TECHNICAL REPORT NO. 239

Searching with Uncertainty

by

Ricardo A. Baeza-Yates, Joseph C. Culberson
and Gregory J. E. Rawlins

February, 1988

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY
Bloomington, Indiana 47405-4101

TECHNICAL REPORT NO. 239

Searching with Uncertainty

by

Ricardo A. Baeza-Yates, Joseph C. Culberson
and Gregory J. E. Rawlins

February, 1988

COMPUTER SCIENCE DEPARTMENT
INDIANA UNIVERSITY
Bloomington, Indiana 47405-4101

Searching with Uncertainty

by

Ricardo A. Baeza-Yates
Dept. of Computer Science
University of Waterloo
Waterloo, Ontario
N2L 3G1, Canada

Joseph C. Culberson
Dept. of Computing Science
University of Alberta
Edmonton, Alberta
T6G 2E7, Canada

and

Gregory J. E. Rawlins
Computer Science Department
Indiana University
Bloomington, Indiana 47405, USA

TECHNICAL REPORT No. 239

Searching with Uncertainty

By

Ricardo A. Baeza-Yates, Joseph C. Culberson
and Gregory J.E. Rawlins

February, 1988

Searching with Uncertainty

Ricardo A. Baeza-Yates * Joseph C. Culberson †
Gregory J. E. Rawlins ‡

Abstract

In this paper we initiate a new area of study dealing with the best way to search a possibly unbounded region for an object. The model for our search procedures is that we must pay costs proportional to the distance of the next probe position relative to our current position. This model is meant to give a realistic cost measure for a robot moving in the plane. Also we examine the effect of decreasing the amount of *a priori* information given to a class of search problems.

Problems in this class are very simple analogues of non-trivial problems on searching with bounded error, searching an unbounded region, processing digitized images, robot navigation and optimization.

We show that for some simple search problems, the relative information of knowing the general direction of the goal is much higher than knowing the distance to the goal.

1 Introduction

The problems we consider in this paper were suggested by general problems in graph searching, finding optimal paths, robotic navigation, and boundary detection in digital images ([3]). All problems are of the following form:

*Supported by Natural Sciences and Engineering Research Council Grant No. A-3353, the Institute for Computer Research at Waterloo and the University of Chile. Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L3G1, Canada. email: rabaeza%watdaisy@waterloo.csnet

†Supported by Natural Sciences and Engineering Research Council Grant No. A-8053. Room 3-38 Assiniboia Hall, Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G2E7, Canada. email: joe@alberta.uucp

‡Supported by Natural Sciences and Engineering Research Council Grant No. A-5692 and the Institute for Computer Research at Waterloo. Department of Computer Science, 101 Lindley Hall, Indiana University, Bloomington, IN 47405, USA. email: rawlins@iuvax.cs.indiana.edu

Suppose that we are searching for an object in some space under the restriction that for each new "probe" we must pay costs proportional to the distance (under some chosen metric) of the probe position to our current probe position. This is meant to model the cost in real terms of a robot (or human) searching for an object when the mobile searcher must move about to find the object. For example, when searching a graph or maze we usually assume that we have some representation of the maze or graph. Suppose that we do not, as is quite reasonable if we wish to have a robot explore a maze or if we are playing a computer maze game. How can we best walk the maze? Let us abstract this problem to the following very simple one.

Consider searching for a line in the plane starting at the origin. There is an oracle who knows where the line is and who is willing to sell you information about the line.

The oracle can tell you:

- The line's position (that is, two points on the line).
- That the line's orientation is in the set $\{\theta_1, \theta_2, \ldots, \theta_n\}$.
- That the line's distance (under some metric) from you is at most n units.
- Any combination of the above.
- Nothing.

What should the oracle charge you for the information?

In general, we are at the origin in d-dimensional space and we wish to search for a k-dimensional body. There is an oracle willing to sell us information about various parameters of the body at various prices. What information, if any, should we buy?

For example, suppose that we are searching for a 1 dimensional point (a line) in a 2 dimensional space. Suppose that the line is distance n steps away from the origin.

- If we are given a normal to the line then we can find the line in n steps.
- If we are given the line's distance and orientation then we can find the line in $3n$ steps.

- If we are given the line's distance and that the line orientation is one of two possible then we can find the line in $(3/\sin(\theta/2))n$ steps where θ is the angle difference of the two orientations. (If the orientations are orthogonal then we can do this in $3\sqrt{2}n \approx 4.242n$ steps.)
- If we are only given the line's distance then we can find the line in $1 + \sqrt{3} + 7\pi/6 \approx 6.392n$ steps.
- If we are only given the line's orientation then we can find the line in $9n$ steps.
- If we are only given that the line orientation is one of two possible then we can find the line in $\leq 13.02n$ steps.
- If we know nothing at all then we can find the line in $\leq 13.49n$ steps.

(Most of these scenarios appear in this paper but under different guises.)

2 The Lost Cow Problem

A cow comes to an infinitely long straight fence. The cow knows that there is a gate in the fence, and she wants to get to the other side. Unfortunately, she doesn't know where the gate is located. Assume that the gate is positioned an integer number of steps away from the cow and that the cow can only recognize the gate when directly upon it. How can she optimally find the gate?

If the cow knows that the gate is exactly n steps away then the obvious algorithm is also optimal: Go left for n steps then turn and go right for $2n$ steps. Consider the last time we turn. Suppose without loss of generality that we were going left and now we are turning for the last time and heading right. Since it is the last time we are heading right then we must have explored all n steps to the left. Also, since we are heading right then we must have not explored all n steps on the right. The adversary then places the gate n steps to the right. This forces us to take $3n$ steps.

2.1 Gate Arbitrarily Far Away

Now suppose the cow knows that there is a gate somewhere *but she does not know how far*. What is the minimum number of steps she must make to find the gate as a function of the actual (unknown) distance to the gate?

If the gate is n steps away then we show that the cow has a $9n$ step algorithm and that this is optimal up to lower order terms.

Geometric Search Algorithm: Choose a constant k and some constants a_l, a_r, b_l, b_r, c_l and c_r. We execute cycles of steps as before except that now the functions are $l_i = a_l k^i + b_l i + c_l$ for the left branch and $r_i = a_r k^i + b_r i + c_r$ for the right branch.

There are only two possibilities for the worst case position of the gate. Either the gate is at a distance $n = l_i + 1$ to the left for some i, or the gate is at a distance $n = r_i + 1$ to the right for some i.

If the gate is at a distance $n = l_i + 1$ to the left then the total distance walked is

$$2\sum_{j=1}^{i} l_j + 2\sum_{j=1}^{i} r_j + l_i + 1 =$$

$$2k(a_l + a_r)\frac{k^i - 1}{k-1} + (b_l + b_r)i(i+1) + 2(c_l + c_r)i + l_i + 1$$

Which is of order

$$\left(1 + 2\frac{k}{k-1}\frac{a_l + a_r}{a_l}\right) n$$

A similar result is obtained when the gate is placed on the right. In this case the distance walked is of order

$$\left(1 + 2\frac{k}{k-1}\frac{a_l k + a_r}{a_r}\right) n$$

The best choice for a_l and a_r is such that the result is the same in both sides. This gives $a_r = a_l\sqrt{k}$. Now, the factor in the order n term is

$$1 + 2\frac{k}{k-1}(1 + \sqrt{k}) = 1 + 2\frac{k}{\sqrt{k} - 1}$$

This factor has its minimum value of 9 for $k = 4$. A similar approach for the lower order terms gives: $b_r = -b_l$ and $c_r = -c_l$. Finally, minimizing the resultant expression we obtain that $c_l = 0$, $b_l = 0$. a_l must be chosen such that l_i and r_i are integers for all i and the constant term is minimized. These two conditions force a_l to be 1.

Hence, the best geometric algorithm is $l_i = 4^i$ and $r_i = 2 \cdot 4^i$ (up to lower order terms). This means that the algorithm may be described by the following simple function:

$$f_i = 2^{i+1} \quad \forall i \geq 1$$

where the odd terms are the steps to the left and the even terms are the steps to the right. The total distance walked in this case is no more than $9n - 16$ steps (for $n > 2$).

We conjecture that no algorithm which has cycle lengths growing faster or slower than some polynomial times a constant raised to the i^{th} power can give a constant ratio.

Interestingly, if we require all the worst case ratios to be *equal* to some constant then we get that $l_i = (2i-1)4^i$ and $r_i = i4^{i+1}$ and the constant is exactly 9. This algorithm may be described by the simple function:

$$f_i = i2^{i+1} \quad \forall i \geq 1$$

THEOREM **2.1** *Geometric Search is optimal up to lower order terms.*

Proof: Any algorithm to solve the cow problem can be described as a function of steps f_i from the origin, where the odd terms are the steps to the left and the even terms are the steps to the right.

If the algorithm is to find the gate then f must be such that $f_1 \geq 1, f_2 \geq 1$ and f is a strictly increasing function in i for i even and for i odd.

That is,

$$f_i \geq f_{i-2} + 1 \quad \forall i \geq 1 \quad \text{where } f_{-1} = f_0 = 0$$

Note that there is no *a priori* relation between f_i and f_{i+1} although we intuitively expect them to be equal. This intuition turns out to be incorrect. The best we can say is that *if* we are to beat the $9n$ bound then we must have that

$$f_{i+1} < 3f_i \quad \forall i \geq 1$$

This implies that

$$f_i < 3^{i-1} + 4 \quad \forall i \geq 1$$

It may be that careful consideration of this case can lead to a $9n$ lower bound by contradiction.

Suppose that for some i the unknown distance to the gate, n, lies between $f_i + 1$ and f_{i+2}. That is, assume that the gate will be found after the $(i+1)^{th}$ turn and before the $(i+2)^{th}$ turn.

The worst case ratio of the total distance walked divided by the distance to the gate is then

$$\frac{2\sum_{j=1}^{i+1} f_j + f_i + 1}{f_i + 1} = 2\frac{\sum_{j=1}^{i+1} f_j}{f_i + 1} + 1$$

Suppose that f is such that

$$\frac{\sum_{j=1}^{i+1} f_j}{f_i + 1} \leq c \quad \forall i \geq 1$$

where c is a constant. A lower bound on c gives us a lower bound of $2c+1$ for the cow problem.

Since f is monotone increasing for even or odd i then we can choose a sufficiently large i such that $\sum_{j=1}^{i+1} f_j - c > f_i + 1$. For ease of description we change variables to $g_j = f_j/(f_i+1)$.

For a fixed and sufficiently large i we have that c must satisfy the following infinite set of inequalities:

$$\begin{aligned}
c &> 1 + g_{i+1} \\
g_{i+1} &> \frac{1 + g_{i+2}}{c-1} \\
g_{i+2} &> \frac{1 + g_{i+1} + g_{i+3}}{c-1} \\
&\cdots \\
g_{i+k} &> \frac{1 + g_{i+1} + g_{i+2} + \cdots + g_{i+k-1} + g_{i+k+1}}{c-1} \\
&\cdots
\end{aligned}$$

We solve this system inductively by solving the first k inequalities. For a fixed k all we need do is discard the g_{i+k+1} term in the last inequality, substitute the bound for g_{i+k} in the expression for g_{i+k-1}, solve for g_{i+k-1}, and substitute the bound found in the inequality for g_{i+k-2} and so on.

For example, for $k = 0$ we have that

$$c > 1 + g_{i+1} > 1$$

Therefore, $c > 1$ giving us a lower bound of $2c + 1 = 3$.

For $k = 1$ we have that

$$g_{i+1} > \frac{1 + g_{i+2}}{c-1} > \frac{1}{c-1}$$

Therefore,

$$c > 1 + g_{i+1} > 1 + \frac{1}{c-1}$$

This implies that,

$$c^2 - 2c > 0$$

Therefore, $c > 2$ giving us a lower bound of $2c + 1 = 5$.

Continuing this process we find that c must be such that the infinite set of polynomials

$$c - 1$$

$$
\begin{aligned}
&c^2 - 2c \\
&c^3 - 3c^2 + c \\
&c^4 - 4c^3 + 3c^2 \\
&c^5 - 5c^4 + 6c^3 + c^2 \\
&\ldots
\end{aligned}
$$

must each be positive.

By induction on k we can show that these polynomials are of the form

$$f(k) = \sum_{i=0}^{\infty} \binom{k-i}{i} (-1)^i c^{k-i-1}$$

We require the *minimal* value of c for which *each* of these polynomials is greater than zero.

These polynomials obey the recurrence

$$f(k) = cf(k-1) - cf(k-2)$$

Which has characteristic equation

$$\lambda^2 - c\lambda + c = 0$$

This equation has roots:

$$\frac{c \pm \sqrt{c^2 - 4c}}{2}$$

If the roots are distinct then

$$f(k) = \frac{1}{c\sqrt{c^2 - 4c}} \left(\left(\frac{c + \sqrt{c^2 - 4c}}{2} \right)^{k+1} - \left(\frac{c - \sqrt{c^2 - 4c}}{2} \right)^{k+1} \right)$$

and this function is positive for all $k > 0$ if and only if $c > 4$.

If the roots are equal ($c = 4$) then

$$f(k) = (k+1)2^{k+2}$$

and this function is positive for all $k > 0$. ■

Therefore any algorithm to solve the cow problem must take at least $9n$ steps where n is the unknown distance to the gate.

Note that this is a lower bound on the *constant multiple* of the distance walked to the actual distance to the gate. We can in fact obtain algorithms which take no more than $9n - \Theta(\lg n)^k$ time for any k.

2.2 The m-Lane Cow Problem

Suppose that instead of a fence the cow is at the meeting point of one of m lanes and the cow wants to find an exit into one of the fields.

If $m = 1$ then the cow is at one end and that end is blocked the cow needs only to go along the lane looking for an exit. If $m = 2$ then we have the equivalent of the straight fence problem.

We visit the lanes cyclically since there is no advantage to favouring one over another, the goal may be in any one of the lanes. It does not matter if we use some other order to search the lanes, or even if we change the visiting order at random since all this does is complicate the description, it cannot affect the worst case.

THEOREM **2.2** *Let m be the number of lanes numbered in order of visits (assuming a cyclic visiting pattern) $0, 1, 2, \ldots, m-1$. Let f_i be the distance moved counting from the origin before the i^{th} turn. The best search function (up to lower order terms) is:*

$$f_i = (\frac{m}{m-1})^i \quad \forall i \geq 1$$

Proof: For a fixed lane, f must be increasing, otherwise we are wasting time.

That is, we must have that

$$f_i \geq f_{i-m} + 1 \quad \forall i \geq m \quad \text{where } f_{-j} = 0 \ \forall 0 \leq j \leq m-1$$

Suppose that the goal is found after the i^{th} and before the $(i+1)^{th}$ turn. In the worst case the goal will be exactly $f_i + 1$ away from the origin. In general the worst case ratio will differ *depending on which lane the goal lies in.* The best possible case occurs when the gate is in the 0^{th} lane (the first lane visited in cyclic order) and the worst case occurs when the goal is in the $(m-1)^{th}$ (the last lane visited in cyclic order). To minimize the worst case we must make these ratios equal (symmetry within all the lanes).

The worst case ratio is:

$$\frac{2\sum_{j=1}^{i+m-1} f_j + f_i + 1}{f_i + 1}$$

Let c be a constant such that

$$\frac{\sum_{j=1}^{i+m-1} f_j}{f_i + 1} \leq c$$

We wish to bound c from below to derive a lower bound on the worst case ratio of $2c+1$.

Suppose that we are using geometric search. We need to determine constants $a_j \ \forall 0 \leq j \leq m-1$, such that the function

$$f_i = a_{i \bmod m} k^i$$

makes the worst case ratios the same.

This leads to the following set of simultaneous equations

$$\begin{aligned}
\frac{\sum_{j=0}^{m-1} a_j}{a_0} &= \frac{k\sum_{j=0}^{0} a_j + \sum_{j=1}^{m-1} a_j}{a_1} \\
&= \frac{k\sum_{j=0}^{1} a_j + \sum_{j=2}^{m-1} a_j}{a_2} \\
&= \cdots \\
&= \frac{k\sum_{j=0}^{m-2} a_j + \sum_{j=m-1}^{m-1} a_j}{a_{m-1}}
\end{aligned}$$

Solving these equations we find that

$$a_j = k^{j/m} \quad \forall 0 \leq j \leq m-1$$

Substituting these values we find that the worst case ratio is

$$1 + 2\frac{k}{k-1}\sum_{j=0}^{m-1} k^{j/m} = 1 + 2\frac{k}{\sqrt[m]{k}-1}$$

Minimizing this function we find that

$$k = \left(\frac{m}{m-1}\right)^m$$

Just as in the lower bound analysis of the straight fence (2-lane) problem we can construct a sequence of functions of c each of which must be positive. Each function places bounds on how small c can be. These functions obey the recurrence:

$$f(n+m) = cf(n+m-1) - c^{m-1}f(n)$$

Which has characteristic equation:

$$\lambda^m - c\lambda^{m-1} + c^{m-1} = 0$$

This equation has a positive real double root at $c = m^m/(m-1)^{m-1}$, namely $\lambda = c(m-1)/m$. All other roots are negative or imaginary.

Dividing by c^m we see that this equation is equivalent to:

$$x^m - x^{m-1} + \frac{1}{c} = 0$$

where $x = \lambda/c$.

This implies that

$$c = \frac{1}{x^{m-1}(1-x)}$$

Thus, if $1 - x = 1/m$ then

$$c = \frac{m^m}{(m-1)^{m-1}}$$

The functions are (by inspection)

$$f(n,m,c) = c^{n-1} \sum_{i=0}^{\infty} \binom{n+m-2-(m-1)i}{i} (-1/c)^i$$

The value of c is such that for any fixed $m \geq 2$ we have $f(n,m,c) \geq 0 \ \ \forall n \geq 1$. ■

The worst case ratio is then

$$1 + 2\frac{m^m}{(m-1)^{m-1}} \approx 1 + 2e(m-1) \quad \text{(for large } m\text{)}$$

where $e = 2.71...$ is the base of natural logarithms.

Thus to search 2 lanes (equivalently, the straight fence) we use increasing powers of 2, to search 3 lanes we use increasing powers of 3/2, etc. and the constant multiple factor is roughly $1 + 2em$.

3 The 2-Dimensional Lost Cow Problem

Suppose that the cow is lost in a field and it wants to find the *fence* (to find the gate etc.). Assume that the fence is a straight line.

3.1 Fence a Bounded Distance Away

A cow is lost one dark night at most one kilometer away from the fence. The fence is perfectly straight and infinitely long. What is the minimum distance the cow must walk to be assured of reaching the fence?

Call the circle radius 1 kilometer centered at the cow's position the *reference circle.*

3.1.1 Orthogonal Turns

Suppose that the cow can only travel in straight lines or make turns of 90° to the left or right.

Here is a simple algorithm which takes 8 kilometers: (for clarity we describe it in terms of north, east, south and west, this does not imply that the cow knows the directions.) Go 1 kilometer east, 1 kilometer south, 2 kilometers west, 2 kilometers north, 2 kilometers east.

(See Figure 1. The cow's reference circle has been added for clarity.)

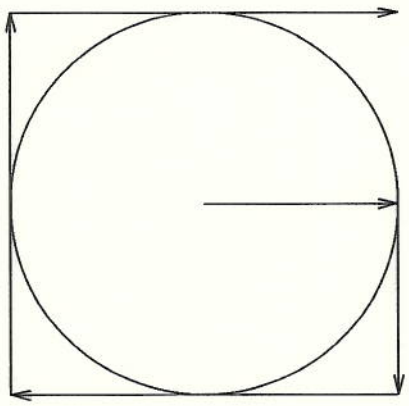

Figure 1: A Simple Orthogonal Algorithm

We can immediately improve this algorithm by the rather counter-intuitive idea of going $1 + \epsilon$ kilometers before our first turn. The idea being that although we initially go further east than is made necessary by our reference circle when we are returning east we only have to go 1 plus some function of ϵ kilometers rather than 2 kilometers. (See Figure 2. The worst possible position of the fence for this algorithm is shown as a dashed line.)

We shall now find the optimal ϵ extra kilometers to walk.

Since the fence must be tangent to the reference circle the distance we must cover is:

$$f(\epsilon) = 8 + 3\epsilon - \sqrt{(1+\epsilon)^2 - 1}$$

This function is minimized when $\epsilon = 3/\sqrt{8} - 1 \approx 0.06066$ and the minimum value is $5 + 2\sqrt{2} \approx 7.8284$ kilometers.

It seems that we cannot get a better bound if we make more turns. If we end up $1 + \epsilon$ kilometers east of the origin without getting there in a straight line then we will decrease the amount of kilometers we save when completing

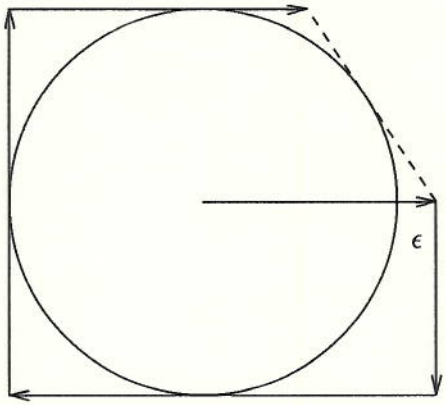

Figure 2: A More Complicated Orthogonal Algorithm

our path round the circle. For, the tangent would then be drawn from a point south of the origin making us travel further east when we return east to complete the tour.

If we break any of the three other turns into sequences of orthogonal subturns then we will still cover the same distance.

The only other option is to go east $1+\epsilon$ kilometers, north for δ kilometers east for a further γ kilometers then south for $1+\delta$ kilometers etc. This path can be transformed into the following: walk $1+\epsilon+\gamma$ kilometers east, δ kilometers north then $1+\delta$ kilometers south. This path can in turn be modelled much more simply by amalgamating the γ term into ϵ. Thus we are "paying" 2δ in order to save $2\delta\sqrt{(1+\epsilon)^2-1}$ off of our original algorithm.

The distance walked is then

$$f(\epsilon,\delta) = 7 + 3\epsilon + (2\delta+1)\left(1-\sqrt{(1+\epsilon)^2-1}\right)$$

Fixing ϵ we see that this function is monotone in δ. Since δ must be greater than or equal to zero, the function is minimized for $\delta = 0$, giving us the previous solution shown in Figure 2.

We do not have a good lower bound for this problem.

3.1.2 Arbitrary Turns

If arbitrary turns are allowed, the simplest algorithm is reach the perimeter of the reference circle along a radius, follow the perimeter for $\frac{3}{2}\pi$ radians and then take the tangent to the worst possible position of the fence. This give us $2+\frac{3}{2}\pi \approx 6.7124$ kilometers.

As in the orthogonal case it is possible to improve this algorithm by first going $1+\epsilon$ kilometers from the origin. ¿From this point we follow one of the tangents to the circle, then continue along the perimeter, until we find the tangent perpendicular to the worst case shoreline (see Figure 3).

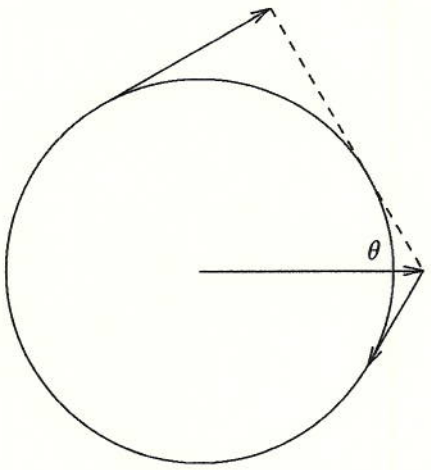

Figure 3: The G Algorithm (To Scale)

The distance we must cover is:

$$f(\epsilon) = \frac{3}{2}\pi + 2 + \epsilon + \sqrt{(1+\epsilon)^2 - 1} - 2\arctan\left(\sqrt{(1+\epsilon)^2 - 1}\right)$$

This function is minimized for $\epsilon = \frac{2}{\sqrt{3}} - 1$ and the minimum value is $1 + \sqrt{3} + \frac{7}{6}\pi \approx 6.3972$ kilometers ([7,6,1,5]).

It is perhaps more enlightening to write the function in terms of the half-angle subtended at the point $1+\epsilon$ away from the origin. Let this angle be θ then the total distance travelled is

$$f(\theta) = \frac{1}{\sin\theta} + \frac{1}{\tan\theta} + \frac{\pi}{2} + 2\theta + 1$$

Which has its minimum at $\theta = 60°$. This result has a simple geometric interpretation. If we circumscribe the reference circle by a regular hexagon then the G algorithm takes the following path: start at the origin, travel to one of the vertices of the circumscribing hexagon, follow one of the edges of the hexagon until the circle is intersected then follow the circle's perimeter until we have covered seven-twelfths of it, then follow the tangent from that point for at most 1 kilometer.

3.2 Proof of Optimality

Even though this algorithm has already been proved optimal [7,6,1,5] we here sketch an alternative way of proving optimality. We build our lower bound argument demonstrating the optimality of the G algorithm in a sequence of lemmas. First though we need the following lemma.

Lemma 3.1 *Given two rectifiable functions f_1 and f_2 defined on the interval $[a, b]$ $(a < b)$.*
If

1. *f_1 is convex on $[a, b]$,*
2. *$f_1(a) = f_2(a) = f_1(b) = f_2(b) = 0$, and*
3. *$0 \leq f_1(x) < f_2(x)\,, \quad \forall a < x < b.$*

Then, in the interval $[a, b]$, the arc length of f_1 is strictly less than the arc length of f_2.

The following lemmas follow directly from the above:

Lemma 3.2 *No minimal length path can intersect itself*

Lemma 3.3 *If P is any point inside the circle which is on a minimal length path then we must have arrived at P along a radius of the circle.*

Lemma 3.4 *If P is any point outside the circle which is on a minimal length path then we must have arrived at P along one of the tangents from P to the circle or directly from the center of the circle.*

3.3 Fence Arbitrarily Far Away

Now suppose that the cow *does not know the distance to the fence*, how many kilometers must he walk as a function of the actual (unknown) distance to the fence?

We can analyse this problem in a similar way to the unbounded orthogonal cow problem showing that the cow cannot hope for a better algorithm than about $13n$ steps.

3.3.1 Orthogonal Orientation

Suppose that the cow knows that the fence is orthogonally oriented and that it is situated some integer number of steps away horizontally or vertically. What is the minimum number of steps the cow must make to find the fence?

If the cow knows the direction of the fence then it has an optimal n-step algorithm. If the cow only knows that the fence is within n-steps then it has an optimal $3\sqrt{2} \approx 4.24n$-step algorithm to find the gate.

For the proof observe that it is necessary and sufficient to reach all four edges of the square of side length $2n$ centered at the origin. Consider the *last* time we turn. Suppose that we are going NE, then we must have explored all n steps to the south and west.

Now suppose that the fence is n steps away (horizontally or vertically) and the cow does not know n. What is the minimum number of steps the cow must make as a function of the actual (unknown) distance to the fence?

If the cow is restricted to only make spirals then we can establish a $12.74n$ lower bound on this problem using similar techniques to the 1-dimensional cow problem.

Let the distances that the cow covers *along the axes measured from the origin* be denoted by f_i. Figure 4 shows the early behaviour of this algorithm, a worst case position of the fence is shown as a dashed line.

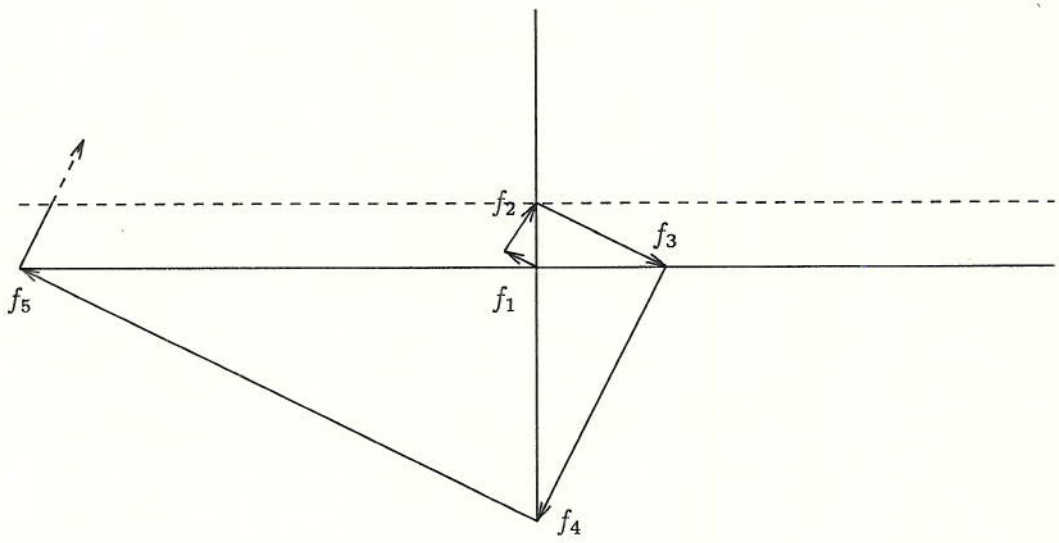

Figure 4: The 2-Dimensional cow Problem

Because of the difficulty of the analysis we drop the requirement that

the function values be integers. Note that the same could be done for the one dimensional problems as well.

We must have that

$$f_i > f_{i-4} \quad (\text{where} \quad f_{-j} = 0 \ \ \forall 0 \leq j \leq 3)$$

For a constant worst case ratio algorithm we must have that

$$\frac{\sum_{j=0}^{i+2} \sqrt{f_j^2 + f_{j+1}^2} + \sqrt{f_{i-1}^2 + f_i^2} + 1}{f_i} \leq c$$

where c is some constant.

The best geometric algorithm uses the constant $k = 1.3816\cdots$ to give a bound of $13.0212\cdots$. We do not have an analytic expression for this constant, it was found numerically.

3.3.2 Arbitrary Orientation

Now we wish to consider the problem in which the fence may have any orientation. The following lemmas give necessary properties of the optimal search path.

Lemma 3.5 *In the worst case, any curve has a segment arbitrarily close to the fence and for which the fence is a tangent and they do not intersect.*

Proof: Otherwise we may give the fence another orientation, thereby increasing the arclength of the curve. ■

Lemma 3.6 *The optimal curve is convex with respect to the origin in any segment subtending an angle $\theta < 2\pi$.*

Proof: If not, we can construct a path, using a convex polygon that encloses the curve, which has a lower path length than the optimal path, a contradiction. ■

Lemma 3.7 *In the optimal algorithm, after a complete cycle 2π, $r(\theta + 2\pi)$ must be strictly greater than $r(\theta)$. This is valid for all θ.*

Proof: If not, it is possible to construct a circle, with smaller path length, and which is at least as good as the previous one, a contradiction. ■

Lemma 3.8 *The optimal algorithm must have the same worst case for any line orientation.*

Proof: If not, we can replace the curve for the cases that are bigger, using a rotation of the curve that has the lower worst case. This implies that the curve is not optimal for that orientation, a contradiction. ■

Corollary 3.1 *The optimal curve must be similar against rotations. This implies that the derivative of the curve is continuous.*

Lemma 3.9 *The optimal curve, $r(\theta)$, is a monotonic increasing function. Hence the derivative is strictly positive.*

Proof: This follows from the above lemmas. ■

These lemmas, tell us, that the optimal curve must be similar with respect to rotations and dilations (that is, the curve must have *spiral similarity*). The only known curve with these properties is the logarithmic spiral.

In this case the cow executes a logarithmic spiral $r = e^{a\theta}$ where $a = 0.22325\cdots$. This value (a) is a numerical approximation for the best logarithmic spiral. If the fence is n units away this algorithm takes approximately $13.49n + O(\ln n)$ kilometers. Note that $e^{0.22325} = 1.250\cdots$.

We do not yet have a good lower bound for this problem, beyond the bound established for the G algorithm. By analogy with the cow problem it might pay to look at the class of functions which are monotone in θ and obey the following inequality for some constant c:

$$\frac{\int_0^{x+\alpha} r(\theta)d\theta}{r(x)} \leq \frac{\int_0^{x+\alpha} r(\theta)d\theta}{d(x)} \leq c \quad \forall x$$

where $d(x)$ is the distance between the fence and the curve in the worst case and α is the angle between the point of tangency and the point of intersection measured respect to the origin. We then ask for the smallest possible c.

Lemma 3.10 *The value of α is bounded by $\pi \leq \alpha \leq 2\pi$.*

Proof: This follows from the previous lemmas. ■

If a linear spiral is used ($r = a\theta$) the total distance in the worst case is $n^2/2a + O(n)$. Note the similarity between the algorithms for the lost cow problem (1 dimension) and the lost cow problem (2 dimensions) (a linear type search yields $O(n^2)$ steps and a geometric type search yields $O(n)$).

4 The Lost Robot Problem

Suppose a robot has to find a trapdoor in the plane given that it lies within n steps (horizontally or vertically). Assume that the plane is a rectangular lattice and that the robot can move left, right, up or down in one step. What is the minimum number of steps the robot must make to find the trapdoor?

Call the vertices of the lattice which lie within n steps of the origin the *reference diamond.* This is of course a circle under the $\mathcal{L}_1$ metric (also called the Manhattan or taxicab metric). As in the lost cow problem we shall use compass bearings to describe algorithms, this is for ease of description only. Note the close relationship between this problem and lost man problem using orthogonal turns except that the robot must find an object of *finite size.*

Suppose, without loss of generality, that any algorithm always begins by going north. Here is a simple algorithm to solve this problem. First, follow a rectangular spiral until the border of the diamond is reached. That is, the algorithm begins with the following sequence of moves:

N E S S W W N N N E E E ...

Having reached a vertex on the border of the diamond, we go up and down covering the four triangular corners outside the spiral. (See the example in Figure 5).

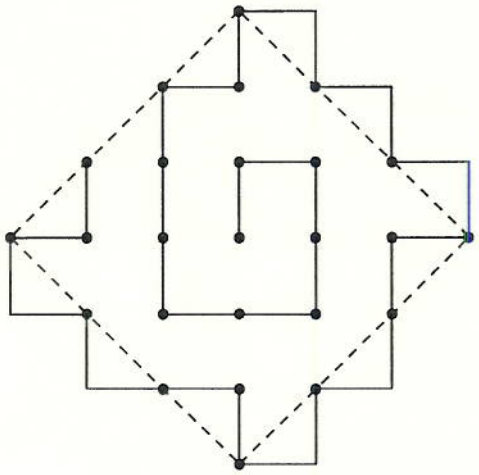

Figure 5: A Spiral Algorithm for $n = 3$

The number of steps needed to execute the spiral portion of the search is $n(n-1)$ for even n and n^2 for odd n. The number of steps necessary to

cover the four triangles is given by the following recurrence

$$f(n) = \begin{cases} 6 & n = 1 \\ 14 & n = 2 \\ f(n-2) + 4n + 6 - 2(n \bmod 2) & n \geq 3 \end{cases}$$

The total number of steps is then

$$f(n) = 2n^2 + 4n + n \bmod 2$$

Since we must at least take $2n^2 + 2n$ steps (to examine all possible vertices in the diamond) then this algorithm is within $2n + 1$ steps of the lower bound.

This algorithm takes 7 moves to reach the last vertex of distance 1, and this is optimal for $n = 1$. (The bound of 7 can be shown by brute force enumeration). It takes 16 moves to reach the last vertex of distance 2.

If we know that the trapdoor is exactly n steps away, then we can do considerably better. In this case, we move directly to one of the points at distance n. We then follow the zigzag path that visits the the nodes at distance n in sequence. The total number of moves is $9n - 2$, which is optimal. It is interesting to note that the ratio is the same as in the best cow and fence solution.

4.1 Trapdoor Arbitrarily Far Away

Suppose as before that the robot knows that there is a trapdoor *but it doesn't know how far away it is*. What is the minimum number of steps the robot must make to find the trapdoor?

As in the cow problem, our search strategy must ensure that every point within a finite distance is covered in a finite number of steps. Unlike the cow problem, it is not necessary to visit any point more than once, since we can traverse a spiral path which will visit each point exactly once.

The following pattern yields an algorithm requiring only $2n^2 + 5n + 2$ moves.

N E S S W S W N N

(These visit all vertices at distance 1.)

W N E N E N E S E S S

(These visit all unvisited vertices at distance 2.)

The next 15 moves visit all unvisited vertices at distance 3 and so on.

This algorithm also generates a spiral, but the shape of the spiral more closely approximates the diamond shape formed by the points at distance n (see Figure 6).

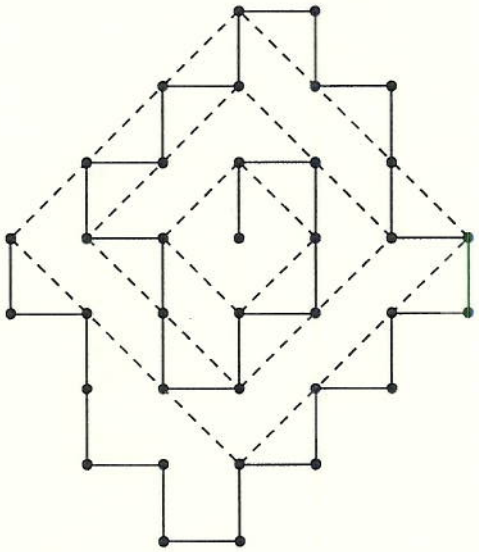

Figure 6: Early Steps of the Modified Spiral Algorithm

This algorithm makes some counter intuitive steps. For example, the sixth move, S, causes us to use two extra moves to reach the last point at distance one. However, if we use the shorter route to explore all vertices at distance one, then we require more than 13 additional moves to visit all the remaining points at distance 2, and so the adversary can beat the cost of 20 of the previous algorithm. Thus, for each n, we must tradeoff between exploring all vertices at distance n quickly against paying a little more at each layer so that if it happens that the trapdoor is further out we gain.

This algorithm is within $3n + 2$ of the simple $2n^2 + 2n$ lower bound.

4.2 A Better Lower Bound

We can improve our lower bound when the trapdoor is arbitrarily far away to show that the modified spiral algorithm is optimal to within $n+3$ moves.

THEOREM **4.1** *Any algorithm which can find a trapdoor at any finite distance n requires at least $2n^2 + 4n - 1$ steps.*

Proof:

Suppose that we visit all points within distance $n+1$ before visiting some point at distance n. The adversary places the trapdoor at the last position we search at distance n, and we have thus used at least $2(n+1)^2+2(n+1) =$

$2n^2+6n+4$ moves to find the trapdoor. This bound exceeds modified spiral search by at least $n+2$. Therefore any optimal search must search all points at distance n *before* visiting *all* points within distance $n+1$.

Suppose that we visit all points within distance $n-1$ before visiting any at distance n. Consider those points along the edges of the diamond containing all points at distance n or less. Since these points lie along diagonals, between visits to any two points on this boundary, we must visit at least one other point. There are $4n$ such points which implies that we need $4n-1$ inbetween points, for a total of $8n-1$ points. Note that we do not imply that these are the only inbetween points visited, nor that the boundary points are visited in sequence.

The adversary chooses the last point visited by the algorithm at distance n. This will require at least $8n-1$ moves in addition to the $2(n-1)^2+2(n-1)$ minimum required to search the points at distance $n-1$ or less, for a total of $2n^2+6n-8$, which also exceeds modified spiral search (for $n > 10$).

Thus, any optimal algorithm must visit some number of points at distance $n+1$ (or farther) before completing those at distance n. Suppose that this number is $K(n)$. If $K(n)$ is decreasing in n, then for some n, $K(n)=0$, and the above argument applies for $n+1$. Otherwise, $K(n)$ is constant or increasing for some values of n.

Clearly, $K(n) < 4n$, else the adversary chooses the last point visited at distance n for a total of $2n^2+6n$ points which requires more moves to visit than modified spiral search.

Suppose, for some n, that $K(n) \geq K(n-1)$ and that k_1 of the $K(n-1)$ are at distance n. Then we need at least $2(n-1)^2+2(n-1)+K(n-1)$ moves to visit the nodes at distance $n-1$ or less. There are $4n-k_1$ more nodes at distance n left to visit, and using the same argument as before, this implies at least $2(4n-k_1)-1$ moves will be required to complete that layer. Finally, we need to add at least k_1 nodes at a distance greater than n to make $K(n) \geq K(n-1)$. However, these may be added without extra moves if they are the inbetween points used in completing the points at distance n. To have all k_1 points added as inbetween points requires that $4n-k_1-1 \geq k_1$ which implies that $k_1 \leq 2n$. Thus, we at have least

$$2(n-1)^2+2(n-1)+K(n-1)+2(4n-k_1)-1 \geq 2n^2+6n-k_1-1 \geq 2n^2+4n-1$$

■

¿From this it follows that modified spiral search is within $n+3$ steps of optimality.

Essentially, the above proof says that we must explore two levels at once. Interestingly, if we are given *just the parity of the trapdoor's distance* then we can improve the algorithm by n steps! That is, we can get a $2n^2 + 4n + n \bmod 2$ algorithm by exploring the appropriate pair of levels at each step.

To visit all points within a distance of n without repeating any given that n is odd use:

N E S S W W N

(visits all at distance 1 in 7 moves and this is optimal)

W N E N E N E S E S S E S W S W W S W N W N W N

(visits all within 3 steps in 31 moves)

In general it will take $2n^2 + 4n + 1$ steps to visit all within distance n, where n is odd. However, if n is even, then this algorithm takes $2n^2 + 6n + 4$ moves.

To visit all within n, given that n is even use:

N E S E S W S W N W N W N E N E

(visits all within 2 in 16 moves)

The next cycle starts

N E S E ...

and follows around again to visit all within 4 in 48 moves. In general, this can visit all within distance n for n even in $2n^2 + 4n$ moves. However, if the trapdoor is placed at an odd distance n, then this algorithm takes $2n^2 + 6n + 3$ moves.

For both algorithms, if the adversary is free to place the trapdoor at a distance of opposite parity to the parity that the algorithm was designed for, then the bound is worse than modified spiral search.

The lower bound says that we can't eliminate more than half of the "wasted" moves (moves outside of the diamond), and the $2n^2 + 5n + 2$ algorithm results from trading off between the even and odd algorithms, making some of the wasted moves at even levels useful at the odd and vice versa.

It is interesting that just knowing whether the distance is odd or even is sufficient to improve the algorithm by nearly n moves! (Also, this gives the best bound we have been able to achieve even if we are given that the trapdoor is within n steps!)

5 Searching with Error

When searching in the plane, two different kinds of measurement errors may occur. One error may occur when measuring the *distance* and the other when measuring the *direction*.

For example, if you are walking along a path from point A to point B while counting your steps and checking your orientation you may miscount your steps or you may misjudge your orientation (or both). How can you guarantee that you will indeed reach B starting from A if you have good estimates on how much you could possibly misjudge your current position and direction?

This problem is a very simple analogue of the general problem of ensuring that a robot keeps to its correct path while mobile or that a robot can find an object it is working on.

We will see that these two types of errors have different implications when constructing a search strategy, and that, again, knowing the general *direction* of the goal is more important than knowing its distance away from you.

In the cow problem, only one type of measurement error is possible: an error in the distance. This is because the only two possible directions are given by the fence.

The simplest way to model this kind of error is to assume that an error of no more than δx distance units is made in each step. Suppose that you know the distance, d, to the gate, you need to consider your error, and make the appropriate correction in the path. Hence, you need to walk a distance d' such that

$$d' \geq d + \delta x d'$$

However, you can make errors in the opposite direction as well, so the overall distance walked is

$$\frac{1+\delta x}{1-\delta x} d$$

Hence, in the worst case you will have to walk

$$d + 2\frac{1+\delta x}{1-\delta x} d = 3d + 4\delta x d + O(\delta x^2 d)$$

If you don't know the distance, d, then the error cannot affect any search algorithm, *provided that* the increase in the number of steps is greater than the worst possible error you could have made. The same applies to the k-way cow problem.

In the lost cow problem, a direction error is more important.

To analyse possible consequences of this let us assume that a path consists of a sequence of straight line segments and that the cow can walk a straight line but when he changes direction he can make an error of $\delta\theta$ in his intended direction.

A consequence of this model is that in a convex path, if you want to be assured of being "outside" of the path then you need to make a correction in your direction.

For example, if the cow wants to go around the reference circle, she must be sure that she is never *in* the circle. (If not, she may never find the fence.) Hence, in each segment of his path she must make a correction of $\delta\theta$ in his orientation. However, in the worst case, the overall error in the angle is $2\delta\theta$. If the first segment has length l, then after traversing it we can be as much as $2l \sin \delta\theta$ units away from the exact path. After j segments the angle error will be $2j\delta\theta$ and with a distance depending on the segment lengths of the exact path.

Suppose now that the error $\delta\theta$ is made in one unit step. That is, the cow cannot even walk in a straight line without drifting off his intended direction by $\delta\theta$. (Perhaps she had a little too much to drink.)

In this case, around the circle, after walking a distance l the angle error is $2l\delta\theta$, that is, a spiral, instead of a circle. A consequence of this, is that it is not possible to guarantee that we reach an arbitrary distance away. This is because in a straight line it is possible to have a $\delta\theta$ error in the direction *on the same side of the line.* Hence, in the worst case, the line degenerates to a circle of radius $r = 1/(2\sin(\delta\theta/2))$. Therefore, the maximum distance from the origin in the worst case is $2r \approx 2/\delta\theta$ for small $\delta\theta$.

If we have both types of errors at the same time, the corrections are more difficult. Also, the resulting path is worse. A simple model in this case is that after each unit distance, the final point is inside a circle of radius $\delta\rho$. The maximum distance is of the same order as previously for small $\delta\rho$ (that is, $2/\delta\rho$).

Finally, in an arbitrary path, it is difficult to compute a correction, even if the average error made throughout the path may be small. This is because an error in one step may be compensated for in another step.

6 Open Problems

6.1 The Lost Swimmer in a River

The lost swimmer problem stated at the beginning of this paper is still open ([3]). We conjecture that the best path is of length $1 + 2\arctan(3/4) \approx 2.287w$ where w is the known width of the river.

What if the river's width is unknown?

6.2 Billiard Paths

Generalizing from the restricted version of the 2 dimensional cow problem we ask: given a billiard table shaped like a convex polygon, starting at the origin we wish to bounce a billiard ball off of all of the edges. What is the minimum length of the ball's path as a function of the width of the polygon?

If we restrict the polygon to be a regular n-gon then we have the following results for small n: (assume that the polygons are inscribed in a circle of radius 1).

If $n = 3$ then the minimal length seems to be $2n$. The best path seems to be along the perpendicular bisector of any side.

If $n = 4$ the minimum length is $3\sqrt{2}$. This solution is reminiscent of the 1 dimensional cow problem since the best path is a diagonal.

For general n?

6.3 The Lost Cow with a Little Lamb

Suppose that the lost cow has a little lamb and she is lost as before. The cow decides to leave the lamb at their reference position and find the fence. Having found the fence she intends to return for her lamb and return to the fence. What is the minimum distance she must walk to complete the trip?

We conjecture that a symmetric G algorithm is the optimal solution. This algorithm is such that the final path is symmetric to the initial path.

6.4 The Tired Lost Cow

Suppose as before that the cow is lost at most 1 kilometer away from the nearest fence. The cow is very tired and can only walk x kilometers[1] what is the best path for her to take to maximize her chance of reaching the road?

[1] Or she happens to be in a jeep and only has enough gas to travel x kilometers.

In this case we need to maximize the amount of the perimeter of the reference circle *covered* by the algorithm as a function of the distance walked. The optimal path chosen will, in general, depend on x and it may be that the optimal solution is not continuous in x.

The simplest algorithm is just to walk for x kilometers in some direction. The amount of perimeter covered is:

$$f(x) = \begin{cases} 0 & x < 1 \\ 2\arccos\left(\frac{1}{x}\right) & x \geq 1 \end{cases}$$

This algorithm can only cover half of the circle's perimeter (asymptotically).

The second algorithm is based on the G algorithm. Walk $1+\epsilon$ kilometers in some direction, then take the tangent to the reference circle, then follow the perimeter of the circle until y kilometers are left from the initial x kilometers. Then follow the tangent for the y remaining kilometers.

Let $\epsilon_1 = 1 + \epsilon$. The amount of perimeter covered is:

$$f(x) = \begin{cases} 0 & x < 1 \\ 2\arccos(\frac{1}{x}) & 1 \leq x < \epsilon_1 \\ 2\arccos(\frac{1}{\epsilon_1}) & \epsilon_1 \leq x < \epsilon_1 + \sqrt{\epsilon_1^2 - 1} \\ 2\arccos(\frac{1}{\epsilon_1}) + & \\ 2\arctan\left(x - \epsilon_1 - \sqrt{\epsilon_1^2 - 1}\right) & \epsilon_1 + \sqrt{\epsilon_1^2 - 1} \leq x < \epsilon_1 + \sqrt{\epsilon_1^2 - 1} + y \\ 2\arccos(\frac{1}{\epsilon_1}) + x & \\ -y - \epsilon_1 - \sqrt{\epsilon_1^2 - 1} + 2\arctan(y) & x \geq \epsilon_1 + \sqrt{\epsilon_1^2 - 1} + y \end{cases}$$

The best choice of y for this algorithm turns to be 1 independently of x and ϵ.

The best ϵ for this algorithm is

$$\epsilon = \begin{cases} \dfrac{2x(1+x^2) - x^2\sqrt{2x^2-1}}{1+2x^2} - 1 & 1 \leq x < 1 + \sqrt{3} \\ \dfrac{2}{\sqrt{3}} - 1 & x \geq 1 + \sqrt{3} \end{cases}$$

The distance of the last part of this algorithm (found by substituting the best ϵ) is

$$x - 1 - \sqrt{3} + \frac{5}{6}\pi \approx x - 0.114 \quad \text{for} \quad x \geq 1 + \sqrt{3} \approx 2.732$$

Comparing the two algorithms proposed we see that if $x < 2.4297$ then it is better to go straight forward. If $x \geq 2.4297$ it is better to use the last algorithm. The breakeven point (2.4297) was found numerically, we have no analytic expression for it.

Hence, given x, the best known strategy is:

If $x < 1$ then do anything you like or wait for help.

If $1 \leq x < 2.4297$ then walk straight forward in an arbitrary direction.

If $2.4297 \leq x < 1 + \sqrt{3}$ then compute the optimal ϵ. Walk $1 + \epsilon$ kilometers in any direction, and then take a tangent to the circle.

If $x \geq 1 + \sqrt{3}$ then walk $2/\sqrt{3}$ kilometers in any direction, turn through 120° and walk to the perimeter of the reference circle. Follow the perimeter until exactly 1 kilometer is left from your x original kilometers, then take the tangent from your current point. (Of course, if $x \geq 1 + \sqrt{3} + 7\pi/6$ then this is the G algorithm.)

We looked at other other possible algorithms, for example go around the circle or go out to some distance α then turn through some angle θ and walk in a straight line. None of these other algorithms beat the above algorithm for x in each range.

The best algorithm for bounded search is still an open question.

6.5 The Lost Swimmer Problem

Suppose a swimmer is lost at sea on a dark night 1 kilometer away from a circular island of known radius r.

Using a variation of the G algorithm, that at the end takes a tangent that is orthogonal to the worst case position of the island (See Figure 7) the distance is

$$\begin{aligned} f(x) &= x + \sqrt{x^2 - 1} + 2\pi + \sqrt{r^2 + 2r} - r \\ &\quad - \arctan\left(\sqrt{x^2 - 1}\right) - \arccos\left(\frac{1}{r+1}\right) - \arccos\left(\frac{x^2 + 2r + 1}{2x(r+1)}\right) \end{aligned}$$

where $x = 1 + \epsilon$.

This is a difficult function to minimize because of the dependence on x and r in the last term. Table 1 gives some optimal values of x for various r (computed numerically).

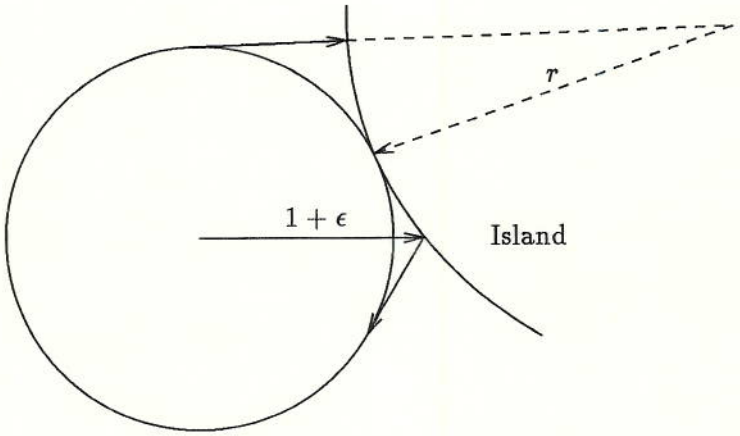

Figure 7: The G Algorithm Searching for an Island

r	ϵ
0	0
1/2	0.063
1	0.088
2	0.111
5	0.133
∞	0.155

Table 1: Some Optimal Values of ϵ for Various r

If the island is only *at most* 1 kilometer away then the above algorithm does not work when r is small. In that case we need a curve that fills the circle. One possibility is a linear spiral such that the maximum distance between two points on the spiral with the same angle is less that or equal to $2r$.

This problem may be related to Oglivy's River problem ([10]).

6.6 The Lost Swimmer in an Inlet

Suppose that the swimmer is lost in an inlet. The shoreline of the inlet is almost a complete circle except for a small portion of the perimeter which is open to the sea. Let the inlet have radius 1 kilometer and let the portion of the missing perimeter subtend an angle of θ at the center of the circle. What is the minimum distance the swimmer needs to swim to reach land?

6.7 The Lost Diver Problem

Suppose that a deep-sea diver is diving on a dark night. Given that he knows that he cannot be more than 1 fathom deep but that it is so dark that he cannot tell where surface lies what is the minimum distance he must swim to find the surface?

Call the sphere of radius 1 fathom centered on her position the *reference sphere.*

Suppose we circumscribe the reference sphere with a polyhedron with n faces, each face of which is tangent to the sphere.

Lemma 6.1 *It is sufficient to visit each vertex of the circumscribing polyhedron.*

Proof: Any plane which is tangent to the sphere and does not contain a face, must intersect at least one face. Therefore it must separate at least one vertex from the sphere, hence there is at least one path that crosses the plane. ■

It is possible that we can do better if we select non-vertex points on each face.

Consider the case when three faces meet at a vertex. Take the spherical triangle induced by the tangent points of the faces, and then consider a plane tangent to the sphere at any interior point of the spherical triangle. We need only have one point on each line from the tangent points of the faces to the vertex located so that any such tangent plane will separate

at least one of those points from the sphere. Deciding on the appropriate enclosing polyhedron, the location of the points and the connecting path seems non-trivial. We cannot even guarantee that the polyhedron must be regular as is shown below.

Circumscribe the sphere with a regular tetrahedron. First, swim straight out to one of the vertices, then follow a sequence of 3 edges to visit each of the other vertices.

We might be able to improve on this algorithm by distorting the tetrahedron. Suppose we move the last vertex visited directly away from the sphere. Now only the opposite face will be touching the sphere. Shrink this face in some manner so that the other three faces are again touching the sphere. (Note that uniform shrinking is not necessarily the best method, since two edges of the face are traversed, but not the third.)

The net effect of this process is that the length of the path from the origin to the first vertex is reduced, as are the lengths of two of the traversed edges. The last edge will be increased. If there exists an optimum stretch, what is it?

If now the deep-sea diver does not know how far away from the surface he is; What is the minimum distance he must travel to find the surface as a function of the actual (unknown) distance to the surface?

6.8 The Average cow Problem

If the cow knows that the gate is within n steps and that it is distributed uniformly about the interval of length $2n$ centered on the cow's position then the best worst case algorithm is also the best average case algorithm, with a average distance of $3n/2$. However, if the gate's distribution is not uniform then the best average algorithm is dependent upon the probability distribution.

If the cow remembers that the gate is likely to be near by, then it might want to turn after a smaller number of steps, since going very far from the origin the probability of finding the gate further on is much less than finding it nearer the starting point on the other side. Suppose, for example, that the gate has a triangular distribution with origin in the starting point given by

$$p(x) = \left| \frac{n - x}{n^2} \right|$$

for $-n \leq x \leq n$, and 0 everywhere else. At what points should the cow turn?

Let be $t_1 n$, $t_2 n$, ... , $t_m n$ the absolute distance from the origin of the turning points of an algorithm with m turning points. That is, go $t_1 n$ steps to the right then return to the origin and go $t_2 n$ steps to the left, etc. If after the last turn we have not found the gate, we go all the way to the end of the other side. The average distance travelled by an optimal algorithm is

$$E[d] = n(1/3 + F(t_1, ..., t_m))$$

where

$$F(t_1, ..., t_m) = \min_{t_1,...,t_m} ((1 - t_m)^2 + \sum_{i=1}^{m} 2t_i - t_i^2(2 - t_i) + t_i t_{i-1}(2 - t_{i-1}))$$

with $t_0 = 0 < t_1 < t_2 < ... < t_m < 1$.

The optimal points are such that

$$t_{i+1} = \frac{2 - 4t_i + 3t_i^2 - t_{i-1}(2 - t_{i-1})}{2(1 - t_i)}$$

for $i > 0$ and

$$3t_m^2 + t_{m-1}^2 - 2(t_m + t_{m-1}) = 0$$

Some solutions to this system of equations are:

m	t_1	t_2	t_3	$E[d]$
0				$4/3 \approx 1.3333$
1	$2/3$			$32/27 \approx 1.1851$
2	$\approx .656094$	$\approx .969746$		≈ 1.183521
3	$\approx .656093$	$\approx .969742$	$\approx .9998$	≈ 1.183520

It is possible to show that the solution for m turning points, must be different from any of the solutions of i turns, with $i < m$, because this points are singularities of the equation. For the same reason, each time the next value for t_i is closer to the previous one. We conjecture that the optimal algorithm for this distribution is such that the number of turning points is infinite. Intuitively, this happen, because there is always a point, where is better to turn back and look at places where the probability of finding the fence is greater.

6.9 The Cow Problem with Polygonal Fences

Suppose that the fence is a simple polygonal curve in the plane bisecting the plane into two halfplanes.

Suppose without loss of generality that the cow is in the "upper" half-plane. We may solve this problem by finding the upper convex hull of the fence and then solving the problem in the reflex subregions (if any).

A suitable example is searching a "V-shaped" region. There is a simple algorithm which might be called "Bow-tie search" which is asymptotically optimal.

As the angle is reduced to zero, the problem reduces to the 1-way lane case. As the angle is increased to 180 degrees, the problem becomes the 2-way lane problem. Note however that the lane problem is intrinsically different from the fence problem. In the fence problem the cow can, should it so desire, leave the fence and walk in the field. In the lane problem this is not possible, the cow is constrained to walk in a lane.

What about arbitrarily shaped regions? If each reflex subregion is convex with respect to the cow then we may solve the problem by a simple extension of previous ideas. If the subregion is not convex with respect to the cow then it is unclear what to do. Perhaps we could solve each "sub-reflex region" recursively?

7 Conclusions

To the best of our knowledge these kinds of incompletely specified problems have only occurred in recreational mathematics and have not been studied as a class. We think that they are deserving of comprehensive study as simple optimality arguments (in particular variants of convexity and symmetry properties) are applicable to a wide sub-class of these problems. Further, and more importantly, these problems are (very simple) models of searching in the real-world. It is very often the case that we do not know many of the parameters that are usually taken for granted when designing search algorithms.

Bentley and Yao ([2]) considered a similar kind of problem to the cow problem. They constructed an almost optimal algorithm to find an integer chosen from an *unbounded* set of positive integers. The problems differ in that we have to pay costs proportional to the distance of a probe from the last position probed whereas they have random access to any location. Rivest *et al.* ([11]) examined problems in which we wish to search for an integer but the adversary can tell a bounded number of lies. This is similar to the searching with error problem. Karp, Saks and Widgerson's ([8]) consider "wandering RAMs" with bounded memory searching binary trees.

For them the number of node visits was the cost measure, this problem is closer in spirit to the class of problems we consider here.

The most striking result about the lost cow, cow, and robot problems is that the relative information of knowing the general *direction* of a goal is much higher than knowing just the *distance* to the goal. (Using 20/20 hindsight this result is intuitively obvious.)

These results suggest that it is better to search a search space *backwards from the goal towards the start* (assuming the goal is known) rather than searching *forwards from the start towards the goal.* Of course these are very simple problems and results from the more comprehensive problems currently under investigation may be more enlightening.

Problem	Knowledge		
	Direction	Distance	Nothing
1-d cow	n	$3n$	$9n$
m-lane cow	n	$(2m-1)n$	$(1+2m^m/(m-1)^{m-1})n$
2-d Ortho-cow	n	$4.24\cdots n$	$\leq 13.02n, \geq 8.66n$
2-d cow	n	$6.39\cdots n$	$\leq 13.49n$
Robot	n	$\leq 2n^2+4n+n \bmod 2$	$\leq 2n^2+5n+2$
		$\geq 2n^2+2n$	$\geq 2n^2+4n-1$
Robot with Parity	n	$\leq 2n^2+4n+n \bmod 2$	$\leq 2n^2+4n+n \bmod 2$

Table 2: The Advantage of Knowing Where Things Are

8 Acknowledgements

A variant of the 2-dimensional cow problem (fence a known distance away) was posed in [10] and was solved independently by Joris ([7], referenced in [4]). We wish to thank Gaston Gonnet for bringing this to our attention. We wish to thank Ron Graham for giving us further references ([1,6,5]) on this variant. Melzak ([9], page 153) has claimed a solution, however this solution is incorrect giving a bound of $6.459\cdots$ instead of $6.397\cdots$.

Jon Bentley and Andrew Yao (private communication) independently studied a variant of the 1-dimensional cow problem (unpublished).

We would like to thank Jon Bentley, Gaston Gonnet, Ron Graham, Bob Reckhow, and Chee Yap for many clarifying discussions on these problems.

References

[1] Bellman, R.; "A Minimization Problem," *Bulletin of the American Mathematical Society,* **62**, 270, 1956.

[2] Bentley, J. L., and Yao, A. C.-C.; "An Almost Optimal Algorithm for Unbounded Searching," *Information Processing Letters,* **5**, 82-87, 1976.

[3] Chang, S.-K.; "A Triangular Scanning Technique for Locating Boundary Curves," *Computer Graphics and Image Processing,* **3**, 313-317, 1974.

[4] Faber, V. and Mycielski, J.; "The Shortest Curve that Meets all the Lines that Meet a Convex Body," *American Mathematical Monthly,* **93**, 796-801, 1986.

[5] Gluss, B.; "An Alternative Solution to the the 'Lost at Sea' Problem," in 16^{th} *National Meeting of the Operations Research Society of America,* Pasadena, 1959.

[6] Isbell, J. R.; "An Optimal Search Pattern," *Naval Research Logistics Quarterly,* 4, 357-359, 1957.

[7] Joris, H.; "Le Chasseur Perdu dans la Forêt," (in French), *Element der Mathematik,* **35**, 1-14, 1980.

[8] Karp, R. M., Saks, M. and Widgerson, A.; "On a Search Problem Related to Branch-and-Bound Procedures," 27^{th} *Annual Symposium on Foundations of Computer Science,* 19-28, 1986.

[9] Melzak, Z. A.; *Companion to Concrete Mathematics: Mathematical Techniques and Various Applications,* John Wiley and Sons, Inc., 1973.

[10] Oglivy, C. S.; *Tomorrow's Math: Unsolved Problems for the Amateur,* Oxford University Press, 1962.

[11] Rivest, R. L., Meyer, A. R., Kleitman, D. J. and Winklmann, K.; "Coping with Errors in Binary Search Procedures," *Journal of Computer and System Sciences,* **20**, 396-404, 1980.