A Robber Locating Strategy for Trees

Axel Brandt[*], Jennifer Diemunsch[†], Catherine Erbes[‡], Jordan LeGrand[§], Casey Moffatt[¶]

March 16, 2015

Abstract

The *robber locating game*, introduced by Seager in [8], is a variation of the classic cops and robbers game on a graph. A cop attempts to locate an invisible robber on a graph by *probing* a single vertex v each turn, from which the cop learns the robber's distance. The robber is then permitted to stay at his current vertex or move to an adjacent vertex other than v. A graph is *locatable* if the cop is able to locate the robber in a finite number of probes, and the *location number* of a graph is the minimum number of probes necessary to determine the robber's location regardless of the robber's evasion strategy. We provide a strategy that locates the robber on trees. The number of probes used under this strategy meets, and often improves, the theoretical bound on the location number of trees.

Keywords: robber locating game, cops and robbers, trees

1 Introduction

The classic cops and robbers game on graphs is a pursuit–evasion game introduced independently by Nowakowski and Winkler [6] and Quilliot [7]. This is a game with perfect information played by a cop and robber on a graph. The cop and robber take turns in which they are permitted to move to an adjacent vertex or remain at their present vertex. The cop wins if at any point he occupies the same vertex as the robber. Bonato and Nowakowski's book [1] contains numerous results on the cops and robbers game along with variations thereof. The variation in [5] is of particular interest because of its role in determining the tree–width of a graph [11].

In this paper we consider the *robber locating game*, introduced by Seager in [8]. In this game, an invisible robber on a graph tries to evade a cop, who is not on the graph. The cop pursues the robber by *probing* a single vertex v each turn, from which she learns the robber's distance. The robber is then permitted to stay at his current vertex or move to an adjacent vertex other than v. Disallowing the robber from moving to the most recently probed vertex is referred to as the *no–backtrack condition.* A graph G is *locatable* if there is a strategy under which the cop is guaranteed to determine the location of the robber in the vertex set of G. For a locatable graph

[*]Dept. of Math. and Stat. Sciences, University of Colorado Denver, Axel.Brandt@ucdenver.edu, partially funded by NSF GK-12 Transforming Experiences Grant DGE-0742434

[†]Dept. of Math. and Stat. Sciences, University of Colorado Denver, Jennifer.Diemunsch@ucdenver.edu, partially funded by NSF GK-12 Transforming Experiences Grant DGE-0742434

[‡]Mathematics Department, Hiram College, erbescc@hiram.edu, partially funded by NSF GK-12 Transforming Experiences Grant DGE-0742434

[§]Dept. of Math. and Stat. Sciences, University of Colorado Denver, jordan.r.legrand@gmail.com

[¶]Dept. of Math. and Stat. Sciences, University of Colorado Denver, caseymoffatt@msn.com

G, the *location number*, denoted $\operatorname{loc}(G)$, is the minimum number of probes necessary to determine the robber's location, regardless of the robber's evasion strategy.

The robber location game has also been studied without the no–backtrack condition. Subdivisions of graphs are studied in [2] and locatable trees are characterized in [10]. A similar game is studied in [4], where the robber must move on each turn, and probes return distance 0 or nothing at all.

Additionally, versions where the invisible robber is stationary have also been studied. Seager studies locating an invisible, stationary robber with a single probe per turn in [9]. In the graph locating problem, which was independently posed by Slater [12] and Harary and Melter [3], a cop is allowed a set of probes on each turn. The *metric dimension* of a graph is defined within this problem as the minimum size of a probe set that locates the robber on the cop's first turn.

Here, we consider the original robber locating game with the no–backtrack condition. In her paper introducing the game, Seager proves the following.

Theorem 1 (Seager, [8])**.** *If T is a tree with $n \geq 3$ vertices, then $\operatorname{loc}(T) \leq n-2$, with equality if and only if $T = K_{n-1,1}$.*

It should be noted that [8] only gives explicit strategies for K_3, $K_{2,3}$, C_n with $n \neq 5$, and spiders (i.e. subdivided stars). Here we present an explicit strategy for the robber locating game played on general trees and prove that this strategy meets and in many cases improves the bound in Theorem 1. The *eccentricity* of a vertex u, denoted $\epsilon(u)$, is $\max_{v \in V(G)} \operatorname{dist}(u,v)$ where $V(G)$ denotes the vertex set of G and $\operatorname{dist}(u,v)$ is the distance from u to v. The *radius* of a graph G, denoted $\operatorname{rad}(G)$, is defined as $\min_{u \in V(G)} \epsilon(u)$.

Theorem 2. *Let T be a tree on $n \geq 3$ vertices with ℓ leaves and maximum degree Δ. Then $\operatorname{loc}(T) \leq \min\{n-2, 2\ell-3, \Gamma(T)\}$, where*

$$\Gamma(T) = \begin{cases} 2\Delta^2(\operatorname{rad}(T)-2) + \Delta(17 - 6\operatorname{rad}(T)) + 5\operatorname{rad}(T) - 18, & \text{if } \operatorname{rad}(T) \geq 3 \\ 3\Delta - 4, & \text{otherwise.} \end{cases}$$

A strategy S to locate the robber on G has the *Root Location Property (RLP)* for a root $r \in V(G)$ provided that if the robber is initially at r, or if at any point he moves to r, then the next probe will locate him at r. In proving Theorem 2, we will rely heavily upon the following lemma.

Lemma 1 (Seager, [8])**.** *Let T be a tree with root r. Let T' with root r' be a proper subtree of T such that if $r \neq r'$, then $r \notin V(T')$. Let S' be a strategy to locate the robber on T' that has the RLP for r'. Let S_0 be a strategy with the RLP for r that either locates the robber on T or determines that he is on $T' - r'$. Let S be the strategy that consists of applying S_0 and then, if the robber is not located, applying S'. This strategy S locates the robber on T and has the RLP for r.*

2 A Robber Locating Strategy

We begin by introducing and motivating certain terminology and notation. For a tree T, let the order of T, $|V(T)|$, be denoted by $n(T)$. We will abbreviate $n(T)$ by n when the context is clear. Let $N(v)$ denote the neighborhood of a vertex v, and $\ell(T)$ the number of leaves in T. For vertices x and y, an (x,y)–path is a path with endpoints x and y. *Branch vertices* are vertices with degree at least three. A vertex is *central* in T if its distance from any other vertex is at most $\operatorname{rad}(T)$.

Let T be a rooted tree with root r. The vertices at distance d from r form *level* d of T. For $X \subseteq V(T)$, let $T \setminus X$ be the forest obtained by deleting the vertices of X. If $X = \{v\}$, we write $T \setminus v$, or, for brevity, T^v. We define a partial order $(V(T), \leq_r)$ where $v \leq_r u$ if there exists an (r, u)-path in T that contains v. We call this the *tree–order* with respect to r. We will use this tree–order to identify subtrees of T in which the robber could be positioned. Let $\lfloor v \rfloor_r$ be the subtree of T induced by the vertices $u \in V(T)$ such that $v \leq_r u$ under the tree–order with respect to r. For example, in the tree T in Figure 1, $\lfloor e_4 \rfloor_{f_2} = T \setminus f_2$ and $\lfloor e_4 \rfloor_a$ is the subgraph induced by $\{e_4, f_2, f_3\}$. The latter subtree can also be described in many ways including $\lfloor e_4 \rfloor_{b_2}$, $\lfloor e_4 \rfloor_{c_1}$, and $\lfloor e_4 \rfloor_{f_1}$. In general, observe that if vertices v and u are not comparable under the tree-order with respect to r, then $\lfloor v \rfloor_r \cap \lfloor u \rfloor_r = \emptyset$.

When probing vertex v, let $P(v)$ be the distance from v to the robber. Let Λ_v be the set of vertices in T on which the robber could be positioned, that is those vertices at distance $P(v)$ from v. For example, if $P(e_3) = 3$ then $\Lambda_{e_3} = \{b_3, d_8, f_2, f_3, f_4, f_5\}$. Notice that after probing the root r, Λ_r is precisely level $P(r)$ of T.

We aim to isolate the robber on a subtree of T, which typically focuses on components of T^r when T is rooted at r. After probing v, let b_v be the maximum vertex u under the tree order for which $\Lambda_v \subseteq \lfloor u \rfloor_r$. We refer the reader to Figure 1 for an example. Note that b_v is well-defined since all vertices u with $\Lambda_v \subseteq \lfloor u \rfloor_r$ are comparable. If the robber is not located by probing v, the maximality condition in the definition forces b_v to be a branch vertex. Thus, there are at least two components in $\lfloor b_v \rfloor_r \setminus b_v$.

We define a (u, v)*–thread* to be a (u, v)–path such that u is a leaf of T and the path contains no branch vertices except possibly v. Note that every tree with a branch vertex contains a branch vertex v and leaf u such that the (u, v)–path is a (u, v)–thread; this (u, v)–thread may be a single edge.

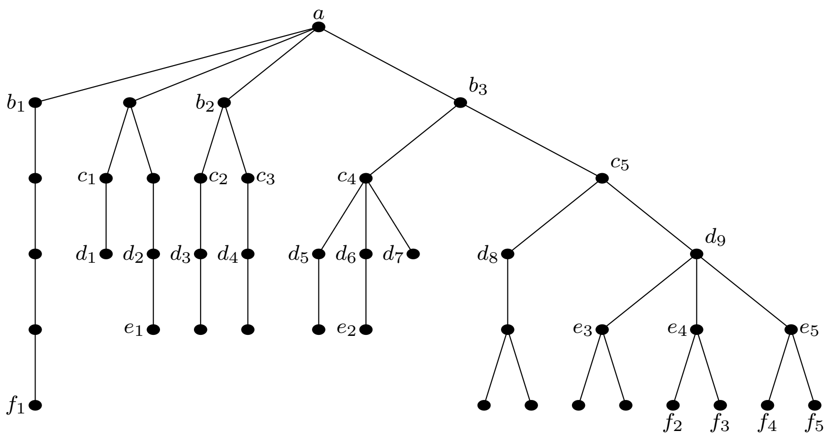
The following definitions pertain specifically to our strategy. After probing v, if the robber is known to be in $\lfloor r' \rfloor_r \setminus r'$ for some $r' \in V(T)$, then the strategy will *reroot* T to r'. The process of rerooting to r' consists of setting r' to be the new root, tracking what level of the newly–rooted tree the robber is on, deleting components of $T^{r'}$ that cannot contain the robber, and proceeding to Step 1 of the strategy. It is possible to reroot to the same root, that is let $r' = r$ when rerooting. In this case, both tracking the robber's level and deleting components are significant pieces in locating the robber.

We will argue during presentation of the strategy that RLP is maintained for each root throughout the algorithm. This allows us to delete components of T^v to which the robber is unable to move for the remainder of the game. We will then use Lemma 1 to show that the strategy locates the robber.

Now we describe the robber locating strategy. Note that an example game is shown in Figure 1. As the strategy proceeds we will select components of T^r as described in the following initialization. The initialization of the strategy depends on the minimum of $n - 2$, $2\ell(T) - 3$, and $\Gamma(T)$; this minimum will be denoted by M. In the event that more than one of the following cases for initialization hold, we use the first applicable case.

Initialization 1: $M = n - 2$ and T is not a path. Set the initial root to be a branch vertex that is the endpoint of a thread, and when selecting a component of T^r, select one of minimum order.

Initialization 2: $M = 2\ell(T) - 3$ and T is not a path. Set the initial root to be a branch vertex that is the endpoint of a thread, and when selecting a component of T^r, select one having the



Robber	Root	Probe	$P(v)$	Λ_v	b_v	Eliminated
d_9	a	f_1	8	$\{d_1, \dots, d_9\}$	a	$\lfloor b_1 \rfloor_a$
c_5	a	c_1	4	$\{e_1, c_2, \dots, c_5\}$	a	$-$
d_9	a	b_2	4	$\{d_2, d_5, \dots, d_9\}$	a	$\lfloor b_2 \rfloor_a$
d_9	a	c_1	5	$\{d_5, \dots, d_9\}$	b_3	$\lfloor a \rfloor_{b_3}$
c_5	b_3	d_5	3	$\{e_2, c_5\}$	b_3	$-$
d_9	b_3	b_3	2	$\{d_6, d_8, d_9\}$	b_3	$-$
d_9	b_3	d_6	4	$\{d_8, d_9\}$	c_5	$\lfloor b_3 \rfloor_{c_5}$
d_9	c_5	c_5	1	$\{d_8, d_9\}$	c_5	$-$
e_4	c_5	d_8	3	$\{e_3, e_4, e_5\}$	d_9	$\lfloor c_5 \rfloor_{d_9}$
e_4	d_9	e_3	2	$\{e_4, e_5\}$	d_9	$\lfloor e_3 \rfloor_{d_9}$
e_4	d_9	d_9	1	$\{e_4, e_5\}$	d_9	$-$
f_3	d_9	e_4	1	$\{f_2, f_3\}$	e_4	$\lfloor d_9 \rfloor_{e_4}$
f_3	e_4	f_2	2	$\{f_3\}$	f_3	$\{e_4, f_2\}$

Figure 1: This is an example of the robber locating game, where the robber's initial location is d_9. The table above details the implementation of our strategy on the pictured tree T. Since $2\ell(T) - 3 = 29 < n(T) - 2 = 39 < \Gamma(T) = 51$, we use Initialization 2 and root at a. Step 1 makes our first probe f_1, which gives $P(f_1) = 8$. Hence, the robber is known to be at one of $\{d_1, \dots, d_9\} = \Lambda_{f_1}$, which makes $b_{f_1} = a$. We deduce that the robber is not on $\lfloor b_1 \rfloor_a$ and eliminate that subgraph to consider a "new" tree which contains the robber. Since there are now no threads from a and the robber was not on level 1 or 2 during the last probe, Case 3 of Step 2 makes our next probe c_1. The remaining pursuit continues in this way, taking into consideration the possible locations of the robber from the previous probe. For instance, d_5 does not appear in Λ_{d_6} in line 7 because the robber could not have moved to d_5 from a vertex in Λ_{b_3} in line 6.

minimum number of leaves.

Initialization 3: $M = \Gamma(T)$ or T is a path. Set the initial root to be a central vertex of T, and arbitrarily select components of T^r.

The algorithm proceeds as follows.

Step 0: The First Probe.
If there are no threads from r and we have not yet probed a vertex, probe r and reroot to b_r. Otherwise, begin with Step 1.

Notice that the probe in Step 0 will only occur if the initial root is established in Initialization 3 and has no threads. The following two–step process is applied after each rerooting.

Step 1: Eliminate Threads Containing r.
If there are no threads from r, proceed to Step 2. Otherwise, arbitrarily pick a thread containing r and probe its leaf, say v. Since any vertex on a (v, r)–thread is a unique distance from v, the robber is located if he is on the (v, r)–path. Additionally, by the definition of b_v, if the robber is on the (r, b_v)–path, he is on b_v and is located. Thus, if the robber is not on the (v, b_v)–path, he is determined to be on $\lfloor b_v \rfloor_v \setminus b_v$. Consider the subtree $\lfloor b_v \rfloor_v$ as the "new" tree T containing the robber and reroot to b_v.

Step 1 repeats until the robber is located or there are no threads from the root. In Step 2, we direct our efforts to eliminating the components of T^r. Our approach differs based on which level of T the robber was on at the previous probe and will probe vertices in at most two components before rerooting again.

Step 2: Eliminate Other Components of T^r.

Case 1: The robber was on level 1 at the previous probe. If the previous probe was not at r, then the robber could move to r. Thus, we probe r to maintain RLP and reroot to r. If the previous probe was at r, we select a component of T^r according to our initialization, say T_1^r, and probe the neighbor of r in the selected component, say v. Since the robber was on level 1 and could not have moved to r under the no–backtrack condition, the robber must be on level 1 or level 2.

(a) If $P(v) = 1$ and the robber is not located, then the robber is on $\lfloor v \rfloor_r \setminus v$, which has at least two components (else the robber would have been located). We consider $\lfloor v \rfloor_r$, which is T_1^r, as the new T and reroot to v.

(b) If $P(v) > 1$, the robber is not on T_1^r. We consider $\lfloor b_v \rfloor_r \setminus T_1^r$ as the new T and reroot to b_v.

Case 2: The robber was on level 2 at the previous probe. Select a component of T^r (according to our initialization), say T_1^r, and let u be the neighbor of r in T_1^r whose neighbors in T_1^r are labeled $u_1, \ldots, u_m$ so that $n(\lfloor u_j \rfloor_r)$ is non–increasing. We may have to probe multiple neighbors of u without being able to eliminate any component of T^r, so we establish this reference point:

$(*)$ Probe the first u_j not already probed in this iteration of Step 2.

Since the robber was on level 2 of T before the probe at u_j, he is on level 1, 2, or 3 of T, which means $P(u_j) \leq 5$.

(a) If $P(u_j) = 1$, then the robber is on a vertex in $N(u_j)$. If he is on u, he could move to r. To maintain RLP for r, probe r. If the robber is not located, he must be on $\lfloor u \rfloor_r \setminus \{u, u_j\}$.

  i. If $P(r) = 2$, then by the no–backtrack condition, the robber is on u_i for some $i \neq j$. Consider $\lfloor u \rfloor_r \setminus \lfloor u_j \rfloor_r$ as the new T and reroot to u.

  ii. Otherwise $P(r) \geq 3$ and the robber is on $\lfloor u_j \rfloor_r \setminus u_j$. Note that $b_r \in \lfloor u_j \rfloor_r$. Consider $\lfloor b_r \rfloor_r$ as the new T and reroot to b_r.

(b) If $P(u_j) = 2$, then the robber is on u_i for some $i \neq j$. Consider $\lfloor u \rfloor_r \setminus \lfloor u_j \rfloor_r$ as the new T and reroot to b_{u_j}, which is u.

(c) If $P(u_j) = 3$, then the robber is either on level 1 in $T \setminus T_1^r$, or on level 3 in $T_1^r \setminus \lfloor u_j \rfloor_r$. If u_i is a leaf for each $j < i \leq m$, then the robber is on level 1 in $T \setminus T_1^r$ (he cannot be on T_1^r). We handle this case separately.

  i. If the robber cannot be on T_1^r, then he must be adjacent to r but not on u. Consider $T \setminus T_1^r$ as the new T and reroot to b_{u_j}, which is r.

  ii. Otherwise, probe r, which maintains RLP for r if the robber was on level 1.

    A. If $P(r) = 1$, note that the robber was not on T_1^r at u and cannot have moved there. Consider $T \setminus T_1^r$ as the new T and reroot to b_r, which is r.

    B. If $P(r) = 2$, then the robber is on level 2. Specifically, if the robber is on T_1^r, he is on $\lfloor u \rfloor_r \setminus \{\lfloor u_i \rfloor_r \colon\ i \leq j\}$. If the robber could be on T_1^r, then return to $(*)$; otherwise, consider $\lfloor b_r \rfloor_r$ as the new T and reroot to b_r.

    C. Otherwise $P(r) \geq 3$ and the robber is on $T_1^r \setminus \{\lfloor u_i \rfloor_r \colon\ i \leq j\}$. Consider $\lfloor b_{u_j} \rfloor_r \setminus \{\lfloor u_i \rfloor_r \colon\ i \leq j\}$ as the new T and reroot to b_{u_j}.

(d) Otherwise $P(u_j) \in \{4, 5\}$ and the robber is not on T_1^r. Consider $\lfloor b_{u_j} \rfloor_r \setminus T_1^r$ as the new T and reroot to b_{u_j}.

Case 3: The robber was not on level 1 or 2 at the previous probe. Assume the robber was on level d, with $d \geq 3$, at the previous probe. It will take the robber at least three turns to move to the root. Since we will probe at most two vertices in this case, the robber is unable to move to r. Thus, RLP is maintained for r. Select a component of T^r, say T_1^r, and probe a vertex, say v, in T_1^r on level $d - 1$ of T.

(a) If $P(v) < 2d - 2$, then the robber is on T_1^r. Thus $b_v \in T_1^r$. Consider $\lfloor b_v \rfloor_r$ as the new T and reroot to b_v.

(b) If $P(v) = 2d - 2$, then the robber is either on level $d + 1$ of $T_1^r \setminus \lfloor v \rfloor_r$, or on level $d - 1$ of some other component of T^r. Thus, at the next probe the robber could be on level d, $d + 1$, or $d + 2$ in T_1^r, or on level $d - 2$, $d - 1$, or d in any other component of T^r. Select another component of T^r distinct from T_1^r, say T_2^r, and probe a vertex in T_2^r on level $d - 2$ of T, say u.

   i. If $P(u) < 2d-3$, then the robber is not on T_1^r. Consider $\lfloor b_u \rfloor_r \setminus T_1^r$ as the new T and reroot to b_u.
   ii. If $P(u) = 2d-3$, then the robber is not on T_1^r or T_2^r. Consider $\lfloor b_u \rfloor_r \setminus \{T_1^r, T_2^r\}$ as the new T and reroot to b_u.
   iii. Otherwise $P(u) > 2d-3$ and the robber is not on T_2^r. Consider $\lfloor b_u \rfloor_r \setminus T_2^r$ as the new T and reroot to b_u.

(c) Otherwise $P(v) > 2d-2$ and the robber is not on T_1^r. Consider $\lfloor b_v \rfloor_r \setminus T_1^r$ as the new T and reroot to b_v.

3 Analysis

3.1 Analyzing the Strategy

Recall that a strategy S to locate the robber on a graph G has RLP for $r \in V(G)$ provided that if the robber is initially at r, or if at any point he moves to r, then the next probe will locate him at r. Because each root r is either probed or a unique distance from the probed leaf of a thread whenever the robber has the opportunity to be on or move to r, the strategy in Section 2 maintains RLP for each root r. Notice also that when we reroot from r to r' the robber is on the proper subgraph $\lfloor r' \rfloor_r \setminus r'$, and $T^{r'}$ has at least two components. The strategy therefore satisfies the assumptions of Lemma 1, which means it locates the robber.

Knowing that the strategy is a robber locating strategy, we turn our focus to proving Theorem 2 by counting the number of probes used to locate the robber. This amounts to counting the number of probes used between each rerooting and comparing that count to what was eliminated when narrowing our search to a subtree.

Proof of Theorem 2. In updating to a new tree before rerooting, the strategy identifies proper subgraphs of T on which the robber cannot be positioned and eliminates them from consideration. Each update eliminates at least one component of T^r. We consider the number of probes used to locate the robber after the robber is isolated to the "final" subtree T^*. Since at each step of the strategy T^r is a forest with at least two components, T^* contains at least two leaves and three vertices. If the robber is located by the probe in Step 1, the strategy uses at most $n(T^*) - 2$ and at most $2\ell(T^*) - 3$ probes after beginning to consider T^*. Otherwise, each component of $T^* \setminus r$ has at least three vertices, two of which are leaves in T^*, which makes $n(T^*) \geq 7$ and $\ell(T^*) \geq 4$. If the robber is located by a probe in Case 1 or 3 of Step 2, the two probes used by the strategy is less than $n(T^*) - 2$ and less than $2\ell(T^*) - 3$. Otherwise the robber is located by a probe in Case 2 of Step 2 where each probe is either at r or in a selected component of $T^* \setminus r$ with m leaves. In this case, $n(T^*) \geq 2m + 3$ and $\ell(T^*) \geq m + 2$ and the $2m$ probes used is less than $n(T^*) - 2$ and at most $2\ell(T^*) - 4$. Thus to show that the strategy meets the desired $n(T) - 2$ and $2\ell(T) - 4$ bounds, it suffices to show that the number of probes used before the strategy isolates the robber on T^* is at most $n(T \setminus T^*)$ and $\ell(T \setminus T^*)$, respectively. We show that at least x vertices and at least $2x$ leaves are eliminated when x probes are used between consecutive reroots.

In Step 1, a single probe either locates the robber or eliminates a thread that has at least one vertex, one of which is a leaf. Since threads containing r are eliminated exclusively in Step 1, components of T^r eliminated in Step 2 have at least three vertices, two of which are leaves. Case 3 of Step 2 uses at most two probes and eliminates at least one component of T^r. In order to eliminate a component in Cases 1 and 2 of Step 2, the strategy may require a single probe at the start of Case 1. Based on the robber's distance from the root after this probe, the strategy proceeds to either Case 1 or 2 of Step 2. If the strategy returns to Case 1, a second probe eliminates at least one component of T^r. If the strategy proceeds to Case 2, a second probe eliminates at least one component of T^r unless the strategy enters subcase c.ii.B. In this case, we execute $(*)$ at most m times since u has m neighbors in T_1^r. Assume $(*)$ is executed k times in Step 2 before rerooting. In order to repeat subcase c.ii.B, each $\lfloor u_j \rfloor_r$ must have at least 2 vertices, which means $|T_1^r| \geq 2k+1$ and $\ell(T_1^r) \geq k$. Hence, we use at most $2k$ probes and eliminate at least one component of T^r. Thus, each step eliminates at least as many vertices as the number of probes it uses. Therefore, if the strategy began with Initialization 1 and selects components in the prescribed order, the strategy uses at most $n(T) - 2$ probes to locate the robber. Also, if the strategy began with Initialization 2 and selects components in the prescribed order, the strategy uses at most $2\ell(T) - 3$ probes to locate the robber.

We now argue that the strategy uses at most $\Gamma(T)$ probes when it begins with Initialization 3. Recall

$$\Gamma(T) = \begin{cases} 2\Delta^2(\operatorname{rad}(T) - 2) + \Delta(17 - 6\operatorname{rad}(T)) + 5\operatorname{rad}(T) - 18, & \text{if } \operatorname{rad}(T) \geq 3 \\ 3\Delta - 4, & \text{otherwise,} \end{cases}$$

where Δ is the maximum degree of the initial tree T, which is at least 2 since $n(T) \geq 3$. Note that $\Gamma(T) \geq 3$ when $\operatorname{rad}(T) \geq 3$, and $\Gamma(T) \geq 2$ otherwise. A single probe from Step 1 locates the robber when T is a path. Since $1 \leq \Gamma(T)$, assume T is not a path. The first probe of the strategy is during Step 0 at r or Step 1 at a leaf. If $\operatorname{rad}(T) \leq 2$, a probe returns distance at most 2 when the robber is on the same component of T^r as the probe. Thus, the strategy uses at most two probes to eliminate any component of T^r. This implies that the strategy uses at most $2(\Delta - 1)$ probes to determine the component of T^r on which the robber is positioned. After rerooting, the remaining tree is a star with the center serving as the root. In this case, we use Step 1 at most $\Delta - 2$ times to locate the robber, each time using precisely one probe. Hence, we use at most $2(\Delta - 1) + (\Delta - 2) = 3\Delta - 4$ probes to locate the robber on a tree with radius at most 2.

We may thus assume that T has radius at least 3. We begin by examining Case 2 of Step 2 in which we repeat subcase c.ii.B. For each of the Δ neighbors of r, the strategy may require two probes for each of their $\Delta - 1$ other neighbors to eliminate a component of T^r. Of the Δ neighbors of r, the strategy only needs to consider $\Delta - 1$ of them before isolating the robber on a subtree of a component of T^r and rerooting to some r'. Note that the new tree under consideration has a smaller radius unless r' is a central vertex of the original tree; this can happen at most once since there are at most two central vertices in a tree. Similarly, the strategy implemented on a subtree of the initial tree may need to probe the root and use two probes while considering $\Delta - 2$ of the $\Delta - 1$ neighbors of the root. This repeats until either the robber is located or the tree under consideration has radius at most 2. At this point, the strategy could use two probes for the $\Delta - 2$ neighbors of r to isolate the robber on a star, and then at most $\Delta - 2$ probes to locate the robber. Thus, we have

$$\begin{aligned} \operatorname{loc}(T) &\leq 1 + 2(\Delta - 1)^2 + (\operatorname{rad}(T) - 3)(1 + 2(\Delta - 1)(\Delta - 2)) + 2(\Delta - 2) + (\Delta - 2) \\ &= 2\Delta^2(\operatorname{rad}(T) - 2) + \Delta(17 - 6\operatorname{rad}(T)) + (5\operatorname{rad}(T) - 18), \end{aligned} \tag{1}$$

as desired. If the strategy reroots anywhere other than subcase c.ii.B of Case 3 in Step 2, the strategy uses fewer probes to eliminate a component. Therefore, the bound obtained in (1) holds. □

3.2 Sharpness

In this section, we provide examples of trees for which the strategy meets the bounds in Theorem 2. Theorem 1 states that for a tree T on n vertices, $n \geq 3$, a robber is locatable in at most $n-2$ probes, with equality if and only if T is a star. We show that our strategy requires $n-2$ probes precisely when T is a star.

Corollary 1. *For a tree T on $n \geq 3$ vertices, the strategy requires $n-2$ probes to locate the robber if and only if $T = K_{1,n-1}$.*

Proof. Let $T = K_{1,n-1}$. Then $\mathrm{rad}(T) = 1$ and $\Delta(T) = \ell(T) = n-1$. So $2\ell(T) - 3 = 2n-5 \geq n-2$ and $\Gamma(T) = 3\Delta(T) - 4 = 3n-7 > n-2$ for $n \geq 3$. Thus, $M = n-2$ and the strategy begins in Initialization 1. Our strategy repeats Step 1, which results in iteratively probing the leaves of T. This is the same strategy presented in Theorem 3.3 of [8] for spiders that proves $\mathrm{loc}(K_{1,n-1}) = n-2$ as a corollary.

Now, let T be an arbitrary tree with maximum degree at least 3 and assume that the strategy uses $n-2$ probes to locate the robber. This only happens when $M = n-2$. Thus, the strategy begins in Initialization 1 and the first probe occurs in Step 1. As we progress through the strategy, since each T^r is a forest with at least two components, the last probe is on a tree with at least three vertices. Thus, the last probe eliminates at least two vertices. To attain $n-2$ probes before locating the robber, exactly two vertices must be eliminated by the last probe and the first $n-3$ probes must delete a total of $n-3$ vertices. From the proof of Theorem 2, we see that Step 1 deletes at least one vertex using one probe, and that the number of vertices deleted in Step 2 is always strictly greater than the number of probes used. Thus, in order to use $n-3$ probes to eliminate precisely $n-3$ vertices, each probe must occur during an iteration of Step 1 and eliminate exactly one vertex. Therefore, $T = K_{1,n-1}$. $\square$

Corollary 2. *For a tree T with ℓ leaves, the strategy requires $2\ell - 3$ probes to locate the robber if and only if T is a path.*

Proof. If T is a path, then $\ell = 2$ and $2\ell - 3 = 1$; the robber is located in Step 1 by the first probe, which is an endpoint of the path.

Suppose T is not a path. We will show that applying the strategy given in Section 2 uses at most $2\ell - 4$ probes to locate the robber. If the strategy begins in Initialization 3, then $M = \Gamma(T) \leq 2\ell - 4$, and the proof of Theorem 2 shows that at most $\Gamma(T)$ probes are needed. If the strategy begins with Initialization 1, then $M = n-2$ and we use at most $2\ell - 4$ probes unless $n-2 = 2\ell - 3$ and $n-2$ probes are actually needed. However, Corollary 1 shows that we use $n-2$ probes if and only if T is a star, and the only star that satisfies $n(T) - 2 = 2\ell(T) - 3$ is the path on 3 vertices.

Thus, we consider what happens when the strategy begins in Initialization 2. In this case, the initial root r is the endpoint of at least one thread, so the first probe occurs in Step 1 and eliminates a component of T^r that contains one leaf of T. We have shown that Cases 1 and 3 of Step 2 eliminate at least two leaves with two probes, and Case 2 of Step 2 may require up to $2k$ probes to eliminate k leaves. The last probe will eliminate at least two leaves of T, no matter what happens earlier. Thus, we see that there are $\ell - 3$ leaves that must be eliminated between the first and last probes. In the worst case, we must repeat Case 2 of Step 2 and use up to $2\ell - 6$ probes to eliminate these $\ell - 3$ leaves, for a total of $2\ell - 6 + 2 = 2\ell - 4$ probes, showing that $\mathrm{loc}(T) \leq 2\ell - 4$. $\square$

We give a family of trees that require $2\ell - 4$ probes to guarantee locating the robber using our strategy. Let T consist of a branch vertex r such that there is exactly one (u, r)-thread of order at least 4 from a leaf u to r, and the remaining neighbors of r are branch vertices $v_1, \ldots, v_{\Delta(T)-1}$ such that for each i satisfying $1 \leq i \leq \Delta(T) - 1$, $\lfloor v_i \rfloor_r$ is a path and $\lfloor v_i \rfloor_r \setminus v_i$ consists of exactly two paths, each of order at least 3 (see Figure 2). Since the radius of this tree is at least 4 and $\ell(T) = 2\Delta(T) - 1$, we have $\Gamma(T) \geq 2\ell - 4$. Thus, the strategy begins in Initialization 2 and the robber can evade capture for up to $2\ell - 4$ probes by starting at some v_i of T and alternating between levels 1 and 2 until T^r has two components, at which point the robber can prolong his evasion by one additional probe if he moves to level 1, then back to level 2, and then to level 2 of $\lfloor v_i \rfloor_r$ rather than to level 1.

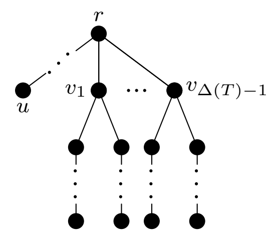


Figure 2: Trees that require $2\ell(T) - 4$ probes to locate the robber.

Finally, we give a family of trees that requires $\Gamma(T)$ probes for each T. This family contains trees T whose vertices have degree either $\Delta(T)$ or 1, and can be rooted at a branch vertex r so that all the leaves appear in a single level at least distance 3 from r. Note that $\Delta(T) \geq 3$ and T has at least 3 levels. Consider tree T with k levels and let $\Delta = \Delta(T)$. Under these conditions level i of T has $\Delta(\Delta - 1)^{i-1}$ vertices of degree Δ for $1 \leq i < k$, and level k has $\Delta(\Delta - 1)^{k-1}$ leaves. If T satisfies $\Gamma(T) < \min\{n(T) - 2, 2\ell(T) - 4\}$, then Initialization 3 makes a vertex of degree Δ the initial root and the strategy proceeds as in the proof of Theorem 2, requiring $\Gamma(T)$ probes to locate the robber if the robber iteratively adopts the strategy described in the previous paragraph. For a specific example, let $k = 4$ and $\Delta = 3$. Then $n(T) - 2 = 22$, $2\ell(T) - 3 = 21$, and $\Gamma(T) = 17$.

4 Concluding Remarks

The strategy presented in this paper answers a problem presented in [8] by providing a better bound for the location number of trees using parameters other than $n(T)$. This bound is not optimal since Corollary 1 implies $\mathrm{loc}(T) < n - 2$ for any tree T that satisfies $2\ell(T) - 3 = n - 2 \leq \Gamma(T)$ and is not a star. In particular, any caterpillar with one more leaf than non–leaf has a sub–optimal bound on $\mathrm{loc}(T)$. However, the strategy may use $\mathrm{loc}(T)$ probes for a tree even when the bound from Theorem 2 is not optimal. For example, Seager shows that $\mathrm{loc}(T) = \Delta(T) - 1$ when T is a spider [8]. Although Theorem 2 does not give a sharp bound for many spiders, the strategy in fact locates the robber using no more than $\mathrm{loc}(T)$ probes when T is a spider.

Given a specific tree, simple alterations of our strategy can locate the robber in fewer probes than we have shown. For example, when starting with some v_i for the initial root of the tree in Figure 2, two probes are used to eliminate $\lfloor v_i \rfloor_r$ and then the tree is rerooted to r. As described previously, the strategy uses at most another $2(\ell(T) - 2) - 4$ probes to locate the robber. Hence at most $2\ell(T) - 6$ probes are used. This means that the strategy presented in Section 2 does not necessarily use loc(T) probes for each tree T, which raises some natural questions. For which trees T does this strategy use loc(T) probes? Is there a method to choose an initial root for which this strategy uses loc(T) probes for all T? Finally, what is loc(T) for a given tree T, and is there a unified strategy that uses loc(T) probes to locate the robber on any tree?

References

[1] A. Bonato and R. Nowakowski. *The Game of Cops and Robbers on Graphs.* American Mathematical Society, Providence, RI, 2011.

[2] J. Carraher, I. Choi, M. Delcourt, L. H. Erickson, and D. B. West. Locating a robber on a graph via distance queries. *Theoret. Comput. Sci.*, 463(0):54 – 61, 2012. Special Issue on Theory and Applications of Graph Searching Problems.

[3] F. Harary and R. A. Melter. On the metric dimension of a graph. *Ars Combin.*, 2:191–195, 1976.

[4] J. Haslegrave. An evasion game on a graph. *Discrete Math.*, 314:1–5, 2014.

[5] A. S. LaPaugh. Recontamination does not help to search a graph. *J. Assoc. Comput. Mach.*, 40(2):224–245, 1993.

[6] R. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Math.*, 43(2-3):235–239, 1983.

[7] A. Quillot. *Jeux et pointes fixes sur les graphes.* Thèse de 3ème cycle, Université de Paris VI, 1978, pp. 131–145.

[8] S. Seager. Locating a robber on a graph. *Discrete Math.*, 312(22):3265 – 3269, 2012.

[9] S. Seager. A sequential locating game on graphs. *Ars Combin.*, 110:45–54, 2013.

[10] S. Seager. Locating a backtracking robber on a tree. *Theoret. Comput. Sci.*, 539:28–37, 2014.

[11] P. D. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *J. Combin. Theory Ser. B*, 58(1):22–33, 1993.

[12] P. J. Slater. Leaves of trees. In *Proceedings of the Sixth Southeastern Conference on Combinatorics, Graph Theory, and Computing (Florida Atlantic Univ., Boca Raton, Fla., 1975)*, pages 549–559. Congressus Numerantium, No. XIV. Utilitas Math., Winnipeg, Man., 1975.