On-Line Navigation Through Weighted Regions

John Reif and Hongyan Wang
Department of Computer Science
Duke University
Durham, NC, 27708-0129 *

Abstract

This paper studies some extensions to the *wall problem* of Blum *et al.*. In the *wall problem*, a point robot is to reach an oriented infinite line while moving amid oriented retangular obstacles. In the extension called the *weighted wall problem*, we generalize an obstacle to be a weighted region of a constant weight while the weights of different regions my differ. The *weighted Euclidean distance* between two points inside a region is the product of the corresponding weight and the Euclidean distance between them. The objective is to reach the target line with a minimum weighted Euclidean distance. For this problem, we present an optimal on-line algorithm, which is a generalization of the *sweeping strategy* of Blum *et al.*. Using this strategy, the robot is guaranteed to reach the target while traveling a weighted Euclidean distance of no more than $O(\sqrt{n})$ times the optimal weighted Euclidean distance, where n is the distance between the start point and the target line.

*Supported by Grants NSF CCR-9725021, NSF IRI-9619647, NSF/DARPA CCR-92725021, and ARO contract DAAH-04-96-1-0448.

1 Introduction

The study of robotic motion planning falls into two categories. In *off-line* motion planning, the robot has complete information of the environment and plans out the paths before it moves. On the contrary, in *on-line* motion planning, the robot has no prior knowledge of the enviornment but has to gain such knowledge as it navigates and plans as it navigates. While almost all previous *theoretical* work (see [14, 17] and references therein) has focused on the off-line motion-planning problems, in recent years, on-line motion planning has begun to receive a lot of attention. On-line motion planning arises when a robot has to reach a goal or explore in an unknown environment.

In on-line motion planning problems, the goal is usually to minimize the Euclidean distance traveled by the robot to finish its task (e.g., reach a destination, draw a map). Because of its on-line nature, *competitive ratio* (introduced by Sleator and Tarjan [18]) is used as a measure of goodness of on-line algorithms. The competitive ratio is defined to be the *worst case* ratio of the performance (e.g., Euclidean distance) of the on-line algorithm to the optimal off-line performance (i.e., when all the information is known). An algorithm is *optimal* if the competitive ratio of the algorithm matches the lower bound on the competitive ratios of the problem.

While most of the previous work on on-line navigation focused on the problem of navigating through an unknown terrain with obstacles which are impenetrable (see [4, 16, 2]), it is quite interesting and practical to consider on-line navigation problems with regions that are penetrable but are associated with possibly different weights. A weight indicates *effort per unit length* upon traveling. The goal is to finish the task with a minimum total effort. An application of such models is terrain navigation in a field with different landscapes like grass land, lakes, swamps and hills. The landscapes are penetrable, but require different effort per unit length.

In particular, this paper studies some extensions to the *wall problem*, first studied by Blum *et al.* [4]. It is termed as *wall problem* because the target is an oriented infinite line. The robot is a point moving on the plane amid oriented rectangular obstacles. For this problem, Blum *et al.* [4] gave an on-line algorithm called the *sweeping strategy* with a competitive ratio of $O(\sqrt{n})$, where n is the Euclidean distance from the starting point to the target line. Since Papadimitriou and Yanakakis [16] showed that the lower bound on the competitive ratio for this problem is $\Omega(\sqrt{n})$, the *sweeping strategy* is optimal. A randomized algorithm given by Berman *et al.* [3] achieved a better competitive ratio of $O(n^{4/9} \log n)$. However, there is still a big gap between this upper bound and the known lower bound for randomized *wall problem* given by Karloff *et al.* [11]. Even though the scenario is simple, the *wall problem* is very interesting in that it embodies many of the key issues of on-line navigation. Also, the solution to this problem, together with that for the *room problem* [4], can provide a solution to a more general point-to-point navigation problem.

1.1 Our model and results

We consider the robot as a point moving on a plane. The robot is *tactile* which means that it senses by touching. For example the robot can only learn the contour of a region by touching every point on the boundary.

The plane is divided into regions. Each region has a constant weight associated with it, which is the *effort per unit length* upon traveling through the region. If the robot travels in a region of weight ρ for a distance of d, then we define the *effort* which it spends to be the *weighted Euclidean distance* ρd. The objective is to travel to the target with a minimum total effort.

The robot does not know the positions, the extents, or the weights of the regions; rather, it finds out about a region as it encounters it. We assume that the robot can learn the weight of a region once it touches it.

Let $e_A(S)$ be the total effort spent by a robot to reach the target in an obstacle situation S using strategy A, and let $e_O(S)$ be the minimum effort. We use as the figure of merit for a strategy the ratio

$$\max_{S \in \mathcal{S}} \frac{e_A(S)}{e_O(S)},$$

where $\mathcal{S}$ is the collection of all obstacle situations concerned.

In this paper, we first give an extension to the *wall problem*, called the *weighted wall problem* (see Section 2). In the original *wall problem*, the robot is to reach an oriented infinite line while moving amid oriented rectangular obstacles which are impenetrable. We generalize an obstacle in the *wall problem* to a region of a constant weight, while the weights for different regions may differ. The objective is to reach the target line with a minimum effort. For this problem, we give an optimal on-line algorithm which is a generalization of the *sweeping strategy*. We show that the competitive ratio of our algorithm is $O(\sqrt{n})$ and matches the lower bound, where n is the Euclidean distance from the start point to the target line.

Second, we make a further generalization such that a weighted rectangular region may contain rectangular regions of higher weights. We term this as the *recursive weighted wall problem* (RWWP) (Section 3). We show that the lower bound on the competitive ratio of the RWWP depends not only on the Euclidean distance between the start point and the target line, but also depends on an upper bound of the *weighted widths* of the weighted rectangles.

1.2 Other related work

Another related on-line navigation problem is called the Room Problem [4]. In this problem, the obstacles are oriented rectangles confined to lie within a square "room", and the target is a point in the room. An optimal algorithm was given by Bar-Eli *et al.* [2], whose competitive ratio is $O(\log n)$, where n is the size of the room.

There has also been work on other types of on-line navigation problems; some examples are the *searching* problem where the goal is to locate (instead of reaching) some target [5, 12, 13], and the *mapping* problem where the goal is to gain full information of an environment [6, 7, 8, 9].

Planning with weighted regions has been studied in an off-line setting by Mitchell and Papadimitriou [15].

2 The Weighted Wall Problem

The difference between the *weighted wall problem* and the original *wall problem* is that an obstacle in the *wall problem* is generalized to a region which is penetrable and of a constant weight. The weights of all such regions are larger than 1, while the weight of the free space is 1. An obstacle in the *wall problem* is a special case here with a region of weight ∞. For convenience, we will still refer to such regions in the *weighted wall problem* as *obstacles*.

As in the *wall problem*, the obstacles are oriented rectangles and the target is an oriented infinite line. The obstacles are disjoined. This implies that when two obstacles touch the robot can "squeeze" between them. The length of either side of an obstcle is at least 1.

Following the convention of [4], we put on obstacle scene in an xy-coordinate system. Let the start point s be at $(0, 0)$. Let the target line t be parallel to the y-axis and at distance n from s. Let the *width* of an obstacle be its dimension parallel to the x axis, and the *height* of an obstacle its dimension parallel to the y axis. We use "go north" (resp. "go south") when the robot goes along (resp. against) y axis. We use "east" to indicate the direction of the x axis.

2.1 The generalized sweeping strategy

We generalize the original sweeping strategy of [4] to adapt to the *weighted wall problem*. We also maintain four variables, a *window size* W, a *threshold* τ, a *sweep direction* which is either north or south, and a *sweep counter*. A sweeping window of size W is defined by the two window sides $y = \pm W/2$. Initially, W is set to n, the sweep direction is south, and the sweep counter is set to 0. The threshold τ is always set to $W/\sqrt{n}$. Whenever the sweep counter reaches $\sqrt{n}$, it is reset to 0 and the window size W and the threshold τ get doubled.

Let (x, y) be the point where the robot hits an obstacle. Let y_n and y_s be the y-coordinates of the north and south corners of the obstacle. Let w denote the width of the obstacle. Let E be the effort of going through the obstacle, arriving at $(x + w, y)$, and let N (resp. S) be the effort of traveling along the height of the obstacle, reaching the point (x, y_n) (resp. (x, y_s)). Even though the robot is tactile, by using the *doubling strategy* of Baeza-Yates *et al.* [1], the robot can find out which one is the minimum of E, N and S, by spending an effort of no more than $c\min(E, N, S)$ (see Section A), where $c = 4.5911$. Since c is a constant compared to the competitive ratio $\sqrt{n}$ that we want to show, we can assume that the robot knows which one is the minimum of E, N and S as it encounters an obstacle.

The generalized sweeping strategy works as follows. Starting from point s, the robot travels due east until it either reaches t or hits an obstacle. Suppose the robot hits an obstacle at point (x, y). Let y_n, y_s, w, E, N, S be the same as defined in the above paragraph. When the robot encounters an obstacle, the following three rules are applied.

Rule 1: If $\min(E, N, S) < \tau$, do the following. If E is the smallest of the three, the robot goes through the obstacle, arriving at $(x + w, y)$. Otherwise it travels along the boundary of the obstacle through the nearest corner, and arrives at $(x + w, y)$. It continues to travel due east until it hits another obstacle. (See Figure 1(a).)

Rule 2: If $y_n > W/2$, $y_s < -W/2$, and $E > W$, do the following. Without loss of generality, we can assume that $|y_s| < y_n$. If $E < S$, the robot goes through the obstacle, arriving at $(x + w, y)$. Otherwise, it travels along the height of the obstacle to the south corner, and then along the width of the obstacle to $(x + w, y_s)$. In the first case, the window size W is set to $2E$, and in the second case, it is set to $4|y_s|$. The sweep counter is set to 1 and the sweep direction is set to north. The robot continues east from where it arrives. (See Figure 1(b).)

Rule 3: Otherwise, do the following. Without loss of generality, we assume that the current sweep direction is south. If $E = \min(E, N, S)$, the robot goes through the obstacle to point $(x + w, y)$, and then travels south until it arrives at $(x + w, \max(y_s, -W/2))$. Otherwise, it goes along the boundary of the obstacle, through the nearest corner, until it arrives at $(x + w, \max(y_s, -W/2))$. In both cases, if the robot arrives at $(x + w, -W/2)$, the sweep counter is incremented by 1 and the sweep direction is flipped to north. The robot continues east from where it arrives. (See Figure 1(c).)

2.2 Analysis

Lemma 1 *The total effort spent by the robot is $O(W_f\sqrt{n})$, where W_f is the final window size.*

Proof: We divide the effort into three components: (1) the effort of traveling along the horizontal segments in the free space, (2) the effort of traveling using Rule 1, and (3) the effort of traveling using the other rules, and bound each of the three comonents separately.

Since the robot never backtracks horizontally, the effort of traveling along the horizontal segments in the free space is bounded by $O(n)$. Let $\tau_f = W_f/\sqrt{n}$. Since the width of each obstacle is at least 1, the effort of traveling using Rule 1 is bounded by $O(n\tau_f) = O(W_f\sqrt{n})$. Therefore, to prove the lemma, it suffices to bound the third compopnent by $O(W_f\sqrt{n})$.

Fix a window size W, in one sweep, the effort of traveling using Rule 2 or 3 is bounded by $O(W)$. This is obviously true if the robot uses Rule 2. Otherwise, assume that the sweep direction is south. We can further decompose the effort into two parts: the forward-going effort – the effort of making progress toward south, and the backtracking effort, which includes the effort of going through obstacles and the effort of going around a north corner and arriving at a point which has the same y-coordinate as the point where the robot hits the obstacle. The total forward-going effort in a sweep is $O(W)$. If we can

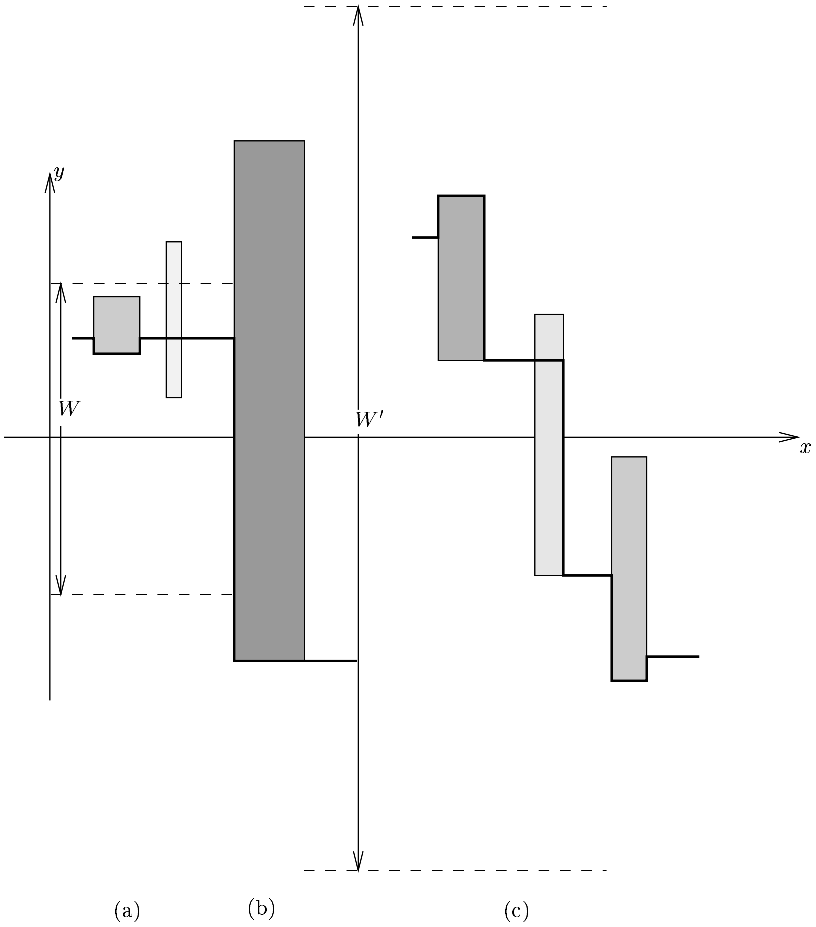

Figure 1: The three rules of the generalized sweeping strategy.

show that the backtracking effort can be charged to the forward-going effort by a constant factor, then the total effort spent in a sweep is bounded by $O(W)$. We will show for the case when the robot goes through an obstacle using Rule 3. The other cases can be dealt with in a similar manner. If $y_s > -W/2$, the robot travels south to point $(x+w, y_s)$ after it goes through the obstacle to point $(x+w, y)$. Thus after if spends an effort of E, it makes progress toward south by an amount of S. Since $E < S$ (that's why the robot chooses to go through the obstacle), we can charge E to S. If $y_s < -W/2$, the robot travels south to point $(x+w, -W/2)$ after it goes through the obstacle to point $(x+w, y)$. In this case, E may be larger than the effort of progressing toward south. But notice that $E < N < W$ (otherwise the robot has to use Rule 2), and this case happens only once in a sweep. Thus the total effort of going through obstacles in a sweep is bounded by $O(W)$.

Since the robot does at most $\sqrt{n}$ sweeps for each window size W, the total effort spent for a fixed window size is $O(W\sqrt{n})$. The window size is at least doubled each time it is changed, thus the total effort for traveling using Rule 2 and Rule 3 is also $O(W_f\sqrt{n})$. This completes the proof of the lemma. □

Lemma 2 *The minimum effort required to travel from s to t is $\Omega(W_f)$, where W_f is the final window size.*

Proof: Since the Euclidean distance from s to t is n, the minimum effort is obviously $\Omega(n)$. If $W_f \le n$, then we are done. Otherwise, we prove that the minimum effort is $\Omega(W_f)$.

Suppose that the robot has completed $\sqrt{n}$ sweeps for some window sizes, and that W is the largest such window size. We claim that the minimum effort is $\Omega(W)$. The minimum-effort path may either be going a vertical distance of $\Omega(W)$ to go around all the obstacles, or be cutting through each sweep. The minimum effort of cutting through one sweep is $\Omega(\tau)$, where $\tau = W/\sqrt{n}$.

Comment 1 *To see this, let's assume that the current sweep direction is south and O_i is the first obstacle encountered by the optimal path in this sweep. For simplicity, we assume here that N is the minimum of E, N and S for every obstacles encountered. Let y_o be the y-coordinate of the optimal path right before it hits O_i, and y_r the y-coordinate of the robot before it hits O_i (See Figure 2). Obviously, $y_r > y_o$. Let y_i be the y-coordinate of the north corner of O_i. If $y_i - y_o > \tau$, then our claim holds. Otherwise, the y-coordinate of the robot after it overcomes obstacle O_i is still y_r, because it must have applied Rule 1 to this obstacle. If later obstacles encountered by the line $y = y_r$ all have their north corners lie below the line $y = y_r + \tau$, then the y-coordinate of the robot remains to be y_r. But we know that the robot has made a sweep to the south, so there exits an obstacle whose north corner lies above $y = y_r + \tau$. To overcome this obstacle, the optimal path has to go a vertical distance of at least $y_r + \tau - y_o > \tau$, thus spend an effort of $> \tau$. This proves our claim.*

To see this, let's consider the obstacles met by the robot during one sweep. Without loss of generality, we can assume that the sweep direction is south. If we omit all the obstacles to which the robot applied Rule 1, and omit the horizontal free space between two successive obstacles met by the robot, then we obtain a "barrier" similar to the one shown in Figure 2. Initially, from the north window side, to overcome the first obstacle in the sweep takes an effort of at least τ. Then from the south-east corner of an obstacle, to overcome the next obstacle takes an effort of at least τ. Otherwise, the robot should have applied Rule 1 to the obstacle. This barrier stretches at least τ over both window sides. To cut through this sweep, the minimum-effort path needs to overcome one of these obstacles. However, to overcome any one of them takes at least τ effort. Thus the effort of cutting through all the $\sqrt{n}$ sweeps is $\Omega(\sqrt{n}\tau) = \Omega(W)$.

If $W_f \le 2W$ then we are done. Otherwise (i.e. $W_f > 2W$ or there is no window for which the robot has completed $\sqrt{n}$ sweeps), then it must be the case that W_f is determined by Rule 2, in which case the minimum effort is clearly $\Omega(W_f)$. □

Combining Lemma 1 and Lemma 2, we obtain

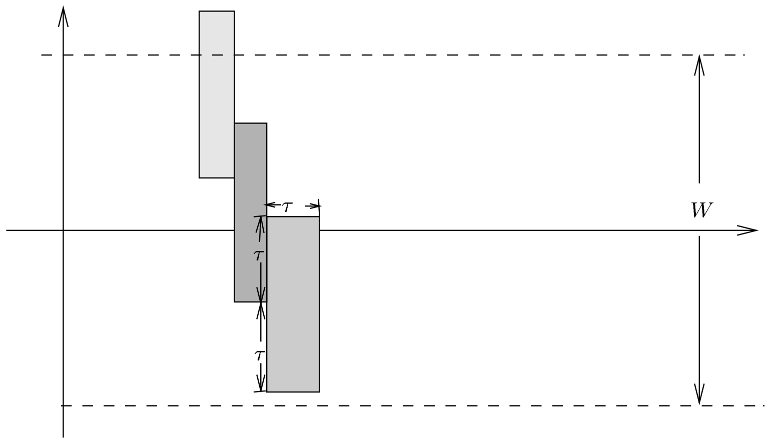

Figure 2: To cut through one sweep, the minimum effort is $\Omega(\tau)$.

Theorem 3 *The competitive ratio of the generalized sweeping strategy is bounded by* $O(\sqrt{n})$.

2.3 The lower bound

Lemma 4 *Any on-line algorithm for the weighted wall problem has a competitive ratio of* $\Omega(\sqrt{n})$.

Proof: We prove the lemma by an "adversary" argument, similar to the one given in [16]. The adversary builds a scene from a set of identical obstacles. Each obstacle has a height of n, a width of 1, and a density of n.

Once the robot overcomes an obstacle, the adversary always puts another obstacle right in front of the robot, such that the robot is in the middle of the obstacle's height. Thus no matter which way the robot chooses to overcome an obstacle, the effort is $\Omega(n)$. Since there are n obstacles, the total effort spent by the robot to get to the goal is $\Omega(n^2)$.

On the other hand, by the pigeon hole theorem, either all the obstacles lie within the strip bounded by $y = \pm n^{3/2}$, or there is a line $y = k$, where $-n^{3/2} \leq k \leq n^{3/2}$, such that the line passes at most $n^{1/2}$ obstacles. In the first case, the robot can go a vertical distance of $n^{3/2}$ to go around all the obstacles. In the second case, the robot follows the line $y = k$, encountering $\sqrt{n}$ obstacles and spending a total effort of $O(n^{3/2})$ to overcome them. Either way, the effort spent by the adversary is at most $O(n^{3/2})$.

Thus the competitive ratio is lower bounded by $O(\sqrt{n})$. □

Corollary 5 *The generalized sweeping strategy is an optimal on-line algorithm up to a constant factor for the weighted wall problem.*

3 The Recursive Weighted Wall Problem

3.1 The model

We extend the model of the wall problem with penetrable obstacles to include penetrable obstacles which contain obstacles of higher densities within.

The *depth* of an obstacle is defined recursively as follows. An obstacle which is placed immediately inside the free space has a depth of 0. An obstacle which is placed immediately inside an obstacle of depth i has a depth of $i + 1$. The *level of recursion* of a scene is R if the depth of the deepest obstacle is R. If an obstacle has a width of w and a density of d, then the *expanded Euclidean distance* of this obstacle is dw.

In the following, we let n denote the Euclidean distance from the source point to the wall. And let n_i denote the upper bound on the expanded Euclidean distances of the obstacles of depth i.

The following section gives a lower bound on the Recursive Wall Problem when $R = 1$. The result can be generalized to higher recursive level. The difficulty in giving an upper bound is described in Section 4.

Comment 2 .

3.2 The generalized algorithm and its analysis

The sweeping strategy still applies, but to get the correct bounds, the variables are set differently. At the outermost level, we still maintain the four variables, the window size W, the threshold τ, the sweep direction and the sweep counter. The window size is initialized to n. The threshold τ is always set to $W{n_0}^{1/4}/n^{1/2}$. The window size is doubled if the sweep counter reaches $n^{1/2}{n_0}^{1/4}$, i.e., if the robot make $n^{1/2}{n_0}^{1/4}$ sweeps for a fixed window size.

A 0-level obstacle is called a big obstacle *with respect to a window size W, if the obstacle extends past both window sides, and $E > W$, where E is the robot's effort of going through the obstacle, by applying the sweeping strategy to the 1-level obstacles within the 0-level obstacle. We maintain an additional counter that counts the number of big obstacles encountered during a fixed window size. Let $O_1, \ldots, O_k$, where $k = {n_0}^{1/2}$ be the big obstacles encountered during a fixed window size. Let E_i be the robot's effort of going through obstacle O_i. If the counter reaches ${n_0}^{1/2}$ before the sweep counter reaches $n^{1/2}{n_0}^{1/4}$, the window size is set to $2W_{\min}$.*

If the robot meets a 0-level obstacle, it probes in three ways, i.e., north, south and east, to find an efficient way of getting around this obstacle. To find a path going through the obstacle can be considered as a weighted wall problem, *and the generalized sweeping strategy can be applied. However, in this case, the robot may not find the optimal path going through an obstacle. The consequence is that even for one obstacle, the effort spent by the robot is not within a constant factor of the minimum effort. If the expanded Euclidean distance of this obstacle is n', then the robot is guaranteed to find a path that takes effort no more than $\sqrt{n'}$ times the minimum effort. I found that this claim which is one of the foundations of the proof is not ture. It is ture only when the robot and the adversary starts from the same point going through an obstacle. This is implicit in the weighted wall problem since the starting point is s. Now it's like the robot starts at point s while the adversary can start at $(0, y)$ with an arbitrary y. Starting at different points, how to define the minimum effort of overcoming an obstacle? Also notice that in order to achieve this bound, the robot needs to know the expanded Euclidean distance of this obstacle.*

Lemma 6 *The competitive ratio of the algorithm is $O(\max(n^{1/2}{n_0}^{1/4}, {n_0}^{1/2}))$ when $R = 1$.*

Proof: *We will prove this by induction on the window size W. Let $e(W)$ (resp. $e_O(W)$) be the effort spent by the robot (resp. the minimum effort) up to setting the window size to be W. We will show that $e(W)/e_O(W) = O(\max(n^{1/2}{n_0}^{1/4}, {n_0}^{1/2}))$, and $e_O(W) = \Omega(W)$.*

The base case is when $W = n$. We can assume that $e(n) = n$ and $e_O(n) = n$, because the effort to travel from s to t is at least n. Thus the hypothesis holds for the base case.

Assume that the hypothesis holds for window size W, and let W^+ be the window size immediately after W. We need to show that the hypothesis also holds for W^+. The case when W is the largest window size (i.e., the target is reached before the window size is changed again) can be dealt with in a similar manner.

The effort spent by the robot using Rule 1 is bounded by

$$O(n\tau) = O(nW{n_0}^{1/4}/n^{1/2}) = O(Wn^{1/2}{n_0}^{1/4}).$$

We can charge this effort to the previous minimum effort. Since $e_O(W) = \Omega(W)$, the ratio is at most $O(n^{1/2}{n_0}^{1/4})$.

Before the window size is changed, the robot may make $m \leq n^{1/2}{n_0}^{1/4}$ sweeps, spending an effort of $O(mW)$. To cut through the m sweeps, the optimal path may either go a vertical distance of $\Omega(W)$ to go around all the obstacles, or cut through each sweep. In the first case, the minimum effort is obviously $\Omega(W)$. To cut through a sweep, the minimum effort is $\Omega(\tau/{n_0}^{1/2})$ (notice that the obstacles can be arranged so that the optimal path always goes through the obstacles and gains a factor of $1/{n_0}^{1/2}$ over the robot). Thus the total effort of cutting through m sweeps is $\Omega(m\tau/{n_0}^{1/2})$. In either case, to make m sweeps, the ratio of the effort spent by the robot to the minimum effort is bounded by $O(n^{1/2}{n_0}^{1/4})$. If the window size is doubled, i.e., $W^+ = 2W$, then $e_O(W^+) = \Omega(W) = \Omega(W^+)$.

The robot may meet $k \leq {n_0}^{1/2}$ long obstacles before the window size is changed. If the optimal way is to go a vertical distance of $\Omega(W_{\max})$ to go around all the obstacles, spending an effort of $\Omega(W_{\max})$, then the ratio is bounded by $O(k) = O({n_0}^{1/2})$. If the optimal way is to go through each long obstacle, then for each long obstacle, the ratio is $O({n_0}^{1/2})$. If the window size is changed to $W_{\min}$, then obviously, choosing either optimal way, $e_O(W^+) = \Omega(W_{\min}) = \Omega(W^+)$.

Combining all the cases above, it is easy to see that $e(W^+)/e_O(W^+) = O(\max(n^{1/2}{n_0}^{1/4}, {n_0}^{1/2}))$.

By induction hypothesis, the lemma holds. □

3.3 The lower bound

Lemma 7 *When $R = 1$, the lower bound of the competitive ratio is*

$$\begin{cases} n^{1/2}{n_0}^{1/4} & \text{if } n_0 \leq n^2, \\ {n_0}^{1/2} & \text{otherwise.} \end{cases}$$

Proof: The adversary can build a scene using a set of identical 0-level obstacles. Each obstacle has a height of n_0^2, a width of 1, but a density of n_0 (thus the expanded Euclidean distance of each 0-level obstacle is n_0). Using a construction similar to the proof of the lower bound in Lemma 4, the adversary can construct 1-level obstacles within a 0-level obstacle in such a way that the minimum effort of going through a 0-level obstacle is $O({n_0}^{3/2})$, while any on-line algorithm has to spend an effort of $\Omega({n_0}^2)$ (notice that this is possible even though the height of the 0-level obstacle is limited – a point different from the situation in Lemma 4).

No matter which way the robot chooses to overcome a 0-level obstacle, the adversary puts another 0-level obstacle immediately in front of it. Thus the total effort the robot has to spent is $\Omega(n{n_0}^2)$.

By the pigeon hole theorem, the optimal path may be either going a vertical distance of $n{n_0}^2/x$ to go around all the obstacles, spending an effort of $O(n{n_0}^2/x)$, or following a path that hits at most x obstacles, spending an effort of $O({n_0}^{3/2})$ on each and $O(x{n_0}^{3/2})$ in total. The maximum of $\min(x{n_0}^{3/2}, n{n_0}^2/x)$ is achieved when $x = n^{1/2}{n_0}^{1/4}$. Since $1 \leq x \leq n$, when $n_0 \leq n^2$, $x = n^{1/2}n_0^{1/4}$ and when $n_0 > n^2$, $x = n$. The effort spent by the adversary is $O(n^{1/2}{n_0}^{7/4})$, when $n_0 \leq n^2$, and $O(n{n_0}^{3/2})$ when $n_0 > n^2$. Thus the competitive ratio is lower bounded by $\Omega(n^{1/2}{n_0}^{1/4})$ when $n_0 \leq n^2$, and $\Omega({n_0}^{1/2})$ otherwise. □

4 Discussions

If first seems that we can give an upper bound for the recursive wall problem, by applying the generalized sweeping strategy recursively to overcome each 0-level obstacles. We need to guarantee that for each 0-level obstacle, the robot spends an effort no more than $O(\sqrt{n_0})$ times the optimal effort. However, the upper bound of $O(\sqrt{n_0})$ applies only when the robot and the adversary enter the obstacle at the same point. If they enter an obstacle at different points, it is obvious that the adversary can arrange the 1-level obstacles within this obstacle in such a way that it takes the robot much more effort to go through than it does the adversary. Because of this difficulty, we can not obtain an upper bound by applying the generalized sweeping strategy.

It becomes an intriguing variation of the Wall Problem with Penetrable Obstacles if we also allow flows within obstacles. The flows have velocities along the y-axis and the robot can be carried away by the flow while it is exploring the interior of the obstacle. The velocity of the flow inside an obstacle is known to the robot as soon as it touches the obstacle. We don't know if the sweeping strategy can be used here.

If the obstacles are general polygons and if the general polygons can be encircled in disjoint rectangles (or rectilinear polygons in 3D), the problem can be converted to the Wall Problems studied above. Otherwise it seems that the sweeping strategy does not work for general polygons because in case of general polygons, the robot sometimes has to go backwards and the length of the path going backwards is not bounded.

References

[1] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Information and Computation*, 106:234–252, 1993.

[2] E. Bar-Eli, P. Berman, A. Fiat, and P. Yan. On-line navigation in a room. In *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms*, pages 237–249, 1992.

[3] P. Berman, A. Blum, A. Fiat, H. Karloff, A. Rosen, and M. Saks. Randomized robot navigation algorithms. In *Proc. 7th ACM-SIAM Symp. on Discrete Algorithms*, pages 75–84, 1996.

[4] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 494–504, 1991.

[5] A. Datta and C. Icking. Competitive searching in a generalized street. In *Proc. 10th Ann. ACM Symp. on Computational Geometry*, pages 175–182, 1994.

[6] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 298–303, 1991.

[7] B. Kalyanasundaram and K. Pruhs. Visual searching and mapping. In *DIMACS Workshop on Online Algorithms*, 1991.

[8] B. Kalyanasundaram and K. Pruhs. A competitive analysis of algorithms for searching unknown scenes. *Comput. Geom. Theory Apply.*, 3(3):139–155, 1993.

[9] B. Kalyanasundaram and K. Pruhs. Constructing competitive tours from local information. In *Proc. 20th Int. Colloquium on Automata, Languages and Programming*, pages 102–113, 1993.

[10] M. Kao, J. Reif, and S. Tate. Searching in an unknown environment: an optimal randomized algorithm for the cow-path problem. In *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms*, 1993.

[11] H. Karloff, Y. Rabani, and Y. Ravid. Lower bounds for randomized k-server and motion-planning algorithms. *SIAM J. on Computing*, 23:293–312, 1994.

[12] R. Klein. Walking in an unknown street with bounded detour. *Computational Geometry: Theory and Applications*, 1:325–351, 1992.

[13] J. Kleinberg. On-line search in a simple polygon. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, pages 8–15, 1994.

[14] J.C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Norwell, MA, 1991.

[15] J.S.B. Mitchell and C. Papadimitriou. The weighted region problem. In *Proc. 3rd Annual Symposium on Computational Geometry*, pages 30–38, 1987.

[16] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. In *Proc. 16th Int. Colloquium on Automata, Languages and Programming*, pages 610–620, July 1989.

[17] J.T. Schwartz and M. Sharir. Algorithmic motion planning in robotics. In J.V. Leeuwen, editor, *Algorithms and Complexity, Volume A of Handbook of Theoretical Computer Science*, chapter 8, pages 391–430. Elsevier, Amsterdam, 1990.

[18] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28:202–208, 1985.

A The Cow-Path Problem

The cow-path problem is as follows. Consider a cow, standing at a crossroad s with d paths leading off into an unknown territory. On one of the paths there is a grazing field (the target t) at distance n from the intersection, and all of the other paths go on forever. The cow won't know that she has found the field until she is standing in it. Without prior knowledge of the paths, the problem is to find the grazing field while traveling the least distance possible. A simple deterministic algorithm (and also an optimal one), called the *doubling strategy*, given in [1], goes as follows. Label the d paths as $1, \ldots, d$. In one trial, the cow goes along one of the paths for a certain distance and returns to point s if the target is not found. In the ith trial, the cow tries the $(imodd)$-th path with a distance of 2^{i-1}. The cow travels in this fashion until the grazing field is found.

The competitive ratio of this strategy, i.e. the worst case ratio of the path length traveled by the cow to the shortest path length (n in this case) is $2\frac{d^d}{(d-1)^{(d-1)}} + 1$. When d is fixed, the ratio is a constant and is the best possible for any deterministic algorithms.

Kao *et al.* [10] designed a randomized algorithm, called the cow-path algorithm. Randomization is used only to pick a random permutation of the order of paths and a random "initial search distance" at the very beginning of the search. The expected performance of the randomized algorithm was shown to be twice as good as the deterministic algorithm and is optimal for $d = 2$. They conjecture that the algorithm is also optimal for $d \geq 3$.

The robot in the *weighted wall problem* can apply this *doubling strategy* when it encounters an obstacle. In this case, $d = 3$. It guarantees to overcome an obstacle with an effort of no more than $O(1)$ times the minimum effort overcoming the obstacle.