Integer Cow-path Problem and Simple Robot Street Search

Azadeh Tabatabaei[*] Farehe Soheil[†] Mohammad Aletaha[‡] Mohammad Ghodsi[§]

Abstract

In this paper, we revisit the well-known cow-path problem and introduce a new variation called Integer Cow-path Problem (ICP). In the general cow-path problem, w rays with one common end-point and a robot standing on the end-point are given. A target point is put along one of the rays and can be detected only when it is reached by the robot. The robot has to find the target by traversing the rays starting from the end-point. In the ICP, the robot is restricted to take an integer number of steps. The goal is to design a strategy for the robot to find the target such that the length of the traveled path is as small as possible. We present a randomized strategy that gives an upper bound on the competitive ratio for the ICP. Furthermore, as an application of this variation, we study the Simple Robot Street Search problem and give a randomized strategy that is inspired from the strategy for the ICP.

1 Introduction

The problem of searching in unknown geometric environments is a fundamental problem in the fields of computational geometry, robotics, and online algorithms [13]. The cow-path problem is a well-known problem that has been studied by mathematicians and computer scientists [3, 6]. Many geometric search problems have applied the cow-path search algorithms.

In the general cow-path problem, we are given $w \geq 2$ rays with a common end-point s and a target point t that is put along one of the rays, see Figure 1(a). A searcher (robot), which is not aware of the location of the target t, must find it. The searcher can only move along the rays and cannot detect t before reaching it. The name «cow-path» comes from the scenario in which, metaphorically a cow is searching for her calf in w concurrent paths [15]. Note that the cow-path problem is also called star-search or ray-search. Furthermore, when $w = 2$, it is called searching on a line or linear search, see Figure 1(b).

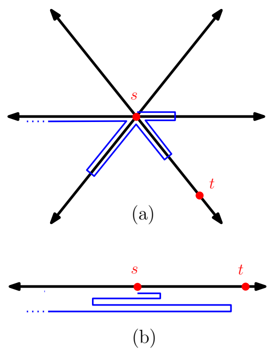

Figure 1: (a) An example of cow-path problem with $w = 6$. The robot's search path is shown in blue (b) Searching on a line ($w = 2$).

Searching in an unknown environment can be seen as an online problem since the robot does not have access to the information about the position of the target and must decide in an online manner. The notion of the competitive analysis is proposed for measuring the performance of the online algorithms [17]. In the cow-path problem, the competitive ratio is the length of the path traveled by the robot from the start point s to the target point t over the shortest path from s to t. A strategy is α-competitive if its competitive ratio is at most α.

The distances in this problem are expressed in steps. A *step* is the unit of measurement in the literature [6]. We revisit the general cow-path problem in a way that the robot is limited to take an integer number of steps along the rays. While in the general problem such a limitation does not exist and the robot can take any real number of steps, see Figure 2. Notice that like the general cow-path problem, the searcher stops whenever it achieves the target and does not go further even if it is in a middle of a step. We name this new variation *Integer Cow-path Problem* (ICP) and will study it in section 2.

The motivation behind considering such a variation (ICP) is to use simpler and cheaper searching agents (robots). The robots which can take any real number of steps need powerful movement capabilities, while the robots whose number of steps are limited to integers have simpler and less powerful movement capabilities. More powerful and complicated dynamic abilities will always lead to more expenses and costs. A stepper-

[*]Department of Computer Engineering, University of Science and Culture, Iran. a.tabatabaei@usc.ac.ir
[†]Department of Computer Engineering, Sharif University of Technology, Iran. soheil@ce.sharif.edu
[‡]Department of Computer Engineering, Sharif University of Technology, Iran. mohammadaletaha@ce.sharif.edu
[§]Sharif University of Technology and Institute for Research in Fundamental Sciences (IPM), Iran. ghodsi@sharif.edu

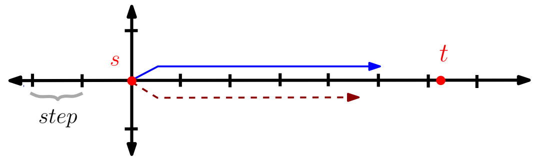

Figure 2: Example of the cow-path problem with $w = 4$. Traversing an integer number of steps (shown in blue) in the ICP, and traversing a real number of steps (shown in dashed line) in the general problem. The target t is placed $n \in \mathbb{R}$ $(n \geq 1)$ steps away from s.

motor (or stepping-motor) is an electric motor that instead of rotating continuously, rotates in a number of discrete equal steps. Then, one can command the motor to move or hold at each step without any position sensor for feedback (see [2] for more details about stepper-motors). A stepper-motor makes the robot take only an integer number of steps when attached to one. Note that we configure the step-size of the robot's stepper-motor to be equal to the step-size of the problem. In the general cow-path problem the searcher might take any real number of steps along the rays. Hence, we have introduced the ICP to find the same target in the same environment with a simpler robot equipped with a stepper-motor.

As an application of the ICP, we study the *Simple Robot Street Search* problem. This problem refers to searching for a specific target in a particular polygonal environment (a street polygon) using a minimal sensing robot. We will discuss it properly in section 3.

1.1 Our contribution

In this paper, we introduce and study the Integer Cow-path Problem (ICP) and present the first randomized search strategy for it. In fact, this strategy gives an upper bound on the competitive ratio of the ICP. Our randomized strategy for the ICP is β-competitive where

$$\beta = \min_{r \in \mathbb{Z}, r > 1} \left\{ \frac{2\,(r^w - 1)}{w(r-1)\ln r} + (w+1)\cdot(0.56) + 1 \right\}$$

and w is the number of rays, given as the input of the problem. We have summarized the competitive ratio of our strategy for some values of w in Table 1.

Furthermore, we study the Simple Robot Street Search problem and show that in the worst case, this problem can be seen as the ICP with $w = 2$. So, we give a 6.29-competitive randomized strategy that uses the strategy for the ICP as a subroutine.

2 Integer Cow-path Problem (ICP)

In this section, first, we express the Integer variation of the Cow-path Problem (ICP) in detail and mention some related works. Then, we present a randomized strategy for the ICP and finally discuss the performance of the proposed strategy in the remaining of this section.

w	β
2	6.29
3	9.98
4	14.63
5	22.25
6	35.22

Table 1: The competitive ratio β for some values of w for the *integer* cow-path problem.

2.1 Problem Definition

Given $w \geq 2$ concurrent rays sharing a common endpoint s, and a robot $\mathcal{R}$ standing on s. A target point t is put $n \in \mathbb{R}$ $(n \geq 1)$ steps away from s along an arbitrary ray. The robot $\mathcal{R}$ moves in a round-to-round manner and can only move back and forth along the rays. At each round $i \in \mathbb{Z}^{0+}$, $\mathcal{R}$ takes $d_i \in \mathbb{N}$ steps on ray $l_i \in \{0, 1, ..., w-1\}$ and if it does not find t, returns back to s and advances on round $i+1$. So, we use a function $S(i) = (d_i, l_i)$ to formulate the robot's moving strategy at round i.

Note that before t is reached, $\mathcal{R}$ always takes an integer number of steps. Since the target's distance can be non-integer $(n \in \mathbb{R})$, $\mathcal{R}$ is allowed to take only one non-complete step during its travel, which is when it encounters t. The robot $\mathcal{R}$ immediately stops as soon as it reaches t even in a middle of a step. Let D denote the length of the path (in steps) traveled by $\mathcal{R}$ from s to find t. The problem refers to designing an online strategy minimizing $C = D/n$, where C is the competitive ratio of the strategy.

Note that a lower bound on n is necessary for having a bounded competitive ratio. Otherwise, the competitive ratio will be unbounded in the worst case. Consider t is placed at distance $\epsilon > 0$ steps from s on ray j. Now, if the robot takes λ steps in the first round, on any other ray $k \neq j$, then the competitive ratio is at least $\frac{\lambda}{\epsilon}$ and it might be unbounded. Hence, like many previous works, we assume $n \geq 1$.

2.2 Related works

The general cow-path problem was first proposed by Bellman [8] and then, studied by mathematicians like Beck and Newman [7] and Gal [12] who gave solutions to this problem. After that, the problem was rediscovered by computer scientists. Baeza-Yates et al. [6] and Kao et al. [15] gave optimal deterministic and randomized strategies for the general problem, respectively. The deterministic strategy in [6] is λ-competitive where

$$\lambda = 2 \cdot \frac{w^w}{(w-1)^{w-1}} + 1$$

and the randomized strategy in [15] is γ-competitive where

$$\gamma = \min_{r>1} \left\{ \frac{2}{w} \cdot \frac{1 + r + r^2 + \cdots + r^{w-1}}{\ln r} + 1 \right\}.$$

Here, we mention the values of λ and γ for some w in Table 2. So far, several variations of the general problem

w	λ	γ
2	9	4.6
3	14.5	7.74
4	19.97	10.85
5	25.42	13.95
6	30.86	17.04

Table 2: The competitive ratios λ and γ for some values of w for the *general* cow-path problem.

have been introduced and studied by researchers, including moving target, having turn cost, the existence of lower and upper bounds on the target's location, and maximum clearance [5, 9, 10, 11].

2.3 Algorithm for the ICP strategy

Here, we present a randomized strategy for the ICP. Our strategy is a modified version of the SmartCow strategy by Kao et al. [15] such that it always generates integer values for d_i at each round i. In fact, we apply the idea of rounding using the ceiling function. We denote this strategy by *IntegerCow* and express it as an algorithm in Algorithm 1. The analysis of the algorithm can be

Algorithm 1: IntegerCow

```
Compute a random permutation $\sigma$ of
  $\{0, 1, ..., w-1\}$
Compute the best value for $r$
Choose $\varepsilon$ u.a.r.[1] from $[0, 1)$
$d \leftarrow r^{\varepsilon}$
$i \leftarrow 0$
while the target t is not achieved do
    Move along ray $\sigma(i \bmod w)$ up to $\lceil d \rceil$ step(s)
    Move back to the start point $s$
    $d \leftarrow r \cdot d$
    $i \leftarrow i + 1$
end
```

represented in terms of $r > 1$ which is a constant real value. Also, r approximately determines by what factor the number of steps in the next round should be more than the current round. We will show how to find the best value for r in the analysis section. Compared to the SmartCow algorithm, the modification is slight but it makes the analysis more complicated and non-trivial.

2.4 Analysis

Now, we obtain the competitive ratio of the IntegerCow algorithm. Since this algorithm is randomized, its competitive ratio is defined as the *expected* distance traveled by the robot over the actual distance between s and t. The algorithm is designed to move the robot $\lceil r^{i+\varepsilon} \rceil$ steps from s on round $i \geq 0$, i.e. $S(i) = (\lceil r^{i+\varepsilon} \rceil, l_{\sigma(i \ mod \ w)})$. As we noted earlier, n denotes the distance between s and t in the worst case. Let $k \in \mathbb{Z}^{0+}$ and $0 \leq \delta < 1$ such that $n = r^{k+\delta}$. Suppose t lies on ray l_j and let m be the *first round* where $\mathcal{R}$ travels a distance of *at least* $\lceil r^k \rceil$ steps from s on l_j.

Depending on the random order of rays in σ, the value of m in the *worst case* satisfies $k \leq m \leq k + w - 1$ (It is possible for m to take values less than k because of the ceiling function, but this only decreases the competitive ratio). Let D be the random variable denoting the overall distance traveled by $\mathcal{R}$ to find t. We denote the competitive ratio of the algorithm by C that is computed as $C = E[D]/n$. To obtain C, we need to calculate $E[D]$ for which there are two cases:

Case 1: When $m = c \geq k + 1$, $\mathcal{R}$ certainly finds t at round c and the expected value of D is calculated as

$$\begin{aligned} E[D|m=c] &= E\left[2\sum_{i=0}^{c-1} \lceil r^{i+\varepsilon} \rceil + n\right] \\ &= E\left[2\sum_{i=0}^{c-1} (r^{i+\varepsilon} + \lceil r^{i+\varepsilon} \rceil - r^{i+\varepsilon}) + n\right] \\ &= 2\sum_{i=0}^{c-1} \left(E\left[r^{i+\varepsilon}\right] + E\left[\lceil r^{i+\varepsilon} \rceil - r^{i+\varepsilon}\right]\right) + n. \end{aligned}$$

Now, we define functions $f(r, i)$ and $f_M(r)$ as follows:

$$f(r,i) = E\left[\lceil r^{i+\varepsilon} \rceil - r^{i+\varepsilon}\right] = \int_0^1 (\lceil r^{i+\varepsilon} \rceil - r^{i+\varepsilon}) d\varepsilon,$$

$$f_M(r) = \max_{i \in \mathbb{Z}^{0+}} f(r, i).$$

So, the expression for $E[D|m=c]$ can be written as

$$\begin{aligned} E[D|m=c] &= 2\sum_{i=0}^{c-1} \left(E\left[r^{i+\varepsilon}\right] + f(r,i)\right) + n \\ &\leq 2\sum_{i=0}^{c-1} \left(E\left[r^{i+\varepsilon}\right] + f_M(r)\right) + n \end{aligned}$$

[1]uniformly at random

$$= \frac{2(r^c-1)}{r-1}\cdot E[r^{\varepsilon}] + 2c\cdot f_M(r) + n$$
$$= \frac{2(r^c-1)}{r-1}\cdot \int_0^1 r^{\varepsilon} d\varepsilon + 2c\cdot f_M(r) + n$$
$$= \frac{2(r^c-1)}{r-1}\cdot (\frac{r-1}{\ln r}) + 2c\cdot f_M(r) + n$$
$$= \frac{2(r^c-1)}{\ln r} + 2c\cdot f_M(r) + n.$$

Case 2: When $m = k$, depending on the values of ε and δ, $\mathcal{R}$ may find t on round k or may fail. In case of failure, it continues to search and finally finds t on round $k+w$. We classify this case into three events as follows:

F_1**:** When $\varepsilon > \delta$, then trivially $\lceil r^{k+\varepsilon}\rceil > r^{k+\delta}$ and $\mathcal{R}$ will find t on round k.

F_2**:** When $\varepsilon \leq \delta$ and $\lceil r^{k+\varepsilon}\rceil \geq r^{k+\delta}$, again $\mathcal{R}$ will find t on round k.

F_3**:** When $\varepsilon < \delta$ and $\lceil r^{k+\varepsilon}\rceil < r^{k+\delta}$, $\mathcal{R}$ will not find t on round k. So, it continues searching and will find t on round $k+w$.

So, the expected traveled distance in case 2 is

$$E[D|m=k] = Pr(F_1)\cdot E\left[2\sum_{i=0}^{k-1}\lceil r^{i+\varepsilon}\rceil + n \mid F_1\right]$$
$$+ Pr(F_2)\cdot E\left[2\sum_{i=0}^{k-1}\lceil r^{i+\varepsilon}\rceil + n \mid F_2\right]$$
$$+ Pr(F_3)\cdot E\left[2\sum_{i=0}^{k+w-1}\lceil r^{i+\varepsilon}\rceil + n \mid F_3\right].$$

Similar to calculations of case 1, we have

$$E[D|m=k] \leq Pr(F_1)\cdot\left[\frac{2(r^k-1)}{r-1}\right]\cdot E[r^{\varepsilon}|F_1]$$
$$+ Pr(F_2)\cdot\left[\frac{2(r^k-1)}{r-1}\right]\cdot E[r^{\varepsilon}|F_2]$$
$$+ Pr(F_3)\cdot\left[\frac{2(r^{k+w}-1)}{r-1}\right]\cdot E[r^{\varepsilon}|F_3]$$
$$+ Pr(F_1)\cdot 2k\cdot f_M(r) + Pr(F_2)\cdot 2k\cdot f_M(r)$$
$$+ Pr(F_3)\cdot 2(k+w)\cdot f_M(r) + n.$$

To calculate the expectations on r^{ε}, we define $\alpha = \min x \in [0,\delta]$ such that $\lceil r^{k+x}\rceil \geq r^{k+\delta}$. Notice that $r > 1$ and $0 \leq \alpha \leq \delta$. By the definition of α, for event F_2 we have $\alpha \leq \varepsilon \leq \delta$ and for event F_3 we have $0 \leq \varepsilon < \alpha$. So, in events F_2 and F_3, we have $r^{\alpha} \geq 1$ and $r^{\alpha} \leq r^{\delta}$, respectively. Now, we can compute the conditional expectations as follows:

$$E[r^{\varepsilon}|F_1] = \int_{\delta}^{1}\frac{r^{\varepsilon}d\varepsilon}{Pr(F_1)} = \frac{r-r^{\delta}}{Pr(F_1)\cdot\ln r},$$
$$E[r^{\varepsilon}|F_2] = \int_{\alpha}^{\delta}\frac{r^{\varepsilon}d\varepsilon}{Pr(F_2)} = \frac{r^{\delta}-r^{\alpha}}{Pr(F_2)\cdot\ln r} \leq \frac{r^{\delta}-1}{Pr(F_2)\cdot\ln r},$$
$$E[r^{\varepsilon}|F_3] = \int_{0}^{\alpha}\frac{r^{\varepsilon}d\varepsilon}{Pr(F_3)} = \frac{r^{\alpha}-1}{Pr(F_3)\cdot\ln r} \leq \frac{r^{\delta}-1}{Pr(F_3)\cdot\ln r}.$$

Putting all these together, we have

$$E[D|m=k] \leq 2(k+w)\cdot f_M(r) + n$$
$$+ \frac{2\cdot\left[(r-r^{\delta})(r^k-1) + (r^{\delta}-1)(r^{k+w}+r^k-2)\right]}{(r-1)\ln r}.$$

Thus, considering both cases and knowing that $Pr(m=j) = \frac{1}{w}$ for any $k \leq j \leq k+w-1$, the total expected traveled distance equals to

$$E[D] = \sum_{j=k}^{k+w-1} Pr(m=j)\cdot E[D|m=j]$$
$$= \frac{1}{w}\cdot E[D|m=k] + \sum_{j=k+1}^{k+w-1}\frac{1}{w}\cdot E[D|m=j].$$

By the results of case 1 and case 2, we get

$$E[D] \leq \frac{2\cdot\left[(r-r^{\delta})(r^k-1) + (r^{\delta}-1)(r^{k+w}+r^k-2)\right]}{w(r-1)\ln r}$$
$$+ \frac{2}{w}(k+w)\cdot f_M(r) + \frac{n}{w}$$
$$+ \frac{1}{w}\sum_{c=k+1}^{k+w-1}\left(\frac{2(r^c-1)}{\ln r} + 2c\cdot f_M(r) + n\right).$$

After simplification, we have

$$E[D] \leq \frac{2\cdot\left[r^{k+\delta+w} - r^k - r^{\delta} - rw + w + 1\right]}{w(r-1)\ln r}$$
$$+(2k+w+1)\cdot f_M(r) + n.$$

Since $k,\delta \geq 0$ and $r > 1$, we have $-r^k < -1$, $-r^{\delta} \leq -1$, and $-rw < -w$. So,

$$E[D] \leq \frac{2\left(r^{k+\delta+w}-1\right)}{w(r-1)\ln r} + (2k+w+1)\cdot f_M(r) + n.$$

Since $k,\delta \geq 0$ and $n = r^{k+\delta}$, we have

$$C \leq \frac{2\left(r^{k+\delta+w}-1\right)}{r^{k+\delta}w(r-1)\ln r} + \frac{(2k+w+1)\cdot f_M(r)}{r^{k+\delta}} + 1. \qquad (1)$$

Using (1), we demonstrate the following lemma.

Lemma 1 *The competitive ratio C is upper bounded by*

$$\min_{r>1}\left\{\frac{2\left(r^w-1\right)}{w(r-1)\ln r} + (w+1)\cdot f_M(r) + 1\right\}$$

Proof. We are interested to obtain an upper bound on C and we want this upper bound to be as small as possible. The number of rays, w, is the input of the problem. The value of r is a parameter of the algorithm for which we determine its value. After deciding about the value of r, the adversary determines the value of k and δ to maximize the competitive ratio C. As a result, we must choose the proper value for r to assure that for a fixed value of δ and k, given by the adversary, still we obtain a minimum upper bound for C. So, we consider the right hand side of (1) as a function of r. This function has a minimum in $r \in [1, \infty)$ and it can be verified by taking the first derivative of the $r.h.s$ function with respect to r. Now, inspiring from (1), we define function $F_{rhs}(k, \delta)$ as below

$$\min_{r>1} \left\{ \frac{2\left(r^{k+\delta+w}-1\right)}{r^{k+\delta} w(r-1)\ln r} + \frac{(2k+w+1)\cdot f_M(r)}{r^{k+\delta}} + 1 \right\}.$$

It is obvious that $C \leq F_{rhs}(k, \delta)$. The adversary can determine the value of k and δ to maximize $F_{rhs}(k, \delta)$. It can be proven that $F_{rhs}(k, \delta)$ is maximized when $k, \delta = 0$ (by taking the derivatives, one can show this function is non-increasing with respect to $k, \delta \geq 0$). So, we can conclude that $C \leq F_{rhs}(0, 0)$ and this completes the proof of lemma. $\square$

From now on, we focus on proving an upper bound for $f_M(r)$. Since we use Ramanujan's bounds for $\ln(n!)$ further, we claim it as the following theorem.

Theorem 2 [4] *For* $\ln(n!)$*, the following lower and upper bounds hold:*

$$\ln(n!) < n\ln(n) - n + \tfrac{1}{6}\ln\left(8n^3 + 4n^2 + n + \tfrac{1}{30}\right) + \tfrac{1}{2}\ln(\pi),$$
$$\ln(n!) > n\ln(n) - n + \tfrac{1}{6}\ln\left(8n^3 + 4n^2 + n + \tfrac{1}{100}\right) + \tfrac{1}{2}\ln(\pi).$$

Now, we state the following lemma which gives an upper bound for $f_M(r)$.

Lemma 3 *For all integers* $r > 1$*, we get* $f_M(r) < 0.56$.

Proof. According to definition of $f(r, i)$, we have

$$\begin{aligned} f(r,i) &= \int_0^1 \left(\lceil r^{i+\varepsilon} \rceil - r^{i+\varepsilon}\right) d\varepsilon \\ &= \int_0^1 \lceil r^{i+\varepsilon} \rceil \, d\varepsilon - \int_0^1 r^{i+\varepsilon} d\varepsilon. \end{aligned} \tag{2}$$

The second part of (2) is easily calculated as

$$\int_0^1 r^{i+\varepsilon} d\varepsilon = \frac{r^{i+1} - r^i}{\ln r} \tag{3}$$

In order to compute the first part of (2), let $z = i + \varepsilon$, then $dz = d\varepsilon$ which gives

$$\begin{aligned} \int_0^1 \lceil r^{i+\varepsilon} \rceil \, d\varepsilon &= \int_i^{i+1} \lceil r^z \rceil \, dz \\ &= \int_0^{i+1} \lceil r^z \rceil \, dz - \int_0^i \lceil r^z \rceil \, dz. \end{aligned}$$

Let

$$g(i) = \int_0^i \lceil r^z \rceil \, dz,$$

then

$$\int_0^1 \lceil r^{i+\varepsilon} \rceil \, d\varepsilon = g(i+1) - g(i). \tag{4}$$

Considering the graph of $f(x) = \lceil r^x \rceil$ in Figure 3, we can compute $g(i)$ by summing up the areas of the blue rectangles.

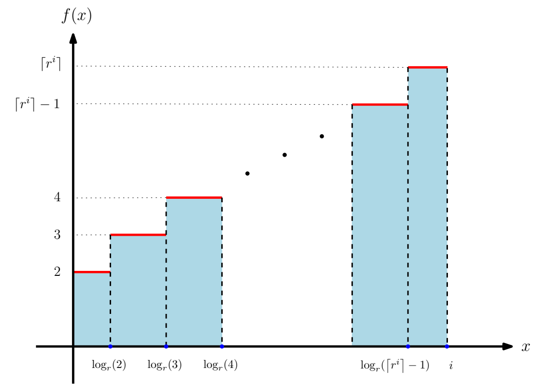

Figure 3: The function $f(x) = \lceil r^x \rceil$ is shown in red. The blue region is $g(i)$.

According to Figure 3, $g(i)$ can be calculated as:

$$\begin{aligned} g(i) = \int_0^i \lceil r^z \rceil \, dz = \sum_{j=2}^{\lceil r^i \rceil - 1} j \cdot \left(\log_r(j) - \log_r(j-1)\right) \\ + \lceil r^i \rceil \cdot \left(i - \log_r\left(\lceil r^i \rceil - 1\right)\right). \end{aligned}$$

The above expression is hard to simplify for all real values of $r > 1$ because of the ceiling functions. So, by considering only integer values for r, one can reduce the difficulties induced by the ceiling functions. Since i is an integer, by letting r be an integer we will have $\lceil r^i \rceil = r^i$ and $i = \log_r \lceil r^i \rceil$. Note that r is a parameter of the algorithm not the input of the problem, so we are allowed to decide its value. Henceforth, we study r only for integer values, i.e. $r > 1$ and $r \in \mathbb{Z}$. So the above summation can be simplified as

$$\begin{aligned} g(i) &= \sum_{j=2}^{r^i} j \cdot \left(\log_r(j) - \log_r(j-1)\right) \\ &= r^i \cdot \log_r(r^i) - \sum_{j=2}^{r^i - 1} \log_r(j) \\ &= i \cdot r^i - \log_r\left((r^i - 1)!\right). \end{aligned} \tag{5}$$

By applying (5) in (4) and using (3), we can calculate (2) as follows:

$$\begin{aligned} f(r,i) &= (i+1)r^{i+1} - \log_r\left((r^{i+1}-1)!\right) \\ &\quad - (i)r^i + \log_r\left((r^i-1)!\right) - \frac{r^{i+1}-r^i}{\ln r} \\ &= r^i\left((r-1)\,i + r - \frac{r-1}{\ln r}\right) - \log_r\left(\frac{(r^{i+1}-1)!}{(r^i-1)!}\right) \\ &= r^i\left((r-1)\,i + r - \frac{r-1}{\ln r}\right) - \log_r\left(\frac{r^{i+1}!}{r\cdot r^i!}\right) \\ &= r^i\left((r-1)\,i + r - \frac{r-1}{\ln r}\right) \\ &\quad - \frac{1}{\ln r}\left(\ln\left(r^{i+1}!\right) - \ln\left(r^i!\right)\right) + 1. \end{aligned}$$

Using inequalities in Theorem 2, we get

$$\begin{aligned} f(r,i) &< 1 + r^i\left((r-1)\,i + r - \frac{r-1}{\ln r}\right) \\ &+ \frac{1}{\ln r}\left(r^i\ln\left(r^i\right) - r^i - r^{i+1}\ln\left(r^{i+1}\right) + r^{i+1}\right) \\ &- \frac{1}{\ln r}\left(\frac{1}{6}\ln\left(8\left(r^{i+1}\right)^3 + 4\left(r^{i+1}\right)^2 + r^{i+1} + \frac{1}{100}\right)\right) \\ &+ \frac{1}{\ln r}\left(\frac{1}{6}\ln\left(8\left(r^i\right)^3 + 4\left(r^i\right)^2 + r^i + \frac{1}{30}\right)\right). \quad (6) \end{aligned}$$

We denote the right hand side of (6) by $f_{ub}(r,i)$. According to the definition of $f_M(r)$, any upper bound on $f(r,i)$ for all $i \in \mathbb{Z}^{0+}$ yields an upper bound on $f_M(r)$. If we fix r (for all integers $r > 1$), then $f_{ub}(r,i)$ is non-increasing with respect to $i \in [0,\infty)$ (Section A in the Appendix covers the proof of this statement). So, the maximum value of $f_{ub}(r,i)$ happens at $i = 0$.

The function $f_{ub}(r,0)$ is non-increasing with respect to r (proof is similar). Therefore, for all integers $r > 1$ the maximum value of $f_{ub}(r,0)$ happens at $r = 2$ and it approximately equals to 5.557. So, we conclude that for all integers $r > 1$, we have

$$f_M(r) < f_{ub}(2,0) < 0.56.$$

This completes the proof of lemma. □

Now, by Lemma 1 and Lemma 3, we conclude the following theorem which is the main contribution of this paper.

Theorem 4 *The IntegerCow algorithm for the ICP is β-competitive where*

$$\beta = \min_{r\in\mathbb{Z},r>1}\left\{\frac{2\left(r^w-1\right)}{w(r-1)\ln r} + (w+1)\cdot(0.56) + 1\right\}.$$

By Theorem 4, for any $w \geq 2$, Algorithm 1 computes the best value for r (denoted by r^*) which minimizes the above function. For some values of w, the values for r^* and their corresponding β are given in Table 3.

w	r^*	β
2	4	6.29
3	2	9.98
4	2	14.63
5	2	22.25
6	2	35.22

Table 3: The values of r^* and β for some w

3 Simple Robot Street Search

In this section, we introduce an application of the ICP called Simple Robot Street Search and give a randomized strategy for it, inspired by the ICP strategy.

3.1 Preliminaries

A simple polygon P with two distinct vertices s and t is called a *street* if the clockwise chain (L_{chain}) and counter-clockwise chain (R_{chain}) which are constructed on the polygon from s to t, are mutually weakly visible. In other words, each point on L_{chain} must see at least one point on R_{chain} and vice versa, see Figure 4.

The set of points that are visible from a point q in P is called the visibility polygon of q, denoted by $vis(q)$. An edge of $vis(q)$ which does not belong to the boundary of P is called a *gap* (sometimes called window), see Figure 4. Each gap g is induced by a reflex vertex called $ref(g)$ and a region of P is hidden on one of its sides called pocket of g. We assume a gap g is oriented such that its start point is $ref(g)$. The gap g is called a l-gap (r-gap), denoted by L (R), if the pocket of g lies on its left (right) side, see Figure 4.

A *simple robot* $\mathcal{B}$ is a point robot which can only detect gaps in a cyclical order, see Figure 4. As a technical point of view, it is equipped with a sensor which detects the order of discontinuities in depth information (gaps) in its visibility polygon. A simple robot can only

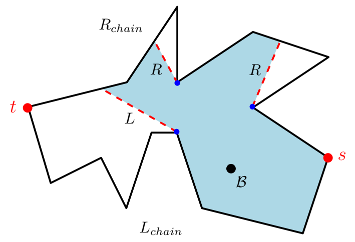

Figure 4: A street polygon with respect to s and t, R_{chain} and L_{chain}, two r-gaps and one l-gap. The visibility polygon of the robot $\mathcal{B}$ is shown in blue.

move toward gaps and the target t when it becomes visible. The unit of simple robot's movement is called *step* which is a constant distance that has been specified by the robot's manufacturer. Technically speaking, a stepper-motor is put on the robot which makes it move in discrete equal distances (steps). Furthermore, a simple robot has a *stopping function* that, whenever called, forces the robot to stop immediately even if in a middle of a step. For example, at the point that the target t becomes visible, one can call its stopping function to stop the robot and prevent it from going further in the wrong direction.

3.2 Problem Definition

We are given a street polygon P with point s and t and a simple robot $\mathcal{B}$. The robot $\mathcal{B}$ is placed at s and moves through P in an arbitrary number of steps. We assume a step is small enough compared to the scale of P. The robot is presumed to take an integer number of steps except when the strategy forces the robot to stop. The goal is to reach the target t starting from s such that the traversed path by $\mathcal{B}$ is as short as possible.

3.3 Related Works

In 1992, Klein [16] introduced street polygons and proposed a 5.73-competitive strategy for the street search problem using a 360°-vision robot. The 360°-vision robots can detect any vertex and edge of the polygon, measure any angle and distance between objects, and move freely in any direction. After several improvements, finally, Icking et al. [14] presented the optimal $\sqrt{2}$-competitive strategy for this problem.

The limited sensing model (gap sensor) that a simple robot is equipped with, was introduced by Tovar et al [22]. Note that unlike 360°-vision robots, a simple robot's vision and movement are strictly limited. For the first time, Tabatabaei et al. [20] used the simple robots for the street search problem and gave an 11-competitive strategy using auxiliary tools called pebbles. After that, Tabatabaei et al. [19] and Wei and Tan [23] independently presented the optimal 9-competitive deterministic strategies for the Simple Robot Street Search problem. In this paper, we study the latter problem and give a randomized 6.29-competitive strategy. The cooperation of two robots [1] and minimizing the number of turns in street searching [18], and searching in generalized streets [21] are other works that used simple robots.

3.4 Motion primitives

Gaps are maintained in a data structure called S-GNT [20]. While $\mathcal{B}$ is moving, combinatorial changes called critical events, occur in its visibility polygon which update S-GNT. There are four critical events:

- *Appearance* and *Disappearance* of gaps occur when the robot crosses the inflection rays.
- *Merge* and *Split* of gaps occur when the robot crosses the bitangent complements.

as illustrated in Figure 5. An inflection ray is the ex-

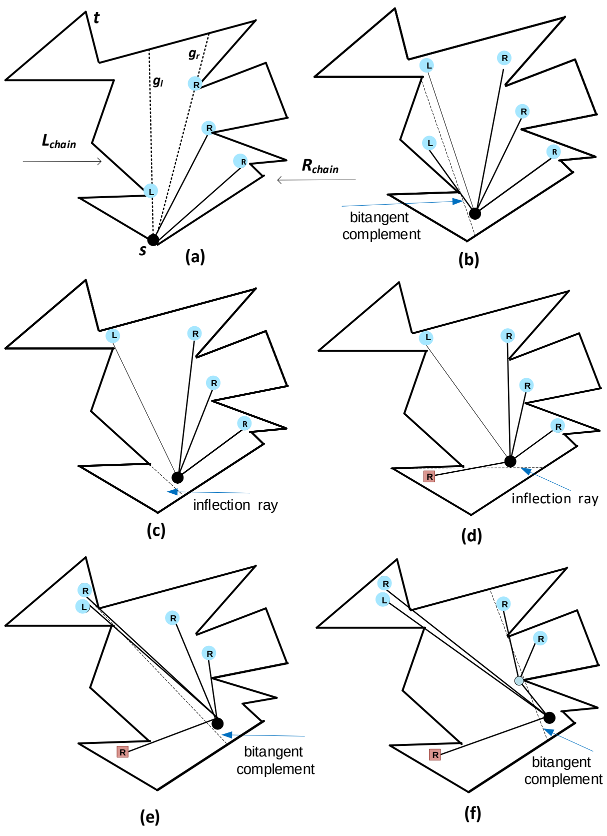

Figure 5: A street polygon and the dynamical changes of the gaps as the robot moves. The black circle is the location of $\mathcal{B}$. Pink squares and blue circles denote primitive and non-primitive gaps, respectively. (a) Existing gaps at s. (b) A split event. (c) A disappearance event. (d) An appearance event. (e) Another split event. (f) A merge event.

tension of the hidden edge of a gap into the interior of the polygon, see Figure 5(c). A bitangent complement is a line that is tangential to two reflex vertices of the polygon, see Figure 5(b). A new gap whose pocket was formerly visible is called *primitive*. Otherwise, it is called *non-primitive*, see Figure 5. We are only interested in probing non-primitive gaps since the region behind a non-primitive gap had never been visible to the robot before. We define g_l to be the rightmost non-primitive l-gap, and g_r to be the leftmost non-primitive r-gap, see Figure 5(a). As the robot moves, the critical events may dynamically change g_l and g_r in a way that one the followings happens:

- *Uni-gap:* if there exists only one of g_r or g_l or they are collinear, a uni-gap occurs, see Figure 6(a).
- *Funnel:* When both g_r and g_l exist, we say a *funnel* is created, see Figure 6(b). Also, by the robot's

movement, a funnel may end or a new one may start. We refer to the point, in which a funnel ends, a *critical point* of that funnel, see point 2 in Figure 6(b).

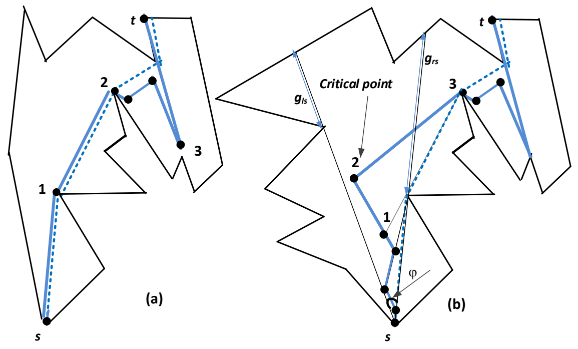

Figure 6: The bold path is the robot's search path, the dotted path is the shortest path. (a) There is only g_r. (b) Both g_r and g_l at the start point s and a funnel case. The angle φ between the gaps, is the opening angle.

In a funnel case, as the robot moves toward g_l or g_r, the following events update the location of g_l and g_r as well as the location of the funnel.

1. When g_r/g_l splits into itself and a new r-gap/l-gap. Then the g_r/g_l will be replaced by this new r-gap/l-gap (point 1 in Figure 6(b)).

2. When g_r/g_l splits into itself and a new l-gap/r-gap, then this new l-gap/r-gap will be set as g_l/g_r. This point is a critical point in which the funnel ends (point 2 in Figure 6(b)).

3. When g_l or g_r disappears, a critical point has been achieved and the funnel ends (point 3 in Figure 6(a)).

Note that the split and disappearance events may occur concurrently (point 3 in Figure 6(b)).

3.5 Strategy

When the target t is not visible, it is hidden behind one of the gaps. Otherwise, the robot could see it. The following theorem states that when t is not visible, it is behind one of g_l or g_r. So, we can ignore other ones and focus only on g_l and g_r.

Theorem 5 [20] *While t is not visible, it is behind g_l or g_r.*

In a uni-gap case, where there exists only one of g_r or g_l or they are collinear, according to Theorem 5, t is behind that gap, see Figure 6(a). But at each funnel, where both g_r and g_l exist, the robot is not aware behind which of g_r or g_l the target t is hidden. So, usually, a detour from the shortest path (an extra undesirable longer path) is unavoidable. Notice that potentially, there might be more than one critical point for a funnel. But, as soon as the robot achieves a critical point, according to its strategy, the funnel ends. So, $\mathcal{B}$ achieves only one critical point for each funnel.

Now, we demonstrate our strategy. Until $\mathcal{B}$ does not see the target t, it continues to search. If t becomes visible, $\mathcal{B}$'s stopping function will be called. So, $\mathcal{B}$ immediately stops and moves directly toward t. Before t becomes visible, at each point of the search path, two cases might happen:

1. If there exists only one of g_r or g_l (uni-gap case), then t is behind the existing gap, see Figure 6(a). In this case, $\mathcal{B}$ moves toward this gap until one of these happens: either t becomes visible, a funnel case occurs or $\mathcal{B}$ reaches the reflex vertex of this gap. In any of these three conditions, $\mathcal{B}$'s stopping function will be called.

2. If a funnel case occurs, to bound the detour, the robot moves toward g_r and g_l alternatively, see Figure 6(b). In fact, a funnel can be seen as two concurrent rays created by g_r and g_l. So, we can apply the ICP strategy when $w = 2$ with a minor difference. In the ICP when $w = 2$, the robot goes d steps along one ray, takes another d steps to turn back to the starting point, and then goes d' steps along the other ray. However in this strategy, after traversing d steps along a gap, the robot stops but does not turn back to the starting point of the funnel. Rather, it changes its direction toward the other gap and takes $d + d'$ steps in the new direction. As the robot moves, whenever t becomes visible or $\mathcal{B}$ reaches the critical point of the funnel, the strategy will call $\mathcal{B}$'s stopping function.

Each time the strategy calls $\mathcal{B}$'s stopping function, the robot stops and checks which of the above two cases is happening. We have demonstrated the above strategy as an algorithm in Algorithm 2.

3.6 Analysis

Throughout the search, in uni-gap cases or when the target t is visible, the search path coincides with the shortest path. But as noted earlier, in funnel cases detours are unavoidable. So, to prove the competitive ratio of the strategy, we compare the length of the traversed path and the shortest path in funnel cases. In this case, the angle between g_r and g_l, which is always smaller than π, is called the opening angle, denoted by φ, see Figure 6(b). In Lemma 6, we show that the detour from the shortest path depends on the size of φ.

Algorithm 2: Randomized Street Search

```
while the target $t$ is not visible do
    if a uni-gap case occurs then
        repeat
            Move toward the existing gap
        until a funnel case happens or reflex vertex of the gap is achieved or $t$ becomes visible;
    else if a funnel case occurs then
        $\{d_0, d_1\} \leftarrow \{r, l\}$
        Choose $i$ u.a.r. from $\{0, 1\}$
        Choose $\varepsilon$ u.a.r. from $[0, 1)$
        $j \leftarrow 4^{\varepsilon}$
        Move toward $g_{d_i}$ up to $\lceil j \rceil$ step(s)
        $i \leftarrow (i + 1) mod(2)$
        repeat
            $d \leftarrow \lceil j \rceil + \lceil 4 \cdot j \rceil$
            Move toward $g_{d_i}$ up to $d$ steps
            $i \leftarrow (i + 1) mod(2)$
            $j \leftarrow 4 \cdot j$
        until a critical point of the funnel is achieved or $t$ becomes visible;
end
Move directly toward $t$
```

Lemma 6 *In a funnel case, the detour from the shortest path for a small opening angle is shorter than the detour for a large opening angle.*

Proof. According to our strategy, in a funnel case, $\mathcal{B}$ moves toward g_r and g_l alternatively. In this alternative movement, moving toward one of the directions is correct and moving toward the other is a deviation. Assume that at point p_i, when a funnel case occurs, $\mathcal{B}$ moves toward g_r and reaches point p_{i+1} while the target t is behind g_l, see Figure 7(a). Now, to achieve t, $\mathcal{B}$ should traverse at least a distance

$$\delta = \sqrt{|p_i p_{i+1}|^2 + |p_i v_l|^2 - 2 \cdot |p_i p_{i+1}| \cdot |p_i v_l| \cdot \cos\varphi}$$

by the law of cosines, see Figure 7(a). It can be verified that δ is strictly increasing as a function of φ by taking the derivative with respect to φ where $0 < \varphi < \pi$. So, greater opening angle φ results in greater detour taken by the robot. □

The following theorem demonstrates the competitive ratio of the strategy.

Theorem 7 *Given a street P and a simple robot $\mathcal{B}$, Our randomized strategy for the street search problem is* 6.29*-competitive.*

Proof. As we noted before, except in funnel cases, the search path coincides with the shortest path. So, the detours from the shortest path happen only in the presence of funnels. Therefore, to compute the competitive ratio of our strategy, only funnel cases matter. In a funnel case, by Theorem 6, for larger opening angles the robot deviates more from the shortest path. Since φ never exceeds π, the worst case happens when the opening angle φ is adequately close to π and the robot can only move toward left and right. In this situation, searching for a critical point in a funnel case reduces to the ICP with $w = 2$, see Figure 7(b). Hence, to obtain the worst-case analysis, we consider the case when φ is very close to π. According to Algorithm 2, our strategy in a funnel case with opening angel φ close to π, operates the same as the strategy for the ICP presented in Algorithm 1 with $w = 2$ and $r = 4$. Note that for the ICP, we have obtained $r^* = 4$ for the case of $w = 2$, see Table 1. By Theorem 4, the strategy for the ICP is 6.29-competitive for $w = 2$ and this completes the proof of the theorem. □

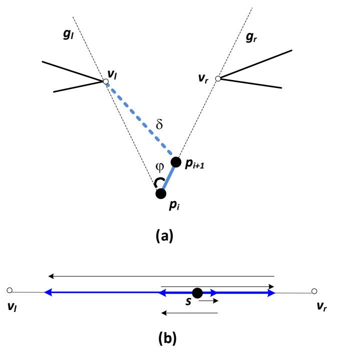

Figure 7: (a) $p_i p_{i+1}$ is a detour from the shortest path. (b) The worst case.

4 Conclusion

We introduced and studied a new variation of the cow-path problem called the Integer Cow-path Problem (ICP). In this variation, the robot is restricted to take only an integer number of steps. We present a randomized strategy for this problem which gives an upper bound on the competitive ratio of all randomized strategies. Moreover, we use this strategy for the problem of searching in a street using a simple robot. Improving the upper bound and proving a lower bound on the competitive ratio can be one of the future works. Also, it is interesting to design deterministic strategies for the ICP. Albeit the doubling strategy in [6] is a deterministic solution to the ICP for $w = 2$, the ICP remains an interesting open problem for $w > 2$.

References

[1] M. Abouei Mehrizi, M. Ghodsi, and A. Tabatabaei. Robots' cooperation for finding a target in streets. In *International Conference on Topics in Theoretical Computer Science*, pages 30–43. Springer, 2015.

[2] P. P. Acarnley. *Stepping motors: a guide to theory and practice*. Number 63. Iet, 2002.

[3] S. Alpern and S. Gal. *The theory of search games and rendezvous*, volume 55. Springer Science & Business Media, 2006.

[4] G. E. Andrews and B. C. Berndt. *Ramanujan's lost notebook*, volume 1. Springer, 2005.

[5] S. Angelopoulos and M. Voss. Online search with maximum clearance. *arXiv preprint arXiv:2011.14144*, 2020.

[6] R. A. Baezayates, J. C. Culberson, and G. J. Rawlins. Searching in the plane. *Information and computation*, 106(2):234–252, 1993.

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

[8] R. Bellman. Problem 63-9, an optimal search. *SIAM review*, pages 274–274, 1963.

[9] P. Bose and J.-L. De Carufel. A general framework for searching on a line. *Theoretical Computer Science*, 703:1–17, 2017.

[10] P. Bose, J.-L. De Carufel, and S. Durocher. Searching on a line: A complete characterization of the optimal solution. *Theoretical Computer Science*, 569:24–42, 2015.

[11] E. D. Demaine, S. P. Fekete, and S. Gal. Online searching with turn cost. *Theoretical Computer Science*, 361(2-3):342–355, 2006.

[12] S. Gal. Minimax solutions for linear search problems. *SIAM Journal on Applied Mathematics*, 27(1):17–30, 1974.

[13] S. K. Ghosh and R. Klein. Online algorithms for searching and exploration in the plane. *Computer Science Review*, 4(4):189–201, 2010.

[14] C. Icking, R. Klein, E. Langetepe, S. Schuierer, and I. Semrau. An optimal competitive strategy for walking in streets. *SIAM Journal on Computing*, 33(2):462–486, 2004.

[15] M.-Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–79, 1996.

[16] R. Klein. Walking an unknown street with bounded detour. *Computational Geometry*, 1(6):325–351, 1992.

[17] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[18] A. Tabatabaei and M. Ghodsi. Optimal strategy for walking in streets with minimum number of turns for a simple robot. In *International Conference on Combinatorial Optimization and Applications*, pages 101–112. Springer, 2014.

[19] A. Tabatabaei and M. Ghodsi. Randomized strategy for walking in streets for a simple robot. *arXiv preprint arXiv:1512.01784*, 2015.

[20] A. Tabatabaei and M. Ghodsi. Walking in streets with minimal sensing. *Journal of Combinatorial Optimization*, 30(2):387–401, 2015.

[21] A. Tabatabaei, F. Shapouri, and M. Ghodsi. A competitive strategy for walking in generalized streets for a simple robot. In *CCCG*, pages 75–79, 2016.

[22] B. Tovar, R. Murrieta-Cid, and S. M. LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, 23(3):506–518, 2007.

[23] Q. Wei, X. Tan, and Y. Ren. Walking an unknown street with limited sensing. *International Journal of Pattern Recognition and Artificial Intelligence*, 33(13):1959042, 2019.

.

Appendix

A

To prove that $f_{ub}(r,i)$ is non-increasing with respect to $i \in [0,\infty)$, all we need is to show that the derivative of $f_{ub}(r,i)$ is least equal to zero. It can be calculated as:

$$\frac{d}{di} f_{ub}(r,i) = r^i(i(r-1)+r)\ln(r)$$
$$+ (\frac{1}{3})r^i\left(-3r\ln\left(r^{(i+1)}\right) + 3\ln\left(r^i\right)\right)$$

$$-(\frac{1}{3})r^i\left(\frac{50r\left(8r^{(i+1)}+24r^{(2i+2)}+1\right)}{100r^{(i+1)}+400r^{(2i+2)}+800r^{(3i+3)}+1}\right)$$

$$+(\frac{1}{3})r^i\left(\frac{15\left(24r^{2i}+8r^i+1\right)}{240r^{3i}+120r^{2i}+30r^i+1}\right)$$

After multiplying all the above terms, we get:

$$= r^i(ir-i+r)\ln r - r^i(ir-i+r)\ln r$$

$$+\frac{9600r^{4i+3}(1-r)+4800^{3i+1}(1-r^2)}{(800r^{3i+3}+400r^{2i+2}+100r^{i+1}+1)(240r^{3i}+120r^{2i}+30r^i+1)}$$

$$+\frac{1200r^{2i+1}(r-1)+24r^i(1-\frac{10}{3}r^2+3r^i-10r^{i+3})}{(800r^{3i+3}+400r^{2i+2}+100r^{i+1}+1)(240r^{3i}+120r^{2i}+30r^i+1)}$$

Since the first expression yields 0 and the denominator of the remaining fraction above is positive, we only need to consider the sign of the numerator, so:

$$= (1-r)(9600r^{4i+3}+4800^{3i+1}(1+r)+1200r^{2i+1})$$
$$+ 24r^i(1-\frac{10}{3}r^2+3r^i-10r^{i+3}). \qquad (7)$$

According to the fact that $r>1$, with a precise look at (7) we get

$$(9600r^{4i+3}+4800^{3i+1}(1+r)+1200r^{2i+1}) > 0$$

and we know $(1-r)<0$. So,

$$(1-r)(9600r^{4i+3}+4800^{3i+1}(1+r)+1200r^{2i+1}) < 0.$$

Also, we have $-\frac{10}{3}r^2 < -\frac{10}{3}$, and $3r^i < 10r^{i+3}$. Then,

$$24r^i(1-\frac{10}{3}r^2+3r^i-10r^{i+3}) < 0.$$

Hence, (7) is always least equal to zero and this completes the proof.