Randomized Navigation to a Wall through Convex Obstacles

Piotr Berman [*] Marek Karpinski [†]

Abstract

We consider the problem of navigating through an unknown envi-roment in which the obstacles are convex, and each contains a circle of diameter 1. The task is to reach a given straight line, in the distance n from our original position. Our randomized algorithm has competitive ratio $n^{0.75}$, and it uses tactile information only. This is the first algorithm that offers any competitive ratio for the problem.

[*]Dept. of Computer Science and Engineering, The Pennsylvania State University, berman@cse.psu.edu, partially supported by NSF Grant CRR-9114545

[†]Dept. of Computer Science, University of Bonn, marek@cs.uni-bonn.de, research partially supported by DFG Grant KA673/4-1 and ESPRIT Grants 7097 and ECUS 030

1 Introduction

The research on navigation, i.e., motion planning, is motivated by the possibility of using robots in some unknown enviroment, e.g. a surface of distanta moon. In such circumstances, the robot's operations should be highly autonomous, as the communication is a subject of very high delays. In planning such an operation, we may consider two alternatives. One is to invest first some resources and make a map of all impassable obstacles. Another is to make the robot to navigate among the obstacles that are learned as they are encountered.

The choice between two alternatives clearly depends on the difference between the cost of the robot reaching its objectives with and without the map. This difference depends on the nature of the robot's task, the shape of the obstacles, and, quite crucially, on the quality of the navigating algorithm that robot may use. Most commonly, the quality of a navigating algorithm is measured by its competitive ratio, i.e., the ratio of the lengths of the paths traversed by the robot without a map (using the algorithm) and with a map (the shortest obstacle-free path from the starting point to the target) respectively.

Navigation problems have four aspects: the target (a point, an infinite line, etc.), the placement of the obstacles (can be unrestricted, within a rectangle that has the starting on perimeter and the target point in the center, etc.), the shape of the obstacles (oriented rectangles, arbitrary rectangles, arbitrary convex figures, arbitrary polygons, etc.), and, lastly, the information available from the scene (visual or tactile). One can also formulate a motion planning problem in a graph, as in [4].

The complexity parameter n is the distance between the starting point and the target. To assure that there is a relationship between the complexity parameter and the competitive ratio, we assume that every obstacle contains a unit circle.

In several instances, tight bounds were derived for the attainable competitive ratio. One example is the room problem: the target in the center of a square containing all the obstacles, and the start at the perimeter. If obstacles are oriented rectangles, the competitive ratio for deterministic algorithms is $\Theta(\log n)$ (see [1]). If obstacles are convex, Blum *et al.* [3] show the deterministic lower bound $\Omega(n^{1/2})$ and randomized upper bound $O(n^{1/2})$ (both bounds assume that the robot receives the visual information).

In the *wall problem*, introduced by Papadimitriou and Yannakakis [7], the target is an infinite straight line. If the obstacles are oriented rectangles, then the competitive ratio for deterministic algorithms is $\Theta(n^{1/2})$ (see [3]). Recently, [2], an randomized algorithm was found that is better than the deterministic lower bound. If the obstacles are arbitrary convex figures, no better lower bound is known, but no matching competitive ratio was obtained. If the obstacles are arbitrary rectangles, one can apply the deterministic algorithm of Lumelsky and Stepanov [6] to obtain competitive ratio $O(n)$. For arbitrary convex obstacles, NO upper bound on deterministic competitive ratio is known.

In this paper, we provide a randomized algorithm for the last problem, with a competitive ratio $O(n^{3/4})$.

In practical terms, this competitive ratio is so high that in most circumstances we should rather invest in creating a map. We conjecture that there should exists much better randomized strategies. For example, there exists a simple algorithm that handles well the examples of scenes that are maximally difficult for the our algorithm, but which is not analyzed as yet. In general, for the case of arbitrary convex obstacles, there are more open problems than solved. To mention just a few: to find ANY competitive ratio for point to point navigation, to improve the lower bounds (currently, very weak for randomized algorithms, the best, $\Omega(\log\log n)$ is provided by Karloff *et al.* [5]), to improve the upper bound, to generalize to three dimensions.

2 Preliminaries

We assume that the robots moves on a plane, and that is always knows the coordinates of its current position. The initial position of the robots is (0,0), later we refer to this point as the **source**. The **target**is the line $\{x = n\}$.

To better present our algorithm, we will show it in four versions. The simplest forms an obvious solution for the case of oriented rectangles, with competitive ratio $n^{1/2}$. The second will solve the case of arbitrary rectangular obstacles, with competitive ratio $n^{3/4}$. The third will handle arbitrary convex obstacles, provided that they are “short”; the fourth will solve the general case.

All our algorithms will have the following structure:

set(m, l);
repeat
  rand.set(r);
  for *phase.no* $:= 1$ **to** m **do**

```
for attack.no := -(m - 1) to m - 1 do
    lower := line {y = (i - 1)l};
    upper := line {y = (i + 1)l};
    central := line {y = il};
    attack( attack.no );
  l := 2 * l
forever
```

When we discuss an algorithm with this structure, we refer to a single execution of the **repeat** loop as stage(l), referring to the value of l during this execution (before it is reset at the end). In a stage we execute m phases, each consisting of $2m - 1$ attacks. We assume that the robot interrupts the execution of the algorithm as soon as it reaches the **target**. In randomized algorithms, we will also enforce a cost limit for each stage, if the robot traverses the distance equal to the limit while executing a stage, it interrupts this stage and starts the subsequent one.

3 Oriented rectangles

Now we present an algorithm that handles *oriented rectangles*, i.e. the rectangles with sides parallel to the axes. Then we will analyze this algorithm in a way that can be generalized in the subsequent sections.

Initially, in set(m, l), the robot sets $m = \lceil n^{1/2} \rceil$ and $l = m$. In rand.set the robot does nothing (because this algorithm is deterministic). In attack(i) the robot moves toward the target as much as possible, but within the "corridor" bounded by **lower** and **upper** lines. At all times the robot moves according to the following rule: if possible, move directly toward the **target**(directly = parallel to x axis), if such motion is blocked byan obstacle, follow the algorithm.

```
attack(i):
  repeat
    move to central
    if the adjacent obstacle extends from lower to upper then
      exit
    else
      follow the obstacle around its nearer end back to central
```

forever

To show that the resulting algorithm leads the robot to the **target**with competitive ratio $O(m)$ it suffices to prove the following two properties:

- In stage(l) the robot traverses $O(m^2 l)$.
- If the robot does not reach the **target**before the end of stage(l), then the length of the shortest obstacle-free path from the **source**to the **target**is larger than ml.

To prove the first property, notice first that we need to estimate only the length of those moves of the robot that are parallel to y axis; the sum of movements parallel to x axis is bounded by n. A stage consists of m phases; the length of the first move of a phase (to **central** line of the first attack, i.e. to line $\{y = -(m-1)l\}$, is less than $2ml$, hence the total cost of such first moves is less than $2m^2 l$. In a stage, $(2m-1)m$ attacks are performed, the first move of an attack (except the first attack of a phase, which was already counted) has length lm, the last has length $2l$, the total cost of such moves is $3l(2m-1)m < 6m^2 l$. Any other move in an attack has length $2l$ and brings the robot closer to the **target**by at least 1, hence there are at most n such moves, with the total cost $n2l = 2m^2 l$. Summarizing, the total distance traversed by the robot is below $10m^2 l$.

We prove the second property in a way that we will be able to generalize later. Consider only those obstacles that were stopping the attacks of stage(l). Note first that if some two attacks from two different phases are stopped on the same obstacle, then this obstacle extends from $\{y = -ml\}$ to $\{y = ml\}$; such an obstacle alone forces the shortest path from the **source**to the **target**to be longer than ml. Thus later we may assume that this does not happen.

Let as define the *corridor* of obstacles A and B as the set of points (x_1, y_1) such that

- $(x_1, y_1) \notin A \cup B$, and
- $(x_0, y_1) \in A$ and $(x_2, y_1) \in B$ for some $x_0 < x_1 < x_2$.

Corridors of a phase are formed by pairs of obstacles that stop consecutive attacks (note though that some consecutive attacks are stopped by the same obstacle). The union of the obstacles and corridors of a phase forms a connected polygon, which we may call a *phase barrier*, that extends from $\{y = -ml\}$ to $\{y = ml\}$. If a path from the **source**to the **target**of length at most ml exists, it must be contained in the strip bounded by these two lines. After removing the phase barriers from this strip, it will be split into $m + 1$ connected components, the leftmost containing the **source**and the rightmost containing the the **target**. Therefore any path must traverse at least one corridor in each of the m phase barriers; as each corridor has the length of at least l, we can conclude that the length of such a path is at least ml.

4 Arbitrary rectangles

The previous algorithm does not work in the case of arbitrary rectangular obstacles. The reason is the following: when the robot moves to starts a subsequent attack, it may encounter an obstacle that can be followed either toward the **target**, but away from the **central** line of the attack, or toward the **central** line, but away from the **target**. The first choice could lead it to the **target**in a very ineficient manner, so the second choice may be necessary. However, when the robot is may move away from the **target**, it is difficult to estimate the number of moves in the attacks: the sum of the x components of moves may be much larger than n, so we loose our upper bound for these moves.

The solution to this dilemma is to choose a threshold angle that would work according to the following intuition. Imagine that the robot is pulled toward the **target**by gravitation, and that obstacles are slippery. On sufficiently steep obstacles (according to the threshold angle) the robot looses its footing and slides toward the target.

With every choice of the threshold angle there exists an associated danger: the robot may enter an exceedingly long oscillation, alternatingly sliding away from the **central** line of the current attack and climbing back toward it. To minimize this danger, we will select that angle at random, thus assuring that, on the average, only a small proportion of the obstacles in the scene has angles that are dangerously close to the threshold.

In set(m, l) the robot sets $m = \lceil n^{1/4} \rceil$ and $l = m^3$. The limit on the length of the path traversed in stage(l) is $cm^4 l$, where the constant c will be established later. In rand.set(r) the robot selects r, uniformly at random, from the set $\{m^2 + 1, m^2 + 2, \ldots, 2m^2\}$.

When the robot is in contact with an obstacle, it can sense the slope of its surface. We formalize it as follows. If the line tangent to the obstacle is parallel to the line $\{sy = lx\}$, we say that the robot's feel [of the slope] is $\lceil |s| \rceil$. If the tangent line is parallel to $\{y = 0\}$, the feel is defined as ∞. The robot uses the feel of the slope in the following routine (where **line** of the form $\{y = \text{constant}\}$):

stabilize(**line**):
    repeat
        if not adjacent to an obstacle **then**
            move directly toward the **target**
        else if feel $> r$ **then**
            follow the obstacle toward the **target**
        else
            follow the obstacle toward **line**
    until you are at **line** and you are adjacent
            to an obstacle and feel $\leq r$

Now we can formulate the new attack procedure:

attack(i):
    repeat
        stabilize(**central**);
        if the adjacent obstacle extends from **lower** to **upper then**
            exit
        else
            follow the obstacle around its nearer end back to **central**
    forever

To analyze this algorithm, observe first that the obstacles that stop the attacks form essentially the same pattern as in the previous algorithm. Therefore we can conclude that when stage(l) is completed and the robot does not reach the **target**then the shortest obstacle-free path from the **source**to the **target**is longer than ml. However, stage(l) is not always completed: sometimes it is interrupted, because the robot already traversed the allowed distance of cm^4l. Nevertheless, as we prove below, the probability of such interruption is lower than 1/4.

Before we proceed, we should notice that the above two claims imply that the expected competitive ratio is $O(m^3)$. Indeed, if the shortest path from the **source**to the **target**has length L, then the stages with $l < L/m$ traverse less than $2cm^4L/m = 2cm^3L$ because of the interruptions; for the subsequent stages we should notice the following: while the allowed cost doubles, the probablity that the stage will be attempted at all (i.e. that the robot have not reached the **target**already) goes down by the factor 4; hence the [upper bound of the] contribution of such a stage to the expected path length is half of that for the previous stage.

The proof that with probability at least 3/4 the robot completes stage(l) is divided into four propositions. Each statement should be conditioned "during the execution of stage(l)".

Proposition 1. Each obstacle is encountered by the robot at most once. If both obstacle A and obstacle B are encountered, and A is above B (precisely: the corridor of A and B is non-empty, see the previous section), then A is encountered before B.

Sketch of proof. More or less obvious. However, if false, the subsequent reasoning makes no sense.

Proposition 2. The motion of the robot is restricted to the trapezoid with the following corners: $(-2m^3, \pm ml)$ and $(n, \pm(m+1)ml)$.

Sketch of proof. Assume that r was selected. Then the robot may move away from the **target**only along obstacles with feel at most r, and toward the **central** line of an attack; when it moves toward the **central** of the first attack from (0,0), it changes the y coordinate by less than ml, and consequently, it decreases the x coordinate by at most less than $(r/l)ml = rm \leq 2m^3$. Similarly, when the robot moves away from the lines $\{y = \pm ml\}$, it moves away from the **central** line of the current attack; in this case it must "slide" along an obstacle with feel greater than r. Say that the slide started at $(-rm, ml)$; as the x coordinate increases by at most $rm+m^4$, the y coordinate decreases by at most $(l/r)(rm+m^4) = (1+m^3/r)lm \leq (1+m)ml$.

Proposition 3. With probability at least 3/4, the joint length of the obstacles with feel r encountered by the robot is at most $6m^4l$.

Sketch of proof. First note that the joint length of the obstacles in the trapesoid of Proposition 2 is bounded by its area, which is $(1+o(1)m^5l < 1.5m^6l$. Because we choose r out of m^2 possibilities, the expected joint length of obstaces with feel r is less than $1.5m^4l$. Thus the claim follows from the Chebyshev inequality: probability of exceeding the expected value more than 4 times is at most 1/4.

Proposition 4. Assume that the joint length of obstacles with feel r encountered by the robot is at most $6m^4l$. Then the robot traverses at most 30^4ml.

Sketch of proof. With a little loss of precision, we may assume that all the movements of the robot are executed within stabilize and while going around obstacles in the attacks. Note that an execution of stabilize(**central**) that starts on **central** has the net effect of moving the robot toward the **target**(Proposition 1). Going around an obstacle has a similar effect. However, when we start an attack, the robot performs a stabilize that changes its y coordinate. The sum of such changes per phase is $4ml$, for the total sum of $4m^2l$. The associated sum of decreases of x coordinate is at most $(r/l)4m^2l < 8m^4$. Therefore, the executions of stabilize that start and end at the same **central** line, as well as going around obstacles, move the robot toward the **target**by at most $9m^4$. When the robot goes around an obstacle, it traverses at most $2l$ and gets closer to the **target**by at least 1. The movement in stabilize can be split into balanced zigzags of "toward" [the **target**] and "away" motions. If such a zigzag involves an obstacle with feel r, we cannot estimate its net progress, but the joint length of such zigzags is at most $12m^4l$. Otherwise, one part of the zigzag has feel it least $r+1$, while the other at most $r-1$. One can see that two such motions, each of length l, move the robot closer toward the **target**by at least 1. Hence the joint length of such zigzags and "going around" is at most $(9m^4)(2l) = 18m^4l$. Together, we get $30m^4l$.

5 Short obstacles

One reason that the last algorithm does not solve the problem of general convext obstacles is the reasoning needed in the proof of Proposition 3. There we used the following observation: the joint length of the obstacles inside the trapezoid bounding the robot's movements is limited by its area. This was true, because the obstacles were rectangles of width at least 1. If they are, say, elongated right triangles with short edge $1+\varepsilon$, then we need to double the joint length estimate. The real problem arises however with extremally long triangles that have only their ends in the trapezoid. The joint length of their portions included in the trapezoid is unbounded. We delay handling of this problem to the subsequent section, by assuming that the obstacles are contained in the trapezoid, in particular, they have length at most $2m^2l$.

Another reason could be that an obstacle does not have a unique feel. This is not a problem, however. First, it is important to estimate the length of *surfaces* with feel r, and of course each piece of the surface has its unique feel (with the exception of places without a tangent line, but these have measure 0). Secondly, once the robot decides on the direction to follow because of the feel, the changes of the feel will not change the direction. This follows from convexity of the obstacles: if we follow an obstacle away from the **target** because the slope (and, hence, the feel) is small, further away it will be the same or smaller. When we follow an obstacle toward the **target** because the slope is high, along the way the slope can only increase.

A more serious reason is that the following argument does not apply: if during an attack the robot goes around an obstacle, then it moves forward (closer toward the **target**) by at least 1. Now, it could merely walk around the narrow tip of an elongated obstacle.

Actually, in the latter case, we do not need to move forward by 1. The only needed condition is the following: if we go around an an end of an obstacle that is in the distance αl from the **central** line, we move forward by at least $\alpha/2$.

Still, even this may turn to be false. Let us call such a case "small forward". Recall that each obstacle contains a unit circle and is convex. Hence, if the robot experiences "small forward", it knows that the adjacent obstacle has length at least $2l$. Such obstacles can be incorporated into the phase barrier, generalizing the concept used before. We apply this idea to change the attack as follows:

attack(i):
  $C := 0$;
  repeat
    stabilize(**central**);
    if the adjacent obstacle extends from **lower** to **upper then**
      exit
    else
      follow the obstacle around its nearer end back to **central** ;
      if "small forward" **then**
        add the distance to the nearer end (in
        y-coordinate) to C; move to this end;
        reset **central** to the current y coordinate
  until $C > l$

Clearly, the new attack assures that the sum of distances traversed around obstacles with "small forward" is at most $4l$ per attack, and less than $4m^2l$ in the entire stage. This extra cost is negligible. We also have to double the estimate of the joint length of the obstacles with feel r, (detail of this estimate in the next section), and half the estimate of the "efficiency" of going around obstacles. Therefore we can prove that stage(l) can be completed, with probability 3/4, by traversing $O(m^4l)$.

What remains is to show that if stage(l) us completed and the **target** not reached, then no obstacle-free path from the **source** to the **target** exists. Now the phase barriers consist not only the obstacles that stopped attacks, but also all obstacles that were traversed with "small frward". Now to cross a barrier we have an option of either following a corridor of length at least l, or to go through a zigzag of short corridors; however our use of the counter C in the loop condition assures that the lengths of such short corridors must add to l.

6 Arbitrary convex obstacles

Now we show how to modify the attack and the analysis to take care of the fact that there is no limit on the joint length of the fragments of obstacles that are inside the "reachable trapesoid". To address this problem, we need first to analyze for which obstacles we needed such an estimate.

Our estimate on the length of path traversed along surfaces with feel different than r did not depend on their joint length; similarly we did not need that argument for the obstacles encountered at the end of stabilize procedure. In the remaining case—obstacles encountered within stabilize and with feel r—we can change the procedure without changing the analysis performed so far. Our change is simple: when the robot encounters such an obstacle, it flips a fair coin, and chooses one of the two alternative directions accordingly (of course, it may happen that these two directions—toward the **target** and toward the **central** line—coinside).

The first benefit is that we handle very well the obstacles that cross the **target**: if we encouter such, with 50% chance we reach the **target**, thus the average path length travesed along such obstacles is at most m^2l (twice the maximal lenght of a single traverse).

Amound the remaining obstacles, those that do not cross the line $\{x = -m^3\}$ do not cause major problems, because the joint length of their surfaces that has feel in the range from m^+1 to m is $O(m^4 l)$ (a detailed proof will be provided in the full version). Even simpler geometric argument applies to the obstacles that cross neither $\{y = -m^2 l\}$ nor $\{y = m^2 l\}$.

If we encouter one of the remaining obstacles outside the strip $|y| < ml$ do not cause problems either: either the robot follows such an obstacle simultaneously toward the **central** and **target**lines, which does not pose any problem in the analysis, or this is the last obstacle of the stage, as it will end all the remaining attacks. If we encounter such an obstacle within the strip, but outside the current corridor of attack, we have 50% chance of ending the attack, while traversing less than $2ml$; the joint length of such traverses in a stage is less than $4m^3 l$.

Thus we may conclude with the following theorem.

Theorem. There exist a randomized navigation algorithm that allows the robot to reach a straight line in a distance n, separated by convex obstacles, each containing a unit circle, with competitive ratio $O(n^{3/4})$.

References

[1] E. Bar-Eli, P. Berman, A. Fiat and P. Yan, *On-line Navigation in a Room*, Journal of Algorithms 17, pp. 319-341, 1994.

[2] P. Berman, A. Blum, A. Fiat, H. Karloff, A. Rosen, M. Saks, *Randomnized Algorithm for Wall Problem*,

[3] A. Blum, P. Raghavan, and B. Schieber, *Navigation in Unfamiliar Terrain*, Proc. 23rd STOC, pp 494-504, May 1991.

[4] A. Fiat, D.P. Foster, H.J. Karloff, Y. Rabani, Y. Ravid and S. Vishwanathan, *Competitive algorithms for layered graph traversal*, Proc. 32nd FOCS, pp. 288-297, October 1991.

[5] H. Karloff, Y. Rabani, and Y. Ravid, *Lower Bounds for Server Problems and Motion Planning*,

[6] V.J. Lumelsky, A.A. Stepanov, *Dynamic path planning for a mobile automaton with limited information on the environment*, IEEE Transactions on Automatic Control, AC-31:1058-1063, 1986. submitted to 28th STOC, 1995. Proc. 23rd STOC, pp.278-288, May 1991.

[7] C.H. Papadimitriou and M. Yannakakis. *Shortest Paths Without a Map*, In *Proc. 16th ICALP*, pp 610–620, July 1989.