Gridworlds as Testbeds for Planning with Incomplete Information

Craig Tovey and **Sven Koenig**
College of Computing, Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{ctovey, skoenig}@cc.gatech.edu

Abstract

Gridworlds are popular testbeds for planning with incomplete information but not much is known about their properties. We study a fundamental planning problem, localization, to investigate whether gridworlds make good testbeds for planning with incomplete information. We find empirically that greedy planning methods that interleave planning and plan execution can localize robots very quickly on random gridworlds or mazes. Thus, they may not provide adequately challenging testbeds. On the other hand, we show that finding localization plans that are within a log factor of optimal is NP-hard. Thus there are instances of gridworlds on which all greedy planning methods perform very poorly, and we show how to construct them. These theoretical results help empirical researchers to select appropriate planning methods for planning with incomplete information as well as testbeds to demonstrate them.

Introduction

Testbeds (prototypical test domains) are planning domains that allow researchers to evaluate their planning methods, communicate performance results of their methods to others, interpret published performance results of others more easily, and compare their methods against these performance results (Hanks, Pollack, & Cohen 1993). Testbeds should be easy to describe, but they should also provide a wide enough variety to mimic real domains. In particular, testbeds must include cases that are not too easy to solve because otherwise planning methods would appear to be more efficient than they actually are in some of the domains of interest. Consequently, planning researchers have studied in detail the properties of their testbeds for planning with complete information, such as blocksworlds and sliding tile puzzles. Examples of such experimental and theoretical studies include (Gupta & Nau 1992; Reinefeld 1993; Slaney & Thiébaux 1996; Koenig & Simmons 1996).

In recent years, planning researchers have become interested in planning with incomplete information. This is an important research direction because, in the real world, complete information is often not available. Gridworlds appear to be by far the most frequently used testbeds for this work. However, not much is known about their properties.

In this paper, we therefore investigate whether gridworlds are good testbeds for planning with incomplete information. We study localization tasks, which are fundamental planning tasks for robots. We find experimentally that greedy planning methods that interleave planning and plan execution can localize robots very quickly on gridworlds with random obstacles or random mazes. Thus, random gridworlds or mazes may not provide adequately challenging cases to push the state of the art. Although the theoretical planning community has shown the complexity of planning tasks with incomplete information to be difficult in general (Littman 1994; Madani, Hanks, & Condon 1999), the reported success of current greedy methods on some gridworlds (Nourbakhsh 1997; Koenig & Simmons 1998a) and our experiments on random gridworlds and mazes reported here suggest that the constrained topology of gridworlds may make them easy to solve. However, we analyze the performance of one greedy planning method in detail, namely the Delayed Planning Architecture (Genesereth & Nourbakhsh 1993), and show that there exist gridworlds on which its performance is poor. Furthermore, we prove that localization even with only suboptimal worst-case performance is NP-hard. Thus there are instances of gridworlds on which *all* greedy planning methods perform very poorly, and we show how they can be constructed. In general, our results improve the understanding of previously used planning methods and testbeds for planning with incomplete information and help empirical researchers to select appropriate planning methods as well as testbeds to demonstrate them.

Gridworlds

We study planning with incomplete information in gridworlds of the kind shown in Figure 1. Gridworlds are finite rectangular areas of square cells. Each cell can be either traversable or untraversable. A robot is always in exactly one cell. It starts in a traversable cell and can then always move north, east, south, or west. Gridworlds have been used as testbeds for different planning methods. Planning tasks with a known start cell and deterministic movement have often been modeled as traditional graph search problems. Planning tasks with nondeterministic movement and automatic sensing that determines the current cell uniquely have been modeled as totally observable Markov decision process models (Dean *et al.* 1993). Planning tasks with

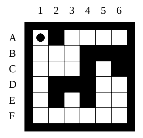

Figure 1: Simple Gridworld

incompletely known start cell or nondeterministic movement, and nondeterministic sensing or sensing on demand have been modeled as partially observable Markov decision process models (McCallum 1995; Hansen 1997). Planning tasks with incompletely known start cell, deterministic movement, and automatic deterministic sensing have been modeled as AND-OR search tasks (Nourbakhsh 1997; Koenig & Simmons 1998a). In this paper, we present a first analysis of the last case.

The Gridworld Planning Tasks

We study localization tasks in gridworlds. Localization is a prototypical planning task with incomplete information. The robot knows a map of the gridworld but does not know its start cell. Evidently, the robot may need to localize prior to performing many other tasks. The sensors on-board the robot tell it in every cell whether the cells immediately adjacent to it in the four compass directions (north, east, south, west) are traversable. (The border of the gridworld is untraversable and observed as such.) The robot can then move one cell to the north, east, south, or west, unless that cell is outside of the gridworld or untraversable (in which case the robot remains in its current cell). We assume that there is no uncertainty in actuation and sensing and that the robot always knows its orientation from the on-board compass. These assumptions are simplifying but sufficiently close to reality to enable one to use the resulting planning methods on real robots (Nourbakhsh 1996).

The robot is localized if it knows its current cell. A deterministic (randomized) localization plan specifies (a probability distribution for) the movement to execute based on all previous movements and observations. A localization plan is valid iff, no matter which cell the robot is started in, it eventually prints out its current cell or correctly determines that localization is impossible. The objective of planning then is to determine a valid deterministic or randomized localization plan that minimizes the expected number of movements for the worst possible start cell (the "worst-case expected performance"), in the following sense: We first calculate the expected number of movements for each possible start cell. The expectation is only important for probabilistic plans and is taken with respect to the randomization of the probabilistic plans. The worst-case expected performance is then the maximum of these values.

Modeling the Planning Tasks

The gridworld planning tasks can be modeled as tree search tasks. The states of the tree search tasks are sets of cells, corresponding to cells that the robot could be in. For example, if the robot has no knowledge of its start cell for the gridworld

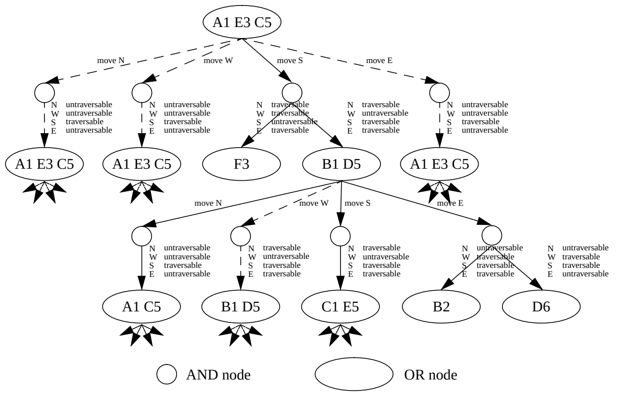
A1 E3 C5
move N
move W
move S
move E
N untraversable W untraversable S traversable E untraversable
N untraversable W untraversable S traversable E untraversable
N traversable W traversable S untraversable E traversable
N traversable W untraversable S traversable E traversable
N untraversable W untraversable S traversable E untraversable
A1 E3 C5
A1 E3 C5
F3
B1 D5
A1 E3 C5
move N
move W
move S
move E
N untraversable W untraversable S traversable E untraversable
N traversable W untraversable S traversable E traversable
N traversable W untraversable S traversable E traversable
N untraversable W traversable S traversable E traversable
N untraversable W traversable S traversable E untraversable
A1 C5
B1 D5
C1 E5
B2
D6
AND node
OR node

Figure 2: AND-OR Search Tree

planning task from Figure 1 but observes untraversable cells in all compass directions except to its south, then the start state of the robot contains three possible start cells: A1, E3, and C5. In each state, the robot can choose a movement ("OR" nodes of the search tree) and, if it does not stop, then makes a new observation ("AND" nodes of the search tree). For example, Figure 2 shows the beginning of a search tree for the gridworld planning task. (The dashed part of the tree is unnecessary and could be deleted.) A deterministic localization plan assigns each OR node a movement ("deterministic OR node"). Similarly, a randomized localization plan assigns each OR node a probability distribution over the movements ("randomized OR node"). The following theorem shows that valid randomized localization plans cannot perform better than valid deterministic localization plans.

Theorem 1 *No valid randomized localization plan has a better worst-case expected performance than a valid deterministic localization plan with optimal worst-case performance.*

Proof Sketch: Given a valid randomized localization plan, transform it into a valid deterministic localization plan by iteratively pruning all but the best alternative, in the worst-case sense, from the lowest nondeterministic node. ■

This theorem implies that there is no point in having robots flip coins (that is, move nondeterministically) to localize them in gridworlds with optimal worst-case (expected) performance. In the following, we therefore consider only deterministic localization plans. The tree search tasks in this case are AND-OR search tasks and valid localization plans are decision trees.

Solving the Planning Tasks Greedily

Nourbakhsh and Genesereth noticed experimentally that using a complete AND-OR search to find valid localization plans with optimal worst-case performance for gridworld planning tasks was completely infeasible but that planning methods that interleave planning and plan execution could efficiently find valid localization plans with good worst-case performance for their gridworlds (Genesereth & Nourbakhsh 1993; Nourbakhsh 1997). Their Delayed Planning

Table 1: Random Gridworlds

gridworld size	obstacle density	av. number of subplans	av. number of steps per subplan	av. total number of steps
11×11	9.9 %	3.1	1.0	3.2
	29.8 %	1.7	1.0	1.8
	49.6 %	1.2	1.1	1.3
	70.2 %	0.6	1.0	0.7
	90.1 %	0.0	1.0	0.0
31×31	10.0 %	5.7	1.1	6.1
	30.0 %	3.0	1.0	3.2
	50.0 %	2.3	1.1	2.5
	70.0 %	1.3	1.1	1.5
	90.0 %	0.3	1.0	0.3
51×51	10.0 %	7.2	1.1	7.5
	30.0 %	3.7	1.0	3.9
	50.0 %	2.8	1.1	3.1
	70.0 %	1.6	1.2	1.1
	90.0 %	0.3	1.0	0.3

Table 2: Acyclic Mazes

gridworld size	obstacle density	av. number of subplans	av. number of steps per subplan	av. total number of steps
11×11	41.3 %	2.4	1.5	3.6
21×21	45.4 %	3.3	1.7	5.4
31×31	46.8 %	3.8	1.7	6.6
41×41	47.6 %	4.1	1.8	7.5
51×51	48.1 %	4.5	1.8	8.0
61×61	48.4 %	4.7	1.8	8.6
71×71	48.6 %	4.9	1.9	9.1

Architecture with the viable plan heuristic uses breadth-first search (iterative deepening) in the deterministic part of the state space around the current state in conjunction with pruning rules to find a subplan (movement sequence) that reduces the number of possible robot cells with the smallest number of steps (movements). The robot executes the subplan and then repeats the process until it is localized or detects that localization is impossible. Subsequently, Koenig and Simmons developed a generalization of the Delayed Planning Architecture (Koenig & Simmons 1998a). The Delayed Planning Architecture has been applied to a variety of planning problems with incomplete information, including the Bay Area Transit Problem (Hsu 1990) and the Tool Box Problem (Olawsky, Krebsbach, & Gini 1993). In the context of gridworlds, it has been demonstrated experimentally by their authors on real robots and in simulation.

We re-implemented the Delayed Planning Architecture and performed experiments in gridworlds with random obstacles, averaged over 1000 runs with randomly selected start cells. The results in Table 1 show that current planning methods perform very well on random gridworlds. No matter what the size or obstacle density of the gridworld is, the robot can gain information with only slightly more than one move on average, and localizes in a small number of total moves. Table 2 shows that similar results also hold for random acyclic mazes that were generated by depth-first search (using code provided by Joseph Pemberton). Therefore, we need new gridworlds to push the state of the art. Furthermore, the following theorem shows that the worst-case performance of the Delayed Planning Architecture can be extremely suboptimal, much worse than the experimental results in random gridworlds and mazes suggest. All of our example gridworlds are connected and not completely sym-

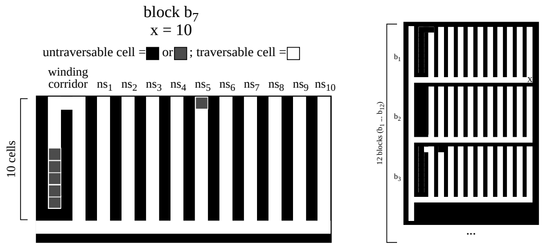
block b_7
x = 10
untraversable cell = ■ or ▨; traversable cell = □
winding corridor
ns_1 ns_2 ns_3 ns_4 ns_5 ns_6 ns_7 ns_8 ns_9 ns_{10}
10 cells
12 blocks (b_1 ... b_{12})
b_1
b_2
b_3
x
...

Figure 3: Greedy Performance

metrical (localization is possible). This demonstrates that our lower bounds already hold for these kinds of gridworlds.

Theorem 2 *The worst-case performance of plans generated by the Delayed Planning Architecture can be a factor of $\Omega(\sqrt[3]{s})$ worse than the optimal worst-case performance, where s is the number of cells of the gridworld, even in gridworlds that are connected (every cell can be reached from every other cell).*

Proof Sketch: We construct a gridworld on which the Delayed Planning Architecture has the worst-case performance ratio claimed in the theorem. The gridworld contains many copies of rectangular "blocks" of size $(2x + 4) \times (x + 2)$. Along the south side of each block is a wall of length $2x + 4$ and immediately to the north of the wall is an east-west corridor of length $2x + 4$. There are x north-south corridors $ns_1 \ldots ns_x$ of length x each, separated by walls, that branch off of the east-west corridor to the north, starting in the sixth column of the block. To their immediate left is a winding corridor that goes up x cells, goes left two cells, and then goes down $x - 2$ cells. Figure 3 (left) shows an example. The gridworld consists of a column of $x + 2$ blocks, from block b_1 on top to block b_{x+2} at the bottom. In the extreme west, we add a full length north-south hallway, which makes the gridworld connected. Figure 3 (right) shows an example. We make the last cell of north-south corridor ns_{x-i} of block b_{x+2-i} untraversable, for all $0 \leq i \leq x - 1$. We also make the last i cells of the winding corridor of block b_{x+2-i} untraversable, for all $0 \leq i \leq x + 1$. This completes the description of how the gridworld is constructed. Clearly, the gridworld is connected and has $s = (2x+5)(x+2)(x+2) = \Theta(x^3)$ cells (which does not include the untraversable border of the gridworld).

The robot can find the beginning of some winding corridor from any starting point with at most $3x + 2$ movements and then move at most $2x - 1$ into the winding corridor, counting its length, which identifies the block and thus localizes the robot. Thus, the worst-case number of movements in an optimal plan is at most $5x + 1 = \Theta(x)$. Now we show that the Delayed Planning Architecture performs many more than $\Theta(x)$ movements if the robot starts at the east end of the east-west corridor of block b_1 (in the figure: marked X). When the robot is started, it knows where it is with the exception of which block it is in. The robot makes $x - 1$ movements into north-south corridor ns_x because this is the fastest way of reducing the number of possible robot cells. At this point the robot can eliminate block b_{x+2}. The robot then returns to the east-west corridor and makes $x-1$ movements into north-south corridor ns_{x-1}, at which point it can eliminate block b_{x+1}, and so on. Finally, it makes $x-1$ movements into the winding corridor, at which point it can eliminate block b_2 and has localized. The robot has made a total number of movements equal to $2x^2 + x - 1 = \Theta(x^2)$.

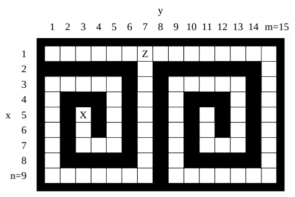

Figure 4: Two Spirals

It follows that the worst-case performance of plans generated by the Delayed Planning Architecture is $\Omega(x^2/x) = \Omega(x) = \Omega(\sqrt[3]{s})$ worse than the worst-case performance of an optimum plan. ■

Analysis of the Planning Tasks

The Delayed Planning Architecture can be applied to finding homing sequences or adaptive homing sequences for deterministic finite state automata whose states are colored, a concept from theoretical computer science. A homing sequence is a linear plan (movement sequence) with the property that the state colors observed during its execution uniquely determine the resulting state (Kohavi 1978), and it is known that finding a shortest homing sequence is NP-hard in general (Schapire 1992). It could be the case that the constrained topology of gridworlds makes them easy to solve and thus that valid localization plans with optimal worst-case performance for the gridworld planning tasks can be found in polynomial time. We prove, however, that finding valid localization plans even with suboptimal worst-case performance in gridworlds is NP-hard and thus that there are instances of gridworlds on which the Delayed Planning Architecture does not perform well. The following theorem shows that it is easy to find valid localization plans in gridworlds.

Theorem 3 *(Part 1:) For every gridworld of size $m \times n$, there exists a valid localization plan that executes $O(mn)$ movements and that can be found in time $O(mn)$. (Part 2:) This result is the best possible in the sense that there exist gridworlds of size $m \times n$ in which every valid localization plan must execute $\Omega(mn)$ movements and can only be found in time $\Omega(mn)$, even in gridworlds that are connected.*

Proof Sketch: For Part 1, first determine the connected components M_i of the given map of the gridworld. Second, acquire a map M' of the gridworld component where the robot is, by moving the robot in a depth-first search manner. Third, determine which of the M_i are identical to map M', by depth-first search of every map, starting from the west-most cell of the north-most traversable cells. If exactly one M_i matches M', the robot has been localized. If more than one M_i matches M' then the robot cannot localize. This algorithm is correct, needs time $O(mn)$, and executes $O(mn)$ movements. For Part 2, construct a gridworld consisting of two spirals as shown in Figure 4. The gridworld is connected. If the start cell is at the end of one of the spirals (in the figure: X), then any valid localization plan has to execute $\Omega(mn)$ movements (in the figure: move to cell Z) before it can distinguish which spiral it is in. Thus, every valid localization plan must execute $\Omega(mn)$ movements and can only be found in time $\Omega(mn)$. ■

Theorem 3 leaves open the possibility that optimum localization plans may be so complex that they cannot be encoded in polynomial length. Fortunately, the following theorem shows that finding valid localization plans with optimal worst-case performance is in NP.

Theorem 4 *Determining whether there exists a valid localization plan that executes no more movements than a given value is in NP.*

Proof Sketch: First, use the method from Theorem 3 to find the start cells from which the robot cannot localize. Second, guess a deterministic localization plan (decision tree). Third, simulate a fictitious robot that executes the localization plan on the map of the gridworld from each possible start cell, verifying that the number of movements is smaller than the given value, and that the localization plan yields a correct answer, when it is possible to localize. Therefore, a decision tree can be guessed and verified in polynomial time, *provided that the decision tree has polynomial size.* We now ensure this provision to complete the proof. The algorithm needs to guess only decision trees that have as many leaf nodes as there are possible start cells, that is, no more than mn. This is so because there is at most one branch from the root to a leaf for each start cell. Since there are at most mn leaves, there are at most $mn - 1$ AND nodes with two or more children. The algorithm needs to guess only decision trees that have, on any branch, at most $mn - 1$ AND nodes with only one child between two AND nodes with two or more children. If there were more, then a belief state would repeat on that branch and the part of the branch between two repeating belief states (including one of them) could be cut out. Thus, the decision tree has $O(m^2n^2)$ AND nodes and thus $O(m^2n^2)$ nodes, which is at most quadratic in the size of the problem description. ■

The following theorem, our main theorem, shows that finding valid localization plans even with suboptimal worst-case performance is NP-hard.

Theorem 5 *It is NP-hard to find a valid localization plan in gridworlds of size $m \times n$ whose worst-case performance is within a factor $O(\log(mn))$ of optimum, even in gridworlds that are connected.*

Proof Sketch: An instance of set cover consists of a base set $S = \{e_1 \dots e_x\}$ and a collection of sets $S_1, \dots, S_y \subseteq S$. A set cover is a collection of these sets whose union is S, and the objective is to find a set cover of small cardinality. Finding a set cover whose cardinality is within a factor $O(\log x)$ of minimum is NP-hard (Lund & Yannakakis 1994). Let $y^* \leq x$ denote the cardinality of a minimum set cover for the given instance of the set cover problem. We reduce this problem to finding a valid localization plan in a gridworld of size $m \times n$ with $m = 3x^3y + 1$ and $n = (xy+2)(x+1)$ whose worst-case performance is within a factor $O(\log(mn))$ of optimal. We assume without loss of generality that $\log y = O(\log x)$.

We now explain how the gridworld is constructed from the given instance of the set cover problem. The gridworld contains many copies of rectangular "blocks" of size $(3y) \times (xy + 2)$. Along the south side of each block is a wall of length $3y$ and immediately to the north of the wall is an east-west corridor of length $3y$. There are y north-south corridors $ns_1 \dots ns_y$ of length xy each, separated by walls, that branch off of the east-west corridor to the north, starting in the second column of the block. Figure 5 shows an example. The gridworld contains an array of $(x^3) \times (x + 1)$ blocks. Thus,

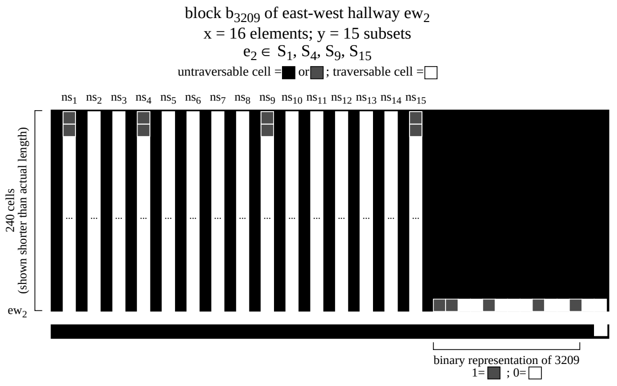
block b_{3209} of east-west hallway ew_2
x = 16 elements; y = 15 subsets
$e_2 \in S_1, S_4, S_9, S_{15}$
untraversable cell = ■ or ▨; traversable cell = □
ns_1 ns_2 ns_3 ns_4 ns_5 ns_6 ns_7 ns_8 ns_9 ns_{10} ns_{11} ns_{12} ns_{13} ns_{14} ns_{15}
240 cells (shown shorter than actual length)
ew_2
binary representation of 3209
1= ▨ ; 0= □

Figure 5: Block for NP-Completeness Proof

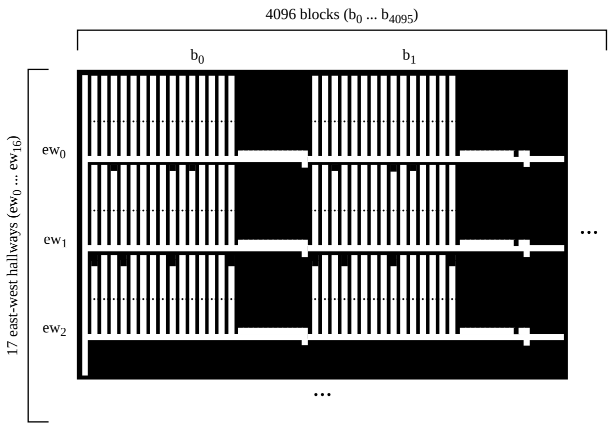
4096 blocks (b_0 ... b_{4095})
b_0
b_1
17 east-west hallways (ew_0 ... ew_{16})
ew_0
ew_1
ew_2
...
...

Figure 6: Gridworld for NP-Completeness Proof

there are x^3 blocks $b_0 \ldots b_{x^3-1}$ in the same row. Their east-west corridors form one long east-west hallway. There are $x+1$ east-west hallways *ew*$_0 \ldots$ *ew*$_x$ of length $3x^3y$ each. In the extreme west, we add a full length north-south hallway, which makes the gridworld connected. Figure 6 shows an example. We make the last i cells of north-south corridor *ns*$_j$ of each block in east-west hallway *ew*$_i$ untraversable iff $e_i \in S_j$. (Since there is no element e_0, no north-south corridor of any block in east-west hallway *ew*$_0$ is shortened.) To be able to distinguish between the blocks in the same east-west hallway, we put a "signature" at the east end of each block . For block b_k, this signature encodes k in binary form, which needs at most $3 \log x$ bits. The signature is in the form of northerly "alcoves," followed by a southerly alcove which marks the beginning of the signature. This completes the description of how the gridworld is constructed in polynomial time.

We now calculate an upper bound z^U on the number of movements of a valid localization plan with optimal worst-case performance. Consider the following localization plan: If only north is unblocked, move north one place (since the robot was in a southerly alcove). Otherwise, move south until the robot sees an opening to the west or east (the robot is now in an east-west hallway), then move east to the end of a signature or until the robot gets blocked (the robot is now directly east of a signature). Move west and read the signature. At this point, the robot knows where it is with the exception of which east-west hallway it is in. The robot then moves west and, every time it encounters one of the y^* north-south corridors in the current block that corresponds to a smallest set cover

Table 3: Gridworlds used to Prove Theorem 2

gridworld size	obstacle density	av. number of subplans	av. number of steps per subplan	av. total number of steps
11×25	50.2 %	4.5 (4)	2.3 (5)	10.2 (20)
13×36	50.2 %	5.9 (5)	2.9 (7)	16.9 (35)
15×49	50.2 %	7.4 (6)	3.2 (9)	23.7 (54)
17×64	50.2 %	8.9 (7)	3.4 (11)	30.6 (77)
19×81	50.2 %	10.4 (8)	4.0 (13)	42.0 (104)
21×100	50.1 %	11.5 (9)	4.4 (15)	50.0 (135)
23×121	50.1 %	13.4 (10)	4.5 (17)	60.4 (170)
25×144	50.1 %	14.4 (11)	4.9 (19)	71.1 (209)
27×169	50.1 %	16.0 (12)	5.2 (21)	82.5 (252)
29×196	50.1 %	18.0 (13)	5.4 (23)	98.0 (299)
31×225	50.1 %	19.4 (14)	5.7 (25)	110.5 (350)
33×256	50.1 %	20.8 (15)	5.8 (27)	121.5 (405)
35×289	50.1 %	22.5 (16)	6.1 (29)	137.7 (464)

for the given instance of the set cover problem, it moves to the end of the north-south corridor and back to the east-west hallway. If the robot is in east-west hallway *ew*$_j$ with $j > 0$ then it will visit at least one north-south corridor that is shorter than xy. Its length uniquely identifies the east-west hallway the robot is in, which localizes the robot. Otherwise the robot must be in east-west hallway *ew*$_0$ and is localized as well. Thus, the localization plan is valid. An easy calculation shows that the total number of movements is bounded by $z^U \leq 2y^*xy + 6y \leq 3y^*xy$.

It remains to be shown that a solution to the localization problem implies a solution to the set cover problem. Assume that we have found a valid localization plan whose performance is within a factor $O(\log(mn))$ of optimal. An upper bound on the number of movements of this localization plan is $O(\log(mn))z^U = O(\log((3x^3y+1)(xy+2)(x+1)))3y^*xy = O(\log(x^5y^2))3y^*xy = O(5\log(x) + 2\log(y))3y^*xy = O(7\log(x))3y^*xy = O(\log(x))3y^*xy \leq x3y^*xy \leq 3x^3y$. Thus, the number of movements is no larger than the length of an east-west hallway. Now assume that the robot starts at the east end of east-west hallway *ew*$_0$. Thus, it cannot visit a different east-west hallway and, as part of the localization, must determine that no north-south corridor in a block is shorter than xy. If the robot moves into a north-south corridor less than $xy-x-1$, it cannot detect whether the corridor is shorter than xy because all north-south corridors are at least $xy-x$ long. Thus, consider all north-south corridors that the robot moves into at least $xy-x-1$. The collection of subsets that these corridors correspond to must be a set cover, for otherwise the robot could not distinguish between the east-west hallways *ew*$_0$ and *ew*$_i$ for the elements e_i not covered by the collection of subsets. Let y' denote the cardinality of this set cover. To determine how close to minimum the set cover is, we determine a lower bound on the total number of movements of the robot. A straightforward calculation shows that the robot makes at least $(2y'-1)(xy-x-1)$ moves. Combined with the $O(\log(x))3y^*xy$ upper bound shown earlier, this implies that $y' = O(\log(x))y^*$, which implies that the set cover is within a factor $O(\log(x))$ of minimum. ■

To summarize, Theorem 3 shows that is easy to find valid plans in gridworlds, while Theorems 4 and 5 together show that it is NP-hard to find valid localization plans with optimal or even near-optimal worst-case performance. Combining Theorems 4 and 5 also tells us that the problem stated in Theorem 4 is NP-complete.

Testbeds

As part of the theoretical results in the previous two sections we constructed hard instances of gridworlds on which greedy planning methods, such as the Delayed Planning Architecture, do not perform well. These gridworlds can be used in test suites in addition to random gridworlds or mazes. Table 3 contains the results of the same experiments that Table 1 reported on, except that we now use the Delayed Planning Architecture in conjunction with the gridworlds that we constructed as part of the proof of Theorem 2 instead of random gridworlds. (The numbers in parentheses refer to the particularly bad case used in the proof, where the robot starts at the east end of the east-west corridor of the top-most block.) Clearly, the robot now needs to execute a larger number of movements to reduce the number of possible cells than in random gridworlds or mazes, and the total number of movements is larger than for random gridworlds or mazes of comparable sizes and obstacle densities as well. Similarly, the gridworlds that we constructed as part of the proof of Theorem 5 also provide challenging testbeds.

Conclusions

While testbeds for planning with complete information have been studied extensively, this paper provides a first study of testbeds for planning with incomplete information. We studied localization tasks in gridworlds. Previous experimental work had shown that greedy planning methods, such as the Delayed Planning Architecture (Genesereth & Nourbakhsh 1993; Nourbakhsh 1997), can efficiently find valid localization plans with good performance in random gridworlds and mazes. Our theoretical analysis showed that it is easy to find valid localization plans in arbitrary gridworlds but, perhaps surprisingly, NP-hard to find valid localization plans even with only suboptimal worst-case performance in some gridworlds, even if the gridworlds are connected. This suggests that gridworlds are appropriate testbeds for planning with incomplete information. As part of our proofs, we also showed how to construct hard instances of gridworlds on which the Delayed Planning Architecture and all other greedy planning methods do not perform well at all. These gridworlds can be used in addition to random gridworlds and mazes. In the future, we intend to apply similar ideas to localization tasks where the robots use partially observable Markov decision process models (Koenig & Simmons 1998b).

Acknowledgments

The Intelligent Decision-Making Group is supported by an NSF Career Award under contract IIS-9984827 to Sven Koenig and DARPA's Mobile Autonomous Robot Software (MARS) Program under contract DARPA/SMDC DASG60-99-C-0081.

References

Dean, T.; Kaelbling, L.; Kirman, J.; and Nicholson, A. 1993. Planning with deadlines in stochastic domains. In *Proceedings of the National Conference on Artificial Intelligence*, 574–579.

Genesereth, M., and Nourbakhsh, I. 1993. Time-saving tips for problem solving with incomplete information. In *Proceedings of the National Conference on Artificial Intelligence*, 724–730.

Gupta, N., and Nau, D. 1992. On the complexity of blocks-world planning. *Artificial Intelligence* 56(2–3):223–254.

Hanks, S.; Pollack, M.; and Cohen, P. 1993. Benchmarks, test beds, controlled experimentation, and the design of agent architectures. *AI Magazine* 14(4):17–42.

Hansen, E. 1997. Markov decision processes with observation costs. Technical Report CMPSCI 97–01, Department of Computer Science, University of Massachusetts, Amherst (Massachusetts).

Hsu, J. 1990. Partial planning with incomplete information. In *Proceedings of the AAAI Spring Symposium on Planning in Uncertain, Unpredictable, or Changing Environments*.

Koenig, S., and Simmons, R. 1996. Easy and hard testbeds for real-time search algorithms. In *Proceedings of the National Conference on Artificial Intelligence*, 279–285.

Koenig, S., and Simmons, R. 1998a. Solving robot navigation problems with initial pose uncertainty using real-time heuristic search. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, 154–153.

Koenig, S., and Simmons, R. 1998b. Xavier: A robot navigation architecture based on partially observable Markov decision process models. In Kortenkamp, D.; Bonasso, R.; and Murphy, R., eds., *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*. MIT Press. 91–122.

Kohavi, Z. 1978. *Switching and Finite Automata Theory*. McGraw-Hill, second edition.

Littman, M. 1994. Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*.

Lund, C., and Yannakakis, M. 1994. On the hardness of approximating minimization problems. *Journal of the ACM* 41:960–981.

Madani, O.; Hanks, S.; and Condon, A. 1999. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the National Conference on Artificial Intelligence*, 541–548.

McCallum, R. 1995. Instance-based utile distinctions for reinforcement learning with hidden state. In *Proceedings of the International Conference on Machine Learning*, 387–395.

Nourbakhsh, I. 1996. *Robot Information Packet*. Distributed at the AAAI-96 Spring Symposium on Planning with Infomplete Information for Robot Problems.

Nourbakhsh, I. 1997. *Interleaving Planning and Execution for Autonomous Robots*. Kluwer Academic Publishers.

Olawsky, D.; Krebsbach, K.; and Gini, M. 1993. An analysis of sensor-based task planning. Technical Report 93-94, Computer Science Department, University of Minnesota, Minneapolis (Minnesota).

Reinefeld, A. 1993. Complete solution of the eight-puzzle and the benefit of node ordering in IDA*. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 248–253.

Schapire, R. 1992. *The Design and Analysis of Efficient Learning Algorithms*. MIT Press.

Slaney, J., and Thiébaux, S. 1996. Linear-time near-optimal planning in the blocks world. In *Proceedings of the National Conference on Artificial Intelligence Planning*.