The Robot, the Grid, and the Algorithm

Chinda Wongngamnit Dana Angluin

YALE/DCS/TR-1188

October 1999

This research was supported by the National Science Foundation, grant CCR-9610295

The Robot, the Grid, and the Algorithm

Chinda Wongngamnit
Yale University

Dana Angluin[*]
Yale University

Abstract

We consider the problem of localizing a mobile robot in a bounded grid environment containing unknown obstructions. The robot is initially given the height and width of the grid, and is started in an unknown cell of the grid. We present a new algorithm, the Homing Robot, that successfully localizes using $O(\log n)$ bits of memory and no more than $4n$ moves, where n is the number of unobstructed cells in the grid.

1 Robot Localization in Grids

Imagine that a mobile robot has been placed somewhere on a two-dimensional surface (perhaps a floor) that is comprised of a finite number of obstacles (walls, poles, or furniture), as well as unobstructed space over which it can move (hallways and rooms). This clear space is enclosed all around by walls, so that the area that the robot can visit is bounded. The robot can locally sense obstructions and move through any unobstructed space it chooses to visit. The robot also has the ability to determine how far and in what direction it has traveled from its initial position; in other words, the robot has odometry. Finally, suppose that the robot has a map or other information about the space, but is unaware of its initial location for some reason. The robot's task is to orient itself, that is, to "figure out" as quickly as possible where on the map it is. This is the *robot localization problem.*

We study the robot localization problem in the particular case that the robot's environment is usefully represented by an occupancy grid. The grid partitions the space into square cells, each of which is either obstructed or unobstructed. The robot can occupy any unobstructed cell and can sense the obstructed/unobstructed status of each of the neighboring cells that share at least a corner with the current cell. The robot can move into any unobstructed cell that shares an edge with the current cell, making it the new current cell. We assume that the grid is enclosed by obstructed cells. An example of a grid is shown in Figure 1.

In this setting, we assume that a robot is placed in an unknown, unobstructed cell of a grid, and given some global information about the grid. The robot then makes observations and moves around the grid until it has determined which grid square it currently occupies, at which point it halts. The resources we consider are the number of moves made by the robot before it halts and the number of bits of memory required by the robot's computations. We seek worst-case bounds on these quantities as a function of n, the number of unobstructed cells in the grid. We propose a

[*]Supported by NSF grant CCR-9610295.

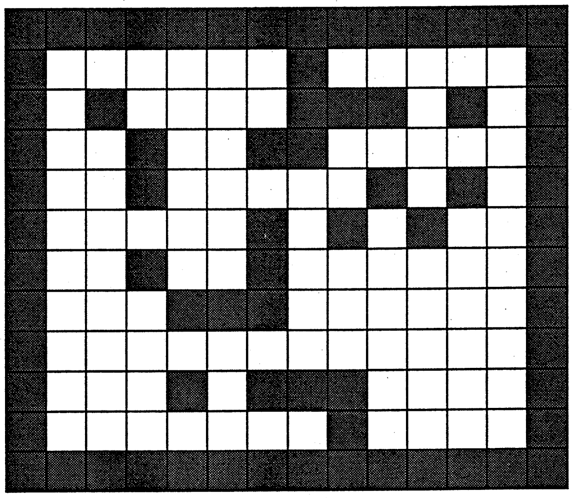

Figure 1: A grid with height 10, width 12, and five normal obstacles.

localization algorithm, the Homing Robot, that uses no more than $4n$ moves and $O(\log n)$ bits of memory in the worst case.

The robot localization problem is important in the practical design of mobile robots, and a variety of approaches have been proposed. The grids we consider are a deterministic version of probabilistic occupancy grids, which are one important approach to representing the environment of a mobile robot for localization and other navigation tasks. As one example, the museum-guide robot RHINO designed by Fox et al. used a localization algorithm based on a Markov-chain representation of a fine-grained grid covering the area of the museum travelled by the robot [4].

Blum and Kozen [2] considered mazes, which are equivalent to our grids, and addressed the theoretical question of how much computational power a robot needs in order to search a maze. The problem of searching a maze requires a robot to start in an unknown cell of an unknown maze and move in such a way that it eventually visits every unobstructed cell in the maze. It was already known that a robot with a finite-state controller and no additional memory cannot successfully search every maze. Blum and Kozen gave a method for a robot with a finite-state controller and one counter (or two pebbles) to search every maze. Their method uses $O(n^2)$ moves and $O(\log n)$ bits of memory.

The problem of robot localization has also been considered in the setting of a polygonal environment, assuming that the robot is equipped with vision, and using a competitive ratio as the figure of merit. Dudek et al. give a localization algorithm for this setting that has a competitive ratio of c, where c is the number of places in the polygonal environment that have the same visibility polygon as the initial point [3].

The problem of robot localization in a grid is also related to the problem of finding an adaptive homing sequence for a finite state automaton. This connection is discussed in Section 8.

2 Definitions

Consider a grid. The number of unobstructed cells in the grid is denoted by n.

If we specify one cell as the origin of the grid, there is a natural coordinate system associated with the grid cells. The pair of integers (H, V) specifies the cell that is H cells east and V cells north of the origin cell. The *westernmost* unobstructed cells are those unobstructed cells for which the H coordinate is minimum, and the *easternmost* unobstructed cells are those unobstructed cells for which the H coordinate is maximum. The *width* of a grid is one more than the difference between the H coordinates of the easternmost and westernmost unobstructed cells. The *southernmost* and *northernmost* unobstructed cells and the *height* of a grid are defined analogously in terms of the V coordinates of the cells. The grid in Figure 1 has width 12 and height 10.

We define obstacles and their boundaries in a grid as follows. Two cells are *edge-adjacent* if they share an edge, and *corner-adjacent* if they share at least a corner. The *neighbors* of a cell are the cells that are corner-adjacent to it; a cell has at most eight neighbors. The *neighborhood* of a cell consists of the cell and its neighbors. The neighbors of a cell are designated by their compass directions from the cell: N, NW, W, SW, S, SE, E, and NE.

A set T of cells is *edge-connected* if for every pair of cells in T, there is a path of edge-adjacent cells in T joining them, and similarly for *corner-connected.* We assume that the set of all unobstructed cells in the grid is edge-connected, in other words, all unobstructed cells are accessible to the robot.

An *obstacle* is a maximal corner-connected set of obstructed cells in the grid. A cell is on the *boundary* of an obstacle if it is unobstructed and corner-adjacent to some cell of the obstacle. Note that a cell may be on the boundary of up to four obstacles.

An obstacle is *normal* if its boundary cells enclose the cells of the obstacle. There is one obstacle, the *border*, in each grid that is not normal – its boundary cells are interior to it in the plane. The *outer boundary* consists of the boundary cells of the border. Note that the westernmost, easternmost, southernmost, and northernmost unobstructed cells in a grid must be part of its outer boundary. The grid in Figure 1 contains six obstacles: five normal obstacles and the border.

The robot is initially placed in some unobstructed cell of a grid, and is given the height and width of the grid. We do not assume that the robot has access to a map of the grid. The robot must determine its position in the grid by figuring out its horizontal displacement from the westernmost unobstructed cells and its vertical displacement from the southernmost unobstructed cells in the grid. Note that each of these quantities can be represented by $O(\log n)$ bits.

We assume that the robot has local sensors capable of determining whether each of the eight neighbors of the current cell is obstructed or unobstructed. We do not assume a capability (such as vision) for sensing distant obstacles. The robot may move to one of the unobstructed cells edge-adjacent to the current cell, making it the new current cell. In effect, the robot has a compass, and its moves may be described as taking one step N, W, S, or E. We assume that the sensations and actions of the robot are error-free. Relaxing this assumption is an interesting direction for future work.

3 A Framework for Localization in a Grid

How can the robot find out where it is in the grid? We describe a specific framework for algorithms to localize a robot in a grid under the assumptions we have made. This framework requires no map and $O(\log n)$ bits of memory to implement.

```
Initialize $H$, $V$, $H_{max}$, $H_{min}$, $V_{max}$, $V_{min}$ to 0.
While $((H_{max} - H_{min} + 1 < w)$ or $(V_{max} - V_{min} + 1 < h))$
    Choose a direction to move: N, S, E, or W.
    If N then
        $V \leftarrow V + 1$; $V_{max} = \max(V, V_{max})$
    If S then
        $V \leftarrow V - 1$; $V_{min} = \min(V, V_{min})$
    If E then
        $H \leftarrow H + 1$; $H_{max} = \max(H, H_{max})$
    If W then
        $H \leftarrow H - 1$; $H_{min} = \min(H, H_{min})$
Endwhile
Localization achieved.
```

Figure 2: A framework for localization algorithms.

The inputs to the robot when it is started are the height, h, and width, w, of the grid. The robot treats its initial position as the origin of the coordinate system described in the preceding section, and uses two registers, H and V, to keep track of its horizontal and vertical displacement in cells from its initial position.

Initially, H and V are set to 0, and if the robot moves to the cell to the north of its current position, it increments V by 1. For a move to the south, it decrements V by 1, and similarly for east (increment H) and west (decrement H).

In four other registers, the robot tracks the minimum and maximum values of V and H for cells that it has visited: V_{max}, V_{min}, H_{max}, and H_{min}. If, at some point, the robot observes that

$$V_{max} - V_{min} + 1 = h,$$

then it knows the vertical displacement of its current cell from the southernmost unobstructed cells, namely $V - V_{min}$. If the robot observes that

$$H_{max} - H_{min} + 1 = w,$$

then it knows the horizontal displacement of its current cell from the westernmost unobstructed cells, namely $H - H_{min}$. When both conditions are satisfied, the robot has localized itself in the grid, and halts.

In this framework, our basic localization strategy is for the robot to move around the grid, updating the the values of the six registers until both of these conditions are satisfied, and then to declare success. A concise description of this framework for localization is given in Figure 2. Clearly, the two conditions will be satisfied when the robot has visited a northernmost, a southernmost, a westernmost, and an easternmost unobstructed cell, in any order. The framework uses $O(\log n)$ bits of memory, to store the two input values and the six variables it uses; each value must be an integer in the range $[-n, n]$. We now describe four localization algorithms that share this framework but differ in how moves are chosen.

4 Three Methods Based on Searching

4.1 The Random Robot

A very simple localization algorithm, the Random Robot, can be implemented if the robot has a random number generator. At each step, the Random Robot chooses randomly one of the unobstructed edge-adjacent neighbors of the current cell, and moves there. To analyze this algorithm, we consider an undirected graph related to the grid. The *grid graph* is an undirected graph with n vertices, one corresponding to each unobstructed cell of the grid, and an edge between two vertices if and only if they represent edge-adjacent cells in the grid. Then the moves of the Random Robot are equivalent to a random walk on the grid graph. By a result of Aleliunas et al. [1], the expected number of moves until every vertex is visited is $O(n^2)$. Hence the Random Robot is a randomized localization algorithm that uses $O(\log n)$ bits of memory and $O(n^2)$ expected moves.

4.2 The BK Robot

A deterministic localization algorithm with the same resource bounds as the Random Robot may be obtained from the maze searching algorithm of Blum and Kozen [2]. In particular, if Blum and Kozen's seaching algorithm is used to choose the moves of the robot, then we are guaranteed that the robot will visit every unobstructed cell within $O(n^2)$ moves. This method, which we term the BK Robot, successfully localizes with $O(n^2)$ moves and $O(\log n)$ bits of memory.

5 The Depth-first Robot

The Random Robot and the BK Robot use very little memory and eventually visit every unobstructed cell. By contrast, the Depth-first Robot visits every unobstructed cell quickly by doing a depth-first search, but it uses much more memory. The moves of the Depth-first Robot on the grid correspond to a depth-first search of the grid graph, defined in section 4.1.

The Depth-first Robot maintains a current map of the part of the grid that it has explored, relative to the start cell. At each cell, the robot proceeds depth-first, moving to the first (in some ordering, say, N, W, S, and then E) unobstructed edge-adjacent neighbor that it has not yet visited, adding an entry for the cell in the current map, and continuing. A cell is recognized as previously visited by checking for an entry for it in the current map. When there are no more unvisited unobstructed cells edge-adjacent to the current one, the robot backtracks (using information stored in the current map) to the cell explored before this one, and so on, until the robot backtracks to the initial cell and finds no more cells left to explore.

An upper bound for the number of moves used by the Depth-first Robot is $2n-2$, because each of the $n-1$ edges in a depth-first search tree for the grid graph corresponds to two moves of the robot, one in each direction. The auxiliary storage space used by the Depth-first Robot to store the current partial map in a straightforward fashion is $O(n \log n)$ bits; a reduction to $O(n)$ bits is possible at the expense of additional computation. This is considerably more than the $O(\log n)$ bits used by the Random Robot and the BK Robot. (Note that the Depth-first Robot does much more than localize; it constructs a complete map of the environment from scratch.)

6 The Homing Robot

The Homing Robot is a method that achieves almost the memory efficiency of the Random Robot and the BK Robot, and almost the move efficiency of the Depth-first Robot. It uses $O(\log n)$ bits of memory and localizes the robot in at most $4n$ moves.

The basic idea of the Homing Robot is to head as directly as possible for the outer boundary of the grid and then move around the outer boundary, guaranteeing visits to the northermost, easternmost, southernmost, and westernmost unobstructed cells. As shown in Section 3, this ensures the localization of the robot. In particular, the Homing Robot does not attempt to visit every unobstructed cell.

The problem with moving directly to the outer boundary is that the robot does not know where it is; there may be obstacles when it tries to follow a direct path. Our method for dealing with obstacles is based on Lumelsky and Stepanov's Bug algorithm for point-to-point navigation of a robot in an unknown environment [6]. It consists of circumnavigating the obstacle in order to find a good cell on its boundary from which to leave, that is, a cell that guarantees the robot will not encounter the obstacle again.

We pick a target direction – north, east, south, or west – in which the robot attempts to move towards the outer boundary. We assume for our exposition that the direction chosen is west. We also choose a direction – right or left – that the robot will turn to begin its circumnavigation of an obstacle. We arbitrarily choose left; this means that the robot will keep the edge of the obstacle to its right.

There are two different behaviors that the robot engages in: *go-west*, and *go-around*; its initial behavior is *go-west.* In the *go-west* behavior, the robot looks at the five neighboring cells that are at least as far west as the current cell, namely, the N, NW, W, SW, and S neighbors of the current cell. If none of these cells is obstructed, the robot makes a move west and then repeats the *go-west* behavior.

Otherwise, at least one of the five cells is obstructed. The robot then chooses one of the obstructed cells and begins the *go-around* behavior, which will cause it to circumnavigate the obstacle containing the chosen cell. During circumnavigation, the robot determines the H coordinate of the westernmost cell(s) on the boundary of the obstacle. When the robot detects that it is about to complete the circumnavigation, it returns to a westernmost cell on the boundary of the obstacle and resumes the *go-west* behavior.

Figure 3 shows the path of the Homing Robot in the grid of Figure 1 when started in the cell labelled A.

6.1 More on the *go-around* behavior

We now specify the *go-around* behavior of the robot in more detail. The main part of the procedure is the circumnavigation of an obstacle. The circumnavigation of an obstacle can revisit a boundary cell more than once, which means that more than the robot's position is required to keep track of its progress in the circumnavigation. In addition, the robot maintains its current heading in a variable D indicating which direction (N, W, S, or E) the robot is currently facing.

The robot begins its *go-around* behavior when at least one of the five cells N, NW, W, SW, or S

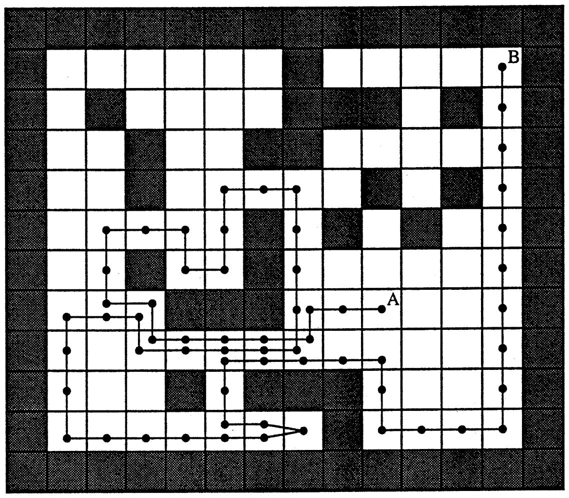

Figure 3: When started at A, the Homing Robot follows the path shown and localizes at B after 61 moves.

of the current cell is obstructed. It chooses one of these obstructed cells (arbitrarily); the obstacle containing the chosen cell will be circumnavigated. The current cell at the start of circumnavigation is designated as the *hit cell* for this instance of the *go-around* behavior.

Within the nine-cell square consisting of the current cell and its eight neighbors, the robot determines the connected component containing the chosen obstructed neighbor cell. There is a unique path P of moves through the square that keeps the edge of this connected component to its right. The initial heading D is chosen to be the direction of the move that would bring the robot to the current cell in the path P.

The robot saves the initial values of H, V, and D in H_0, V_0 and D_0. In addition to these three initial values, the robot keeps track of S_b, the number of moves it has made on the boundary, H_b, the minimum H value among all the boundary cells it has visited so far, and S_m, the number of moves made since the robot last visited a boundary cell with H value equal to H_b.

For each step of the circumnavigation, the robot first turns as far left as necessary to face an unobstructed cell, updating D accordingly. It then checks to see if moving in the direction of D would bring it back to its original position and heading (stored in H_0, V_0, and D_0.) If not, it makes a move in the direction of its current heading, D, and updates the values of H, V, S_b, H_b, and S_m. When the robot's next move would return it to its original position and heading, it has completed a circumnavigation of the obstacle boundary and, therefore, has visited every boundary cell of the obstacle.

The robot now follows the boundary of this obstacle back to a boundary cell with H value equal to H_b, in other words, to a westernmost cell on the boundary of this obstacle. To ensure that the number of moves used is at most half the number required to circumnavigate the obstacle, the robot compares S_m to half of S_b. If S_m is smaller, then the robot reverses direction, following boundary cells while keeping the edge of the obstacle to its left; otherwise, it continues in the same direction as the circumnavigation.

When the robot next reaches a boundary cell with H value equal to H_b, it resumes its *go-west* behavior. The cell at which it does this is designated as the *leave cell* for this instance of the *go-around* behavior.

The behaviors *go-west* and *go-around* alternate until localization is achieved. Recall that in the framework described in Section 3, the robot constantly updates the minimum and maximum H and V values for cells it has visited, and checks whether localization has been achieved.

6.2 Localization succeeds

The following theorem says the Homing Robot successfully localizes in any grid.

Theorem 1 *If the Homing Robot is started on any unobstructed cell of any grid of height h and width w, it will localize itself after a finite number of moves.*

Proof. If the robot ever begins its *go-around* behavior on the obstacle that is the border, then it traverses the outer boundary of the grid and successfully localizes by the time it completes the traversal, because the northernmost, easternmost, southernmost, and westernmost unobstructed cells are all part of the outer boundary.

Consider what happens when the robot begins its *go-around* behavior for a normal obstacle. Because a normal obstacle is interior to its boundary, there must be some boundary cell of the obstacle farther west than the hit cell. Thus, the leave cell will be strictly to the west of the hit cell for this instance of *go-around* behavior. Because the *go-west* behavior can only move the robot west, the hit cell for the next instance of *go-around* behavior (if any) must be at least as far west as the leave cell for this instance. By induction, the sequence of H coordinates of hit cells and leave cells in the order encountered by the robot must be nonincreasing, and strictly decreasing from a hit cell to a leave cell, as long as the robot continues to hit normal obstacles. This implies that the robot can hit a normal obstacle at most once, because after it leaves the normal obstacle for the first time, every subsequent hit cell is strictly west of every cell in the obstacle, and therefore cannot be a hit cell for this obstacle again.

Thus, the robot can engage in its *go-around* behavior for each normal obstacle at most once, and is guaranteed to localize if it engages in its *go-around* behavior for the border obstacle. During its *go-west* behavior, the robot cannot travel more than $w - 1$ moves west in total, because w is the width of the grid. Thus, the robot must localize after a finite number of moves. $\square$

6.3 Bounding the number of moves and the storage used

Now our goal is to show an upper bound of $4n$ on the number of moves made by the Homing Robot before it localizes. This is accomplished in a series of lemmas.

The first lemma gives a bound in terms of the number of nonboundary cells and the number of moves to circumnavigate every obstacle in the grid. Let b denote the number of cells that are on the boundary of one or more obstacles; then $(n - b)$ is the number of unobstructed cells not on the boundary of any obstacles. Let the obstacles in the grid be numbered from 1 to k, and for each i, let c_i be the number of moves for a robot to circumnavigate obstacle i. In other words, c_i is the number of moves used by a robot that moves around the boundary of the obstacle visiting every boundary cell and stopping when it returns to the initial cell.

Note that c_i may be larger than the number of boundary cells of obstacle i because some boundary cells may be visited more than once in a circumnavigation of the obstacle. For example, for the four-cell obstacle in Figure 3 the number of boundary cells is 18, but the value of c_i is 20.

Lemma 1 *Let M denote the total number of moves made by the Homing Robot before it localizes. Then*

$$M \leq (n-b) + \frac{3}{2}(c_1 + c_2 + \ldots + c_k).$$

Proof. The moves of the robot take it to cells of two types: (1) cells that are not obstacle boundary cells and (2) cells on the boundaries of obstacles.

The only time that the robot can move to a cell that is not on the boundary of an obstacle is during its *go-west* behavior. This can occur only when the robot is between the initial cell and the first hit cell or between a leave cell and the next hit cell on a subsequent obstacle. Because of the monotonicity of the H coordinate of the hit and leave cells, every cell moved to during the *go-west* behavior must have a different H coordinate. Therefore, every cell not on the boundary of an obstacle can be visited at most once by the robot. Thus, there are at most $(n-b)$ moves to cells of type (1).

We are now concerned with finding the total number of moves to boundary cells. Every move to a boundary cell either initiates or is part of the *go-around* behavior of the robot. The robot engages in the *go-around* behavior at most once for each obstacle. The number of moves made if the robot engages in the *go-around* behavior for obstacle i is bounded above by $\frac{3}{2}c_i$, because there are at most c_i moves to initiate the behavior and circumnavigate the obstacle (stopping before the last move), and at most $\frac{1}{2}c_i$ more moves made in returning to the leave cell. Considering all k obstacles, an upper bound on the number of moves to cells of type (2) is at most

$$\frac{3}{2}(c_1 + c_2 + \ldots + c_k).$$

The total number of moves made by the robot before it localizes is bounded by the sum of these two quantities, which shows

$$M \leq (n-b) + \frac{3}{2}(c_1 + c_2 + \ldots + c_k).$$

This establishes the lemma. □

A Labelling

In the interests of bounding $c_1 + c_2 + \ldots + c_k$, we introduce a particular labelling of the unobstructed cells of the grid. Each unobstructed cell in the grid gets a label from the set $\{0, 1, 2, 3, 4\}$ based on the pattern of obstruction of its eight neighbors. In particular, if within the square of nine cells consisting of the center and the eight neighbors, the obstructed cells form s connected components, then the label of the center cell is s. Figure 4 shows the labels of the cells in the grid from Figure 1. Note that the labelling depends only on the eight neighbors of the cell; for example, the cell labelled 4 is actually adjacent to only 3 different obstacles, but that is not apparent without looking beyond its eight neighbors.

Now for each $s = 1, 2, 3, 4$, let m_s be the number of cells that receive label s. From Figure 4, we see that $m_0 = 10$, $m_1 = 47$, $m_2 = 32$, $m_3 = 5$, and $m_4 = 1$ for the grid in Figure 1. Clearly,

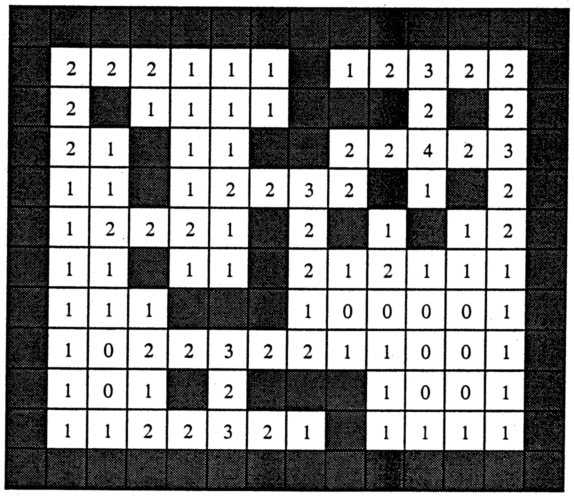

Figure 4: Labelling of the grid from Figure 1.

every boundary cell receives exactly one of the labels 1, 2, 3, or 4, so

$$m_1 + m_2 + m_3 + m_4 = b. \qquad (1)$$

The next lemma gives the relationship between this labelling and the quantity $c_1 + c_2 + \ldots + c_k$.

Lemma 2 $c_1 + c_2 + \cdots + c_k = m_1 + 2m_2 + 3m_3 + 4m_4$.

Proof. The key observation is that if we consider the moves made in circumnavigating all k obstacles, the number of moves made into a cell with label s is precisely s. This is easy to see for nonboundary cells; they receive label 0 and are not visited in the course of circumnavigating all k obstacles. Generally, if we look just at the square of nine cells with a given cell labelled s at the center, there will be s parts of circumnavigating paths entering and leaving the square, and each part of a path will move into the center cell once. Thus, each cell counted in m_s contributes a total of s moves to the total number of moves to circumnavigate all k obstacles, which proves the lemma. □

The next lemma indicates a relationship between a bound on the quantity $m_1+2m_2+3m_3+4m_4$ in terms of $m_1 + m_2 + m_3 + m_4$ and the total number of moves used by the Homing Robot. Our goal becomes to find such a bound for as small a value of R as possible.

Lemma 3 *Suppose that $R \geq 1$ is such that*

$$m_1 + 2m_2 + 3m_3 + 4m_4 \leq R(m_1 + m_2 + m_3 + m_4).$$

Then the total number of moves used by the Homing Robot to localize in a grid of n unobstructed cells is at most $\frac{3}{2}Rn$.

Proof. Note that Using Lemma 1, if M is the number of moves used by the Homing Robot in a grid with n unobstructed cells, then

$$M \leq (n - b) + \frac{3}{2}(c_1 + c_2 + \ldots + c_k).$$

Using equation (1) and Lemma 2, the assumption that $m_1+2m_2+3m_3+4m_4 \leq R(m_1+m_2+m_3+m_4)$ implies that

$$c_1 + c_2 + \ldots + c_k \leq Rb.$$

Thus

$$M \leq (n-b) + \frac{3}{2}Rb = n + (\frac{3}{2}R - 1)b.$$

By definition, $b \leq n$, and by assumption, $R \geq 1$, which implies that $(\frac{3}{2}R - 1) > 0$ and

$$M \leq n + (\frac{3}{2}R - 1)n \leq \frac{3}{2}Rn,$$

as claimed. □

For $R = 4$, it is immediate that

$$m_1 + 2m_2 + 3m_3 + 4m_4 \leq 4(m_1 + m_2 + m_3 + m_4)$$

because every coefficient on the lefthand side is at most 4. Lemma 3 then yields a bound of $6n$ on the total number of moves used by the Homing Robot. Our next lemma shows the inequality holds for $R = 3$, which yields the improved bound of $4.5n$ on the total number of moves used by the Homing Robot.

Lemma 4 *In any grid, if m_i is the number of cells labelled i, then*

$$m_1 + 2m_2 + 3m_3 + 4m_4 \leq 3(m_1 + m_2 + m_3 + m_4).$$

Proof. Consider any grid, and let m_i denote the number of cells labelled i for $i = 0, 1, 2, 3, 4$. We show that the number of cells labelled 4 is bounded by half the number of cells labelled 1 or 2, or, in other words,

$$m_4 \leq \frac{1}{2}(m_1 + m_2). \tag{2}$$

We note that an unobstructed cell whose N and S neighbors are obstructed must be labelled 1 or 2. This is because there cannot be more than two connected components formed by the obstructed neighbors of the cell, no matter which other neighbors of the cell are obstructed. This also holds for an unobstructed cell whose W and E neighbors are obstructed.

A cell can be labelled 4 only if the cell itself and its N, W, S, and E neighbors are unobstructed, and its NW, SW, SE, and NE neighbors are obstructed. By the preceding observation, this means that the cell's N, W, S, and E neighbors must each be labelled 1 or 2. We can define a one-to-two correspondence between any cell labelled 4 and its N and W neighbors. Notice that these two cells can be neither N nor W of any other cell labelled 4, which proves that there are at least twice as many cells labelled 1 or 2 as cells labelled 4, that is, inequality (2) is true. See Figure 5 for an illustration of the one-to-two correspondence.

To conclude the proof of the lemma, note that inequality (2) implies that

$$m_1 + 2m_2 + 3m_3 + 4m_4 \leq m_1 + 2m_2 + 3m_3 + 3m_4 + \frac{1}{2}(m_1 + m_2).$$

Thus,

$$m_1 + 2m_2 + 3m_3 + 4m_4 \leq \frac{3}{2}m_1 + \frac{5}{2}m_2 + 3m_3 + 3m_4,$$

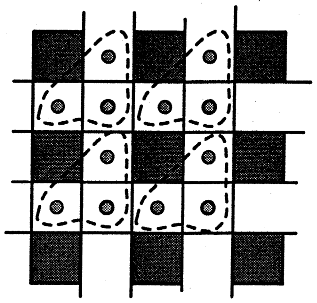

Figure 5: Illustration of the one-to-two correspondence between cells labelled 4 and their N and W neighbors, which are labelled 1 or 2.

which proves the lemma. □

Our final lemma of this subsection improves upon Lemma 4 to give $R = 8/3$ and a bound of $4n$ on the total number of moves of the Homing Robot. Lemma 8 in the Appendix shows that the constant $8/3$ cannot be further reduced.

Lemma 5 *In any grid, if m_i is the number of cells labelled i, then*

$$m_1 + 2m_2 + 3m_3 + 4m_4 \le \frac{8}{3}(m_1 + m_2 + m_3 + m_4).$$

Proof. Consider any grid, and let m_i be the number of cells labelled i, for $i = 0, 1, 2, 3, 4$. We subdivide the sets of cells labelled 1 and 2 as follows. For $i = 1, 2$, let m'_i be the number of cells labelled i that are N or W neighbors of a cell labelled 4, let m''_i be the number of cells labelled i that are N or W neighbors of a cell labelled 3, and let m'''_i be the number of cells labelled i that are not N or W neighbors of a cell labelled 3 or 4. Note that the N neighbor of a cell labelled 4 is the W neighbor of an obstructed cell, and the W neighbor of a cell labelled 4 is the N neighbor of an obstructed cell, so the set of cells N or W of a cell labelled 4 is disjoint from the set of cells N or W of a cell labelled 3. Hence we have

$$m_i = m'_i + m''_i + m'''_i$$

for $i = 1, 2$.

Because the one-to-two correspondence constructed in the preceding lemma involved all and only the cells labelled 1 or 2 that are N or W neighbors of cells labelled 4, the proof of the preceding lemma actually showed that

$$m_4 = \frac{1}{2}(m'_1 + m'_2). \tag{3}$$

The following inequality is proved in the Appendix, Lemma 7.

$$m_3 \le 2(m''_1 + m''_2). \tag{4}$$

Combining (3) and (4),

$$\frac{1}{3}m_3 + \frac{4}{3}m_4 \le \frac{2}{3}(m''_1 + m''_2 + m'_1 + m'_2),$$

so because $m_i = m'_i + m''_i + m'''_i$ for $i = 1, 2$,

$$\frac{1}{3}m_3 + \frac{4}{3}m_4 \le \frac{2}{3}m_1 + \frac{2}{3}m_2.$$

This implies that

$$m_1 + 2m_2 + 3m_3 + 4m_4 \leq \frac{5}{3}m_1 + \frac{8}{3}m_2 + \frac{8}{3}m_3 + \frac{8}{3}m_4,$$

which proves the lemma. □

We bound the storage used by the Homing Robot as follows.

Lemma 6 *If the Homing Robot is started on any unobstructed cell of a grid with n unobstructed cells, it uses $O(\log n)$ bits of storage to localize itself.*

Proof. The auxiliary storage used by the Homing Robot consists of thirteen registers storing integers in the interval $[-4n, 4n]$, for which $O(\log n)$ bits suffice. □

We prove our main theorem for the Homing Robot.

Theorem 2 *For every integer $n \geq 1$ and for every grid with n unobstructed cells, if the Homing Robot is started on any unobstructed cell of the grid, it localizes itself using $O(\log n)$ bits of auxiliary storage and at most $4n$ moves.*

Proof. By Lemma 3 and Lemma 5, if M is the total number of moves made by the Homing Robot on a grid with n unobstructed cells then

$$M \leq \frac{3}{2} \cdot \frac{8}{3}n = 4n.$$

Lemma 6 shows that $O(\log n)$ bits of storage suffice. □

6.4 A lower bound for the Homing Robot

Although we have shown an upper bound of $4n$ on the number of moves used by the Homing Robot in a grid with n unobstructed cells, we have been unable to construct a class of grids for which the Homing Robot uses more than $3n$ moves. Consider the class of grids exemplified in Figure 6. Let k and m be positive integers. In the grid $G_{k,m}$ there are m chevron-shaped normal obstacles, each consisting of $2k + 1$ obstructed cells. Figure 6 shows $G_{4,5}$. The number of unobstructed cells is $n = 4km + 8m + 4k + 4$.

We assume the Homing Robot is started in the cell labelled A. The robot follows the dotted path shown in Figure 6 until it reaches the cell labelled B. This path consists of $12k+7$ moves. The robot then repeats this path around each of the other normal obstacles, for a total of $m(12k + 7)$ of moves until it begins to move around the outer boundary. As k and m increase without bound, we see that the average number of moves per unobstructed cell approaches 3.

The gap between $3n$ and $4n$ leads us to ask what the true worst-case number of moves is for the Homing Robot. Our conjecture is that the upper bound can be reduced. Lemma 8 in the Appendix suggests that a different proof technique would be required to show this.

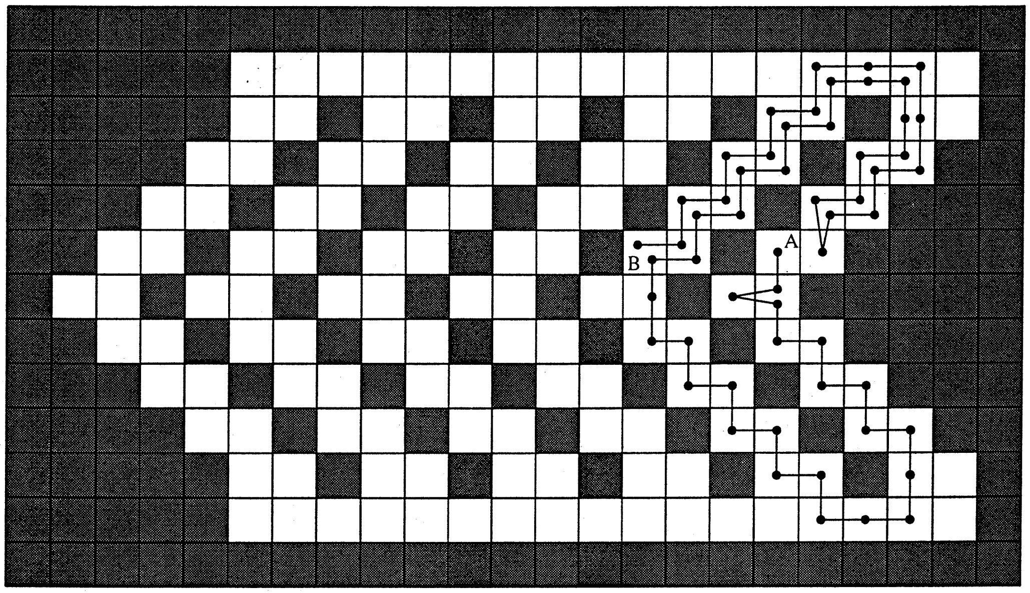

Figure 6: Example of a class of grids in which the Homing Robot takes nearly $3n$ moves to localize.

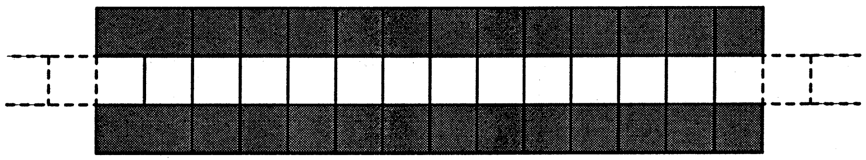

Figure 7: An infinitely long corridor of white cells.

7 A general lower bound for deterministic localization

Allowing any deterministic localization algorithm, what is the worst-case number of moves required to localize a robot in any grid with n unobstructed cells? The Depth-first Robot shows that this number is at most $2n-2$. We show that at least $n-3$ moves are required for any deterministic localization algorithm. Note that this bound does not apply to randomized algorithms like the Random Robot.

Theorem 3 *For any integer $n \geq 3$, there exists a grid with n unobstructed cells such that any deterministic localization algorithm requires that a robot make at least $n-3$ moves from some initial position.*

Proof. Construct a grid consisting of an infinitely long horizontal row of unobstructed cells, bordered to the north and south by obstructed cells (see Figure 7). Place a robot on any of the unobstructed cells in this corridor; it may move in only two directions, east and west. Let it run for $n-4$ moves using any deterministic algorithm. At this point, the robot will have visited at most $n-3$ cells, including the one in which it started. Now truncate the corridor at both ends so that it no longer includes cells that the robot did not visit. The grid now consists of at most $n-3$ unobstructed cells in a row, surrounded on all four sides by the border. Replace the two obstructed cells at the eastern and western ends of the corridor with unobstructed cells. Let p denote the number of unobstructed cells in this grid; we know that $p \leq n-1$.

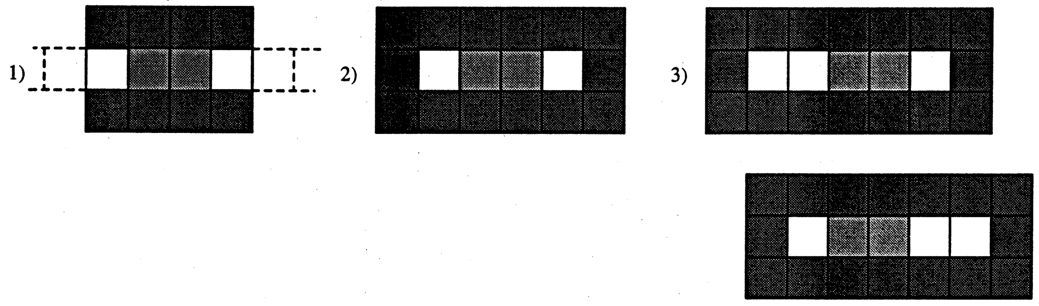

Figure 8: Light gray cells represent those visited by the robot. 1) The robot begins in the leftmost light gray cell and makes one move to the east. 2) White cells are added at both ends of truncated corridor. 3) The two possible grids that result from adding a white cell to either end. We see that the robot cannot determine whether it has halted in the third cell or the fourth cell.

We now consider the two different grids of size n constructed by (1) adding one unobstructed cell at the western end, and $n - 1 - p$ at the eastern end, or (2) adding one unobstructed cell at the eastern end and $n - 1 - p$ at the western end.

Figure 8 illustrates the moves involved in this construction for the case when $n = 5$ and $p = 4$. We can think of these two grids as representing two possible grids that the robot could have started out in; another way of thinking about these grids is of representing the robot's two possible current positions in a single grid. Therefore, after $n - 4$ moves, the robot cannot distinguish between two possible positions. $\square$

8 Grids and Finite State Machines

A grid can be transformed into a finite state machine that also represents the environment of the robot. A finite state machine obtainable in this way is a *grid machine.*

Given a grid, the grid machine obtained from it has one state corresponding to each unobstructed cell in the grid. The output of the grid machine at a given state represents the pattern of obstructed and unobstructed cells among the neighbors of the cell represented by the state. The output is a vector of eight bits, corresponding to the eight neighbors of the cell, with 1 representing unobstructed, and 0 representing obstructed. A fixed ordering is chosen for the neighbors, for example, N, NW, W, SW, S, SE, E, NE. The inputs to the grid machine are N, W, S, and E, representing possible moves of the robot. If the neighboring cell to the north is unobstructed, then the input N takes the machine to the corresponding state, otherwise, the input N keeps the machine in its current state, and similarly for E, S, and W.

Because of the complementary nature of the finite state machine representing the robot's environment, the robot's inputs (sensations) are the environment's outputs and the robot's outputs (actions) are the environment's inputs. See Figure 9 for a small grid and its corresponding grid machine, where the inputs that are self-loops have been omitted for clarity.

The problem of robot localization in a known grid is equivalent to the problem of finding an *adaptive homing sequence* for the corresponding grid machine. For unrestricted finite state machines, Hibbard [5] shows that for every integer n there exists a finite state machine M of n states such that any adaptive homing sequence for M must have length at least $n(n - 1)/2$.

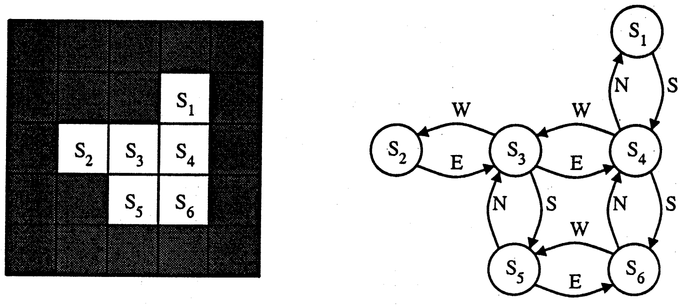

Figure 9: A grid and the corresponding grid machine. The output at a state is determined by the obstructed (0) or unobstructed (1) status of the eight neighbors. The output at S_1 is 00011000, at S_2 is 00000110, and so on.

However, in the case of grid machines, we discover that odometry (the ability of the robot to determine relative geometric positions in its environment) helps. The bound for the Depth-first Robot shows that every grid machine with n states has an adaptive homing sequence of length at most $2n - 2$. The lower bound in Section 7 shows that there are grid machines for which a shortest adaptive homing sequence must have length at least $n - 3$. The gap between these bounds suggests the following open question. What is the worst-case minimum length of an adaptive homing sequence for a grid machine with n states?

9 Conclusion

The Homing Robot is a simple localization algorithm for a robot in a grid environment. It uses very little storage, $O(\log n)$ bits, and does not require a map of its environment, only the values of its height and width. This property could be useful in a situation where there are many very simple robots in a common environment communicating with a more complex central controller. The worst-case number of moves used by the Homing Robot is bounded above by $4n$, which is not much worse than the Depth-first Robot's $2n - 2$.

If the robot (or its central controller) is permitted enough storage to represent a map, there is a simple optimization that allows possible early termination of any localization method. This is to keep track of the set P of cells on the map that are consistent with all the observations made by the robot. P is initialized to the set of cells in the map that have the same pattern of obstruction of neighbors as the robot observes from its initial cell. After a move, say to the W, P is updated as follows. Let the set P' be initially empty. For each cell in P, include its western neighbor in P' if it has the same pattern of obstruction of neighbors as observed from the current cell. After all the cells in P have been considered, set P to P'. When P contains just one cell, the robot has localized.

If this optimization is included, then typically the robot will localize long before it has visited northernmost, westernmost, southernmost, and easternmost boundary cells. However, the "bad" environment in Figure 6 still causes the Homing Robot to use nearly $3n$ moves, since it may have to circumnavigate all but two of the normal obstacles to localize, even with the early termination criterion. It is an interesting question to find robust and practical algorithms for this variant of the problem.

The following questions may be promising areas for progress.

- Determine the worst-case number of moves of the Homing Algorithm more accurately. We have shown it is between $3n$ and $4n$.
- Find a localization algorithm that uses $O(\log n)$ bits of memory and substantially fewer than $4n$ moves.
- Find more accurate bounds on the worst-case length of adaptive homing sequences for grid machines of n states. We have shown it is between $n - 3$ and $2n - 2$.
- (From [2]) Find an algorithm to search (visit every cell of) a grid that uses $O(\log n)$ bits of memory and substantially less than $O(n^2)$ moves.
- Generalize this model in an interesting way to incorporate errors in the robot's sensations and actions.

10 Acknowledgments

Most of the results in this paper were presented by the first author in her Senior Thesis in Applied Mathematics at Yale University in May, 1998. We would like to thank David Eisenstat for his suggestion and proof of the localization criterion in Section 3 and other helpful discussions.

References

[1] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 218–223. IEEE, 1979.

[2] Manuel Blum and Dexter Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, pages 132–142. IEEE, 1977.

[3] G. Dudek, K. Romanik, and S. Whitesides. Localizing a robot with minimum travel. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 437–446. ACM, 1995.

[4] D. Fox, W. Burgard, S. Thrun, and A. Cremers. Position estimation for mobile robots in dynamic environments. In *Proceedings AAAI*, pages 983–988, 1998.

[5] Thomas N. Hibbard. Least upper bounds on minimal terminal state experiments for two classes of sequential machines. *Journal of the Association of Computing Machinery*, 8:601–612, 1961.

[6] Vladimir J. Lumelsky and Alexander A. Stepanov. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Transactions on Automatic Control*, AC-31:1058–1062, 1986.

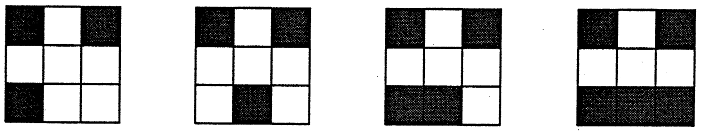

Figure 10: Every cell labelled 3 has one of these four neighborhoods, up to rotation and reflection.

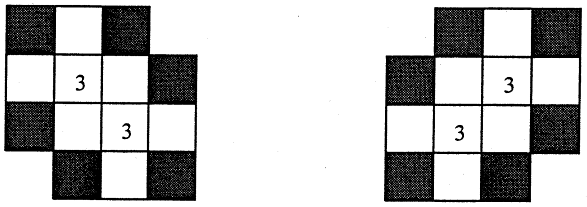

Figure 11: A type I cell can have a neighbor labelled 3 in one of two possible ways.

Appendix

We provide a proof of inequality (4) from Section 6. We also show that the constant of 8/3 in Lemma 5 cannot be reduced.

Lemma 7 *Consider any grid. Let m_3 be the number of cells labelled 3. For $i = 1, 2$, let m_i'' be the number of cells labelled 1 or 2 that are N or W neighbors of cells labelled 3. Then*

$$m_3 \leq 2(m_1'' + m_2'').$$

Proof. We first consider cells labelled 3 in more detail. For any cell labelled 3, its neighborhood can be rotated and reflected as necessary to obtain one of the four neighborhoods shown in Figure 10.

A cell labelled 3 is of *type I* if none of its edge-adjacent neighbors is obstructed. The first neighborhood shown in Figure 10 centers on a cell of type I. A cell labelled 3 is of *type II* if exactly one of its edge-adjacent neighbors is obstructed. The other three neighborhoods shown in Figure 10 center on cells of type II. We subdivide the type II cells into II-N, II-W, II-S, and II-E, depending on which edge-adjacent neighbor (N, W, S, or E) of the cell is obstructed.

Now we consider how two cells of type 3 can be neighbors. A type I cell can be adjacent to at most one other cell labelled 3, and that cell must be of type I and be corner-adjacent but not edge-adjacent. The only two possibilities are shown in Figure 11.

Type II cells can only be neighbors of type II cells. If they are edge-adjacent vertically, they must be of types II-W and II-E, and if they are edge-adjacent horizontally, they must be of types II-N and II-S. These possibilities are shown in Figure 12.

In addition, there are four possible ways for type II cells to be corner-adjacent diagonally. These are shown in Figure 13. Note in particular that a type II-N cell cannot have a SW neighbor labelled 3, and a type II-W cell cannot have a NE neighbor labelled 3.

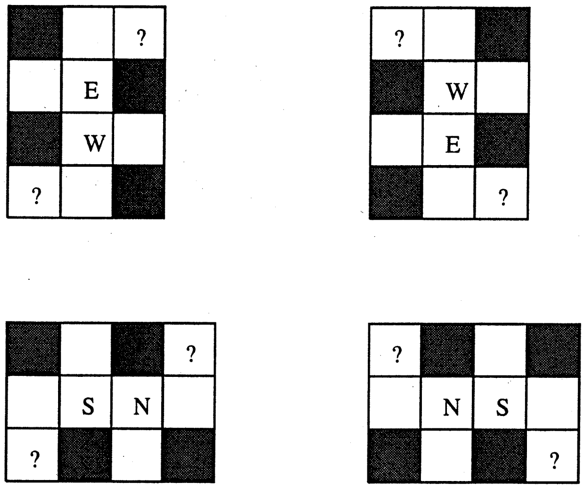

Figure 12: Type II cells can be edge-adjacent vertically or horizontally. The labels N, W, S, and E indicate types II-N, II-W, II-S, and II-E. Cells labelled with ? may be obstructed or unobstructed.

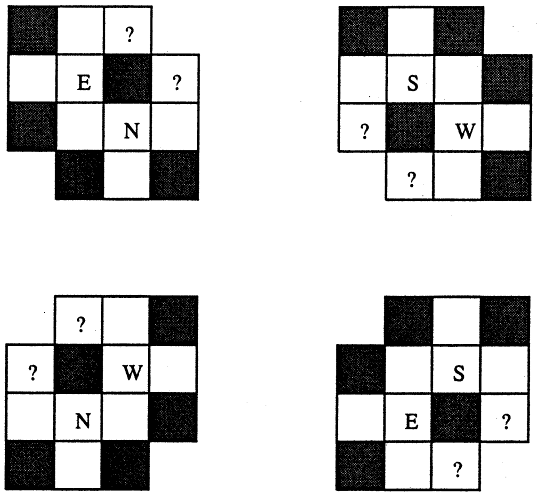

Figure 13: Type II cells can be corner-adjacent diagonally in four possible ways. The labels N, W, S, and E indicate types II-N, II-W, II-S, and II-E. Cells labelled with ? may be obstructed or unobstructed.

To prove Lemma 7, we divide all the cells labelled 3 in the grid into four kinds of disjoint sets as follows.

(1) Type I singletons, cells of type I that have no neighbors labelled 3.

(2) Type I pairs, pairs of cells of type I that are adjacent diagonally.

(3) Maximal sets of horizontally adjacent cells of type II.

(4) Maximal sets of vertically adjacent cells of type II.

For each such set S_i of cells labelled 3, we associate another set T_i of cells labelled 1 or 2 that are N or W neighbors of cells in S_i. We show that the sets T_i are pairwise disjoint, and that

$$|S_i| \leq 2|T_i|.$$

If we add up these inequalities over all i, the lefthand side is just the total number of cells labelled 3, namely m_3, and the right-hand side is at most $2(m_1'' + m_2'')$, which proves the lemma.

We now specify how T_i is determined from S_i, and prove that it has the required properties.

(1) Suppose S_i is a singleton type I cell, $\{C\}$. There are four unobstructed edge-adjacent neighbors of C labelled 1 or 2. Thus at least one of them, say C' is a N or W neighbor of C. In this case, let $T_i = \{C'\}$. The cell C' cannot also be a N or W neighbor of any other cell labelled 3, for that would require that C have a neighbor labelled 3, which would violate the definition of a singleton type I cell. We have $|S_i| \leq 2|T_i|$.

(2) Suppose S_i consists of a type I pair. There is at least one cell, say C', that is labelled 1 and is a N or W neighbor of at least one cell of the pair. (Refer to Figure 11.) In this case, let $T_i = \{C'\}$. The cell C' is not a N or W neighbor of any cell labelled 3 outside the pair. We have $|S_i| \leq 2|T_i|$.

(3) Suppose S_i is a maximal set of horizontally adjacent cells of type II. Then S_i must consist of an alternating row of type II-N and type II-S cells. First we put all the N neighbors of the type II-S cells from S_i into T_i; these cells are labelled 1 or 2. Then, if the leftmost cell in the row is of type II-N, we put its W neighbor into T_i; this cell is also labelled 1 or 2. Clearly, all the N neighbors of type II-S cells from S_i are not W of any cell labelled 3. If the leftmost cell in the row is of type II-N, then its W neighbor cannot be N of any cell labelled 3, for otherwise, a type II-N cell would have a SW neighbor labelled 3, which is impossible, as indicated in Figure 13. Thus, in either case, the cells in T_i cannot be N or W neighbors of cells labelled 3 that are not in S_i. To see that $|S_i| \leq 2|T_i|$, we note that if the leftmost cell in the row is of type II-S, then there are at least as many type II-S cells in the row as type II-N cells. If the leftmost cell in the row is of type II-N, then there is at most one fewer type II-S cells in the row than type II-N cells, and we add the W neighbor of the leftmost cell to T_i.

(4) Suppose that S_i is a maximal set of vertically adjacent cells of type II. This case is analogous to (3). S_i must consist of an alternating column of type II-W and type II-E cells. To construct T_i we first take all the W neighbors of the type II-E cells in the column. Then, if the topmost cell in the column is of type II-W, we add its N neighbor to the set T_i. Note that if we add this cell to T_i, then it is not W of any cell labelled 3, because a type II-W cell cannot have a NE neighbor labelled 3, as indicated in Figure 13. We argue as in the preceding case that $|S_i| \leq 2|T_i|$ and that

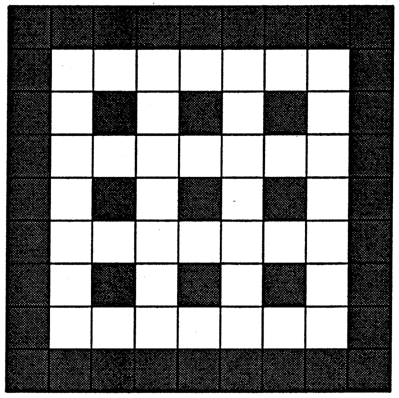

Figure 14: G_3, an example of a class of grids for which the ratio of c to b approaches 8/3.

the cells in T_i are labelled 1 or 2 and are not N or W neighbors of cells labelled 3 that are not in S_i. This concludes the proof of Lemma 7. □

Finally, we give a construction to show that the constant of 8/3 in Lemma 5 cannot be reduced.

Lemma 8 *There is a class of grids G_k such that if m_i is the number of cells labelled i in G_k then as k goes to infinity the ratio of*

$$c = m_1 + 2m_2 + 3m_3 + 4m_4$$

to

$$b = m_1 + m_2 + m_3 + m_4$$

approaches 8/3.

Proof. Consider a class of square grids of height and width $2k + 1$ with k^2 normal obstacles, each consisting of one cell, and a square outer boundary of $8k$ cells. G_3 is shown in Figure 14.

In G_k there are k^2 normal obstacles, and each of them takes 8 moves to circumnavigate. Also, it takes $8k$ moves to circumnavigate the border. Applying Lemma 2,

$$c = 8k^2 + 8k.$$

Every unobstructed cell in the grid is a boundary cell, and there are

$$b = (2k + 1)^2 - k^2 = 3k^2 + 4k + 1$$

unobstructed cells in G_k. Clearly, the ratio c/b approaches 8/3 as k increases without bound. □