# The Fast Cop, the Slow Cop, and the Oblivious Fugitive

Jared Coleman[1] and Evangelos Kranakis[2][3]

[1] Dept. of Computer Science, Loyola Mount University, Los Angeles, CA, USA
[2] School of Computer Science, Carleton University, Ottawa, Ontario, Canada.
[3] Research supported in part by NSERC Discovery grant.

January 12, 2026

**Abstract.** Officer Judy Hopps of the Zootopia Police Department faces a challenge: a fugitive has escaped from jail and is fleeing along Savanna Central's main street at constant speed $v < 1$. Judy, starting at position $s$ on this half-infinite street $[0, +\infty)$, must catch the fugitive and return him to jail at the origin. The catch? She doesn't know the fugitive's starting position $d$. To make matters more interesting, her usual partner Nick Wilde is on vacation, leaving her paired with the friendly but slow Officer Clawhauser.

We give search and delivery algorithms to solve Judy's problem under two scenarios: 1) when $v$ is known and $d$ is unknown, and 2) when both $v$ and $d$ are unknown. We prove that the corresponding competitive ratios with respect to an omniscient searcher are $1+\frac{v+\sqrt{2-v^2}}{1-v}$ and $1+\frac{2(1+v)}{1-v}$, respectively, and these are optimal. We also analyze a two-agent variant—should Judy and Clawhauser split up?—and characterize when cooperation outperforms going solo.

Further, we consider the competitive ratio of an algorithm for the case of two cooperating policemen with different speeds starting from the same point at the same time to capture the fugitive and bring him to jail and compare it to the single policeman case.

**Key words and phrases.** Agent, Fugitive, Knowledge, Mobile, Oblivious, Policeman, Searcher.

## 1 Introduction

Officer Judy Hopps (Figure 1a) is an ambitious, up-and-coming member of the Zootopia Police Department. Tomorrow she is scheduled for patrol duty along Savanna Central's main street—a long, straight road leading away from the city's central jail. Her usual partner, Nick Wilde, is on vacation, so she will be paired with Officer Benjamin Clawhauser (Figure 1b). Clawhauser is a friendly cheetah who typically works the front desk; while very personable, he is not exactly known for his speed. Judy

(a) Officer Judy Hopps (b) Officer Clawhauser

**Fig. 1.** The officers of the Zootopia Police Department: (a) Judy Hopps, an ambitious and fast rabbit officer, and (b) Benjamin Clawhauser, a friendly but slow cheetah who works the front desk.

wants to be prepared: if a fugitive escapes during her shift, should she and Clawhauser split up to cover more ground, or would she be better off pursuing the fugitive alone while Clawhauser stays behind?

This scenario motivates the algorithmic problem we study in this paper. The "cops and robbers" problem was introduced by Nowakowski and Winkler in [18] and Quilliot [19], independently, cf., also Frieze et al. [14]. These are perfect information games with two players consisting of a set of cops and a robber which can move along the vertices of a graph. The cops and the robber occupy vertices of the graph in alternate rounds and the game ends if the cops and the robber are colocated on a vertex of the graph. For additional information on search and robber problems, cf. Bonato [3].

We study a similar problem here whereby a good cop wants to catch an oblivious fugitive and bring him back to jail. The cop is "good" because she wants to employ the fairest and fastest algorithm, depending on her knowledge about the fugitive, while the fugitive is "oblivious" in that it moves away from the origin without ever changing its behaviour throughout the search process. The algorithm is executed in a continuous (as opposed to discrete) setting on the infinite half-line.

To answer Judy's question systematically, we proceed in two parts. First, we analyze the single-agent case: what is the best strategy for Hopps alone? This establishes a baseline and identifies optimal algorithms under different knowledge assumptions. Then, we turn to the two-agent case: given that Clawhauser is available (albeit slow), when does cooperation actually help?

## 1.1 Catching an Oblivious Fugitive

Officer Hopps (modelled as an autonomous mobile agent) and a fugitive are on the half-infinite line $[0, +\infty)$. Hopps starts at a point $s \geq 0$ and can move with speed 1, while the fugitive starts at a point $d > 0$ and can move with constant speed $v < 1$ away from the origin (see Fig. 2). The jail is located at the origin. The fugitive is oblivious and moves with

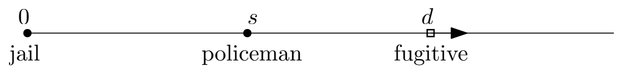

**Fig. 2.** Starting positions of Hopps at the point $s \geq 0$ and the fugitive at the point $d \geq 0$ on the infinite half line. The jail is at the origin. The fugitive is oblivious and is moving away from the jail at constant speed $v < 1$.

constant speed. We are interested in solving the following problem.

*Problem 1.* Give an algorithm with optimal competitive ratio for Hopps to catch the fugitive and bring him to jail.

## 1.2 Model and Assumptions

We call the half-open interval $[0, +\infty)$ the "half-line". During the search the fugitive is oblivious in that it moves at constant speed $0 \leq v < 1$ away from the origin and never stops or changes direction of movement. Hopps can move anywhere on the infinite half-line, change direction, stop and restart, and "carry" the fugitive (when she catches him) without exceeding her max speed 1; her speed is not affected when carrying the fugitive. Hopps starts at a point $s \geq 0$ and the fugitive at a point $d > 0$ on the half-line. The starting position $d$ is unknown to Hopps, and both Hopps and the fugitive start at the same time.

An instance $I$ of the search and delivery problem is determined by the starting locations of Hopps ($s$) and the fugitive ($d$), along with the fugitive's speed $v < 1$. $T_{opt}(I)$ is the optimal time it takes an omniscient searcher who has full knowledge of the input $I$ in order to catch and bring the fugitive to jail; $T_A(I)$ is the time it takes the algorithm $A$ to perform the same task given its (limited) knowledge. We denote by $CR_A = \sup_I \frac{T_A(I)}{T_{opt}(I)}$ the worst-case competitive ratio of algorithm $A$ to deliver on any input $I$. Further, we denote the competitive ratio of search and delivery by $CR = \inf_A CR_A$, where the infimum is taken over all possible delivery algorithms $A$ which successfully return the fugitive to jail.

### 1.3 Related Work

Competitive algorithms for search have been considered in numerous scientific studies due to their importance for the analysis of algorithms in theoretical computer science. The seminal book by Alpern and Gal [1] is a good source for studies on search games and rendezvous. Baeza-Yates et al. [2] initiated and considered search problems by a single robot in an infinite line or in the plane. Our approach will not involve the doubling, but is worth mentioning Chrobak and Kenyon-Mathieu in [8] which give a survey of the "doubling" method which uses geometrically increasing estimates on the optimal solution to produce fragments of the desired solution in online and offline approximation algorithms.

The concept of searching for a moving target was initiated by McCabe [17] in a stochastic setting on an infinite discrete line. Demaine et al. [13] initiated search analysis when turning is counted as a cost. Bose et al. [6,5] derive optimal competitive search strategies for a search problem variant where the target is moving and the searcher's cost at each step is a constant times the length of the step plus a fixed constant turn cost. Bonato et al. [4] study $p$-Faulty Search, whereby a probabilistically faulty robot searches the half-line for a hidden item. More recently, Coleman et al. [12] consider and analyze knowledge competitive ratio tradeoffs in searching for a mobile target in an infinite line and Coleman et al. in [11] obtain optimal competitive ratio when the speed of the mobile is unknown. Additional work in more complex topologies can be found in Ortiz and Schuierer [16] for search on $m$-rays, and more generally in Koutsoupias et al. [15] which combinatorial optimization problems related to searching a known graph, for a target located at an unknown node.

Related to our present work there are also several recent studies on the search and delivery problem especially due to its applicability to automated robotic delivery of objects. Carvalho et al. [7] analyze the problem on graphs, and Coleman et al.[10] consider the problem of how to efficiently recover and complete the delivery of an object in the plane under the presence of a faulty robot. Closely related to our study is the recent work of Coleman et al. [9] which analyzed the search and delivery problem considered in our present paper for a static target in both a finite segment and the infinite half-line.

### 1.4 Results

We design algorithms for the cases when the starting position $d$ of the fugitive is either known or unknown to Hopps. In all cases we compare the ratio of the time it takes the algorithm to catch the fugitive divided by the time it takes an omniscient searcher (having full knowledge of the environment) to find the fugitive and deliver him to the origin.

If $d$ is known to Hopps then it is simple to prove tight upper and lower bounds regardless of whether or not the speed of the fugitive is known. We consider two cases.

**Case 1: $d \leq s$.**
The algorithm for Hopps is

$$s \to 0. \tag{1}$$

Hopps moves in the direction of the origin; she catches the fugitive and delivers him to the origin.

**Case 2: $s < d$.**
The algorithm for Hopps is

$$s \to +\infty \to 0. \tag{2}$$

Hopps moves in the direction of the fugitive. She catches the fugitive at time $\frac{d-s}{1-v}$. At that time the fugitive is in position $d+\frac{(d-s)v}{1-v}$. The total time is $\frac{d-s}{1-v}+d+\frac{(d-s)v}{1-v}=d+\frac{(d-s)(1+v)}{1-v}$. It is easy to see that both algorithms have optimal competitive ratio equal to 1.

Since the case when $d$ is known is trivial, in the sequel we consider only the case when $d$ is unknown to Hopps. This includes the two cases when $v$ is known and when $v$ is unknown. Here is a summary of the results.

**Table 1.** Upper and lower bounds on the competitive ratio of search and delivery for various knowledge models; $v$ is the speed of the fugitive and $d$ is the distance of the fugitive at the start.

| Knowledge | UB & LB |
|---|---|
| $d$ Known | 1 |
| $d$ Unknown & $v$ Known | $1+\frac{v+\sqrt{2-v^2}}{1-v}$ |
| $d$ Unknown & $v$ Unknown | $1+\frac{2(1+v)}{1-v}$ |

In the rest of the paper we consider the case when $d$ is unknown to Hopps. In Section 2 we consider the case when $v$ is known, and in Section 3 the case when $v$ is unknown. In both instances we prove tight upper and lower bounds. Finally, in Section 4, we analyze whether Hopps should team up with Clawhauser or go it alone.

## 2 Speed $v$ is Known to Hopps

We prove upper and lower bounds.

### 2.1 Upper Bound

In the sequel we employ a parameter $x > s$ indicating a guess made by the searcher and consider the following algorithm abbreviated as

$$s \to x \to 0 \to +\infty.$$

The parameter $x$ used in the algorithm will be determined in the sequel. In more detail the algorithm is as follows.

1. The searcher follows path $s \to x$;
2. **If** at any time during this traversal the searcher finds the fugitive, it brings him to the origin and stops;
3. **Else** searcher upon reaching the point $x$ it returns to the origin by following the path $x \to 0$;
4. Upon reaching the origin, the searcher follows the path $0 \to +\infty$ and when it catches the fugitive it brings him to the origin;

In the analysis of the algorithm we will show that the optimal choice for $x$ will be

$$x = \frac{2 - v + \sqrt{2 - v^2}}{2(1 - v)} \cdot s,$$

which the searcher can employ because it knows the value of $v$.

**Theorem 1 (Upper Bound when Speed $v$ is Known).** *The competitive ratio of the previous search and delivery algorithm for a mobile fugitive moving away from the origin with constant speed $v < 1$ is*

$$1 + \frac{v + \sqrt{2 - v^2}}{1 - v}.$$

*Proof.* (Theorem 1) Let $A$ denote the algorithm above and $T_A$ its cost. Also $T_{Opt}$ is the cost of the optimal offline algorithm. We consider several cases depending on the relative sizes of the parameters $d, s, x$.

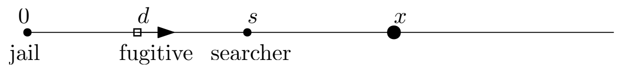

**Fig. 3.** $d < s$.

**Case 1:** $d < s$**.** (see Fig. 3.)
Since $d < s$ it is clear that $T_{opt} = s$. Since $x > s$, we have that $T_A = x - s + x = 2x - s$. It follows that

$$CR_1 = \frac{T_A}{T_{opt}} = \frac{2x - s}{s} = \frac{2x}{s} - 1. \tag{3}$$

**Case 2:** $s \le d \le x$**.** (see Fig. 4.)
The searcher will catch the fugitive in time $\frac{d-s}{1-v}$; since it started at $s$ it will

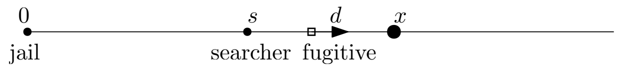

**Fig. 4.** $s \le d \le x$.

now be located at the point $s + \frac{d-s}{1-v}$ so it will take the searcher additional time $s + \frac{d-s}{1-v}$ to reach the origin. Therefore $T_{opt} = s + 2\frac{d-s}{1-v}$.

**Case 2a. If** $s + \frac{d-s}{1-v} \le x$ **then** $CR_2 = 1$**.**
The searcher starts at $s$ and the fugitive starts at $d$, where $s \le d \le x$. The searcher could reach the fugitive in time $\frac{d-s}{1-v}$. Since $s + \frac{d-s}{1-v} \le x$ the searcher will in fact catch the fugitive before reaching $x$ and can therefore return him to the origin. As a consequence, $T_A = s + 2\frac{d-s}{1-v} = T_{opt}$ and $CR_2 = 1$.

**Case 2b. If** $s + \frac{d-s}{1-v} > x$ **then** $CR_2$ **is given in Equation (4).**
As before, the searcher starts at $s$ and the fugitive starts at $d$, where $s \le d$. The searcher could reach the fugitive in time $\frac{d-s}{1-v}$ but since $s + \frac{d-s}{1-v} > x$ the searcher will not be able to travel far enough (since according to the algorithm it would have to stop and change direction at $x$) and will therefore miss the fugitive in its first. So it will return to the origin and follow the trajectory $0 \to +\infty$ until it catches the fugitive and return him to the origin. When the searcher returns to the origin for the first time, total time

$$T_0 = 2x - s$$

has already passed and the fugitive will be in position

$$d + ((x - s) + x)v = d + (2x - s)v.$$

when the searcher is at the origin. The searcher will now follow the path $0 \to +\infty$ and will catch the fugitive in additional time

$$T_1 = \frac{d + (2x - s)v}{1 - v}$$

and return him to the origin in further additional time $T_1$. Therefore algorithm $A$ takes total time

$$T_A = T_0 + 2T_1 = 2x - s + 2\frac{d + (2x - s)v}{1 - v}.$$

The competitive ratio as a function of $x$ will be

$$CR_2(d) = \frac{T_A}{T_{opt}} = \frac{2x - s + 2\frac{d+(2x-s)v}{1-v}}{s + 2\frac{d-s}{1-v}}. \tag{4}$$

Equation (4) is valid as long as the condition $s + \frac{d-s}{1-v} > x$, which is equivalent to $d > s + (x - s)(1 - v)$.

The resulting competitive ratio will be

$$\begin{aligned} CR_2 &= \sup_{s \le d \le x} CR_2(d) \\ &= \sup_{s \le d \le x} \frac{2x - s + 2\frac{d+(2x-s)v}{1-v}}{s + 2\frac{d-s}{1-v}} \\ &= \sup_{s \le d \le x} \left(1 + \frac{2x(1+v)}{2d - s(1+v)}\right) \\ &= 1 + \frac{2x(1+v)}{2(s + (x-s)(1-v)) - s(1+v)} \\ &= 1 + \frac{2x(1+v)}{(1-v)(2x-s)} \end{aligned} \tag{5}$$

**Case 3:** $x < d$**.** (see Fig. 5.)
When the searcher starts its search, its distance from the fugitive will be $d - s$ and it will catch him in time $\frac{d-s}{1-v}$. When that happens its location will be $s + \frac{d-s}{1-v}$ and it will take additional time $s + \frac{d-s}{1-v}$ to return to the origin. It follows that $T_{opt} = s + 2\frac{d-s}{1-v}$.

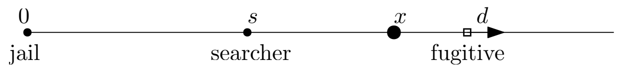

**Fig. 5.** $s \le d \le x$.

Since the searcher traverses the path from $s$ only up to position $x$ it will miss the fugitive during its first pass. It will have spent total time $T_0 = x - s + x = 2x - s$ to return to the origin for the first time. The fugitive will now be in position $d + (2x - s)v$. The searcher can catch the fugitive and return him to the origin in additional total time $T_1 = 2\frac{d+(2x-s)v}{1-v}$. It follows that

$$T_A = T_0 + T_1 = 2x - s + 2\frac{d + (2x - s)v}{1 - v}.$$

As a consequence the competitive ratio as a function of $d$

$$CR_3(d) = \frac{T_A}{T_{opt}} = \frac{2x - s + 2\frac{d+(2x-s)v}{1-v}}{s + 2\frac{d-s}{1-v}}. \tag{6}$$

As with Case 2, we have

$$\begin{aligned} CR_3 &= \sup_{d>x} CR_2(d) \\ &= \sup_{d>x} \frac{2x - s + 2\frac{d+(2x-s)v}{1-v}}{s + 2\frac{d-s}{1-v}} \\ &= \sup_{d>x} \left(1 + \frac{2x(1+v)}{2d - s(1+v)}\right) \\ &= 1 + \frac{2x(1+v)}{2x - s(1+v)}. \end{aligned} \tag{7}$$

It is easy to see that $CR_2 \le CR_3$. It follows that the resulting competitive ratio of the algorithm will be

$$\begin{aligned} CR &= \max\{CR_1, CR_2\} \\ &= \max\left\{\frac{2x}{s} - 1, 1 + \frac{2x(1+v)}{(1-v)(2x-s)}\right\}. \end{aligned} \tag{8}$$

If we solve

$$\frac{2x}{s} - 1 = 1 + \frac{2x(1+v)}{(1-v)(2x-s)}$$

in the unknown $x$ we obtain the solutions

$$x = \frac{2 - v + \sqrt{2 - v^2}}{2(1 - v)} \cdot s$$

The resulting competitive ratio will be

$$\begin{aligned}\frac{2x}{s} - 1 &= \frac{2\frac{2-v+\sqrt{2-v^2}}{2(1-v)} \cdot s}{s} - 1 \\ &= \frac{2 - v + \sqrt{2 - v^2}}{1 - v} - 1 \\ &= \frac{1 + \sqrt{2 - v^2}}{1 - v}.\end{aligned}$$

This proves Theorem 1. □

### 2.2 Lower Bound

**Theorem 2 (Lower Bound when Speed $v$ is Known).** *The competitive ratio of any correct algorithm for capturing a mobile fugitive moving away from the origin with constant speed $v < 1$ is at least*

$$1 + \frac{v + \sqrt{2 - v^2}}{1 - v}.$$

*Proof.* (Outline) Consider any correct algorithm $A$ and let $x$ be the furthest point reached by the searcher prior to visiting the origin. Clearly $s \leq x$. We consider the following cases.

**Case 1:** $d < s$.
The optimal offline algorithm takes time $T_{opt} = s$. The algorithm takes time $T_A \geq x - s + x$. The resulting competitive ratio satisfies $CR_1 \geq \frac{2x-s}{s} = \frac{2x}{s} - 1$.

**Case 2:** $s < d < x$.
We consider two subcases.

**Subcase 2a:** $s + \frac{d-s}{1-v} \leq x$.
As before the competitive in this case satisfies $CR_{2a} \geq 1$.

**Subcase 2b:** $x < s + \frac{d-s}{1-v}$.
As before the competitive in this case satisfies $CR_{2a} \geq 1 + \frac{2x(1+v)}{(1-v)(2x-s)}$.

**Case 3:** $x < d$.
As before the competitive in this case satisfies $CR_{2a} \geq 1 + \frac{2x(1+v)}{(1-v)(2x-s)}$.

It follows that

$$CR \geq \max\left\{\frac{2x}{s} - 1, 1 + \frac{2x(1+v)}{(1-v)(2x-s)}\right\}.$$

This leads to the lower bound

$$\frac{1+\sqrt{2-v^2}}{1-v}.$$

This proves Theorem 2. □

## 3 Speed $v$ Is Unknown to Hopps

We prove upper and lower bounds.

### 3.1 Upper Bound

The searcher executes the following search and delivery algorithm abbreviated as

$$s \to 0 \to +\infty.$$

More formally the algorithm is as follows.

1. The searcher follows path $s \to 0$;
2. If at any time during this traversal the searcher finds the fugitive, it brings him to the origin and stops;
3. Upon reaching the origin if the searcher has not found the fugitive, follows the path $0 \to +\infty$ and when it catches the fugitive it brings him to origin;

**Theorem 3 (Upper Bound when Speed $v$ is Unknown).** *The competitive ratio of the previous search and delivery algorithm for a mobile fugitive moving away from the origin with constant speed $v < 1$ is*

$$1 + \frac{2(1+v)}{1-v}.$$

*Proof.* (Theorem 3) First we analyze the optimal offline algorithm.

**Case 1: $d \leq s$.**

In this case $T_{opt} = s$.

**Case 2: $d > s$.**

The optimal offline algorithm takes time

$$\begin{aligned} T_{opt} &= \frac{d-s}{1-v} + d + \frac{(d-s)v}{1-v} \\ &= d + \frac{(d-s)(1+v)}{1-v}. \end{aligned}$$

Now we analyze algorithm $A$ above.

**Case 1: $d \le s$.**
In this case $T_A = s$.

**Case 2: $d > s$.**
In this case

$$T_A = s + 2\frac{d + sv}{1 - v}.$$

The competitive ratio as a function of $d$ will be

$$\begin{aligned}
CR(d) &= \max\left\{ \left(\frac{T_A}{T_{opt}}\right)_{d \le s}, \left(\frac{T_A}{T_{opt}}\right)_{d > s} \right\} \\
&= \max\left\{ 1, \frac{s + 2\frac{d+sv}{1-v}}{d + \frac{(d-s)(1+v)}{1-v}} \right\} \\
&= \max\left\{ 1, \frac{s(1-v) + 2(d+sv)}{(1-v)d + (d-s)(1+v)} \right\} \\
&= \max\left\{ 1, \frac{2d + s(1+v)}{2d - s(1+v)} \right\} \\
&= 1 + \frac{2s(1+v)}{2d - s(1+v)}
\end{aligned}$$

It follows that

$$\begin{aligned}
CR &= 1 + \sup_{d > s} CR(d) \\
&= 1 + \sup_{d > s} \frac{2s(1+v)}{2d - s(1+v)} \\
&= 1 + \frac{2s(1+v)}{2s - s(1+v)} \\
&= 1 + \frac{2(1+v)}{1-v}.
\end{aligned}$$

This completes the proof of Theorem 3. □

## 3.2 Lower Bound

We assume that the searcher knows neither the starting distance $d$ nor the speed $v < 1$ of the fugitive. We prove the lower bound as follows.[4]

[4] A better and more direct approach could follow along the lines of the proof of Theorem 2. The proof would take into account the farthest distance, say $x$, the searcher reaches prior to visiting the origin.

**Lemma 1.** *If $d < s$ then any algorithm in which the searcher decides to move away from the origin cannot terminate.*

*Proof.* Assume $d < s$. If the searcher does not know neither starting position $d$ nor the speed of the fugitive then it does not make sense for the searcher to move away from the origin. The reason is that regardless of the distance that the searcher decides to move away from the origin the adversary can counter by making the speed $v$ arbitrarily closer to 1. Therefore regardless of the searcher's effort no "distinguishable" event could occur that can help him determine the futility of continuing to move in this direction and should rather stop and turn back. In other words, the searcher will be moving away from the origin for ever. $\square$

**Theorem 4.** *Any algorithm in which the searcher visits the origin before it meets (and catches) the fugitive has competitive ratio at least*

$$1 + \frac{2(1+v)}{1-v}.$$

*Proof.* This is similar to the proof of the upper bound. Clearly, the hypothesis of the lemma implies that the fugitive's starting position must be to the right of the searcher's, i.e., $s < d$. The searcher will take time at least $s$ just to visit the origin. At the time the searcher reaches the origin the fugitive must have moved to a location $\geq d + sv$. It takes additional time at least $2\frac{d+sv}{1-v}$ for the searcher to catch the fugitive and deliver him to the origin. It follows that every correct search and delivery algorithm must take time $T_A \geq s + 2\frac{d+sv}{1-v}$. Since the optimal offline algorithm takes time $\frac{d-s}{1-v}$ to catch the fugitive plus additional $d + \frac{d-s}{1-v}$ to bring him to the origin we conclude that $T_{opt} = \frac{d-s}{1-v} + d + \frac{(d-s)v}{1-v}$ we conclude that the competitive ratio of any correct algorithm must satisfy

$$\begin{aligned} CR_A(d) &\geq \frac{s + 2\frac{d+sv}{1-v}}{d + \frac{d-s}{1-v} + \frac{(d-s)v}{1-v}} \\ &= \frac{s(1-v) + 2d + 2sv}{d(1-v) + (d-s)(1+v)} \\ &= \frac{2d - s(1+v) + s(1+v) + s(1-v) + 2sv}{2d - s(1+v)} \\ &= 1 + \frac{2s(1+v)}{2d - s(1+v)}. \end{aligned}$$

It follows that

$$\begin{aligned} CR = \sup_{d>s} CR_A(d) &\geq 1 + \sup_{d>s} \frac{2s(1+v)}{2d - s(1+v)} \\ &= 1 + \frac{2s(1+v)}{2s - s(1+v)} \\ &= 1 + \frac{2(1+v)}{1-v}, \end{aligned}$$

which completes the proof of the theorem. □

## 4 Two Agents

We now return to Judy's original question: should she and Clawhauser split up? Recall that Hopps has speed 1 (the "fast" agent) while Clawhauser has speed $u < 1$ (the "slow" agent). Both agents start at the same position $s \geq 0$ and must work together to catch the fugitive and return him to jail. As before, the fugitive starts at an unknown distance $d > 0$ from the origin and moves away from the jail at constant speed $v < 1$.

### 4.1 The Split-Up Algorithm

One option is for Hopps to simply execute the single-agent algorithm by herself, ignoring Clawhauser entirely. However, we consider an alternative approach which we call the *split-up algorithm*:

- Hopps moves toward the jail (origin) while Clawhauser moves away from the jail in pursuit of the fugitive.
- If Hopps reaches the jail without encountering the fugitive, she knows $d > s$ and turns around to move away from the jail.
- Hopps continues until she either catches the fugitive herself, or meets Clawhauser returning with the captured fugitive.

Note that Hopps may catch up to and pass Clawhauser before he finds the fugitive, in which case Clawhauser provides no assistance with the capture or delivery.

**Theorem 5 (Competitive Ratio of the Split-Up Algorithm).** *The competitive ratio of the split-up algorithm is*

$$CR = \begin{cases} \frac{3+v}{1-v} & \text{if } \frac{d-s}{u-v} \geq \frac{d+s}{1-v}, \\ \frac{2(1-v)}{u(1+u)(u-v)} & \text{if } \frac{d-s}{u-v} < \frac{d+s}{1-v}. \end{cases}$$

*Proof.* We analyze the competitive ratio of this algorithm. Note that if $d \leq s$, Hopps catches the fugitive while traveling to the origin, achieving optimal competitive ratio 1. Since an adversary would not choose such a $d$, we assume $d > s$ in what follows. Two cases may occur depending on which agent catches the fugitive first.

**Case 1: Hopps catches the fugitive.**
This case occurs when $\frac{d-s}{u-v} \geq \frac{d+s}{1-v}$, i.e., Hopps passes Clawhauser before Clawhauser finds the fugitive, or if $u \leq v$ (so Clawhauser cannot catch up to the fugitive at all). The optimal offline time is

$$T_{opt} = \frac{2(d-s)}{1-v} + s,$$

since in the optimal algorithm Hopps goes straight to the target and returns to the jail. Hopps's capture time is

$$T_A = \frac{d+s}{1-v},$$

since Hopps first travels to the origin (time $s$), then chases the fugitive who is now at position $d + sv$ (additional time $\frac{d+sv}{1-v}$). The total algorithm time including delivery is $2T_A - s$. The competitive ratio as a function of $d$ is

$$CR_1(d) = \frac{2 \cdot \frac{d+s}{1-v} - s}{\frac{2(d-s)}{1-v} + s} = \frac{4d}{2d - s(1+v)} - 1 = \frac{4}{2 - s(1+v)/d} - 1. \tag{9}$$

Observe that $CR_1(d)$ is decreasing in $d$. Thus the supremum over $d > s$ is achieved as $d \to s^+$, giving

$$CR_1 = \sup_{d>s} CR_1(d) = \lim_{d \to s^+} \left( \frac{4}{2 - s(1+v)/d} - 1 \right) = \frac{4}{2 - (1+v)} - 1 = \frac{3+v}{1-v}. \tag{10}$$

**Case 2: Clawhauser catches the fugitive first.**
This case occurs when $u > v$ and $\frac{d-s}{u-v} < \frac{d+s}{1-v}$. Clawhauser catches the fugitive in time

$$T_{capture} = \frac{d-s}{u-v}.$$

At this time, Clawhauser (with the fugitive) is at location

$$L_{capture} = s + \frac{d-s}{u-v} \cdot u = s + \frac{u(d-s)}{u-v}.$$

Meanwhile, observe that regardless of when Clawhauser catches the fugitive, Hopps must reach the origin before Clawhauser does, at which time she turns around to move back out towards Clawhauser (and the target). After Clawhauser catches the fugitive, he turns around and moves toward the origin at speed $u$, while Hopps moves away from the origin at speed 1. To compute when they meet, we use a convenient reflection trick: the meeting point of the two agents is the same as if Hopps had started at position $-s$ (reflected around the origin) and Clawhauser had started at position $s + 2(L_{capture} - s)$ (reflected around $L_{capture}$), with both agents moving toward each other from time 0. The initial distance between these reflected positions is $2s + 2(L_{capture} - s) = 2L_{capture}$, so the meeting time is

$$T_{meet} = \frac{2L_{capture}}{1+u} = \frac{2s + 2(L_{capture} - s)}{1+u}.$$

The competitive ratio in this case is

$$CR_2(d) = \frac{\frac{2}{1+u}\left(2s + \frac{2(d-s)}{u(u-v)}\right) - s}{s + \frac{2(d-s)}{1-v}}. \tag{11}$$

Observe that $CR_2(d)$ is increasing in $d$. Thus the supremum over $d > s$ is achieved as $d \to \infty$, giving

$$CR_2 = \sup_{d>s} CR_2(d) = \lim_{d\to\infty} CR_2(d) = \frac{\frac{2}{1+u} \cdot \frac{2}{u(u-v)}}{\frac{2}{1-v}} = \frac{2(1-v)}{u(1+u)(u-v)}. \tag{12}$$

□

### 4.2 Comparison with Single-Agent Algorithm

If the speed $v$ is known, Hopps could alternatively execute the single-agent algorithm from Section 2, which achieves a competitive ratio of $1 + \frac{v+\sqrt{2-v^2}}{1-v}$.

The split-up algorithm may outperform the single-agent algorithm in certain parameter regimes. Figure 6 shows the region in the $(u, v)$ parameter space where the split-up algorithm achieves a lower competitive ratio than the single-agent algorithm.

The results are intuitive: when Clawhauser's speed $u$ is sufficiently larger than the fugitive's speed $v$, it becomes advantageous for the officers to split up, with Clawhauser intercepting the fugitive while Hopps positions herself for efficient delivery. However, if the fugitive's speed $v$ is

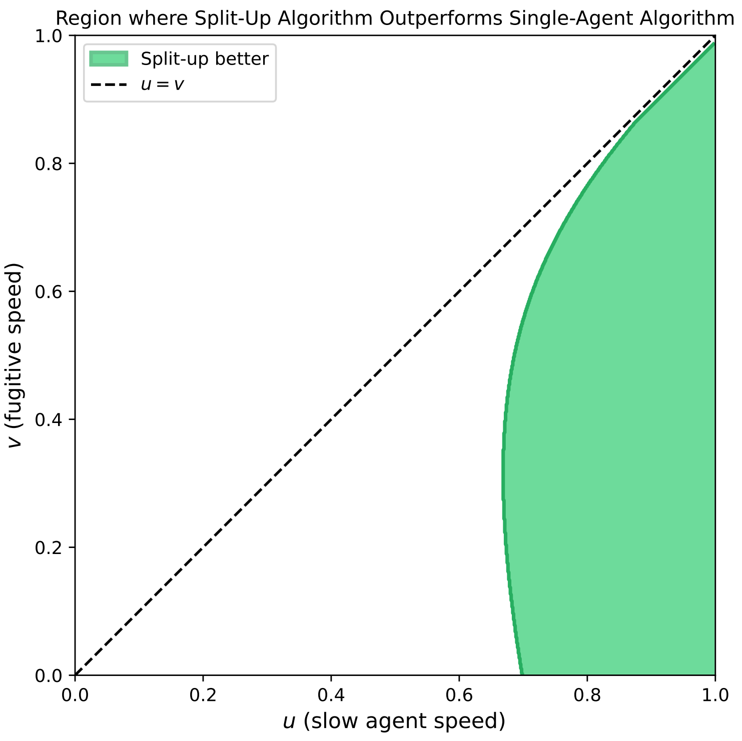

**Fig. 6.** The shaded region shows parameter values $(u, v)$ for which the split-up algorithm outperforms the single-agent algorithm (when $v$ is known). Here $u$ is Clawhauser's speed and $v$ is the fugitive's speed.

too high, the benefit of Clawhauser's assistance diminishes, as he cannot catch up to the fugitive quickly enough.

While the exact boundary of the advantageous region does not admit a simple closed-form expression, it can be computed numerically or using symbolic tools such as Mathematica.

Note that the split-up algorithm requires knowledge of $v$. If $v$ is unknown, the competitive ratio of the split-up algorithm is unbounded, since $\lim_{v\to 1^-} \frac{3+v}{1-v} = \infty$. In contrast, the single-agent algorithm from Section 3 achieves a bounded competitive ratio of $1 + \frac{2(1+v)}{1-v}$ even when $v$ is unknown. Thus, when $v$ is unknown, Hopps should go it alone.

## 5 Conclusion

We analyzed the competitive ratio of algorithms for Officer Hopps to catch a mobile oblivious fugitive and bring him back to jail. We gave optimal algorithms in all knowledge instances concerning the speed $v < 1$ and starting position $d$ of the fugitive. We also considered a two-agent variant—should Hopps team up with Clawhauser?—and found that cooperation via a split-up strategy can outperform going solo, but only when Clawhauser is fast enough relative to the fugitive and when the fugitive's speed is known. When $v$ is unknown, Hopps is better off on her own.

## References

1. S. Alpern and S. Gal. *The theory of search games and rendezvous*, volume 55 of *International series in operations research and management science*. Kluwer, 2003.
2. R. A. Baezayates, J. C. Culberson, and G. J. Rawlins. Searching in the plane. *Information and computation*, 106(2):234–252, 1993.
3. A. Bonato. *The game of cops and robbers on graphs*. American Mathematical Soc., 2011.
4. A. Bonato, K. Georgiou, C. MacRury, and P. Prałat. Probabilistically faulty searching on a half-line. In *Latin American Symposium on Theoretical Informatics*, pages 168–180. Springer, 2020.
5. P. Bose and J.-L. De Carufel. A general framework for searching on a line. *Theoretical Computer Science*, 703:1–17, 2017.
6. P. Bose, J.-L. De Carufel, and S. Durocher. Revisiting the problem of searching on a line. In *European Symposium on Algorithms*, pages 205–216. Springer, 2013.
7. I. A. Carvalho, T. Erlebach, and K. Papadopoulos. On the fast delivery problem with one or two packages. *J. Comput. Syst. Sci.*, 115:246–263, 2021.
8. M. Chrobak and C. Kenyon-Mathieu. Sigact news online algorithms column 10: Competitiveness via doubling. *ACM SIGACT News*, 37(4):115–126, 2006.
9. J. Coleman, L. Cheng, and B. Krishnamachari. Search and rescue on the line. In *International Colloquium on Structural Information and Communication Complexity*, pages 297–316. Springer, 2023.
10. J. Coleman, D. Krizanc, E. Kranakis, and O. Morales-Ponce. Optimal delivery with a faulty drone. In *CCCG 2025 (Canadian Conference on Computational Geometry), August 13 - 15, York University, Toronto, Canada*, 2025.
11. J. R. Coleman, D. Ivanov, E. Kranakis, D. Krizanc, and O. Morales-Ponce. Linear search for an escaping target with unknown speed. *JCSS, Journal of Computer and System Sciences*, 157:103737, 2026. Preliminary version in proceedings of IWOCA 2024, pages 396–407.
12. J. R. Coleman, E. Kranakis, D. Krizanc, and O. Morales-Ponce. Line search for an oblivious moving target. In E. Hillel, R. Palmieri, and E. Rivière, editors, *26th International Conference on Principles of Distributed Systems, OPODIS 2022, December 13-15, 2022, Brussels, Belgium*, volume 253 of *LIPIcs*, pages 12:1–12:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
13. E. D. Demaine, S. P. Fekete, and S. Gal. Online searching with turn cost. *Theoretical computer science*, 361(2-3):342–355, 2006.

14. A. Frieze, M. Krivelevich, and P.-S. Loh. Variations on cops and robbers. *Journal of Graph Theory*, 69(4):383–402, 2012.
15. E. Koutsoupias, C. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *International Colloquium on Automata, Languages, and Programming*, pages 280–289. Springer, 1996.
16. A. López-Ortiz and S. Schuierer. The ultimate strategy to search on m rays? *Theoretical Computer Science*, 261(2):267–295, 2001.
17. B. J. McCabe. Searching for a one-dimensional random walker. *Journal of Applied Probability*, 11(1):86–93, 1974.
18. R. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239, 1983.
19. A. Quilliot. A short note about pursuit games played on a graph with a given genus. *Journal of combinatorial theory, Series B*, 38(1):89–92, 1985.