WORST-CASE ROBOT NAVIGATION IN DETERMINISTIC ENVIRONMENTS

A Thesis
Presented to
The Academic Faculty

by

Apurva Mudgal

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
May 2010

WORST-CASE ROBOT NAVIGATION IN DETERMINISTIC ENVIRONMENTS

Approved by:

Professor Craig A. Tovey, Advisor
College of Computing
Georgia Institute of Technology

Professor H Venkateswaran
College of Computing
Georgia Institute of Technology

Professor Eric Vigoda
College of Computing
Georgia Institute of Technology

Professor R. Gary Parker
School of Industrial and Systems Engineering
Georgia Institute of Technology

Professor Vijay Vazirani
College of Computing
Georgia Institute of Technology

Date Approved: 21 October 2009

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof. Craig Tovey, for his patience, wisdom, and the love of knowledge and independence that he has imbued in me. His subtle understanding of human nature has changed my life in innumerable ways. Everything he has taught me has left an indelible imprint on my mind and will be guiding me for the rest of my life. The decision to work with him happened by chance and I couldn't have been more lucky in getting a mentor.

I would also like to thank Prof. H. Venkateswaran for his help and guidance throughout my Ph. D. program. I have been a Teaching Assistant for him several times and this experience will be valuable for my future academic career. I would also like to thank him for agreeing to serve on my thesis committee.

Thanks are due also to Prof. Eric Vigoda for agreeing to serve on my thesis committee and for his several comments and suggestions. It was a joy to be a Teaching Assistant with him over the past several years.

Prof. Vijay Vazirani took a personal interest in my future career and thanks to him I have a much more clear idea of what I want to do after I graduate. I would also like to thank him for serving on the thesis committee.

Working with Prof. Joseph S. B. Mitchell on adapting our algorithm to the polygonal model was a very fruitful experience. Several theorems and constructions in Chapter 6 are due to him and helped make the thesis defendable. By interacting with him, I learnt a lot about computational and combinatorial geometry and these areas are sure to form a part of my long-term research interests.

Finally, I would like to thank Prof. Gary Parker for taking out his valuable time to serve as an external committee member. His encouragement after the proposal was very important and that led to me defending within a short period of time.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

We design and analyze algorithms for the following two robot navigation problems:

1. TARGET SEARCH. Given a robot located at a point s in the plane, how will a robot navigate to a goal t in the presence of unknown obstacles ?
2. LOCALIZATION. A robot is "lost" in an environment with a map of its surroundings. How will it find its true location by traveling the minimum distance ?

Since efficient algorithms for these two problems will make a robot completely autonomous, they have held the interest of both robotics and computer science communities.

Previous work has focussed mainly on designing competitive algorithms where the robot's performance is compared to that of an omniscient adversary. For example, a competitive algorithm for target search will compare the distance traveled by the robot with the shortest path from s to t.

We analyze these problems from the worst-case perspective, which, in our view, is a more appropriate measure. Our results are :

1. For target search, we analyze an algorithm called Dynamic A^*. The robot continuously moves to the goal on the shortest path which it recomputes on the discovery of obstacles. A variant of this algorithm has been employed in Mars Rover prototypes.

   We show that D^* takes $O(n \log n)$ time on planar graphs and also show an $O(n \log^2 n)$ bound on arbitrary graphs. Thus, our results show that D^* combines the optimistic possibility of reaching the goal very soon while competing with depth-first search within a logarithmic factor.

2. For the localization problem, worst-case analysis compares the performance of the robot with the optimal decision tree over the set of possible locations.

No approximation algorithm has been known. We give an $O(\log^3 n)$-approximation algorithm and also show a $\Omega(\log^{2-\epsilon} n)$ lower bound for the grid graphs commonly used in practice. The key idea is to plan travel on a "majority-rule map" which eliminates uncertainty and permits a link to the $\frac{1}{2}$-Group Steiner problem. We also extend the problem to polygonal maps by discretizing the domain using novel geometric techniques.

CHAPTER I

INTRODUCTION

This thesis involves the study of robot navigation problems in deterministic environments. These problems are special cases of planning problems where we have partial information about our models.

There are three central problems to robot navigation : (i) searching for a goal, (ii) localization i.e., finding one's location and (iii) mapping.

Of these three, we design and analyse algorithms for the first two problems. Our results involve finding the exact combinatorial and algorithmic complexity of theoretical problems inspired from real-life applications.

Traditionally, robot navigation problems have been studied from a competitive viewpoint i.e., the robot's performance in the presence of uncertainities is compared with that of a robot with perfect information about the "world".

Our *main thesis* is that it is more natural to look at robot navigation problems from the worst-case viewpoint. Thus, we compare the quality of our plans with the best plans any other robot can come up with, given the same partial information about its world.

Rather than making greedy or locally-optimal decisions, our robot navigation routines allow the robot to make long-term and globally-optimal decisions. Thus, a robot following our algorithms has the ability to make choices which may appear sub-optimal in near future but can lead to huge savings in the long run.

1.1 Models

We study localization within two well-studied two-dimensional models: models based on grid graphs and models based on polygons. The most intuitive of them is a grid graph or an occupancy grid.

A grid graph G is a finite rectangular region consisting of a union of unit square cells, as shown in Figure 1.1(a). Each cell can be either blocked or traversable. In the grid graph model, a robot is always in exactly one traversable cell. It starts in a traversable cell and can move in a single step to any neighboring traversable cell, to its north, south, east or west. Tactile sensors allow the robot to determine the states (blocked / traversable) of its four neighboring cells.

In the polygonal model [56, 12], the environment is a polygon P and the robot occupies exactly one point $p \in P$. The robot is equipped with a *range finder*, a device that emits a beam (laser or sonic) and determines the distance to the first point of contact with P's boundary in that direction. The robot sends out a series of beams spaced at regular angular intervals about its position, measuring the distance to the boundary at each of these angles. The points of contact are then joined together to obtain a *visibility polygon* $\mathcal{V}$ (see Figure 1.1(b)). In other words, the visibility polygon is the part of the polygon visible to a person by turning around by 360 degrees.

We use n to denote the combinatorial size of the map: for grid graphs n is the number of cells in G, and for a polygonal model n is the number of vertices in the map polygon P.

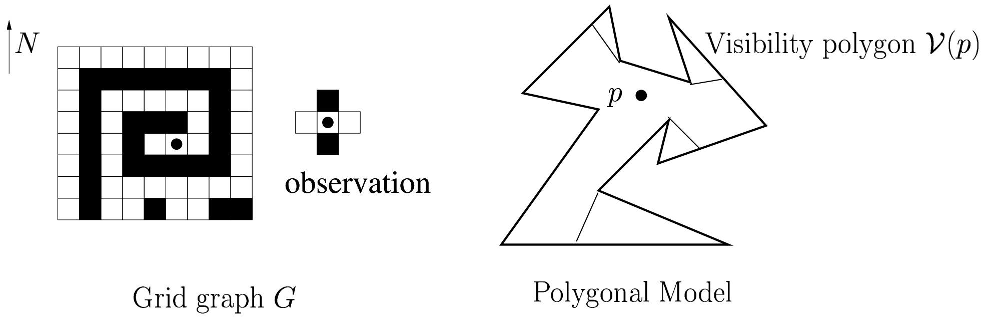


Figure 1: Two basic models : grid graphs and polygons

A third model consists of planar embedded graphs (see Figure 2). The environment $\mathcal{E}$ is modeled as a graph $G(V, E)$ embedded in the plane. The nodes correspond to locations of the robot in $\mathcal{E}$ and the edges (which may be curved) correspond to paths the robot can take to go from one node to another inside $\mathcal{E}$. At any time the robot is located at a node or is traversing an edge between two nodes. Some of the edges and vertices in the graph may be blocked due to debris, crevices, or other obstacles. An obstacle is not known until the robot's sensors detect it, for example, as the robot attempts to move to it.

Example. (Dual of a grid graph) Given a grid graph G, we can construct an associated embedded planar graph $D(G) = (V, E)$ as follows : (i) for each traversable cell $c \in G$, we introduce a

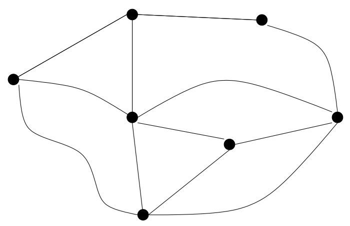


Figure 2: Planar embedded graphs

vertex $v_c \in V$. We make two vertices v, v' adjacent iff they correspond to adjacent traversable cells in the grid graph G (see Figure 3). $D(G)$ is called the *dual* to grid graph G.

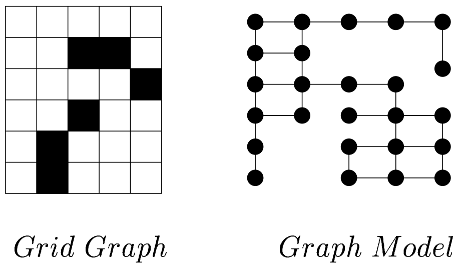


Figure 3: The dual of a grid graph

We note that the graph formulation can also be applied to other situations such as visibility graphs or a network of roads.

1.2 Assumptions regarding the robot

1. The robot is point-sized, omnidirectional and is capable of movement in the two dimensional plane. Clearly, these assumptions are exactly true for the polygonal model. In the grid graph model, one can assume that at any moment the robot is located at a point of the *unit square lattice* in the plane and is capable of making a move to any neighbouring lattice point.

2. The robot has *perfect sensing* : its computation of the visibility polygon is exact. The same holds true for the states of the four neighbouring cells in the grid graph model.

3. The robot is equipped with a *compass*. This allows it to know whether it points to the north, east, west or south. In the polygonal model, it amounts to exactly knowing the angle θ it makes with the x-axis.

4. The robot is equipped with a *perfect odometer* which allows it to exactly calculate the distance traveled by its wheels.

Clearly, a robot with an odometer and a compass can exactly calculate the path it has traversed in the map. This is straightforward to see in the grid graph model. For the polygonal model, our algorithms make the robot move on piecewise-linear paths P (as the shortest distances between two points inside a polygon are piecewise linear). By maintaining the angle of rotation at each new discontinuity of the piecewise-linear path P using the compass and the distance traveled along the straight line segments using the odometer, the robot can maintain an exact estimate of its position.

If one fixes an origin in the euclidean plane, the position of the robot at any moment is given by a set of coordinates (x, y). For the grid graph model, these numbers are integers whereas for the polygonal model they can be arbitrary real numbers. A complete description of the robot's state will by given by three coordinates (x, y, θ) where θ is the angle with respect to the x-axis.

Our assumptions amount to the following :

Assumption 1 (Perfect Locomotion Assumption) *If the robot's initial position p is given a coordinate $(0, 0)$, after any sequence of rotations about its axis and translations the robot correctly knows its current coordinate (x, y) with respect to p*

This brings us to the following question : how is the performance of a robot navigation algorithm to be measured ? The answer to this is unanimous : robots move much more slowly than computers process and hence we seek to minimize the total distance traveled by the robot. The only constraint that we impose on our algorithms is that they take polynomial computation time in the size of the problem instance. In other words, they should belong to complexity class P (for an introduction to complexity classes and the notion of polynomial-time algorithms, see [45]). We now state this as our assumption :

Assumption 2 (Minimizing Travel Distance) *As long as the algorithms on-board a robot's computer are polynomial-time in the problem size, their performance will be measured with respect to the total distance traveled by the robot.*

Note. We warn the reader that whereas Assumption 2 holds for real-world applications, the assumptions regarding perfect locomotion in Assumption 1 are not true due to mechanical drift and sensor noise. The robot's wheels can rotate on slippery floors and the odometer may give a false postive reading. These errors are particularly common when the robot makes sharp turns and moves on rough terrains. Similarly, due to changes in the environment (fogs, dim lighting etc.) the robot's sensors may give only partially correct information.

1.3 Problem Description

We now come to description of the problems considered in this thesis. We have looked at two distinct problems : target search and robot localization.

Both target search and localization are well-known in robotics literature as they form the core of a robot's locomotion routines. To make things clear, let us first describe these problems in the most general form.

TARGET SEARCH: Given the initial location s and the final destination t of a mobile robot, how will it find its way to the target in the presence of unknown obstacles ?

LOCALIZATION: A robot is "lost" in an environment. How will it find its location given a map of the environment ?

As one can see, a combination of these two subroutines will make a robot completely autonomous and therefore designing good algorithms for them is an important task. As such these problems in one form or another have carried the interest of both robotics and theoretical computer science communities. We leave the issues of practical robotics and instead focus on the algorithmic aspects.

We now summarize the models and informational assumptions underlying the problems studied. In the target search problem, the position of the robot is exactly known but the map is only partially known i.e., there are unknown obstacles. We will study the target search problem on two models : (i) grid graphs, and (ii) planar embedded graphs. In the robot localization problem, the position is known only upto a finite set H but the map is known exactly. We will describe approximation algorithms for the localization problem on (i) grid graphs, and (ii) the polygonal model.

In other words, there are two uncertainties involved. The first is *incomplete knowledge of the* *map*. This is the informational uncertainty for target search. The second is *incomplete knowledge of one's position*. This is the informational uncertainty for optimal robot localization.

1.4 Target Search

For target search, we analyze an algorithm called D^* (due to Anthony Stentz [47, 48, 49]) or greedy target search. The name derives from the fact that it is a dynamic version of A^*, the ubiquitous *best-first* search algorithm in Artificial Intelligence.

D^* continuously directs the robot on the optimal path to the goal given the current information about the map. It optimistically presumes that there are no undetected obstacles. The robot starts from the source vertex s and starts traveling on the presumed shortest path to the goal t. This shortest path is computed based on the robot's current knowledge of the map. If it encounters an obstacle on its path, the robot updates its map and then recomputes the shortest path from its current location to the goal. This process is repeated until either the robot reaches the goal vertex t or finds that it is unreachable given its present knowledge of obstacles.

Example. We consider a typical situation with an autonomous robot trying to reach a goal point in an unknown terrain (see Figure 4). The environment is a 6×5 grid graph and the robot has neighbourhood sensors. All cells are presumed to be traversable unless the robot makes an observation and finds that they are blocked.

The robot is initially located at $s = (0, 0)$ and the goal is at $t = (0, 4)$. Figure 4 illustrates an example run of D^*. The robot recomputes the shortest path four times before it reaches the goal. The obstacles discovered by the robot are the cells at $(0, 2), (1, 2), (2, 4)$, and $(0, 3)$ respectively. The total distance traveled by the robot is 8 steps.

In the first part of the thesis, we will try to answer the following question :

Question 1 *We are given a $m \times n$ grid graph G, a starting location s and a target location t. A subset B of cells is blocked, of which the robot has no knowledge. Suppose the robot tries to reach t using successive iterations of D^*. What is the maximum number of steps it can take in the worst-case ?*

Whereas the question is asked for grid graphs, we will instead study the maximum number of steps taken by D^* on planar embedded graphs $G(V, E)$.

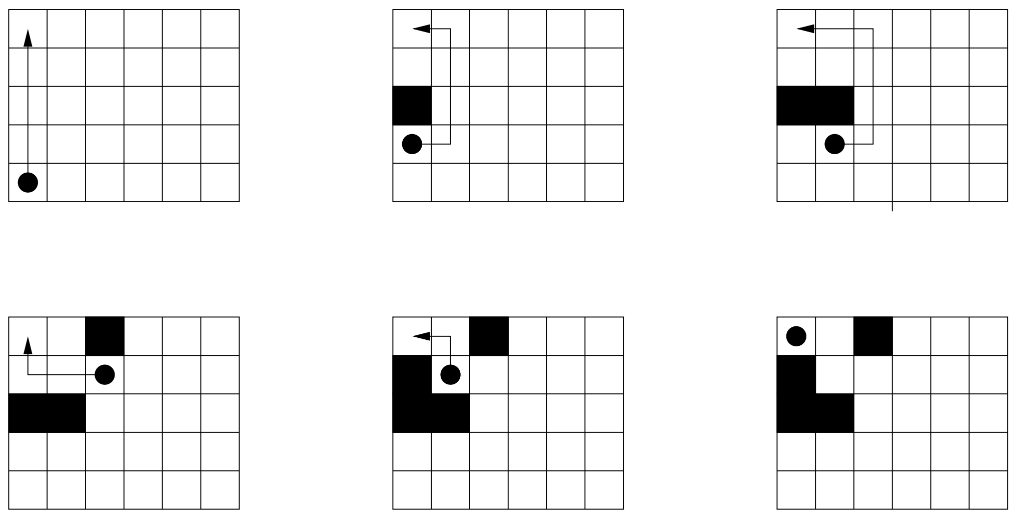


Figure 4: Example run of D^*

Clearly, a grid graph G can be modeled as a planar embedded graph by constructing its dual $D(G)$. Thus we can derive the results for grid graphs as a special case.

Planar embedded graphs $G(V, E)$ as a model for robot navigation were described in Section 1.1. At any time the robot occupies a vertex $v \in V$. The robot can travel from one vertex to another along the edges. We assume that the travel cost of the robot along any edge is 1.

From its current vertex, the robot can sense its environment. Any sensing operation returns a subset $S \subseteq V$ of vertices which are found to be blocked. For example, in Figure 4, the sensing operation at vertex $(0, 1)$ returned the singleton set $S = \{(0, 2)\}$.

However, the robot's sensors can range from tactile to sonars and lasers. For robots with long-range sensors, the set S will usually consist of more than one vertices. Further, in the presence of already discovered obstacles, the set of new blocked vertices may have gaps and may also contain later vertices on the current shortest path. This may complicate our analysis considerably.

We resolve this problem as follows. First we prove our theorems for a robot with **purely tactile** sensors : the only way the robot can discover that a vertex $v \in V$ is blocked is by trying to move to it from its current postions $u \in V$ along edge (u, v).

Then we show that our upper and lower bounds on D^* can be extended to robots with arbitrary sensors (this includes lasers, sonars, limited-range range-finders etc.) as long as they are at least as powerful as purely tactile sensors.

Our main result is an $O(n \log n)$ upper bound on the maximum travel distance for D^* on planar embedded graphs. The approach can be extended to give an $O(n \log^2 n)$ upper bound for D^* on general graphs :

Theorem 1 *For robot sensors satisfying the purely tactile property, D^* traverses $O(n \log^2 n)$ edges on connected (general) graphs $G = (V, E)$. It traverses $O(n \log n)$ edges on connected planar graphs $G = (V, E)$.*

Note that in principle the robot can take $\Omega(n^2)$ steps, as it may oscillate and replan n times, each time taking close to linear in n steps. Before our work, the best bound known was $O(n^{3/2})$ [28].

We also show that the upper bound is not far from optimal as we also prove the following theorem :

Theorem 2 *The worst-case travel distance of D^* on vertex-blocked grids as well as planar graphs $G = (V, E)$ is $\Omega(\frac{n \log n}{\log \log n})$ steps.*

A worst-case performance of $O(n \log n)$ for D^* is very surprising, when one observes that unlike breadth-first or depth-first search, the robot keeps no extra information (such as parent-child pointers, marking times, last node visited, queues etc.) on the graph except whether the nodes were last found to be blocked or traversable.

The gap between the best known lower and upper bounds is now quite small, namely $O(\log \log n)$ for planar graphs, and $O(\log n \log \log n)$ on arbitrary graphs. We leave it as an open question to close the gap between the upper and lower bounds for planar graphs.

However, D^* is not an interesting algorithm from the point of view of competitive analysis. We will show in Section 1.8 that D^* can have an arbitrarily bad performance competitively whereas there exist $O(\sqrt{n})$ competitive algorithms for target search in environments with rectangular obstacles.

1.5 Localization

Consider the following problem: a mobile robot is placed at an unknown position in an environment for which it has a map $\mathcal{E}$. The robot constructs a map $\mathcal{E}'$ of its local environment by going to different places and sensing the environment from there. It rules out positions whose local environment does not agree with map $\mathcal{E}'$, until it infers the unique position where it was originally located. The objective is to complete this task by traveling the minimum possible distance. This is known as the *kidnapped robot* or *localization* problem [12, 55].

In general, robots must localize when they are switched on because they may have been moved while switched off. Also, the control systems guiding a robot gradually accumulate error due to mechanical drift and sensor noise [14]. Thus, it is necessary to localize from time to time to verify the actual position of the robot in the map, and then apply corrections. In this context, localization eliminates the need for complex and expensive position-guidance systems, such as radio beacons [40, 12], to be installed in buildings or streets with tall buildings, where three satellites are not in view and so GPS is not effective. For situations in which such systems cannot be built, such as a Mars rover (see [32]), localization is the only possibility.

Following Dudek *et al.* [15], we view the localization problem as having two phases: *hypothesis generation* and *hypothesis elimination*. The first phase is to determine the set H of hypothetical locations, or *hypotheses*, that are consistent with the sensing data obtained by the robot from its initial location (see Figure 5). The second phase is to determine which location $h \in H$ is the true location of the robot. (The second phase is unnecessary if $|H| = 1$.)

We first briefly describe hypothesis generation. In the following, we will use k to denote the number of hypothetical locations. For the grid graph model, H is simply the set of all traversable cells, and therefore $k = |H|$ is less than or equal to n, the size of the grid graph.

For the polygonal model, a simple algorithm to construct the set H of hypothetical locations is as follows. First, pick a reflex vertex v of the visibility polygon $\mathcal{V}$ (if the visibility polygon has no reflex vertices, the map polygon will be convex and hence the robot will immediately localize). Then try all placements of $\mathcal{V}$ such that v coincides with a vertex of the map polygon P. If the visibility polygon for the initial position h of the robot given by a placement coincides with the initial observation $\mathcal{V}$, output h into the set of hypothetical locations. Since the number of placements is at most the number of vertices in P, the set H of hypothetical locations obtained by this algorithm will have size at most n. Figure 5 gives an example where one obtains a set of $k = 4$ hypotheses from the initial visibility polygon $\mathcal{V}$.

Guibas, Motwani and Raghavan [20] describe how to implement the above algorithm in $O(mn)$ time where m and n are the number of vertices in $\mathcal{V}$ and P respectively. They also describe a way to preprocess the polygon so that any hypothesis generation query can be answered quickly. Their algorithm preprocesses P in $O(n^5)$ time and space, and generates hypotheses in $O(m + \log n + k)$ time, where m is the number of vertices in the observed visibility polygon $\mathcal{V}$, and $k = |H|$ is the number of hypotheses generated.

The above discussion shows that the hypothesis generation phase for both grid graphs and polygons produces a set of at most $k \leq n$ hypotheses, where n is the size of the map (for grid graphs, n is the total number of cells and for polygons, n is the total number of vertices). Therefore, in this thesis, we will focus on the hypothesis elimination problem.

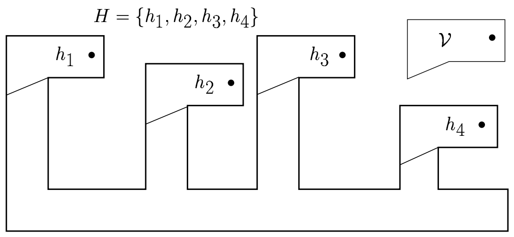


Figure 5: Hypothesis generation. Based on the observed visibility polygon $\mathcal{V}$, we generate the set $H = \{h_1, h_2, h_3, h_4\}$ of hypotheses as the possible locations of the robot.

By a *strategy* S we mean the hypothesis elimination routine employed in the robot's computer. We measure the effectiveness of a strategy based on its *worst-case* performance. For strategy S, let $W(h, S)$ be the distance traveled to localize if the robot is placed at hypothesis $h \in H$. Then the cost $W(S)$ of strategy S is defined to be the maximum distance, $W(S) = max_{h \in H} W(h, S)$, traveled for any starting position h. An optimal strategy S^* has cost $W(S^*) = \min_{S \in \mathcal{S}} W(S)$, where $\mathcal{S}$ denotes the set of all possible localization strategies. OPT(G,H) denotes the cost of an optimal strategy, where G is the map and H is the set of hypotheses. We say that a strategy is *α-approximate* ($\alpha \geq 1$) if its cost is at most $\alpha \cdot$ OPT(G,H).

Fig. 6 gives an example of a localization strategy. Fig 6(a) shows the map in which the robot is located. We assume that the robot has already made an observation from its current location and has generated a set of four hypothetical positions $H = \{h_1, h_2, h_3, h_4\}$.

Fig. 6(b) describes an "execution" of the hypothesis elimination routine which can also be

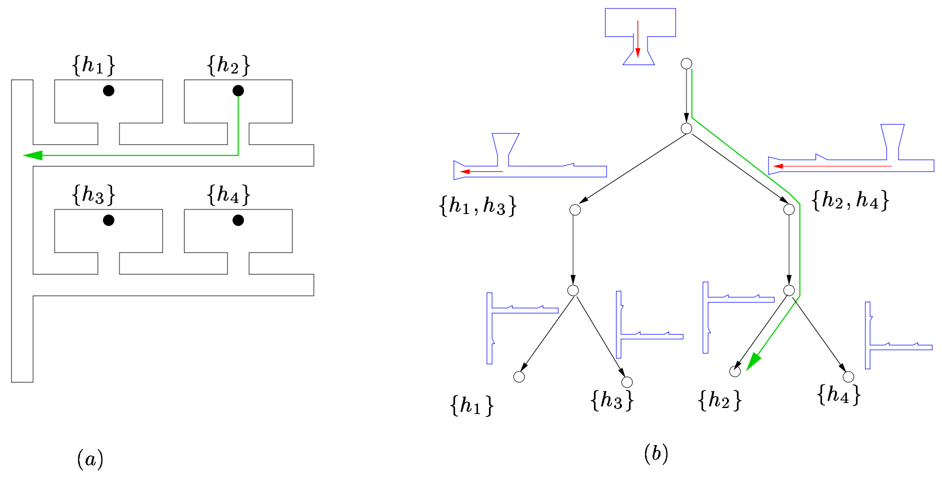


Figure 6: Hypothesis elimination

modeled as a decision tree [15]. The red arrows describe the next move made by the robot. First, the robot moves down to the left-right corridor and makes an observation. According to its position inside it, the robot is able to distinguish between $H_1 = \{h_1, h_3\}$ and $H_2 = \{h_2, h_4\}$. Once it has decided between the two sets H_1 and H_2, it needs to know whether it is in the top corridor or the bottom corridor. It does so by going west till it can make a measurement in the north-south corridor. The length of the path traveled in the two cases is different, with the longer distance being traveled for set H_2. Clearly, we have that $W(h_1, S) = W(h_3, S) = d + d_1$ and $W(h_2, S) = W(h_4, S) = d + d_2$ with $d_2 > d_1$. The cost of the strategy is given by the maximum of these two quantities which is $W(S) = d + d_2$. The green arrow describes the path taken by a robot if it were "actually" located at h_2.

Our main contribution is a polynomial-time strategy, **R**epeated **H**alf **L**ocalization, which localizes the robot with travel distance within a factor $O(\log^3 n)$ of that of an optimal strategy; more precisely, the approximation factor is $O(\log^2 n \log k)$, where $k = |H| \leq n$ is the number of hypotheses. The key algorithmic idea is to plan travel in a "majority-rule" map, which eliminates uncertainty and permits a link to the $\frac{1}{2}$-Group Steiner (not Group Steiner) problem.

An instance of the Group Steiner problem consists of a weighted graph $G(V, E)$ with k groups of vertices $g_1, g_2, \ldots, g_k \subset V$. The objective is to find a minimum weight tree which contains at least one vertex from each group. In the $\frac{1}{2}$-Group Steiner problem, the objective is to find a minimum weight tree which contains at least one vertex from *half* the groups. Both these problems are

NP-complete and we refer the reader to Section 2.2 for a discussion of approximation algorithms available for them.

The approximation factor of our localization strategy, **R**epeated **H**alf **L**ocalization, is not far from optimal: we prove a $c\log^{2-\epsilon} n$ lower bound, assuming $NP \not\subseteq ZTIME(n^{polylog(n)})$, for the grid graphs commonly used in practice. We also extend the algorithm to polygonal maps by discretizing the problem using novel geometric techniques.

The basic framework of the strategy is to break localization into a sequence of *half-localize* steps:

HALF-LOCALIZE (G,H): Devise a strategy by which the robot can correctly eliminate at least half of the hypotheses in H. The robot should travel (worst-case) distance as small as possible to achieve this. HALF-OPT(G,H) denotes the cost of the optimal strategy.

Intuitively it might appear that an $O(\log^2 n)$ algorithm for half-localization should be a by-product of our $O(\log^3 n)$ localization strategy and not vice-versa. As an example of this, consider the $\frac{1}{2}$-Set Cover problem, in which the objective is to cover half the elements at minimum cost. There is a constant factor approximation for this; and it is obtained by stopping the $O(\log n)$ greedy algorithm for Set Cover as soon as we cover half the elements. (Another example is the algorithm for $\frac{1}{2}$-Group Steiner [16], which is obtained by stopping the rounding scheme of [19] as soon as the tree covers half the groups.)

However, half-localize seems to play a more fundamental role in our context. We briefly discuss only the simpler grid graph case here. We construct a "majority-rule map", in which each cell is blocked or unblocked depending on what the majority of the current hypotheses in H assert. This majority-rule map permits *three inter-related simplifications*. If the robot tries to follow a route within the majority-rule map, but makes a minority observation (one inconsistent with at least half of the hypotheses), then the robot has half-localized. This permits a plan to be a path rather than a decision tree. Distances in the real environment are uncertain, but distances on the majority-rule map are fixed. This permits us to model half-localization as a Steiner type problem on a graph, although we are not able to model localization as such. Finally, there is an essential equivalence between optimally half-localizing and halving paths on the majority-rule map.

We refer the reader to the introduction to Chapter 4 for a detailed discussion of the ideas underlying our algorithm for the grid graph model. Similarly, the introduction to Chapter 5 describes the basic ideas underlying our lower bound.

The following figure illustrates the structure of our results :

$$\begin{tikzcd}
Robot\ Localization \arrow[r, shift right, "\times" description] \arrow[d] & Group\ Steiner\ problem \arrow[l, shift right, "\Omega(\log^{2-\epsilon} n)\ lower\ bound"'] \arrow[d] \\
Half-localization \arrow[r, "O(\log^2 n \log k)\ upper\ bound", "Majority-rule\ map"'] & \frac{1}{2} - Group\ Steiner\ problem
\end{tikzcd}$$

Figure 7: Logical Structure of Our Results

Note that relationships are one-way. The only way we know of deriving an algorithm for the robot localization problem is by connecting the half-localization problem using majority-rule maps to the $\frac{1}{2}$-Group Steiner problem. On the other hand, the hardness of approximation is obtained by a direct reduction (by direct we mean of the NP-hardness type) from the Group Steiner problem.

1.6 Polygonal Model

The third part of the thesis is an extension of the algorithm for grid graph localization to the polygonal model. We note that the hardness of approximation extends in a straightforward fashion to polygons (by adding “twists” to block the laser sensors) (see Chapter 5).

We consider the extension of our algorithm to the polygonal model as the most technical part of the thesis.

The main difficulty in the polygonal model is that the number of robot positions (x, y) is not finite. Instead the robot can occupy any of the inifinite positions in the interior of the map polygon P i.e., the number of positions is of the same cardinality as the real numbers.

We first describe the standard approach to this problem. Cell decompositions of a set of points and lines have been known for a long time such that all points within a cell C satisfy the same properties (Delaunay triangulation, Voronoi diagrams etc.).

Here we need decompositions of the map polygon into cells so that the robot makes essentially the same observation within the same cell (for example, sees the same set of edges or vertices in P). Once we have such a decomposition, by cleverly choosing a set of "representative points" within each cell, one can discretize the map polygon into a finite set of points S each with its associated observation $o_p, p \in S$. This is the line of approach used in the papers by Guibas *et. al.* [20] and Dudek,Romanik and Whitesides [15].

A preliminary version of this work tried to adapt these ideas to our case. We first constructed visibility decomposition for pairs of hypotheses $\{h_i, h_j\}$ such that each cell C was either able to distinguish between them or was unable to distinguish betweem them. By overlaying these k^2 pairs over each other, we were able to construct a partition $D(P, H)$ of the plane such that within each cell the majority observation and sets remain the same. The complexity of $D(P, H)$ was $O(n^8k^4)$ and after choosing representative points for the $\frac{1}{2}$-Group Steiner problem the exponent nearly doubled !

Even if we assume that in real situations that $k << n$, our algorithms were not practical because of the large order $O(n^{16})$ in the map size n. Therefore, reducing the number of representative points was important for making our algorithms practical.

Note that we are now concerned with reducing the computation time of the robot's computer rather than its travel distance.

The sizes of Group Steiner instances constructed for the grid graph model are of order $O(kn)$. The bound is linear in n and so computation is not an issue (though the map size n here is the number of grid cells in G). We would like to have a similar performance for the polygonal model. In fact, the results of Chapter 6 take us almost to this goal.

In Chapter 6, we will show that for the polygonal model one can get a set of representative points of size $O(nk^{10/3}\alpha(n)\log^{5/3} k)$ where $\alpha(n)$ is the inverse of the Ackermann function. Note that this bound is nearly linear in n (the inverse of the Ackermann function grows so extremely slowly that for all practical purposes it can be taken to be less than 4) and one may expect that, in practice, $k << n$.

In Section 6.7, we further reduce the set of representative points to $O(k^2 \log k)$. Note that this bound is independent of n and hence the size of Group Steiner instance constructed is just quadratic in the number of hypotheses k. We adapt the same ideas to get a set of $O(hk^2 \log k)$ representative points for the half-localization problem in polygons with convex holes. For polygons with non-convex holes, we get a set of $O(rk^2 \log k)$ reference points, where r is the number of reflex vertices on the boundary of holes.

The argument which gives $O(nk^{10/3}\alpha(n)\log^{5/3} k)$ points for polygons, gives $O(k^3n^4)$ reference points for polygons with holes (see Section 7.1.3). The new argument leads to a considerable improvement by lowering this number to $O(rk^2 \log k)$.

Thus we can say that our algorithms are now practical for the polygonal model with arbitrary holes. Note that by practical we mean the computation time needed to construct the Group Steiner problems for half-localization. The current algorithms for Group Steiner have large running times, however that doesn't concern us here. For a discussion of their running times, see section 2.2 of this thesis.

To get a construction with a small number of points we needed to introduce several novel geometric and combinatorial ideas suited to the structure of our algorithm. Even a general overview of them is difficult to give and instead we refer the reader to Chapter 6 of the dissertation.

We would like to **thank Prof. J. S. B. Mitchell** for helping us adapt our algorithm to the polygonal model. Several of the results, improvements and theorems in Chapter 6 are due to him and they use state-of-the-art machinery in computational geometry.

1.7 Extensions to other models

One unifying theme in our work on the robot localization problem is that the algorithms and lower bounds obtained for the simplest model of grid graphs generalize to wide variety of models with appropriate modifications. In Chapter 6, we will look at some of these generalizations. The following Figure describes the set of models for which the results for grid graphs can be generalized (An arrow between models 1 and 2 denotes that the second one is a generalization of the first one) :

One feature of our strategy is that after every half-localize phase the robot returns to the origin. In Section 7.2 we show that a variant of our strategy which doesn't return to the origin after each half-localize step performs very poorly.

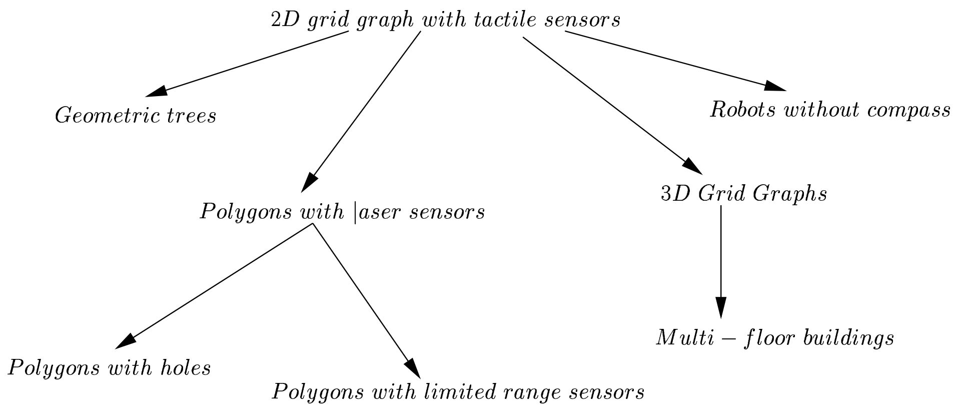

2D grid graph with tactile sensors
Geometric trees
Robots without compass
Polygons with laser sensors
3D Grid Graphs
Multi – floor buildings
Polygons with holes
Polygons with limited range sensors

Figure 8: Models for the Localization Problem

1.8 Competitive Analysis vs. Worst-case analysis

As noted above, a large part of previous work on robot navigation problems is with reference to competitive analysis or online algorithms. Competitive analysis compares the performance of a robot navigation algorithm with respect to an omniscient adversary. For the robot navigation problems considered in this thesis, an omniscient adversary will have *complete knowledge* of both the *map* and the *robot's position* inside it. We now convey the flavor of results achieved in this direction and compare them with our results. The reader can refer to Chapter 2 for a detailed description.

Now let us first look at results for the target search problem. Since the target search problem becomes trivial if the map is known, it makes sense only with respect to scenarios where the robot has no or little knowledge of the obstacles. As discussed above, competitive analysis will compare the performance of the robot with respect to an omniscient adversary. For target search, full knowledge of the map and the robot's position inside it will allow the omniscient adversary to compute the shortest path to the goal. Hence, the performance ratio of competitive algorithms for target search is measured with respect to the length of the shortest path.

Consider an environment with rectangular obstacles where the robot has to go from s to t but has *no knowledge* of the obstacles. It is known that the best competitive ratio achievable for this problem is $\Theta(\sqrt{n})$, where n is *the length of the shortest path.* Blum, Raghavan and Schieber [6] showed that there is an algorithm which always makes the robot travel distance within $O(\sqrt{n})$ times the shortest path. On the other hand, Papadimitriou and Yannakakis [39] showed that an adversary can choose the location of obstacles to force the robot to travel at least $\Omega(\sqrt{n})$ times the shortest path.

For target search, we have analyzed a shortest-path heuristic called D^* : given the current knowledge of the robot about its environment, the robot always moves on the shortest path from the current position to the goal t. We have shown that its worst-case performance is $O(n \log n)$ for planar embedded and vertex-blocked grid graphs.

However, D^* can perform arbitrarily badly when considered competitively. Fig. 9(a) gives an example of an environment with rectangular obstacles.

A tactile robot has to go from s to t in a room where there is a long and thin rectangular obstacle. The robot is much closer to the southern tip of the obstacle. The first time the robot hits the obstacle, it needs to move on the new shortest path. If the robot chooses to move up, it will keep going upwards till it comes out of the top of the obstacle.

Thus, the shortest path has length $\sim d_1$, whereas the path taken by D^* has length $\sim 2 \cdot d_2 + d_1$. The competitive ratio is then given by $\frac{2 \cdot d_2 + d_1}{d_1}$. Choosing $d_2 >> d_1$, this ratio can be made arbitrarily large.

We note that the same example can be adapted to work for robots with range finders (see Fig. 9(b)). The obstacle now extends across the height of the room with evenly spaced east-west corridors along its boundary. The robot can go to the other side of the room by passing through one of these east west corridors. However, only the top and the bottom corridors are unblocked. The width of the corridors is so small that even a robot with range finders has to occupy the cell in front of them to find out whether they are blocked or traversable. One can then show similarly that D^* will make the robot move upward and then check each corridor for blockage till it comes out of the top corridor. However, the optimal plan consists of taking the bottom corridor. One gets similarly that if $\frac{d_2}{d_1} \to \infty$, then the robot performs arbitrarily badly.

Our motivation for analyzing the performance of D^* lies in the fact that it gives empirically good performance on real terrains and is widely used in practice. On the other hand, algorithms with good competitive ratios are rarely used in practical robotics. However, theoretically D^* gives an arbitrarily bad competitive ratio even in very simple environments. This difficulty is taken care of in algorithms with good competitive ratios by doing a “spiral search” [2] or “sweeps” with increasing window sizes [6].

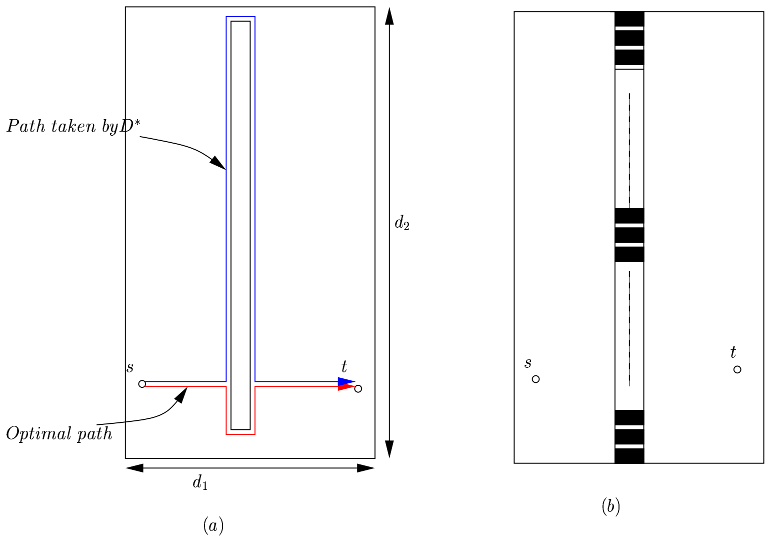


Figure 9: Competitive performance of D^*

The notion of competitive analysis leads to a subtle definition when applied to robot localization.

The main issue here is that the map is known but the location is unknown. Thus both the robot and the omniscient adversary have a map of the environment. If the adversary even knows its initial location, what will we compare the robot's performance with ? We make a comparison with the *optimal verification tour* i.e., the distance the adversary needs to travel to *confirm* for himself that his belief about his location is correct. We stipulate that the optimal verification tour always returns back to the origin.

Dudek, Romanik and Whitesides [15] show that the competitive complexity of this problem for polygons is exactly $k-1$ where k is the number of possible candidate (or hypothetical) locations. Their algorithm greedily takes the robot to the next informative location and then retraces its path back to the origin. As each new information rules out at least one location, the robot finds its location in k phases.

The upper bound is as follows. Suppose the robot is located at $h \in H$. Suppose the strategy makes m phases and let H_i denote the set of hypotheses at the start of the ith phase. Since $h \in H_i$, the distance to the nearest informative point will be at most the minimum verification tour V for h. For consider the first point p on V where the robot distinguishes h from at least one other hypothesis in H_i. This point p is then an informative point for phase i and hence lies at a distance greater than the nearest informative point. Since there are at most $k-1$ steps, the strategy localizes by traveling distance within $k-1$ of the optimal verification tour.

The lower bound consists of k buildings where each building has k spoke-like corridors around a central common area (see Fig. 10). The corridors are distinguished as follows : in building i, the ith corridor has a “hook” on its end. To localize, the robot needs to search for a corridor with a hook.

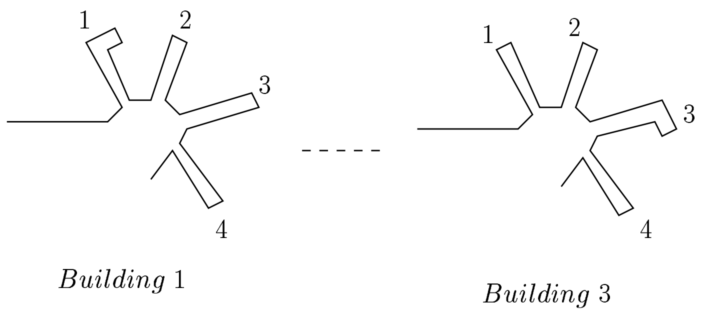


Figure 10: Lower bound for greedy localization

If the robot knows its building number i, it can go to the end of spoke i and verify its location by checking for the hook and return back to the origin. Thus the optimal verification tour for any hypotheses is equal to twice the spoke length.

On the other hand, any strategy which visits the spokes in a given order can be defeated by putting the robot in the building corresponding to the last spoke visited. Thus, in the worst-case (competitively) the robot will check $k-1$ spokes whereas an omniscient verifier will just check one spoke. Note that if the robot does not find a hook in $k-1$ spokes, it automatically localizes to the remaining spoke.

Since we stipulate that the robot always returns to the origin, the robot will travel distance equal to $k-1$ times twice the length of a hook. Thus, an omniscient adversary can make the robot travel $k-1$ times the minimum verification tour.

However, we believe that it is more natural to look at the robot localization problem from the worst-case perspective. Here we list some points which make a strong case of using worst-case robot localization over the competitive approach :

1. In online navigation problems, the map is not known and hence the informational assumption of competitive analysis holds for the robot. But in the localization problem the map is given *a*

priori to the robot. Hence the information available to the robot is precisely what is needed for standard worst-case analysis. A competitive analysis assumes too little information available to the robot, and too much to the omniscient verifier, than is realistic.

2. From a practical standpoint, it better matches the roboticist's concerns with guaranteed rapid localization, rather than with comparisons against a nonexistent omniscient verifier.

3. From a theoretical standpoint, it admits an $O(\log^3 n)$ approximation algorithm, whereas it is NP-hard even to achieve a strategy with competitive ratio $o(\sqrt{n})$ on polygons [15]. Thus the planner now has a much more richer structure, which he/she can exploit and adjust according to the application.

4. It turns out that the simplest greedy strategy to localize is also the optimal from a competitive viewpoint and it consists of always going to the next informative point on the map. However this strategy uses *absolutely no structure* of the map (even when the robot has it) and for all purposes takes the map as a *black box*. On the other hand, our localization routines make essential use of the map by using them to compute the majority-rule map.

1.9 Outline

Figure 11 describes the organization of the thesis and the dependencies among the various chapters.

In chapter 2, we discuss the previous work on both target search and robot localization problems. Chapter 3 contains the lower and upper bounds for the performance of D^* in planar and general graphs. Chapter 4 describes the $O(\log^3 n)$-approximate localization strategy for grid graphs. In chapter 5, we show an $O(\log^{2-\epsilon} n)$ lower bound on the performance of any polynomial-time computable localization strategy. Chapter 6 forms the most technical part of the thesis where we extend the localization results for grid graphs to the the polygonal model. Chapter 7 describes various other extensions of our localization algorithms such as those for three-dimensional grid graphs, limited-range sensors, geometric trees etc. The general rule is that our algorithm can be extended to any model provided we define the majority-rule map and give a finite set of coordinates for the $\frac{1}{2}$-Group Steiner problem inside it. Finally chapter 8 summarizes our results and lists some open problems for future work.

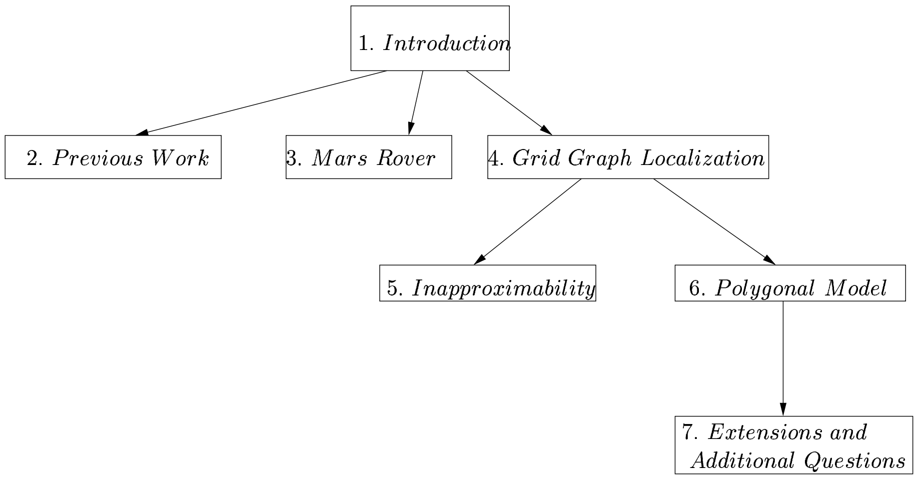


Figure 11: Outline

CHAPTER II

PREVIOUS WORK

2.1 Competitive Analysis

2.1.1 Target Search

The motivation for competitive algorithms comes from theoretical work of a similar flavor on robot navigation in *unknown* environments. The objective of the robot is to navigate from a point s to a target t while avoiding obstacles/walls in the scene, which are not known to the robot *a priori*, but which the robot learns by encountering them. The goal is to minimize the competitive ratio of the distance traveled by the robot to the length of the shortest obstacle-free path from s to t. Papadimitriou and Yannakakis [39] gave the first such results, achieving a competitive ratio of 1.5 (which they show is best possible) in the case that obstacles are unit squares. They, along with Eades, Lin and Wormald [38] also give a lower bound of $\Omega(\sqrt{n})$ on the competitive ratio in the case that t is an infinite wall and the obstacles are axis-aligned rectangles. Baeza-Yates, Culberson and Rawlins [2] introduce the technique of *spiral search*, with which they obtain a $(9 + o(1))$-competitive algorithm for finding a point on a line, and a 13.81-competitive algorithm to search for a line at distance n from the origin. A restricted spiral search in a geometric tree forms the first part of Kleinberg's localization algorithm. Blum, Raghavan and Schieber [6] use a variant of the spiral search technique to give a strategy that matches the $\Omega(\sqrt{n})$ lower bound for navigating between two points among axis-aligned rectangular obstacles. The navigation problem has also been studied in the polygonal model, for which Klein [24] gives a lower bound of $\sqrt{2}$ on the competitive ratio and gives a 5.72-competitive algorithm for a subclass known as *street polygons*. Later, Kleinberg [26] improved the ratio to $2\sqrt{2}$, and Datta and Icking [13] gave a 9.06-competitive algorithm for the broader class of *generalized streets*.

2.1.2 Robot Localization Problem

Despite the considerable attention it has received in the robotics literature (e.g. [12, 55, 43, 53, 56]), localization has been the subject of relatively little theoretical work. Guibas, Motwani, and

Raghavan [20] show how to preprocess the polygon P so that the set of hypotheses H consistent with a single observation $\mathcal{V}$ can be returned quickly. Their algorithm preprocesses P in $O(n^5)$ time and space, and generates hypotheses in $O(m + \log n + k)$ time, where m is the number of vertices in the observed visibility polygon $\mathcal{V}$, and $k = |H|$ is the number of hypotheses generated. (Note that $k \leq n$, and, in fact, k is at most the number of reflex vertices of P.)

Kleinberg [25] was the first to give interactive strategies for the hypothesis elimination problem. He measures the performance of his strategies using the *competitive ratio* criterion, in contrast with our worst-case criterion. The competitive ratio compares the distance traveled by a robot following a strategy to that traveled by an *omniscient verifier*, i.e., a robot that has *a priori* knowledge of its position $h \in H$ and probes the environment just to verify this information. The distance traveled by an omniscient verifier at hypothesis h is exactly $min_{S \in \mathcal{S}} W(h, S)$, and an α-competitive strategy enables a robot initially located at hypothesis h to travel distance at most $\alpha \cdot min_{S \in \mathcal{S}} W(h, S)$ prior to completing localization.

In Kleinberg's model the environment is a *geometric tree*, $G(V, E)$, where V is a set of points in $\mathbb{R}^d$ and E is a set of line segments whose endpoints all lie in V. The edges do not intersect except at V and do not form cycles. The robot occupies a point on one of the edges, and is capable of moving along an edge in either direction. Kleinberg further assumes that the only information available to the robot is the orientation of all edges incident at its current position $p \in E$. He gives an $O(n^{2/3})$-competitive algorithm on geometric trees having bounded degree, and he gives an $\Omega(\sqrt{n})$ lower bound. He also gives an $O(n\sqrt{\frac{\log n}{\log \log n}})$-competitive algorithm for a geometric model consisting of a packing of rectangles (obstacles) in the plane, with no two rectangles "stuck together" (i.e., two rectangles can nearly touch, but there remains a traversable gap between them) and each rectangle having at least unit width. In Section 7.1.4, we give an $O(\log^3 n)$-approximate strategy not just for geometric trees, but for geometric graphs in any Euclidean space $\mathbb{R}^d$.

Dudek, Romanik and Whitesides [15] consider the problem of designing competitive strategies for the polygonal model; however, they assume that the robot can only compute the *visibility skeleton* $\mathcal{V}^*(p)$, which is an approximation of visibility polygon $\mathcal{V}(p)$. The visibility skeleton $\mathcal{V}^*(p)$ (see [20]) is a contraction of $\mathcal{V}(p)$, consisting of only those vertices in $\mathcal{V}(p)$ that can be certified to be vertices of P. For this model, they give a greedy $2(k-1)$-competitive strategy **MDL** (stands for

Minimum **D**istance **L**ocalization) for hypothesis elimination, where $k = |H|$ is the number of hypotheses. They also show that there are polygons P and sets of hypotheses H for which the best strategy is $2(k-1)$-competitive. We believe that this line of work stands closest to ours in both geometric and algorithmic structure. We refer the reader to Section 6.8.1 for a discussion of the recent work on this strategy as well as a comparison with our results.

Dudek, Romanik and Whitesides were also the first to study the localization problem from the worst-case perspective, which they describe as the height of a *localizing decision tree*. They prove that computing an optimal localizing decision tree (i.e., an optimal worst-case strategy) is NP-hard by a reduction from the Abstract Decision Tree problem [23]. Tovey and Koenig [54] show that it is NP-hard even to find a $c \cdot \log n$-approximate strategy, both for grid graphs and for polygons, using a reduction from the Set Cover problem [30]. Schuierer [44] proposes a technique that uses geometric overlay trees to reduce the running time of Dudek *et al.*'s greedy strategy. His technique, along with a careful choice of data structures, allows the robot to localize in computation time $O(kn \log n)$ and space $O(kn)$.

Brown and Donald [7] describe algorithms for localization that allow for uncertainty in the measurements of range sensors. Fox, Burgard and Thrun [18] use Markov localization to deduce the position of the robot from sensor data. In their work, global localization is achieved as a side effect of robot movement, and the length of the localizing trajectory relative to the optimum is not considered. In Markov localization and related approaches, localization and action are viewed in a compound setting; the effects of various actions are interpreted probabilistically and the robot is able to predict the belief states ensuing from various actions. Long-range path planning using these approaches remains problematic because of the large state space involved.

2.2 Group Steiner Problem

The Group Steiner problem is the following:

(Rooted) Group Steiner Problem. Given a weighted graph $G = (V, E)$ with k groups of vertices $g_1, g_2, \ldots, g_k \subset V$, find a minimum weight tree that contains at least one vertex from each group. There is a distinguished vertex r (the *root* vertex) that must be included in the tree.

The Group Steiner problem generalizes both minimum Steiner tree and set cover problems. For purposes of our algorithm, we need a variant called the $\frac{1}{2}$-Group Steiner problem [16], in which the goal is to find a minimum-weight tree that contains vertices from at least half of the groups.

An $O(\log^2 n)$ algorithm for Group Steiner on trees is given by Garg, Konjevod and Ravi [19]. They first solve a linear programming relaxation to get a fractional solution and then use an innovative randomized rounding scheme. A modification of the algorithm, by Even, Kostartz and Slany [16], yields an $O(\log n)$-approximation for the $\frac{1}{2}$-Group Steiner problem on trees. For general graphs, one can first probabilistically approximate the graph by a tree, using a result of Fakcharoenphol, Rao, and Talwar [17] (which is a recent improvement to Bartal [4]), losing an $O(\log n)$ factor in the process. Then the algorithm of Garg *et al.* [19] is applied to the resulting tree, giving an $O(\log^3 n)$-approximation for Group Steiner and an $O(\log^2 n)$-approximation for the $\frac{1}{2}$-Group Steiner problem:

Theorem 3 *[19, 16, 17] There exists an $O(\log^2 n)$-approximation algorithm $\mathcal{A}$ for the rooted $\frac{1}{2}$-Group Steiner problem that runs in randomized polynomial time.*

The running time of this algorithm is high, and hence the computation time of the robot will be large. As the approximation algorithm is used only as a black box, we will denote the running time by the (polynomial) $\mathcal{P}(n')$, and instead concentrate on reducing the size n' of the instance. However, if we are willing to trade off between running time and approximation factor, there are much faster algorithms available. Bateman *et al.* [5] give a $\sqrt{k}\ln k$-approximation algorithm that runs in $O(nk^2 \log k)$ time. Their algorithm is based on the fact that there exists a Group Steiner tree of depth 2 rooted at r with cost within $\sqrt{k}$ of the optimal. By adapting their algorithm to the $\frac{1}{2}$-Group Steiner problem, we get an $O(\sqrt{n}\log^2 n)$-approximation strategy for localization on grid graphs with computation time $O(n^3 \log^2 n)$ (the best previous factor was $\Omega(n)$). A more smooth trade-off can be obtained by using the algorithm of Charikar *et al.* [10] for the Directed Steiner tree problem (which includes the $\frac{1}{2}$-Group Steiner problem as a special case), yielding an $i(i-1)k^{(1/i)}$-approximation with running time $O(n^i k^{2i})$. For any $\epsilon > 0$, the robot localizes by traveling distance within factor $O(\frac{n^\epsilon}{\epsilon^2} \cdot \log n)$ of the optimal and spending computation time $O(n^{\frac{3}{\epsilon}} \log^2 n)$. We hope that future work on the $\frac{1}{2}$-Group Steiner problem will lead to algorithms with better running times.

Chekuri and Pál [11] have recently described a $O(\log^2 n)$-factor *quasi-polynomial time* algorithm for the Group Steiner problem. Since the algorithm involves set cover style arguments, this gives a $O(\log n)$ algorithm for the $\frac{1}{2}$-Group Steiner problem by stopping it when it covers half the groups. Thus our approximation algorithm is optimal if we allow for quasi-polynomial time.

The problem they solve is actually the SOP or sub-modular orienteering problem. Here each subset $X \subseteq V$ of a directed graph $G(V, E)$ has a reward function $f(X)$ which satisfies the sub-modular property. The objective is to construct a walk with maximum given length B such that the subset of vertices $V' \subseteq V$ covered by the walk has maximum reward $f(V')$. Their algorithm is reminiscent of Savitch's algorithm : the algorithm guesses the middle node of the optimal walk and then recurses two times. However here the second recursive call is dependent on the output of the first recursive call (i.e., the subset of vertices covered by it), unlike in Savitch's algorithm where the two calls are independent. We add that the question of a polynomial-time $O(\log^2 n)$ algorithm for Group Steiner is still open.

CHAPTER III

MARS ROVER

3.1 Introduction

D^* is a greedy heuristic planning method that is widely used to direct a robot in a terrain with initially unknown obstacles from given start to given goal coordinates. D^* always moves the robot along a shortest presumed unblocked path from its current coordinates to the goal coordinates, presuming that as-yet-unobserved portions of the terrain have no obstacles. It stops when it has reached the goal coordinates or determined that this is impossible. If movement along the current path is blocked by an obstacle, the shortest presumed unblocked path changes and D^* needs to replan. This can be implemented efficiently [48] and easily [27].

In robotics applications, the continuous terrain is usually discretized into a grid. Robot movement then corresponds to traversal from vertex to adjacent vertex in a grid graph. The graph is known in the sense that the vertices (grid cells) and edges are known. Impassable features of the terrain, which determine the graph's structure, may be known via satellite reconaissance, prior exploration, or mapping. The graph is unknown in the sense that vertices of the graph may be blocked by debris, crevices, or other obstacles. An obstacle is not known until the robot's sensors detect it, for example, as the robot attempts to move to it.

D^* is also used in other AI applications to reach a desired goal state from an initial starting state [51, 22, 31, 52]. In these applications, and in some terrains such as buildings, the graph may be a Voronoi or other type of graph rather than a grid graph. In all of these applications the vertices can be recognized – in the case of robot movement, by the physical coordinates; in other planning problems by state identifiers.

The D^* algorithm has some advantages over depth first search (DFS) in practice, including ease of replanning if the robot is moved to a new location, empirically good average performance, and effective use of partial terrain information ([29]). D^* has been used outdoors on an autonomous high-mobility multi-wheeled vehicle that navigated 1,410 meters to the goal location in an unknown area of flat terrain with sparse mounds of slag as well as trees, bushes, rocks, and debris [51]. As a result of this demonstration, D^* is now widely used in the DARPA Unmanned Ground Vehicle (UGV) program, for example, on the UGV Demo II vehicles. D^* is also being integrated into a Mars Rover prototype (according to Anthony Stentz), tactical mobile robot prototypes and other military robot prototypes for urban reconnaissance [22, 31, 52]. Furthermore, it has been used indoors on Nomad 150 mobile robots in robot-programming classes to reach a goal location in unknown mazes [37, 36]. D^* has also been used as the key method in various robot-navigation software [8, 50].

Given its simple form and many applications it would be quite interesting to know analytically how well D^* performs. The measure by which we assess performance here is the worst case distance traveled by the robot. We focus on travel distance in the terrain rather than travel planning time because robots move so slowly that the task-completion times are completely dominated by their travel times.

For the rest of the chapter, n denotes the number of vertices in the terrain graph $G = (V, E)$. In practice D^* seems to perform reasonably well and, in many domains, exhibits a performance that is linear in n [29], i.e. the same order as DFS, but it is not known whether this is due to properties of the test terrains or whether the plan-execution times are indeed guaranteed to be good in any terrain. However, in [29] it was also shown that for arbitrary graphs the performance is $\Omega(n \log n / \log\log n)$ (see section 3.5). A considerably modified version of the construction in [29] gives the same $\Omega(n \log n / \log\log n)$ bound for grid graphs [35]. This part of the research was done jointly with Sam Greenberg and Craig Tovey. This establishes that D^* has superlinear worst case performance on the class of graphs used in real robotics applications.

The best upper bound on D^* previously known was $O(n^{3/2})$ [28]. We prove an upper bound of $O(n \log n)$ for planar graphs. This leaves only a $\log\log n$ gap, and establishes that D^* is only slightly inferior to DFS in this worst-case performance sense. As mentioned above, D^* is also employed for other applications in which the graph may not possess the grid structure. For arbitrary graphs we prove an upper bound of $O(n \log^2 n)$. Thus D^* has a rather good performance guarantee in general.

In sections $3.2 - 3.3$ we assume that the robot has tactile (short-range) sensors. In section 3.4.1

we extend the results to long-range sensors. In particular, the lower bound applies to any line-of-sight sensor, and the upper bounds apply to all sensor types. In section 3.4.2 we extend results to the case where both vertices and edges may be blocked.

3.2 Definitions

We assume that the robot is omni-directional, point-sized, equipped with a tactile (short-distance) sensor, and capable of error-free motion and sensing. The sensors on board the robot uniquely identify its location. We model the terrain as a graph. Vertices in the graph represent locations in the terrain. Traversing an edge in the graph corresponds to traveling from one location to an adjacent location in the terrain. We are interested in the quality of the plans determined by D^* as a function of the number of vertices of the graph.

With these assumptions, we can formalize the behavior of D^* as follows. We call a graph $H = (V, E)$ vertex-blocked by $B \subset V$ if B is the set of blocked vertices, vertices that cannot be traversed. On a finite undirected graph $H = (V, E)$ vertex-blocked by B, a robot has to reach a designated goal vertex t from a start vertex s. D^* always moves the robot from its current vertex along a shortest presumed unblocked path to the goal vertex. A presumed unblocked path is one that contains no vertices which are known to be blocked. Initially, the robot has no information about B except that $s \notin B$. If the robot attempts to move to a blocked vertex v, it learns that $v \in B$. D^* then recomputes a new presumed unblocked path to begin the next iteration. D^* terminates when the robot reaches the goal vertex or there are no presumed unblocked paths to the goal vertex, in which case the goal vertex is unreachable from the start vertex. Additional notation to formalize the information state of the robot is given in the next section

3.3 D^*: An $O(n \log n)$ Upper Bound

3.3.1 Notation

As defined in section 3.2, the robot knows the graph $H = (V, E)$, the starting location $s \in V$ and a goal vertex $t \in V$. However, it does not know which vertices in V are blocked. D^* travels along a shortest presumed unblocked path to t. If the robot has tactile sensors it replans whenever it encounters a blocked vertex along its currently planned path. To prepare the way for an extension to long-range sensors in the next section, we analyze here a slightly more general case. We permit the robot to detect a blocked vertex some distance ahead on its planned path. For example, in Figure 12, the robot starting from 0 might travel as far as 2, and then detect blocked vertex 6. Note that an earlier vertex such as 4 might be blocked, but go undetected at this iteration.

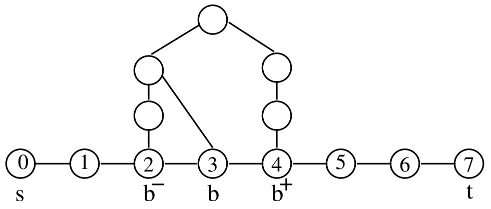


Figure 12: When the blockage at b of P_i is detected, $d(2,t)$ increases by at most $d(b^-,b^+)^{H^{i+1}} - 2 = 4$.

We assume that the initial graph $H = (V, E)$ given to the robot is connected with $n = |V|$ vertices (if not, take the component containing the starting vertex). The starting and target vertices are denoted $s, t \in V$ respectively. At the start of the ith iteration of D^*, let v_{i-1} denote the robot's location and $H^i = (V, E_i)$ denote its current information about the environment. E_i is obtained from E by removing all edges incident on vertices that have been found to be blocked. Initially $v_0 = s$ and $H^1 = H$. Let P_i denote the shortest path in H^i from v_{i-1} to t that the robot decides to follow. If i is not the final iteration, let b_i be the vertex found to be blocked by the robot while following P_i. H^{i+1} is obtained from H^i by removing edges incident on b_i. Let b_i^- and b_i^+ denote respectively the vertices preceding and following b_i on P_i. See Figure 12. Let v_i be the starting vertex for the next iteration. Clearly v_i either is or precedes b_i^- in P_i and the subpath of P_i between v_i and b_i^- exists in H^{i+1}. Also, the subpath of P_i from b_i^+ to t exists in H^{i+1}.

Let $d(u, v)^H$ denote the shortest distance between vertices u and v in graph H. If u and v are not connected then $d(u, v)^H = \infty$.

Let $v_0, v_1, \ldots, v_k$ be a run of the method.This captures a run up to ties in shortest viable paths. If the robot reaches t then $v_k = t$. The total distance traveled by the robot is:

$$C = \sum_{i=1}^{k} d(v_{i-1}, v_i)^{H^i}$$

3.3.2 Telescoping

Lemma 1 $C \leq n + \sum_{i=1}^{k-1} d(b_i^-, b_i^+)^{H^{i+1}}$

Proof: Since v_i lies on the shortest path P_i from v_{i-1} to t in H^i, by the principle of optimality:

$$C = \sum_{i=1}^{k} d(v_{i-1}, v_i)^{H^i} = \sum_{i=1}^{k} (d(v_{i-1}, t)^{H^i} - d(v_i, t)^{H^i})$$

$$= d(v_0, t)^{H^1} - d(v_k, t)^{H^k} + \sum_{i=1}^{k-1} (d(v_i, t)^{H^{i+1}} - d(v_i, t)^{H^i})$$

$$\leq n + \sum_{i=1}^{k-1} (d(v_i, t)^{H^{i+1}} - d(v_i, t)^{H^i}).$$

This formula has the following intuitive explanation: the robot optimistically thinks that undetected vertices are unblocked. When the robot gets to v_i and detects a blockage, it is set back in the distance it thinks it is from t, by the amount $(d(v_i, t)^{H^{i+1}} - d(v_i, t)^{H^i})$. The sum of these setbacks, plus the initial optimistic distance to t, equals the total distance traveled by the robot.

By the triangle inequality,

$$d(v_i, t)^{H^{i+1}} \leq d(v_i, b_i^-)^{H^{i+1}} + d(b_i^-, b_i^+)^{H^{i+1}} + d(b_i^+, t)^{H^{i+1}}. \quad (1)$$

By the principle of optimality, the subpath of P_i from v_i to b_i^- in H^i has length $d(v_i, b_i^-)^{H^i}$, the subpath of P_i from b_i^- to b_i^+ has length $d(b_i^-, b_i^+)^{H^i} = 2$, and the subpath of P_i from b_i^+ to t in H^i has length $d(b_i^+, t)^{H^i}$. Hence,

$$d(v_i, t)^{H^i} = d(v_i, b_i^-)^{H^i} + 2 + d(b_i^+, t)^{H^i}. \quad (2)$$

Observe that the first and third of these subpaths exist in H^{i+1}. Only the path of length 2 through b_i between b_i^- and b_i^+ is no longer viable in H^{i+1}. Therefore, $d(v_i, b_i^-)^{H^i} = d(v_i, b_i^-)^{H^{i+1}}$ and $d(b_i^+, t)^{H^i} = d(b_i^+, t)^{H^{i+1}}$. Plugging these equations, inequality 1 and equation 2 into the bound for C above yields the lemma. ■

In words, the amount of the setback when at v_i can't be more than the revised distance $d(b_i^-, b_i^+)^{H^{i+1}} - 2$ since the robot could splice in that path to replace the blocked b_i^-, b_i, b_i^+ portion of P_i. Notice that $d(b_i^-, b_i^+)^{H^{i+1}} < \infty$ because the following pairs are all in the same connected component in H^{i+1}: v_i and b_i^-; v_i and t; b_i^+ and t.

3.3.3 Time Reversal and Weighted Edges

Define the following function:

CYCLE-WEIGHT(T, S). *Input*: a tree $T = (V, F)$ and an ordered list $S = \{e_k, e_{k-1}, \ldots, e_1\}$ of distinct edges from the complete graph on V such that $S \cap F = \phi$. Define the weight w_i of edge $e_i \in S$ to be the length of a shortest cycle that contains e_i in the graph $T_i = (V, F \cup \{e_k, e_{k-1}, \ldots, e_i\})$.
Output: $\sum_{i=1}^{k} w_i$.

We next show that $\sum_{i=1}^{k-1} d(b_i^-, b_i^+)^{H^{i+1}}$ $\leq$CYCLE-WEIGHT(T, S) for a suitably constructed tree T and $S = \{e_i = (b_i^-, b_i) : 1 \leq i \leq k-1\}$.

The basic idea relating the edge weights in CYCLE-WEIGHT to the $d(b_i^-, b_i^+)^{H^{i+1}}$ values can be understood by considering a special case. Suppose H^k is connected except for the isolated vertices $b_1, b_2, \ldots, b_{k-1}$. Reverse the time perspective so that the robot motion *adds* edges, first the edges incident on b_{k-1}, then the edges incident on b_{k-2}, and so on. Pick T to be a spanning tree of the graph $(V, E_k \cup \{(b_1, b_1^+), (b_2, b_2^+), \ldots, (b_{k-1}, b_{k-1}^+)\})$ and S to be $e_i = (b_i^-, b_i) : 1 \leq i \leq k-1$. Then $w_i \geq 2 + d(b_i^-, b_i^+)^{H^{i+1}}$ because any cycle containing (b_i^-, b_i) in T_i must also contain (b_i, b_i^+).

Unfortunately such a simple construction does not work in the general case as multiple connected components may be formed when the edges incident to a blocked vertex are removed. To get around this problem, we define a new sequence of graphs $F_k, F_{k-1}, \ldots, F_1$ as follows:

1. F_k is a spanning forest of H^k.
2. For $1 \leq i \leq k-1$, let C^i be the connected component of H^{i+1} containing b_i^+ and b_i^-. Then F_i is a spanning forest of H^i containing the subgraph $F_{i+1} \bigcup \{(b_i, b_i^+)\}$.

The following lemma follows by induction directly from the definition of F_i:

Lemma 2 *For $1 \leq i \leq k$ and all vertices u and v, F_i is acyclic; $d(u,v)^{F_i} < \infty$ iff $d(u,v)^{H_i} < \infty$; and $d(u,v)^{F_i} \geq d(u,v)^{H^i}$.*

Consider the cycle weight problem with $T = F_1$ and $S = \{e_i = (b_i, b_i^-) : 1 \leq i \leq k-1\}$. The next lemma bounds the cost of our method by CYCLE-WEIGHT(T,S):

Lemma 3 *Let $H^1, H^2, \ldots, H^k$ be a sequence of graphs as defined in section 3.3.1. Let $T = F_1$ and $S = \{e_i = (b_i^-, b_i) : 1 \leq i \leq k-1\}$. Then $\sum_{i=1}^{k-1} d(b_i^-, b_i^+)^{H^{i+1}}$ $\leq$CYCLE-WEIGHT(T,S).*

Proof: According to Lemma 2, F_{i+1} and H^{i+1} have the same connected components. The subgraph of F_1 induced by C^i is connected since C^i is a component of H^{i+1}. The edges e_j for $i < j < k$ are contained in C^i since $b_j^-, b_j, b_j^+ \in C^i$ for all $i < j < k$. Thus, the graph obtained by contracting all vertices of C^i in T_{i+1} is acyclic. Since T_i is obtained from T_{i+1} by adding e_i, every cycle that contains $e_i = (b_i^-, b_i)$ in T_i must also contain (b_i, b_i^+). Thus, w_i is equal to 2 plus the distance between b_i^- and b_i^+ in the subgraph G' of T_i induced by C^i. But G' is also a subgraph of H^{i+1} and hence it holds that $w_i \geq 2 + d(b_i^-, b_i^+)^{H^{i+1}}$. Consequently, $\sum_{i=1}^{k-1} d(b_i^-, b_i^+)^{H^{i+1}} \leq \sum_{i=1}^{k-1} w_i = \text{CYCLE-WEIGHT}(T, S)$. ■

3.3.4 An Extremal Problem on Graphs

We now bound CYCLE-WEIGHT$((V, E), S)$ in terms of $|V|$ and $|S|$. Let $E_w = \{e_i; w_i \geq w\}$ be the set of edges with weight at least w. Recall that the *girth* of a graph is the length of its shortest cycle. Define $\Gamma(n, w)$ (respectively $\Gamma_P(n, w)$) to denote the maximum number of edges in a graph (respectively planar graph) with n vertices and a girth of at least w. The following lemma relates E_w and $\Gamma(n, w)$.

Lemma 4 $|E_w| \leq \Gamma(|V|, w) - |V| + 1$ *for all CYCLE-WEIGHT$((V, E), S)$ and all w.*

Proof: Consider the graph $T_w = (V, E \cup E_w)$. We claim that T_w has a girth of at least w. To see this, assume that it does not and thus has a cycle C of length $w' < w$. Since (V, E) is a tree, at least one edge of C must belong to E_w. Consider the edge $e_j \in E_w \cap C$ with the smallest j. Then T_j contains C and thus $w_j \leq w' < w$. On the other hand, $w_j \geq w$ since $e_j \in E_w$, which is a contradiction. Thus, T_w has a girth of at least w. This implies that $\Gamma(|V|, w) \geq |E \cup E_w| = |E| + |E_w| = |V| - 1 + |E_w|$ and the lemma follows. ■

Corollary 1 $|E_w| \leq \Gamma_P(|V|, w) - |V| + 1$ *for all CYCLE-WEIGHT*$((V, E), S)$ *such that* $(V, E \cup S)$ *is planar, and all* w.

Proof: In the proof of lemma 4, T_w is planar because it is a subgraph of planar graph $(V, E \cup S)$. Hence $\Gamma(|V|, w)$ may be replaced by $\Gamma_P(|V|, w)$. ■

We now bound CYCLE-WEIGHT$((V, E), S)$ by making use of bounds on $\Gamma(n, w)$, a well studied problem in extremal combinatorics. We first consider the case that the graph $(V, E \cup S)$ is planar.

Lemma 5 $\Gamma_P(n, w) \leq \frac{wn}{w-2}$ *for all* n *and* w.

Proof: Since the sum of the lengths of all faces of any planar graph $G = (V, E)$ is at most $2|E|$ and every face has length at least w, the number of its faces can be at most $2|E|/w$. The bound of the lemma follows from substituting this relationship in Euler's formula. ■

Note that the weight of any edge in S is at most $|V|$. Define $E_{w,2w} = \{e_i \in S : w \leq w_i < 2w\}$. Then, by corollary 1 and lemma 5 it holds that

$$\begin{aligned}
\text{CYCLE-WEIGHT}((V, E), S) &\leq \sum_{i=1}^{\log|V|} 2^{i+1}|E_{2^i,2^{i+1}}| \\
&\leq O(|S|) + \sum_{i=3}^{\log|V|} 2^{i+1}|E_{2^i}| \\
&\leq O(|S|) + \sum_{i=3}^{\log|V|} 2^{i+1}(\Gamma_P(|V|, 2^i) - |V| + 1) \\
&\leq O(|S|) + \sum_{i=3}^{\log|V|} 2^{i+1}(\frac{2^i|V|}{2^i - 2} - |V| + 1) \\
&\leq O(|S|) + \sum_{i=3}^{\log|V|} 2^{i+1}\, 4|V|/2^i \\
&= O(|V| \log |V|).
\end{aligned}$$

The last inequality depends on planarity (so $S = O(|V|)$) and $|V| \geq 6$. We now repeat the analysis for general graphs. In this case, we use a recent result by Alon, Hoory and Linial that states that any graph $G = (V, E)$ with average degree $d > 2$ has a girth of at most $\log_{d-1} |V|$ [1], resulting in the following lemma.

Lemma 6 $\Gamma(n, w) \leq n(n^{\frac{1}{w}} + 1)/2$ *for all n and w.*

Proof: Consider any graph $G = (V, E)$ with $|V| = n$, $|E| \geq |V| + 1$ and a girth of at least w. Then, its average degree is $d = 2|E|/n > 2$ and thus, according to the result by Alon, Hoory and Linial, $w \leq \log_{2|E|/n-1} n$. Solving this inequality for $|E|$ yields the lemma. ■

This lemma allows us to bound CYCLE-WEIGHT$((V, E), S)$ for general graphs.

Lemma 7 $w(|V|(|V|^{\frac{1}{w}} - 1)) = O(|V| \log |V|)$ *for* $|V| \geq w > \log^2 |V|$.

Proof: Let $n = |V|$ and remove the common factor $|V|$ from the statement of the lemma. The resulting left hand side defines the function $f(w) = w(n^{\frac{1}{w}} - 1)$. Its derivative is

$$f'(w) = n^{\frac{1}{w}}(1 - \frac{\ln n}{w}) - 1$$

and its second derivative is

$$f''(w) = \frac{n^{\frac{1}{w}} \ln^2 n}{w^3} > 0.$$

Therefore f is convex (in the range $w > 0$). Hence $\arg\max_{n \geq w \geq \log^2 n} f(w)$ occurs at one of the endpoints of the range, n or $\log^2 n$. We will show that $f(w) = O(\log n)$ for both endpoints.

At $w = n$, let $t = \frac{\ln n}{n} \to 0$ as $n \to \infty$. The Taylor series for e^t around 0 then gives

$$n^{\frac{1}{n}} - 1 = e^{\frac{\ln n}{n}} - 1 = e^t - 1 = \frac{\ln n}{n} + \frac{ln^2 n}{2n^2} + o(n^{-2}) = O(\frac{\log n}{n}).$$

Thus $f(n) = O(\log n)$.

At $w = \log^2 n$, let $t = \frac{\ln 2}{\log n}$, so

$$\frac{f(w)}{\log n} = \log n(n^{\frac{1}{\log^2 n}}) - 1 = \log n(e^{\frac{\ln n}{\log^2 n}} - 1) = \log n(e^{\frac{\ln 2}{\log n}} - 1) = \frac{\ln 2}{t}(e^t - 1).$$

Again using the Taylor series we get $\frac{f(w)}{\log n} = \ln 2(1 + \frac{t}{2} + \frac{t^2}{6} + \ldots) = \ln 2(1 + o(1)) = O(1)$. ■

Using lemmata 7, 4 and 6, we have

$$\begin{aligned}
\text{CYCLE-WEIGHT}((V,E),S) &= \sum_{i:w_i \le \log^2|V|} w_i + \sum_{i:w_i > \log^2|V|} w_i \\
&\le |S|\log^2|V| + \sum_{i=2\log\log|V|}^{\log|V|} 2^{i+1}|E_{2^i,2^{i+1}}| \\
&\le |S|\log^2|V| + \sum_{i=2\log\log|V|}^{\log|V|} 2^{i+1}|E_{2^i}| \\
&\le |S|\log^2|V| + \sum_{i=2\log\log|V|}^{\log|V|} 2^{i+1}(\Gamma(|V|,2^i) - |V| + 1) \\
&= |S|\log^2|V| + \sum_{i=2\log\log|V|}^{\log|V|} 2^{i+1}(|V|(|V|^{\frac{1}{2^i}} - 1)/2 + 1) \\
&= |S|\log^2|V| + \sum_{i=2\log\log|V|}^{\log|V|} O(|V|\log|V|) \\
&= O((|V| + |S|)\log^2|V|).
\end{aligned}$$

We now state these results as a lemma.

Lemma 8 *CYCLE-WEIGHT*$((V,E),S) = O((|V| + |S|)\log^2|V|)$. *If the graph* $(V, E \cup S)$ *is planar, CYCLE-WEIGHT*$((V,E),S) = O(|V|\log|V|)$.

3.3.5 Worst-Case Travel Bound

We are now ready to prove an upper bound on the worst-case travel distance of D^*.

Theorem 4 *For robot sensors as described in section 3.3.1,* D^* *traverses* $O(n\log^2 n)$ *edges on connected graphs* $G = (V,E)$. *It traverses* $O(n\log n)$ *edges on connected planar graphs* $G = (V,E)$.

Proof: According to Lemmata 1 and 3, D^* traverses at most $O(n) + \sum_{i=1}^{k-1} d(b_i^-, b_i^+)^{H^{i+1}} \le O(n) +$ CYCLE-WEIGHT$((V,E'),S)$ edges, where $|S| < n$ and $(V, E' \cup S)$ is a subgraph of G. According to Lemma 8, it holds that CYCLE-WEIGHT$((V,E'),S) = O((n + |S|)\log^2 n) = O(n\log^2 n)$ and, if G and thus $(V, E' \cup S)$ are planar, CYCLE-WEIGHT$((V,E'),S) = O(n\log n)$. ■

3.4 Extensions

3.4.1 Long Range Sensors

The upper bounds of the previous sections extend to the case of long range sensors, rather than the tactile sensors we have assumed so far. Many real robots are equipped with sonar, radar, or laser sensors, so it is worthwhile to consider this case. In directions where the view is not blocked by obstacles, these sensors can detect at moderate or even unlimited distances.

We now extend the upper bound to the case of long range sensors. We will not require that the sensors be field-of-vision. They may see around corners, have gaps in their vision, etc. We only require that if the robot attempts to move to vertex $v \in B$ from a vertex adjacent to v, then the robot will detect that $v \in B$. This is a minimal property required for any functioning robot.

Theorem 5 *Suppose that the robot follows the D^* algorithm on graph $H = (V, E)$. Each time the robot attempts to move to an adjacent vertex, it either moves successfully or it detects that the vertex is blocked. After an attempted move (whether successful or not) the robot may detect additional blocked vertices in H. Then the bounds of Theorem 4 apply.*

Proof:

Our proof consists of two parts. Part 1 shows that our bounds apply if the robot detects blocked vertices that are not on the planned path to the target. Part 2 shows that if more than one blocked vertex on the planned path is detected, then there exists a different robot whose movements are the same, but which does not detect more than one blocked vertex on the planned path.

We preface part 1 by stating the very simple ideas hidden in the technical statements. Blocked vertices off the path do not affect the telescoping formula of Lemma 1, because, by definition, they don't affect the current path. When we reverse time and add the special edges $e_k, \ldots, e_1$, we add extra edges (those connected to the off-path vertices). Our upper bound is on the length of a smallest cycle containing e_i, so adding extra edges can only make this smaller. Therefore the upper bound, which is computed in Lemma 8 as though there were no extra edges, is still valid.

Let $B_i \subset V$ denote the off-path vertices detected as blocked in iteration i. The definitions of v_i and H^i remain the same as in section 3.3.1, but now H^{i+1} is obtained from H^i by removing all edges incident on b_i or incident on any $b \in B_i$. Lemma 1 remains true in this setting because no vertices in B_i are on the path P_i. In particular, the subpaths of P_i from v_i to b_i^- and from b_i^+ to t still exist in H^{i+1}. Intuitively, the blockages B_i contribute to the setback amount suffered by the robot, but this setback is still bounded by the change in distance from b_i^- to b_i^+.

For the associated cycle weight problem, we define a sequence of forests $F_k, F_{k-1}, \ldots, F_1$. As before, F_k is a spanning forest of H_k and F_i is a spanning forest of H_i containing the subgraph $F_{i+1} \bigcup \{(b_i, b_i^+)\}$. It is easy to show that taking $T = F_1$ and $S = \{e_i = (b_i, b_i^-) : 1 \leq i \leq k-1\}$ satisfies Lemma 3. Therefore we've verified part 1.

Based on part 1, the bounds of Theorem 4 apply as long as the robot never detects more than one blocked vertex on the current planned path to t. For the second part of the proof, whenever the robot detects more than one such blocked vertex, categorize the detected vertices as follows:

off-path all vertices not on the current planned (shortest presumed unblocked) path to t.

first-path the nearest detected blocked vertex on the current planned path to t.

more-path all other detected blocked vertices on the current planned path to t.

Consider now a fictional robot whose movements have been identical to the real one, and which until the present step has detected the same set of blocked vertices. Now, however, our fictional robot only detects the off-path vertices and first-path vertex. It replans the shortest presumed unblocked path to t, moves zero steps, and then considers detecting the more-path vertices (more-path with respect to the original plan, not the new plan). It detects all of those which are off the newly replanned path. It can also detect one vertex on the new path, if there is one. If more than one of these are on the new path it re-categorizes them with respect to the new path and repeats the procedure.

This procedure must terminate, because each replan strictly decreases the number of more-path vertices. At termination, the fictional robot has performed precisely the same set of physical movements as has the real robot, and it has detected the same set of blocked vertices. The fictional robot has never detected more than one blocked vertex on its current planned path. The desired bounds therefore apply to both it and the real robot. ■

3.4.2 Blocked Edges

Another natural extension is when in addition to blocked vertices $B \subset V$, some edges $B' \subset E$ might also be blocked. This can be reduced to the vertex blocking case by adding a new vertex v_e in the middle of every edge $e \in E$. Blocking of e then corresponds to blocking of vertex v_e in the tranformed graph. We consider two cases.

First assume that the robot does not expend travel cost to detect an incident blocked edge. Then if the robot encounters a blocked edge (u, v) while going from u to v, it can sense all other edges emanating from u to check which ones are blocked at zero additional cost. Thus the robot will stop in at most n iterations. To bound the travel cost, let (b_i^-, b_i^+) be the edge found blocked by the robot in iteration i. Lemma 1 remains true in this setting as the subpaths of P_i from v_i to b_i^- and from b_i^+ to t still exist in H^{i+1}. For the associated cycle weight problem, H^{i+1} is now obtained from H^i by removing all edges found blocked by the robot in iteration i. Define the sequence $F_k, F_{k-1}, \ldots, F_1$ by taking F_k a spanning forest of H^k and F_i a spanning forest of H^i containing the subgraph F_{i+1}. Similar arguments show that $T = F_1, S = \{(b_i^-, b_i^+) : 1 \leq i \leq k-1\}$ satisfies Lemma 3. By arguments for long-range sensors above, the bounds in Theorem 4 also hold when the robot detects a combination of blocked vertices and edges in each iteration.

Next we assume that the robot must traverse an edge in order to detect edge blockage. In this case detecting blocked e in the original graph corresponds to traveling to vertex v_e in the transformed graph. However the number of vertices in the transformed graph is $|V| + |E|$ and theorem 4 gives an $O(|E| \log |E|)$ upper bound for planar graphs and $O(|E| \log^2 |E|)$ upper bound for general graphs. For planar graphs this is still $O(n \log n)$ since $|E| = O(n)$. We next show a lower bound of $\Omega(|E|)$ for D^* on general graphs. Thus our bounds leave a $O(\log^2 |E|) = O(\log^2 n)$ gap.

Consider the graph $H = (\{s, t\} \bigcup X \bigcup Y, E)$ as shown in Figure 13 where $|X| = |Y| = \frac{n}{2}$. Assume that all edges $E' \subset E$ between X and Y are blocked without the knowledge of the robot. Imagine a little twist towards the end of each edge $e \in E'$, so the robot has to travel to the twist to find out whether e is blocked. Now consider running D^* with start vertex s and target vertex t. As long as there exists a "presumed unblocked" edge $(x, y) \in E'$ at the start of iteration i, the robot has a length 2 path $v_{i-1} - y - t$ or a length 4 path $v_{i-1} - s - x - y - t$ available to it. Therefore

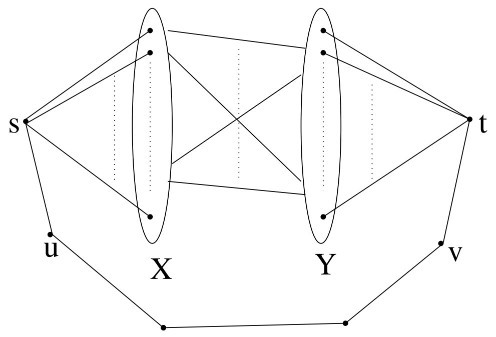


Figure 13: Lower bound example for blocked edges.

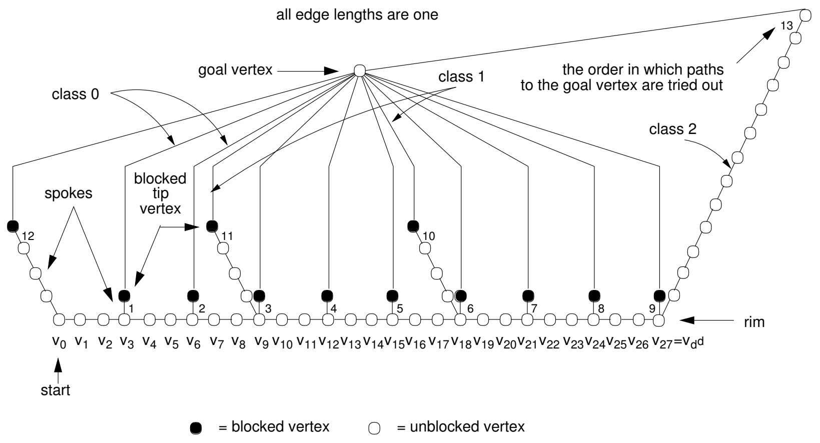


Figure 14: Example Graph for Lower Bound [29]

the robot will not take the length 6 path $v_{i-1} - s - u - \ldots - v - t$ until iteration $|E'| + 1$. In each preceding iteration, the robot will travel on edge (x, y) till the twist near y, find it to be blocked and then come back to x. Therefore its travel cost is at least $\Omega(|E'|) = \Omega(|E|)$ steps on H.

3.5 D^*: Lower Bound on Planar Graphs

We review the construction of [46, 29], which employs the key idea of tricking the robot into traversing the same long path back and forth many times. Second, we give an overview of how to transform that example into a grid without losing the key idea. Third, we explain exactly how the grid is constructed. Last, we analyze the worst-case travel distance of D^* on our grid graph, proving the lower bound.

The analysis of [46, 29] proved that the worst-case travel distance of D^* is $\Omega(\frac{n \log n}{\log \log n})$ steps on vertex-blocked graphs $H = (V, E)$. This lower bound is achieved with graphs of the structure shown in Figure 14. We now sketch the main idea of its construction, but with our own "rim-and-spoke" terminology, in order to introduce our much more complex grid construction.

The graph of Figure 14 consists of a long horizontal path of length d^d (where d is a integer parameter), which we call the "rim", and a set of "spokes" of varying lengths attached to the rim at various vertices. The uppermost "tip" vertex of each spoke is blocked and connected to the goal vertex by an edge. Note that the edges from the tips to the goal are physically unrealistic edges, because they allow the robot to move from any tip to the goal in one step. The possible spoke lengths are $\sum_{i=0}^{h} d^i$ for the nonnegative integers $h = 0 \ldots d-1$. We refer to a spoke of length $\sum_{i=0}^{h} d^i$ as a "class h spoke". Longer spokes are spaced farther apart from each other than short spokes. In particular, the vertices where class h spokes attach to the rim have distance d^{h+1} from each other. Hence, if the robot is at a vertex where a class h spoke attaches to the rim, then it is shorter to go to the goal along the rim to the next class h spoke, than it is to go via any class $h+1$ spoke.

In particular, in Figure 14 there are three classes of spokes: 0,1 and 2. The robot does not know that the shortest unblocked path to the goal from starting vertex v_0 is to traverse the rim to v_{27}, then the long class 2 spoke, and reach the goal vertex. Instead, the robot tries to reach the goal through the shortest presumed unblocked path via the short class 0 spoke at v_3, then the class 0 spoke at v_6, and so on until it tries the class 0 spoke at the right end of the rim, v_{27}. From there, the shortest presumed unblocked path to the goal is via the class 1 spoke at v_{18}. Thus the robot is led to traverse the rim from right to left, checking each class 1 spoke. Finally the robot traverses the rim a 3rd time, reaching the goal via the class 2 spoke.

In general, the robot starts at vertex v_0; it traverses the rim from left to right, checking the class 0 spokes for a path to the goal vertex; then it returns along the rim from right to left, checking class 1 spokes for a path to the goal vertex, and so on. Each class forces the robot to traverse the rim once. Thus the total travel distance is $\geq d^{d+1}$. A computation shows that there are $O(d^d)$ vertices in the graph, and hence the total travel distance is $\Omega(\frac{n \log n}{\log \log n})$.

CHAPTER IV

GRID GRAPH LOCALIZATION

4.1 Introduction

To understand the ideas underlying our algorithm, let us assume a robot R with neighbourhood sensors moving on a grid graph G. The set of hypotheses H is a subset of traversable cells in G.

Place the robot at the origin of an infinite grid. For each grid cell $\gamma = (i, j)$, we can break the set of hypotheses into two sets $T(\gamma)$ and $B(\gamma)$. $T(\gamma)$ consists of all hypotheses $h \in H$ such that the grid cell $h + \gamma$ in the map G is traversable. $B(\gamma)$ consists of all $h \in H$ for which it is blocked.

We make cell $\gamma = (i, j)$ traversable or blocked according to whether $|T(\gamma)| > \frac{1}{2}|H|$ or not. The set of all traversable cells reachable from the origin is called the majority-rule map. We denote it by G_{maj} (see Fig. 15). Note that in general the majority-rule map will have holes.

The first thing to note is that *any deviation* from the majority-rule map will half-localize the robot. For if it finds an interior cell to be blocked, it will localize to $|B(\gamma)| \leq \frac{1}{2}|H|$. On the other hand, if it finds a cell on the boundary ∂G_{maj} to be traversable, it will half-localize to $T(\gamma)$ which is again less than $\frac{1}{2}|H|$.

This has two interesting consequences. First, a half-localizing robot will never cross the boundary of G_{maj}. Thus the execution of any half-localizing strategy S will lie inside the majority-rule map. Further, if the robot makes an observation o at a coordinate $\gamma \in G_{maj}$, it will half-localize unless o "agrees" with the majority-rule map. We call the observation consistent with the majority-rule map at γ the *majority observation*. The set of hypotheses $h \in H$ for which a robot located at $h + \gamma$ makes the majority observation is denoted by $Maj(\gamma)$ (see Fig. 15).

Now, let S be a half-localizing strategy. Place the robot at a hypothesis $h \in H$ for which it travels the maximum distance. Call the path traced by the robot with respect to the origin P. The only information gathered by the robot is at the observation points $(\gamma_0, \gamma_1, \ldots, \gamma_m)$. If the robot's observation at coordinate γ_i differs from the majority-rule map, it will half-localize and stop excecuting the strategy. Therefore, if the robot reaches γ_m, it means that it has made the

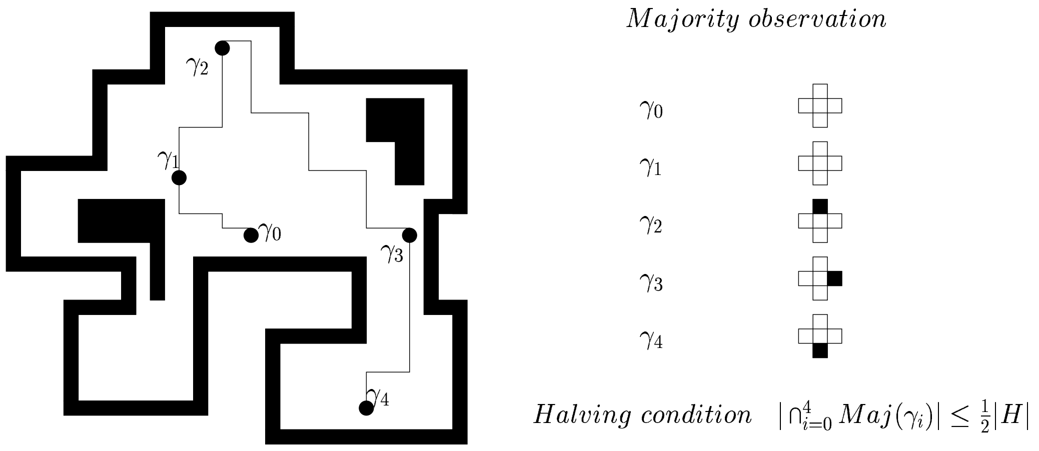


Figure 15: Majority-rule map G_{maj} with a halving path

majority observation at all previous γ_i. The set of hypotheses consistent with this are given by the set $\bigcap_{i=0}^{m} Maj(\gamma_i)$. Since the robot has half-localized, we get the condition that $|\bigcap_{i=0}^{m} Maj(\gamma_i)| \le \frac{1}{2}|H|$ (see Fig. 15 for an example with $m = 4$).

We call paths in the majority-rule map satisfying this condition $|\bigcap_{i=0}^{m} Maj(\gamma_i)| \le \frac{1}{2}|H|$ "halving paths". We have just shown that a half-localization strategy gives a halving path of cost $w(S)$. The converse is also true : a halving path P gives a strategy S of cost $|P|$!

The strategy given by P consists of moving the robot along P and making observations at $(\gamma_0, \gamma_1, \ldots, \gamma_m)$. The above argument shows that the robot will half-localize if (i) it makes a minority observation at γ_i, or (ii) if it makes majority observations at all γ_i, this being a consequence of the halving condition.

However, the proof is still not complete. We are left with one more possibility : what if the robot detects a blocked cell $\gamma \in P$? In this case, the robot will not be able to execute its strategy any further.

Here, the majority-rule map comes to our rescue again ! Since $\gamma \in G_{maj}$, finding it blocked will give us a deviation from the majority-rule map. Thus, the robot will half-localize to a subset of $B(\gamma)$ which is less than $\frac{1}{2}|H|$. (If it were otherwise, applying the majority-rule would have made cell γ blocked).

The second part of the algorithm consists of computing near-optimal halving paths in the majority-rule map. Let us try to find the exact algorithmic problem that captures this notion.

First, we will complement the sets $Maj(\gamma)$ so that the constraints on a halving path correspond to union of sets rather than their intersections. Thus, we will instead require that $|\bigcup_i \overline{Maj(\gamma_i)}|$ is at least $\frac{1}{2}|H|$.

Let $G(V,E)$ be the dual graph of the grid graph G_{maj}. This problem can now be visualized as follows : with each node $v \in V$ of $G(V,E)$ is associated a set S_v (this we take to be $\overline{Maj(v)}$). We want to visit a set of vertices $V' \subseteq V$ at minimum cost such that the union of their sets equals at least **half** the universe H.

We will strengthen this condition and instead require that the union equals the whole universe H.

Clearly this problem contains Set Cover as a special case. Given any instance $(\mathcal{U}, \{S_1, S_2, \ldots, S_m\})$ of set cover make a star of size m (a star is a tree with a single root and m leaves). We associate set S_i with the ith leaf.

On the other hand, one finds that Steiner Tree is also a special case. Given graph $G(V,E)$ with required vertices R, we associate to each $v \in R$ a unique set $\{a_v\}$. The symbols $\{a_v\}_{v \in R}$ are all distinct. Then a tree covering the universe $\mathcal{U} = \{a_v\}_{v \in R}$ will be a tree covering all required vertices R.

Now, it turns out that there is exactly one algorithmic problem which generalizes both Set Cover and Steiner Tree : the Group Steiner problem.

The formulation is a little different : instead of sets attached to vertices, we make sets of vertices (which we call "groups") $g_1, g_2, \ldots, g_{|H|}$ where set g_i consists of all vertices such that $h_i \in S_v$. Covering all elements in H is then equivalent to covering at least one vertex from each group. This gives the following definition :

(Rooted) Group Steiner Problem. Given a weighted graph $G = (V,E)$ with k groups of vertices $g_1, g_2, \ldots, g_k \subset V$, find a minimum weight tree that contains at least one vertex from each group. There is a distinguished vertex r (the *root* vertex) that must be included in the tree.

The first polylogarithmic approximation algorithm for Group Steiner problem was given by Garg, Konjevod and Ravi [19]. Their algorithm computes a Group Steiner tree of cost at most $O(\log^2 n \log k)$ in randomized polynomial time (see Section 2.2 and theorem 3).

To compute halving paths, we need to cover half the groups, therefore we use a variant, the $\frac{1}{2}$-Group Steiner problem [16], for which we have a $O(\log^2 n)$ algorithm.

Applying this algorithm to the halving path problem on G_{maj} with the root r as the origin γ_0 gives as an $O(\log^2 n)$-approximate half-localization algorithm. By repeatedly half-localizing at most $\log k$ times, we get an $O(\log^2 n \log k)$-approximate localization strategy.

We will now describe our algorithm in detail.

4.2 Preliminaries

During localization the robot makes observations from different positions in its environment (grid graph G) to make a larger and larger *local map* G', until there is exactly one hypothesis in H that is consistent with G'. We say that a hypothesis $h \in H$ is *active* if the robot's local map is consistent with it being located at h. We denote the set of active hypotheses by H'.

We distinguish between the absolute (global) position of the robot in the grid graph G, and its relative (local) position in G' by using Greek letters for the latter (whenever possible). Let γ_0 denote the initial position of the robot with respect to the local map G'. We call γ_0 the **origin**, and denote any other position in G' by a pair of coordinates. Coordinate $\gamma = (x, y)$ denotes the cell in G' lying x units to the east and y units to the north of γ_0. Negative values of x, y denote west and south, respectively. Thus a robot at coordinate $\gamma \in G'$ will be located at position $p_0 + \gamma$ in grid graph G, where $p_0 \in G$ is its *initial position.* The robot can keep track of its local coordinates by taking successive readings on the compass and odometer (we assume error-free motion and sensing during localization). At any point of time, the robot is sure of its local coordinates but knows its global position only up to cells in $H' + \gamma$.

Suppose the robot makes an observation when at coordinate $\gamma \in G'$. The outcome depends on its starting location $h \in H$. If the robot started from hypothesis h, the observation will be the same as that by a robot located at $h + \gamma$ in G. We denote this observation by $\mathcal{O}(h, \gamma)$ and call it the *opinion* of h about γ. If $h + \gamma$ is blocked, we set $\mathcal{O}(h, \gamma) = \emptyset$. The *hypothesis partition* $\mathcal{H}(\gamma)$ is a partition of the set of hypotheses according to the following equivalence relation: $h_1 \sim h_2$ if and only if $\mathcal{O}(h_1, \gamma) = \mathcal{O}(h_2, \gamma)$. $Maj(\gamma)$ denotes the largest size class of $\mathcal{H}(\gamma)$.

The "majority opinion" at γ is the opinion common to the plurality of hypotheses $h \in Maj(\gamma)$.

Note it may occur that $|Maj(\gamma)| < \frac{1}{2}|H|$. The lemmas that follow are valid in this case because the robot immediately half-localizes. Since there are two choices, *b*locked or *t*raversable, for each of the four neighboring cells of γ, an observation o can be written $o \in \{b,t\}^4 \bigcup \{\emptyset\}$, and we let $G(\gamma, o)$ denote the class of $\mathcal{H}(\gamma)$ with opinion o at γ.

4.3 The Majority-Rule Map

We now describe the majority-rule map G_{maj}, a data structure central to our half-localization algorithm.

Definition 1 *The majority-rule map G_{maj} is a local map in which each cell is blocked or traversable according to what the majority of hypotheses have to say about it (in case of a tie, we consider the cell to be traversable). The majority-rule map also includes the hypothesis partitions for all local coordinates.*

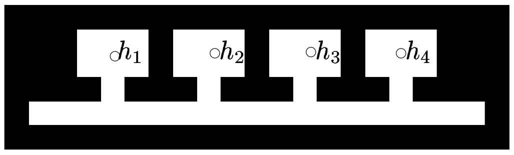
In other words, a cell $\gamma \in G_{maj}$ is blocked if and only if $\mathcal{O}(Maj(\gamma), \gamma) = b$ i.e., the opinion of the majority hypotheses is blocked.

If G is a $l \times m$ grid graph, the majority-rule map has size bounded by $(2l-1) \times (2m-1)$, since the absolute values of x- and y-coordinates for any hypothesis are at most $(l-1)$ and $(m-1)$, respectively. Clearly, G_{maj} requires space $4nk$ (there are at most $4n$ cells, and we need to store the partition for each cell) and can be computed in time $O(nk)$. Figure 16 shows the majority-rule map for grid graph G and $H = \{h_1, h_2, h_3, h_4\}$. The black region is unreachable by the robot for any starting hypothesis. The hypothesis partition is constant within each of the regions $R_0, R_1, R_2, R_3, R_4, R_5$ and R_6. R_5 and R_6 lie outside the grid graph for 3 different hypotheses and are blocked. Thus the only traversable regions are R_0, R_1, R_2, R_3 and R_4, with $Maj(R_0) = \{h_1, h_2, h_3, h_4\}$, $Maj(R_1) = \{h_2, h_3, h_4\}$, $Maj(R_2) = \{h_1, h_2, h_3\}$, $Maj(R_3) = \{h_3, h_4\}$ and $Maj(R_4) = \{h_1, h_2\}$.

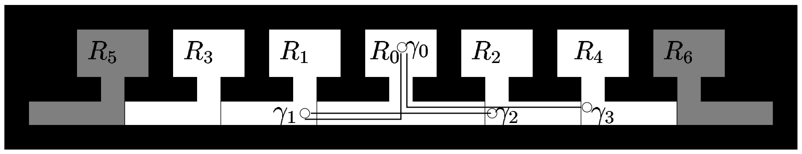
4.4 Halving Paths

We now define the notion of a halving path in the majority-rule map:

Definition 2 *A halving path is a (possibly self-intersecting) path $\mathcal{C} = (\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_m)$ in the majority-rule map satisfying $|\bigcap_{i=0}^{m} Maj(\gamma_i)| \leq \frac{1}{2}|H|$.*



(a)



(b)

Figure 16: (a) A half-localization problem with grid graph G and $H = \{h_1, h_2, h_3, h_4\}$. (b) The majority-rule map for HALF-LOCALIZE(G,H) with two halving paths $(\gamma_0, \gamma_1, \gamma_2)$ and (γ_0, γ_3).

The next two lemmas show an essential equivalence between half-localization strategies and halving paths.

Lemma 9 *Let $\mathcal{C}$ be a halving path. There exists a strategy $S(\mathcal{C})$ for half-localizing the robot with travel cost at most $|\mathcal{C}|$.*

Proof. Let $\mathcal{C} = (\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_m)$, where γ_{i+1} is a neighbor of γ_i in G_{maj}. A description of strategy $S(\mathcal{C})$ is as follows (see Algorithm 1): the robot traces path $\mathcal{C}$ from its initial position, taking observation o_i at each new coordinate γ_i. If the robot finds that the next coordinate is blocked, it stops. We next show that this will half-localize the robot correctly.

After observation o_i, the robot keeps only those hypotheses whose opinion at γ_i is o_i. Thus, it updates H' (the set of active hypotheses) correctly. We show that $S(\mathcal{C})$ reduces the set of hypotheses by half. If the robot finds that the cell at coordinate γ_i is blocked, it localizes to a set of size at most $|G(\gamma_i, \emptyset)| \leq \frac{1}{2}|H|$ (since $\gamma_i \in G_{maj}$). If observation o_i is different from the majority opinion at γ_i, $H' \subseteq G(\gamma_i, o_i)$, which has size at most $\frac{1}{2}|H|$. Thus the robot reaches γ_m if and only if for each $\gamma_i, 0 \leq i \leq m-1$, o_i is the majority opinion at γ_i. Now there are two cases: if o_m is different from the majority opinion, the robot half-localizes; otherwise $H' = \bigcap_{i=0}^{m} Maj(\gamma_i)$ which is again at most $\frac{1}{2}|H|$ (since $\mathcal{C}$ is a halving path) ■

In Figure 16, the halving path $\mathcal{C}_1 = (\gamma_0, \gamma_1, \gamma_2)$ satisfies $|\bigcap_{i=0}^{2} Maj(\gamma_i)| = |\{h_2, h_3\}| \leq \frac{1}{2}|H|$. The path (γ_0, γ_3) is an optimal halving path, with $|Maj(\gamma_0) \bigcap Maj(\gamma_3)| = |\{h_1, h_2\}| \leq \frac{1}{2}|H|$.

```
Data    : Grid graph $G$, set of hypotheses $H$ and a halving path $(\gamma_0, \gamma_1, \ldots, \gamma_m) \in G_{maj}$.
Result  : The robot half-localizes in at most $m$ steps.
Initialize $H' = H$
for $i = 0$ to $m - 1$ do
    begin
        Make observation $o_i$ at coordinate $\gamma_i$
        Update $H' = H' \bigcap G(p_i, o_i)$. Stop if $|H'| \leq \frac{1}{2}|H|$
        Move to coordinate $\gamma_{i+1}$
    end
end
Make observation $o_m$ at $\gamma_m$; Update $H' = H' \bigcap G(\gamma_m, o_m)$. Stop.
```

Algorithm 1: Strategy $S(\mathcal{C})$

For brevity we did not include intermediate uninformative cells in the description, assuming that the robot uses any shortest path in the majority-rule map to go from γ_i to γ_{i+1}. The behavior of a robot following strategy $S(\mathcal{C}_1)$ will be as follows. If it was placed at h_1, it will hit a wall at γ_1 and stop with $H' = \{h_1\}$. If it was placed at h_4, it will see a wall at γ_2, and stop with $H' = \{h_4\}$. If it was placed at either h_2 or h_3, it will make majority observations at both γ_1, γ_2 and half-localize to the set $\{h_2, h_3\}$ of hypotheses.

The next lemma shows that any half-localization strategy S has an associated halving path with length at most $W(S)$ (compare this with localization, for which strategies are decision trees [15], and hence hard to compute):

Lemma 10 *Let S be a strategy for half-localization. There exists a halving path $\mathcal{C}(S)$ of length at most $W(S)$, the cost of the strategy S.*

Proof. Consider a robot guided by S that stops as soon as it half-localizes. Let $\mathcal{C}(S) = (\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_m)$ be the maximum length path traced by the robot in its local map G' for any starting position in H. Let H_i denote the set of active hypotheses just after the robot makes an observation at coordinate γ_i. For $0 \leq i < m$, $|H_i| > \frac{1}{2}|H|$, since otherwise the robot would have stopped at γ_i itself. Each coordinate γ_i is unblocked for at least $|H_i| > \frac{1}{2}|H|$ hypotheses, and hence $\mathcal{C}(S)$ lies in the majority-rule map G_{maj}.

We claim that $I = \bigcap_{i=0}^{m} Maj(\gamma_i)$ is of size at most $\frac{1}{2}|H|$. Consider a robot initially located at some $h \in I$. Guided by S, the robot will follow path $\mathcal{C}(S)$ and make the majority observation o_i

at all coordinates γ_i (since $I \subset Maj(\gamma_i)$). But then $|\bigcap_{i=0}^{m} Maj(\gamma_i)| = |H_m| \leq \frac{1}{2}|H|$, and hence $\mathcal{C}(S)$ is a halving path. ■

4.5 Computing halving paths

Let $\mathcal{C}_H^*$ denote an optimal halving path for the set of hypotheses $H = \{h_1, h_2, \ldots, h_k\}$. We approximate the problem of computing an optimal halving path by reducing it to an instance $\mathcal{I}_{G,H}$ of the $\frac{1}{2}$-Group Steiner problem.

The reduction is a restatement of the problem in terms of groups: we take V as the set of traversable coordinates in the majority-rule map. The weight of edge (γ, γ') is the length of the shortest path joining cells γ and γ' in G_{maj}. Origin γ_0 is taken as the root vertex. We make k groups, one for each hypothesis $h_i \in H$. Group g_i is the set of all coordinates $\gamma \in V$ such that h_i does not share the majority opinion at γ, i.e., $h_i \notin Maj(\gamma)$. Thus a tree T covers k' groups if and only if $\bigcap_{x \in T} Maj(x)$ has size $k - k'$. In particular, T covers at least half the groups if and only if $|\bigcap_{\gamma \in T} Maj(\gamma)| \leq \frac{1}{2}|H|$. In particular, every halving path is a $\frac{1}{2}$-Group Steiner tree:

Lemma 11 *There exists an $O(\log^2 n)$-approximation algorithm for computing an optimal halving path.*

Proof. Let T be the tree output by algorithm $\mathcal{A}$ (see Theorem 3 in section 2.2) on instance $\mathcal{I}_{G,H}$. Then, the weight of T is at most $O(\log^2 n) \cdot w(T^*)$, where T^* is an optimal $\frac{1}{2}$-Group Steiner tree. Let $\mathcal{C}$ be the path of length at most $2 \cdot w(T)$ traced by a depth-first search on T starting from the origin. $\mathcal{C}$ is a halving path since $|\bigcap_{\gamma \in \mathcal{C}} Maj(x)| = |\bigcap_{\gamma \in T} Maj(x)| \leq \frac{1}{2}|H|$. Since any optimal halving path $\mathcal{C}_H^*$ covers half the groups, $w(T^*) \leq |\mathcal{C}_H^*|$. Therefore $|\mathcal{C}| = O(\log^2 n) \cdot |\mathcal{C}_H^*|$ ■

4.6 Strategy RHL

The overall strategy is as follows (see Algorithm 2). In each half-localize phase, the robot computes a near-optimal halving path $\mathcal{C}$, then traces $\mathcal{C}$ to reduce the set of (active) hypotheses by half. It retraces $\mathcal{C}$ to move back to its initial position, and proceeds with the next phase. We now bound the approximation factor and computation time of strategy **RHL**:

```
Data   : Grid graph $G$, the set of hypotheses $H$
Result : The robot localizes to its initial position $h \in H$
while $|H| > 1$ do
    begin
        Compute the majority-rule map $G_{maj}$
        Make instance $\mathcal{I}_{G,H}$ of $\frac{1}{2}$-Group Steiner problem
        Solve $\mathcal{I}_{G,H}$ to compute a halving path $\mathcal{C}$ (lemma 11)
        Half-localize by strategy $S(\mathcal{C})$ (lemma 10)
        Move back to the starting location
    end
end
```

Algorithm 2: Strategy **RHL** - **R**epeated **H**alf **L**ocalization

Theorem 6 *A robot guided by strategy* **RHL***(Algorithm* 2*) correctly determines its initial position* $h \in H$ *by traveling at most* $O(\log^2 n \log k) \cdot OPT(G, H)$ *distance where* $k = |H|$ *and* n *is the size of* G*. Further, the computation time of the robot is polynomial in* n*.*

Proof. Since the number of active hypotheses reduces by at least half after each phase, the robot localizes in $m \leq \lceil \log |H| \rceil = \lceil \log k \rceil$ phases. Let H_i denote the set of active hypotheses at the start of the ith phase. By lemma 11, the distance traveled by the robot in the ith phase is at most $O(\log^2 n) \cdot |\mathcal{C}^*_{H_i}|$. By lemma 10, $|\mathcal{C}^*_{H_i}| \leq$ HALF-OPT(G,H_i) $\leq$ OPT(G,H), where the last inequality follows from the fact that any localization plan also reduces the set of hypotheses by half. Therefore, the distance traveled by the robot in each phase is at most $O(\log^2 n)\cdot$ OPT(G,H). Since there are $O(\log k)$ phases, the total worst-case travel distance is $O(\log^2 n \log k)\cdot$OPT(G,H). Since instance $\mathcal{I}_H$ can be constructed in $O(nk)$ time, the computation time is at most $O(\mathcal{P}(nk) \cdot \log n)$ where $\mathcal{P}()$ (a polynomial) is the time taken by the approximation algorithm for $\frac{1}{2}$-Group Steiner (see Section 2.2).

■

The above theorem shows the performance ratio for a robot with very weak sensors; the robot can only “see” four neighboring cells. We note that all of the theorems of this section hold for robots on grid graphs with other kinds of sensors such as range-finders or sonar. An interesting feature of our strategy is that it is well-suited to handling the problem of accumulation of errors caused by successive motion in the estimates of orientation, distance and velocity by the robot’s odometer. This is because after each half-localize phase the robot always returns to the origin, which it can use to recalibrate its sensors [15].

CHAPTER V

INAPPROXIMABILITY

5.1 Introduction

We now show a $\Omega(\log^{2-\epsilon})$ lower bound for localization by a reduction from the hardness of Group Steiner problem.

Recently, Halperin and Krauthgamer [21] showed that there is no $\Omega(\log^{2-\epsilon} n)$ algorithm for Group Steiner problem unless $NP \subseteq ZTIME(n^{polylog(n)})$. This was one of the first problems for which polylogarithmic inapproximability was shown.

In this chapter we show that their construction can be used to derive a $\Omega(\log^{2-\epsilon})$ lower bound for the robot localization problem.

First we note the following fact : the gap instances constructed by Halperin and Krauthgamer are rooted trees $T(V, E)$ and the groups are the subsets of their leaves. Note that the edges of a tree also have weights. An example instance is shown in Figure 5.1.

One can embed a weighted tree into a grid graph by making vertical corridors for edges and horizontal corridors for connecting all siblings of a parent. The width of the horizontal corridors can be kept small enough so that distances on the grid graph are within a constant factor of the distances on the tree. Figure 5.1 shows how to construct a grid graph for a tree with height 2 and 4 leaves.

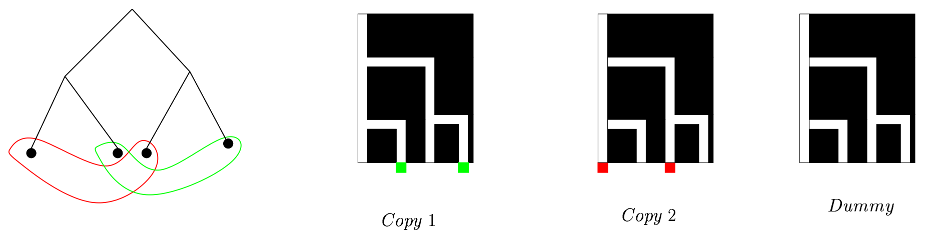


Figure 17: Inapproximability

Now we come to defining our localization problem. We will make k disjoint copies $G_1, G_2, \ldots, G_k$ of the grid graph embedding, one for each group g_i in the tree. The robot will be located at the root of one of these copies and the localization task will consist of finding the index of the copy.

To complete the reduction, we add distinguishing *signatures* at the leaves of these copies : copy G_i contains signature i at all leaves which belong to group i. For the example in Figure 5.1, we make two distinct copies G_1 and G_2. Signature 1 (red) is placed on leaves 1 and 3 of G_1 and signature 2 (green) is placed on leaves 2 and 4 of G_2. We also add a dummy copy G_0 with no signatures at leaves.

Thus, there are k distinct signatures and once the robot reaches a leaf with a signature in its component it finds the index of its copy.

Any localization plan for the robot consists of visiting a sequence of leaves $l_1, l_2, \ldots, l_m$ and reading the signatures at them. If the leaves do not cover a group g_i, a robot located in the ith copy will not be able to distinguish it from the dummy copy G_0.

For the example in Figure 5.1, this translates to saying that a localization plan should visit at least one red vertex and at least one green vertex. For if the robot just visits red vertices, it can never know whether vertices 2 or 4 have a signature at them. Hence it cannot distinguish between G_2 and the dummy G_0.

Thus any localization plan P visits at least one leaf from each group. The subtree consisting of $l_1, l_2, \ldots, l_m$ will then be a Group Steiner tree of cost at most $c(P)$. The converse is also true : given a Group Steiner tree, we can get a plan of twice that cost by doing a depth-first search.

This gives us a factor 2 reduction between Group Steiner problem on trees and robot localization. Applying it to the gap instances of Halperin and Krauthgamer [21] leads to an $\Omega(\log^{2-\epsilon} n)$ lower bound.

5.2 Hardness of Group Steiner

A tree is said to be of *arity* d if each non-leaf vertex has exactly d children. A rooted tree has *height* H if all of its leaves are at distance H from the root. As usual, the *level of a vertex* is its distance from the root; the root itself is at level 0 and there are $H + 1$ levels.

Definition 3 *[4] A hierarchically well-separated tree (HST) is defined to be a rooted, weighted tree in which (i) all leaves are at the same distance from the root; and (ii) the weight of each edge is exactly $\frac{1}{\tau}$ times the weight of its parent edge, where $\tau \geq 1$ is any desired constant.*

To prove the lower bound, we use the recent result of Halperin *et al.* [21] which establishes $\Omega(\log^{2-\epsilon} n)$ hardness for Group Steiner problem on HSTs. The next theorem, extracted from their proof, states their result in a detailed form suited to our purpose:

Theorem 7 *[21] Let L be any NP-complete language. Then there exist constant c_0 and an algorithm $\mathcal{A}$ that, given an instance $\mathcal{I}$ and a sufficiently large constant α, produces in expected running time $O(|\mathcal{I}|^{polylog(|\mathcal{I}|)})$ an instance $\mathcal{I}' = (T, r, \mathcal{G})$ (r is also the root of T) of the Group Steiner problem such that:*

1. *For some $m \leq |\mathcal{I}|^{c_0}$, T is a HST with height $H = (\log m)^\alpha$, arity $d = m^{O(\log m)}$ and $\tau = m^{\log m}$. Further, each group $g \in \mathcal{G}$ is a subset of the leaves of T and there are $k = m^{O((\log m)^{\alpha+1})}$ groups.*
2. *If $\mathcal{I} \in L$, then there is a (rooted) tree $T' \subseteq T$ of weight $(\log m)^\alpha$ covering all the groups.*
3. *If $\mathcal{I} \notin L$, then every tree $T' \subseteq T$ covering all the groups has weight $\Omega((\log m)^{3\alpha+2})$.*

5.3 Reduction from trees

The next theorem describes the reduction to an instance of the localization problem:

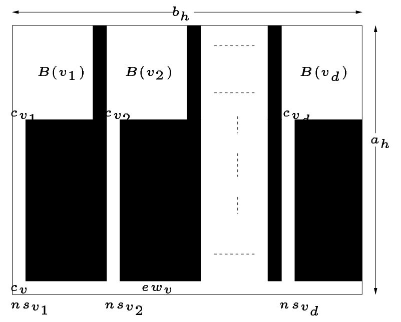
Theorem 8 *Let L be any NP-complete language. Then there exist constant c_0 and an algorithm $\mathcal{A}'$ that, given an instance $\mathcal{I}$ and a sufficiently large constant α, produces in expected running time $O(|\mathcal{I}|^{polylog(|\mathcal{I}|)})$ an instance $\mathcal{I}'' = (G, H)$ of the robot localization problem on grid graphs such that*

1. *For some $m \leq |\mathcal{I}|^{c_0}$, G has $N = m^{O((\log m)^{\alpha+1})}$ cells and H has $m^{O((\log m)^{\alpha+1})}$ hypotheses.*
2. *For some $\beta = m^{O((\log m)^{\alpha+1})}$:*
   (a) *If $\mathcal{I} \in L$, then there exists a localization plan with worst-case cost $O(\beta \cdot (\log m)^\alpha)$.*
   (b) *If $\mathcal{I} \notin L$, every localization plan has cost $\Omega(\beta \cdot (\log m)^{3\alpha+2})$.*

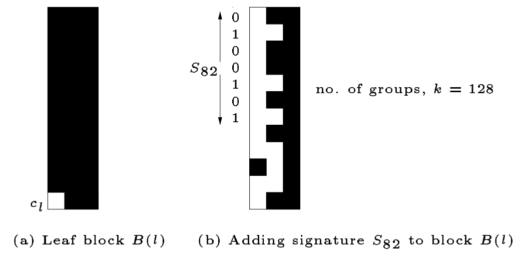
Proof. Let $\mathcal{I}' = (T(V, E), r, \mathcal{G})$ be the instance of Group Steiner on HSTs obtained by running algorithm $\mathcal{A}$ on $\mathcal{I}$ (see Theorem 7 above). Let d, H and τ denote the arity, height and weight factor of HST T, and k denote the number of groups in $\mathcal{G}$. G consists of $k + 1$ (disjoint) copies $B_0, B_1, \ldots, B_k$ of grid graph B, where B is an 'embedding' of HST T respecting the weights on its edges.

The embedding B is best described inductively. Let $B(v)$ denote the embedding of the subtree rooted at vertex $v \in T$. Cell c_v at the southwest corner of each $B(v)$ corresponds to vertex v. For a leaf l, $B(l)$ is a $3 \times (\lceil \log k \rceil + 5)$ rectangle with a single traversable cell c_l at its southwest corner (Figure 18(b)). The reason for adding blocked space to c_l will be clear later, when we use it to add "signatures" to leaf l. For a non-leaf vertex v, $B(v)$ is formed by combining the embeddings of the subtrees rooted at its d children $v_1, v_2, \ldots, v_d$ (see Figure 18(a)). $B(v_1), B(v_2), \ldots, B(v_d)$ are positioned along the top edge of $B(v)$ separated by north-south walls of width 1. There is an east-west corridor ew_v running along the bottom edge of $B(v)$. Cell c_{v_i} is connected to this corridor by a north-south corridor ns_{v_i} which corresponds to edge $vv_i \in T$. We make the length of ns_{v_i} proportional to the weight of vv_i: if v is at level h, $|ns_{v_i}| = \beta \cdot \frac{1}{\tau^h}$, where β is a *scaling factor* to be chosen later. Finally $B = B(r)$ where r is the root of T.

Let a_h, b_h be the length and breadth of the grid required to embed the subtree rooted at a level h vertex $v \in T$. To see that the tree "fits", observe that $B(v)$ fits in an $a_h \times b_h$ rectangle where $a_h = d \cdot a_{h+1} + (d - 1)$ and $b_h = b_{h+1} + \frac{\beta}{\tau^h}$. Hence $b_h = (\lceil \log k \rceil + 5) + \beta \cdot \sum_{\alpha=h}^{H-1} \frac{1}{\tau^\alpha}$, and by induction one can show that $a_h = 4 \cdot d^{H-h} - 1$.



(a)



(b)

Figure 18: (a) Block $B(v)$. (b) A leaf block with signature.

Let w_{xy} denote the weight of path connecting $x, y \in T$. Let P_{uv} be the unique path connecting cells c_u and c_v in B. We show that choosing $\beta = 5d^H \cdot \tau^H$ makes B an embedding of T in the following sense: for all vertices $x, y \in T$, $\beta \cdot w_{xy} \leq |P_{xy}| \leq 2\beta \cdot w_{xy}$. First observe that the length of any north-south corridor ns_v is now at least $5d^H$ while any east-west corridor is less than $4d^H$. Therefore $|ew_x| \leq |ns_v|$ for all $u, v \in T$. We charge the distances traveled along east-west corridors to the north-south corridors immediately preceding it. First assume x is the parent of y. Then P_{xy} consists of the north-south hallway ns_y along with portion of ew_x connecting c_x to ns_y. Clearly, $\beta \cdot w_{xy} = |ns_y| \leq |P_{xy}| \leq |ns_y| + |ew_x| \leq 2\beta \cdot w_{xy}$. Next consider the case when x, y are siblings with common parent z. P_{xy} consists of north-south corridors ns_x, ns_y along with the portion of ew_z connecting them. Hence, $\beta \cdot w_{xy} = \beta \cdot (w_{xz} + w_{zy}) = |ns_x| + |ns_y| \leq |P_{xy}| \leq |ns_x| + |ns_y| + |ew_z| \leq 2\beta \cdot w_{xz} + \beta \cdot w_{zy} \leq 2\beta \cdot w_{xy}$. For general x, y, let $c_{z_0=x}, c_{z_1}, \ldots, c_{z_m=y}$ be be the cells corresponding to vertices of T, in the order they occur along path P_{uv}. By the construction of B, we know that for each i either (i) z_{i+1} is a parent of z_i or vice-versa, or (ii) z_i, z_{i+1} are siblings. Therefore $\beta \cdot w_{z_i z_{i+1}} \leq |P_{z_i z_{i+1}}| \leq 2\beta \cdot w_{z_i z_{i+1}}$. Since $|P_{xy}| = \sum |P_{z_i z_{i+1}}|$, the length of P_{uv} is within factor 2 of $\beta \cdot \sum w_{z_i z_{i+1}} = \beta \cdot w_{xy}$.

Let $g_1, g_2, \ldots, g_k$ be the k groups in $\mathcal{G}$. We make $k+1$ copies $B_0, B_1, \ldots, B_k$ of embedding B. B_i's are the same except for distinguishing 'signatures' at some leaf blocks. $B_0 = B$ is the *dummy* copy and contains no signatures. For $i > 0$, B_i is formed by adding signature s_i (a binary encoding of i) to every leaf block $B(l)$ of B such that $l \in g_i$ (Figure 18(b)). To add s_i, first cell c_l is extended to a north-south corridor along the left edge of $B(l)$. Then a set of $\log k$ eastern "alcoves" encoding i in binary are placed along the eastern edge: the jth alcove from the top is blocked if and only if the jth bit in the binary form is 0. A robot located at c_l can read the value of i by going north and sensing the alcoves to its right for blockage.

Let $x = 2 \cdot a_0 \cdot b_0$. Grid graph G is a $(x + a_0) \times ((k+1) \cdot b_0 + k - 1)$ rectangle formed by connecting group blocks $\{B_i\}_i$ as shown in Figure 19. $B_0, B_1, B_2, \ldots B_k$ are placed along the right edge of G separated by east-west walls of width 1. A north-south corridor NS of width 1 runs alongside the left edge of G. The south-west cell of each block B_i is connected to this corridor by an east-west corridor EW_i of length x. The set of hypotheses H equals $\{h_0, h_1, \ldots, h_k\}$ where h_i denotes the cell at the south-west corner of block B_i. Substituting values of k, H, d, τ as given

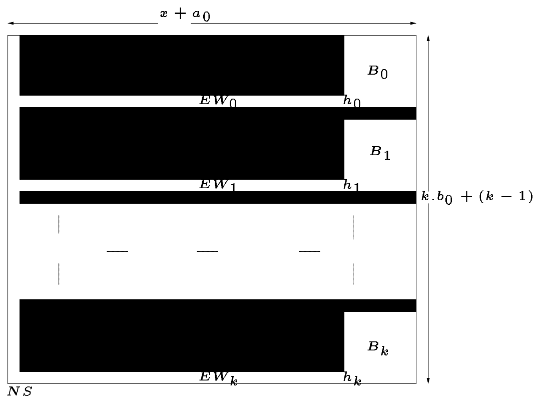


Figure 19: Grid graph G

by Theorem 7, we get $\beta = 5d^H \cdot \tau^H = m^{O((\log m)^{\alpha+1})}$, $|G| = O(a_0 b_0^2 k) = m^{O((\log m)^{\alpha+1})}$ and $|H| = k = m^{O((\log m)^{\alpha+1})}$, where $m \le |\mathcal{I}|^{c_0}$. We complete the proof by showing that the optimal localization plans for $\mathcal{I}'' = (G, H)$ in the 'yes' ($\mathcal{I} \in L$) and 'no' ($\mathcal{I} \notin L$) cases differ by a factor of $\Omega((\log m)^{2\alpha+2})$.

'Yes' case: Suppose $\mathcal{I} \in L$. By theorem 7, there exists a tree $T' \subseteq T$ of weight $(\log m)^\alpha$, which covers all groups in $\mathcal{G}$. As all groups $g \in G$ consist of leaves of T, without loss of generality every root to leaf path in T' ends at a leaf of T. Let $l_0, l_1, \ldots, l_{t-1}$ be the leaves of T' in the order they are visited by a depth-first search from the root. Consider the following plan: read the signatures at leaf blocks $B(l_0), B(l_1), \ldots, B(l_{t-1})$ in that order. As soon as a non-zero signature $s_{i_0}, i_0 > 0$ is read, localize to h_{i_0}. Otherwise, localize to h_0.

To prove correctness, assume the robot was placed (without its knowledge) at hypothesis h_{i_0}. If $i_0 = 0$, the robot will read zero signatures at all leaf blocks and correctly localize to h_0. Suppose $i_0 > 0$. Since T' covers all groups, group g_{i_0} contains at least one leaf vertex from T'. The robot will read signature s_{i_0} at the first such vertex in the sequence $l_0, l_1, \ldots, l_{t-1}$ and localize to h_{i_0}.

The total travel cost of the robot is $|P_{rl_0}| + \sum_{i=0}^{t-2} |P_{l_i l_{i+1}}| \le 2\beta \cdot (w_{rl_0} + \sum_{i=0}^{t-2} w_{l_i l_{i+1}}) \le 2\beta \cdot w(T') = O(\beta \cdot (\log m)^\alpha)$. We neglect the cost of reading signatures at l_i, as it is $O(t \cdot \log k) = O(d^H \log k) \le \beta$.

'No' case: Suppose $\mathcal{I} \notin L$. Assume that we have found a localization plan with cost $o(C \cdot (\log m)^{3\alpha+2})$. The number of movements for the plan is no larger than the length of an east-west hallway EW_i. Now assume the robot starts at cell h_0. Thus, it cannot visit a different east-west

hallway and, as part of the localization, must determine that no leaf block in its group block has a non-zero signature. Let $B(l_0), B(l_1), \ldots, B(l_{t-1})$ be all the leaf blocks, in the order they are visited by the robot. The collection of groups that these leaves cover must equal $\mathcal{G}$, for otherwise the robot could not distinguish between hypotheses h_0 and h_i for the groups g_i not covered by them.

Let T' be the Group Steiner tree formed by taking the union of paths connecting r to l_0 and l_i to l_{i+1} for $0 \leq i \leq t-2$. By Theorem 7, weight of T is $\Omega((\log m)^{3\alpha+2})$. Therefore, the cost of the localization plan is at least $|P_{rl_0}| + \sum_0^{t-2} |P_{l_i l_{i+1}}| \geq \beta \cdot (w_{rl_0} + \sum_{i=0}^{t-2} w_{l_i l_{i+1}}) \geq \beta \cdot w(T') = \Omega(\beta \cdot (\log m)^{3\alpha+2})$ ∎

Corollary 2 *For every fixed $\epsilon > 0$, the robot localization problem cannot be approximated within ratio $\log^{2-\epsilon} N$ on grid graphs of size N, unless $NP \subseteq ZTIME(n^{polylog(n)})$.*

Proof. Apply the algorithm in Theorem 8 with $\alpha = 2 \cdot (\frac{1}{\epsilon} - 1)$. The logarithm of the size of grid graph G is $\log N = O((\log m)^{\alpha+2})$, where $m \leq n^{c_0}$. The optimum localization plans in the 'yes' and 'no' case differ by a factor of $\Omega((\log m)^{2\alpha+2}) = \Omega((\log N)^{2-\epsilon})$ ∎

Corollary 3 *For every fixed $\epsilon > 0$, the robot localization problem cannot be approximated within ratio $\log^{2-\epsilon} N$ on polygons with N vertices, unless $NP \subseteq ZTIME(n^{polylog(n)})$.*

Proof. The grid graph G in Theorem 8 above can be viewed as a polygon P with at most N vertices. Let h_i' denote the center of the cell h_i in G. Consider the localization problem on P with hypothesis set $H' = \{h_0', h_1', \ldots, h_k'\}$. The optimal localization plan in the 'yes' case has cost $O(\beta \cdot (\log m)^{\alpha})$, as a robot with a range finder can only do better. However when $\mathcal{I} \notin L$, a robot with a range finder may read the signatures from a distance, and localize at lesser cost. To rule this out, put small 'twists' in polygon P just before every signature. Thus the robot cannot read the signatures at a distance, and therefore will travel at least $\Omega(\beta \cdot (\log m)^{3\alpha+2})$ distance as in Theorem 7 above. The 'yes' and 'no' cases differ by $\Omega((\log m)^{\alpha+2})$ and the bound follows by choosing $\alpha = 2 \cdot (1 - \frac{1}{\epsilon})$ ∎

5.4 Reduction from grid graphs

We note that any lower bound for Group Steiner on grid graphs can be extended to a similar lower bound for localization on grid graphs. The main idea is the same as above: take a hard instance

$(G, r, \mathcal{G})$ of Group Steiner on grid graphs. Suppose G is an $m \times n$ grid graph, and there are $k = |\mathcal{G}|$ groups. Make a map G' that consists of k disjoint copies $G_1, G_2, \ldots, G_k$ of G. Each copy G_i is a scaled up (by a factor of β) version of G. Thus, each cell of G corresponds to a $\beta \times \beta$ block in G_i. For each cell in group $g_i \in \mathcal{G}$, put a $3 \times \lceil \log k \rceil$ "signature" in the upper left corner of the corresponding block of G_i. As before, choose the scaling factor large enough so that the distance between signatures is much larger than their size. A good choice is $\beta = k$. Initially, the robot is placed at the center of block corresponding to r in one of the G_i's.

In order to localize, the robot has to find the index of its component and, hence, must visit a set of blocks that covers all of the groups. This path can be translated to a Group Steiner tree of proportional cost (divided by β) in G (since β is much larger than $\log k$). Conversely, we can convert any Group Steiner tree in G into a path by doing depth-first search and then using that path in the scaled grid G' as a localization plan. It is easy to see that this extends the same hardness factor to localization on grids.

Thus, it seems that further improvement (in either the lower bound or upper bound) in the approximation factor of our algorithm can come only after progress on the Group Steiner problem on grid graphs.

Note. Fig. 20 describes a version of the above reduction which is similar to that for trees, since it gives a reduction from the Group Steiner problem on the dual graph $D(G)$ of G.

Let G^β be a copy of G scaled by β. We will embed a stretched version (by a factor of β) of the dual graph $D(G)$ in G^β.

For each vertex $v \in D(G)$, take the center cell c_v of the corresponding $\beta \times \beta$ block in G^β. Connect these cells using unit length north-south and east-west corridors to represent the edges in $D(G)$. We assume that all other cells in G^β are blocked. Thus G^β represents a β-stretched embedding of $D(G)$.

We now make k disjoint copies $G_1, G_2, \ldots, G_k$ of G^β. For each vertex v in group $g_i \in \mathcal{G}$, we add a $3 \times \lceil \log k \rceil$ signature at the cell 2 units up the northern corridor at cell c_v (see Fig. 20). (If there is no northern corridor at c_v, we connect the signature to c_v using a north-south corridor of length 2).

We choose the scaling factor so that it allows us to fit a signature at every vertex cell c_v. A good choice is $\beta = 2\log k$. Initially, the robot is placed at the cell c_r corresponding to r in one of the G_i's.

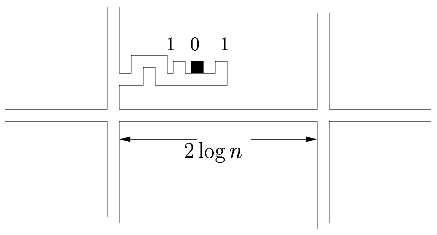


Figure 20: Inapproximability for grid graphs

In order to localize, the robot has to find the index of its component and, hence, must visit a set of vertex cells c_v that cover all of the groups. This path in G^β can be translated to a Group Steiner tree of proportional cost (divided by β) in $D(G)$. Conversely, we can convert any Group Steiner tree in $D(G)$ into a path by doing depth-first search and then using that path in the scaled grid G^β as a localization plan. This extends the same hardness factor to localization on grids.

CHAPTER VI

LOCALIZATION ON POLYGONS

We will now adapt our algorithm to polygonal environments. In this chapter, we will assume that the map polygon has no holes.

As discussed in chapter 1, using visibility cell decompositions one can decompose the plane into polygonal cells such that within each cell the majority observation is the same. The groups can then be formed by including cells $\mathcal{C}$ into groups g_i such that hypotheses $h_i \notin Maj(\mathcal{C})$ does not belong to the majority opinion at $\mathcal{C}$.

Later, we will introduce new ideas which bypass cell decompositions and instead directly compute the group boundaries K_i. The crucial idea is that they will be polygons inside the majority-rule map containing the origin. To half-localize, the robot will need to visit (or cross) the boundaries of at least half the K_i's.

6.1 Introduction

In chapter 4, we have described our algorithm with respect to the grid graph model. We now show how we can adapt it to the polygonal model.

We focus here on the case of simple polygons; in Section 7.1 we discuss the extension to the case of polygons with holes. The outline of the algorithm is the same: the robot works in phases; in each phase reducing the set of hypotheses by half. However, since the robot moves continuously, local coordinates γ lie in the Euclidean plane $\mathbb{R}^2$ (for grid graphs, they were points on the integral lattice). As before, let opinion $\mathcal{O}(h, \gamma)$ denote the observation i.e., the visibility polygon observed by a robot at position $h + \gamma \in P$. If the point $h + \gamma$ lies outside P, we take $\mathcal{O}(h, \gamma) = \emptyset$. For a coordinate $\gamma \in \mathbb{R}^2$, the hypothesis partition $\mathcal{H}(\gamma)$ partitions hypotheses in H according to their opinions $\mathcal{O}(h, \gamma)$. The majority-rule map denotes the subset of coordinates that lie inside P for the majority of hypotheses. In Section 6.2, we will show that the majority-rule map is a polygon with holes of total size $O(k^2n^2)$ and that this bound is tight in the worst case. We let P_{maj} denote the connected component of the majority-rule map that contains the origin γ_0, and often refer to P_{maj}

simply as the majority-rule map, since P_{maj} is the component of interest to us.

In the polygonal model, a *halving path* $\mathcal{C}$ is a curve in the majority-rule map with one endpoint at the origin γ_0 such that $|\bigcap_{x \in \mathcal{C}} Maj(x)| \leq \frac{1}{2}|H|$. (Parameter x varies over the continuum of coordinates along the path $\mathcal{C}$.) It is straightforward to extend Lemmas 9 and 10 to the case of polygons with this new definition. A shortest path $Path(\gamma, \gamma')$ between any two coordinates $\gamma, \gamma' \in P_{maj}$ is piecewise linear with bend points at vertices (this includes the vertices of holes) of P_{maj}. Hence, we can specify a halving path by a sequence $(\gamma_0, \gamma_1, \ldots, \gamma_m)$ where $|\bigcap_{i=0}^{m} Maj(\gamma_i)| \leq \frac{1}{2}|H|$, and a shortest path $Path(\gamma_i, \gamma_{i+1})$ is used to go from γ_i to γ_{i+1}.

Since P_{maj} consists of an infinite number of points, one cannot compute an approximation to the optimal halving path $\mathcal{C}_H^*$ by reducing it to a $\frac{1}{2}$-Group Steiner problem on a finite number of coordinates, as in Section 4.5 for the case of grids. Instead, we discretize the problem to a finite, polynomial-size set of coordinates $Q_H \subset P_{maj}$ such that there exists a halving path $\mathcal{C} = (\gamma_0, \gamma_1, \ldots, \gamma_m)$ such that $\gamma_i \in Q_H$, and the length of $\mathcal{C}$ is at most 2 times the length of an optimal halving path. To do so, we first calculate the boundaries of groups g_i (i.e., coordinates γ such that $h_i \notin Maj(\gamma)$), which are polygons $K_i \subset P_{maj}$ with holes (see section 6.3). Hence the robot just needs to visit the boundary of at least half of the K_i's. In Section 6.4, we describe a way to select a special set of discrete points on the boundary of the K_i's so that a halving path of length at most 2 times that of optimal passes through these points. Next, we construct the instance $\mathcal{I}_{P,H}$ of the $\frac{1}{2}$-Group Steiner problem on the finite set of coordinates Q_H, as we did for the case of grid graphs in Section 6.5. Finally, in Section 6.6 we combine all of the ingredients above to get an $O(\log^2 n \log k)$-approximation algorithm for the polygonal model.

6.2 Computing the majority-rule map

The boundary of the majority-rule map can be constructed as follows. Let P_i denote a translation-congruent copy of the map polygon with hypothesis h_i at the origin γ_0. Clearly, coordinate γ is traversable for hypothesis h_i if and only if it lies inside polygon P_i. The overlay of all of these polygons, $Overlay(P_1, P_2, \ldots, P_k)$, partitions the plane into polygonal regions, known as *cells*. Each cell C either lies completely inside copy P_i or lies completely outside it. The majority-rule map is formed by the union of all cells C that lie inside P for the majority of hypotheses (equivalently, for the majority of P_i's). By this construction, the majority-rule map is a union of polygons (possibly with holes). For computing a half-localization strategy, the robot needs to plan only over the connected component containing the origin. The next lemma gives a tight bound on its worst-case complexity:

Lemma 12 *Let $A_1, A_2, \ldots, A_k$ be k polygons (possibly with holes), each containing the origin and each with $O(n)$ vertices. Then, the face, A_{maj}, containing the origin in the majority-rule map they define has $O(k^2n^2)$ vertices and can be constructed in time $O(k^2n^2)$. Furthermore, the upper bound of $O(k^2n^2)$ on the number of vertices is tight, even if the A_i's are translates of the same simple polygon.*

Proof. The upper bound is immediate, since the set of $O(kn)$ line segments that constitute the edges of the k polygons define an arrangement having at most $O(k^2n^2)$ vertices in total. The lower bound is illustrated in Figure 21. The time to construct A_{maj} follows from the fact that an arrangement of m segments in the plane can be constructed in time $O(m^2)$, and, within this same time bound, the faces of the majority-rule map can be identified, after which the face containing the origin can be constructed by breadth-first search in the dual graph of the arrangement. In fact, using the algorithm of Balaban [3], the arrangement can be constructed in *output-sensitive time*. ■

The above lemma also bounds the complexity of P_{maj} and shows that it is tight, since it arises from the majority-rule map associated with translates of P.

6.3 Computing the group boundaries

6.3.1 Introduction

To compute the groups, we instead construct their complements $\bar{g_i}$, where $\bar{g_i}$ is the maximal region containing the origin such that a robot at its boundary will distinguish h_i from at least half the hypotheses.

It is instructive to first consider the case when one has just two hypotheses h_1 and h_2. Group $\bar{g_1}$ will then be the region at whose boundary the robot can distinguish h_1 from h_2.

What if there are k hypotheses instead of 2 ? It turns out that one can use the construction for two hypotheses as a building block for computing the groups $\bar{g_i}$ in the general case.

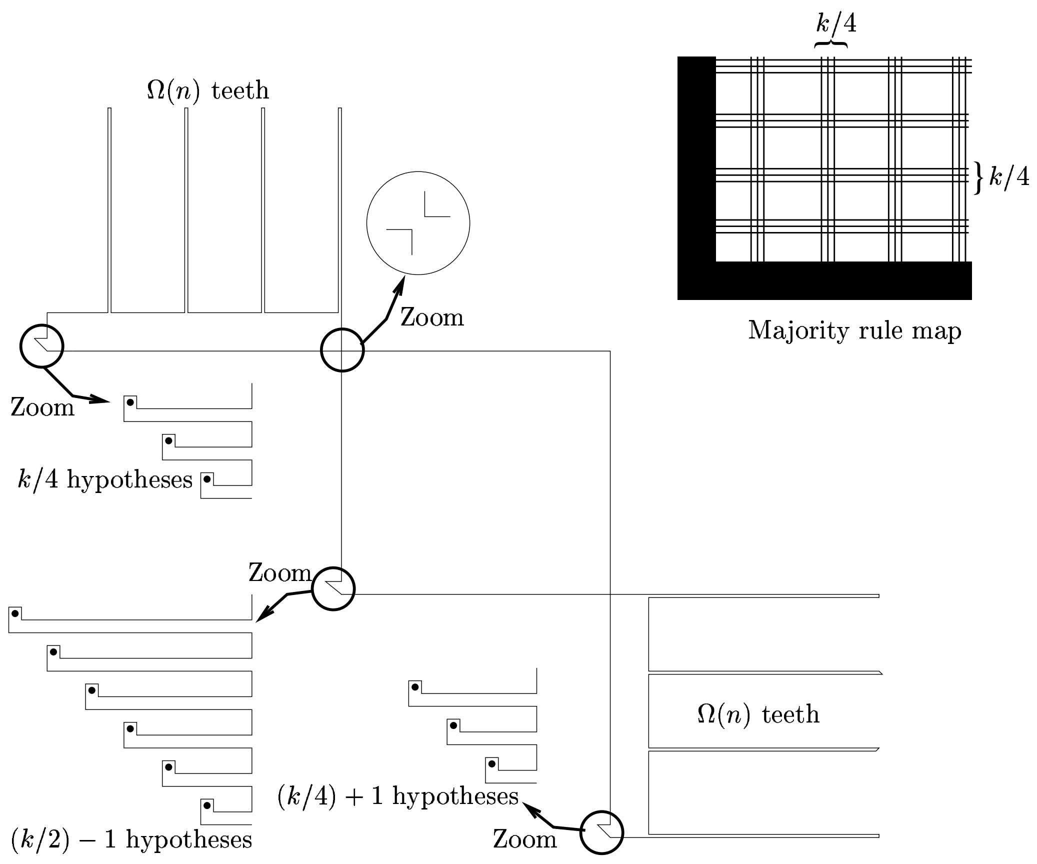


Figure 21: An example with complexity $\Omega(k^2n^2)$ of the majority-rule map obtained by overlaying translates of copies of a simple polygon. The solid dots denote the set of hypotheses.

This can be seen as follows. Let us define polygon K_{ij} to be the maximal connected region containing the origin such that a robot will distinguish h_i from h_j by visiting its boundary. There are $k(k-1)$ such polygons, one for each pair $\{h_i, h_j\}$ where $j \neq i$. One can construct them in the same way as we construct K_{12} for two hypotheses h_1 and h_2.

Now let us construct the group $\bar{g}_i$. Take the $k-1$ polygons K_{ij} where $j \in \{1, 2, \ldots, i-1, i+1, \ldots, k\}$. A robot will distinguish h_i from at least $\frac{k}{2}$ hypotheses if it crosses the boundary of at least half the K_{ij}'s. Now shoot a ray in any given direction. The line segment inside $\bar{g}_i$ will extend from the origin to the $\frac{k}{2}$th intersection point of the ray with K_{ij}'s. By rotating the ray around the origin, group-boundary $\bar{g}_i$ can be computed as the majority-rule map of polygons $\{K_{ij}\}_{j\neq i}$.

Following the definition in Section 4.5, the group g_i is defined to be the set of all coordinates $\gamma \in P_{maj}$ such that h_i does not have the majority opinion at γ, i.e., $h_i \notin Maj(\gamma)$. The *complement* $\bar{g}_i$ of g_i is the set of points $\bar{g}_i = P_{maj} \backslash g_i$ not in g_i.

6.3.2 Case $k = 2$

Consider a hypothesis h_j ($h_j \neq h_i$), and let F_{ij} denote the face in $Overlay(P_i, P_j)$ that contains the origin, γ_0 (see Figure 22). First, we note that:

Lemma 13 *The face F_{ij} has at most $2n$ edges.*

Proof. Consider an edge e of F_{ij}. If e is a subsegment of both P_i and P_j, then one of its endpoints must be a vertex v of P_i or P_j, and we can "charge" e to that vertex. If e is a subsegment of P_i but not of P_j, then it forms a chord of P_j and can be charged off to the vertex of P_j it occludes. Since each vertex is charged at most once, F_{ij} has at most $2n$ edges. ■

Each of the $O(n)$ edges $e \subset \partial F_{ij}$ is of one of three types: (i) e lies on the boundary of P_i, but not of P_j; (ii). e lies on the boundary of P_j, but not of P_i; or, (iii). e lies on the boundary of both P_i and P_j. A robot can distinguish between h_i and h_j if and only if the robot sees an edge e of type (i) or (ii).

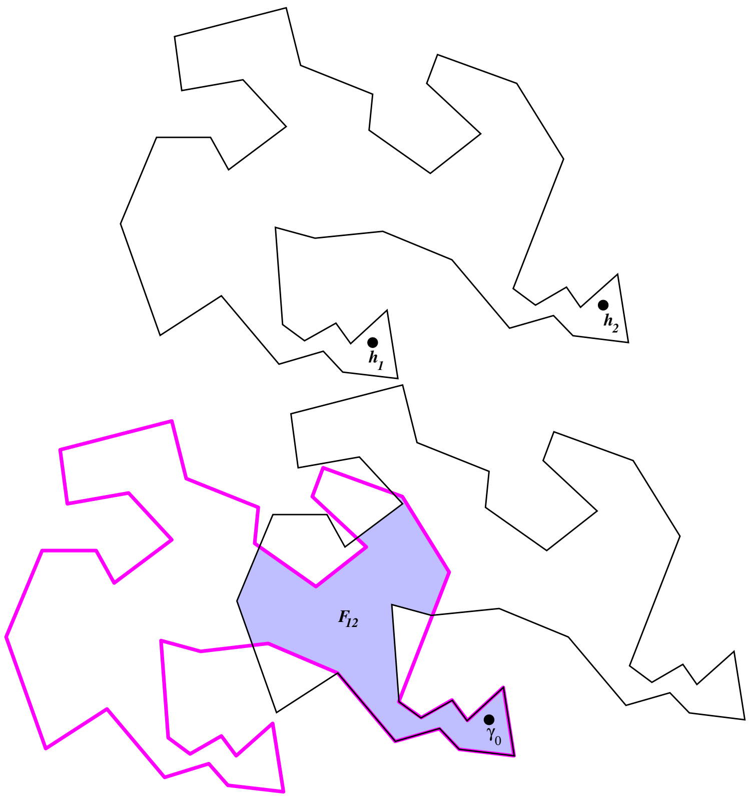


Figure 22: Top: A polygon P with two hypotheses h_1 and h_2. Bottom: The overlap of P_1 and P_2, with the face F_{12} containing γ_0 highlighted.

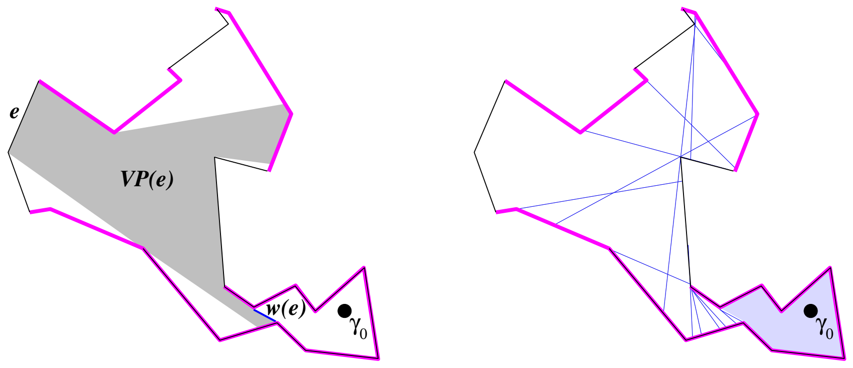


Figure 23: Left: The visibility polygon, $VP(e)$, is shown for an edge e of type (i), and the corresponding chord (window) $w(e)$ is shown highlighted. Right: The arrangement of all windows $w(e)$ for edges of type (i) or (ii) is shown, and the face, G_{12}, containing γ_0 is shaded.

If γ_0 sees any edge of type (i) or type (ii), then the robot can distinguish between h_i and h_j without moving from the origin γ_0. Thus, assume that all edges of F_{ij} that are visible from γ_0 are of type (iii). Let e be an edge of F_{ij} of type (i) or of type (ii). The set $VP(e)$ of points of F_{ij} that are visible to some point of e is a simple polygon (the *visibility polygon* of e) within F_{ij}, which we know, by assumption, does not include point γ_0. There is a chord of F_{ij}, $w(e)$, that lies on the boundary of $VP(e)$, separating e from γ_0. The line segment $w(e)$ is often called a *window* (see Figure 23).

Consider now the arrangement of the $O(n)$ boundary edges of F_{ij} together with the set of all $O(n)$ windows $w(e)$ for edges e of type (i) or (ii). Let G_{ij} denote the face in this arrangement that contains the origin γ_0. Since G_{ij} is a face in an arrangement of chords of a simple polygon, it is a simple polygon having linear ($O(n)$) complexity. (No chord can contribute more than once to the face.) Note too that $G_{ij} \subset F_{ij} \subset P_i$ and that by maximality the elements of the boundary of G_{ij} are of two types: (a) polygonal chains of type (iii) edges, which is on the common boundary, $\partial P_i \cap \partial P_j$, of P_i and P_j, and (b) *window-chains* consisting of convex polygonal chains composed of subsegments of windows. A window-chain of G_{ij} separates γ_0 from one or more edges of F_{ij} of type (i) or type (ii). The next lemma follows from the definition of G_{ij}:

Lemma 14 *A robot can distinguish between hypothesis h_i and hypothesis h_j if and only if it visits a window-chain on the boundary, ∂G_{ij}, of G_{ij}.*

Proof. Each window $w(e)$ cuts off the part of polygon $VP(e)$ from which e of type (i) or (ii) is visible. In other words, as soon as the robot crosses $w(e)$, it can use its sensors to check whether e exists or not, and hence will be able to distinguish h_i from h_j. Since G_{ij} is what remains after all visibility polygons $VP(e)$ of edges of type (i) or (ii) have been chopped off, it satisfies the lemma (see Figure 23). ■

In other words, G_{ij} is the connected component of coordinates including the origin γ_0 for which $\mathcal{O}(h_i, \gamma) = \mathcal{O}(h_j, \gamma)$ i.e., the opinions of h_i and h_j are the same. Next we use the G_{ij}'s to construct the complement, $\bar{g_i}$, of group g_i.

If the robot had tactile sensors, F_{12} will form our group $\bar{g_1}$. However, laser sensors allow it to see an edge of type (i) before actually reaching it and hence the group will be a smaller subset of F_{12}.

6.3.3 Case $k > 2$

Let K_i be the face containing γ_0 in the majority-rule map of the $k - 1$ polygons G_{ij}, for $j \neq i$. Thus, the boundary of K_i consists of polygonal chains on the boundary of P_i and polygonal chains comprising of segments and subsegments of the window-chains that appear on the boundaries of the polygons G_{ij}. We refer to $\partial K_1 \setminus \partial P_i$ as the *window-boundary* of K_i.

It is clear that $K_i \subseteq P_{maj}$, since each point of K_i lies within a majority of the polygons G_{ij}, and therefore of the polygons P_j.

Lemma 15 *K_i is a connected component of the set $\bar{g_i}$. A robot initially located at hypothesis h_i will half-localize if and only if it travels to the window-boundary of K_i.*

Proof. We first show that $K_i \subset P_{maj}$. Let I denote the set of $k - 1$ indices $[1 \ldots n] \backslash i$. Consider any coordinate $\gamma \in K_i$. Let $I' \subset I$ denote the set of indices j such that $\gamma \in G_{ij}$. Any coordinate inside G_{ij} clearly belongs to both polygons P_i and P_j. Hence, γ is inside polygon P for at least $|I'| + 1 \geq \lceil \frac{k-1}{2} \rceil + 1 \geq \lceil \frac{k}{2} \rceil$ hypotheses. Thus, $K_i \subset P_{maj}$. Further, the opinions $\mathcal{O}(h_i, \gamma) = \mathcal{O}(h_j, \gamma)$ are the same for any $j \in I'$. Thus, the majority opinion at γ is the same as $\mathcal{O}(h_i, \gamma)$ and hence $h_i \in Maj(\gamma)$.

For the second statement, note that if the robot crosses the boundary of K_i, it will lie outside at least half of the $k - 1$ sets $G_{i1}, G_{i2}, \ldots, G_{ik}$ and hence by making an observation will be able to distinguish h_i from at least $\lceil \frac{k-1}{2} \rceil$ hypotheses. In the worst case (if the robot is initially at h_i), we will be left with at most $k - \lceil \frac{k-1}{2} \rceil \leq \lceil \frac{k}{2} \rceil$ hypotheses, and hence the robot will half-localize. (Note that the set of hypotheses remaining can be one more than that required for half-localization; however, the number of iterations remains $O(\log k)$, and hence the approximation factor is unchanged.)

For the converse, let P be a half-localization path which does not cross the window-boundary of K_i. Then a robot following P will remain inside at least half of the $k-1$ sets $G_{i1}, G_{i2}, \ldots, G_{ik}$. Let S be the set of indices $j \in [1..n] \backslash \{i\}$ such that P does not cross the window-boundary of G_{ij}. We have that $|S| \geq \lceil \frac{k-1}{2} \rceil$. By lemma 14, a robot initially located at h_i will be unable to distinguish h_i from any hypothesis in S. This will leave the robot with a set of at least $|S|+1 \geq \lceil \frac{k}{2} \rceil$ hypotheses if it reaches the end of path P. This leads to a contradiction, as P is a half-localization path ■

Lemma 16 *K_i has $O(nk^{4/3}\alpha(n)\log^{2/3} k)$ edges, where $\alpha(\cdot)$ denotes the inverse Ackermann function.*

Proof. First note that the boundary of K_i that is not part of the window-boundary has complexity $O(n)$, since it is boundary shared with P_i. Thus, it suffices to bound the complexity of the window-boundary of K_i.

Each of the $O(nk)$ edges of the window-chains of the regions G_{ij} can be mapped to a (finite length) curve in a "polar geodesic" coordinate system defined by the family of all shortest (geodesic) paths within P_i from γ_0 to points $t \in \partial P_i$ on the boundary. Then, we appeal to the fact that the k-level in an arrangement of a set of m pseudo-segments has complexity $O(mk^{1/3}\alpha(m/k)\log^{2/3} k)$ [9]. Since we have $m = O(nk)$, the total complexity of K_i is $O(nk^{4/3}\alpha(n)\log^{2/3} k)$.

(We suspect that the true complexity of K_i is $O(nk^{4/3}\alpha(n))$, which is the complexity of the k-level in an arrangement of (straight) line segments.) ■

Each region $K_i = \bar{g}_i$ shares one or more polygonal chains on its window-boundary with the boundary of set g_i. In order to half-localize, the robot needs to visit at least half of the groups g_i. Thus, the robot needs to visit at least half of the window-boundaries of the K_i's (i.e., at least half of the sets $\partial K_i \cap \partial g_i$), each of which consists of $O(nk^{4/3}\alpha(n)\log^{2/3} k)$ edges that lie within the majority map P_{maj}. Our goal is to find a path within P_{maj} that visits at least half of the sets $\partial K_i \cap \partial g_i$.

This makes us compute the majority-rule map M and the k group boundaries $\bar{g}_i$. A half-localization plan starts from the origin and visits $\frac{k}{2}$ group boundaries inside M. However, here we are faced with another problem. The number of points inside the region M is infinite, whereas we need a finite set of coordinates for the Group Steiner problem.

6.4 The Set of Coordinates Q_H

6.4.1 Basic Geometric Ideas

In order to solve our half-localization problem, we define a discrete set Q_H of points on the edges of $\partial K_i \cap \partial g_i$, and then solve an instance of the $\frac{1}{2}$-Group Steiner problem on the corresponding point set.

To achieve this, we discretize the problem and choose a set Q_H of representative points. As expected, these special points will lie along the edges of the group boundaries. The new halving paths will be obtained by shifting the first point where the robot visits a group boundary to the nearest special point on the same crossing edge.

The crucial idea is this : at distance r from the origin, it is enough to have a grid resolution of $\frac{r}{k}$. This is because we will shift the halving path at most k times once for each hypothesis and the total change will then be at most $(r/k) \cdot k = O(r)$, which is within a constant factor of the total path length..

The set of points is picked as follows. First find the minimum distance r^* such that at least half the group boundaries are within distance r^*. Any halving path will have cost at least r^*. Further, it will be at most $2k \cdot r^*$ as a robot can visit each group boundary at cost $2r^*$ and return to the origin.

This gives us a lower and upper bound on the size of the halving path. To complete the construction, we break this interval into powers of 2 and for each power $2^i r^*$ we construct a square grid of that size with $k \times k$ cells. Thus we have $\log k$ such grids.

For each edge on a group boundary, we add all points at which it intersects the $\log k$ grids into the special point set Q_H. This gives us a total of $O(k \log k)$ points per edge. To see why this works, note that if the robot visits group boundary $\bar{g}_i$ at distance r_i one can take the grid within a factor 2 of this distance and shift to a special point at cost $\leq 2 \cdot (r_i/k)$.

6.4.2 Discretizing the polygon

We now describe the construction of the discrete set Q_H that we use for our approximation. Consider an optimal halving path $\mathcal{C}_H^* \subset P_{maj}$, which visits at least $\lceil \frac{k}{2} \rceil$ of the sets $\partial K_i \cap \partial g_i$.

Let r^* be the (geodesic) radius of the smallest *geodesic disk*, centered on γ_0, that contains $\mathcal{C}_H^*$. Here, "geodesic" refers to shortest path distance within the majority-rule map P_{maj} (A geodesic will be a piecewise linear curve). Let r_{min} be the (geodesic) radius of the smallest geodesic disk, centered on γ_0, that intersects at least $\lceil \frac{k}{2} \rceil$ of the boundaries ∂g_i. Clearly, $r^* \geq r_{min}$. Further, we know that the length of $\mathcal{C}_H^*$ is at most $k \cdot r_{min}$, since one possible halving path stays within the geodesic disk, D_0, of radius r_{min} centered at γ_0, and travels at most distance $2r_{min}$ between any two consecutive groups visited by the path (just go via γ_0, using geodesic paths to get to and from γ_0). Note too that it is easy to compute r_{min} by computing the *shortest path map* with respect to source γ_0 within P_{maj}; see [34].

Consider the sequence of radii, $r_{min}, 2r_{min}, 4r_{min}, \ldots, 2^{\lceil \log_2 k \rceil} r_{min}$. Note that $r^* \in [2^{i'} r_{min}, 2^{i'+1} r_{min}]$ for some choice of i' among the $O(\log k)$ possibilities in the sequence. For each choice of i', we consider the axis-aligned square (this square is not with respect to geodesic distance), centered at γ_0, of side length $2 \cdot 2^{i'} r_{min}$, and decompose the square into a k-by-k grid of subsquares using $k-1$ evenly spaced horizontal/vertical lines. For each segment σ that is an edge of some $\partial K_i \cap \partial g_i$, we mark on σ the crossing points (if any) where σ crosses a grid line (i.e., where σ crosses between subsquares). This results in at most $2k-2$ marked points along σ, for each choice of i', so $O(k \log k)$ marked points in total along σ.

We let Q_H be the union of the set of all marked points for all edges on the boundaries $\partial K_i \cap \partial g_i$, together with the endpoints of these edges. Since there are k sets $\partial K_i \cap \partial g_i$, each with $O(nk^{4/3}\alpha(n) \log^{2/3} k)$ edges/vertices, and we place $O(k \log k)$ marked points per edge, this yields a total of $O(nk^{10/3}\alpha(n) \log^{5/3} k)$ points in Q_H. (Note that this bound is nearly linear in n, and one may expect that, in practice, $k << n$.)

Lemma 17 *Suppose that an optimal halving path $\mathcal{C}_H^*$ visits $\partial g_1, \partial g_2, \ldots, \partial g_m$, with $m = \lceil k/2 \rceil$, and let edge $e_i \subset \partial g_i$ be the first edge of g_i visited along $\mathcal{C}_H^*$ (after leaving γ_0). Then there exists a* *piecewise-linear halving path $\mathcal{C} = (\gamma_0, \gamma_1, \ldots, \gamma_m)$ of length at most $2 \cdot |\mathcal{C}^*_H|$) such that $\gamma_i \in Q_H$, and the shortest (geodesic) path in P_{maj} is used to go from γ_i to γ_{i+1}.*

Proof. Let $p_i \in e_i$ be the first point where $\mathcal{C}^*_H$ visits ∂g_i. Let r^* be the geodesic radius of the smallest geodesic disk D_0 (within P_{maj}) centered at γ_0 that contains $\mathcal{C}^*_H$; let i' be such that $r^* \in [2^{i'} r_{min}, 2^{i'+1} r_{min}]$. Then, we know that each segment e_i intersects D_0 and therefore also intersects the axis-aligned square of side length $2 \cdot 2^{i'+1} r_{min}$ centered at γ_0. Thus, within distance $(1/k)2^{i'+1} r_{min}$ of p_i along the line segment containing e_i there is a marked point γ_i of Q_H associated with the corresponding grid partition into subsquares; in case the endpoint of e_i is encountered along the segment before the marked point, we redefine γ_i to be this endpoint. We can modify the path to go through each γ_i (this is possible, by sliding the endpoint continuously along the edge to the coordinate in Q_H), adding distance at most $(1/k)r^*$ per i. In total, the cost of these detours is at most $k \cdot (1/k) r^* = r^*$, thus proving the claim. ■

6.5 Reduction to $\frac{1}{2}$-Group Steiner

We formulate now the instance of the $\frac{1}{2}$-Group Steiner problem that we need to solve for half-localization:

INSTANCE $\mathcal{I}_{P,H}$: Take G as the complete graph on Q_H. Define the cost of an edge (γ, γ') to be the length of a shortest path joining γ, γ' in the majority-rule map P_{maj}. Take the root as the origin γ_0. Make k groups of points of Q_H corresponding to the sets $g_1, g_2, \ldots, g_k$.

As in section 4.5, a tree T covers k' groups if and only if $\bigcap_{\gamma \in T} Maj(\gamma)$ has size $k - k'$. In particular, T covers at least half the groups if and only if $|\bigcap_{\gamma \in T} Maj(\gamma)| \leq \frac{1}{2}|H|$. Also every halving path gives a $\frac{1}{2}$-Group Steiner tree of the same cost. Lemma 11 extends to this case, given that a halving path of cost within twice of the optimal passes through points in Q_H (by lemma 17).

6.6 Putting everything together

The overall strategy for polygons is as follows (see Algorithm 3). Theorem 9 bounds the approximation factor and computation time of strategy **RHL**.

```
Data    : Map polygon $P$, the set of hypotheses $H$
Result  : The robot localizes to its initial position $h \in H$
while $|H| > 1$ do
    begin
        Compute the polygons $G_{ij}$ for each pair of hypotheses, $h_i$ and $h_j$
        Compute the polygons $g_i$
        Compute the majority-rule map $P_{maj}$
        Compute the set of coordinates $Q_H$
        Make instance $\mathcal{I}_{P,H}$ of $\frac{1}{2}$-Group Steiner problem
        Solve $\mathcal{I}_{P,H}$ to compute a halving path $\mathcal{C} \subset P_{maj}$ (lemma 11)
        Half-localize by tracing $\mathcal{C}$ and making observations at coordinates in $Q_H$
        Move back to the starting location
    end
end
```

Algorithm 3: Strategy **RHL** for polygons

Theorem 9 *A robot guided by strategy* **RHL** *(Algorithm 3) correctly determines its initial position $h \in H$ by traveling at most distance $O(\log^2 n \log k) \cdot OPT(P, H)$, where $k = |H|$ and n is the number of vertices in polygon P. Further, the computation time of the robot is polynomial in n and k.*

Proof. By lemma 17, an optimal halving path on the coordinates Q_H is of length $O(OPT(P, H))$. Since the number of vertices in Q_H is polynomial (bounded by $O(nk^{10/3}\alpha(n) \log^{5/3} k)$), an $O(\log^2 n)$-approximate halving path can be computed in polynomial time by using algorithm $\mathcal{A}$ (see Theorem 3). Since there are $\log |H|$ phases, this gives an $O(\log^2 n \log |H|)$-factor strategy. ■

6.7 *Improving the number of reference points*

We will now describe how to adapt the above construction to decrease the number of reference points to $O(k^2 \log k)$ from $O(nk^{10/3}\alpha(n) \log^{5/3} k)$ (see lemma 16).

Steps 1 (majority-rule map) and 3 (discretizing the boundaries of regions ∂K_i) will remain the same. A *convex chain* is a piecewise linear function with the slopes of linear segments decreasing from left to right. We will modify the definition of K_i so that their boundaries are convex chains with at most r edges. Here r is the number of reflex vertices in the map polygon P.

Since a convex chain can intersect a $k \times k$ grid in at most $3k$ points, we will get $3k \log k$ reference points for each boundary ∂K_i. Since there are k such boundaries, we have a total of $O(k^2 \log k)$

reference points. Note that convexity is crucial to reducing the number of reference points for each chain.

In the following, P_{maj} denotes the majority-rule map and $V_{maj}(\gamma)$ denotes the visibility polygon observed by a robot located at coordinate γ in P_{maj}. We distinguish it from $V(p)$ which is the visibility polygon for a robot located at point p in the original map polygon.

We first make the following observation :

At any coordinate $\gamma \in P_{maj}$, the majority observation is the same as the visibility polygon $V_{maj}(\gamma)$.

This is true because if a robot at coordinate γ sees a visibility polygon V' different from $V_{maj}(\gamma)$ it immediately reduces the set of hypotheses by half, since it has found a deviation from the majority-rule map.

Fig 24 shows this for grid graphs. The majority-rule map is shown on the left with three coordinates c_1, c_2 and c_3. The observations consistent with the majority-rule map o_1, o_2, o_3 are shown on the right. If a robot at c_i makes an observation different from o_i it will either find that a cell inside G_{maj} is blocked or that a cell outside G_{maj} is traversable. In both cases, it will half-localize as a cell is inside G_{maj} based on whether it is blocked or traversable for more than half the hypotheses.

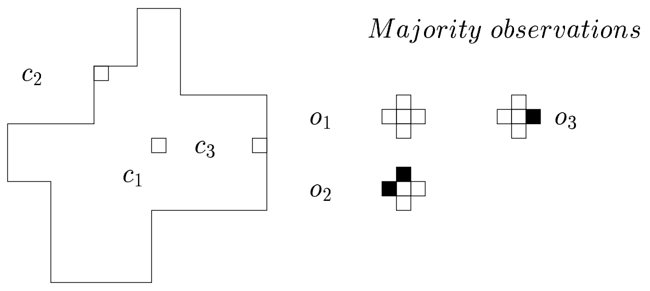


Figure 24: Majority observations inside the majority-rule map

We will change the definitions of K_i's accordingly :

Definition 4 *K_i is the maximal connected region in P_{maj} such that for each $\gamma \in K_i$ the visibility polygon $V(h_i + \gamma)$ seen by a robot initially located at h_i is the same as the visibility polygon $V_{maj}(\gamma)$ for γ in the majority-rule map.*

We denote the number of reflex vertices in the map polygon P by r.

Theorem 10 *K_i's are polygons with $O(n)$ edges of which at most r lie on ∂K_i (i.e., lie inside P_{maj}). Further, the edges composing ∂K_i form a convex chain.*

Proof. Superimpose the polygon P_i on the majority-rule map. Each edge $e \in P_i$ will be broken into sub-edges $e_1, e_2, \ldots, e_m$ such that each e_j either : (i) lies inside P_{maj} (except for its end points) (ii) lies outside P_{maj}, or (iii) lies on the boundary of P_{maj}.

We can call this the subdivision S_i of P_i induced by the majority-rule map. Let us first count the number of new edges introduced.

The endpoint of each edge in S_i is the intersection of polygon P_i with some polygon P_j for $j \neq i$. For $j \neq i$, P_i can intersect P_j in at most $O(n^2)$ points and therefore the total number of subdivisions of P_i will be $O(kn^2)$.

Now a robot located at h_i will find a deviation from P_{maj} only when it observes an edge e of type (i) or (ii). As before, there is a window $w(e)$ of the visibility polygon $VP(e)$ for any such edge which separates γ_0 from e. A robot located at γ_0 will see edge e as soon as it crosses $w(e)$.

We want the interesection of the regions containing the origin for all $w(e)$, where e is an edge of type (i) or (ii) in the subdivision of P_i. Each window is a line with its two end points on the boundary of P_i. The window-boundary will be the lower envelope of the arrangement formed by these lines with respect to the origin (see Figure 23). Therefore, each window will occur at most once on the window-boundary, in analogy with the lower envelope of an arrangement of lines in the plane. Further, the slopes of the edges will decrease from left to right and hence the boundary will be a convex chain. A first bound gives us $O(kn^2)$ as the size of ∂K_i.

Now we reduce it to r, the number of reflex vertices in P_i.

Note that each window $w(e)$ starts at a reflex vertex of P. Let v be such a vertex and let $w_1, w_2, \ldots, w_l$ be the windows associated with v. This constitutes a "fan" of line segments and we just need to take the last window w_l to get the region containing the origin for all windows $w_1, w_2, \ldots, w_l$ (see Fig. 25).

Therefore, each reflex vertex contributes one window. ∂K_i is obtained by taking the *lower envelope* of these edges with respect to the origin. Therefore, its boundary has at most r window edges.

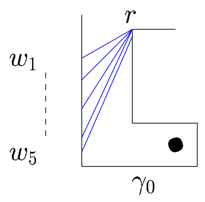


Figure 25: Fan of windows at a reflex vertex r

The other edges bounding K_i belong to P_i and are of type (iii). Hence, they also form a part of the majority-rule map. ■

Our version of lemma 15 will be different. A robot may located at h_i may half-localize even inside K_i, however the fact that any half-localization path has to cross half the K_i's still holds.

Lemma 18 *Any half-localization path crosses the boundary of at least half the K_i's.*

Proof. Let P be a half-localization path. Let S be the set of indices $i \in [1, k]$ such that P stays inside K_i for each $i \in S$. Then a robot located at $h_i, i \in S$ will see $V_{maj}(\gamma)$ at each coordinate $\gamma \in P$. Hence it will be unable to distinguish between any two hypotheses in S. Hence, on reaching the end of P, the robot will be left with a superset of $\{h_i | i \in S\}$ which is greater than $\frac{1}{2}|H|$. This contradicts the fact that P is a halving path. ■

The converse is simple :

Lemma 19 *Any path P inside the majority-rule map which crosses the boundary of at least half the K_i's gives a half-localization strategy of cost $|P|$.*

Proof. Let S be the set of indices $i \in [1, k]$ such that P crosses the boundary of K_i. The half-localization strategy will follow P and make an observation only when the robot crosses a region boundary ∂K_i. A robot initially located at h_i for $i \in S$ will find a deviation from majority-rule map when it crosses the boundary ∂K_i. Thus the robot will reach the end of path P only when it belongs to the h_i where $i \notin S$. By the assumption $|S| \geq \frac{1}{2}|H|$, the set $H' = \{h_i, i \notin S\}$ has cardinality at most $\frac{1}{2}|H|$. Hence the robot will also half-localize if it reaches the end of P. ■

Now the only step remaining is to choose Q_H. Here we will use the intersection of ∂K_i's with the $\log k$ $k \times k$ grids constructed in section 6.4.

By theorem 10, ∂K_i's are convex chains. A convex chain can intersect a $k \times k$ grid in at most $3k$ points. Since there are k chains ∂K_i and $\log k$ grids, the total number of reference points is at most $3k \log^2 k$. Thus, we have that :

Theorem 11 *For polygons P without holes, the set of reference points has size $O(k^2 \log k)$ where k is the number of hypotheses.*

Note that this bound doesn't hold when the boundaries are k-levels in arrangments of pseudosegments as they are non-convex. Further, our bounds don't involve the size n of the map polygon.

Polygons with holes. For polygons with convex holes the above construction gives us $hk^2 \log k$ grid points, where h is the number of holes. This is because each ∂K_i will now consist of at most $h + 1$ disjoint convex chains.

To see this, note that K_i is a region in the translated polygon P_i containing the origin γ_0. Edges from each hole in P will form a contiguous chain on K_i and this chain will occur only once. In between the chains for two holes, we will have a convex chain due to edge windows. Thus one can consider the boundary of K_i as a circle with $h + 1$ disjoint segments, one for each hole and one for the boundary of K_i common with the map polygon. The boundary of K_i then consists of the remaining segments of which there are at most $h + 1$.

Let us state this as a theorem :

Theorem 12 *For polygons P with convex holes, the set of reference points has size $O(hk^2 \log k)$ where h is the number of holes and k is the number of hypotheses.*

When the holes are non-convex, each hole can occur more than once on the boundary of K_i. Thus, there can be as many as r convex chains in each K_i, where r is the number of reflex vertices in the holes.

Therefore, we get $O(rk^2 \log k)$ reference points :

Theorem 13 *For polygons P with non-convex holes, the set of reference points has size $O(rk^2 \log k)$ where r is the number of reflex vertices in holes and k is the number of hypotheses.*

We note that this bound is much better than what we will prove when we discuss polygons with holes in the next chapter.

6.8 Additional Remarks

6.8.1 Bibliographic note

We now compare previous work based on the greedy strategy of Dudek *et al.* [15] with our own algorithms. The greedy strategy **MDL** (stands for Minimum Distance Localization) always goes to the nearest informative point at each iteration.

For the grid graph model, a robot following **MDL** first computes a *unanimous-rule* map i.e., the connected component O of all grid cells which are traversable for all hypotheses. A cell at the boundary of O is blocked if it is blocked relative to *at least one* hypothesis.

Strategy **MDL** (Minimum Distance Localization) visits the nearest blocked cell in O and makes an observation. It updates the set of hypotheses using this observation and then retraces its path back to the origin. We repeat this till we localize.

Clearly, each iteration removes at least one hypothesis, so there will be at most k such iterations. Further, the travel cost in each iteration is less than the optimal verification tour, which is itself less than the optimal strategy. This gives a $O(k)$-competitive algorithm. The same analysis holds for the approximation algorithm.

Note that the majority-rule map allows for the removal of at least half the hypotheses, whereas a robot using unanimous-rule map might remove just one hypothesis in each iteration. This allows for the significantly better approximation factor of strategy **RHL**.

To extend their algorithm to the polygonal model, they compute O by taking the intersection of shifted copies $P_1, P_2, \ldots, P_k$ of the polygon with respect to different hypotheses. A robot has to check the boundary of O to get new information. However, a robot may check whether an edge $e \in O$ exists by going to the boundary of its window $w(e)$ inside O. Therefore, we take the intersection of the regions formed by cutting off O at the various edge windows $w(e_1), w(e_2), \ldots, w(e_m)$. We call this restricted region U. The robot then needs to visit the nearest point on the boundary of U to get new information.

We refer the reader to the paper by Rao *et al* [41, 42] for the above construction as well as randomized variants of **MDL**.

6.8.2 Is a factor of 2 necessary ?

We first note that there does not exist a polynomial-size set of coordinates Q_H such that every *optimal* path that half-localizes has bend points in Q_H. In particular, in Figure 26 we illustrate that there can be an exponential number of distinct points at which an optimal path visits a given subset, S, of a sequence of segments, "reflecting" off of each segment, according to the usual local optimality condition. In particular, there are $2k + 1$ line segments, arranged in two parallel rows of k segments each. Let $l_0, l_2, \ldots, l_{2k-2}$ denote the line segments in the top row and $l_1, l_2, \ldots, l_{2k-1}$ the line segments in the bottom row. The origin γ_0 is located symmetrically to the left of l_0 and l_1. The remaining line segment l_{2k} is placed opposite to the origin on the other side of the rows. Let $S \subseteq [0, 2k-1]$ denote a subset of the line segments forming the two rows. Let C_S denote the shortest length path visiting segments in S in increasing order of index and ending at segment l_{2k}. Then one can show that the $2^{\Omega(k)}$ spanning paths contain an exponential number of distinct reflection points. Figure 26 shows this for the case $k = 2$.

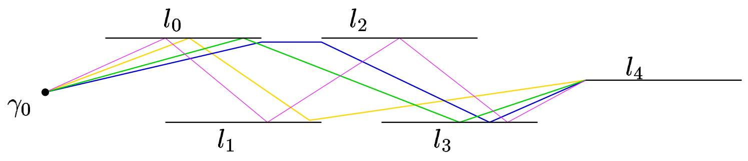


Figure 26: The construction showing the need for approximation with $k = 2$. Four shortest paths for the sequences of cells $(l_0, l_1, l_2, l_3, l_4)$, (l_0, l_2, l_4), (l_0, l_3, l_4) and (l_0, l_2, l_4) are also shown.

6.8.3 Comparison with visibility skeleton

In section 6.3.2 we construct cells of the majority-rule map which distinguish between hypotheses according to their visibility polygons. On the other hand, the previous constructions of Guibas *et. al.* [20] and Dudek et al. [15] decompose the plane according to an approximation called the visibility skeleton. We now show that an algorithm using visibility skeletons can perform much worse than one using visibility polygons.

Intuitively, a visibility skeleton is a contraction of the visibility polygon $\mathcal{V}$ so that the skeleton boundary consists of only those vertices that can be certified to be the vertices of V. The main loss in information is as follows : there may be a partial edge in the visibility polygon whose end points are blocked by two reflex vertices. The visibility skeleton remembers the "slope" of the line containing this edge but not its visible distance and length.

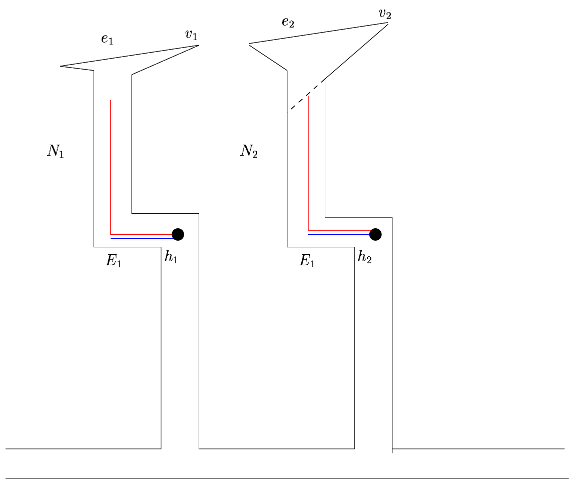


Figure 27: Comparison with visibility skeleton

Figure 27 illustrates the advantage gained by describing decompositions with respect to visibility polygons. The north-south corridors N_1 and N_2 are very long compared to the east-west corridors E_1 and E_2. Edges e_1 and e_2 have the same slopes but edge e_1 is "nearer" than edge e_2. The robot is located at one of the two hypotheses h_1 and h_2.

A robot using visibility polygons will localize as soon as it enters the north-south corridor. This is because the *distance* and *length* of partial edge e_1 for a robot located at the start of N_1 will be smaller than that of partial edge e_2 for a robot located at the start of N_2.

On the other hand, a robot using visibility skeletons will need to go up its northern corridor till it finds a new vertex. The earliest such vertex is v_2 for hypothesis h_2. Therefore the robot will go up the northern corridor till the window formed by v_2. If the robot sees v_2 it concludes that it is at hypothesis h_2, otherwise it localizes to h_1.

Thus our algorithm performs considerably better if we use visibility polygons instead of visibility skeletons.

CHAPTER VII

EXTENSIONS AND ADDITIONAL QUESTIONS

7.1 Extensions to Other Models

Here we sketch some extensions of our algorithm.

7.1.1 Robot without compass

If the robot does not possess a compass, but has no actuator uncertainty with respect to changes in orientation, the lower bound remains valid. For the algorithm, redefine a hypothesis to be a (location, orientation) pair. For grid graphs, with 4 axis-parallel orientations per cell, the size of the set H of possible hypotheses remains $O(n)$, and the algorithm extends naturally as the robot operates on the majority-rule map relative to (location, orientation) pairs in H.

For polygons there are at most n distinct embeddings, corresponding to rotations, of the visibility polygon $\mathcal{V}(h_i)$ for each choice of h_i. This follows since any one edge of $\mathcal{V}(h_i)$ that is not collinear with h_i (as is the case for "shadow edges" or "windows" of $\mathcal{V}(h_i)$) must fall on one of the n edges of P in any candidate pose. Thus, H consists of at most n different poses, (h_i, θ_j), each specified as a (location, orientation) pair. For each pose (h_i, θ_j), we construct a copy $P_{i,j}$ of the map polygon P. $P_{i,j}$ is formed by first translating P so that h_i coincides with the origin, and then rotating it about h_i so that direction θ_j points to the north. The majority-rule map and Algorithm 3 are then directly applied to the polygons $P_{i,j}$, as in the translation-only case.

7.1.2 The limited-range version

Practical sensors have a limited range, D, beyond which the noise levels are too high to give reliable measurements [33]. Our algorithm for grids already assumes limited range of visibility, since we assume that the robot senses only the immediate neighboring grid cell; this can readily be extended to allow the robot to sense all cells within grid graph distance D. Our algorithm for localization in polygons can also be extended to the limited-range case, as we now describe.

In order to distinguish between hypothesis h_i and hypothesis h_j, the robot must get within distance D of an edge of type (i) or type (ii) in the polygon F_{ij}. If γ_0 sees (within distance D) any point on an edge of type (i) or type (ii), then the robot can distinguish between h_i and h_j without moving from the origin γ_0. Thus, assume that all edges of F_{ij} that are visible (within distance D) from γ_0 are of type (iii). Let e be an edge of F_{ij} of type (i) or of type (ii). Assuming an unobstructed space, the set of points within distance D from some point of e is a region bounded by straight edges and circular arcs (of radius D). The portion of the boundary of $VP^{(D)}(e)$ that separates γ_0 from e defines the window, $w(e)$, of e; it consists now of $O(1)$ curves (straight segments and radius-D arcs), instead of a single chord, as in the $D = \infty$ case. We now define $G_{ij}^{(D)}$ to be the face containing the origin γ_0 in the arrangement of the $O(n)$ boundary edges of F_{ij} together with the set of all $O(n)$ windows $w(e)$ for edges e of type (i) or (ii). Again, as in lemma 14, we have that a robot can distinguish between hypothesis h_i and hypothesis h_j if and only if it visits the boundary, $\partial G_{ij}^{(D)}$. This allows us to define the majority-rule map regions g_i, the discrete point set Q_H, and the half-localization algorithm as before. The only technical difference is the presence of straight segments and (fixed radius) circular arcs in the arrangements; this does not affect the polynomiality or the correctness of the algorithm.

7.1.3 Polygons with holes

We note that by Theorem 13, one can construct a set of at most $O(hk^2 \log k)$ reference points for polygons with h convex holes. For non-convex holes, our construction gives a set of $O(rk^2 \log k)$ reference points, where r is the total number of reflex vertices on the boundary of holes.

However, the analysis below is given for the definition of K_i in lemma 15.

In the case that the map polygon P is a polygonal domain with holes $H_1, H_2, \ldots, H_m$, our method still applies, but the complexity of the structures increases. Let n be the total number of vertices in the polygon (including the holes).

First, the polygons F_{ij} are now polygons with holes of complexity $O(n)$. A single edge can now have as many as $O(n)$ windows (one for each hole). The G_{ij}'s are formed as before by taking the intersection of the regions chopped off by these windows; each has complexity $O(n^2)$ since each window-edge can occur only once. Finally, the majority-rule map's K_i's are also polygons with holes; each is formed from $G_{ij}, j \neq i$ and hence has complexity $O(k^2 n^4)$. The specification of the discrete points Q_H applies to the case of multiply-connected domains, and the argument of lemma 17 applies as well. And hence the set of coordinates now has complexity $O(k^3n^4)$. These bounds can be improved somewhat when the holes are convex.

From the above section, it is clear that the same framework works for a robot with *limited range sensors* inside a polygonal map with *holes*.

7.1.4 Geometric Trees

As described before, a *geometric tree* $G = (V, E)$ is a tree with V a set of points in $\mathbb{R}^d$, and E a set of *non-intersecting* line segments whose endpoints all lie in V [25]. An $O(n^{\frac{2}{3}})$-competitive localization strategy for bounded-degree geometric trees was given by Kleinberg [25]. His strategy is $\Omega(n)$-competitive for trees with arbitrary degree.

Our approach gives an $O(\log^2 n \log k)$ algorithm for any geometric graph $G = (V, E)$ in the plane, not just trees. First we can assume that the robot begins at some vertex of G, since the robot can initially perform a two-way spiral search to reach the closest vertex, while traveling at most 9 times the cost of optimal strategy [2, 25]. The set of hypotheses is now of size $k = O(n)$ and consists only of vertices. Make $k = |H|$ translation-congruent copies of G, with the i'th copy G_i having h_i at the origin. To construct the majority-rule map, overlay the copies $G_i, 1 \le i \le k$ and form the arrangement $D(G, H)$ of line segments in $\bigcup_i G_i$. Each edge in the arrangement has the same hypothesis partition, and and hence the robot gains new information only by visiting new edges. Note that several edges may be collinear, since new points are added by translation. Next, construct the majority-rule map G_{maj} by finding the set of all half-traversable edges reachable from the origin γ_0. As the robot can visit an edge of only through one of its endpoints, an $O(\log^2 n)$-factor halving path can be found by solving the $\frac{1}{2}$-Group Steiner problem on vertices of G_{maj}. Since $D(G, H)$ is formed by the intersection of $k \cdot |E|$ edges, it has complexity at most $O(k^2|E|^2)$ and can be computed in $O(k^2|E|^2)$ time by standard methods [3]. Since at least one endpoint of every edge in the majority-rule map G_{maj} is a vertex $v \in \bigcup_i G_i$, G_{maj} has $O(n|E|)$ edges. Hence the computation time of the robot is $\mathcal{P}(n|E|) \cdot \log n$, where $\mathcal{P}(\cdot)$ is the running time for approximating $\frac{1}{2}$-Group Steiner problem. Since a grid graph is a geometric graph in $\mathbb{R}^2$, corollary 2 gives an $\Omega(\log^{2-\epsilon} n)$ lower bound.

7.1.5 Three-dimensional grid graphs

Finally, we consider a three-dimensional grid graph G, which can be used to model buildings or offices with several floors. The majority-rule map G_{maj} is a local-map in which each cell is blocked or unblocked based on what the majority of hypotheses have to say about it. If G is a $l_1 \times l_2 \times l_3$ cuboid, the majority-rule map has size $(2l_1 - 1) \times (2l_2 - 1) \times (2l_3 - 1)$, since the absolute values of of x-, y- and z-coordinates for each hypothesis are $l_1 - 1$, $l_2 - 1$ and $l_3 - 1$, respectively. Hence G_{maj} requires space at most $8n$, and can be computed in $O(nk)$ time. By making one vertex for each cell in G_{maj}, we solve a $\frac{1}{2}$-Group Steiner problem $\mathcal{I}_{G,H}$ of size $O(n)$. The performance ratio remains $O(\log^2 n \log k)$, and the running time is $\mathcal{P}(8n) \log n$.

7.2 Is returning to the origin necessary?

In this section, we show that **RHL** performs very poorly if we do not stipulate that the robot returns to its starting position after each half-localize step. In Section 7.2.1, we construct a grid graph G and a set of hypotheses H such that a robot not returning to the origin travels distance $(k - 1 - \epsilon) \cdot OPT(G, H)$ where $k = |H|$ is the number of hypotheses and $\epsilon > 0$ is an arbitrarily small constant. In Section 7.2.2, we show that our lower bound is tight by proving that a robot not returning to the origin always localizes in at most $(k - 1) \cdot OPT(G, H)$ steps.

7.2.1 Lower Bound

The grid graph G for the lower bound is reminiscent of the Group Steiner tree construction. Let k be the number of hypotheses and x be an integer greater than or equal to 3.

The building block B consists of two orthogonal corridors meeting at a corner where the robot is located (see Figure 28(a)) The northern corridor has length $x + 1 + \log k$ and the eastern corridor has length $(k - 1) \cdot x$

We make k copies $B_0, B_1, \ldots, B_{k-1}$ of block B. B_y's are the same except for distinguishing "alcoves" along their northern and eastern corridors.

We now describe the construction of B_y. A set of $\log k$ alcoves encoding y in binary are added along the western edge of the northern corridor (see Figure 28(a)). The jth alcove from the bottom is blocked iff the jth bit in the binary encoding of y is 0.

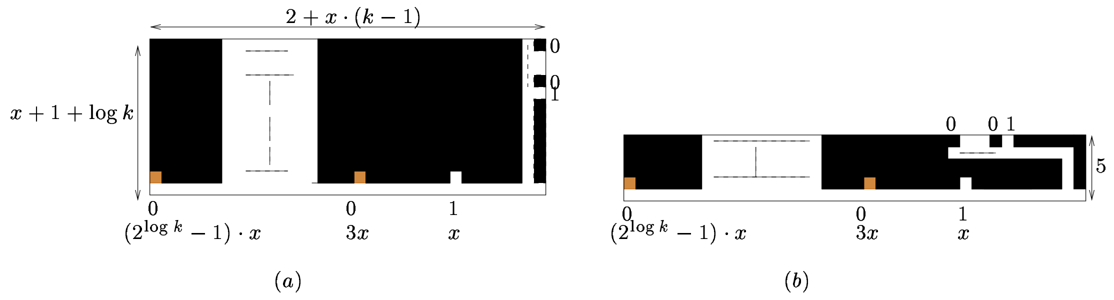


Figure 28: (a) Block B_y where $y = 10\dots0$. (b) The northern corridor is bent at 3 units.

In addition to this, we add $\log k$ alcoves encoding y in binary along the eastern corridor. The ith alcove is placed at distance $(2^i - 1)x$ and is blocked iff the ith bit in binary encoding of y is 0.

Observe that B_y fits in a $a \times b$ rectangle where $a = 2 + x \cdot (2^{\log k} - 1) = 2 + x \cdot (k-1)$ and $b = x + \log k$. One can further reduce the height of B_y by bending the northern corridor as shown in Figure 28(b). After this reduction, each block fits in a $(2 + x \cdot (k-1)) \times 5$ rectangle.

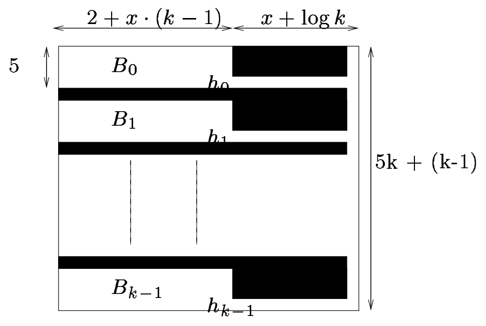


Figure 29: Grid graph G

Grid graph G is a $(2 + x \cdot k + \log k) \times (6k-1)$ rectangle formed by connecting blocks $\{B_y\}_y$ as shown in Figure 29. $B_0, B_1, \dots, B_{k-1}$ are placed along the left edge of G separated by east-west walls of width 1. A north-south corridor of width 1 runs alongside the right edge of G. The south-west cell of each block is connected to this corridor by an east-west corridor of length $x + \log k$. This distance is chosen so that a robot located inside a block B_y never goes outside it to half-localize. Finally, the set of hypotheses H equals $\{h_0, h_1, \dots, h_{k-1}\}$ where h_y denotes the cell at the intersection of the two orthogonal corridors in B_y.

Theorem 14 *Let G be the* $(2 + x \cdot k + \log k) \times (6k-1)$ *grid graph as constructed above and* $H = \{h_0, h_1, \dots, h_{k-1}\}$*. Then a robot which computes the optimal half-localization strategy but* *does not return to the origin travels to the end of the eastern corridor of its block B_y before it localizes.*

Proof. Consider a robot located at h_y where $0 \leq y \leq k-1$. To find its location the robot needs to find all bits in the binary representation of y. To half-localize, it suffices to read 1 new bit in each phase.

The robot can either read the first alcove on the northern corridor or the first alcove on the eastern corridor. Since the alcove on the eastern corridor is nearer by one grid cell, the robot moves x units east and "reads" the first alcove.

Suppose that the alcove is blocked i.e., the first bit in the binary encoding of y is 0 (the case when it is 1 is similar). To read the next bit, the robot can either read the second alcove on the eastern corridor at cost $2x$, or it can go back to the origin and then read the second alcove on the northern corridor at cost $2x+2$. Since the former is optimal, the robot moves $2x$ units west to read the second alcove on the eastern corridor.

In general, at the start of the ith phase ($i \geq 2$) the robot has read the first $i-1$ bits of the binary encoding of y and is located at alcove $i-1$ on the eastern corridor. Either it can move $2^{i-1}x$ steps to the west and check the ith alcove on the eastern corridor or it can go back to read the ith alcove at the northern corridor at cost $2^{i-1}x+i$. The optimal half-localization plan consists of going west to read the ith alcove on the eastern corridor.

Thus in each half-localize phase the robot goes west to read the next alcove on its eastern corridor. The robot will localize after it has gone till the end of the eastern corridor and checked the last alcove. The total distance traveled by the robot is $(k-1) \cdot x$ ■

Corollary 4 *For every fixed $\epsilon > 0$, there is a grid graph G and a set of hypotheses H such that a robot following* **RHL** *without returning to the origin travels at least $(k-1-\epsilon) \cdot OPT(G,H)$ distance before determining its location $h \in H$.*

Proof. Take the grid graph G and the set of hypotheses H constructed above. By Theorem 14 a robot not returning to the origin travels distance $(k-1) \cdot x$ to the end of the eastern corridor to determine its initial position $h_y \in H$

The optimal localization strategy consists of going x units up the northern corridor and then reading the $\log k$ bit signature. It has cost $OPT = x + \log k$.

The approximation factor is $\frac{(k-1)\cdot x}{x+\log k}$. If we take $x = \frac{(k-1)\log k}{\epsilon}$, this is at least $k - 1 - \epsilon$ ■

The next corollary shows that the lower bound in terms of the size of the grid graph is $\Omega(\frac{\sqrt{n}}{\log n})$.

Corollary 5 *There is a grid graph G and set of hypotheses H such that a robot following* **RHL** *without returning to origin travels distance $\Omega(\frac{\sqrt{n}}{\log n}) \cdot OPT$, where $n = |G|$ is the size of the grid graph.*

Proof. Take $x = 3$ in Theorem 14. The grid graph now has size $n = (2 + 3k + \log k) \times (6k - 1) = \theta(k^2)$. The optimal localization strategy has cost $3 + \log k$. The robot travels distance $3(k-1)$. The approximation factor is $\frac{3(k-1)}{3+\log k}$. Since $k = \theta(\sqrt{n})$, this is $\Omega(\frac{\sqrt{n}}{\log n})$ ■

7.2.2 Upper Bound

We now show that a robot following strategy **RHL** without returning to origin localizes in at most $(k-1) \cdot OPT$ steps.

Theorem 15 *Consider a robot which computes the optimal half-localization strategy in each phase, but does not return to the origin after each phase. Then it travels distance at most $(k-1) \cdot OPT$ before determining its initial position $h \in H$, where $k = |H|$ is the number of hypotheses and OPT is the cost of the optimal localization plan.*

Proof. Let P_{i-1} denote the path traced by the robot relative to the origin before the start of the ith half-localize phase. Let Q_i denote the path traced by the robot during the ith phase. Then we have that $P_i = P_{i-1} \circ Q_i$ is the concatentation of P_{i-1} followed by Q_i.

Since the robot always chooses the optimal half-localization strategy, the length of Q_i is less than or equal to any half-localization strategy for phase i. One such strategy makes the robot *retrace* the path P_{i-1} back to the origin and then run the optimal localization plan till the robot half-localizes. This has cost at most $|P_{i-1}| + OPT$ and hence we have that $|Q_i| \leq |P_{i-1}| + OPT$.

Therefore we get that $|P_i| = |P_{i-1} \circ Q_i| = |P_{i-1}| + |Q_i| \leq |P_{i-1}| + (|P_{i-1}| + OPT) = 2 \cdot |P_{i-1}| + OPT$.

Since $|P_0| = 0$, we see that $|P_i| \leq (2^i - 1) \cdot OPT$. As the robot localizes in at most $m \leq \log k$ half-localize steps, the distance traveled $|P_m|$ is at most $(2^m - 1) \cdot OPT \leq (2^{\log k} - 1) \cdot OPT = (k - 1) \cdot OPT$ ■

7.2.3 Discussion

This feature may become a problem in probabilistic environments where the robot may incur noise by returning back to the origin, or if the robot gets trapped in a small corner from which it is hard to get out. However, we still feel our algorithm makes sense, due to the large decrease in uncertainty brought about by each half-localize step. If the robot motion is sufficiently correct, this decrease should more than offset the noise incurred by coming back to the origin. Further, the robot does not need to return to the origin "exactly". Rather, it suffices that the robot be present within a small distance of the origin with high probability, as this will allow for near-optimal behavior in the next step. If continuous sensing and updating while returning back is allowed, the robot should perform reliably with small corrections.

The robot may get trapped in a corner, but in maps with "signatures" (such as the ones we constructed above, as well as those in the NP-hardness construction of Dudek *et al.* [15]), this may be the only way to localize efficiently. In fact, it seems that only in such highly-replicated environments do such localization strategies make sense.

Further, the task of localization is just a prelude to the robot performing other tasks, such as going to a particular location. This new location may lie anywhere in the map, so the robot will not lose by coming back to its starting place.

Finally, it seems that there is no way to bypass this return-to-origin constraint, as not allowing the robot to return to origin leads to exceptionally bad performance. In fact, we believe that no reasonable algorithm for localization can be found unless we stipulate that the robot returns to the starting location after each half-localize step.

We add that a robot following strategy $\mathbf{MDL}$ (see section 6.8.1) which does not return to the origin may travel $(2^k - 1) \cdot OPT$ distance in the worst-case and that this bound is tight.

CHAPTER VIII

CONCLUSION

The popular robot-navigation method that we have analyzed in this thesis, D^*, is appealingly simple and easy to implement from a robotics point of view and appealingly complicated to analyze from a mathematical point of view. Our results, likewise, are satisfying in two ways. First, our tighter upper bounds on worst-case travel distances guarantee that D^* cannot perform badly at all. Second, the gap between the best known lower and upper bounds is now quite small, namely $O(\log\log n)$ for planar graphs, and $O(\log n \log\log n)$ on arbitrary graphs. An open question is to close the gap between the upper and lower bounds.

The main ideas of our localization algorithm are half-localization and the majority-rule map, which permit us to eliminate half the hypotheses in each step. Earlier strategies for localization could eliminate only $O(1)$ hypotheses in each step, thus leading to $\Omega(n)$-approximations for general models. There is a $\log n$ factor gap between the upper and lower bounds; it appears that this gap can only be closed by progress on the Group Steiner problem in grid graphs (and also those given by Euclidean shortest path metrics inside a constrained region).

An appealing feature of our algorithm is its wide adaptability over a variety of robot models: the only issue is to devise algorithms for computing the majority-rule map and the set of coordinates for the model at hand. We believe that the majority-rule map will play an important part in other robot navigation problems.

While our algorithms for localization in polygons have been restricted to two dimensions, we expect that the results can be extended to three dimensional polyhedral domains P in which the robot moves inside P and sends out a series of beams spaced at small solid angles over the sphere and joins them to compute the *visibility polyhedron* $V(p)$. Modern 3D-range finders allow one to estimate the visibility polyhedron from the robot [57].

In this thesis, we do not address models with sensor noise, imperfections in the robot's map and odometer errors. While sensor noise can be easily accommodated [7], devising a good strategy for a model with odometer errors remains a major open problem. This not only entails redefining what we mean by localization, but also requires devising strategies that balance the need for resolving global position with the need for removing local pose estimation errors.

The best model for odometer and sensor errors is probabilistic robotics. As a simple example, take a grid graph. If the robot ”decides” to move North, it may do so only by probability 0.7. On the other hand, it may instead move S, E and W with probability 0.1 each.

Recently (since late 90s), probabilistic robotics has become a major paradigm of applied robotics, but has not been imported into theoretical studies. Thus, we model the environment as a Partially Observable Markov Decision Process (POMDP). To be exact, here we are dealing with maze POMDPs. At any time, the robot’s position will be known upto a probability distribution over the set of possible locations. This distribution will be updated using bayesian method based on the next move and sensor observation. The general case is expected to be very hard (above PSPACE etc.). However, a good research direction is to explore some simple problems within the probabilistic framework.

We firmly believe that accurate theoretical analysis of real-life problems in a variety of models is useful for practical robotics. A correct understanding of the hard cases and the underlying structure allows the implementor to design more meaningful heuristics. These heuristics may not be analyzable rigorously but will have a theoretical watermark with respect to which their performance can be measured and intuited.

REFERENCES

[1] Alon, N., Hoory, S., and Linial, N., "The moore bound for irregular graphs," *Graph and Combinatorics*, vol. 18, no. 1, pp. 53–57, 2002.

[2] Baeza-Yates, Culberson, and Rawlins, "Searching in the plane," *INFCTRL: Information and Computation (formerly Information and Control)*, vol. 106, 1993.

[3] Balaban, "An optimal algorithm for finding segment intersections," in *COMPGEOM: Annual ACM Symposium on Computational Geometry*, 1995.

[4] Bartal, Y., "Probabilistic approximations of metric spaces and its algorithmic applications," in *FOCS*, pp. 184–193, 1996.

[5] Bateman, C. D., Helvig, C. S., Robins, G., and Zelikovsky, A., "Provably good routing tree construction with multi-port terminals," in *ISPD '97: Proceedings of the 1997 international symposium on Physical design*, (New York, NY, USA), pp. 96–102, ACM, 1997.

[6] Blum, A., Raghavan, P., and Schieber, B., "Navigating in unfamiliar geometric terrain," *SICOMP: SIAM Journal on Computing*, vol. 26, 1997.

[7] Brown, R. G. and Donald, B., "Mobile robot self-localization without explicit landmarks," *Algorithmica*, vol. 26, pp. 515–559, 2000.

[8] Brumitt, B. and Stentz, A., "GRAMMPS: a generalized mission planner for multiple mobile robots," in *Proceedings of the International Conference on Robotics and Automation*, 1998.

[9] Chan, T. M., "On levels in arrangements of curves," in *Proc. 41st IEEE*, pp. 219–227, 2002.

[10] Charikar, M., Chekuri, C., Cheung, T., Dai, Z., Goel, A., Guha, S., and Li, M., "Approximation algorithms for directed steiner problems," *ALGORITHMS: Journal of Algorithms*, vol. 33, 1999.

[11] Chekuri, C. and Pál, M., "A recursive greedy algorithm for walks in directed graphs," in *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, 2005.

[12] Cox, I. J., "BLANCHE: An experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Trans. Robotics and Automation*, vol. 7, pp. 193–204, 1991.

[13] Datta, A. and Icking, C., "Competitive searching in a generalized street," *Comput. Geom*, vol. 13, no. 2, pp. 109–120, 1999.

[14] Davis, E., *Representing and acquiring geographic knowledge*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1986.

[15] Dudek, G., Romanik, K., and Whitesides, S., "Localizing a robot with minimum travel," *SICOMP: SIAM Journal on Computing*, vol. 27, no. 2, pp. 583–604, 1998.

[16] EVEN, G., KORTSARZ, G., and SLANY, W., "On network design problems: fixed cost flows and the covering steiner problem," *ACM Trans. Algorithms*, vol. 1, no. 1, pp. 74–101, 2005.

[17] FAKCHAROENPHOL, J., RAO, S., and TALWAR, K., "A tight bound on approximating arbitrary metrics by tree metrics," *JCSS: Journal of Computer and System Sciences*, vol. 69, 2004.

[18] FOX, D., BURGARD, W., and THRUN, S., "Active markov localization for mobile robots," *Robotics and Autonomous Systems*, vol. 25, no. 3-4, pp. 195–207, 1998.

[19] GARG, N., KONJEVOD, G., and RAVI, R., "A polylogarithmic approximation algorithm for the group steiner tree problem," *ALGORITHMS: Journal of Algorithms*, vol. 37, no. 1, pp. 66–84, 2000.

[20] GUIBAS, L., MOTWANI, R., and RAGHAVAN, P., "The robot localization problem," *SICOMP: SIAM Journal on Computing*, vol. 26, no. 4, pp. 1120–1138, 1997.

[21] HALPERIN, E. and KRAUTHGAMER, R., "Polylogarithmic inapproximability," in *STOC: ACM Symposium on Theory of Computing (STOC)*, pp. 585–594, 2003.

[22] HEBERT, M., MCLACHLAN, R., and CHANG, P., "Experiments with driving modes for urban robots," in *Proceedings of the SPIE Mobile Robots*, pp. 140–149, 1999.

[23] HYAFIL, L. and RIVEST, R. L., "Constructing optimal binary decision trees is np-complete," *Information Processing Letters*, vol. 5, no. 1, pp. 15–17, 1976.

[24] KLEIN, R., "Walking an unknown street with bounded detour," *Comput. Geom. Theory Appl.*, vol. 1, pp. 325–351, 1992.

[25] KLEINBERG, J. M., "The localization problem for mobile robots," in *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 521–531, 1994.

[26] KLEINBERG, J. M., "On-line search in a simple polygon," in *SODA*, pp. 8–15, 1994.

[27] KOENIG, S. and LIKHACHEV, M., "Improved fast replanning for robot navigation in unknown terrain," in *Proceedings of the International Conference on Robotics and Automation*, pp. 968–975, 2002.

[28] KOENIG, S., TOVEY, C., and HALLIBURTON, W., "Greedy mapping of terrain," in *Proceedings of the International Conference on Robotics and Automation*, pp. 3594–3599, 2001.

[29] KOENIG, S., TOVEY, C., and SMIRNOV, Y., "Performance bounds for planning in unknown terrain," *Artif. Intell.*, vol. 147, no. 1-2, pp. 253–279, 2003.

[30] LUND, C. and YANNAKAKIS, M., "On the hardness of approximating minimization problems," *Journal of the ACM*, vol. 41, no. 5, pp. 960–981, 1994.

[31] MATTHIES, L., XIONG, Y., HOGG, R., ZHU, D., RANKIN, A., KENNEDY, B., HEBERT, M., MACLACHLAN, R., WON, C., FROST, T., SUKHATME, G., MCHENRY, M., and GOLDBERG, S., "A portable, autonomous, urban reconnaissance robot," *Robotics and Autonomous Systems*, vol. 40, pp. 163–172, 2002.

[32] MILLER, D. P., ATKINSON, D. J., WILCOX, B. H., and MISHKIN, A. H., "Autonomous navigation and control of a mars rover," in *In Proc. 11th IFAC Symp. on Aut. Ctrl. in Aerospace*, pp. 127–130, July 1989.

[33] MILLER, G. L. and WAGNER, E. R., "An optical rangefinder for autonomous robot cart navigation," pp. 122–134, 1990.

[34] MITCHELL, J. S., "Geometric shortest paths and network optimization," in *Handbook of Computational Geometry (eds. J.-R. Sack and J. Urrutia)*, pp. 633–701, Elsevier Science Publishers B.V. North-Holland, 1998.

[35] MUDGAL, A., TOVEY, C. A., GREENBERG, S., and KOENIG, S., "Bounds on the travel cost of a mars rover prototype search heuristic," *SIAM J. Discrete Math.*, vol. 19, no. 2, pp. 431–447, 2005.

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

[37] NOURBAKHSH, I. and GENESERETH, M., "Assumptive planning and execution: a simple, working robot architecture," *Autonomous Robots Journal*, vol. 3, no. 1, pp. 49–67, 1996.

[38] P. EADES, X. L. and WORMALD, N. C., "Performance guarantees for motion planning with temporal uncertainty," *Austal. Comput. J.*, vol. 25, no. 1, pp. 21–28, 1993.

[39] PAPADIMITRIOU, C. and YANNAKAKIS, M., "Shortest paths without a map," *Theoretical Computer Science*, vol. 84, pp. 127–150, 1991.

[40] PRIYANTHA, N. B., CHAKRABORTY, A., and BALAKRISHNAN, H., "The cricket location-support system," in *MOBICOM : ACM/IEEE Internation Conference on Mobile Computing and Networking*, pp. 32–43, 2000.

[41] RAO, M., DUDEK, G., and WHITESIDES, S., "Minimum distance localization for a robot with limited visibility," in *ICRA*, pp. 2438–2445, IEEE, 2005.

[42] RAO, M., DUDEK, G., and WHITESIDES, S., "Randomized algorithms for minimum distance localization," *Int. J. Rob. Res.*, vol. 26, no. 9, pp. 917–933, 2007.

[43] RENCKEN, W. D., "Concurrent localization and map building for mobile robots using ultrasonic sensors," in *In Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2129–2197, 1993.

[44] SCHUIERER, S., "Efficient robot self-localization in simple polygons," in *Intelligent Robots: Sensing, Modeling and Planning [Dagstuhl Workshop, September 1-6, 1996]*, pp. 129–146, World Scientific Press, 1997.

[45] SIPSER, M., *Introduction to the Theory of Computation; 2nd ed.* Cambridge: Thomson Course Technology, 2006.

[46] SMIRNOV, Y., *Hybrid Algorithms for On-Line Search and Combinatorial Optimization Problems*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh (Pennsylvania), 1997. Available as Technical Report CMU-CS-97-171.

[47] STENTZ, A., "Optimal and efficient path planning for partially-known environments," in *Proceedings of the International Conference on Robotics and Automation*, pp. 3310–3317, 1994.

[48] STENTZ, A., "The focussed D* algorithm for real-time replanning," in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1652–1659, 1995.

[49] STENTZ, A., "Optimal and efficient path planning for unknown and dynamic environments," *International Journal of Robotics and Automation*, vol. 10, no. 3, pp. 89–100, 1995.

[50] STENTZ, A., "CD*: A real-time resolution optimal re-planner for globally constrained problems," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 605–612, 2002.

[51] STENTZ, A. and HEBERT, M., "A complete navigation system for goal acquisition in unknown environments," *Autonomous Robots*, vol. 2, no. 2, pp. 127–145, 1995.

[52] THAYER, S., DIGNEY, B., DIAZ, M., STENTZ, A., NABBE, B., and HEBERT, M., "Distributed robotic mapping of extreme environments," in *Proceedings of the SPIE: Mobile Robots XV and Telemanipulator and Telepresence Technologies VII*, vol. 4195, November 2000.

[53] THOMPSON, W., PICK, JR., H., BENNETT, B., HEINRICHS, M., SAVITT, S., and SMITH, K., "Map-based localization: The 'drop-off' problem," in *DARPA90*, pp. 706–719, 1990.

[54] TOVEY, C. A. and KOENIG, S., "Gridworlds as testbeds for planning with incomplete information," in *Proc. of the 17th National Conf. on Artificial Intelligence and 12th Conf. on Innovative Applications of Artificial Intelligence*, pp. 819–824, AAAI Press / The MIT Press, 2000.

[55] WANG, C. M., "Location estimation and uncertainty analysis for mobile robots," pp. 90–95, 1990.

[56] WEI, G., WETZLER, C., and PUTTKAMER, E. V., "Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans," in *In Proc. 1994 IEEE Int. Conf. on Intelligent Robots and Systems (IROS '94*, pp. 595–601, 1994.

[57] WULF, O. and WAGNER, B., "Fast 3d-scanning methods for laser measurement systems," in *Intl. Conf. on Control Sys. and Comp. Sc. (CSCS14)*, 2003.