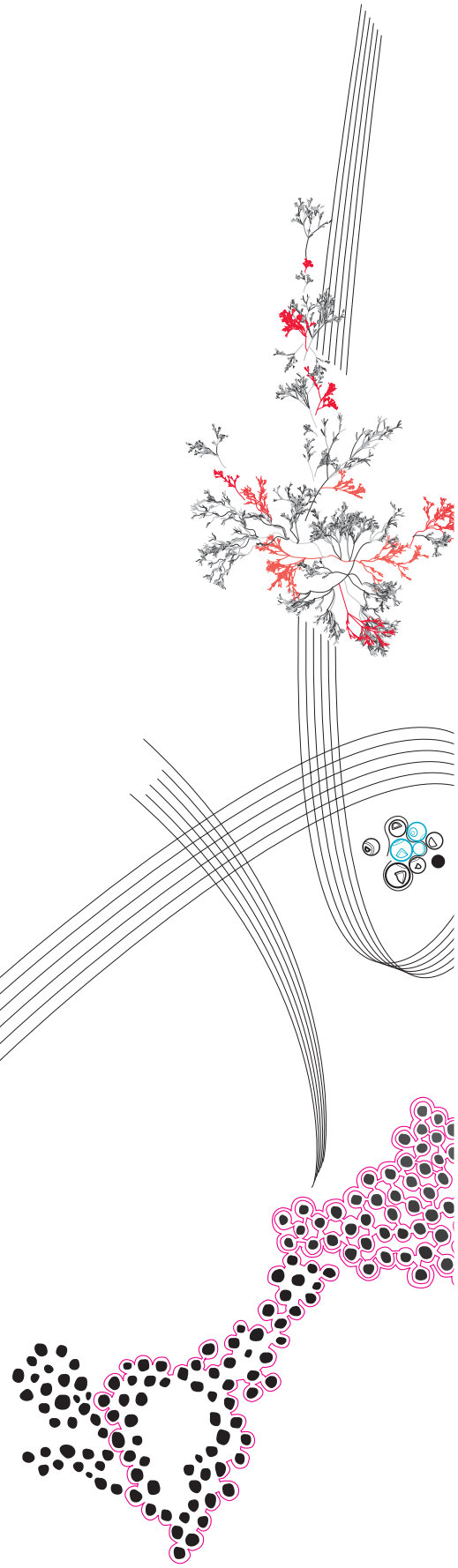

BSc Thesis Applied Mathematics

Finding lower bounds for the competitive ratio of the cow path problem with a non-optimal seeker

M.C. Vos

Supervisor: W. Kern

June 23, 2020

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Finding lower bounds for the competitive ratio of the cow path problem with a non-optimal seeker

M.C. Vos*

June 23, 2020

Abstract

A tight lower bound for the competitive ratio of deterministic algorithms for the cow path is well-known. In this thesis, we generalize the cow path problem by assuming that the seeker finds the hider after some known number of visits. We seek to find lower bounds for the competitive ratio of deterministic algorithms for this problem. The thesis describes the general form of optimal algorithms and succeeds in finding a tight lower bound for the competitive ratio when the required number of visits is odd.

Keywords: cow path, competitive, lower bound, online, algorithm

1 Introduction

Search problems are a well studied and widely applied topic in mathematics and computer science. The cow path problem, also known as the linear search problem, is a well-known example of such a problem. It was first formulated by Richard Bellman in 1963 [3], and was independently considered by Anatole Beck [1]. The problem can be regarded as follows. A cow is looking for a daisy growing somewhere on a straight line. It will find the daisy when it stands exactly on top of it. Mathematically, this line is the real number line with the cow starting at the origin. The daisy can be found at some real number, at least $m_0 > 0$ distance away from the origin. The cow can move up and down the real number line with a velocity of one. The goal of the problem is to find a path (algorithm) that visits each $x \in \mathbb{R}$ after travelling a distance of at most $\rho|x|$. Such an algorithm is said to be ρ-competitive. Algorithms for the cow path problem are said to be online, meaning they rely on imperfect information. For the offline version of the same problem, the location of the daisy is known beforehand. The competitive ratio of an online algorithm is defined as the worst case ratio for the distance travelled before finding the daisy between the online algorithm and an optimal offline algorithm. An optimal online algorithm is one that minimizes this competitive ratio. By viewing the problem from a game-theory perspective, Beck and Newman[2] were the first to show that a 9-competitive deterministic algorithm exists and is optimal among deterministic algorithms. Since then, many different proofs have been found that give this same result.

In the paper describing the problem, Bellman also posed the question of what would happen if the cow has a probability $0 < p \leq 1$ of finding the daisy each time it reached its location. Little research has been done for this problem. Heukers [5] and Maduro [6] have found some algorithms that function better than the optimal algorithm for the "normal" cow path problem. However, nothing is known about the form of an optimal algorithm

*Email: m.c.vos@student.utwente.nl

for this problem. No nontrivial lower bound is even known for the competitive ratio. This thesis concerns itself with finding such lower bounds.

The generalization to an arbitrary probability can be leads two different problems: the E-times cow path problem and the expected value cow path problem. For the E-times cow path problem, it is assumed that the daisy is always found at the Eth visit where E is a known strictly positive integer, and never at an earlier visit. This can be interpreted as the *average-case* discussed by both Maduro and Heukers. In this case, $E = \lceil \frac{1}{p} \rceil$. It can also be assumed that the daisy is found when the probability of it having been found passes a certain threshold. In this case, E can be obtained using the geometric distribution. For the purposes of this thesis, it is irrelevant how the E-times cow path problem is interpreted.
For the expected value cow path problem, the expected distance traveled before finding the daisy is considered. Thus, $d(x)$, the distance traveled before finding the daisy if it were to be at location x, is given by:

$$d(x) = \sum_{i=1}^{\infty} p(1-p)^{i-1} v(i,x)$$

where p is the probability of finding the daisy when visiting its location, and $v(i,x)$ is the distance travelled before the ith visit to x.
This thesis concerns the former of these two: the E-times cow path problem. The main result of this thesis is theorem 3, which gives a tight lower bound for the competitive ratio of deterministic algorithms for the case E odd. In this thesis, we first find the general form of an optimal deterministic algorithm, proceed by finding a more specific candidate algorithm and finish by showing that the competitive ratio of this candidate is a lower bound. We also find (non-tight) lower bounds for the case E even and the discrete version of the E-times cow path problem.

2 Structure of optimal algorithms

2.1 Definitions and notation

The generalization from the normal cow path problem to the E-times version requires some additional definitions. It is not immediately clear what happens when the cow turns. The turning point is visited once to make sure that turning does not increase the total amount of points visited during some time-interval. It is convenient to define a minimum distance between turns, ϵ_s. This $\epsilon_s > 0$ can be taken arbitrarily close to 0. Since it is undesirable to use ϵ_s in proofs directly, the notion of changing velocity is introduced. The algorithm can change its velocity on an interval $[a,b]$ to $\frac{1}{n} (n \in \mathbb{N} \text{ odd})$. Every visit counts as n visits and moving a distance d along the real number line with some velocity v counts as having moving a (time-)distance of $\frac{d}{v}$. The algorithm pays n visits to every point in (a,b), but only pays $\frac{n-1}{2}$ and $\frac{n+1}{2}$ visits to a and b respectively. To see that this notion of velocity is only for ease of notation and does not influence which algorithms can be created, consider the following. Partition the interval $[a,b]$ into sub-intervals $[x_i, y_i]$ (for $i = 1, \ldots, k$) of length $\frac{\epsilon_s}{|a-b|}$ and let the algorithm go n times along each sub-interval before moving on to the next sub-interval (see figure **??**). For $\epsilon_s \downarrow 0$, every visit occurs at the same moment as it would when moving with a velocity of $\frac{1}{n}$.

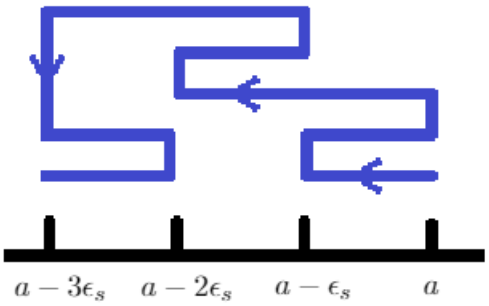

FIGURE 1: Moving from a to $a-3\epsilon_s$ with velocity $\frac{1}{3}$ (vertical movement irrelevant)

Now that we have determined exactly what is meant with an algorithm, a formal definition can be given.

Definition 2.1. Algorithm: A ρ-competitive algorithm A is a path that visits each $x \in \mathbb{R}$ at least E times after traveling at most $\rho|x|$ time. The algorithm moves up and down the real number line. The velocity of this movement is $v(t) = \frac{1}{2n+1}$ $n \in \mathbb{N}$ on time-intervals $(a, b]$. Each point attained on the interval is visited $\frac{1}{v(t)}$ times. A defined by the $\mathbb{R}^+ \to \mathbb{R}$-function $A(t)$ and its velocity function $v(t)$.

It is useful to define some other functions associated with an algorithm A. Denote by $V_A(x,t)$ the number of visits to location x in time-interval $[0,t]$. A point x is said to be saturated at t if and only if $V_A(x,t) \geq E$. The saturation time of x, $m_A(x)$, is the time of the Eth visit to x. It is often useful to consider the situation when the A crosses the origin. Let z_n be the sequence of all such times t, with $z_{k+1} > z_k$ for all k. So for all k, $\exists \epsilon > 0$ s.t. $A(z_k - \delta)A(z_k + \delta) > 0$ $\forall \delta$ s.t. $0 < \delta < \epsilon$.

To perform operations on algorithms, the notion of parts is required.

Definition 2.2. Part: A part Ω of A on some time interval (t_s, t_e) is the $(t_s, t_e) \to \mathbb{R}$ function with $\Omega(t) = A(t) \forall\ t\ \in (t_s, t_e)$.

We can perform two operations on algorithms using these parts: deletion and insertion. Let Ω be the part of A on (t_s, t_e). Then the algorithm A', obtained from A by deleting Ω is defined as follows:

$$A'(t) = \begin{cases} A(t) & \text{for } t \leq t_s \\ A(t + (t_e - t_s)) & \text{for } t > t_s \end{cases}$$

Let Ω be the part of some algorithm on (t_s, t_e). Then the algorithm A', obtained from A inserting Ω into at the point t_0 is defined as follows:

$$A'(t) = \begin{cases} A(t) & \text{for } t \leq t_0 \\ \Omega(t) & \text{for } t \in (t_0, t_0 + (t_e - t_s)) \\ A(t - (t_e - t_s)) & \text{for } t \geq t_0 + (t_e - t_s) \end{cases}$$

Clearly, removal and insertion can only be done if $\lim_{a \to t_s^+, b \to t_e^-} \Omega(a) = \Omega(b)$.
The velocity of $A'(t)$ for both deletion and insertion is the same as the velocity in the right-hand side of the equations.

Definition 2.3. Restrictive points: The *individual ratio* of a point $x \in \mathbb{R}$ for some algorithm A is defined as $\frac{m_A(x)}{|x|}$. A point is said to be *restrictive* for A if its individual ratio equals the competitive ratio of A.

For ease of notation, the constant K is introduced:

$$K := \begin{cases} E & \text{E is odd} \\ E+1 & \text{E is even} \end{cases}$$

2.2 Properties of optimal algorithms

We start our search for the structure of optimal algorithms by defining some properties of these algorithms.

Definition 2.4. Completing algorithm: Let $\beta(t)$ be the set of the left- and rightmost visited points at time t. An algorithm A is said to be completing if for all $k \geq 1$, no visited point is unsaturated at z_k, with the exception of the the left- and rightmost visited points. That is, $\forall k \in \mathbb{N}, x \in \mathbb{R} \setminus \beta(z_k))$, either $V_A(x, z_k) = 0$ or $V_A(x, z_k) \geq E$ holds. A is said to be completing until M if the above condition holds for all k s.t. $z_k \leq M$.

Our first result shows that there exist optimal algorithms that are completing up to an arbitrary point. For all practical purposes, this means that they are completing.

Lemma 1. *Let A be an algorithm and let $M \in \mathbb{R}^+$. There exists an algorithm A' s.t. $\rho(A') \leq \rho(A)$ where A' is completing until M.*

Proof. If A is completing until M, choose $A' = A$ and we are done. Assume A is not completing. Let z_k be the largest element of z_n s.t. A is completing until and including z_k. W.l.o.g. assume that A is moving to the right at z_k. Since A is not completing until z_{k+1}, $\exists x > 0$ that is visited but not saturated in time-interval $[z_k, z_{k+1}]$. Clearly, the positive axis is saturated up until and including some point y at time z_{k+1}. Now consider the sub-intervals (p_i, q_i) of $[z_k, z_{k+1}]$ with $A(p_i) = A(q_i) = y$ on which only points to the right of y are visited.

Let A' be obtained from A by removing all intervals (p_i, q_i) and then inserting those same intervals at the first time that $A(t)$ visits y after z_{k+1}. Clearly, A' is completing until z_{k+1}. To see that $\rho(A') \leq \rho(A)$, first notice that all points that get saturated outside (z_k, z_{k+2}) in A have the same saturation time for both A and A'. The values attained on $[z_{k+1}, z_{k+2}]$ by A and A' are identical, but the entirety of this interval occurs earlier for A' than for A. Thus, for all points $x' < 0$ that get saturated by A on the interval $[z_{k+1}, z_{k+2}]$, we have $m_{A'}(x') < m_A(x')$. Let c be a point to the right of y s.t. every point in the interval $(y, c]$ is visited but not saturated in the time-interval $[z_k, z_{k+1}]$ (such a point is guaranteed to exist). For all points $x^* > c$ that get saturated by A on the interval $[z_k, z_{k+1}]$, we have:

$$\frac{m_{A'}(x^*)}{|x^*|} < \frac{m_{A'}(c)}{|c|} = \frac{m_A(c)}{|c|} \leq \rho(A).$$

There is thus no point in A' with an individual ratio larger than $\rho(A)$. This gives $\rho(A') \leq \rho(A)$. Repeating the above step multiple times yields an algorithm that is completing up to an arbitrary point.

□

Our next result seems incredibly specific at first look, but turns out to be crucial. It defines exactly how an optimal algorithm moves along an unsaturated interval.

Lemma 2. *Let Ω be the part of an algorithm A on (t_s, t_e) with $A(t_s) = a$ and $A(t_e) = b$ with the following properties:*

- $|a| < |b|$ *and* $ab > 0$
- *A does not visit any point in $(a, b]$ until t_s, and has visited a exactly $\frac{E+1}{2}$ times before t_s*
- *Ω visits every point in (a, b) at least K times, visits the point a at least $\frac{E-1}{2}$ times, does not visit any point outside $[a, b]$ and visits b exactly $\frac{E-1}{2}$ times.*

Then there exists an algorithm A' with $\rho(A') \leq \rho(A)$ obtained from A by deleting Ω and inserting Ω', where Ω' moves from a to b with velocity $\frac{1}{K}$ without turning.

Proof. Let $l(\Omega) := t_e - t_s$. Since $l(\Omega) \geq K|b - a| = l(\Omega')$, we have

$$m_{A'}(c) \leq m_A(c) \quad \forall c \notin [a, b). \tag{1}$$

When determining the competitive ratio of some algorithm, one only has to consider restrictive points. A point x can only be restrictive if all points with a smaller absolute value than x are saturated before x. We define C in such a way to include all such points in the interval $(a, b]$. Let C be the set of all points in $(a, b]$ that are saturated after all points in (a, b) with a smaller absolute value have been saturated. That is, $C := \left\{ x \in (a, b] \;\middle|\; m_A(x) \geq m_A(y)\ \forall y \in (a, x] \right\}$. This gives:

$$m_A(c) \geq t_s + K|c - a| = m_{A'}(c) \quad \forall c \in C \tag{2}$$

where the first inequality follows from the facts that at the saturation time of c, every point in $(a, c]$ is visited at least E times (otherwise $c \notin C$), and every point (see remark 1) is visited an odd number of times (the algorithm moved from a to c). It follows from (1) and (2) that $\rho(A) \geq \rho(A')$. □

Remark 1. *A finite amount of turning points might be visited an even number of times. The number of turning points is finite since otherwise the length of the time-interval containing these turns would be infinite for every $\epsilon_s > 0$, which would make $\rho(A)$ infinite. These points do not influence equation* (2) *as a finite number of visits has no effect on the overall distance travelled. Furthermore, adding turning points only increases the length of the interval.*

Lemma 3. *Let A be an optimal algorithm. Let l_k the point with the largest absolute value attained by $A(t)$ on the interval (z_{k-1}, z_k) for $k \geq 1$ with $l_{-1} = l_0 = 0$. Then there exists an an algorithm A' with $\rho(A) \geq \rho(A')$ where $|l_{k+2}| > |l_k|\ \forall k \geq -1$ for A'.*

Proof. Since $|l_1|$ and $|l_2|$ are at least $m_0 > 0$, the lemma holds for $k = -1$ and $k = 0$. Now assume that $\exists k \geq 1$ such that $|l_k| > |l_{k+2}|$. Since A is completing, $V_A(x, z_k) \geq E$ $\forall x \in [0, l_k)$. Deleting the part of A on (z_{k+1}, z_{k+2}) decreases the saturation time for every point not saturated at z_{k+1}, and does not change it for all other points. Let A' be the algorithm obtained by deleting every such part from A. Then for A' we have $|l_k| \leq |l_{k+2}|\ \forall k > 1$.
Now we show how to guarantee $l_k \neq l_{k+2}$. Assume that $|l_k| < |l_{k+2}|$ for $k < n$ and that $l_n = l_{n+2}$ holds for some $n \geq 1$. First notice that $|l_{n+2}| \geq m_0$. The part of A on (z_{k+1}, z_{k+2}) is only useful if l_{n+2} is saturated on this time-interval, so we assume that this happens.

W.l.o.g. assume $l_{k+2} > 0$. After saturating l_{k+2}, A moves past the origin before saturating any points to the right of l_{k+2}, which implies $m_A(x) > m_A(l_{k+2}) + 2l_{k+2}$ for all $x > l_{k+2}$. Let A' be the algorithm obtained from A by deleting the part of A on (z_{k+1}, z_{k+2}). Then for $\epsilon_s < \frac{2l_{k+2}}{\rho(A)}$ we find:

$$\begin{aligned}
(l_{k+2} + \epsilon_s)\rho(A) &\geq m_A(l_{k+2} + \epsilon_s) > m_{A'}(l_{k+2}) + 2l_{k+2} \\
\Big(l_{k+2} + \frac{2l_{k+2}}{\rho(A)}\Big)\rho(A) &> m_{A'}(l_{k+2}) + 2l_{k+2} \\
l_{k+2}\rho(A) &> m_{A'}(l_{k+2}) \\
\rho(A) &> \frac{m_{A'}(l_{k+2})}{l_{k+2}} \qquad (3)
\end{aligned}$$

Assume $\rho(A) < \rho(A')$. Since $m_A(x) \geq m_{A'}(x)$ for $x \neq l_{k+2}$, the increase in competitive ratio for A' compared to A must be caused by l_{k+2}. However, equation (3) shows that l_{k+2} cannot be restrictive on A' if $\rho(A') > \rho(A)$, a contradiction. Thus, $\rho(A') \leq \rho(A)$. For A', $l_k < l_{k+2}$ holds for $k \leq n$. By repeating these steps multiple times, an algorithm can be obtained with $l_k < l_{k+2}$ for all k up to an arbitrarily large value. □

2.3 The optimal algorithm for E odd

Theorem 1. *For E odd, there exists a sequence of real numbers $(l_n)_{k\geq -1}$ with $l_k l_{k+1} \leq 0$, $l_{-1} = l_0 = 0$ and $|l_{k+2}| > |l_k|$, such that the algorithm A of the following form is optimal: Let $k = -1$ and iterate trough the following steps:*

- *Move from l_k to l_{k+2} with velocity $\frac{1}{E}$*
- *Move from l_{k+2} to l_{k+1} with velocity 1*
- *Set $k = k + 1$*

Proof. Let A be an optimal algorithm. By lemma 1, it can be assumed that A is completing. Let l_k point with the largest absolute value that is visited by A on (z_{k-1}, z_k). Clearly, $l_k l_{k+1} < 0$ $\forall k > 0$. By lemma 3, we can assume that $|l_{k+2}| > |l_k|$ $\forall k \geq -1$.

Consider the values $A(t)$ attains on the interval (z_{k-1}, z_k) for $k \geq 1$. Since $V_A(x, z_{k-1}) \geq E$ $\forall x \in [0, l_{k-2})$, it is irrelevant how often points in that interval are visited. It should be clear that removing any turning points and setting the velocity to 1 in this interval will not increase the saturation time for any point, and lower it for many. This gives $A(z_{k-1} + |l_{k-2}|) = A(z_k - |l_{k-2}|) = l_{k-2}$. Since A is completing, it is furthermore known that A visits each x in $J := (l_{k-2}, l_k)$ at least E times on the time-interval $I := (z_{k-1} + |l_{k-2}|, z_k - |l_{k-2}|)$ and $A(t)$ is contained to $[l_{k-2}, l_k]$ on I.

We now prove that $A(t)$ is optimal if it attains each value on J exactly E times on the time-interval $(z_{k-1} + |l_{k-2}|, z_k - |l_k|)$ with $A(z_k - |l_k|) = l_k$, and then moves with velocity 1 from l_k to l_{k-2} without turning. Every point in J is visited at least E times and every point (with the exception of a finite number of points) is visited exactly an even amount of times. Thus, $(E+1)|J|$ is the minimal length for I, meaning that this technique minimizes the saturation time for all x saturated after z_k, with a possible exception for l_k.
Consider the amount of visits to l_k. Currently, $_k$ is visited $\frac{E+1}{2}$ times in (z_{k-1}, z_k). It is necessary to prove that visiting l_k more often does not result in a lower competitive ratio. W.l.o.g. assume $l_k > 0$. The quickest way to visit l_k more often without visiting any point

to the right of l_k is to move back and forth on the interval $(l_k - \epsilon_s, l_k)$ after the $\frac{E+1}{2}$-th visit to l_k. Since all of the points in this interval are already saturated, the algorithm might as well move back and forth along the interval $(l_k, l_k + \epsilon_s)$ instead. But that algorithm is not completing, and can be made completing again without increasing the competitive ratio by deleting the part that was just inserted and inserting it at the first visit to l_k after $t = z_k$. Thus the version of the algorithm where l_k is visited more often can be transformed into an equivalent or better algorithm that visits l_k only $\frac{E-1}{2}$ times on (z_{k-1}, z_k).
Now consider all points saturated in (z_{k-1}, z_k). By lemma 2, A has the structure specified in the theorem. By induction, it can be assumed that l_{k-2} has been visited $\frac{E+1}{2}$ times at z_{k-1}. Thus, this technique minimizes the saturation time of l_{k-2}, as it pays the remaining $\frac{E-1}{2}$ visits as soon as possible.
Like before, it is only necessary to consider restrictive points. Consider $C := \left\{ x \in J \middle| \ m_A(x) \geq m_A(y) \ \forall y \ \in (l_{k-2}, x) \right\}$. Since visiting every point in some interval of length l exactly D times takes Dl time, we have:

$$m_A(x) \geq z_{k-2} + |l_{k-2}| + E(|x - l_{k-2}|) \quad \forall \ x \in \ C. \tag{4}$$

If A is in the form of the theorem, (4) holds with equality $\forall x \in C$. Thus, the form specified in the theorem is optimal. □

One can see that the form of the optimal algorithm for E odd is very similar to the form of the optimal algorithm for the normal cow path problem, which suggests that we might be able to modify existing proofs for the lower bound of the normal cow path problem to find a lower bound for the competitive ratio of the E-times cow path problem.

2.4 The optimal algorithm for E even

For E odd, there was no way around having to pay $E + 1$ visits to each point that is saturated on (z_{k-1}, z_k). For the case E even, this is not the case. It is possible to move from l_k to l_{k+2} with velocity $\frac{1}{E-1}$ and then back with velocity one, thus only paying E visits to each point. This is advantageous in the long term as points that are saturated after z_k can only benefit from reducing the length of (z_{k-1}, z_k). It is, however, disadvantageous in the short term, as the points in (l_k, l_{k+2}) are saturated in reverse order, causing points close to l_k have a large saturation time. Minimizing the competitive ratio of an algorithm requires balancing these short and long term advantages. This can be done by including a balance point b_{k+2} in the interval (l_k, l_{k+2}), resulting in the following structure:

Theorem 2. *For E even, there exist real sequences $(l_n)_{k \geq -1}$ and $(b_n)_{k \geq 1}$ with $l_k l_{k+1} \leq 0, b_k l_k \geq 0, l_{-1} = l_0 = 0$, $|l_{k+2}| > |l_k|$ and $|l_{k+2}| \geq |b_{k+2}| \geq |l_k|$, s.t. the algorithm A of the following form is optimal:*
Let $k = -1$ and iterate trough the following steps:

- *Move from l_k to b_{k+2} with velocity $\frac{1}{K}$*
- *Move from b_{k+2} to l_{k+2} with velocity $\frac{1}{E-1}$*
- *Move from l_{k+2} to l_{k+1} with velocity 1*
- *set $k = k + 1$*

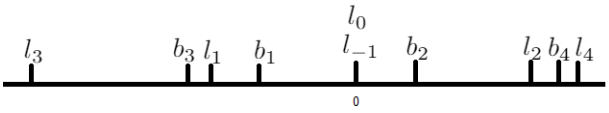

FIGURE 2: Possible values for b_n and l_n

Proof. Let A be an optimal algorithm. By Lemma 1, it can be assumed that A is completing. Let l_k point with the largest absolute value that is visited by A on (z_{k-1}, z_k). Clearly, $l_k l_{k+1} < 0 \ \forall k > 0$. By lemma 3, we can assume that $|l_{k+2}| > |l_k| \ \forall k \geq 0$.

Now consider the values $A(t)$ attains on the interval $[z_{k-1}, z_k]$. Again, it is irrelevant how often points in $[0, l_{k-2})$ are visited. So it can be assumed that:

$$A(z_{k-1} + |l_{k-2}|) = A(z_k - |l_{k-2}|) = l_{k-2}.$$

Since A is completing, it visits each x in $J := (l_{k-2}, l_k)$ at least E times. Consider the time-interval $I := (z_{k-1} + |l_{k-2}|, z_k - |l_{k-2}|)$. Denote with $l(I)$ the length of I. We first find an upper and lower bound for $l(I)$:

$$E|l_k - l_{k-2}| \leq l(I) \leq (E+2)|l_k - l_{k-2}|. \tag{5}$$

To see that the second inequality holds, consider an algorithm A in the form of the theorem with $b_n = l_k$. This algorithm moves from l_{k-2} to l_k with a velocity of $\frac{1}{K}$ and moves back with velocity one. In this case, the second inequality holds with equality. Now consider an algorithm A' which is identical to A outside I, but $l(I)$ is larger for A' than for A. For any point x outside J, we have $m_A(x) \leq m_{A'}(x)$. Furthermore, increasing the number of visits to l_k is useless for the same reason as for E odd.
The form of the theorem on I minimizes the saturation time for l_{k-2} (same argument as for E odd). For points in J, it is only necessary to consider restrictive points. Again, consider the set $C := \left\{ x \in J \middle| \ m_A(x) \geq m_A(y) \ \forall y \ \in (l_{k-2}, x) \right\}$. At the time of saturation of x, every point (with the exception of a finite number of points) in $(l_{k-2}, x]$ has been visited an odd number of times (since the algorithm moved from l_{k-2} to x), and every point has been visited at least E times since $x \ \in C$. Thus, every point in this interval was visited at least K times at time $m_{A'}(x)$. This gives:

$$m_{A'}(x) \geq z_{k-2} + |l_{k-2}| + K|x - l_{k-2}| = m_A(x).$$

Thus, $\rho(A) \leq \rho(A')$, which proves (5).

To see that the form of the theorem is optimal for every value of $l(I)$ that satisfies (5), let $l(I) = L$ for $L \ \in \big[E|l_k - l_{k-2}|, (E+2)|l_k - l_{k-2}|\big]$.

Let A be an algorithm that behaves according to the theorem on I, with b_k such that $l(I) = L$. Let A' be an algorithm with $A(t) = A'(t) \ \forall t \notin I$ that saturates all points in (l_{k-2}, l_k) on I. Since $l(I) = (E+2)|l_k - l_{l-2}| - 2|l_k - b_k|$ is the same for both algorithms, we have $m_A(x) = m_{A'}(x) \ \forall x \notin J + \{l_{k-2}\}$.
Now consider $x \in C$. We first prove by contradiction that $|x| \leq |b_k|$. Assume $\exists x \in C$ s.t. $|x| > |b_k|$. At $t = m_{A'}(x)$ every point in (l_{k-2}, x) has been visited exactly an odd number of times and at least E times. Since $A'(z_k - |l_{k-2}|) = l_{k-2}$, every point in (l_{k-2}, x) is visited at least $(E+2)$ times on I. This gives a contradiction:

$$L \geq (E+2)|l_k - l_{k-2}| - 2|l_k - x| > (E+2)|l_k - l_{k-2}| - 2|l_k - b_k| = L.$$

Thus $x \leq b_k$, which gives:

$$m_{A'}(x) \geq (E+1)|x - l_{k-2}| = m_A(x). \tag{6}$$

Equation (6) holds for all $x \in C$, so $\rho(A) \leq \rho(A')$. There exists an optimal algorithm in the form of the theorem. □

3 Finding lower bounds

3.1 Tight lower bound for E odd

The proof for the lower bound for E odd is a generalization of the proof for the lower bound of the normal cow path problem (the case $E = 1$) by Bernhard Fuchs et al [4]. Their paper considers the discrete version of this problem while we consider the continuous case. This explains why evaluating the equations in this thesis at $E = 1$ does not give the equations from their paper. Many of the proofs and lemmas of this chapter will follow the same general arguments as theirs.

Since we have proven that there exists an optimal algorithm in the form of theorem 1 for E odd, we only consider algorithms of that form. Only very specific points have to be considered to evaluate the competitive ratio of such an algorithm. The point x which has the largest ratio individual ratio of all points saturated in the time-interval (z_k, z_{k+1}) is m_0 for $k = 1$ and l_{k-1} for $k \geq 2$ (it is always the first point saturated after z_k). Let A be an algorithm in the form of theorem 1. Denote with M_k the minimum possible saturation time of the first point saturated after z_k (the saturation time of this point for an optimal offline algorithm) and denote with L_k the saturation time of this point for A. $\rho(A)$ can then be found by solving $\max_{k \geq 1} \frac{L_k}{M_k}$. This gives:

$$\begin{aligned} M_1 &= m_0 \\ M_{k+1} &= |l_k| \quad \forall k \geq 1 \end{aligned}$$

Lemma 4. *At the first saturation after z_k the online algorithm has travelled $L_1 = M_1 + (E+1)M_2$ if $k = 1$ and*

$$2\sum_{2}^{k-1} M_i + (E+2)M_k + (E+1)M_{k+1} \quad \text{if } k \geq 2$$

Proof. $L_1 = (E+1)|l_1| + m_0$ which equals the expression above. For $k \geq 2$ we get

$$\begin{aligned} L_k =& 2\sum_{i=1}^{k} |l_i| + (E-1)(|l_k| + |l_{k-1}|) + M_k \\ =& 2\sum_{i=2}^{k+1} M_i + (E-1)(M_{k+1} + M_k) + M_k \\ =& 2\sum_{i=2}^{k-1} M_i + (E+2)M_k + (E+1)M_{k+1}. \end{aligned}$$

□

These equations can be used to determine the competitive ratio of specific algorithms. We first determine a candidate value for the competitive ratio of an optimal algorithm using the following lemma:

Lemma 5. *Let A be an algorithm for the E-times cow path problem with E odd in the form of theorem 1. Furthermore assume that there is a constant ratio $r > 1$ between consecutive values of M_k (thus $M_{k+1} = rM_k \ \forall k \geq 1$). Then for m_0 sufficiently large, A is optimal among all such algorithms with fixed r if and only if:*

$$\rho(A) = 3 + 2E + 2\sqrt{2(E+1)}$$

Proof. For m_0 sufficiently large, $\frac{L_1}{M_1} < \rho(A)$, so we can ignore $k = 1$. Expressing the sum in the equation for $k \geq 2$ from lemma 4 in terms of M_1 yields:

$$\begin{aligned} L_k &= 2\sum_{2}^{k-1} r^{i-1}M_1 + (E+2)M_k + (E+1)M_{k+1} \\ &= 2M_1\sum_{1}^{k} r^{i-1} - 2(M_1 + M_k) + (E + 2 + (E+1)r)M_k \end{aligned}$$

This summation is a partial sum of an infinite geometric series. Rewriting gives:

$$\begin{aligned} L_k &= 2M_1\frac{1-r^k}{1-r} - 2M_1 + (E + (E+1)r)M_k \\ &= \frac{2M_1 - 2rM_k}{1-r} - 2M_1 + (E + (E+1)r)M_k \\ &= \frac{2rM_k}{r-1} - (2 + \frac{2}{r-1})M_1 + (E + (E+1)r)M_k \\ &= \Big(\frac{2r}{r-1} + E + (E+1)r\Big)M_k - (2 + \frac{2}{r-1})M_1 \end{aligned}$$

Thus, $\frac{L_k}{M_k}$ is increasing on k for $k > 1$ and converges to $\rho = \frac{2r}{r-1} + E + (E+1)r$. This function attains its minimum at $r = 1 + \sqrt{\frac{2}{E+1}}$ for fixed E. Substituting this value for r into the equation for $\rho(A)$ yields the desired equation. □

Let $g(E) := 3 + 2E + 2\sqrt{2(E+1)}$, the candidate-$\rho$ found in the previous lemma. It turns out that $g(E)$ is the the lowest possible competitive ratio for the E-times cow path problem for E odd, not just for algorithms with a fixed ratio between consecutive M_k. To prove this, assume that a $(g(E) - \epsilon)$-competitive algorithm exists for some $\epsilon > 0$. It can be assumed that this algorithm is optimal, and thus that it has the structure specified in theorem 1. The algorithm is thus characterized by M_k and L_k.
Let σ_k and α_k be such that:

$$L_k = (g(E) - \sigma_k)M_k \quad \text{and} \quad M_{k+1} = (1 + \alpha_k)M_k. \tag{7}$$

We introduce the potential function

$$\Phi_k := \sigma_k + (E+1)\alpha_k. \tag{8}$$

It can be shown that this potential approaches $-\infty$ for algorithms with a competitive ratio smaller than $g(E)$, which will lead to a contradiction. Observe that:

$$\begin{aligned}\Phi_1 &= g(E) - (E+1) + \frac{(E+1)M_2 - M_1 - (E+1)M_2}{M_1} \\ &= g(E) - (E+2) = 1 + E + 2\sqrt{2(E+1)}.\end{aligned}$$

The recursion for Φ_k follows:

Lemma 6.

$$\Phi_{k+1} = \Phi_k - \Delta_k \text{ with } \Delta_k = \frac{\alpha_k\sigma_k + (1+E)\alpha_k^2 + 2 - 2\sqrt{2(E+1)}\alpha_k}{1+\alpha_k}. \tag{9}$$

Proof. Use lemma 4 to obtain:

$$(g(E) - \sigma_{k+1})M_{k+1} - (g(E) - \sigma_k)M_k = L_{k+1} - L_k = (E+1)M_{k+2} + M_{k+1} - EM_k$$

Write M_{k+1} and M_{k+2} in terms of M_k and divide by M_k to obtain:

$$\begin{aligned}(g(E) - \sigma_{k+1})(1+\alpha_k) - (g(E) - \sigma_k) &= (E+1)(1+\alpha_k)(1+\alpha_{k+1}) + (1+\alpha_k) - E \\ (g(E) - 1 - \sigma_{k+1})(1+\alpha_k) - (g(E) - E - \sigma_k) &= (E+1)(1+\alpha_k)(1+\alpha_{k+1}) \\ -(\sigma_{k+1} + (E+1)\alpha_{k+1})(1+\alpha_k) &= 2 - \sigma_k - (g(E) - 2 - E)\alpha_k\end{aligned}$$

The left-hand side of the equation equals $-(1+\alpha_k)\Phi_{k+1}$. Multiply by -1 and rewrite the right-hand side such that it includes $(1+\alpha_k)\Phi_k$ to obtain:

$$\begin{aligned}\Phi_{k+1}(1+\alpha_k) &= (\sigma_k + (E+1)\alpha_k)(1+\alpha_k) - \left(\alpha_k\sigma_k + (E+1)\alpha_k^2 - 2\sqrt{2(E+1)}\alpha_k\right) \\ \Phi_{k+1}(1+\alpha_k) &= \Phi_k(1+\alpha_k) - \left(\alpha_k\sigma_k + (E+1)\alpha_k^2 - 2\sqrt{2(E+1)}\alpha_k\right).\end{aligned}$$

Dividing by $(1+\alpha_k)$ yields the desired equation. □

Clearly, $\alpha_k > -1\ \forall k \geq 1$. Furthermore, if a $(g(E) - \epsilon)$-competitive algorithm were to exist, then it would maintain $\sigma_k \geq \epsilon > 0$. This leads to a contradiction if we can show that Δ_k is bounded below by some positive constant. A constant that gives a somewhat neat expression is $\frac{1}{1+2E}$.

Lemma 7. *If $\sigma_k \geq 0$, we have $\Delta_k \geq \frac{1}{1+2E}\sigma_k$. If furthermore, $\sigma_k \geq \epsilon > 0$ for all k then $\Delta_k \geq \frac{1}{1+2E}\epsilon > 0$ for all k.*

Proof.

$$\begin{aligned}\Delta_k - \frac{\sigma_k}{1+2E} &= \frac{\alpha_k\sigma_k + (1+E)\alpha_k^2 + 2 - 2\sqrt{2(E+1)}\alpha_k}{1+\alpha_k} - \frac{\sigma_k}{1+2E} \\ &= \frac{\frac{2E\sigma_k\alpha_k}{1+2E} - \frac{\sigma_k}{1+2E} + (1+E)\alpha_k^2 + 2 - 2\sqrt{2(E+1)}\alpha_k}{1+\alpha_k}.\end{aligned} \tag{10}$$

Since $1+\alpha_k > 0$, it is sufficient to prove that the numerator of this fraction is positive for $\sigma_k > 0$. The numerator attains its minimum value at $\alpha_k = \frac{2}{\sqrt{2E+2}} - \frac{E\sigma_k}{(E+1)(2E+1)}$ for fixed E, σ_k. For this value of α_k, the numerator of (10) is positive (see appendix 6.2). □

Applying lemma 7 to equation (9) shows that $\lim_{k\to\infty} \Phi_k = -\infty$. But $\lim_{k\to\infty} \Phi_k = \lim_{k\to\infty} \sigma_k + (E+1)\alpha_k \geq \epsilon + (E+1)(-1)$, a contradiction. Thus, no $(g(E)-\epsilon)$-competitive algorithm exists for the E-times discrete cow path problem for E odd. This yields the main result of this thesis:

Theorem 3. *For any online algorithm for the E-times cow path problem with E odd, $\rho \geq 3 + 2E + 2\sqrt{2(E+1)}$.*

This lower bound is clearly tight, as we obtained this value for the first time by analyzing competitive ratios of algorithms. An example of such an optimal algorithm is given in the following corollary:

Corollary 1. *The algorithm in the form of theorem 1 with $|l_1| = m_0$ and $|l_k| = \left(1 + \sqrt{\frac{2}{E+1}}\right)|l_{k-1}|$ is optimal.*

3.2 Lower bound for E even

To find the optimal ratio between l_k and b_{k+2} for algorithms in the form of theorem 2, one likely has to perform some quite involved function analysis, which is beyond the scope of this thesis. This ratio becomes an additional unknown which greatly influences the values of L_k. Furthermore, for arbitrary values of b_k, the saturation times for both b_k and l_k might effect the competitive ratio, instead of just l_k. Thus, it is impossible to use the above techniques to find a tight lower bound for E even, as long as this ratio is unknown. To find a lower bound, we design a theoretical algorithm which is guaranteed to behave better than any possible algorithm. The following is known about the part of an algorithm in the form of theorem 2 on (z_k, z_{k+1}):

$$z_{k+1} - z_k \geq 2|l_{k+2}| + (E-2)|l_{k+2} - l_k| \tag{11}$$

$$m_A(x) \geq z_k + |l_k| + E|x - l_k| \quad \forall x \in (l_k, l_{k+2}). \tag{12}$$

It is clear that a theoretical algorithm for which both equations hold with equality for all positive k would be better than any real algorithm. We assume that such an algorithm exists and call it A. This assumption costs us the tightness of the lower bound. Such an algorithm leads to the following version of lemma 4:

$$L_1 = M_1 + E|l_1| = M_1 + EM_2 \tag{13}$$

and for $k \geq 2$

$$\begin{aligned} L_k &= 2\sum_{i=1}^{k} |l_i| + (E-2)(|l_k| + |l_{k-1}|) + M_k \\ &= 2\sum_{i=2}^{k+1} M_i + (E-2)(M_k + M_{k+1}) + M_k \\ &= 2\sum_{i=2}^{k-1} M_i + (E+1)M_k + EM_{k+1}. \end{aligned}$$

These are the exact equations for the optimal algorithm for the E'-times cow path problem for $E' = E - 1$. Since E' is odd, theorem 3 gives us the lower bound for the competitive ratio of this problem. Substituting $E = E' + 1$ into the equation of this theorem gives yields a lower bound for the E-times cow path problem for E even:

Theorem 4. *For any algorithm for the E-times cow path problem with E even,* $\rho \geq 1 + 2E + 2\sqrt{2E}$.

This lower bound for the case E even provides additional structure to the form of an optimal algorithm for this problem:

Corollary 2. *For an optimal algorithm of the form in theorem 2, the following inequalities hold for all k:*

$$|b_{k+2}| < |l_k| + |l_{k+2} - l_k|\Big(2 + \frac{2\sqrt{2E}}{E}\Big)^{-1} < |l_k| + \frac{1}{2}|l_{k-2} - l_k|. \tag{14}$$

Proof. Let A be an algorithm in the form of theorem 2. Increasing $|b_k|$ has a neutral or negative effect on the saturation time for all points outside the interval (b_k, l_k). Increasing b_k should therefore only be done if there exists a pint in (b_k, l_k) that is restrictive on A. Denote with $x^+ \in \mathbb{R}$ a point with an infinitesimally larger absolute value than x with the same sign as x. Since the point b_k^+ is the last point saturated in (b_k, l_k), and also has the smallest absolute value of all these points, it is the only point in (b_k, l_k) that can be restrictive. Similarly, l_{k-2} is the only such point in the interval $[l_{k-2}, b_k]$. Thus, if $\frac{m_A(b_k^+)}{|b_k^+|} < \frac{m_A(l_{k-2})}{|l_{k-2}|} \leq r$, then $|b_k|$ is too large. It is therefore guaranteed that $\frac{m_A(b_k^+) - m_A(l_{k-2})}{|b_k^+ - l_{k-2}|} \geq r \geq 1 + 2E + 2\sqrt{2E}$. Denote with d the distance $|l_k - l_{k-2}|$ and with d_1 the distance $|b_k^+ - l_{k-2}|$, then:

$$\begin{aligned}
\frac{m_A(b_k^+) - m_A(l_{k-2})}{|b_k^+ - l_{k-2}|} &\geq 1 + 2E + 2\sqrt{2E} \\
\frac{Ed + d_1}{d_1} &\geq 1 + 2E + 2\sqrt{2E} \\
d_1 &\leq \frac{d}{2 + \frac{2\sqrt{2E}}{E}}
\end{aligned}$$

replacing d and d_1 by their expressions in terms of l_k and b_k^+ and making the inequality strict by replacing b_k^+ with b_k yields the desired equation. □

4 Discretization

In some applications it is more useful to consider the discrete version of the E-times cow path problem. In this version of the problem, the daisy is guaranteed to be found at an integer point. Turning around is also only allowed at these points. It turns out that we can quite easily find a lower bound for the competitive ratio of this problem with the tools developed for the continuous version.

It is desirable to define the discretization in such a way that it is easy to see that a discretized version of theorem 1 or 2 holds. Since we essentially have $\epsilon_s = 1$, the concept of velocity becomes slightly more involved. To easily define this velocity, we require that every interval on which an algorithm moves with a velocity unequal to one has an even length. If this is not the case for some algorithm, the entire algorithm can be upscaled by doubling the value of m_0 and every turning point to make all these distances even length. This upscaling is allowed for the purposes of this thesis, as we are only looking for a lower bound. In this thesis, it is implicitly assumed that all such intervals have even length.

Since $\epsilon_s = 1$, moving with a velocity $v > 0, v \in \mathbb{N}$ means going back and forth between a pair of points until both are visited $\frac{1}{v}$ times and then moving on to the next pair. As an example, assume that an algorithm moves with a velocity of $\frac{1}{n}$ on $(a, b]$. Then for $x \in (a, b]$ the kth ($k \leq n$) visit to x occurs after $(|x - a| - 1)n + 2k - 1$ if $|x - a|$ is odd and after $(|x - a| - 2)n + 2k$ if $|x - a|$ is even. Note that unlike in the continuous version, a is not visited at all, and b is visited n times on the interval. An observant reader might have noticed that with this definition of velocity, there is no reason why $\frac{1}{v}$ should be odd. Indeed, in the discrete version an algorithm may move with a velocity of $\frac{1}{n}, n \in \mathbb{N}$. This has the advantage that there is no difference between optimal algorithms for E even and odd for $E > 1$. Like with E even for the continuous problem, the ability to move with a velocity of $\frac{1}{E-1}$ for $E > 1$ requires balancing using balance points b_k. This results in the following form for an optimal algorithm:

Corollary 3. *For the discrete E-times cow path problem with $E > 1$, there exist integer sequences $(l_n)_{n \geq -1}, (b_n)_{n \geq 1}$ with $l_k l_{k+1} \leq 0$, $l_{-1} = l_0 = 0$, $|l_{k+2}| > |l_k|$ and $|l_k| \leq |b_{k+2}| \leq |l_{k+2}|$ s.t. the algorithm A of the following form is optimal:*
Let $k = -1$ and iterate trough the following steps:

- *Move from l_k to b_{k+2} with velocity $\frac{1}{E}$*
- *Move from b_{k+2} to l_{k+2} with velocity $\frac{1}{E-1}$*
- *Move from l_{k+2} to l_{k+1} with velocity 1*
- *set $k = k + 1$*

The proof for this corollary is omitted, as it is easy to check that all the proofs in this thesis for the continuous even case are also applicable to the discrete case, provided some small modifications are made concerning the saturation of the elements of l_n and b_n.

In order to find a lower bound on the competitive ratio, we make the same concessions as for the continuous even case. We assume that there exist an algorithm A that follows the general form of theorem 3 and visits every point on the interval $(l_k, l_{k+1}]$ exactly E times on (z_{k-1}, z_k) while saturating every point in this interval in order of absolute value from smallest to largest $\forall k \geq 1$. Again, such an algorithm cannot exist and has a smaller competitive ratio than any existing algorithm.

Remark 2. *This assumption is closer to reality than in the continuous even case. For an algorithm in the form of corollary 3, every point in l_k to b_{k+2} is visited $E + 1$ times on (z_{k+1}, z_{k+2}), while an algorithm in the form of theorem 2 pays $E + 2$ visits to each point in that interval. The fact that we find the same lower bound for these two problems does therefore not suggest that the competitive ratios of optimal algorithms for these two problems would be the same.*

In such an interval, the point x with the worst individual ratio is $l_k \pm 1$ (the point away from the origin from l_k). Like before, denote with M_k the minimum possible saturation time for x and denote with L_k the saturation time of x in A. This gives:

$$\begin{aligned} M_1 &= m_0 + 2E - 2 \\ M_{k+1} &= |l_k| + 2E - 1 \quad \forall k \geq 1 \end{aligned}$$

and L_k is given by

$$L_1 = E|l_1| + M_1 = M_1 + EM_2 - E(2E - 1) \tag{15}$$

for $k = 1$ and

$$\begin{aligned}
L_k =& 2\sum_{i=1}^{k} |l_i| + (E-2)(|l_k| + |l_{k-1}|) + M_k \\
=& 2\sum_{i=2}^{k+1} \Big(M_i - (2E-1)\Big) + (E-2)(M_{k+1} + M_k - 4E + 2) + M_k \\
=& 2\sum_{i=2}^{k-1} M_i + (E+1)M_k + EM_{k+1} - (4E-2)(E-2) - k(4E-2).
\end{aligned}$$

for $k \geq 2$. We need to create a result analogous to that of lemma 5. Let $f(E) = (4E - 2)(E-2) - k(4E-2)$ to find:

$$\begin{aligned}
L_k &= 2\sum_{2}^{k-1} r^{i-1} M_1 + (E+1)M_k + EM_{k+1} - f(E) \\
&= 2M_1 \sum_{1}^{k} r^{i-1} - 2(M_1 + M_k) + (E + 1 + Er)M_k - f(E) \\
&= 2M_1 \frac{1-r^k}{1-r} - 2M_1 + (E - 1 + Er)M_k - f(E) \\
&= \frac{2rM_k}{r-1} - (2 + \frac{2}{r-1})M_1 + (E - 1 + Er)M_k - f(E) \\
&= \Big(\frac{2r}{r-1} + E - 1 + Er\Big)M_k - (2 + \frac{2}{r-1})M_1 - f(E).
\end{aligned}$$

Thus, $\frac{L_k}{M_k}$ is increasing on k for $k > 1$ and converges to $\rho = \frac{2r}{r-1} + E - 1 + Er$. Substituting $E' = E - 1$ yields the exact equation found in lemma 5 for the continuous odd case. Using the same r results in a competitive ratio of $1 + 2E + 2\sqrt{2E}$. It can be proven that this competitive ratio is a lower bound by following the same steps as before. Since this proof is so similar to the continuous case, it is omitted in this thesis, but can be found in appendix 6.1.

In this chapter, we ignored the case $E = 1$, as an optimal algorithm for this case does not have the form specified in theorem 3. It is, however. known that $3 + 2\sqrt{2}$ is a lower bound on the competitive ratio for the normal discrete cow path problem [4]. We can formalize these results in the following theorem.

Theorem 5. *For any algorithm for the discrete E-times cow path problem,* $\rho \geq 1 + 2E + 2\sqrt{2E}$.

5 Suggestions for further research

Although this thesis completely solves the case E odd, the general problem of a non-optimal seeker is still far from solved. Nothing is yet known about the expected value cow path problem mentioned in the introduction of this thesis. A nontrivial lower bound on the competitive ratio for this problem is likely the most widely applicable result still absent. The lower bound for E even found in corollary 2 is likely good enough for most applications (especially for large values of E). It would be nice, however, to be able to consider the E-times cow path problem completely solved, which requires finding a tight lower bound for E even as well. One way to find a larger lower bound for the continuous even case, is to

look more into the discrete version, as it can be shown that any ρ-competitive algorithm for the E-times continuous problem corresponds to a ρ-competitive algorithm for the E-times discrete problem.

References

[1] A. Beck. On the linear search problem. *Israel Journal of Mathematics*, 2(4):221–228, dec 1964.

[2] A. Beck and D. J. Newman. Yet more on the linear search problem. *Israel Journal of Mathematics*, 8(4):419–429, 1970.

[3] R. Bellman. Problem 63-9, An Optimal Search. *SIAM Review*, 5(3):274, 1963.

[4] B. Fuchs, W. Hochstättler, and W. Kern. Online matching on a line. *Theoretical Computer Science*, 332(1-3):251–264, feb 2005.

[5] F. Heukers. Searching with Imperfect Information, Technical report, University of Twente, jun 2017.

[6] G. Maduro. Comparing algorithms for the cow-path problem with a non-optimal seeker, Technical report, University of Twente. 2018.

6 Appendices

6.1 Appendix 1: Proof for discrete lower bound

Let $g(E) := 1 + E + 2\sqrt{2E}$ with α_k and σ_k as before. Then

$$\Phi_k := \sigma_k + E\alpha_k. \tag{16}$$

It can be shown that this potential approaches $-\infty$ for algorithms with a competitive ratio smaller than $g(E)$, which is a contradiction. Observe that:

$$\begin{aligned}\Phi_1 &= g(E) - E + \frac{EM_2 + E(2E-1) - M_1 - EM_2}{M_1} \\ &= g(E) - (E+1) + \frac{E(2E-1)}{m_0 + 2E - 1} \approx E + 2\sqrt{2E}\end{aligned}$$

assuming m_0 to be sufficiently large. The recursion for Φ_k follows:

Lemma 8.

$$\Phi_{k+1} = \Phi_k - \Delta_k + \frac{4E-2}{M_{k+1}} \text{ with } \Delta_k = \frac{\alpha_k\sigma_k + E\alpha_k^2 + 2 - 2\sqrt{2E}\alpha_k}{1+\alpha_k}. \tag{17}$$

Proof. Use lemma of σ_k and α_k to obtain:

$$\begin{aligned}&(g(E) - \sigma_{k+1})M_{k+1} - (g(E) - \sigma_k)M_k = L_{k+1} - L_k \\ &= EM_{k+2} + M_{k+1} + (1-E)M_k - 4E + 2\end{aligned}$$

Write all M_{k+1} and M_{k+2} in terms of M_k and divide by M_k to obtain:

$$\begin{aligned}(g(E) - 1 - \sigma_{k+1})(1+\alpha_k) - (g(E) + (1-E) - \sigma_k) &= E(1+\alpha_k)(1+\alpha_{k+1}) - \frac{4E-2}{M_k} \\ -(\sigma_{k+1} + E\alpha_{k+1})(1+\alpha_k) &= 2 - \sigma_k - (g(E) - 1 - E)\alpha_k - \frac{4E-2}{M_k} \\ \Phi_{k+1}(1+\alpha_k) - \Phi_k(1+\alpha_k) - \frac{4E-2}{M_k} &= -\left(\alpha_k\sigma_k + 2 + E\alpha_k^2 - 2\sqrt{2E}\alpha_k\right)\end{aligned}$$

dividing by $(1+\alpha_k)$ gives the desired equation. □

Notice that M_{k+1} can be made arbitrarily large (by the choice of m_0), so $\Phi_{k+1} \approx \Phi_k - \Delta_k$. If a $(g(E) - \epsilon)$-competitive algorithm were to exist, then it would maintain $\sigma_k \geq \epsilon > 0$. Replacing E by $E+1$ in the expression for Δ_k in equation (17) yields the expression for Δ_k in equation (9). This gives the following alternative of lemma 7:

Lemma 9. *If $\sigma_k \geq 0, E \geq 2$, we have $\Delta_k \geq \frac{1}{2E-1}\sigma_k$. If furthermore, $\sigma_k \geq \epsilon > 0$ for all k then $\Delta_k \geq \frac{1}{2E-1}\epsilon > 0$ for all k.*

6.2 Appendix 2: additional proof for lemma 7

We want to show that

$$\frac{2E\alpha_k\sigma_k}{2E+1} - \frac{\sigma_k}{2E+1} + (1+E)\alpha_k^2 + 2 - 2\alpha_k\sqrt{2(E+1)} \text{ with } \alpha_k = \frac{\sqrt{2}}{\sqrt{E+1}} - \frac{E\sigma_k}{(E+1)(2E+1)}.$$

(18)

is strictly positive for $\sigma_k > 0, E \geq 1$. It is sufficient to show that it is positive for σ_k very small, as this would show that a competitive ratio of $g(E) - \sigma_k$ is impossible. Thus, we can view this equation as a polynomial in terms of σ_k and look at the coefficients of the lowest power first. Setting $\sigma_k = 0$ gives

$$(1+E) * \Big(\frac{\sqrt{2}}{\sqrt{E+1}}\Big)^2 + 2 - 2\sqrt{2(E+1)} * \frac{\sqrt{2}}{\sqrt{E+1}} = 2 + 2 - 4 = 0 \tag{19}$$

so the coefficient of the constant is 0. Now consider the linear coefficient:

$$\begin{aligned}
&\frac{2E}{2E+1} * \frac{\sqrt{2}}{\sqrt{E+1}} - \frac{1}{2E+1} + (1+E) * 2\frac{-\sqrt{2}E}{\sqrt{E+1}(E+1)(2E+1)} \\
&\qquad - 2\sqrt{2(E+1)} * \frac{-E}{(E+1)(2E+1)} \\
&= \frac{1}{2E+1}\Big(\frac{2E\sqrt{2}}{\sqrt{E+1}} - 1 - \frac{2E\sqrt{2}}{\sqrt{E+1}} + 2\sqrt{2(E+1)}\frac{E}{E+1}\Big) \\
&= \frac{1}{2E+1}\Big(-1 + 2\sqrt{2(E+1)}\frac{E}{E+1}\Big) = \frac{1}{2E+1}\Big(\frac{(2\sqrt{2(E+1)}-1)E-1}{E+1}\Big) \\
&\geq \frac{1}{2E+1}\Big(\frac{3E-1}{E+1}\Big) > 0
\end{aligned}$$

Thus, the numerator of the fraction is always positive for small values of $\sigma_k > 0$.