Cellular ANTomata
(Extended Abstract)

Arnold L. Rosenberg

Dept. of Computer Science, Univ. of Massachusetts, Amherst, MA 01003, USA
rsnbrg@cs.umass.edu

Abstract. Cellular automata can form the basis of a practical model for a broad range of tasks that require the coordination of many simple computing devices. We propose using "semi-synchronous" cellular automata as a platform for efficiently realizing ant-inspired algorithms that coordinate robots within a fixed, geographically constrained environment. We present an appropriate formalization of the resulting *Cellular ANTomaton* model, illustrated via "proof-of-concept" problems that have ant-robots move and aggregate in various ways.

Keywords: Ant-inspired Robotics; Cellular automata; Mini-factories.

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. [3,4,6]) 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 [2,13]—that we name *Cellular ANTomata*—as a platform for developing the algorithmics of robotic mobile computers within constrained geographic environments. We specify 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

I. Stojmenovic et al. (Eds.): ISPA 2007, LNCS 4742, pp. 78–90, 2007.
© Springer-Verlag Berlin Heidelberg 2007

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 constrained environments.

1.2 Ant-Robots in a Factory

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 collisions with obstacles and one another;
- communicate with and sense one another, by "direct contact" (as when real ants meet) and by "timestamped message passing" (as when real ants deposit volatile pheromones);
- discover goal objects (*"food"*) and convey food from one location to another;
- assemble in desired locations, in desired physical 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 "timestamped" communication. Indeed, researchers (cf. [6]) have equipped robots with small transceivers that function 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, say, manufacturing aids in a geographically constrained environment such as a factory floor, two features of this approach call out for emendation.

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

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. Our model revisits the relationship of ants to their environment (henceforth, the *"(factory) floor"*). In the "standard" approach, all intelligence and initiative resides in the ants; the factory 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 outdoors in unconstrained, uncontrollable environments, it is eminently avoidable on a factory floor. We propose to invert the active-passive relationship between ants and the floor, by tesselating the floor with tiles that are identical in size and shape, embedding within each tile a copy of some standard 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 (cf. [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 neighboring tiles—those it shares an edge or corner with—by receiving one message from each and transmitting one message 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." (Note that we use "King's-move" adjacencies.)

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. Moreover, ants are now much simpler, containing little electronics except as 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 Model Features

(a) Factory 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 "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 messages, rather than shared "clock ticks."

Space constraints limit us to high-level sketches of algorithms and analyses.

1.4 Related Work

Our use of *Cellular automata* (*CA*, for short) to realize ant-inspired algorithms and behavior is not original. In [4], CA underlie an ant-inspired algorithm for a genre of flow problem; in [3], CA implement an ant-inspired clustering algorithm. Most closely related to our model is the *DFMS* model of [1], an iterative array of minicomputers that can be programmed to compute source-target paths for microscopic droplets that co-reside on the array with obstacles and other special features. All of these sources depart from our goal by focusing on a single computational problem and by positing centrally programmed models, with, e.g., global name spaces for tiles/computers. Also closely related to our model, but with a specialized focus, are the many papers on the "firing squad" synchronization problem for CA, [10], especially the 2-dimensional variant [8]. Finally, "numerical pheromones" appear in, e.g., [6].

2 Cellular ANTomata

CA are a natural candidate for realizing intelligent factory floors, being composed of *finite-state machines* (*FSMs*) arranged in simply structured arrays. Studied since the 1960s [5,10,12], CA remain of interest to this day [7,8,14], providing a formal model of computers [13,14] that combines mathematical simplicity with levels of efficiency that make them feasible for many real computational tasks. Indeed, CA are remarkably efficient for a broad range of tasks that require the tight coordination of many simple agents [3,4,5,8,10]. Our variant of CA, *Cellular ANTomata*, is tailored to the algorithmics of ants on a factory floor.

2.1 Basics

As with CAs, Cellular ANTomata place a copy of a single FSM at each node of a *square mesh*. (One easily restricts our *2-dimensional* model to one dimension or extends it to three. We define our model in great detail, to facilitate implementation of our model ("hardware") and algorithms ("software").

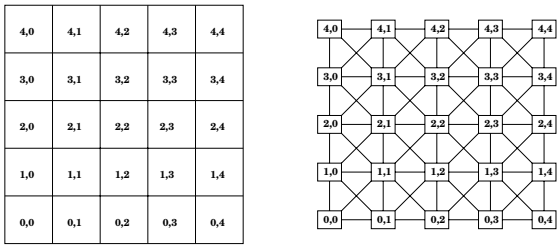

Fig. 1. A 5×5 tiled floor; its associated graph (edges represent mated opposing arcs)

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*, $\mathcal{M}_n$. The nodes of $\mathcal{M}_n$ are the tiles, formally, the set[2] $[0, n-1] \times [0, n-1]$. $\mathcal{M}_n$'s *King's-move* arcs are tiles' adjacencies, labeled by the compass directions: E, SE, S, SW, W, NW, N, NE; cf. Fig. 2(left) and Table 1. Each node $v = \langle i, j \rangle$ of $\mathcal{M}_n$ is connected by a mated in-arc and out-arc to each of its (≤ 8) neighboring nodes (see Table 2).

- If $i, j \in \{0, n-1\}$ then v is a *corner* node and has 3 neighbors.
- If $i = 0$ (resp., $i = n-1$) and $j \in [1, n-2]$, then v is a *bottom* (resp., *top*) node. If $j = 0$ (resp., $j = n-1$) and $i \in [1, n-2]$, then v is a *left* (resp., *right*) node. These four are collectively *edge* nodes; each has 5 neighbors.
- If $i, j \in [1, n-2]$, then v is an *internal* node and has 8 neighbors.

FSMs: Finite-State Machines. An FSM $\mathcal{F}$ in a cellular ANTomaton is given by:

[1] We discuss only square tiles; simple modifications allow, say, hexagonal tiles.

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

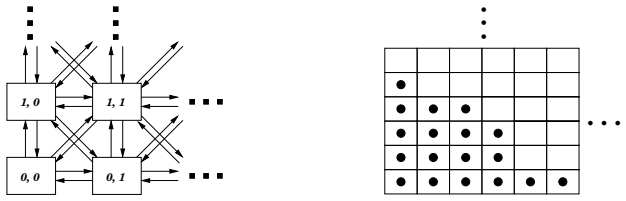

Fig. 2. (Left) The 2×2 corner of a mesh, with all incident arcs. (Right) The 6×6 "prefix" of $\mathcal{Q}_{SW}$, with 18 optimally parked ants (denoted by dots); cf. Section 3.

Table 1. The NEWS and diagonal arc-labels, and their actions

Arc label	Arc action	Leads node $\langle i, j \rangle$	to node ...
E	move to *East*	$(i \in [0, n-1],\ j \in [0, n-2])$	$\langle i, j+1 \rangle$
W	move to *West*	$(i \in [0, n-1],\ j \in [1, n-1])$	$\langle i, j-1 \rangle$
N	move to *North*	$(i \in [0, n-2],\ j \in [0, n-1])$	$\langle i+1, j \rangle$
S	move to *South*	$(i \in [1, n-1],\ j \in [0, n-1])$	$\langle i-1, j \rangle$
NE	move to *Northeast*	$(i \in [0, n-2],\ j \in [0, n-2])$	$\langle i+1, j+1 \rangle$
SW	move to *Southwest*	$(i \in [1, n-1],\ j \in [1, n-1])$	$\langle i-1, j-1 \rangle$
NW	move to *Northwest*	$(i \in [0, n-2],\ j \in [1, n-1])$	$\langle i+1, j-1 \rangle$
SE	move to *Southeast*	$(i \in [1, n-1],\ j \in [0, n-2])$	$\langle i-1, j+1 \rangle$.

- a finite set Q of *states*. $Q = K \times [0, I_1] \times [0, I_2] \times \cdots \times [0, I_\ell]$, where K is a set of "control" variables, and each $I_j \in \mathbb{N}$. This endows ants with ℓ types of pheromones, the kth of which can exist at any intensity $i \in [0, I_k]$.
- an *input "alphabet"* IN, which is the union of:
  —the set of all messages that $\mathcal{F}$ can receive from neighboring FSMs,
  —$\{0,1\}^3$: binary indicators of the presence of an ant, an obstacle, food;
- an *output "alphabet"* OUT, which is the union of:
  —the set of all messages that $\mathcal{F}$ can send to neighboring FSMs,
  —the set O: possible orders to the resident ant (if it exists).
- a state-transition function $\delta : Q \times IN \to Q \times OUT$, 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 a 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 *by an ant* ($+1$ change).

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* from $[0, n-1] \times [0, n-1]$ and places each $\mathcal{F}_{i,j}$ 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 that a resident ant will respond to.* **(c)** *connecting each FSM with a bidirectional communication channel to each of its King's-move neighbors.* Thus, the input

Table 2. The communication links of mesh nodes

Corner node	talks to node:	via out-arc:	via in-arc:
$\langle 0,0\rangle$	$\langle 1,0\rangle$	N	S
	$\langle 1,1\rangle$	NE	SW
	$\langle 0,1\rangle$	E	W
$\langle n-1,0\rangle$	$\langle n-2,0\rangle$	S	N
	$\langle n-2,1\rangle$	SE	NW
	$\langle n-1,1\rangle$	E	W
$\langle n-1,n-1\rangle$	$\langle n-2,n-1\rangle$	S	N
	$\langle n-2,n-2\rangle$	SW	NE
	$\langle n-1,n-2\rangle$	W	E
$\langle 0,n-1\rangle$	$\langle 0,n-2\rangle$	W	E
	$\langle 1,n-2\rangle$	NW	SE
	$\langle 1,n-1\rangle$	N	S

Node	talks to node:	via out-arc:	via in-arc:
$\langle i,j\rangle$	$\langle i,j-1\rangle$	W	E
	$\langle i+1,j-1\rangle$	NW	SE
	$\langle i+1,j\rangle$	N	S
	$\langle i+1,j+1\rangle$	NE	SW
	$\langle i,j+1\rangle$	E	W
	$\langle i-1,j+1\rangle$	SE	NW
	$\langle i-1,j\rangle$	S	N
	$\langle i-1,j-1\rangle$	SW	NE

Node-type	talks to node:	via out-arc:	via in-arc:
Bottom node $\langle 0,j\rangle$	$\langle 0,j-1\rangle$	W	E
	$\langle 1,j-1\rangle$	NW	SE
	$\langle 1,j\rangle$	N	S
	$\langle 1,j+1\rangle$	NE	SW
	$\langle 0,j+1\rangle$	E	W
Top node $\langle n-1,j\rangle$	$\langle n-1,j-1\rangle$	W	W
	$\langle n-2,j-1\rangle$	SW	NW
	$\langle n-2,j\rangle$	S	N
	$\langle n-2,j+1\rangle$	SE	NE
	$\langle n-1,j+1\rangle$	E	E
Left node $\langle i,0\rangle$	$\langle i-1,0\rangle$	S	N
	$\langle i-1,1\rangle$	SE	NW
	$\langle i,1\rangle$	E	W
	$\langle i+1,1\rangle$	NE	SW
	$\langle i+1,0\rangle$	N	S
Right node $\langle i,n-1\rangle$	$\langle i-1,n-1\rangle$	S	N
	$\langle i-1,n-2\rangle$	SW	NE
	$\langle i,n-2\rangle$	W	E
	$\langle i+1,n-2\rangle$	NW	SE
	$\langle i+1,n-1\rangle$	N	E

and output alphabets of each 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 by replacing inputs from/outputs to nonexistent neighbors by "NIL;" e.g., the message-set for $\mathcal{F}_{0,j}$ is: $\Sigma_N \times \Sigma_{NE} \times \Sigma_E \times \{\text{NIL}\} \times \{\text{NIL}\} \times \{\text{NIL}\} \times \Sigma_W \times \Sigma_{NW}$.

Note 1. **(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* (cf. Section 1.3), we insist that algorithms treat n as an *unknown*, never exploiting its specific value.

$\mathcal{F}_{0,0}$ is the *general* of $\mathbf{F}_n(\mathcal{F})$, the only FSM that accepts inputs (usually commands) from the "outside world"—say, for definiteness, via its western input port (so that Σ_W must contain all external commands). We expect the "outside world" to tell the general when to initiate activities.

3 "Parking" the Ants

We now have an FSM-array "park" its ants "compactly:" each ant moves as close as possible to the corner cell of the quadrant that it resides in when parking is initiated. This activity is easily achieved within our model; *it is not achievable at all within the standard setting of "smart" ants navigating a passive earth.*

3.1 The Formal Parking Problem

A *quadrant* of $\mathcal{M}_n$ is the induced subgraph[3] on the following node-set.

Quadrant	Node-set	Quadrant	Node-set
$\mathcal{Q}_{SW}$	$[0, \lceil n/2 \rceil - 1] \times [0, \lceil n/2 \rceil - 1]$	$\mathcal{Q}_{NW}$	$[\lceil n/2 \rceil, n-1] \times [0, \lceil n/2 \rceil - 1]$
$\mathcal{Q}_{SE}$	$[0, \lceil n/2 \rceil - 1] \times [\lceil n/2 \rceil, n-1]$	$\mathcal{Q}_{NE}$	$[\lceil n/2 \rceil, n-1] \times [\lceil n/2 \rceil, n-1]$

$\mathcal{M}_n$'s kth *diagonal* ($k \in [0, 2n-2]$) is: $\Delta_k = \{\langle i, j \rangle \in [0, n-1] \times [0, n-1] \mid i+j = k\}$. For $d \in [1, 2n-1]$, $\mathcal{Q}_{SW}$'s *radius-d* L_1*-quarter-sphere* is $\bigcup_{k=0}^{d-1} \Delta_k$. Other quadrants' quarter-spheres are defined analogously.

The *parking problem for* $\mathcal{Q}_{SW}$ requires the ants in $\mathcal{Q}_{SW}$ to cluster within the most compact L_1-quarter-sphere "centered" at $\langle 0, 0 \rangle$; cf. Fig. 2(right).[4] Formally, one must minimize the *parking potential function:* $\Pi(t) = \sum_{k=0}^{2n-2} (k+1) \times$ (the number of ants residing on Δ_k at step t). Thus, if parking begins with $m(m-1)/2 + p$ ants in $\mathcal{Q}_{SW}$, where $p \leq m$, then for all sufficiently large t, $\Pi(t) = \sum_{i=1}^{m-1} i^2 + mp = m(m-1)(2m-1)/6 + mp$.

3.2 Initiating Parking

We begin to park ants by partitioning the parking problem into independent quadrant-specific subproblems (that are solved in parallel). A central subproblem is to let ants know which quadrant they reside in.

3.2.1 Sketch: Algorithm Activate_Center_Cells

$\mathcal{F}_{0,0}$ activates the center FSM(s) via two messages. A northward message (eventually[5]) enlists $\mathcal{F}_{n-1,0}$ in the activation. A northeasterly message "semi-activates"

[3] The *induced subgraph* of $\mathcal{G} = (N, A)$ on $N' \subseteq N$ has all arcs from A both of whose endpoints are in N'.

[4] One easily adapts the problem definition and algorithm to $\mathcal{M}_n$'s other quadrants.

[5] Since messages proceed from neighbor to neighbor, this process takes $\Theta(n)$ steps.

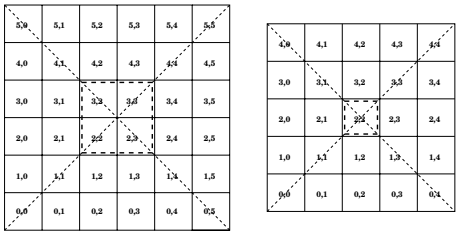

Fig. 3. Illustrating the "centers" of even- and odd-sided meshes

all FSMs along the SW-NE diagonal. Once activated, $\mathcal{F}_{n-1,0}$ sends a southeasterly message, which "semi-activates" all FSMs along the NW-SE diagonal. For even n, activation is achieved when both $\mathcal{F}_{n/2-1,n/2-1}$ and $\mathcal{F}_{n/2,n/2}$ become "semi-activated" along the SW-NE diagonal and both $\mathcal{F}_{n/2-1,n/2}$ and $\mathcal{F}_{n/2,n/2-1}$ become "semi-activated" along the NW-SE diagonal. A center cell identifies its quadrant from the directions of its fellow center cells. For odd n, activation is achieved when $\mathcal{F}_{\lceil n/2 \rceil,\lceil n/2 \rceil}$ becomes "semi-activated" along both the SW-NE and NW-SE diagonals. See Fig. 3.

3.2.2 Sketch: Algorithm Initiate_Parking_SW

Once activated, $\mathcal{M}_n$'s center cell(s) each broadcast a message into its quadrant that tells ant-holding FSMs to initiate the parking process appropriate to that quadrant. (When n is odd, the unique center cell broadcasts distinct messages into each quadrant.)

3.3 Parking Within $\mathcal{Q}_{SW}$

Table 3 specifies the permissible local ant-moving rules used in our parking algorithm. For brevity, we word rules as though every $\mathcal{F}_{i,j}$ is internal; trivial modifications accommodate extremal FSMs. Note that each ant-move may create "holes" that enable further ant-moves.

3.3.1 Sketch: Algorithm Park_in_$\mathcal{Q}_{SW}$

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 3.* 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 a diagonal 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. Fig. 4 illustrates Algorithm Park_in_$\mathcal{Q}_{SW}$ (without the termination phase).

Table 3. The repertoire of local ant-moves, in decreasing order of priority

Rule *SW*. A southwesterly ant-move: $\mathcal{F}_{i,j}$ sends its ant to $\mathcal{F}_{i-1,j-1}$ (along arc SW).
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 not needed for correctness, but it enhances efficiency ↑↑↑
Rule *S*. A southerly ant-move: $\mathcal{F}_{i,j}$ sends its ant to $\mathcal{F}_{i-1,j}$ (along arc S).
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 to $\mathcal{F}_{i,j-1}$ (along arc W).
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 to $\mathcal{F}_{i+1,j-1}$ (along arc NW).
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 to $\mathcal{F}_{i-1,j+1}$ (along arc SE).
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.

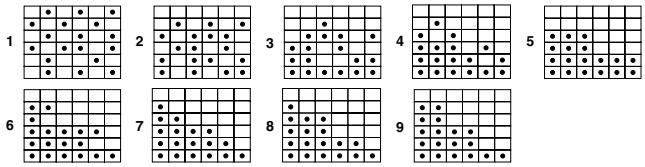

Fig. 4. Nine steps of our parking algorithm

Note 2. The "systolic" strategy of moving ants in phases by direction facilitates the orchestration of ants for many computational problems.

3.3.2 Algorithm Verification and Analysis

Theorem 1. (a) *When Algorithm* Park_in_$\mathcal{Q}_{SW}$ *terminates, all ants reside in a configuration that minimizes* $\Pi(t)$. **(b)** *Each ant in* $\mathcal{Q}_{SW}$ *when the parking process is initiated reaches its final parking position in* $O(n^2)$ *FSM cycles.*

Proof (Sketch). Because Algorithm Park_in_$\mathcal{Q}_{SW}$ executes all rules continually: **(a)** Our rules preclude each way that ants can fail to end up properly parked. Specifically: • *Ants end up "left justified" in each row and "bottom justified" in each column.* • *If row* i *(resp., column* j*) contains* c *ants, and row* $i+1$ *(resp., column* $j+1$*) contains* d *ants, then* $c-2 \leq d \leq c$. • *At most one diagonal contains both an ant and a hole.* Thus, the ants end up in a potential-minimizing configuration, then halt. **(b)** The (very conservative) bound on timing follows because each application of Rule SW (resp., S or W) decreases $\Pi(t)$ by 2 (resp., by 1). Some ant follows one of these rules at least every n steps, since ants that cannot follow either rule wander along their current diagonals searching for a hole in the next lower diagonal. Since $\mathcal{Q}_{SW}$ has $\approx n/2$ diagonals, the bound would follow even if only one ant were to move per cycle (which is very unlikely).

4 Having Ants Find Food Quickly

We consider now a problem in which $r \geq 1$ cells of $\mathcal{M}_n$ each contain a single ant and $s \geq 1$ cells each contain a single food item; a cell can contain both an ant and food. Our goal is to ensure that: if $r \leq s$, then each ant gets food; if $r > s$, then every food item is taken by some ant. We consider two variants of this problem. In Section 4.1, each cell of $\mathcal{M}_n$ can contain an ant or a food item; in Section 4.2, a cell can contain an obstacle that precludes the presence of both food and ants, thereby inhibiting the movement of ants (*but not of messages!*). It is simple to allow many kinds of food, even endowed with different priorities.

4.1 Finding Food with No Obstacles

Our algorithm proceeds greedily.

4.1.1 Sketch: Algorithm Find_Food

Food finding begins with $\mathcal{F}_{0,0}$ broadcasting an INITIATE message. Let $\mathcal{F}_{i,j}$ be a generic FSM in $\mathbf{F}_n(\mathcal{F})$. **(a)** If $\mathcal{F}_{i,j}$ contains food, then it broadcasts a food-announcement, with an indication of the direction toward it; e.g., it sends the message "I HAVE FOOD_S" to its northen neighbor, $\mathcal{F}_{i+1,j}$. **(b)** $\mathcal{F}_{i,j}$ relays each food-announcement that it receives into the quadrant "opposite" to the one the message came from, as illustrated in Fig. 5. **(c)** If $\mathcal{F}_{i,j}$ possesses both food and

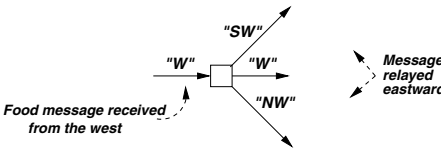

Fig. 5. The regimen for relaying food-announcing messages, for direction W

an ant, then it matches the two and does not announce that food item (though it continues to relay announcements about other food). **(d)** If $\mathcal{F}_{i,j}$ possesses an ant but no food, then it sends "I HAVE AN UNMATCHED ANT" in the direction of $\mathcal{F}_{0,0}$. If it receives any food-announcements, then it selects one (say, for definiteness, in clockwise priority from the north) and moves its ant in the direction of that food. It continues to relay that food-announcement in case it combines multiple collinear messages. **(e)** Food finding is terminated by $\mathcal{F}_{0,0}$ when it ceases to receive either "I HAVE FOOD" or "I HAVE AN UNMATCHED ANT" messages.

4.1.2 Algorithm Verification and Analysis

Theorem 2. *Say that there are r ants and s food items on $\mathcal{M}_n$ when Algorithm* Find_Food *is initiated. If $r \geq s$, then some ant will reach every food item; if $s \geq r$, then every ant will get food.*

Proof (Sketch). A food-announcement is broadcast until an ant reaches the food. Hence, if $s \geq r$, every ant will eventually get food, since competition diminishes

with every food-ant match. Conversely, foodless ants keep moving in response to food-announcements; hence, if $r \geq s$, some ant will eventually reach every food item. Termination is guaranteed by the two "unmatched-item" messages. When $\mathcal{F}_{0,0}$ stops hearing either message, it knows that either food or ants have run out. "Eventual" is hard to quantify, but the process is "locally efficient:" each step by a food-announcing message and a food-pursuing ant follows a shortest path to/from food.

An enhancement. By endowing each food-announcing message with a pheromone whose level decreases with each relay, one can have ants pursue the closest food item—at least to within the resolution of the pheromone-level "counter."

4.2 Finding Food with Obstacles

We now allow some nonempty cells of $\mathcal{M}_n$ contain food and/or an ant (as before) while others contain (portions of) obstacles; see Fig. 6. Our goal is to adapt Algorithm Find_Food to accommodate obstacles that inhibit the passage of ants—but, recall, *not* of messages.

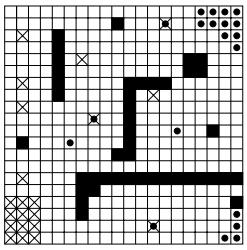

Fig. 6. A mesh some of whose cells contain ants (dots), food (X-es) and obstacles (filled cells); three ants have food

4.2.1 Sketch: Algorithm Find_Food-Avoid_Obstacles

Each FSM $\mathcal{F}$ functions as in Algorithm Find_Food, with the following exceptions. **(a)** If $\mathcal{F}$ contains an obstacle, then it informs its neighbors of that fact. **(b)** If $\mathcal{F}$ contains an ant A *and* receives a food-announcement from an obstacle-containing neighbor, then $\mathcal{F}$ sends A on a clockwise circuit of the obstacle. A proceeds until it either finds a food trail to follow or receives the termination signal. If A's circuit is interrupted by: *(i)* an edge of $\mathcal{M}_n$ for the first time, then A reverses direction; *(ii)* an edge of $\mathcal{M}_n$ for the second time, then A halts and announces "FOOD INACCESSIBLE;" *(iii)* an ant B that has food or is proceeding clockwise around another obstacle, then A switches roles with B; *(iv)* an ant that is proceeding counterclockwise around this obstacle, then A reverses direction.

4.2.2 Algorithm Verification and Analysis

Theorem 3. *Say that there are r ants and s food items that are mutually accessible on $\mathcal{M}_n$ when Algorithm* Find_Food-Avoid_Obstacles *is initiated. If $r \geq s$,*

then one of these ants will reach every food item; if $s \geq r$, then every one of these ants will get food.

Proof (Sketch). One of the following happens when ant A attempts a clockwise circuit of an obstacle. (**1**) *A finds a food trail that is unblocked by an obstacle.* A follows that trail away from the current obstacle. (Of course, competition may cause it to return later.) (**2**) *A encounters an edge of $\mathcal{M}_n$.* Reversing direction allows A to continue pursuing food, unless all available food is cut off by the obstacle—which causes A to encounter a second edge. (**3**) *A encounters another ant, B, that blocks its path.* If B is following a food trail, then A would recognize that trail also—so we know that B is either sitting on food or skirting an obstacle also. If B's obstacle is distinct from A's, then B, too, is proceeding in a clockwise sense; in this case and when B is sitting on food, having A and B switch roles avoids an impasse. If B is skirting the same obstacle as A, then B has encountered an edge of $\mathcal{M}_n$, so A reverses direction immediately, thereby avoiding an impasse. (**4**) *A keeps circling the obstacle, never encountering an edge of $\mathcal{M}_n$, a food trail, or another ant.* A's journey then ends eventually because of Algorithm Find_Food's termination signal. This occurs, e.g., if other ants reach food item(s), so that all trails are terminated. In all cases, A eventually either is free to pursue food or is told to terminate its search because no unmatched food is accessible.

5 Conclusion

Progress. Our goal has been to motivate and illustrate a conceptual/algorithmic framework that inverts the "natural" relationship between mobile robots and the environment that they navigate. Our adapted "semi-synchronous" cellular automata create a world in which the "factory floor" contains the intelligence, while the identical "ants" are simple devices that respond to commands from the "floor." We have formalized *Cellular ANTomata* and presented two "proof-of-concept" problems that illustrate different aspects of the model's capabilities.
Plans. Ongoing work focuses on extended algorithmics—e.g., having ants move obstacles, assemble food to specified locations, thread mazes, deal with faults—and on high-level ant-orchestration via, e.g., the "systolic" phasing in Algorithm Park_in_$\mathcal{Q}_{SW}$ and other mechanisms inspired by sources such as [11].

Thanks to: O. Brock, R. Grupen, H. Lee for helpful comments and pointers.

References

1. Böhringer, K.F.: Modeling and controlling parallel tasks in droplet-based microfluidic systems. IEEE Trans. Computer-Aided Design of Integrated Ccts. and Systs. 25, 329–339 (2006)
2. Burks, A.W. (ed.): Essays on Cellular Automata. Univ. Illinois Press, Urbana-Champaign, IL (1970)

3. Chen, L., Xu, X., Chen, Y., He, P.: A novel ant clustering algorithm based on cellular automata. In: IEEE/WIC/ACM Intl. Conf. Intelligent Agent Technology (2004)
4. Chowdhury, D., Guttal, V., Nishinari, K., Schadschneider, A.: A cellular-automata model of flow in ant trails: non-monotonic variation of speed with density. J. Phys. A: Math. Gen. 35, L573–L577 (2002)
5. Cole, S.N.: Real-time computation by n-dimensional iterative arrays of finite-state machines. In: 7th IEEE Symp. on Foundations of Computer Science, pp. 53–77 (1966)
6. Geer, D.: Small robots team up to tackle large tasks. IEEE Distr. Systs. Online 6(12) (2005)
7. Goles, E., Martinez, S. (eds.): Cellular Automata and Complex Systems. Kluwer, Amsterdam (1999)
8. Gruska, J., La Torre, S., Parente, M.: Optimal time and communication solutions of firing squad synchronization problems on square arrays, toruses and rings. In: Calude, C.S., Calude, E., Dinneen, M.J. (eds.) DLT 2004. LNCS, vol. 3340, pp. 200–211. Springer, Heidelberg (2004)
9. `http://www.kivasystems.com/`
10. Moore, E.F. (ed.): The firing squad synchronization problem. Sequential Machines, Selected Papers, pp. 213–214. Addison-Wesley, Reading, MA (1962)
11. Spezzano, G., Talia, D.: The CARPET programming environment for solving scientific problems on parallel computers. Parallel and Distributed Computing Practices 1, 49–61 (1998)
12. von Neumann, J.: The Theory of Self-reproducing Automata. In: Burks, A.W. (ed.) Univ. of Illinois Press, Urbana-Champaign, IL (1966)
13. Wolfram, S. (ed.): Theory and Application of Cellular Automata. Addison-Wesley, Reading, MA (1986)
14. Wolfram, S.: Cellular Automata and Complexity: Collected Papers. Addison-Wesley, Reading, MA (1994)