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" to perform complex tasks. This paper is a step toward understanding the algorithmic concomitants of this metaphor, the strengths and weaknesses of ant-based computation models. We study the ability of *finite-state* ant-robots to *scalably* perform a simple path-planning task called *parking*, within fixed, geographically constrained environments ("factory floors"). This task: (1) has each ant head for its nearest corner of the floor and (2) has all ants within a corner organize into a maximally compact formation. Even without (digital analogues of) pheromones, many initial configurations of ants *can* park, including: (a) a single ant situated along an edge of the floor; (b) any assemblage of ants that begins with two designated adjacent ants. In contrast, a single ant in the middle of (even a one-dimensional) floor *cannot* park, even with the help of (volatile digital) pheromones.

Keywords: Ant-inspired robots, Finite-state machines, Path planning.

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., [1,4,6,7,8,9]—have done precisely that. This paper is a step toward understanding the algorithmic concomitants of the robot-as-ant metaphor within the context of a simple, yet nontrivial, path-planning problem.

Ant-robots in a "factory." We focus on mobile robotic computers (henceforth, *ants*, to stress the natural inspiration) that function within a fixed geographically constrained environment (henceforth, a *factory floor* [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 real ants meet) and "timestamped message passing" (as when real ants deposit pheromones);
- assemble in desired locations, in desired configurations.

Although not relevant to the current study, we would also want ants to be able to discover goal objects ("food") and to convey "food" from one location to another; cf. [4,6,8,9].

Indeed, the "parking" problem that we study might be the aftermath of the preceding activities, e.g., if one has ants convey collected "food" to designated stations.

In the "standard realization" of ants studied here, ants are endowed with "intelligence" via embedded computers. These "intelligent" ants are responsible for planning and orchestrating assigned activities. In a quest to understand how a large assemblage of ants of *very* limited ability can "cooperate" to accomplish complex tasks, we focus on ants that are *mobile finite-state machines*. In particular, we want to understand what paths ants can plan in a *scalable* manner, i.e., without counting (beyond a fixed limit).

Having ants park. We study the preceding issue by focusing on a simple, yet algorithmically nontrivial, path-planning task that we studied in [8] under a rather different ant-based model. This task, *parking:* (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 (see Section 2.2 for details). While we have not yet characterized which initial configurations of ants can park successfully, 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 2);
  - *any assemblage of ants that begins with two distinguished ants that are adjacent*—i.e., on tiles that share an edge or a corner (Theorem 3).
- In contrast: *A single ant can generally* not *park, even on a* one-dimensional *floor and even with the help of (volatile digital) pheromones* (Theorem 1).

The algorithmic setting. We require algorithms to be *simple*, *scalable* and *decentralized*.
(1) *Algorithms must work on floors of arbitrary sizes.* An algorithm cannot exploit information about the size of the floor; it 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.B. These guidelines may be too costly to observe in practical systems; cf. [2,4,6].

2 Technical Background

2.1 Ant-Robots Formalized

A. Pheromone-bearing floors. Floors and connectivity (cf. [5]). The $n \times n$ floor is a square[1] *mesh* of *tiles*, n along each side (Fig. 1(a)); tiles are indexed by the set $[0, n-1] \times [0, n-1]$.[2] We represent the floor algorithmically by the $n \times n$ *grid(-graph)*

[1] Our study adapts to *rectangular* floors with little difficulty.

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

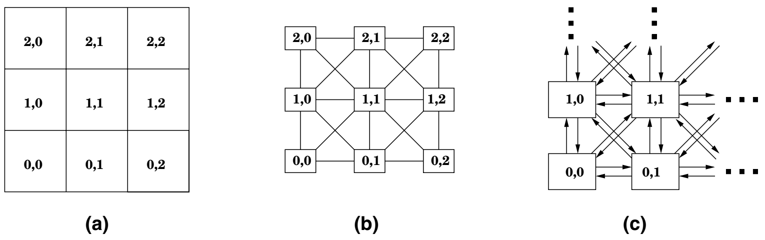

Fig. 1. (a) A 3×3 floor $\mathcal{M}_3$; (b) the associated grid (edges represent mated opposing arcs); (c) the 2×2 corner of a grid with all incident arcs

(Fig. 1(b)). $\mathcal{M}_n$ ambiguously denotes the mesh or the grid, according to context. Ants move along $\mathcal{M}_n$'s *King's-move* arcs, which are labeled by the compass directions. Each tile $v = \langle i, j \rangle$ of $\mathcal{M}_n$ is connected by mated in- and out-arcs to its neighbors (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[3] on the following sets of tiles:

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 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:

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 $[x + y < n-1]\}$

The asymmetries in boundaries give each tile a unique quadrant and wedge.

For $k \in [0, 2n-2]$, $\mathcal{M}_n$'s kth *diagonal* is the set of tiles $\Delta_k = \{\langle i, j \rangle \in [0, n-1] \times [0, n-1] \mid i + j = k\}$. For $d \in [1, 2n-1]$, the *radius-d quarter-sphere* of $\mathcal{Q}_{SW}$ is the union: $\bigcup_{k=0}^{d-1} \Delta_k$.

"Virtual" pheromones. Each tile of $\mathcal{M}_n$ contains a fixed number c of counters; each counter i can hold an integer in the range $[0, I_i]$. Each counter represents a *"virtual"*

[3] 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 that have both in N'.

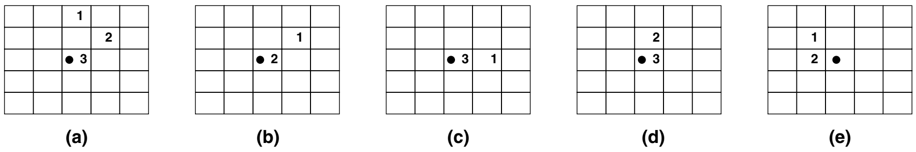

Fig. 2. Snapshots of a pheromone of intensity $I = 3$ changing as FSM-ant $\mathcal{F}$ (the dot) moves. All snapshots have $\mathcal{F}$ on the center tile; unlabeled tiles have level 0. (a) $\mathcal{F}$ has deposited a maximum dose of pheromone on each tile that it has reached via a 2-step SE-SW path; note that the pheromone has begun to "evaporate" on the tiles that $\mathcal{F}$ has left. (b) $\mathcal{F}$ stands still for one time-step and deposits no pheromone. (c) $\mathcal{F}$ moves W and deposits a maximum dose of pheromone. (d) $\mathcal{F}$ moves S and deposits a maximum dose of pheromone. (e) $\mathcal{F}$ moves E and does not deposit any pheromone.

pheromone (cf. [4])—a digital realization of real ants' volatile organic compounds; each value within $[0, I_i]$ is an *intensity level* of pheromone i. The number c and the ranges $[0, I_j]_{j=1}^{c}$ are characteristics of a specific realization of the model. The volatility of real 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.

B. Computers and programs. Each ant contains a computer that endows it with "intelligence." Each computer possesses I/O ports that allow it to communicate with the outside world and with computers on adjacent tiles. In a single "step," a computer can:

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

As discussed earlier, we have embedded computers function as identical copies of a fixed, specialized *finite-state machine* (*FSM*) $\mathcal{F}$; cf. [10]. We specify FSMs' state-transition functions in an algorithmically friendly fashion, as programs that are *finite sets of case statements;*[4] an FSM $\mathcal{F}$ is, thus, specified as follows:

– $\mathcal{F}$ has s states, named $\text{LABEL}_1, \ldots, \text{LABEL}_s$.
– $\mathcal{F}$ responds to a fixed repertoire of *inputs,* each INPUT_i being a combination of:
  - the messages that $\mathcal{F}$ receives from neighboring FSMs and/or 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 pheromones that are present on the current tile.
– $\mathcal{F}$ responds to the current input by
  - emitting an *output* from a repertoire, each OUTPUT_k being a combination of:
    * the messages that $\mathcal{F}$ sends to neighboring FSMs;
    * pheromone-related actions: deposit a type-h pheromone at intensity $I \leq I_h$, enhance a type-j pheromone, increasing its intensity to $I \leq I_j$;
    * "food"-related actions: pick up and carry the item on the current tile, deposit the item that $\mathcal{F}$ is currently carrying;

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

  - $*$ stand still or move to an appropriate neighboring tile (one that will not cause $\mathcal{F}$ to "fall off" $\mathcal{M}_n$).
- changing state (by specifying the next case statement to execute).

An FSM is thus specified in a format similar to the following:

```
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$_{s,m}$
```

We specify algorithms in English—aiming to supply enough detail to make it clear how to craft finite-state programs that implement the algorithms.

2.2 The *Parking Problem* for Ants

The parking problem has each ant $\mathcal{F}$ move as close as possible to the corner tile of $\mathcal{M}_n$ that is closest to $\mathcal{F}$ when it receives the command PARK (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). Formally: *A configuration of ants solves the parking problem for $\mathcal{Q}_{SW}$ precisely if it minimizes the* parking potential function

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

This simply specified, yet algorithmically nontrivial, path-planning problem is a good vehicle for studying what ants can determine about the "floor" without "counting."

3 Single Ants and Parking

3.1 The Simplified Framework for Single Ants

The input and output repertoires of this section's FSMs need not contain pheromone levels, because pheromones do not enhance the path-planning power of a single ant.

Proposition 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 (Sketch). We eliminate pheromones one at a time, so we may assume that $\mathcal{F}$ uses a single pheromone that it deposits with intensity levels from the set $[0, I]$. The pheromone-less FSM $\mathcal{F}'$ "carries around" (in finite-state memory) a *map* that specifies all necessary information about the positions and intensities of deposits of $\mathcal{F}$'s pheromone. For $\mathcal{F}'$ to exist, the map must be: (a) "small"—with size independent of n—and (b) easily updated as $\mathcal{F}$'s steps are simulated.

Map size. 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 $(2I-1) \times (2I-1)$ mesh centered at $\mathcal{F}$'s current tile. Because $\mathcal{F}$ is the only FSM, at most one tile of the map contains the integer I (a maximum level of the pheromone at this step), at most one contains the integer $I-1$ (a maximum level one step ago), ..., at most one contains the integer 1 (a maximum level $I-1$ steps ago). Fig. 2 displays a sample map, with four sample one-step updates.

Updating the map. Because of a map's restricted size and contents, there are fewer than $1 + \prod_{j=0}^{I-1}((2I-1)^2 - j)$ distinct maps (even ignoring the necessary adjacency of tiles that contain the integers k and $k-1$). $\mathcal{F}'$ can, therefore, carry the set of all possible maps in its finite-state memory, with the then-current map clearly "tagged." Thus, $\mathcal{F}'$ has finitely many states as long as $\mathcal{F}$ does. The state-transition function of $\mathcal{F}'$ augments $\mathcal{F}$'s by updating each state's map-component while emulating $\mathcal{F}$'s state change. □

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

Theorem 1. *One cannot design an FSM-ant that successfully parks when started on an arbitrary tile of (even the one-dimensional version of) $\mathcal{M}_n$.*

Proof (Sketch). To the end of deriving a contradiction, say that we have an FSM $\mathcal{F}$ with state-set Q that can park successfully no matter where we place it in an arbitrarily large mesh $\mathcal{M}_n$. By Proposition 1, we may assume that $\mathcal{F}$ does not use pheromones.

Let us place $\mathcal{F}$ far enough within $\mathcal{M}_n$ that its path to its (allegedly correct) parking corner is longer than $q = |Q|$. Let us label each tile along this path with the state that $\mathcal{F}$ is in *the last time it leaves the tile.* Because $\mathcal{F}$ starts out far from its parking corner, there must be tiles along the path that are labeled with the same state, $s \in Q$, despite the fact that one tile is farther from $\mathcal{F}$'s parking corner than the other. (In Fig. 3(left), $\mathcal{M}_n$'s NE corner is depicted as $\mathcal{F}$'s parking corner.) Let us now lengthen $\mathcal{M}_n$'s side-length—i.e., increase n—by "cutting and splicing" portions of $\mathcal{M}_n$ that begin and end with state-label s, as in Fig. 3(right). Because $\mathcal{F}$ is deterministic, it cannot "distinguish" between two tiles that have the same state-label. This means that $\mathcal{F}$'s ultimate behavior will be the same in the original and stretched versions of $\mathcal{M}_n$: it will end its journey in both meshes at the NE corner, despite the fact that the "cut and splice" operation has lengthened $\mathcal{F}$'s parking path. However, if we perform the "cut and splice" operation enough times, we can make $\mathcal{F}$'s original tile as far as we want from its parking corner. In particular, we can make this distance greater than the distance between $\mathcal{F}$'s original tile and some other corner of (the stretched) $\mathcal{M}_n$. Once this happens, $\mathcal{F}$ is no longer parking successfully. The theorem follows. □

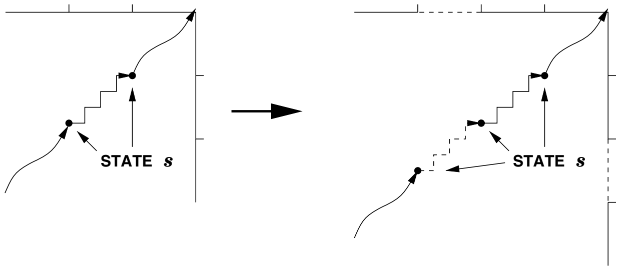

Fig. 3. Illustrating the "cut and splice" operation of Theorem 1

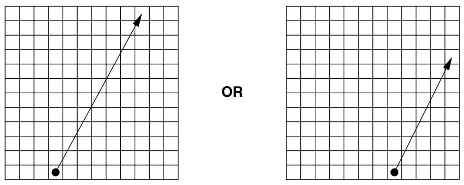

Fig. 4. The two possible edge-terminating trajectories for a single ant

3.3 Single-Ant Configurations That Can Park

Theorem 2. *A single FSM-ant $\mathcal{F}$ can park in time $O(n)$ when started on an arbitrary* edge-tile *of $\mathcal{M}_n$.*

Proof (Sketch). With no loss of generality, let $\mathcal{F}$ begin on $\mathcal{M}_n$'s bottom edge, at tile $\langle 0, k\rangle$. Clearly, $\mathcal{F}$'s target parking tile is either $\langle 0, 0\rangle$ or $\langle 0, n-1\rangle$. To decide between these alternatives, $\mathcal{F}$ begins a $60°$ northeasterly walk from $\langle 0, k\rangle$, i.e., a walk consisting of "Knight's-move supersteps" of the form $(0, +1)$, $(+1, 0)$, $(+1, 0)$ (see Fig. 4), continuing until it encounters an edge or a corner of $\mathcal{M}_n$. Note that after s "supersteps," $\mathcal{F}$ has moved from $\langle 0, k\rangle$ to $\langle 2s, k+s\rangle$. Consequently, if $\mathcal{F}$'s walk terminates in:

- $\mathcal{M}_n$'s right edge, then $\mathcal{F}$'s parking tile is $\langle 0, n-1\rangle$, because $2s < n-1 \le k+s$;
- $\mathcal{M}_n$'s top edge or NE corner, then $\mathcal{F}$'s parking tile is $\langle 0, 0\rangle$, because $k+s \le n-1 \le 2s$. □

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 one simple restriction enables successful parking. Two ants are *adjacent* if they reside on tiles that share an edge or a corner.

Theorem 3. *Any collection of ants that contains two designated adjacent ants can park in $O(n^2)$ synchronous steps.*

Proof (Sketch). We focus, in turn, on 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 (1).

4.1 Quadrant Determination with the Help of Adjacent Ants

Lemma 1. *Any collection of ants that contains two designated adjacent ants can determine their home quadrants in $O(n^2)$ synchronous steps.*

We address the problem of quadrant determination in three parts. Section 4.1.A presents an algorithm that allows *two* adjacent ants to park. Because this algorithm is a bit complicated, we present in Section 4.1.B a much simpler algorithm that allows *three* adjacent ants to park. We then show in Section 4.1.C how two adjacent ants can act as "shepherds" to help any collection of ants determine their home quadrants.

A. Two adjacent ants can park. The following algorithm was developed in collaboration with Olivier Beaumont. The essence of the algorithm is depicted in Fig. 5.

Say, for definiteness, that $\mathcal{M}_n$ contains two horizontally adjacent ants: A_L on tile $\langle x, y\rangle$ and A_R on tile $\langle x, y+1\rangle$. (This assumption loses no generality, because the ants can remember their true initial configuration in finite-state memory, then move into the left-right configuration, and finally compute the adjustments necessary to make the correct determination about their home quadrants.) Our algorithm has two phases.

Determining home wedges. A_L performs two roundtrip walks within $\mathcal{M}_n$ from its initial tile, one at $45°$ (a sequence of $(+1, +1)$ moves, then a sequence of $(-1, -1)$ moves) and one at $-45°$ (a sequence of $(+1, -1)$ moves, then a sequence of $(-1, +1)$ moves); see Fig. 5(a). The termini of the outward walks determine ants' home wedges:

$45°$ walk terminus	$-45°$ 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 home quadrants. A_L and A_R use the knowledge of their home wedge(s) to determine their home *quadrant(s)*, via one of the following subprocedures.

• *The ants did* not *both begin in either $\mathcal{W}_E$ or $\mathcal{W}_W$.* In this case, A_L and A_R move in lockstep to the bottom edge of $\mathcal{M}_n$ (see Fig. 5(b)). They set out thence in lockstep on independent diagonal *roundtrip* walks, A_L at an angle of $-45°$ (via $(+1, -1)$ moves)

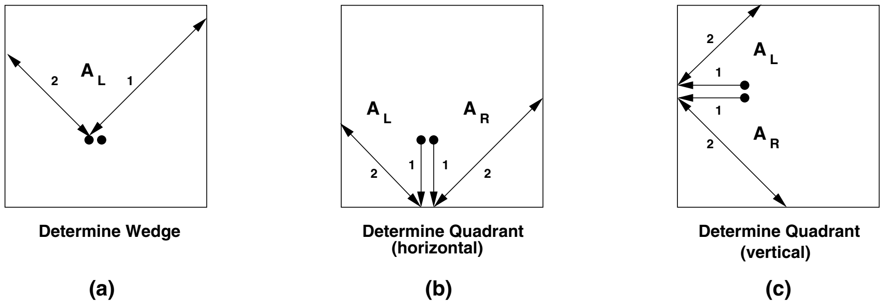
A_L
1
2
Determine Wedge
(a)
A_L
A_R
1
1
2
2
Determine Quadrant (horizontal)
(b)
A_L
A_R
1
2
1
2
Determine Quadrant (vertical)
(c)

Fig. 5. Quadrant determination for two adjacent ants: (a) wedge determination; (b) quadrant determination, horizontal version; (c) quadrant determination, vertical version

and A_{R} at an angle of $45°$ (via $(+1, +1)$ moves). When an ant returns to its original tile on the bottom edge, it checks for the other ant's presence in the appropriate adjacent tile, to determine the outcome of their roundtrip "race." A discrete set of "race" outcomes provides the ants bounds on the columns where each began to park: Did the "race" end in a tie? Did one ant win by exactly one step? by more than one step? These bounds allow A_{L} and A_{R} to determine their home quadrants.

• *The ants both began in either $\mathcal{W}_E$ or $\mathcal{W}_W$.* In this case, a "vertical" analogue of the preceding "race" (see Fig. 5(c)) allows A_{L} and A_{R} to determine their home quadrants.

Once A_{L} and A_{R} know their respective home quadrants, they can park greedily.

B. Three adjacent ants can park. Let the ants A_1, A_2, and A_3, be *adjacent* on $\mathcal{M}_n$, in that their adjacency graph is connected. Via local communication, each ant can recognize and remember its initial location relative to the others'. The ants can thereby adjust the result of the following algorithm and determine their respective home quadrants.

Determining home quadrants. *Horizontal placement* (Fig. 6(Left)). The ants align vertically in the top-to-bottom order A_1, A_2, A_3, with A_2 retaining its original tile. (A simple clerical adjustment is needed if A_2 begins on either the top or bottom edge of $\mathcal{M}_n$.) A_1 marches leftward to $\mathcal{M}_n$'s left edge and returns to its pre-march tile; in lockstep, A_3 marches rightward to $\mathcal{M}_n$'s right edge and returns to its pre-march tile.

– If A_1 and A_3 return to A_2 at the same step, or if A_1 returns one step before A_3, then A_2's home is either $\mathcal{Q}_{NW}$ or $\mathcal{Q}_{SW}$. A_1 and A_3 can now determine their home quadrants by noting how they moved in order to achieve the vertical alignment.

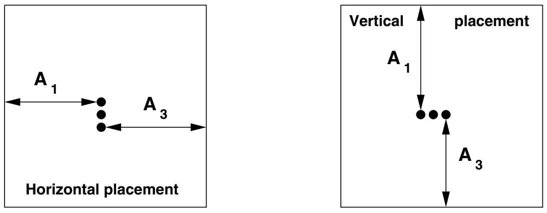

Fig. 6. The essence of the parking algorithm for three adjacent ants: (Left) Determining each ant's horizontal placement. (Right) Determining each ant's vertical placement.

– If A_1 returns to A_2 two or more steps before A_3, then all three ants have $\mathcal{Q}_{NW}$ or $\mathcal{Q}_{SW}$ as home.

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

Vertical placement (Fig. 6(Right)). This determination is achieved by "rotating the horizontal-placement algorithm by 90°." Details are left to the reader.

After horizontal and vertical placement, each ant knows its home quadrant, hence can park as specified in Section 4.2.

C. Two adjacent ants acting as shepherds. All ants in any collection that contains two designated adjacent ants are able to determine their respective home quadrants—with the help of the two adjacent ants. As we verify this, we encounter instances of ants blocking the intended paths of other ants. We resolve this problem by having conflicting ants *switch roles*—which is possible because all ants are identical. If ant A is blocking ant B, then A "becomes" B and continues B's blocked trajectory; and B "becomes" A and moves onto the tile that A has relinquished (by "becoming" B).

We have $m \geq 2$ ants, $A_1, \ldots, A_m$, with two designated adjacent ants—without loss of generality, A_1 and A_2. We present a three-phase algorithm that takes $O(n^2)$ synchronous steps and that has A_1 and A_2 act as *shepherds* to help all other ants determine their home quadrants.

Phase 1: A_1 and A_2 determine their home quadrant(s), using the algorithm of Section 4.1.A. They remember this information, which they will use to park in phase 3.

Phase 2: A_1 and A_2 help other ants determine their home quadrants, via four subphases.

Subphase a: A_1 and A_2 distinguish east from west. A_1 and A_2 head to corner SW. A_1 determines the parity of n via a roundtrip from tile $\langle 0, 0\rangle$ to tile $\langle 0, n-1\rangle$ and back.

If n is even, then:

– A_2 moves one tile eastward at every time-step until it reaches $\mathcal{M}_n$'s right edge. It then *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 are adjacent: A_1 on tile $\langle 0, \frac{1}{2}n - 1\rangle$, A_2 on tile $\langle 0, \frac{1}{2}n\rangle$.

A_1 and A_2 thus determine the midpoint of $\mathcal{M}_n$'s bottom row (hence, of every row):

– A_2's trajectory, $\langle 0, 0\rangle \rightsquigarrow \langle 0, n-1\rangle \rightsquigarrow \langle 0, \frac{1}{2}n\rangle$ takes $\frac{3}{2}n - 2$ time-steps.
– A_1's trajectory $\langle 0, 0\rangle \rightsquigarrow \langle 0, \frac{1}{2}n - 1\rangle$ takes $\frac{1}{2}n - 3$ steps. Because it starts one step later than A_2 does, A_1 arrives at $\langle 0, \frac{1}{2}n - 1\rangle$ after $\frac{3}{2}n - 2$ time-steps.

If n is odd, then:

– A_2 moves one tile eastward at 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 are adjacent: A_1 on $\langle 0, \lceil\frac{1}{2}n\rceil - 1\rangle$, A_2 on $\langle 0, \lceil\frac{1}{2}n\rceil\rangle$.

A_1 and A_2 thus determine the midpoint of $\mathcal{M}_n$'s bottom row (hence, of every row):

– A_2's trajectory, $\langle 0, 0\rangle \rightsquigarrow \langle 0, n-1\rangle \rightsquigarrow \langle 0, \lceil\frac{1}{2}n\rceil\rangle$, takes $3\lceil\frac{1}{2}n\rceil - 4$ time-steps.
– A_1's trajectory, $\langle 0, 0\rangle \rightsquigarrow \langle 0, \lceil\frac{1}{2}n\rceil - 1\rangle$, takes $3\lceil\frac{1}{2}n\rceil - 3$ steps. Because it starts one step later than A_2 does, A_1 arrive at $\langle 0, \lceil\frac{1}{2}n\rceil - 1\rangle$ after $3\lceil\frac{1}{2}n\rceil - 4$ time-steps.

Subphase b: *A_1 and A_2 identify easterners, westerners.* A_1 walks through the western half of $\mathcal{M}_n$, column by column, informing each encountered ant that it resides in either $\mathcal{Q}_{NW}$ or $\mathcal{Q}_{SW}$. Simultaneously, A_2 does the symmetric task in the eastern half of $\mathcal{M}_n$, informing each encountered ant that it resides in either $\mathcal{Q}_{NE}$ or $\mathcal{Q}_{SE}$.

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

Subphase c: *A_1 and A_2 distinguish north from south,* via a process analogous to that of Subphase a.

Subphase d: *A_1 and A_2 identify northerners and southerners,* via a process analogous to that of Subphase b.

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; A_1 and A_2 await the end of Phase 2, having learned their home quadrants in Phase 1.

4.2 Completing the Parking Process

We describe finally how ants that know their home quadrants travel to their parking corner and configure themselves optimally compactly within that corner.

Traveling to the corner. Each ant follows a two-stage trajectory. It proceeds horizontally to the left edge of $\mathcal{M}_n$. Having achieved that edge, it proceeds vertically toward its parking corner. An ant that is proceeding horizontally: (a) moves only to an empty tile; if none exists, then it waits; (b) yields to an ant that is already proceeding vertically.

Organizing within the corner. We describe this process in terms of $\mathcal{Q}_{SW}$; other quadrants are treated analogously; see Fig. 7. The first ant that reaches its parking corner (it may have started there) becomes an *usher*. It directs vertically arriving ants into the next adjacent downward diagonal; thus-directed ants proceed down this diagonal (k) and up the next higher one ($k + 1$)—moving only when there is an ant behind them that wants them to move! When an ant reaches the top of the upward diagonal, it "defrocks" the current usher and becomes an usher itself (via a message relayed by its lower neighbor). The corner of $\mathcal{M}_n$ thus gets filled in compactly, two diagonals at a time. One shows that this manner of filling the corner organizes the ants into a configuration that minimizes the parking potential function (1).

This completes the parking algorithm and the proof of Theorem 3. □

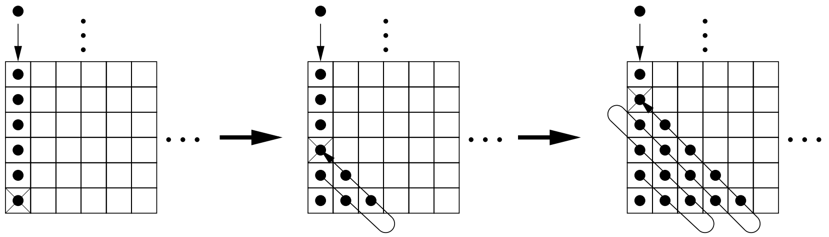

Fig. 7. Three stages in the snaked parking trajectory within $\mathcal{Q}_{SW}$; X-ed cells contain "ushers"

5 Conclusions

We have reported on progress in understanding the algorithmic strengths and weaknesses of ant-inspired robots within geographically constrained rectangular “floors.” We have obtained this understanding via the simple path-planning problem we call *parking:* have ants configure themselves in a maximally compact manner within their nearest corner of the floor. 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 shown that—even in a one-dimensional world—a single ant cannot park.

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. [10].

Where do we go from here? Most obviously, we want to solve the parking problem definitively, by characterizing which initial configurations enable parking and which do not. It would also be valuable to understand the capabilities of ant-inspired robots within the context of other significant tasks that involve path planning [1,6,7,8,9], including tasks that involve finding and transporting “food” and avoiding obstacles (as in [8,9]), as well those that involve interactions among multiple genres of ants.

Acknowledgments. This research was supported in part by US NSF Grant CNS-0905399. Thanks to O. Beaumont and O. Brock and the anonymous reviewers for insights and suggestions.

References

1. Chen, L., Xu, X., Chen, Y., He, P.: A novel ant clustering algorithm based on Cellular automata. In: IEEE/WIC/ACM Int’l. Conf. Intelligent Agent Technology (2004)
2. 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)
3. Folino, G., Mendicino, G., Senatore, A., Spezzano, G., Straface, S.: A model based on Cellular automata for the parallel simulation of 3D unsaturated flow. Parallel Computing 32, 357–376 (2006)
4. Geer, D.: Small robots team up to tackle large tasks. IEEE Distributed Systems Online 6(12) (2005)
5. Goles, E., Martinez, S. (eds.): Cellular Automata and Complex Systems. Kluwer, Amsterdam (1999)
6. http://www.kivasystems.com/
7. Marchese, F.: Cellular automata in robot path planning. In: EUROBOT 1996, pp. 116–125 (1996)
8. Rosenberg, A.L.: Cellular ANTomata. In: Stojmenovic, I., Thulasiram, R.K., Yang, L.T., Jia, W., Guo, M., de Mello, R.F. (eds.) ISPA 2007. LNCS, vol. 4742, pp. 78–90. Springer, Heidelberg (2007)
9. Rosenberg, A.L.: Cellular ANTomata: food-finding and maze-threading. In: 37th Int’l. Conf. on Parallel Processing (2008)
10. Rosenberg, A.L.: The Pillars of Computation Theory: State, Encoding, Nondeterminism. Universitext Series. Springer, Heidelberg (2009)
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)