Ants in Parking Lots

Arnold L. Rosenberg
Electrical & Computer Engineering
Colorado State University
Fort Collins, CO 80523, USA
rsnbrg@colostate.edu

Abstract

Ants provide an attractive metaphor for robots that "cooperate" in performing complex tasks. What, however, are the algorithmic consequences of following this metaphor? This paper is a step toward understanding the algorithmic strengths and weaknesses of ant-based computation models. We study the ability of ant-robots that are essentially mobile finite-state machines to perform a simple path-planning task called *parking*, within fixed, geographically constrained environments ("laboratory floors"). This task: (1) has each ant head for the nearest corner of the floor and (2) has all ants within a corner organize into a maximally compact formation. Even without using (digital analogues of) pheromones, many initial configurations of ants *can* park. These configurations include: (*a*) a single ant that starts anywhere along an edge of the floor and (*b*) any assemblage of ants that begins with at least two ants adjacent to one another. In contrast, a single ant on a *one-dimensional* "floor" *cannot* park, even with the help of (volatile digital) pheromones.

1 Introduction

As we encounter novel computing environments that offer unprecedented computing power, while posing unprecedented challenges, it is compelling 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, mainly ants (because robots typically operate within a two-dimensional world), when contemplating how to employ the computers effectively and efficiently in a variety of geographical environments; indeed, many sources—see, e.g., [1, 3, 4, 5, 7, 8]—have done precisely that. This paper is a step toward understanding the algorithmic consequences of the robot-as-ant metaphor within the context of a simple, yet significant path-planning problem.

Ant-inspired robots in a "laboratory." We focus on mobile robotic computers (henceforth, *ants*, or *ant-robots*, to stress the natural inspiration) that function within a fixed geographically constrained environment (henceforth, a *laboratory floor*, suggesting a possible application domain) that is tesselated with identical (say, square) tiles. We expect ants to be able to:

- navigate the floor, while avoiding collisions (with obstacles and one another);

- communicate with and sense one another, by "direct contact" (as when natural ants meet) and by "timestamped message passing" (as when natural ants deposit volatile pheromones);
- assemble in desired locations, in desired configurations.

Although it is not relevant to the current study, one would also expect ants to discover goal objects ("food") and convey "food" from one location to another; cf. [3, 4, 7, 8].

In the "standard realization" of ant-robots—which is what we study here—robots are endowed with "intelligence," in the form of embedded computers. The resulting "intelligent" ant-robots, are responsible for planning and orchestrating assigned activities. In the hope of enhancing the economic feasibility of our model by having ants require few computational resources, we focus on ants that function essentially as mobile finite-state machines.

Having ants park on the floor. This paper focuses on a simple path-planning task that we studied in [7] under a rather different ant-based computational model ("Cellular ANTomata"). This task, called *parking:* (1) has each ant head for the corner of the floor that it is nearest to when told to park, and (2) has all ants within a corner organize into a maximally compact formation (Section 2.2 supplies formal details). While we have not yet determined definitively which initial configurations of ants on the floor can park successfully and which cannot, we report here on progress toward this goal:

- Even without using (digital analogues of) pheromones, *many initial configurations of ants* can *park*. These configurations include:
  - *a single ant that starts anywhere along an edge of the floor* (Theorem 3.2);
  - *any assemblage of ants that begins with at least two ants adjacent to one another*—i.e., on tiles that share an edge or a corner (Theorem 4.1).
- In contrast: *A single ant on a* one-dimensional *"floor"* cannot *park, even with the help of (volatile digital) pheromones* (Theorem 3.1).

The algorithmic setting. We require algorithms to be *simple*, *scalable* and *decentralized*.
(1) *Algorithms must work on laboratory floors of arbitrary size.* In particular, no algorithm can exploit any information about the size of the floor. Thus, algorithms must treat the side-length n of the floor as an *unknown*, never exploiting its specific value.
(2) *Algorithms must work with arbitrarily large collections of ants.* This means, in particular, that all ants are identical; no ant has a "name" that renders it unique.
(3) *Algorithms must employ noncentralized coordination.* All coordination among ants is achieved in a *distributed, noncentralized* manner, via messages that pass between ants on neighboring tiles.
(4) *Algorithms must be "finite-state."* All ants must execute the same program (in SPMD mode), and this program must have the very restricted form described in Section 2.1.2.

These guidelines are usually violated in practical implementations (cf. [2, 3, 4]), for reasons involving, say, cost and simplicity.

2 Technical Background

2.1 Ant-Robots Formalized

The "heart" of the model that underlies our study is the floor that ants operate on and the computers that endow ants with "intelligence."

2.1.1 Pheromone-bearing laboratory floors

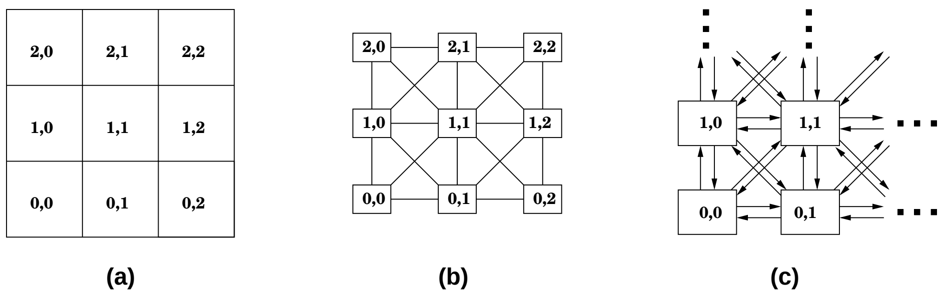

Figure 1: (a) *A* 3×3 *floor* $\mathcal{M}_3$*;* (b) *the associated grid (edges represent mated opposing arcs)*; (c) *the* 2×2 *corner of a grid with all incident arcs.*

Floors and ants' move-repertoires. The $n \times n$ laboratory floor is a square *mesh* of *tiles*, n along each side (Fig. 1(a)); the tiles are indexed by the set $[0, n-1] \times [0, n-1]$.[1] We represent the floor algorithmically by the $n \times n$ *grid(-graph)* (Fig. 1(b)). $\mathcal{M}_n$ ambiguously denotes the mesh and the grid; context will disambiguate each reference. Ants move along $\mathcal{M}_n$'s *King's-move* arcs, which are labeled by the compass directions, E, SE, S, SW, W, NW, N, NE. Each tile $v = \langle i, j \rangle$ of $\mathcal{M}_n$ is connected by mated in- and out-arcs to its neighboring tiles (Fig. 1(c)).

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

$\mathcal{M}_n$'s regions. $\mathcal{M}_n$'s *quadrants* are its induced subgraphs[2] on the sets of tiles indicated in Table 1. (The asymmetry in quadrant boundaries gives each tile a unique quadrant.)

For $k \in [0, 2n-2]$, $\mathcal{M}_n$'s kth *diagonal* is the set of tiles

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

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

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

[2]Let the directed graph $\mathcal{G}$ have tile-set N and arc-set A. The *induced subgraph of* $\mathcal{G}$ *on the set* $N' \subseteq N$ is the subgraph of $\mathcal{G}$ with tile-set N' and all arcs from A both of whose endpoints reside in N'.

Table 1: *The four quadrants of* $\mathcal{M}_n$.

Quadrant	Name	Tile-set
SOUTHWEST	$\mathcal{Q}_{SW}$	$[0, \lceil n/2 \rceil - 1] \times [0, \lceil n/2 \rceil - 1]$
NORTHWEST	$\mathcal{Q}_{NW}$	$[\lceil n/2 \rceil, n-1] \times [0, \lceil n/2 \rceil - 1]$
SOUTHEAST	$\mathcal{Q}_{SE}$	$[0, \lceil n/2 \rceil - 1] \times [\lceil n/2 \rceil, n-1]$
NORTHEAST	$\mathcal{Q}_{NE}$	$[\lceil n/2 \rceil, m-1] \times [\lceil n/2 \rceil, n-1]$

In analogy with its quadrants, which are "fenced off" by imaginary vertical and horizontal side-bisecting lines, $\mathcal{M}_n$ has four *wedges*, $\mathcal{W}_N$, $\mathcal{W}_E$, $\mathcal{W}_S$, $\mathcal{W}_W$, which are "fenced off" by imaginary diagonals that connect its corners; see Table 2. (Again, asymmetric boundaries ensure unique wedges.)

Table 2: *The four wedges of* $\mathcal{M}_n$.

Wedge	Name	Tile-set
NORTH	$\mathcal{W}_N$	$\{\langle x, y\rangle \mid [x \geq y]$ and $[x+y \geq n-1]\}$
SOUTH	$\mathcal{W}_S$	$\{\langle x, y\rangle \mid [x < y]$ and $[x+y < n-1]\}$
EAST	$\mathcal{W}_E$	$\{\langle x, y\rangle \mid [x < y]$ and $[x+y \geq n-1]\}$
WEST	$\mathcal{W}_W$	$\{\langle x, y\rangle \mid [x \geq y]$ and $[x+y < n-1]\}$

"Virtual" pheromones. Each tile of $\mathcal{M}_n$ contains some fixed number c of counters, with counter i capable of holding an integer in the range $[0, I_i]$. Each counter represents a *"virtual" pheromone* (cf. [3])—a digital realization of nature's ants' volatile organic compounds—and each value in the range $[0, I_i]$ is an *intensity level* of pheromone i. The number c and the ranges $[0, I_1], \ldots, [0, I_c]$ are characteristics/parameters of a specific realization of the model. The volatility of nature's pheromones is modeled by a schedule of decrements of every pheromone counter; see Fig. 2. Every computation begins with all tiles having level 0 of every pheromone.

2.1.2 Computers and finite-state programs

Each ant contains a computer that endows it with "intelligence." Each computer possesses a number of I/O ports that is sufficient for communicating with the outside world and with the computers on adjacent tiles of $\mathcal{M}_n$. In a single "step," a computer is capable of:

1. detecting an ant in an adjacent tile;
2. recognizing $\mathcal{M}_n$'s four edges/sides and its four corners;
3. communicating with each *neighboring* computer—i.e., one on a tile that shares an edge or corner—by receiving one message and transmitting one message at each time-step.
4. receiving a command from the outside world.

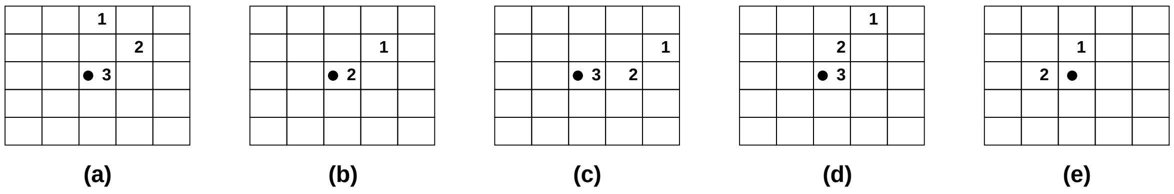

Figure 2: *Snapshots of a pheromone of intensity* $I = 3$ *changing as an FSM-ant* $\mathcal{F}$ *(the dot) moves. All snapshots have* $\mathcal{F}$ *on the center tile. Unlabeled tiles have level* 0. (*a*) $\mathcal{F}$ *has deposited a maximum dose of the pheromone on each tile that it has reached via a* 2*-step* SE*-*SW *path; note that the pheromone has begun "evaporating" on the tiles that* $\mathcal{F}$ *has left.* (*b*) $\mathcal{F}$ *stands still for one time-step.* (*c*) $\mathcal{F}$ *moves* W *and deposits a maximum dose of the pheromone.* (*d*) $\mathcal{F}$ *moves* S *and deposits a maximum dose of the pheromone.* (*e*) $\mathcal{F}$ *moves* E *and does not deposit any pheromone.*

In order to enhance the likelihood that we are studying an economically feasible computational model, we insist that, for every task, all embedded computers function as identical copies of a fixed, specialized *finite-state machine* (*FSM*) $\mathcal{F}$; cf. [9]. Rather than employ the standard way of specifying FSMs, which is more useful for proofs than for algorithms, we specify FSMs via programs that are *finite sets of case statements* of the form depicted in Table 3;[3] the table

Table 3: *A case-statement program that specifies an FSM* $\mathcal{F}$.

```
LABEL$_1$:  **if** INPUT$_1$ **then** OUTPUT$_{1,1}$ **and goto** LABEL$_{1,1}$
              ⋮
            **if** INPUT$_m$ **then** OUTPUT$_{1,m}$ **and goto** LABEL$_{1,m}$
------------------------------------------------------------------
   ⋮          ⋮
------------------------------------------------------------------
LABEL$_s$:  **if** INPUT$_1$ **then** OUTPUT$_{s,1}$ **and goto** LABEL$_{s,1}$
              ⋮
            **if** INPUT$_m$ **then** OUTPUT$_{s,m}$ **and goto** LABEL$_{s,m}$
```

specifies an FSM $\mathcal{F}$ as follows:

- $\mathcal{F}$ has s states, named LABEL$_1$, ..., LABEL$_s$.
- $\mathcal{F}$ responds to a fixed repertoire of *inputs* from the set {INPUT$_1$, ..., INPUT$_m$}. Each INPUT$_i$ is a combination of:
  - the messages that $\mathcal{F}$ receives—from neighboring FSMs and from the outside world;
  - the presence/absence of an edge/corner of $\mathcal{M}_n$, a "food" item to be manipulated, an "obstacle" to be avoided;
  - the levels of intensity of the c pheromones that could be present on the current tile. Fig. 2 depicts a pheromone of intensity $I = 3$ changing as an FSM $\mathcal{F}$ (the dot) moves. Observe the "evaporation" of the pheromone throughout the figure.

[3]The CARPET programming environment [11] employs a similar programming style.

- $\mathcal{F}$ responds to the current input by:
  - emitting an *output* from a fixed repertoire; each specific output is denoted by OUTPUT$_k$, with some appropriate identifying subscript k. Outputs are combinations of:
    - the messages that $\mathcal{F}$ sends to neighboring FSMs;
    - pheromone-related actions: deposit a pheromone of type h at intensity $I \leq I_h$, enhance a pheromone of type j by increasing its current intensity to level $I \leq I_j$;
    - "food"-related actions: pick up and carry the item on the current square, deposit the item that $\mathcal{F}$ is currently carrying;
    - stand still or move to a neighboring tile in a direction that is appropriate for $\mathcal{F}$'s current tile—i.e., that will not cause $\mathcal{F}$ to "fall off" $\mathcal{M}_n$ by trying to follow a nonexistent arc.
  - changing state (by specifying the next case statement to execute).

We specify algorithms in English—but hopefully in enough detail to make it clear how to craft equivalent finite-state programs.

2.2 The *Parking Problem* for Ants

The parking problem has each ant A move as close as possible to the corner tile of $\mathcal{M}_n$ that resides in the same quadrant as A does at the moment when it receives the INITIATE_PARKING command (from the outside world). Additionally, the ants in each quadrant must cluster within the most compact quarter-sphere "centered" at the quadrant's corner tile. Focus on $\mathcal{Q}_{SW}$ (easy clerical changes work for the other quadrants), and consider Fig. 3. *A configuration of ants solves*

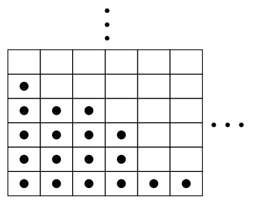

Figure 3: *The* 6×6 *"prefix" of* $\mathcal{Q}_{SW}$*, with* 18 *optimally parked ants (denoted by dots).*

the parking problem for $\mathcal{Q}_{SW}$ *precisely if it minimizes the* parking potential function

$$\Pi(t) \overset{\text{def}}{=} \sum_{k=0}^{2n-2} (k+1) \times \text{(the number of ants residing on } \Delta_k \text{ at step } t). \qquad (2.1)$$

3 Single Ants and Parking

Although we have not yet settled the question of whether single ants can park on $\mathcal{M}_n$, we have made partial progress.

3.1 The Simplified Framework for Single Ants

This section focuses on the parking abilities of a *single* ant, $\mathcal{F}$, on $\mathcal{M}_n$. This focus allows us to simplify the input and output sets of the FSMs of this section, because the FSMs are alone on $\mathcal{M}_n$. Easily, these sets need no longer contain messages from/for, because there are no ants for $\mathcal{F}$ to receive message from or to send messages to. Less obviously, these sets need no longer contain pheromone levels, because pheromones do not enhance the path-planning power of a single ant.

Proposition 3.1. *Given any FSM $\mathcal{F}$ that employs pheromones while navigating $\mathcal{M}_n$, there exists an FSM $\mathcal{F}'$ that follows the same trajectory as $\mathcal{F}$ while not using pheromones.*

Proof. For simplicity, assume that $\mathcal{F}$ employs just one type of pheromone; if it uses many, then we can eliminate them one at a time. Say that $\mathcal{F}$'s single pheromone has possible intensities $0, \ldots, I$. We replace $\mathcal{F}$ by an FSM $\mathcal{F}'$ that employs no pheromones and that simulates $\mathcal{F}$ by "carrying around" (inside its finite-state control) a *map* that encapsulates all necessary information about the positions and intensities of the pheromone on $\mathcal{M}_n$. To verify that $\mathcal{F}'$ exists, we must check: (a) that the map is "small"—i.e., has size independent of n—and (b) that $\mathcal{F}'$ can update the map as it simulates any series of steps by $\mathcal{F}$. Of course, the map indicates that initially there are no tiles that contain the pheromone.

The size of the map. The portion of $\mathcal{M}_n$ that could contain nonzero levels of the pheromone is no larger than the "radius"-I subgrid of $\mathcal{M}_n$ that $\mathcal{F}$ has been in during the most recent I steps. No trace of pheromone can persist outside this region because of volatility. Thus, the map needs only be a mesh of size $(2I-1) \times (2I-1)$ centered at $\mathcal{F}$'s current tile. (The map may be smaller if $\mathcal{F}$ is currently near an edge of $\mathcal{M}_n$.) Because $\mathcal{F}$ is the only FSM in our scenario, at most one tile of the map can contain the integer I (indicating a maximum level of the pheromone), at most one tile the integer $I-1$ (indicating that it contained a maximum level one step ago), and so on, down to at most one tile that contains the integer 1 (indicating that it contained a maximum level $I-1$ steps ago). Fig. 2(a) displays a sample map for the case $I = 3$, together with four sample one-step updates: Figs. 2(b)–(e).

Updating the map. Because of the restrictions on a map's size and contents, there are strictly fewer than $1 + \prod_{j=0}^{I-1}((2I-1)^2 - j)$ distinct possible maps ("strictly fewer" because this reckoning ignores the necessary adjacency of tiles that contain the integers k and $k-1$). $\mathcal{F}'$ can, therefore, carry around the set of all possible maps in its finite-state memory, with the then-current map in a special "register." Formally, this amounts to having each state of $\mathcal{F}'$ have the form $\langle q, \text{ (the set of all possible maps), (the current map)} \rangle$, where q is a state of $\mathcal{F}$. Clearly $\mathcal{F}'$ has only finitely many states as long as $\mathcal{F}$ does. The state-transition function of $\mathcal{F}'$ augments that of $\mathcal{F}$ by updating the map-component of the state while emulating $\mathcal{F}$'s control-state change. $\square$

3.2 A Single Ant Cannot Park on a One-Dimensional Mesh

Theorem 3.1. *It is not possible to design an FSM-ant that successfully parks, when started in an arbitrary tile of the one-dimensional mesh M_n.*

Proof. The one-dimensional parking problem has the following form. We place an FSM $\mathcal{F}$ with state-set Q on a one-dimensional M_n and order it to park. Clearly $\mathcal{F}$ must end up at either the eastern endpoint of M_n (tile $n-1$) or the western endpoint (tile 0), whichever is closer to its initial tile; ties are broken in favor of the western end. By Proposition 3.1, we lose no generality by positing that $\mathcal{F}$ does not employ pheromones.

Focus on a "large" instance of this parking problem, where "large" means that $n \gg |Q|$. Say that this instance initially places $\mathcal{F}$ at a tile $k \in \{\lfloor \frac{1}{3}(n-1) \rfloor, \ldots, \lceil \frac{2}{3}(n-1) \rceil\}$ of M_n. Focus on a situation in which $k \leq \frac{1}{2}(n-1)$, so that $\mathcal{F}$'s target endpoint is tile 0. (A symmetric argument works if $k > \frac{1}{2}(n-1)$, so that $\mathcal{F}$'s target endpoint is tile $n-1$.)

Label each tile t of M_n with symbols from the set $Q \cup \{\#\}$, as follows.

- If $\mathcal{F}$ never visits tile t on its trajectory from tile k to tile 0, then label t with the symbol #. (There may be no tiles that $\mathcal{F}$ does not visit.)
- If $\mathcal{F}$ does visit tile t on its trajectory from tile k to tile 0, then label t with (the name of) the state that $\mathcal{F}$ is in the *last time* that it visits t. ($\mathcal{F}$ *must* visit every tile between 0 and k, because of M_n's one-dimensionality.)

Because tile k is at least $\lfloor \frac{1}{3}(n-1) \rfloor$ tiles away from $\mathcal{F}$'s target tile 0, and because $n \gg |Q|$, there must be distinct tiles strictly between tiles 0 and k that have the same label $q \in Q$. Because the FSM $\mathcal{F}$ is deterministic, it cannot "distinguish" between any two like-labeled tiles. In particular, if we lengthen M_n—i.e., increase n—by "cutting and splicing" portions of M_n that begin and end with label q—see Fig. 4—then this will not impact $\mathcal{F}$'s ultimate behavior: $\mathcal{F}$ will still end its journey on tile 0, despite the fact that the "cut and splice" operation has lengthened the distance between tile k and tile 0. This determinism is $\mathcal{F}$'s downfall! If we perform the "cut and splice"

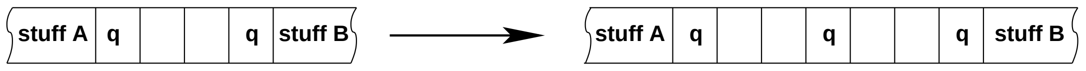

Figure 4: *Illustrating the "cut and splice" operation of Theorem 3.1.*

operation enough times, we can make tile k as far from tile 0 as we want. In particular, we can make this distance greater than the distance between tile k and tile $n-1$. Once this happens, $\mathcal{F}$ is no longer parking successfully.

Because we have made no assumptions about $\mathcal{F}$ other than its finiteness, the theorem follows. ☐

3.3 Single-Ants Configurations that Can Park

Theorem 3.2. *A single FSM-ant can park in time $O(n)$ when started on an arbitrary* edge-tile *of $\mathcal{M}_n$.*

Proof. Say, with no loss of generality, that the FSM $\mathcal{F}$ begins on the bottom edge of $\mathcal{M}_n$, say at tile $\langle 0, k \rangle$. Clearly, $\mathcal{F}$'s parking corner will be either tile $\langle 0, 0 \rangle$ or tile $\langle 0, n-1 \rangle$. We show how $\mathcal{F}$ can decide which of these alternatives is the correct one.

Let $\mathcal{F}$ begin a $60°$ northeasterly walk from tile $\langle 0, k\rangle$, i.e., a walk consisting of "supersteps" of the form $(0, +1)$, $(+1, 0)$, $(+1, 0)$ (alternating single easterly steps and double northerly steps; see Fig. 5). Continuing thus until it encounters either an edge or a corner of $\mathcal{M}_n$, $\mathcal{F}$ can determine

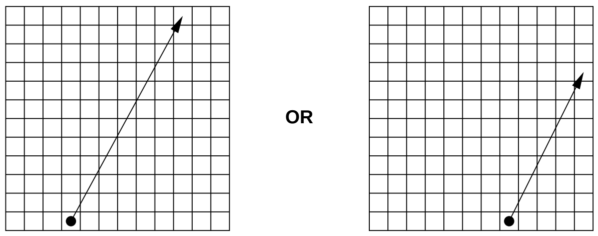

Figure 5: *The two possible edge-terminating trajectories for a single ant.*

its target parking tile from the endpoint of its walk. To see this, note that after s "supersteps", $\mathcal{F}$ has moved from tile $\langle 0, k\rangle$ to tile $\langle 2s, k + s\rangle$. Consequently:

- If $\mathcal{F}$'s walk terminates in the right edge of $\mathcal{M}_n$, then $\mathcal{F}$ must park in tile $\langle 0, n - 1\rangle$. This is because $2s < n - 1$, while $k + s \geq n - 1$, so that $k > \frac{1}{2}(n - 1)$.
- If $\mathcal{F}$'s walk terminates in either the top edge or the NE corner of $\mathcal{M}_n$, then $\mathcal{F}$ must park in tile $\langle 0, 0\rangle$. This is because $2s \geq n - 1$, while $k + s \leq n - 1$, so that $k \leq \frac{1}{2}(n - 1)$.

This completes the proof. $\square$

4 Multi-Ant Configurations that Can Park

We do not yet know if a collection of ants can park successfully when started in an arbitrary initial configuration in $\mathcal{M}_n$. But we do not need to specialize the initial configuration very much in order to ensure successful parking.

We call a pair of ants *adjacent* if the ants reside on tiles that share an edge or a corner. We call a collection of more than two ants *adjacent* if the King's-move adjacency graph of the tiles they reside in is connected.

Theorem 4.1. *Any collection of ants that contains at least two adjacent ants can determine their home quadrants in $O(n^2)$ synchronous steps.*

The proof of Theorem 4.1 addresses the three components of the activity of parking:

1. having each ant determine its home quadrant;
2. having each ant use this information to plan a route to its target corner of $\mathcal{M}_n$;
3. having ants that share the same home quadrant—hence, the same target corner—organize into a configuration that minimizes the parking potential function (2.1).

Section 4.1 treats the algorithmically most complex of these activities, identifying home quadrants; Section 4.2 treats the other two activities.

4.1 Quadrant Determination with the Help of Adjacent Ants

We address the problem of quadrant determination in three parts. Section 4.1.1 presents an algorithm that allows *two* adjacent ants to park. Because this algorithm is somewhat complicated, we present in Section 4.1.2 a much simpler algorithm that allows *three* adjacent ants to park. We then show in Section 4.1.3 how two adjacent ants can act as "shepherds" and help any collection of ants to determine their home quadrants.

4.1.1 Two adjacent ants can park

The following algorithm was developed in collaboration with Olivier Beaumont (INRIA and Univ. of Bordeaux). The essence of the algorithm is depicted in Fig. 6.

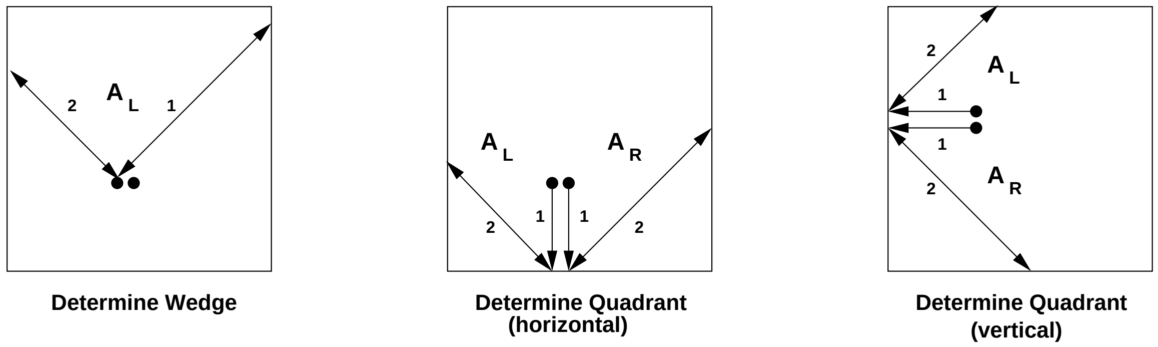
2
A L
1
Determine Wedge
A L
A R
2
1
1
2
Determine Quadrant (horizontal)
2
A L
1
1
A R
2
Determine Quadrant (vertical)

Figure 6: *The essence of quadrant determination for two adjacent ants:* (left) *wedge determination;* (middle) *quadrant determination, horizontal component;* (right) *quadrant determination, vertical component.*

Say that there are two adjacent ants on $\mathcal{M}_n$ when the command to park is issued. Say, for definiteness, that the ants are horizontally adjacent: $A_{\rm L}$, on tile $\langle x, y\rangle$, resides to the left of $A_{\rm R}$, whose tile is $\langle x, y+1\rangle$. (This assumption loses no generality, because the ants can remember their true initial configuration in their finite-state memories, then move into the assumed left-right configuration, and finally compute the adjustments necessary to make the correct determination about their home quadrants.) Our algorithm has two phases.

Determining ants' home wedges. We begin the quadrant-identification process by having $A_{\rm L}$ perform two walks within $\mathcal{M}_n$ from its initial tile; see Fig. 6(left).

The 45° *walk.* $A_{\rm L}$ executes a sequence of $(+1, +1)$ moves until it encounters $\mathcal{M}_n$'s northeastern corner or its top edge or its right edge. It then retraces its steps to tile $\langle x, y\rangle$, which is detected by the horizontal adjacency of $A_{\rm R}$. The terminus of $A_{\rm L}$'s outward walk narrows down both $A_{\rm L}$'s and $A_{\rm R}$'s home wedges to two possibilities:

Walk terminated by:	A_L's home wedge:	A_R's home wedge:
$\mathcal{M}_n$'s corner	$\mathcal{W}_N$ or $\mathcal{W}_W$	$\mathcal{W}_E$ or $\mathcal{W}_S$
$\mathcal{M}_n$'s top edge	$\mathcal{W}_N$ or $\mathcal{W}_W$	$\mathcal{W}_N$ or $\mathcal{W}_W$
$\mathcal{M}_n$'s right edge	$\mathcal{W}_E$ or $\mathcal{W}_S$	$\mathcal{W}_E$ or $\mathcal{W}_S$

The 315° *walk.* A_L executes a sequence of $(+1, -1)$ moves until it encounters $\mathcal{M}_n$'s northwestern corner or its top edge or its left edge. It then retraces its steps to tile $\langle x, y \rangle$. Here again, the terminus of A_L's outward walk narrows down both A_L's and A_R's home wedges to two possibilities.

Walk terminated by:	A_L's home wedge:	A_R's home wedge:
$\mathcal{M}_n$'s corner or top	$\mathcal{W}_N$ or $\mathcal{W}_E$	$\mathcal{W}_N$ or $\mathcal{W}_E$
$\mathcal{M}_n$'s left edge	$\mathcal{W}_W$ or $\mathcal{W}_S$	If walk ends within one tile of $\mathcal{M}_n$'s corner then: $\mathcal{W}_N$ or $\mathcal{W}_E$ else: $\mathcal{W}_W$ or $\mathcal{W}_S$

Thus, after the two walks, each of A_L and A_R knows its home wedge:

45° walk terminus	315° walk terminus	A_L's home wedge:	A_R's home wedge:
corner	corner/top	$\mathcal{W}_N$	$\mathcal{W}_E$
corner	left edge	$\mathcal{W}_W$	If walk ends within one tile of $\mathcal{M}_n$'s corner then: $\mathcal{W}_E$ else: $\mathcal{W}_S$
top edge	corner/top	$\mathcal{W}_N$	$\mathcal{W}_N$
top edge	left edge	$\mathcal{W}_W$	If walk ends within one tile of $\mathcal{M}_n$'s corner then: $\mathcal{W}_N$ else: $\mathcal{W}_W$
right edge	corner/top	$\mathcal{W}_E$	$\mathcal{W}_E$
right edge	left edge	$\mathcal{W}_S$	If walk ends within one tile of $\mathcal{M}_n$'s corner then: $\mathcal{W}_E$ else: $\mathcal{W}_S$

Determining ants' home quadrants. A_L and A_R use the knowledge of their home wedge(s) to determine their home *quadrant(s)*. Having been very detailed in determining the ants' home wedges, we relax a bit in determining their home quadrants, by leaving detailed calculations to the reader. Quadrant determination involves two subprocedures.

The horizontal procedure (Fig. 6(middle)). A_L and A_R move in lockstep to the bottom edge of $\mathcal{M}_n$. Having achieved this edge, they set out in lockstep on independent diagonal roundtrip walks. A_L proceeds outward at an angle of 315°, via $(+1, -1)$ moves until it hits $\mathcal{M}_n$'s left edge; it then reverses direction until it regains the bottom edge. Simultaneously, A_R proceeds at an angle of 45°, via $(+1, +1)$ moves until it hits $\mathcal{M}_n$'s right edge; it then reverses direction until it regains the bottom edge. *Importantly, the angles of the outward and return walks ensure that each ant returns to its original tile on the bottom edge.*

Once an ant regains the bottom edge, it checks for the presence of the other ant in the appropriate adjacent tile, to determine the outcome of their round-trip "race." Note that A_L has traveled $2m_\mathrm{L}$ steps, while A_R has traveled $2m_\mathrm{R}$ steps, where $m_\mathrm{L} + m_\mathrm{R} = n - 2$. A discrete set of outcomes of the "race" provides A_L and A_R bounds on the columns of $\mathcal{M}_n$ where each began to park: Did the "race" end in a tie? Did the winner win by exactly one step? Did the winner win by more than one step? Provided that A_L and A_R did not both begin in wedge $\mathcal{W}_E$ and that both did not begin in wedge $\mathcal{W}_W$, the information about home columns allows A_L and A_R to determine their home quadrants. In order to accommodate situations wherein A_{L} and A_{R} both began in $\mathcal{W}_E$ or both began in $\mathcal{W}_W$, we need to supplement the information about home columns obtained from the horizontal procedure with information about home rows. To this end, we employ a vertical analogue of the horizontal procedure.

The vertical procedure (Fig. 6(right)). A_{L} and A_{R} move in lockstep to the left edge of $\mathcal{M}_n$, whence they set out in lockstep on independent diagonal roundtrip walks. A_{L} proceeds outward at an angle of 45°, via $(+1, +1)$ moves until it hits $\mathcal{M}_n$'s top edge, where it reverses direction until it regains the left edge. Simultaneously, A_{R} proceeds at an angle of 135°, via $(-1, +1)$ moves until it hits $\mathcal{M}_n$'s bottom edge, where it reverses direction until it regains the left edge.

In direct analogy with the information about columns that the horizontal procedure yields, the vertical procedure procedure affords bounds on the home rows of A_{L} and A_{R}.

After executing both the horizontal and vertical procedures, A_{L} and A_{R} can determine their respective home quadrants. As noted earlier, this final determination requires A_{L} and A_{R} to adjust the results from the various walks we have described, if necessary, to account for the initial realignment. Once possessing this knowledge, it is an easy matter for A_{L} and A_{R} to park greedily.

4.1.2 Three adjacent ants can park

Say that there are three adjacent ants, A_1, A_2, and A_3, on $\mathcal{M}_n$ when the command to park is issued. Via local communication, the ants can recognize which of the six possible distinct configurations they occupy as a group, and each can remember its location relative to the others. It can, therefore, adjust the result of the quadrant determination algorithm that we describe now and depict schematically in Fig. 7.

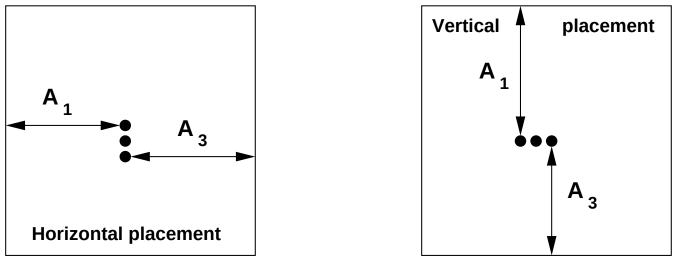

Figure 7: *The essence of the parking algorithm for three adjacent ants:* (Left) *Determining each ant's horizontal placement.* (Right) *Determining each ant's vertical placement.*

Determining ants' home quadrants.
Horizontal placement (Fig. 7(Left)). The ants align themselves vertically in the top-to-bottom order A_1, A_2, A_3, with A_2 retaining its original position. (If A_2 originally sits on either the top or bottom edge of $\mathcal{M}_n$, then the ants will have to shift one tile northward or southward. We leave the clerical details to the reader.) A_1 marches leftward to the left edge of $\mathcal{M}_n$ and returns to its pre-march position, above A_2. In lockstep, A_3 marches rightward to the right edge of $\mathcal{M}_n$ and returns to its pre-march position, below A_2.

- If A_1 and A_3 return to A_2 at the same step, or if A_1 returns to A_2 one step before A_3 does, then A_2 began this activity residing in one of $\mathcal{M}_n$'s western quadrants. Moreover, A_1 and A_3 can determine their original quadrants by noting how they moved in order to achieve the vertical alignment.
- If A_1 returns to A_2 two or more steps before A_3 does, then all three ants began this activity residing in one of $\mathcal{M}_n$'s western quadrants.

The situation when A_3 returns to A_2 before A_1 does is almost the mirror image—except for the asymmetry in eastern and western quadrants' boundaries.

Vertical placement (Fig. 7(Right)). This determination is achieved by "rotating the horizontal-placement algorithm by 90°." One begins with the ants in the left-to-right configuration A_1, A_2, A_3, with A_2 retaining its original position. (Again, a simple modification works if A_2 began on either the left or right edge of $\mathcal{M}_n$.) A_1 and A_3 reprise their scouting roles from the horizontal-placement algorithm, except now with upward and downward initial trajectories, respectively. Mirroring the analysis of the horizontal-placement algorithm, each ant can now determine whether it began in a northern quadrant of $\mathcal{M}_n$ or a southern one.

After horizontal and vertical placement, each ant knows which quadrant it began in. Possessing this knowledge, the ants can park in the manner detailed in Section 4.2.

4.1.3 Two adjacent ants acting as shepherds

All ants in any collection that contains (at least) two adjacent ants are able to determine their respective home quadrants—with the help of the adjacent ants. As we verify this, we encounter for the first time instances of ants blocking the intended paths of other ants. We deal with this problem by simply having conflicting ants *switch roles*—which is possible because all ants are identical. If ant A is blocking ant B's progress, then A "becomes" B and continues B's blocked trajectory; and ant B "becomes" ant A and moves onto the tile that A relinquishes when it "became" B. We henceforth invoke this strategy without further comment.

Proposition 4.1. *Any collection of ants that contains two or more adjacent ants can determine all home quadrants within $O(n^2)$ synchronous steps.*

Proof. We have a collection of $m \geq 2$ ants, $A_1, \ldots, A_m$, that contains at least two adjacent ants—without loss of generality, ants A_1 and A_2. We have A_1 and A_2 act as *shepherds* that help all other ants, $A_3, \ldots, A_m$, determine their home quadrants. The quadrant-determining algorithm operates in three phases.

Phase 1: A_1 and A_2 determine their home quadrant(s), using the algorithm of Section 4.1.1. They store this information in their memories; they will use it to park in phase 3.

Phase 2: A_1 and A_2 help other ants determine their home quadrants. There are four subphases to this phase:

Subphase a: *A_1 and A_2 distinguish east from west.* A_1 and A_2 head to the southwestern corner of $\mathcal{M}_n$, where they begin to walk. A_1 makes a round trip from tile $\langle 0,0\rangle$ to tile $\langle 0,n-1\rangle$ and back, in order to determine the parity of n.

If n is even, then:

- A_2 moves one tile eastward at *every time-step* until it reaches $\mathcal{M}_n$'s right edge. At that point, it *reverses direction* and begins to move one tile westward at every time-step.
- Starting one step later, A_1 moves one tile eastward at *every third time-step.*
- A_1 and A_2 stop when they meet in adjacent tiles, with A_1 on tile $\langle 0, \frac{1}{2}n-1\rangle$ and A_2 on tile $\langle 0, \frac{1}{2}n\rangle$.

When A_1 and A_2 meet, they have determined the midpoint of $\mathcal{M}_n$'s bottom row (hence, of every row). To wit:

- A_2's U-shaped trajectory from tile $\langle 0,0\rangle$ to tile $\langle 0,n-1\rangle$ and thence to tile $\langle 0, \frac{1}{2}n\rangle$ takes $(n-1)+(\frac{1}{2}n-1) = \frac{3}{2}n-2$ time-steps.
- A_1's trajectory from tile $\langle 0,0\rangle$ to tile $\langle 0, \frac{1}{2}n-1\rangle$ takes $\frac{1}{2}n-3$ steps. Because it starts one step later than A_2 does, A_1 arrive at tile $\langle 0, \frac{1}{2}n-1\rangle$ after $\frac{3}{2}n-2$ time-steps.

If n is odd, then:

- A_2 moves one tile eastward at *every time-step* until it reaches $\mathcal{M}_n$'s right edge. At that point, it *reverses direction* and begins to move one tile westward at every time-step.
- Starting one time-step later, A_1 moves one tile eastward at *every third time-step.*
- A_1 and A_2 stop when they meet in adjacent tiles, with A_1 on tile $\langle 0, \lceil \frac{1}{2}n\rceil - 1\rangle$ and A_2 on tile $\langle 0, \lceil \frac{1}{2}n\rceil\rangle$.

When A_1 and A_2 meet, they have determined the midpoint of $\mathcal{M}_n$'s bottom row (hence, of every row). To wit:

- A_2's U-shaped trajectory from tile $\langle 0,0\rangle$ to tile $\langle 0,n-1\rangle$ and thence to tile $\langle 0, \lceil \frac{1}{2}n\rceil\rangle$ takes $(n-1)+(\lfloor \frac{1}{2}n\rfloor - 1) = 3\lceil \frac{1}{2}n\rceil - 4$ time-steps.
- A_1's trajectory from tile $\langle 0,0\rangle$ to tile $\langle 0, \lceil \frac{1}{2}n\rceil - 1\rangle$ takes $3\lceil \frac{1}{2}n\rceil - 3$ steps. Because it starts one step later than A_2 does, A_1 arrive at tile $\langle 0, \lceil \frac{1}{2}n\rceil - 1\rangle$ after $3\lceil \frac{1}{2}n\rceil - 4$ time-steps.

Subphase b: *A_1 and A_2 identify easterners and westerners.* A_1 walks through the western half of $\mathcal{M}_n$, column by column, telling each encountered ant that it is a westerner—i.e., that it resides in either $\mathcal{Q}_{NW}$ or $\mathcal{Q}_{SW}$. Simultaneously, A_2 does the same in the eastern half of $\mathcal{M}_n$, column by column, telling each encountered ant that it is an easterner—i.e., that it resides in either $\mathcal{Q}_{NE}$ or $\mathcal{Q}_{SE}$.

A_1 and A_2 meet at the northwestern corner of $\mathcal{M}_n$ after their walks.

Subphase c: *A_1 and A_2 distinguish north from south.* A_1 and A_2 start at the northwestern corner of $\mathcal{M}_n$ and begin to walk:

- A_1 moves one tile southward at *every second time-step.*
- A_2 moves one tile southward at *every time-step* until it reaches $\mathcal{M}_n$'s bottom edge. At that point, it *reverses direction* and begins to move one tile northward at every time-step, until it meets A_1.

When A_1 and A_2 meet, they have determined the midpoint of $\mathcal{M}_n$'s left column (hence of every column).

Subphase d: *A_1 and A_2 identify northerners and southerners.* A_1 walks through the northern half of $\mathcal{M}_n$, row by row, telling each encountered ant that it is a northerner—i.e., that it resides in either $\mathcal{Q}_{NE}$ or $\mathcal{Q}_{NW}$. A_2 does the same in the southern half of $\mathcal{M}_n$, row by row, telling each encountered ant that it is a southerner—i.e., that it resides in either $\mathcal{Q}_{SE}$ or $\mathcal{Q}_{SW}$.

By the end of Phase 2, every ant knows its home quadrant.

Phase 3: Ants park. Every ant except for A_1 and A_2 begins to park as soon as it learns its home quadrant—which occurs no later than the end of phase 2. Because A_1 and A_2 learned their own home quadrants in phase 1, they are able to park at the completion of phase 2. □

4.2 Completing the Parking Process

We now describe how an ant that knows its home quadrant plans a route to its target corner of $\mathcal{M}_n$ and how all ants that share the same home quadrant—hence, the same target corner—organize themselves into a configuration that minimizes the parking potential function (2.1).

The route to the target corner. We simplify this final component of the parking process by having all ants follow very simple routes to their respective target corners. Specifically: (1) Each ant goes horizontally to its closer vertical edge, proceeding at the rate of one tile every second time-step, in order to resolve traffic congestion (contention for tiles). (2) As ants reach their vertical edge, they proceed single file along that edge to their target corner. If an ant at the end of the horizontal component of its route encounters an ant that is already walking along the vertical edge, then the horizontal ant defers to the vertical ant. The half-rate horizontal progress permits the delayed ant to tell the horizontal ant behind it to stop and wait ... and that ant tells the one behind it, etc. Empty tiles alert ants that were proceeding horizontally to start walking again.

Organizing within the target corner. When ants that are walking along the vertical edge reach their target corner, they begin to fill in that corner via the snaking trajectory illustrated in Fig. 8. One shows easily that this manner of filling the corner organizes the ants into a configuration that minimizes the parking potential function (2.1).

This completes the parking algorithm and the proof of Theorem 4.1. □

5 Conclusions

We have reported here on progress in understanding the algorithmic strengths and weaknesses of ant-inspired robots within a geographically constrained environment. The vehicle for obtaining

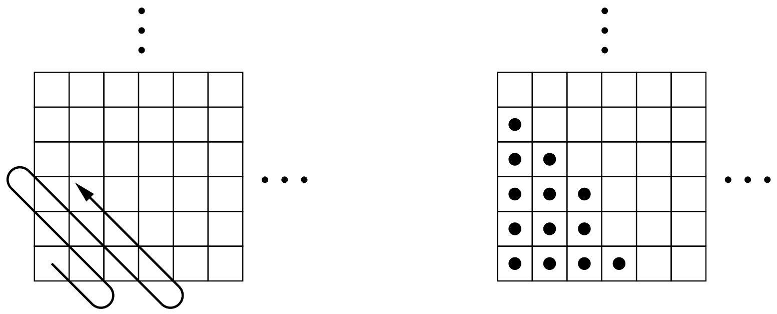

Figure 8: (Left) *The generic snaking parking trajectory, as observed in* $\mathcal{Q}_{SW}$*.* (Right) *The final parking configuration of* 13 *ants in* $\mathcal{Q}_{SW}$*.*

this understanding has been the simple path-planning problem we call *parking:* have ants configure themselves in maximally compact manner within the nearest corner. We have illustrated a variety of initial configurations of a collection of ants that enable successful, efficient parking, the strongest being just that the collection contains two ants that are initially adjacent. We have also exposed a situation—a single ant in a one-dimensional world—where parking is impossible. We mention "for the record" that if efficiency is unimportant, then any collection of ants that contains at least four adjacent ones can perform a vast array of path-planning computations (and others as well), by simulating an autonomous (i.e., input-less) 2-counter Register Machine whose registers have capacity $O(n^2)$; cf. [9].

Where do we go from here? Most obviously, we would like to solve the parking problem for ants definitively, by identifying precisely which initial configurations enable parking and which do not. It would also be valuable to understand the capabilties of ant-inspired robots within the context of other significant tasks that involve path planning [1, 4, 5, 7, 8], including those that involves finding and transporting "food" and avoiding obstacles (as in [7, 8]).

Acknowledgments. This research was supported in part by NSF Grant CNS-0905399. I am indebted to O. Beaumont and O. Brock for insightful comments and suggestions.

References

[1] 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.*

[2] 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.

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

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

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

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

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

[8] A.L. Rosenberg (2008): Cellular ANTomata: food-finding and maze-threading. *37th Intl. Conf. on Parallel Processing (ICPP).*

[9] A.L. Rosenberg (2009): *The Pillars of Computation Theory: State, Encoding, Nondeterminism.* Universitext Series, Springer, New York.

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

[11] 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.