Greedy Localization*

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

Abstract

In this paper, we show that finding localization plans with optimal worst-case execution time for localization tasks with short-range sensors in discretized domains is NP-hard, *even within a logarithmic factor*. This strongly suggests that finding and executing localization plans with optimal or *even near-optimal worst-case execution time* cannot be done in polynomial time. Greedy localization methods interleave planning and execution and keep the amount of planning performed between moves small. We analyze one such greedy localization method, the Delayed Planning Architecture, and show that it can find and execute localization plans in polynomial time and thus substantially reduce the sum of planning and execution time compared to localization methods that find localization plans with optimal or near-optimal execution time. We also characterize how suboptimal the execution time of its localization plans can be. These results provide a first step towards analyzing other greedy localization methods.

1 Introduction

A mobile robot cannot predict the observation that it will make after its next moves if it does not know its current position. Localization tasks are therefore planning tasks in nondeterministic domains. In principle, one could solve them by first determining plans with optimal execution time and then executing them. However, finding such plans can be time-consuming. One principle that has often been used in mobile robotics to reduce the sum of planning and execution time is to interleave planning and execution, which is often called sensor-based planning [4]. The sensors on-board a robot can typically sense the terrain only near its current position, and the robot thus has to interleave planning and execution to be able to sense new parts of the terrain. As the robot moves, it is able to reduce its uncertainty about its position and thus the number of situations that its plans have to cover. This makes subsequent planning more efficient.

*This project is partly supported by an NSF award under contract IIS-0098807. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations and agencies or the U.S. government. The results of this paper overlap with some of our previous results in [24].

Greedy localization methods interleave planning and execution and keep the amount of planning performed between moves small. They often use agent-centered search to restrict planning to the part of the domain around the current state. Agent-centered search methods decide on the local search space, search it, and determine which moves to execute within it. Then, they execute these moves (or only the first move) and repeat the overall process from their new state. Agent-centered search methods have the following advantages.

- **Theoretical foundation:** Unlike many existing ad-hoc planning methods that interleave planning and execution, many agent-centered search methods have a solid theoretical foundation, that allows one to characterize their behavior analytically. For example, they are often easy to prove correct and their execution time can be analyzed formally.
- **Anytime property:** Many agent-centered search methods allow for fine-grained control over how much planning to perform between moves by varying the sizes of their local search spaces. Thus, they can be used as any-time contract algorithms [18] for determining which moves to execute next, which allows them to adjust the amount of planning performed between moves to the planning and execution speeds of robots.
- **Simple integration into robot architectures:** Many agent-centered search methods are simple to implement and integrate well into complete robot architectures. Agent-centered search methods are robust towards the inevitable inaccuracies and malfunctions of other architecture components and do not need to have control of the robot at all times. This is important because planning methods should only provide advice on how to act and work robustly even if that advice is ignored from time to time [1].

Agent-centered search methods have been demonstrated on mobile robots that solve complex real-world tasks, including localization and mapping, and perform well in practice [13]. This paper studies localization tasks with short-

range sensors in discretized domains, whereas the literature typically studies localization tasks with long-range sensors in nondiscretized domains. In the literature, localization sometimes just means to find all positions consistent with the current sensor reports [2, 3, 7, 8, 10, 11, 22]. In this paper, however, localization means to determine how to move the robot until it knows its current position, similar to [5, 20]. The Delayed Planning Architecture [6] is a greedy localization method that uses agent-centered search but had not been analyzed before. We show that it can find and execute localization plans in polynomial time and substantially reduces the sum of planning and execution time compared to localization methods that find localization plans with optimal or near-optimal worst-case execution time. Since the Delayed Planning Architecture moves the robot before it knows the complete consequences, it incurs some overhead in execution time but this is outweighed by a reduction in planning time since plans with optimal or near-optimal worst-case execution time cannot be found in polynomial time. We also characterize how suboptimal the execution time of its localization plans can be. Our results complement previous results on the complexity of localization that are typically not about greedy localization methods.

2 Localization Tasks

We study localization tasks where a robot with short-range sensors operates in a known gridworld (a rectangular configuration of square cells) but does not know its start cell. The sensors on-board the robot tell it in every cell which of the four neighboring cells (north, east, south, west) are untraversable. (The border of the gridworld is untraversable and observed as such.) The robot can then move one cell in one of the four compass directions 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 [16].

The robot is localized if it knows its current cell. A localization plan specifies which moves to execute based on all previous moves and observations, until the robot is either localized or correctly determines that localization is impossible. We measure the execution time of a localization plan using the total number of moves. The worst-case execution time of a localization plan then is the total number of moves for the worst possible start cell. We measure the quality of a localization plan using its worst-case execution time and thus differ from those approaches in the literature that use its competitive ratio instead, that is, the ratio of its execution time and the smallest execution time that would allow an omniscient robot to verify its true initial cell [5, 9, 11, 20].

3 Non-Greedy Localization

In this section, we analyze how difficult it is to find localization plans with optimal or near-optimal worst-case execution time. Since the robot does not know its start cell, localization tasks cannot be formulated as planning tasks in small deterministic domains whose states are the cells (*position space*). Rather, the robot has to maintain a belief about its current cell, namely the set of cells that it could possibly be in. The beliefs of the robot depend on its observations, which the robot cannot predict with certainty since it is uncertain about its cell. Localization tasks are therefore planning tasks in large nondeterministic domains whose states are the beliefs of the robot (*belief space*).

Localization tasks are related 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 (sequence of moves) with the property that the state colors observed during its execution uniquely determine the resulting state [14]. An adaptive homing sequence is a conditional plan with the same property [19]. For every reduced deterministic finite state automaton with s states, there exists a (possibly suboptimal) homing sequence with $O(s^2)$ moves (and thus polynomial execution time) that can be found in polynomial planning time but finding a shortest homing sequence is NP-hard in general [19]. (A reduced finite state automaton is one where, for every pair of different states, there exists some sequence of moves that distinguishes them.) Robot localization tasks can be solved with homing sequences since the position space is deterministic and thus can be modeled as a deterministic finite state automaton. However, the following proposition suggests that the constrained topology of gridworlds could make them easier to solve.

Proposition 1 *For every gridworld with s cells, there exists a (possibly suboptimal) localization plan with $O(s)$ moves that can be found in $O(s)$ time.*

Proof Sketch: For every gridworld with s cells, one can first determine the connected components M_i of the given map of the gridworld in $O(s)$ planning time. Second, one can acquire a map M' of the gridworld component where the robot is in $O(s)$ moves, by moving the robot in a depth-first search manner. Third, one can determine in $O(s)$ planning time which of the M_i are identical to map M' using a depth-first search for 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

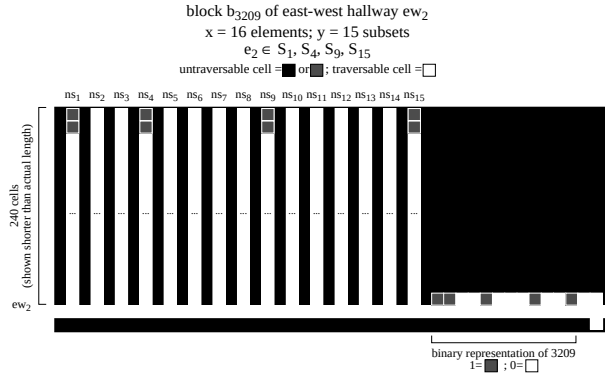

Figure 1: Block for NP-Hardness Proof

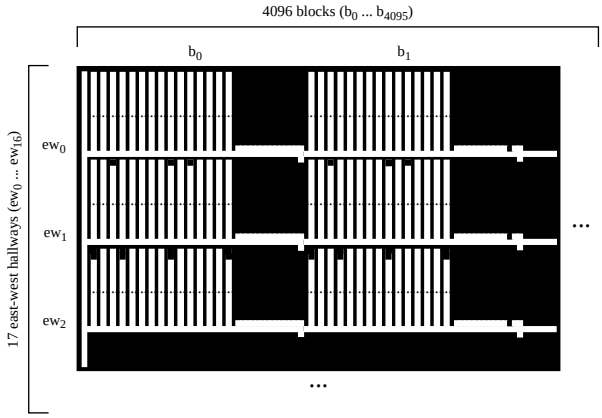

Figure 2: Gridworld for NP-Hardness Proof

M_i matches M' then the robot cannot localize. This algorithm needs $O(s)$ planning time and executes $O(s)$ moves. ■

This theorem suggests that the constrained topology of gridworlds indeed makes them easier to solve and thus that localization plans with optimal worst-case execution time could possibly be found in polynomial planning time. However, we prove in the following that finding localization plans even with near-optimal worst-case execution time remains NP-hard. (Additional results can be found in [24].)

Theorem 1 *Finding a localization plan in gridworlds with s cells whose worst-case execution time is within a factor $O(\log(s))$ of optimum is NP-hard, 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 [15]. 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 localization plan in a gridworld of size $m \times n$ with $m = 3x^3y + 1$ and $n = (xy + 2)(x + 1)$ whose worst-case execution time 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 1 shows an example. The gridworld contains an array of $(x^3) \times (x + 1)$ blocks. Thus, there are x^3 blocks $b_0 \dots b_{x^3-1}$ in the same row. Their east-west corridors form one long east-west hallway. There are $x + 1$ east-west hallways $ew_0 \dots ew_x$ of length $3x^3y$ each. In the extreme west, we add a full length north-south hallway, which makes the gridworld connected. Figure 2 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 planning time.

We now calculate an upper bound z^U on the number of moves of a localization plan with optimal worst-case execution time. Consider the following localization plan: If only north is traversable, 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 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. An easy calculation shows that the total number of moves 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 localization plan whose execution time is within a factor $O(\log(mn))$ of optimal. An upper bound on the number of moves 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 moves 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 moves 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. ■

It had been shown previously that finding localization plans with optimal worst-case execution time for localization tasks with long-range sensors in continuous domains is NP-hard [5]. Our result not only shows that finding localization plans with optimal worst-case execution time for localization tasks with short-range sensors in discretized domains is NP-hard but also that finding localization plans with near-optimal execution time for such localization tasks remains NP-hard. This strongly suggests that finding localization plans with optimal or even near-optimal worst-case execution time cannot be done in polynomial planning time for such localization tasks, which confirms earlier empirical results that indicated that performing a complete minimax (and-or) search in belief space to determine plans with optimal worst-case execution time is often completely infeasible [17]. On the other hand, the localization method sketched as part of the proof of Proposition 1 finds and executes localization plans in polynomial time and thus reduces the sum of planning and execution time substantially compared to localization methods that find localization plans with optimal or near-optimal execution time. However, it does not make good use of prior knowledge or the sensor reports during execution to reduce the number of moves and thus often results in large execution times in practice. Furthermore, it is not simple to integrate into complete robot architectures, for example, if the obstacle avoidance component can alter the move recommendations of the localization component. In the following, we therefore show that a greedy localization method that performs agent-centered search and thus avoids these problems also finds and executes localization plans in polynomial time.

4 Greedy Localization

The Delayed Planning Architecture with the viable plan heuristic [6] always finds a linear plan (sequence of moves) that reduces the number of possible robot cells with the smallest number of moves. The robot executes the plan and then repeats the process. Thus, it performs agent-centered search. Nourbakhsh pioneered the Delayed Planning Architecture in robot programming classes where Nomad 150 mobile robots had to navigate gridworlds that were built with three-foot high and forty inch long cardboard walls [16]. Subsequently, Koenig and Simmons generalized the Delayed Planning Architecture [12].

The Delayed Planning Architecture achieves the same behavior as homing sequences and can thus find and execute localization plans in polynomial time. Theoretical results about homing sequences suggest how to implement the Delayed Planning Architecture with dynamic programming methods to guarantee a polynomial planning time [14]. In practice, however, its implementations use a simple breadth-first search in the deterministic part of the belief space around the current belief state [6]. This also guarantees a polynomial planning time, as the following proposition shows.

Proposition 2 *For every gridworld with s cells, the Delayed Planning Architecture finds a localization plan with $O(s^2)$ moves in $O(s^3)$ time.*

Proof Sketch: For every gridworld with s cells there are $O(s)$ planning episodes because the start belief contains at most s cells and every planning episode eliminates at least one possible cell from the belief state. Each planning episode, in turn, can be completed in $O(s^2)$ planning time by breadth-first search since $O(s)$ belief states can be reached from the current belief state without eliminating at least one possible cell and expanding each belief state takes $O(s)$ planning time since it contains $O(s)$ cells. The robot then executes $O(s)$ moves since it reduces the number of possible robot cells with the smallest number of moves. Thus, the Delayed Planning Architecture needs $O(s^3)$ planning time and executes $O(s^2)$ moves. ■

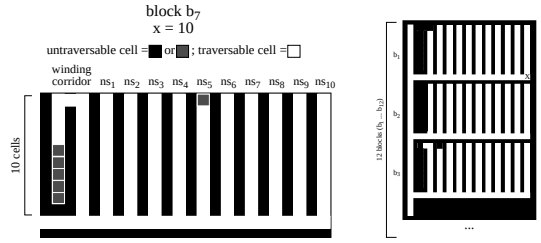

Figure 3: Gridworld for the Delayed Planning Architecture

Theorem 1 strongly suggests that there are instances of gridworlds for which no localization method with polynomial planning time, including the Delayed Planning Architecture, can find localization plans with optimal or near-optimal worst-case execution time. The following theorem provides a lower bound on the worst-case execution time of localization plans found by the Delayed Planning Architecture compared to the optimal worst-case execution time.

Theorem 2 *The worst-case execution time of plans generated by the Delayed Planning Architecture in gridworlds with s cells can be a factor of $\Omega(\sqrt[3]{s})$ worse than the optimal worst-case execution time.*

Proof Sketch: We construct a gridworld on which the Delayed Planning Architecture has the worst-case execution time 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 $3x + 2$ moves 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 moves in an optimal plan is at most $5x + 1 = \Theta(x)$. Now we show that the Delayed Planning Architecture performs many more than $\Theta(x)$ moves 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 $x - 1$ moves 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 $x - 1$ moves into north-south corridor ns_{x-1}, at which point it can eliminate block b_{x+1}, and so on. Finally, it makes $x - 1$ moves into the winding corridor, at which point it can eliminate block b_2 and has localized. The robot has made a total number of moves equal to $2x^2 + x - 1 = \Theta(x^2)$. It follows that the worst-case execution time of plans generated by the Delayed Planning Architecture is $\Omega(x^2/x) = \Omega(x) = \Omega(\sqrt[3]{s})$ worse than the worst-case execution time of an optimum plan. ■

5 Future Work

We have assumed in our analysis that there is no actuator or sensor noise. This is a reasonable assumption in some environments. For example, we mentioned earlier that the Delayed Planning Architecture has been used on Nomad 150 mobile robots. The success rate of moving was at least 99.57 percent in these environments, and the success rate of making the correct observations in all four directions simultaneously was at least 99.38 percent [16]. These large success rates enable one to ignore actuator and sensor noise, especially since the rare failures are usually quickly noticed when the number of possible cells drops to zero, in which case the robot simply reinitializes its belief state to all possible cells and then continues to use the localization method unchanged. In less constrained environments, however, it is important to take actuator and sensor noise into account, resulting in localization methods based on partially observable Markov decision process models (POMDPs). POMDP-based navigation architectures have been shown to result in very reliable navigation [21]. A recent overview can be found in [23]. We are currently working on extending our analysis to this case, which also requires us to move from a worst-case analysis to an average-case analysis. If it turns out that the restricted topology of gridworlds makes POMDPs easier to solve, then it would be possible to develop better localization methods for POMDP-based navigation architectures.

6 Conclusions

We showed that finding localization plans with optimal worst-case execution time for localization tasks with short-range sensors in discretized domains is NP-hard, even within a logarithmic factor. This strongly suggests that finding and executing localization plans with optimal or even near-optimal worst-case execution time cannot be done in polynomial time. Greedy localization methods interleave planning and execution and keep the amount of planning performed between moves small. They often use agent-centered search to restrict planning to the part of the domain around the current position of the robot and thus have a solid theoretical foundation, allow for fine-grained control over how much planning to perform between moves, and are simple to integrate into robot architectures. The Delayed Planning Architecture is a greedy localization method that uses agent-centered search but had not been analyzed before. We showed that it can find and execute localization plans in polynomial time and thus substantially reduces the sum of planning and execution time compared to localization methods that find localization plans with optimal or near-optimal execution time. We also characterized how suboptimal the execution time of its localization plans can be. These results provide a first step towards analyzing other greedy localization methods, including greedy localization methods for POMDP-based navigation architectures.

References

[1] P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the National Conference on Artificial Intelligence*, pages 268–271, 1987.

[2] D. Avis and H. Imai. Locating a robot with angle measurements. *Journal of Symbolic Computation*, 10:311–326, 1990.

[3] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Transactions on Robotics and Automation*, 13(2):251–263, 1997.

[4] H. Choset and J. Burdick. Sensor based planning and nonsmooth analysis. In *Proceedings of the International Conference on Robotics and Automation*, pages 3034–3041, 1994.

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

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

[7] L. Guibas, R. Motwani, and P. Raghavan. The robot localization problem in two dimensions. In *Proceedings of the Symposium on Discrete Algorithms*, pages 259–268, 1992.

[8] L. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics*, pages 269–282, 1995.

[9] C. Icking and R. Klein. Competitive strategies for autonomous systems. In H. Bunke, T. Kanade, and H. Noltemeier, editors, *Modelling and Planning for Sensor-Based Intelligence Robot Systems*, pages 23–40, 1995.

[10] O. Karch and H. Noltemeier. Robot localization – theory and practice. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 850–856, 1997.

[11] J. Kleinberg. The localization problem for mobile robots. In *Proceedings of the Annual Symposium on Foundations of Computer Science*, pages 521–533, 1994.

[12] S. Koenig and R.G. Simmons. Solving robot navigation problems with initial pose uncertainty using real-time heuristic search. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 145–153, 1998.

[13] S. Koenig, C. Tovey, and W. Halliburton. Greedy mapping of terrain. In *Proceedings of the International Conference on Robotics and Automation*, pages 3594–3599, 2001.

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

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

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

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

[18] S. Russell and S. Zilberstein. Composing real-time systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 212–217, 1991.

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

[20] S. Schuierer. Efficient robot self-localization in simple polygons. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 129–146, 1996.

[21] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

[22] K. Sugihara. Some location problems for robot navigation using a single camera. *Computer Vision, Graphics, and Image Processing*, 42:112–129, 1988.

[23] S. Thrun. Probabilistic algorithms in robotics. *Artificial Intelligence Magazine*, 21(4):93–109, 2000.

[24] C. Tovey and S. Koenig. Gridworlds as testbeds for planning with incomplete information. In *Proceedings of the National Conference on Artificial Intelligence*, pages 819–824, 2000.