Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 16, no. 2, pp. 483–506 (2012)
DOI: 10.7155/jgaa.00271

The Parking Problem for Finite-State Robots

Arnold L. Rosenberg

Colorado State University and Northeastern University

Abstract

This paper is a step toward understanding the algorithmic concomitants of modeling robots as mobile finite-state machines (FSMs, for short) that travel within square two-dimensional meshes (which abstract the floors of laboratories or factories or warehouses). We study the ability of FSMs to *scalably* perform a simple path-planning task called *parking*, within fixed square meshes of arbitrary sizes. This task: (1) has each FSM head for its nearest corner of the mesh and (2) has all FSMs within a corner organize into a maximally compact formation (one that minimizes a compactness-measuring potential function). The problem thus requires FSMs to know "where they are" within a mesh, specifically which quadrant they reside in. Indeed, *quadrant determination* is the central technical issue in enabling FSMs to park. Many initial configurations of FSMs *can* park, including: (*a*) a single FSM situated initially along an edge of the mesh; (*b*) any assemblage of FSMs that begins with two designated adjacent FSMs. These configurations can park even without using (digital analogues of) pheromones, an algorithmic aid advocated by some who use FSMs to model ant-inspired robots. In contrast, a single FSM in the interior of (even a one-dimensional) mesh *cannot* park, even with the help of (volatile digital) pheromones.

Key words: FSM-robots; Finite-state machines; Searching and exploration in a mesh

Submitted:	Reviewed:	Revised:	Accepted:	Final:
September 2011	January 2012	March 2012	July 2012	July 2012

Published: August 2012

Article type: Regular paper
Communicated by: P. Ferragina

This research was supported in part by US NSF Grant CNS-0905399. A portion of this paper appeared at *Euro-Par'10;* see [28].

E-mail address: rsnbrg@cs.umass.edu (Arnold L. Rosenberg)

1 Introduction

This paper studies the power of mobile finite-state machines (FSMs, for short) to *scalably* perform a simple path-planning task called *parking*, within square two-dimensional meshes of arbitrary sizes. FSMs are viewed as abstractions of simple mobile robots, and the meshes they navigate are viewed as abstractions of bounded regions such as the floors of factories or warehouses or laboratories. The task of parking: (1) has each FSM head for its nearest corner of the mesh and (2) has all FSMs within a corner organize into a maximally compact formation—one that minimizes the average distance of an FSM from its target corner. The Parking Problem thus requires FSMs to discover "where they are" within a mesh, specifically which quadrant they reside in. Indeed, *quadrant determination* is the central technical issue in enabling FSMs to park. Our results show that many—but not all—initial configurations of FSMs *can* park successfully within arbitrarily large meshes. Among the configurations that can park are: (*i*) a single FSM that resides initially along an edge of the mesh; (*ii*) any assemblage of FSMs that begins with two designated FSMs that are *adjacent*, i.e., that occupy neighboring tiles of the mesh. The preceding configurations can park even without using (digital analogues of) pheromones, an algorithmic aid advocated by some who study ant-inspired robots; cf. [12, 18, 30]. In contrast, a single FSM that resides initially in the interior of (even a one-dimensional) mesh *cannot* park, even with the help of (volatile digital) pheromones. The road to these results builds on the basic problem of how/whether an FSM can determine which quadrant of the mesh it resides in. Other lessons from our study indicate that: (*a*) The limited exploratory/path-planning ability of a single FSM is sometimes much extended if the FSM can use the edges of a mesh for orientation. Indeed, the edges sometimes enable an FSM to appear to count (to n) without actually counting. (*b*) (Digital) pheromones cannot enhance the power of a single FSM, although they can enable a small FSM (in number of states) to perform a task that would otherwise require an exponentially larger one. Pheromones can enhance the power of a team of FSMs, but we do not need them to enable FSMs to park.

1.1 An Informal Summary of the Paper

FSMs on a mesh. We study mobile robotic computers, modeled as *FSMs*, that function within fixed geographically constrained environments, modeled as *square meshes*. To emphasize the robotic inspiration, we think of the mesh as a *floor* that is tesselated with identical square tiles. We expect FSMs to: • navigate the mesh, while avoiding collisions; • communicate with and sense one another, by "direct contact" (as when real ants meet) and by "time-stamped message passing" (as when real ants deposit pheromones); • assemble in desired locations, in desired configurations.

Our study does not require FSMs to avoid obstacles, discover "food" objects, or convey "food" from one location to another (as in, e.g., [12, 16, 25, 26]). Planned sequels to this study will make such demands of FSMs.

The Parking Problem. We study a simple, yet algorithmically nontrivial, path-planning task, *parking,* that: (1) has each FSM head to the nearest corner of the mesh and (2) has all FSMs within a corner organize into a maximally compact formation that minimizes the average distance of an FSM from its target corner; cf. the compactness-measuring potential function in Section 2.4. While we have not yet characterized which configurations of FSMs can park successfully, we report on progress toward this goal:

- Even without using (digital analogues of) pheromones, *many initial configurations of FSMs* can *park*. Examples: (1) *a single FSM that starts along an edge of the mesh* (Theorem 1); (2) *any collection of FSMs containing two distinguished ones that reside initially on adjacent tiles* (Theorem 4).
- In contrast: *A single FSM can generally* not *park, even on a* 1-dimensional *mesh and even with the help of (volatile digital) pheromones* (Theorem 1).

Whence the Parking Problem? Our interest in the Parking Problem has two motivations, one automata theoretic and one robotic (cf. Section 2.3). From the automata-theoretic perspective, the Parking Problem is an instance of the question "What can FSMs discover about where they reside within a mesh?" Specifically, the Problem requires FSMs to discover which quadrant they reside in. Along complementary lines, we are currently preparing a companion paper [29] that studies an instance of the question "How well can FSMs discover designated target tiles within meshes?" From the robotic perspective, one finds commercial systems that employ mobile robots within warehouses [16]: the robots collect desired items and convey them to human dispatchers. A video describing this system left me wondering: where do they "store" idle robots? Having robots assemble in mesh corners came to mind as a viable way to keep idle robots "out of the way" until they are needed next. The Parking Problem combines the essence of these two motivations.

2 Technical Background and Related Work

2.1 Technical Background

Our formal model of *FSM-robots* (*FSM*s, for short) is obtained by augmenting the capabilities of standard finite-state machines (see, e.g., [27] for formal details) with the ability to travel around square *meshes* of *tiles*

Meshes and their tiles. We index the n^2 tiles of the $n \times n$ mesh $\mathcal{M}_n$ by the set[1] $[0, n-1] \times [0, n-1]$; see Fig. 1(left). Tile $\langle i, j \rangle$ of $\mathcal{M}_n$ is: • a *corner* tile if $i, j \in \{0, n-1\}$; each corner tile has 3 neighbors; • a *bottom* (resp., *top*) tile if $i = 0$ (resp., $i = n-1$) and $j \in [1, n-2]$; a *left* (resp., *right*) tile if $j = 0$ (resp., $j = n-1$) and $i \in [1, n-2]$; these four are collectively *(internal) edge* tiles; each has 5 neighbors; • an *internal* tile if $i, j \in [1, n-2]$; each internal tile has 8

[1]For positive integers i and $j \geq i$, $[i, j] \overset{\text{def}}{=} \{i, i+1, \ldots, j\}$.

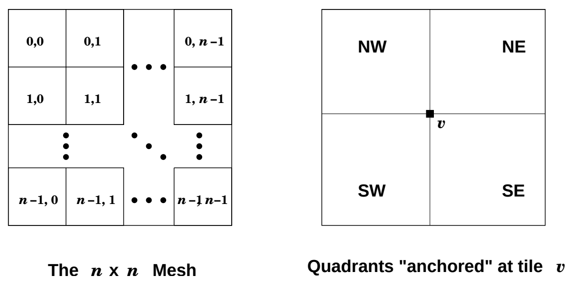


Figure 1: (left) The $n \times n$ mesh $\mathcal{M}_n$; (right) $\mathcal{M}_n$ partitioned into the four *quadrants* determined by *anchor tile* v.

neighbors. *Every edge of every tile v is labeled to indicate which of v's potential neighbors actually exists.* (This enables FSMs to avoid "falling off" $\mathcal{M}_n$).

$\mathcal{M}_n$'s four *quadrants* are determined by lines that cross at an *anchor* tile $\langle i, j\rangle$ and are perpendicular to $\mathcal{M}_n$'s edges; see Fig. 1(right). $\mathcal{M}_n$'s "standard" quadrants—which are anchored at $\mathcal{M}_n$'s "center" tile $\langle \lfloor \frac{1}{2}(n-1)\rfloor, \lfloor \frac{1}{2}(n-1)\rfloor\rangle$, hence are as close to equal in size (i.e., numbers of tiles) as the parity of n allows—comprise the following sets of tiles.

Quadrant	Name	Tile-set
SOUTHWEST	$\mathcal{Q}_{SW}$	$\{\langle x, y\rangle \mid x \geq \lfloor \frac{1}{2}(n-1)\rfloor;\ \ y \leq \lfloor \frac{1}{2}(n-1)\rfloor\}$
NORTHWEST	$\mathcal{Q}_{NW}$	$\{\langle x, y\rangle \mid x < \lfloor \frac{1}{2}(n-1)\rfloor;\ \ y \leq \lfloor \frac{1}{2}(n-1)\rfloor\}$
SOUTHEAST	$\mathcal{Q}_{SE}$	$\{\langle x, y\rangle \mid x \geq \lfloor \frac{1}{2}(n-1)\rfloor;\ \ y < \lfloor \frac{1}{2}(n-1)\rfloor\}$
NORTHEAST	$\mathcal{Q}_{NE}$	$\{\langle x, y\rangle \mid x < \lfloor \frac{1}{2}(n-1)\rfloor;\ \ y \leq \lfloor \frac{1}{2}(n-1)\rfloor\}$

Rounding ensures that each tile of $\mathcal{M}_n$ resides in a unique quadrant.

A single FSM on $\mathcal{M}_n$. At any moment, an FSM $\mathcal{F}$ occupies a single tile of $\mathcal{M}_n$, sharing that tile with no other FSM. At each step, $\mathcal{F}$ can move to any of the (up-to) eight *King's-move*[2] neighbors of its current tile, in any of the eight compass directions: N, NE, E, SE, S, SW, W, NW. (Clerical extensions extend $\mathcal{F}$'s move repertoire to include any *fixed finite* set of atomic moves, each taking $\mathcal{F}$ from a tile $\langle i, j\rangle$ to a tile of the form $\langle i \pm a, j \pm b\rangle$. Note that, as $\mathcal{F}$ plans its next move, it *must* consider the label of its current tile—hence must be aware of residing on an edge tile or corner tile (to avoid "falling off"). But, being an FSM, $\mathcal{F}$ *cannot* exploit any knowledge of the size-parameter n of the mesh it resides in—except for "finite-state" knowledge such as the parity of n.

Multiple FSMs on $\mathcal{M}_n$. A team of FSMs on $\mathcal{M}_n$ can be activated (from the outside world) simultaneously. Distinct FSMs on $\mathcal{M}_n$ operate synchronously, i.e., can follow trajectories *in lockstep.* FSMs that reside on neighboring tiles are aware of each other and can pass messages—such as "I AM HERE." Such simple messages often enable one FSM to act as an "usher" or a "shepherd" for

[2]We employ King's-moves for convenience. We could easily make do with the more spartan *NEWS*-moves, N, E, W, S, or we could add more atomic moves, such as the 16 *knight's* moves ($\langle N, NW\rangle$ and $\langle E, SE\rangle$ and their kin). FSM programs clearly grow with the number of neighbors tiles have.

other FSMs. Each FSM's moves on $\mathcal{M}_n$ are tightly orchestrated. Specifically, an FSM attempts to move in direction:

N	only at steps	$t \equiv 0 \bmod 8$;	NE	only at steps	$t \equiv 1 \bmod 8$;
E	only at steps	$t \equiv 2 \bmod 8$;	SE	only at steps	$t \equiv 3 \bmod 8$;
S	only at steps	$t \equiv 4 \bmod 8$;	SW	only at steps	$t \equiv 5 \bmod 8$;
W	only at steps	$t \equiv 6 \bmod 8$	NW	only at steps	$t \equiv 7 \bmod 8$

(A repertoire of k atomic moves would require a modulus of k.) This orchestration means that *FSMs need never collide!* If several FSMs want to enter a tile v from (perforce distinct) neighboring tiles, then one of the FSMs will have permission to enter v before the others—so all FSMs will learn about the conflict before a collision occurs.

2.2 Algorithmic Standards and Simplifications

Algorithms are finite-state. Each is specified by a single *finite-state program* which all robots execute in SPMD[3] mode. Such programs, as described in [27] and employed in "finite-state" programming systems such as *CARPET* [32], have the form

```
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}
```

with statement labels playing the role of states. Note in particular that *all FSMs are identical;* none has a "name" that renders it unique.

Algorithms are scalable: They work on meshes of arbitrary sizes. In particular, an FSM $\mathcal{F}$ cannot exploit information about the size of a mesh $\mathcal{M}_n$, treating its side-length n as an *unknown.* $\mathcal{F}$ can, however, learn "finite-state" properties of n, such as its parity.

Algorithms are decentralized but synchronous. Once started, FSMs operate autonomously, but their independent clocks tick at the same rate—so that distinct FSMs can follow trajectories *in lockstep.* This assumption is no less realistic than the analogous assumption with synchronous-start human endeavors such as military maneuvers.

These guidelines are often violated in implementations, as in [9, 12, 16, 18, 30], where practical simplicity overshadows algorithmic simplicity.

We specify algorithms in English, trying to tailor the amount of detail to the complexity of the specification. Our goal is to make it clear how to craft a realizing finite-state program. Perhaps the feature of finite-state programs that we exploit the most is their sequential composability.

[3] "SPMD" stands for "Single-Program-Multiple-Data," a "relaxed" analogue of SIMD.

2.3 Related Work

Our study combines concepts and tools from a number of complementary bodies of literature that span several decades. The literature on automata theory and its applications contains studies exemplified by [3, 5, 7, 10, 23] that focus on the (in)ability of FSMs to explore graphs with goals such as finding "entrance"-to-"exit" paths or exhaustively visiting all nodes or all edges. Other studies, exemplified by [14, 17, 20, 22, 31], focus on algorithms that enable FSMs that populate the cells of (possibly multidimensional) meshes to tightly synchronize; such arrays of FSMs are, of course, *cellular automata,* a model dating back a half century [33], yet still of interest today [13, 32, 34, 35]. The robotics literature contains numerous studies—see, e.g., [1, 2, 12, 18, 30]—that explore the use of ants as a metaphor for simple robots that collaborate to accomplish complex tasks; the use of "virtual pheromones" within this metaphor is particularly interesting; we briefly studied this topic in [28]. (The ant metaphor is discussed in an entertaining way in [15].) Cellular automata appear also within robotic applications of automata-theoretic concepts [19, 25, 26]. Notably, the robotic branch of this literature does not consist just of theoretical studies of the model, containing also application- and implementation-oriented studies [6, 12, 16, 30, 32]. The current study melds the goals of the automata-theoretic and robotic studies by studying FSMs that traverse two-dimensional meshes, with goals more closely motivated by robotics than automata theory. The closest relative to our study is [29], which studies the (in)ability of FSMs to find, for arbitrary fixed rationals $0 < \varphi, \psi < 1$, the tile $\langle \lfloor \varphi(n-1) \rfloor, \lfloor \psi(n-1) \rfloor \rangle$ in arbitrary meshes $\mathcal{M}_n$. This simple, yet significant, problem for mobile robots complements the question underlying the Parking Problem– "What can FSMs discover about where they reside within a mesh?"—with the question "Can FSMs discover designated target tiles within meshes?" The results in [29] parallel those here: a single FSM has quite limited path-planning/exploration ability, while teams of two or more FSMs have such ability to a significantly greater extent.

2.4 The ***Parking Problem*** for FSMs

To simplify exposition, we restrict attention to the Parking Problem for $\mathcal{M}_n$'s northwest quadrant $\mathcal{Q}_{NW}$; clerical changes accommodate the other quadrants. To be formal, the kth *diagonal* of $\mathcal{Q}_{NW}$ is the following set of tiles of $\mathcal{M}_n$:

$$\Delta_k \;=\; \{\langle i, j\rangle \mid i + j = k + 1\}. \tag{1}$$

We then have: *A configuration of FSMs solves the Parking Problem for $\mathcal{Q}_{NW}$ if and only if it minimizes the parking potential function*

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

This simple, yet algorithmically nontrivial, path-planning problem lends significant insights into what FSMs can determine about meshes without counting.

3 Toward Understanding Single FSMs

We present three results that enhance our understanding of FSMs as models for mobile robots. The first result exposes the fact that single FSMs cannot park successfully within large meshes (Section 3.1). The second two results shed light on the use of (digital) pheromones to the FSM model, as suggested in, e.g., [12, 18, 30] (Sections 3.2 and 3.3).

3.1 A Single FSM Cannot Park

Theorem 1 *No FSM can successfully park when started on an arbitrary tile of (even the one-dimensional version of) arbitrarily large meshes.*

The proof of the theorem formalizes the insight that a single FSM "gets lost" in the interior of any sufficiently large mesh $\mathcal{M}_n$, hence cannot determine which quadrant of $\mathcal{M}_n$ it resides in. This insight adds to the list of the limitations of FSMs as they strive to explore unbounded domains; cf. [23]. We define the "interior" of $\mathcal{M}_n$ as follows. For any rational α in the range $1/n < \alpha < 1/2$, the *α-interior of $\mathcal{M}_n$* is the submesh with corners at tiles $\langle \lfloor \alpha(n-1) \rfloor, \lfloor \alpha(n-1) \rfloor \rangle$, $\langle \lfloor \alpha(n-1) \rfloor, (n-1) - \lfloor \alpha(n-1) \rfloor \rangle$, $\langle (n-1) - \lfloor \alpha(n-1) \rfloor, \lfloor \alpha(n-1) \rfloor \rangle$, $\langle (n-1) - \lfloor \alpha(n-1) \rfloor, (n-1) - \lfloor \alpha(n-1) \rfloor \rangle$. We say that an α-interior of $\mathcal{M}_n$ is *landlocked* for a q-state FSM $\mathcal{F}$ if $q < \alpha(n-1)$. We define the term "get lost" implicitly, via the coming argument, because its exact definition depends on the path-planning problem that the FSM is trying to solve. The reader should understand the term from the argument.

Let $\mathcal{F}$ be a q-state FSM that is trying to navigate within $\mathcal{M}_n$ from a designated Start Tile to a designated Halt Tile, where at least one of these tiles is landlocked for $\mathcal{F}$. Say for definiteness that *the Start Tile is landlocked*, because this is the problematic situation for the parking problem;[4] a symmetric argument will handle the case of a landlocked Halt Tile. Let us examine the tiles that $\mathcal{F}$ visits during the first $q+1$ steps of its journey from the Start Tile to the Halt Tile, noting that it may visit some of these more than once. We label each of these $q+1$ steps with the name of the state that $\mathcal{F}$ was in at that step. We thereby associate these steps with a sequence of $q+1$ state names: $s_0, s_1, \ldots, s_q$. By the Pigeonhole Principle, at least two of the names, say s_a and s_b, must be identical: say $s_a = s_b = s$, as in Fig. 2(middle). Because $\mathcal{F}$ is an FSM, being in a particular state at a given step of its journey embodies $\mathcal{F}$'s only memory of anything that happened before that step. Therefore, each time $\mathcal{F}$ arrives at some tile in state s, it cannot determine if this is the first time that it entered state s during the journey or the second time or .... In particular, if we move $\mathcal{F}$'s Start Tile *closer* to $\mathcal{M}_n$'s boundary (as in Fig. 2(left)) or *farther* from $\mathcal{M}_n$'s boundary (as in Fig. 2(right)), $\mathcal{F}$ cannot distinguish among the three situations in Fig. 2 as it embarks on the dashed portion of its journey. Note that we can force the *extended* trajectory of Fig. 2(right) only if $\mathcal{F}$'s Start Tile is "sufficiently far" into $\mathcal{M}_n$'s interior, meaning at least distance $3q$ from $\mathcal{M}_n$'s boundary. (The

[4]The Halt Tile in the Parking Problem is a corner tile, hence decidedly *not* landlocked.

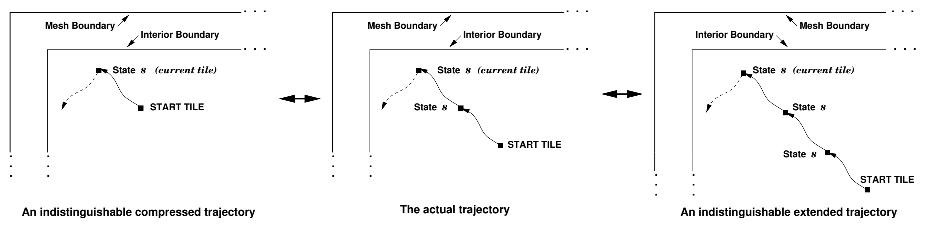


Figure 2: *The "cut and splice" operation. The solid path-segments in the middle subfigure represent the actual first $q+1$ steps as $\mathcal{F}$ leaves its Start Tile. The left and right subfigures indicate alternative path-segments that $\mathcal{F}$ cannot distinguish from the actual ones.*

integer 3 here reflects the three occurrences of state s in Fig. 2(right); a subsequent extension would require that $\mathcal{F}$'s Start Tile be at least distance $4q$ from $\mathcal{M}_n$'s border; and so on.) The claimed repetition must occur because the inputs to $\mathcal{F}$'s state-transitions are the same between successive encounters with state s: $\mathcal{F}$ is the only FSM, it does not employ pheromones, and it never goes near an edge of $\mathcal{M}_n$, so it receives identical stimuli after each occurrence of state s.

Summing up informally, but in a way that is easily formalized, if we start $\mathcal{F}$ sufficiently far from any edge of $\mathcal{M}_n$, then it will not be able to determine its relative distances from $\mathcal{M}_n$'s edges or corners. This insight allows us finally to prove Theorem 1.

Proof of Theorem 1 By the preceding argument: For any FSM $\mathcal{F}$, for all sufficiently large meshes $\mathcal{M}_n$, repeated extensions of $\mathcal{F}$'s trajectory from its Start Tile can change which corner of $\mathcal{M}_n$ is $\mathcal{F}$'s target parking corner, in a way that $\mathcal{F}$ cannot recognize. The reader can easily adapt this argument to a one-dimensional mesh M_n in which $\mathcal{F}$ must find the closer end. □

3.2 Augmenting the Model with Virtual Pheromones

Several sources in the robotics literature—see, e.g., [12, 18, 30]—advocate endowing robots with *virtual pheromones*, a digital realization of real ants' volatile organic compounds. We can accomplish this by endowing each tile of $\mathcal{M}_n$ with a fixed number c of counters, where each counter ℓ can hold an integer in the range $[0, I_\ell]$; each such integer is an *intensity level* of pheromone ℓ. The number c and the ranges $[0, I_j]_{j=1}^{c}$ are characteristics of a specific instance of the model. The volatility of real pheromones is modeled by a schedule of decrements of every pheromone counter, say one unit per step; see Fig. 3. Every computation begins with all tiles having level 0 of every pheromone.

None of our positive results ("Such and such a configurations of FSMs *can* park.") needs virtual pheromones, and none of our negative results ("Such and such a configurations of FSMs *cannot* park.") is helped by virtual pheromones. That said, for completeness, we now hint at the impact of virtual pheromones on our model. In short, pheromones—as modeled above—do not enhance the

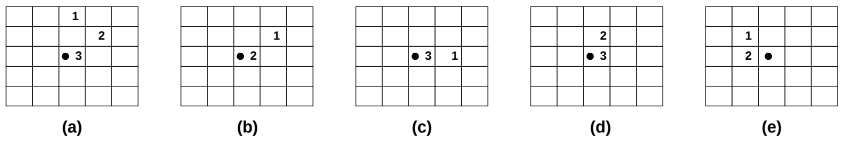


Figure 3: Snapshots of a pheromone of intensity $I = 3$ changing as FSM $\mathcal{F}$ (the dot) moves. All snapshots have $\mathcal{F}$ on the center tile; unlabeled tiles have level 0. (a) $\mathcal{F}$ has deposited a maximum dose of pheromone on each tile that it has reached via a 2-step SE-SW path; 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.

capabilities of single FSMs, but they sometimes enable small FSMs to function as much (in fact exponentially) larger pheromone-less ones. We establish the first claim here and the second in Section 3.3.

Theorem 2 *Given any FSM $\mathcal{F}$ that employs (virtual) 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: We can focus on an FSM $\mathcal{F}$ that uses just one type of pheromone because we eliminate a single type at a time. Say that $\mathcal{F}$'s single pheromone has intensity levels in the set $[0, I]$. We design a pheromone-less FSM $\mathcal{F}'$ with exponentially (in I) more states than $\mathcal{F}$ that emulates $\mathcal{F}$ step by step. $\mathcal{F}'$ "carries around" (in finite-state memory) a *map* that specifies all relevant information about the tiles of $\mathcal{M}_n$ that contain nonzero intensities of $\mathcal{F}$'s pheromone—the tiles' relative locations and the intensities of the pheromone. For $\mathcal{F}'$ to exist, the pheromone map must be: (a) "small"—with size independent of n—and (b) easily updated as $\mathcal{F}'$ emulates successive steps by $\mathcal{F}$.
Map size. The portion of $\mathcal{M}_n$ that could contain nonzero levels of the pheromone is no larger than the "radius"-I submesh of $\mathcal{M}_n$ that $\mathcal{F}$ has visited 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 on $\mathcal{M}_n$, at most one tile of the map contains the integer I (a current maximum level of the pheromone), 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. 3 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. $\mathcal{F}'$'s state-transition function augments $\mathcal{F}$'s by updating each state's map-component while emulating $\mathcal{F}$'s state change. □

Theorem 2 strengthens Theorem 1 to include FSMs that employ pheromones.

3.3 Pheromones Can Enable Exponentially Smaller FSMs

The pheromone-less FSM $\mathcal{F}'$ in our proof of Theorem 2 has a number of states q' that is exponentially larger (in the number of intensity levels I) than the number of states q of the pheromone-using FSM $\mathcal{F}$ that $\mathcal{F}'$ replaces; specifically, q' is roughly $I^{O(I)}q$. It is natural to wonder if this exponential blowup in FSM-size is necessary. We use the Not-So-Close Neighbor Problem to show that the exponential blowup is close to necessary in the worst case. This problem, specialized to the mesh $\mathcal{M}_n$ and the integer k with $0 < k < \log_2 n$, is denoted $\mathsf{NSCN}_{n,k}$ and is specified as follows (using "south" for definiteness only).

The $\mathsf{NSCN}_{n,k}$ Problem: Have FSM $\mathcal{F}$ move 2^k tiles south of its starting tile $\langle i, j\rangle$, to tile $\langle i + 2^k, j\rangle$.

Theorem 3 *Focus on a mesh $\mathcal{M}_n$ and an integer k with $0 < k < \log_2 n$.*

1. *No pheromone-less FSM with fewer than 2^k states can solve the $\mathsf{NSCN}_{n,k}$ Problem within arbitrary meshes $\mathcal{M}_n$.*
2. *The $\mathsf{NSCN}_{n,k}$ problem is always solvable in 2^k steps by a pheromone-less single FSM that has $2^k + 1$ states.*
3. *The $\mathsf{NSCN}_{n,k}$ Problem is always solvable in $O(k2^k)$ steps by a single FSM that has $O(k)$ states and that employs a single type of pheromone with $6k + 1$ intensity levels.*

Proof: *1.* This follows directly from the argument in Section 3.1. If an FSM $\mathcal{F}$ having $q < 2^k + 1$ states is started at a tile $\langle i, j\rangle$ that is landlocked for $\mathcal{F}$, then $\mathcal{F}$ will enter the same state (at least) twice during the first $q + 1$ steps of its journey from $\langle i, j\rangle$ to $\langle i + 2^k, j\rangle$. Moving $\mathcal{F}$'s start tile via either a compression or extension (cf. Fig. 2) will cause $\mathcal{F}$ to end up at a tile other than $\langle i + 2^k, j\rangle$.
2. The following $(2^k + 1)$-state FSM $\mathcal{F}^{(k)}$ solves the $\mathsf{NSCN}_{n,k}$ Problem without using pheromones. $\mathcal{F}^{(k)}$ has states $s_0, \ldots, s_{2^k}$. Starting in state s_0, $\mathcal{F}^{(k)}$ marches southward through successive states, $s_1, s_2, \ldots$, halting when it enters state s_{2^k}.
3. We design an $O(k)$-state FSM $\mathcal{F}_k$ that solves the $\mathsf{NSCN}_{n,k}$ Problem. In broad outline, $\mathcal{F}_k$ uses its phermomone to simulate a k-bit counter that it "carries along" as it makes its way from the interior tile $\langle i, j\rangle$ to $\langle i + 2^k, j\rangle$. We assume for definiteness that $n - i \geq k$ (which creates "westward" room for the counter); clerical changes accommodate the case $n - i < k$ by using "eastward" room.

Designing the counter. $\mathcal{F}_k$ employs a pheromone with $6k + 1$ levels of intensity. It uses levels $1, \ldots, 3k$ to represent bit 0 and levels $3k + 1, \ldots, 6k$ to represent bit 1; as usual, level 0 indicates the absence of the pheromone. (The redundant representation compensates for pheromones' volatility.) For each integer $h \in [0, 2^k - 1]$ and each tile $\langle i+h, j\rangle$ along the southward path

$$\langle i, j\rangle \ \rightarrow \ \langle i+1, j\rangle \ \rightarrow \cdots \rightarrow \ \langle i+2^k, j\rangle, \tag{3}$$

in turn, $\mathcal{F}_k$ forms an encoding (using pheromone intensity levels) of the length-k binary numeral for integer h, using the tiles $\langle i+h, j-(k-1)\rangle \ldots, \langle i+h, j\rangle$ to store (encodings of) the numeral's successive bits: $\langle i+h, j\rangle$ will contain the hth counter's low-order bit and $\langle i+h, j-k+1\rangle$ of that counter's high-order bit. $\mathcal{F}_k$ uses enough levels of pheromone intensity for encoding so that each bit of each counter will survive long enough for $\mathcal{F}_k$ to progress to the next tile to the south and propagate and increment the hth counter to the $(h+1)$th. When $\mathcal{F}_k$ eventually generates the numeral $11\cdots1$ for the integer 2^k-1 on its moving counter, it knows that the next tile to the south is the target tile $\langle i+2^k, j\rangle$. Because $\mathcal{F}_k$ uses pheromone levels to represent bits and because it must explicitly traverse the k tiles that encode each numeral and because the pheromone is volatile, $\mathcal{F}_k$ employs multiple levels of the pheromone to encode the bits 0 and 1, as mentioned earlier.

We now derive a viable encoding—and, thereby, a viable upper bound on the number of levels of intensity for the pheromone—by presenting an explicit algorithm for $\mathcal{F}_k$ and analyzing how long the representation of a bit must persist in order for $\mathcal{F}_k$ to count from 0 to 2^k-1 using length-k numerals. We use the following notation. For $\beta \in \{0,1\}$:

- "(β)" ambiguously denotes any encoding of bit β using pheromone levels; under our scheme, $(0) \in \{1, \ldots, 3k\}$, and $(1) \in \{3k+1, \ldots, 6k\}$;
- "$[\beta]$" denotes the encoding of bit β via a maximum level of pheromone; thus, $[0]$ represents level $3k$ of the pheromone, and $[1]$ represents level $6k$.

$\mathcal{F}_k$ initializes the counter. $\mathcal{F}_k$ begins its counter-constructing journey along the southward path (3) by initializing the counter to the numeral $00\cdots0$. This consists of traversing the following length-k westward path from $\langle i, j\rangle$

Initialize counter 0 to $00\cdots0$

$$\begin{array}{ccccccc} \langle i, j-k+1\rangle & \leftarrow & \langle i, j-k+2\rangle & \leftarrow & \cdots & \leftarrow & \langle i, j\rangle \\ \boxed{\text{deposit } [0]} & & \boxed{\text{deposit } [0]} & & & & \boxed{\text{deposit } [0]} \end{array}$$

depositing dose $[0]$ of the pheromone at each tile. A potential finite-state subprogram for $\mathcal{F}_k$ appears in Fig. 4; the right arrow identifies the initial states of each of the sub-FSMs that jointly comprise $\mathcal{F}_k$. Having initialized the counter, $\mathcal{F}_k$ returns to tile $\langle i, j\rangle$ via the eastward path

$$\begin{array}{l} \text{Return and check counter } h \text{ for } 11\cdots1 \\ \langle i+h, j-k+1\rangle \ \rightarrow \ \langle i+h, j-k+2\rangle \ \rightarrow \cdots \rightarrow \ \langle i+h, j\rangle. \end{array} \tag{4}$$

The initial instantiation of this check-and-return path has $h = 0$; subsequent instantiations will have, in turn, $h = 1, 2, \ldots, k-1$. During these eastward paths, $\mathcal{F}_k$ tests whether the current value of the counter encodes the integer

	Current State	Tile Contents	Action	Move Direction	New State
$\rightarrow$	$\boxed{\text{INITIALIZE}}_0$	ALL	Deposit [0]	west	$\boxed{\text{INITIALIZE}}_1$
	$\boxed{\text{INITIALIZE}}_1$	ALL	Deposit [0]	west	$\boxed{\text{INITIALIZE}}_2$
	$\vdots$				
	$\boxed{\text{INITIALIZE}}_{k-1}$	ALL	Deposit [0]	west	$\boxed{\text{GO-BACK}}_0$

Figure 4: A counter-initializing sub-FSM of $\mathcal{F}_k$

$2^k - 1$ (via the binary numeral $11 \cdots 1$). If it does, then $\mathcal{F}_k$ knows that the next tile to the south of $\langle i+h, j\rangle$ is the target tile $\langle i+2^k, j\rangle$; in this case, $\mathcal{F}_k$ just moves to this tile and halts. If the counter does not encode $2^k - 1$, then $\mathcal{F}_k$ proceeds to propagate and increment the counter southward. A potential check-and-return finite-state subprogram for $\mathcal{F}_k$ appears in Fig. 5. Because $\mathcal{F}_k$

	Current State	Tile Contents	Action	Move Direction	New State

$\rightarrow$	$\boxed{\text{GO-BACK}}_0$	ALL	no action	east	**if** all (1)s thus far
					then $\boxed{\text{GO-BACK}}_{1,y}$
					else $\boxed{\text{GO-BACK}}_{1,n}$

	$\boxed{\text{GO-BACK}}_{1,n}$	ALL	no action	east	$\boxed{\text{GO-BACK}}_{2,n}$

	$\boxed{\text{GO-BACK}}_{1,y}$	ALL	no action	east	**if** all (1)s thus far
					then $\boxed{\text{GO-BACK}}_{2,y}$
					else $\boxed{\text{GO-BACK}}_{2,n}$

	$\boxed{\text{GO-BACK}}_{2,n}$	ALL	no action	east	$\boxed{\text{GO-BACK}}_{3,n}$

	$\boxed{\text{GO-BACK}}_{2,y}$	ALL	no action	east	**if** all (1)s thus far
					then $\boxed{\text{GO-BACK}}_{3,y}$
					else $\boxed{\text{GO-BACK}}_{3,n}$

	$\vdots$				
	$\boxed{\text{GO-BACK}}_{k-1,n}$	ALL	no action	east	$\boxed{\text{COUNTER}}$
	$\boxed{\text{GO-BACK}}_{k-1,y}$	ALL	no action	south	$\boxed{\text{HALT}}$

Figure 5: A check-and-return sub-FSM for $\mathcal{F}_k$

returns to each tile $\langle i+h, j\rangle$ $2k-1$ steps after initially leaving it, we can ensure the persistence of a nonzero dose of pheromone on $\langle i+h, j\rangle$ upon $\mathcal{F}_k$'s return, if we have $[0] \geq 2k$ and $2k < [1] \leq 4k$.

$\mathcal{F}_k$ propagates and increments the counter. Assume inductively that $\mathcal{F}_k$ is on tile $\langle i+h, j\rangle$, where $h \in [0, 2^k - 1]$, and that tiles $\langle i+h, j\rangle, \ldots, \langle i+h, j-k+1\rangle$ contain pheromone levels that encode integer h. $\mathcal{F}_k$ initiates a sawtooth traversal of the following form, during which it deposits at row $h+1$ pheromone levels that encode integer $h+1$.

Propagate incremented counter from row h to row $h+1$:

$$\begin{tikzcd}
\boxed{\text{Pick up bit}} & & \boxed{\text{Pick up bit}} & & & \boxed{\text{Pick up bit}} \\
\langle i+h,j\rangle \arrow[d] & & \langle i+h,j-1\rangle \arrow[d] & & \cdots & \langle i+h,j-k+1\rangle \arrow[d] \\
\langle i+h+1,j\rangle \arrow[rru] & & \langle i+h+1,j-1\rangle \arrow[rru] & & \arrow[ru] & \langle i+h+1,j-k+1\rangle \\
\boxed{\text{Bit + Carry}} & & \boxed{\text{Bit + Carry}} & & & \boxed{\text{Bit + Carry}}
\end{tikzcd}$$

A potential finite-state propagate-and-increment subprogram for $\mathcal{F}_k$ appears in Fig. 6. The reader will recognize the $\boxed{\text{ADD}}$ sub-FSM as a length-k carry-propagate incrementer. In state $\boxed{\text{ADD}}$-(β, α), this FSM generates the sum and carry out-bits in response to the input bit α and carry-in bit β. In the check-and-return sub-FSM, $\mathcal{F}_k$ ensures that the counter contents never exceed $2^k - 1$, which is why state $\boxed{\text{ADD}}$-$(1,1)$ cannot occur. Having completed an update of

	Current State	Tile Contents	Action	Move Direction	New State
$\rightarrow$	$\boxed{\text{COUNTER}}$	(β)	no action	south	$\boxed{\text{ADD}}$-$(\beta,1)_0$
	$\boxed{\text{ADD}}$-$(0,0)_0$	ALL	deposit [0]	northeast	$\boxed{\text{GET-BIT}}$-$(0)_1$
	$\boxed{\text{ADD}}$-$(0,1)_0$	ALL	deposit [1]	northeast	$\boxed{\text{GET-BIT}}$-$(0)_1$
	$\boxed{\text{ADD}}$-$(1,0)_0$	ALL	deposit [1]	northeast	$\boxed{\text{GET-BIT}}$-$(0)_1$
	$\boxed{\text{ADD}}$-$(1,1)_0$	ALL	deposit [1]	northeast	$\boxed{\text{GET-BIT}}$-$(1)_1$
	$\boxed{\text{GET-BIT}}$-$(0)_1$	(β)	no action	south	$\boxed{\text{ADD}}$-$(\beta,0)_1$
	$\boxed{\text{GET-BIT}}$-$(1)_1$	(β)	no action	south	$\boxed{\text{ADD}}$-$(\beta,1)_1$
	$\boxed{\text{ADD}}$-$(0,0)_1$	ALL	deposit [0]	northeast	$\boxed{\text{GET-BIT}}$-$(0)_2$
	$\boxed{\text{ADD}}$-$(0,1)_1$	ALL	deposit [1]	northeast	$\boxed{\text{GET-BIT}}$-$(0)_2$
	$\boxed{\text{ADD}}$-$(1,0)_1$	ALL	deposit [1]	northeast	$\boxed{\text{GET-BIT}}$-$(0)_2$
	$\boxed{\text{ADD}}$-$(1,1)_1$	ALL	deposit [1]	northeast	$\boxed{\text{GET-BIT}}$-$(1)_2$
	$\vdots$				
	$\boxed{\text{GET-BIT}}$-$(0)_{k-1}$	(β)	no action	south	$\boxed{\text{ADD}}$-$(\beta,0)_{k-1}$
	$\boxed{\text{GET-BIT}}$-$(1)_{k-1}$	(β)	no action	south	$\boxed{\text{ADD}}$-$(\beta,1)_{k-1}$
	$\boxed{\text{ADD}}$-$(0,0)_{k-1}$	ALL	deposit [0]	northeast	$\boxed{\text{GO-BACK}}$
	$\boxed{\text{ADD}}$-$(0,1)_{k-1}$	ALL	deposit [1]	northeast	$\boxed{\text{GO-BACK}}$
	$\boxed{\text{ADD}}$-$(1,0)_{k-1}$	ALL	deposit [1]	northeast	$\boxed{\text{GO-BACK}}$

Figure 6: Propagating and incrementing the counter for $\mathcal{F}_k$

the counter, $\mathcal{F}_k$ returns to tile $\langle i+h+1, j\rangle$ via the path (4).

Because the propagate-plus-increment process—$\mathcal{F}_k$'s sawtooth path from tile $\langle i+h, j\rangle$ to tile $\langle i+h+1, j+k-1\rangle$, followed by its return to tile $\langle i+h+1, j\rangle$—takes $3k$ steps, insisting that $\ell \geq 3k$ ensures the persistence of a nonzero dose of pheromone on tile $\langle i+h+1, j\rangle$ upon $\mathcal{F}_k$'s return. In fact, because of the "semantics" of $\mathcal{F}_k$'s total journey, we double the indicated number of levels of the pheromone—because $\mathcal{F}_k$ must, in fact, distinguish between levels of the pheromone that encode bit 0 and levels that encode bit 1.

Validation. The only "danger" while implementing the described strategy is that the pheromone will evaporate somewhere while $\mathcal{F}_k$ still needs access to it—so that it can distinguish 1 from 0. Focusing on a specific but arbitrary tile $\langle r, s\rangle$, in the worst case, $\mathcal{F}_k$:

1. first deposits the pheromone on tile $\langle r, s\rangle$ while copying the numeral from the preceding row, row $r-1$;
2. takes 3 steps per bit to copy the k bits in tiles to the right of $\langle r, s\rangle$ into the current row, row r;
3. returns to row r in preparation for copying this row upward to row $r+1$;
4. takes 3 steps per bit to copy the $< k$ bits in tiles to the right of $\langle r, s\rangle$ to the next row, row $r+1$.

This process thus takes $3k$ steps, after which we no longer care if the pheromone is still detectable on tile $\langle r, s\rangle$. Thus, using a pheromone that has $6k$ nonzero intensity levels—$3k$ to represent 0 and $3k$ to represent 1—ensures that $\mathcal{F}_k$ can always solve the $\mathsf{NSCN}_{n,k}$ Problem. □

4 Home-Quadrant Determination

The Parking Problem can fruitfully be partitioned into two subproblems:
Home-Quadrant determination. Each FSM determines its home quadrant (i.e., the one it starts in) and, thereby, its parking corner in $\mathcal{M}_n$.
Parking within the home quadrant. FSMs that share a home quadrant—hence, a parking corner—assemble in a configuration that minimizes the parking potential function (2).

The Home-Quadrant Determination Problem is significant independent of its use in solving the Parking Problem, as an instance of the fundamental question "What can FSMs discover about where they reside within a mesh?" Therefore, we devote this section to studying initial configurations of FSMs that allow each FSM to determine its home quadrant. We show in Section 5 how such configurations of FSMs can always park in every mesh. We cannot yet characterize the configurations that allow home-quadrant determination, but we can identify two simply described ones.

1. The initial assemblage of FSMs includes one designated FSM that resides on an edge or at a corner of $\mathcal{M}_n$.
2. The initial assemblage of FSMs includes two designated FSMs that are *adjacent,* i.e., reside on tiles that share an edge or a corner.

Theorem 4 **(a)** *Any collection of FSMs that (initially) contains a designated FSM on an edge of $\mathcal{M}_n$ can determine their home quadrants in $\mathcal{M}_n$ within $O(n^3)$ synchronous steps.* **(b)** *Any collection of FSMs that (initially) contains two designated adjacent FSMs can determine their home quadrants in $\mathcal{M}_n$ within $O(n^2)$ synchronous steps.*

The proof of Theorem 4 occupies the following subsections. We focus first on how a single FSM on a mesh-edge can determine its home quadrant (Section 4.1), then on how two initially adjacent FSMs can determine their home quadrants (Section 4.2). We show finally how these knowledgeable FSMs can act as "shepherds" to help an arbitrary collection of FSMs determine their home quadrants (Section 4.3).

4.1 A Single FSM on a Mesh-Edge

Lemma 1 *One can design an FSM that can determine its home quadrant from any* edge-tile *of any mesh $\mathcal{M}_n$ within $O(n)$ steps.*

Proof: We design an FSM $\mathcal{F}$ that can determine its home quadrant when started on $\mathcal{M}_n$'s *top edge.* An easy modification of $\mathcal{F}$, call it $\widehat{\mathcal{F}}$, will be able to start on any edge of $\mathcal{M}_n$. Specifically, $\widehat{\mathcal{F}}$ begins by scanning the edge-labels on its start tile (cf. Section 2.1) to determine which edge of $\mathcal{M}_n$ it is starting on. $\widehat{\mathcal{F}}$ uses that information to determine which "rotation" of the following algorithm to use in order to determine its home quadrant.

Say that FSM $\mathcal{F}$ starts on $\mathcal{M}_n$'s top edge, at an arbitrary tile $\langle 0, k\rangle$, so that $\mathcal{F}$'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° southeasterly walk from $\langle 0, k\rangle$, i.e., a walk consisting of the following "Knight's-move supersteps": two-step moves of the form SE-then-S. (It simplifies our analysis to consider the numerical form of these moves, $(+1,+1)$-then-$(+1,0)$. Consider a superstep that starts with $\mathcal{F}$ on tile $\langle i, j\rangle$ of $\mathcal{M}_n$. (Recall that *we* know the coordinates of the tile; $\mathcal{F}$ does not.)

```
if     $\langle i, j\rangle$ is an edge tile or a corner tile of the mesh
then   $\mathcal{F}$'s walk terminates
else   $\mathcal{F}$ moves SE to tile $\langle i+1, j+1\rangle$
       if     $\langle i+1, j+1\rangle$ is a bottom tile
       then   $\mathcal{F}$'s walk terminates
       else   $\mathcal{F}$ moves S to tile $\langle i+2, j+1\rangle$
```

$\mathcal{F}$'s walk, which is depicted schematically in Fig. 7, terminates when $\mathcal{F}$ encoun-

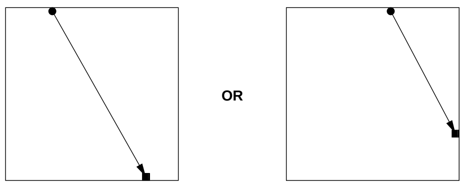


Figure 7: The two possible forms of $\mathcal{F}$'s 60° southeasterly walk.

ters $\mathcal{M}_n$'s bottom edge or right edge or SE corner. We claim that the endpoint of the walk identifies $\mathcal{F}$'s target parking tile. To see this, note that $\mathcal{F}$'s walk terminates in a tile $v = \langle a, b\rangle$. There are two possibilities: $\mathcal{F}$ is prevented (by an edge or corner of $\mathcal{M}_n$) from:

1. taking the second (southward) step of its sth superstep. This means that $\mathcal{F}$ has completed one more SE move than S move, so we have $\big[[a = 2s-1]$ and $[b = k+s]\big]$, and we must also have $\big[[a = n-1]$ and $[b \le n-1]\big]$ (whence the interruption). Combining these inequalities, we have $k < \frac{1}{2}(n-1)$, which means that $\mathcal{F}$'s target parking tile is $\langle 0, 0\rangle$;

2. taking the first (southeasterly) step of its $(s+1)$th superstep. This means that $\mathcal{F}$ has completed the same number of SE and S moves, so we have $\big[[a = 2s]$ and $[b = k+s]\big]$, and we must also have $\big[[a = n-1]$ or $[b = n-1]\big]$ (whence the interruption). Combining these inequalities, we have $k \ge \frac{1}{2}(n-1)$, which means that $\mathcal{F}$'s target parking tile is $\langle 0, n-1\rangle$.

This analysis verifies that the program in Fig. 8 enables $\mathcal{F}$ to park from tile $\langle 1, k\rangle$. □

	Current State	Current Tile	Move Direction	New State
→	MOVE-SE	not bottom or right edge	move southeast	MOVE-S
	MOVE-S	not bottom or right edge	move south	MOVE-SE
	MOVE-SE	bottom edge OR SE corner	move northwest	MOVE-NW
	MOVE-S	bottom edge OR SE corner	move northwest	MOVE-NW
	MOVE-SE	right edge	move north	MOVE-N
	MOVE-S	right edge	move north	MOVE-N
	MOVE-NW	not NW corner	move northwest	MOVE-NW
	MOVE-NW	NW corner	no action	HALT
	MOVE-N	not NE corner	move north	MOVE-N
	MOVE-N	NE corner	no action	HALT

Figure 8: A single FSM parks from the top edge of $\mathcal{M}_n$

4.2 Two Initially Adjacent FSMs

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

The algorithm of Section 4.1 can be adapted to allow two initially adjacent FSMs to determine their home quadrants. The adaptation leads to the following four-phase algorithm, which is illustrated in Fig. 9.

Phase 1. *FSM #1 distinguishes east from west.* Say, for definiteness, that the two FSMs are *horizontally* adjacent, with $\mathcal{F}_1$ to the right of $\mathcal{F}_2$. This assumption loses no generality, because the FSMs can remember their actual initial configuration (in finite-state memory), then move into the left-right configuration, and finally compute the adjustments necessary to accommodate their actual configurations.

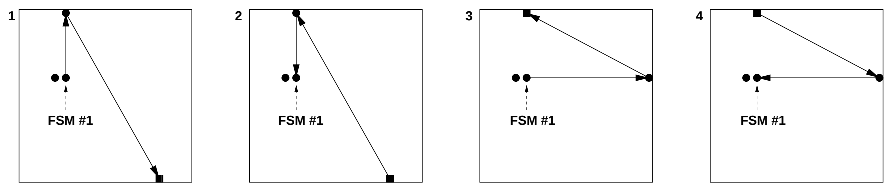


Figure 9: Illustrating home-quadrant determination for two adjacent FSMs: (1,2) FSM #1 discovers that it is a "westerner"; (3,4) FSM #1 discovers that it is a "northerner."

$\mathcal{F}_2$ stays immobile while $\mathcal{F}_1$ proceeds to the top edge of $\mathcal{M}_n$. $\mathcal{F}_1$ thence executes the top-edge algorithm of Section 4.1 to determine whether it is an "easterner" or a "westerner" (Fig. 9.1). $\mathcal{F}_1$ then returns to its home tile by reversing its walk (Fig. 9.2). This reversal is possible because: (a) the first leg of the return walk just reverses the sequence of supersteps that accomplished the outward walk; (b) once $\mathcal{F}_1$ regains the top edge of $\mathcal{M}_n$, it travels southward until it encounters $\mathcal{F}_2$. $\mathcal{F}_2$ thus acts as a "sentry" or "place-holder" for $\mathcal{F}_1$.

Phase 2. *FSM #1 distinguishes north from south.* $\mathcal{F}_2$ stays immobile while $\mathcal{F}_1$ proceeds to the right edge of $\mathcal{M}_n$, $\mathcal{F}_1$ thence executes the right-edge algorithm of Section 4.1 to determine whether it is a "northerner" or a "southerner" (Fig. 9.3). $\mathcal{F}_1$ then returns to its home tile by reversing its walk (Fig. 9.4). Here too, the return is possible by: (a) reversing the sequence of supersteps that accomplished the outward walk; (b) traveling westward from the right edge of $\mathcal{M}_n$ until it encounters $\mathcal{F}_2$.

Phase 3 (*FSM #2 distinguishes east from west*) and Phase 4 (*FSM #2 distinguishes north from south*) have $\mathcal{F}_1$ act as a "sentry" for $\mathcal{F}_2$ while the latter executes analogues of Phases 1 and 2.

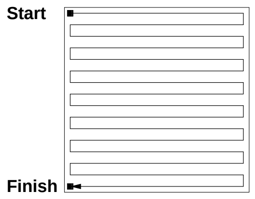
Note. Two FSMs that have a pheromone with $I \geq 2k$ levels of intensity can determine their home quadrants when started within k tiles of one another.

4.3 Knowledgeable FSMs Act as Shepherds

We have $m \geq 2$ FSMs, $\mathcal{F}_1, \ldots, \mathcal{F}_m$. As the FSMs execute the algorithms of this section, some may block the intended paths of others. *We resolve such conflicts by having the involved FSMs* switch roles*—which is possible because all FSMs are identical. If FSM $\mathcal{F}$ is blocking FSM $\mathcal{F}'$, then $\mathcal{F}$ "becomes" $\mathcal{F}'$ and continues the latter's blocked trajectory; simultaneously, $\mathcal{F}'$ "becomes" $\mathcal{F}$ and continues its trajectory.*

4.3.1 One knowledgeable shepherd

Say that FSM $\mathcal{F}_1$ knows its home quadrant (possibly, but not necessarily from executing the algorithm of Section 4.1). $\mathcal{F}_1$ helps other FSMs determine their home quadrants by performing a snaked row-by-row sweep of $\mathcal{M}_n$—say for definiteness from tile $\langle 0, 0\rangle$.



As $\mathcal{F}_1$ encounters each $\mathcal{F}_i$ where $i \in [2, m]$, it acts as a sentry, in the spirit of the 2-FSM algorithm of Section 4.2, thereby allowing $\mathcal{F}_i$ to determine its home quadrant. Once $\mathcal{F}_i$ returns to its original tile, it starts toward its target parking corner—by joining the final parking process of Section 5—and $\mathcal{F}_1$ resumes its shepherding sweep of $\mathcal{M}_n$.

The described process takes $O(n^3)$ steps. $\mathcal{F}_1$'s sweep of $\mathcal{M}_n$ takes $O(n^2)$ moves. Whenever $\mathcal{F}_1$ encounters another $\mathcal{F}_i$, the two FSMs collaborate in an $O(n)$-step algorithm that enables $\mathcal{F}_i$ to determine its home quadrant. □

4.3.2 Two knowledgeable shepherds

Say that FSMs $\mathcal{F}_1$ and $\mathcal{F}_2$ know their home quadrants (possibly, but not necessarily from executing the algorithm of Section 4.2). The following multi-phase $O(n^2)$-step algorithm has $\mathcal{F}_1$ and $\mathcal{F}_2$ help all other FSMs determine their home quadrants.

Note. The following algorithm is designed for *a single pair* of shepherds. It is not clear if enlisting a larger team of shepherds or enlisting multiple teams would lead to a faster algorithm.

Phase 1. *$\mathcal{F}_1$ and $\mathcal{F}_2$ distinguish east from west.* $\mathcal{F}_1$ and $\mathcal{F}_2$ head to $\mathcal{M}_n$'s NW corner (tile $\langle 0, 0\rangle$). Then:

1. $\mathcal{F}_2$ moves one tile eastward per time-step until it reaches $\mathcal{M}_n$'s right edge. It then *reverses direction* and begins to move one tile westward per time-step.

2. $\mathcal{F}_1$ starts one step later than $\mathcal{F}_2$ and moves one tile eastward at *every third time-step.*

3. $\mathcal{F}_1$ and $\mathcal{F}_2$ terminate their walks when they are in adjacent tiles.

If $n-1$ *is even*, then when the FSMs meet, $\mathcal{F}_1$ will be on tile $\langle 0, \frac{1}{2}(n-1)-1\rangle$, and $\mathcal{F}_2$ will be on tile $\langle 0, \frac{1}{2}(n-1)\rangle$. To wit:

- $\mathcal{F}_2$'s trajectory, $\langle 0, 0\rangle \rightsquigarrow \langle 0, n-1\rangle \rightsquigarrow \langle 0, \frac{1}{2}(n-1)\rangle$ takes $\frac{3}{2}(n-1)$ time-steps.
- $\mathcal{F}_1$'s trajectory $\langle 0, 0\rangle \rightsquigarrow \langle 0, \frac{1}{2}(n-1)-1\rangle$ takes $\frac{1}{2}(n-1)-1$ moves. Because $\mathcal{F}_1$ starts one time-step later than $\mathcal{F}_2$ and proceeds at $\frac{1}{3}$ rate, $\mathcal{F}_1$ arrives at $\langle 0, \frac{1}{2}(n-1)\rangle$ after $\frac{3}{2}(n-1)$ time-steps.

If $n-1$ *is odd*, then when the FSMs meet, $\mathcal{F}_1$ will be on tile $\langle 0, \lfloor\frac{1}{2}(n-1)\rfloor\rangle$, and $\mathcal{F}_2$ will be on tile $\langle 0, \lceil\frac{1}{2}(n-1)\rceil\rangle$. To wit:

- $\mathcal{F}_2$'s trajectory, $\langle 0, 0\rangle \rightsquigarrow \langle 0, n-1\rangle \rightsquigarrow \langle 0, \lceil\frac{1}{2}(n-1)\rceil\rangle$ takes $(n-1)+\lfloor\frac{1}{2}(n-1)\rfloor = 3\lfloor\frac{1}{2}(n-1)\rfloor + 1$ time-steps.

- $\mathcal{F}_1$'s trajectory $\langle 0,0\rangle \rightsquigarrow \langle 0, \lfloor \frac{1}{2}(n-1)-1 \rfloor\rangle$ takes $\lfloor \frac{1}{2}(n-1)\rfloor$ moves. Because $\mathcal{F}_1$ starts one time-step later than $\mathcal{F}_2$ and proceeds at $\frac{1}{3}$ rate, $\mathcal{F}_1$ arrives at $\langle 0, \lfloor \frac{1}{2}(n-1)\rfloor\rangle$ after $3\lfloor \frac{1}{2}(n-1)\rfloor + 1$ time-steps.

After these walks, $\mathcal{F}_1$ and $\mathcal{F}_2$ know the midpoint of $\mathcal{M}_n$'s top row.

Phase 2. *$\mathcal{F}_1$ and $\mathcal{F}_2$ identify easterners and westerners.* $\mathcal{F}_1$ sweeps column-wise through the western half of $\mathcal{M}_n$, from column $\lceil \frac{1}{2}(n-1)\rceil - 1$ through column 0, informing each encountered FSM that it is a *westerner*, i.e., resides in either $\mathcal{Q}_{NW}$ or $\mathcal{Q}_{SW}$. Simultaneously, $\mathcal{F}_2$ does the symmetric task in the eastern half of $\mathcal{M}_n$, from column $\lceil \frac{1}{2}(n-1)\rceil$ through column $n-1$, informing each encountered FSM that it is an *easterner*, i.e., resides in either $\mathcal{Q}_{NE}$ or $\mathcal{Q}_{SE}$. After completing their sweeps, $\mathcal{F}_1$ and $\mathcal{F}_2$ rendezvous at tile $\langle 0,0\rangle$.

Phase 3. *$\mathcal{F}_1$ and $\mathcal{F}_2$ distinguish north from south,* via a process analogous to that of Phase 1.

Phase 4. *$\mathcal{F}_1$ and $\mathcal{F}_2$ identify northerners and southerners,* via a process analogous to that of Phase 2.

By the end of Phase 4, every FSM knows its home quadrant.

Phase 5. *FSMs park.* Every FSM except for $\mathcal{F}_1$ and $\mathcal{F}_2$ begins to park as soon as it determines its home quadrant. $\mathcal{F}_1$ and $\mathcal{F}_2$ wait to park until the end of Phase 4, when their shepherding duties are done. All FSMs join the algorithm of Section 5 as they park.

5 Completing the Parking Process

We describe finally how FSMs that know their home quadrants travel to their parking corner and arrange themselves within that corner into a configuration that minimizes the parking potential function (2). We focus just on corner SW of $\mathcal{M}_n$, hence on FSMs that resided initially in $\mathcal{Q}_{SW}$; clerical changes adapt this procedure to the other corners.

Phase 1. *FSMs travel to parking corners.* Each FSM $\mathcal{F}$ follows a two-stage trajectory to its parking corner. For $\mathcal{Q}_{SW}$, the trajectory proceeds westward to the left edge of $\mathcal{M}_n$. Having achieved that edge, $\mathcal{F}$ proceeds southward toward its parking corner. An FSM that is proceeding horizontally: (a) moves only to an empty tile; if none exists, then it waits; (b) yields to an FSM that is already proceeding vertically.

Phase 2. *FSMs organize within their corner.* We depict this process schematically in Fig. 10. The first FSM that reaches its parking corner (it may have started there) becomes an *usher*. It directs vertically-arriving FSMs into the next adjacent diagonal (say that this is diagonal[5] Δ_k). Thus-directed FSMs proceed "down" this diagonal—i.e., in the northwest-to-southeast sense; if they encounter the bottom edge of $\mathcal{M}_n$, then they continue "up" the next higher-index diagonal (Δ_{k+1} in our example)—i.e., in the southeast-to-northwest sense.

[5] See (1) for notation.

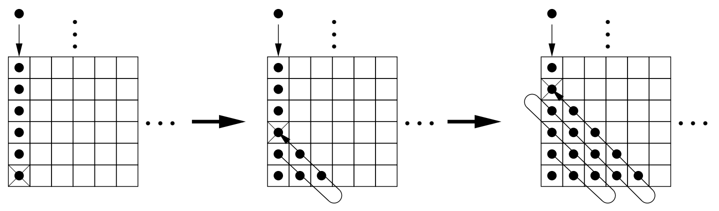


Figure 10: Three stages in the snaked parking trajectory within $\mathcal{Q}_{SW}$; X-ed cells contain "ushers."

An FSM in a diagonal moves only when some other FSM "behind" it wants them to move. When an FSM that is moving "up" a diagonal regains the left edge of $\mathcal{M}_n$, it "defrocks" the current usher and becomes an usher itself (via a message relayed by its lower neighbor). Corner SW of $\mathcal{M}_n$ thus gets filled in compactly, two diagonals at a time, i.e., into a configuration that minimizes the parking potential function (2).

This completes the parking algorithm and the proof of Theorem 4. □

6 Conclusions

We have reported progress in understanding the algorithmic strengths and weaknesses of finite-state robots (FSMs) within square meshes of arbitrary sizes. Our vehicle has been the simple path-planning problem we call *parking,* that has FSMs configure themselves into maximally compact configurations within their nearest corners of the mesh. The Parking Problem is a reasonable subject of study in terms of both the algorithmic and robotic inspirations for our work. From the former perspective, Parking is a tractable instance of the question "What can FSMs discover about where they reside within a mesh?" From the latter perspective, Parking might be a useful capability to achieve efficiently in some of the application domains for mobile robots, e.g., warehouses and factories; cf. [16]. Our main results: (a) illustrate how to formalize the intuition that FSMs "get lost" in the interiors of large meshes (Section 3.1); (b) demonstrate that FSMs can sometimes exploit mesh edges to accomplish tasks that seem to require unbounded counting—which FSMs of course cannot do (Section 4.1); (c) demonstrate that teams of FSMs can sometimes collaborate to accomplish rather sophisticated path planning (Section 4.2); (d) provide perspective on the benefits of digital analogues of pheromones for FSMs, an algorithmic tool advocated in [12, 18, 30] (Sections 3.2, 3.3). The algorithms that establish our positive results adapt with only clerical changes to rectangular meshes with fixed non-unit aspect ratios.

We mention "for the record" that if algorithmic efficiency is not a concern, then any collection of FSMs that contains at least four initially 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. [27].

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. Another major direction for future work is to build on the initial results of [29] to better understand FSMs within the context of the question "How well can FSMs discover designated target tiles within meshes?"—a question that complements the focus of the current study—"What can FSMs discover about where they reside within a mesh?" Within the context of both motivating questions, it would be valuable to understand how FSMs can cope with obstacles that impede their progress and to understand the possible benefits of randomizing the behavior of the FSMs. A valuable source of inspiration for understanding these questions are the robotic studies in sources such as [1, 8, 16, 18]. Perhaps the most important follow-ons to the current work would go beyond pure path planning and exploration by designing FSMs that can scalably (i.e, in arbitrarily large meshes) find and transport goal objects and avoid obstacles. Sources such as [2, 12, 16, 25, 26] consider such issues for related but distinct models. Questions concerning the robustness of collections of FSMs in the face of various kinds of failures and faults can lend further texture to all of the preceding algorithmic questions.

Acknowledgments

I am grateful to O. Beaumont and O. Brock and to the anonymous reviewers for valuable insights and suggestions.

References

[1] Adler, F., Gordon, D.: Information collection and spread by networks of patrolling ants. *The American Naturalist 140* (1992) 373–400.

[2] Basu, P., Redi, J.: Movement control algorithms for realization of fault-tolerant ad hoc robot networks. *IEEE Network*, (July/August 2004), 36–44.

[3] Bender, M., Slonim, D.: The power of team exploration: two robots can learn unlabeled directed graphs. *35th IEEE Symp. on Foundations of Computer Science* (1994) 75–85.

[4] Bhatt, S., Even, S., Greenberg, D., Tayar, R.: Traversing directed eulerian mazes. *J. Graph Algorithms and Applications 6* (2002) 157–173.

[5] Blum, M., Sakoda, W.: On the capability of finite automata in 2 and 3 dimensional space. *18th IEEE Symp. on Foundations of Computer Science* (1977) 147–161.

[6] K.F. Böhringer (2006): Modeling and controlling parallel tasks in droplet-based microfluidic systems. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems 25*, 329–339.

[7] Budach, L.: On the solution of the labyrinth problem for finite automata. *Elektronische Informationsverarbeitung und Kybernetik (EIK) 11(10-12)* (1975) 661–672.

[8] Chen, L., Xu, X., Chen, Y., He, P.: A novel FSM clustering algorithm based on Cellular automata. *IEEE/WIC/ACM Int'l Conf. Intelligent Agent Technology* (2004).

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

[10] Cohen, R., Fraigniaud, P., Ilcinkas, D., Korman, A., Peleg, D.: Label-guided graph exploration by a finite automaton. *ACM Trans. on Algorithms 4* (2008).

[11] 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* (2006) 357–376.

[12] Geer, D.: Small robots team up to tackle large tasks. *IEEE Distributed Systems Online 6(12)* (2005).

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

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

[15] D.R. Hofstadter (1979): *Gödel, Escher, Bach.* Basic Books.

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

[17] Kobayashi, K.: The firing squad synchronization problem for two-dimensional arrays. *Information and Control 34* (1977) 177–197.

[18] Koenig, S., Szymanski, B., Liu, Y.: Efficient and inefficient ant coverage methods. *Annals of Mathematics and Artificial Intelligence 31* (2001) 41–76.

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

[20] Mazoyer, J.: On optimal solutions to the firing squad synchronization problem. *Theoretical Computer Science 168(2)* (1996) 367–404.

[21] Moore, E.F.: Gendanken experiments on sequential machines. In *Automata Studies* (C.E. Shannon, J. McCarthy, eds.) *[Ann. Math. Studies 34]*, Princeton Univ. Press, Princeton, NJ (1956) 129–153.

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

[23] Müller, H.: Endliche Automaten und Labyrinthe. *Elektronische Informationsverarbeitung und Kybernetik (EIK) 11(10-12)* (1975) 661–672.

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

[25] Rosenberg, A.L.: Cellular ANTomata. *5th Int'l Symp. on Parallel and Distributed Processing and Applications.* In *Lecture Notes in Computer Science 4742*, Springer, Heidelberg (2007) 78–90.

[26] Rosenberg, A.L.: Cellular ANTomata: food-finding and maze-threading. *37th Int'l Conf. on Parallel Processing* (2008).

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

[28] Rosenberg, A.L.: Ants in parking lots. *16th Int'l Conf. on Parallel Computing (EURO-PAR'10)*, Part II. In *Lecture Notes in Computer Science 6272*, Springer, Heidelberg (2010) 400–411.

[29] Rosenberg, A.L.: Finite-state robots in the land of Rationalia. In preparation (2012).

[30] Russell, R.: Heat trails as short-lived navigational markers for mobile robots. *Int'l Conf. on Robotics and Automation* (1997) 3534–3539.

[31] Shinahr, I.: Two- and three-dimensional firing-squad synchronization problems. *Information and Control 24* (1974) 163–180.

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

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

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

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