *IBM J. Res. Develop. 3*, 114–125.

[14] A.L. Rosenberg (2007): Cellular ANTomata. *5th Intl. Symp. on Parallel and Distributed Processing and Applications*. In *Lecture Notes in Computer Science 4742*, Springer-Verlag, New York (2007) 78–90.

[15] J.C. Shepherdson (1959): The reduction of two-way automata to one-way automata. *IBM J. Res. Develop. 3*, 198–200.

[16] G. Spezzano and D. Talia (1998): The CARPET programming environment for solving scientific problems on parallel computers. *Parallel and Distributed Computing Practices 1*, 49–61.

[17] J. von Neumann (1966): *The Theory of Self-reproducing Automata*. (Edited and completed by A.W. Burks) Univ. of Illinois Press, Urbana-Champaign, IL.

[18] S. Wolfram (Ed.) (1986): *Theory and Application of Cellular Automata*. Addison-Wesley, Reading, MA.

[19] S. Wolfram (1994): *Cellular Automata and Complexity: Collected Papers*. Addison-Wesley, Reading, MA.