# On the complexity of searching for an evader with a faster pursuer

Florian Shkurti[1] and Gregory Dudek[1]

*Abstract*— **In this paper we examine pursuit-evasion games in which the pursuer has higher speed than the evader. This scenario is motivated by visibility-based pursuit-evasion problems, particularly by the question of what happens when the pursuer loses visual track of the moving evader. In these cases the pursuer has two options for recovering visual contact with the evader: to perform search over the possible locations where the evader might be moving, or to clear the environment, in other words to progressively search it without allowing the evader to move into locations that have already been cleared. It has been shown that in sufficiently complex environments a single pursuer having the same speed as the evader cannot clear the environment. In this work we prove that computing the minimum speed which enables a faster pursuer to clear a graph environment is NP-hard. In light of this result we provide an experimental comparison of randomized and deterministic search strategies on planar graphs, which has practical significance in search and rescue settings.**

## I. INTRODUCTION

The central theme shared by the many variants in the family of pursuit-evasion games is that one or more pursuers want to search for, and in some cases to capture, a moving evader. This search paradigm is thus different from traditional search and path planning strategies, which assume that the target of interest is fixed and its position known.

Pursuit-evasion games are of great interest in robotics because of their numerous applications in settings where search in the physical world is involved. Many applications can be modeled as pursuit evasion, or have worst case performance bounds that are provided by results from pursuit evasion. While in many cases one agent may not actually be actively evading the other, performance guarantees regarding guaranteed capture from pursuit evasion provide assurance on when the agents will be able to meet. For example, search and rescue missions in mountains, where networks of hiking trails can prove to be very difficult to navigate, even for experienced mountaineers, often involve a search party that is looking for missing persons. A typical question studied in pursuit-evasion games is: given a map of the environment, what is the minimum number of searchers and a search plan that will guarantee that the missing persons are going to be found? This question is computationally intractable for general environments, as shown in [1], [2], [3].

Another example of pursuit-evasion games involves evaders that are diffusive. For instance, an oil spill or fire spreading around in the environment from a source can be modeled as an evader that must be completely *cleared* by the pursuers. In such cases, clearing strategies that do not allow *recontamination* of previously cleared regions are sought by the team of robot pursuers. In fact, such strategies are not only useful for restricting diffusive evaders, but also in cases where the pursuers do not know how many evaders are to be found in the environment. This is particularly relevant for intrusion detection performed by robot agents.

In this paper we analyze pursuit-evasion games in which the pursuer is faster than the evader, whom we assume to be non-adversarial but also non-cooperative. In other words, we assume that the evader's motion is independent of the pursuer's actions. We are motivated by scenarios of visual target following and in particular by the question of what happens when the pursuer loses track of the target, due to obstacles in the environment. The majority of existing approaches to visibility-based pursuit-evasion problems focus on pursuer strategies that consistently maintain visual contact with the evader, assuming that they both share the same upper bound on their speed [4], [5]. Depending on the initial positions of the two agents and the structure of the environment consistent visual contact might not always be possible. Yet, the question of whether the pursuer can recover visual contact with the evader if provided with sufficient speed has been largely ignored in the literature, one of the few exceptions being [6].

Clearing and search strategies are both relevant in addressing this question, so in this paper we consider both. We assume throughout this work that the pursuer and the evader have complete knowledge of the map of the environment, of their own location on this map, but not of each other's location. We also make the physically sensible assumptions that the two agents' motion happens concurrently, in other words that they do not take turns moving, and that they have no means of communicating with each other. The contributions of this paper are the following:

- The problem of finding the minimum pursuer speed advantage with which a single pursuer can clear an environment modeled as a graph is proven to be NP-hard. We show that this is the case even for star-shaped tree environments.
- Related problems, such as finding a clearing schedule, or clearing a graph within a certain time frame are proven to be at least as intractable.
- We provide bounds on the minimum speed required to clear complete trees of bounded degree, as well as a linear-time clearing strategy.
- We present simulations that perform a comparison of different deterministic and randomized search methods over planar graphs. These simulations indicate that one of the randomized search methods performs consistently

[1]F. Shkurti and G. Dudek are with the Center for Intelligent Machines (CIM), School of Computer Science, McGill University, Montréal, QC, Canada (`florian@cim.mcgill.ca`, `dudek@cim.mcgill.ca`)

better than the rest when the evader's motion is neither adversarial nor cooperative.

## II. Related Work

There is a vast literature on different variations of pursuit-evasion games, largely overlapping with works on searching. A number of survey papers have been devoted to the coverage of the most important results in these two streams of work. For instance, Fomin and Thilikos [7] provide a summary of the most interesting theoretical developments in this area, while Chung et al. [8] summarize results and experiments that are relevant to pursuit-evasion games performed in a field robotics setting.

A large number of works have been devoted to the study of the complexity of searching. For instance, Parsons [1] raised the question of what is the minimum number of pursuers required to deterministically clear an arbitrary graph, in other words, what is its *search number*. In that work, the search number of trees is computed. but for the case of arbitrary graphs, only bounds are provided. Megiddo et al. [3] showed that finding the search number of an arbitrary graph is NP-complete, and provided an algorithm for computing a clearing strategy for the team of pursuers on trees. Barriere et al. [9] examine how the search number changes if we impose restrictions on the clearing strategy, such as no recontaminations[1] for example. Borie et al. [10] present numerous results on the complexity of clearing schedules that are optimal with respect to time or distance travelled. In a continuous setting, Guibas et al. [2] show that computing the search number for an arbitrary polygonal environment is NP-hard, and they provide bounds on it. Murrieta-Cid et al. [11] prove that deciding whether a pursuer can maintain visual contact with an evader, given a polygonal environment and their initial positions, is NP-complete.

Aside from complexity considerations, many algorithms and heuristics for searching arbitrary environments have been proposed. For instance, Adler et al. [12] present a randomized algorithm for finding an adversarial evader in a graph, and compute the expected time by which the search will terminate. Bhattacharya and Hutchinson [13] compute a Nash equilibrium of purser and evader strategies in a polygonal environment, where the pursuer wants to maintain visibility of the adversarial evader. Kolling and Carpin [14] present an algorithm for clearing trees when the team of robots have limited sensing capabilities. Bhadauria and Isler [15] show that if the pursuers know each other's and the evader's position, then only three pursuers are required to capture the evader in any polygonal environment with obstacles.

A number of algorithms and heuristic have been designed with deployment in a field robotics setting, and several such pursuit-evasion experiments have been performed. Vieira et al. [16] performed experiments with networked robots that have complete knowledge of each other's approximate location, and are trying to find multiple evaders. Hollinger et al. [17] presented an approximation algorithm for the problem of selecting pursuer paths that are likely to cross frequent paths of a non-adversarial evader, and provided experimental validation. Kleiner et al. [18] presented a guaranteed search algorithm that assumes a digital elevation model as an input, and conducted large-scale outdoor experiments with multiple human searchers and evaders to demonstrate their approach. Lastly, Vidal et al. [19] performed pursuit-evasion experiments where the team of pursuers consisted of ground and unmanned aerial vehicles pursuing a ground evader robot. Thus, they managed to demonstrate both the practicality of their heuristics and the complexity inherent in deploying realistic pursuit-evasion scenarios.

## III. The Complexity of Clearing

We first examine the complexity of computing the minimum speed advantage the pursuer needs to have in order to clear the environment and thus find the evader. To this end we assume a discretized model of the environment, which we represent by a graph $G = (V, E)$. We also assume motion takes place in discrete time and that at the end of each time step both the pursuer and the evader will reside on a node of the graph. In other words, we consider node search, where edges signify transitions between nodes, without making it possible for the agents to reside on an edge at the end of a time step. As such, the evader can traverse one edge in one time step, while the pursuer can traverse $s_p \in \{1, ..., 2|E|\}$ edges [2], which is going to denote his speed. Under these assumptions we let $S_n$ be a star-shaped tree on $n$ nodes.

*Definition 1:* Let CLEAR$(S_n, s_p, t, v_0)$ be the decision problem that returns 'yes' if and only if $S_n$ can be cleared by a single pursuer with initial position $v_0$ on the tree and speed $s_p$ in time $t \in \mathbb{Q} \cap [0, \infty]$.
In order to classify the complexity of CLEAR and other related problems, we are going to need the following problem:

*Definition 2:* Let 3PARTITION$(x_1, x_2, ..., x_{3n})$ be the decision problem that returns 'yes' if and only if the positive integers $x_1, x_2, ..., x_{3n}$, whose sum is $nB$ where $B \in \mathbb{N}$, can be partitioned into triples that have the same sum, $B$. 3PARTITION has been shown to be strongly NP-complete [20], in the sense that it remains NP-complete even if the input numbers $x_1, x_2, ..., x_{3n}$ are bounded by a polynomial in $n$.

*Theorem 1:* CLEAR$(S_n, s_p, t, v_0)$ is NP-complete.

*Proof:* We herewith construct a polynomial-time Turing reduction from 3PARTITION to CLEAR. Given a set of positive integers $x_1, x_2, ..., x_{3n}$ we construct a star-shaped tree with $3n$ branches, as shown in Fig. 1. The $i^{th}$ branch contains $x_i$ nodes, including the root. We claim that a 3-partition exists if and only if this tree can be cleared by a single pursuer, having speed $s_p = 2(B-3)$, in time $t = n - \max_i\{x_i - 1\}/(2(B-3))$, starting from the root.
$(\Rightarrow)$ Assume a 3-partition exists, and consider an arbitrary triple. Its sum is $B = \sum_{i=1}^{3n} x_i/n$. A pursuer with speed $s_p = 2(B-3)$ can start from the root, clear the three branches corresponding to the triple and return to the root in one time

[1]Recontamination refers to the evader entering an already explored region

[2]$2|E|$ allows a tour of the graph in one time step

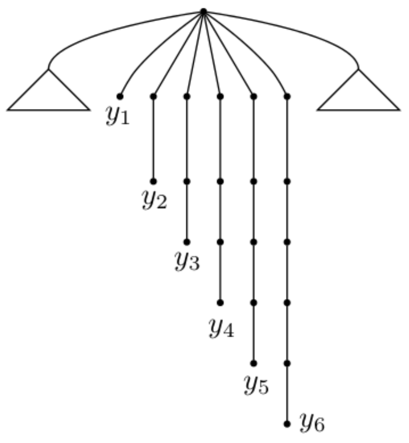

Fig. 1. An example of a tree construction used in the reductions. In this example, $(y_1, y_2, y_3) = (2, 3, 4)$ and $(y_4, y_5, y_6) = (5, 6, 7)$ are two triples of positive integers, some of the $3n$ numbers that are the input of a 3PARTITION instance.

unit, while the evader has crossed only one edge. The pursuer can thus clear the remaining $n-1$ triples in time $n-1$. Since the last branch of the last triple does not require going back to the root, we should only take it into account once. To minimize the time it takes to clear the tree the pursuer must leave the longest branch for last. So, the tree can be cleared in time $t = n - \max_i\{x_i - 1\}/(2(B-3))$.

($\Leftarrow$) Assume a 3-partition does not exist. Then for any partition of $x_1, x_2, ..., x_{3n}$ into triples, we will be able to find two sets $\{y_1, y_2, y_3\}$ and $\{y_4, y_5, y_6\}$ such that without loss of generality: $y_1+y_2+y_3 < B$, $y_4+y_5+y_6 > B$, none of the $y_i$ is $\max\{x_i\}$, and no repartition of these six numbers into two triples would make both sums $B$. An example of this is shown in Fig. 1. We want to show that a pursuer with speed $s_p = 2(B-3)$ cannot clear these two branches in time $t \leq 2$, while being able to clear the remaining $n-2$ branches in time $n - 2 - \max_i\{x_i - 1\}/(2(B-3))$. Suppose the pursuer decides to clear $y_1, y_2, y_3$ first, which will require a portion of the first time unit. If he spends the remaining of that portion at the root, to avoid possible recontamination by the evader, then he will need more than one time unit to clear $y_4, y_5, y_6$, so we are done. If he decides to spend the remaining portion of the first unit clearing part of $y_4, y_5, y_6$, even if he arrives at the root at the end of the second time unit, $y_1, y_2, y_3$ or some of the other triples will have been recontaminated. So, additional time will be required to re-clear them.

This shows that CLEAR is NP-hard. The trajectory of the pursuer on the tree is a polynomial-time-verifiable certificate for CLEAR$(S_n, s_p, t, v_0)$ which makes it NP-complete. ■

It is interesting to compare this result with the problem of computing the minimum number of pursuers, each with speed equal to or less than that of the evader, required to clear a graph. Even though that problem was shown to be NP-complete for general graphs, it is tractable for the case of trees, where a linear-time algorithm for computing the search number is available [1], [3], [10].

In our variant of the problem even the case of star-shaped trees is computationally intractable for a single pursuer. This is due to the size of the configuration space of the game, which for the case of star-shaped trees can be denoted by $(c_1, c_2, ..., c_b, p)$. $c_i$ is the number of nodes that are currently cleared on the $i^{th}$ branch, and $p$ is the current node at which the pursuer resides. Clearly, $c_i \in \Theta(n)$ and $p \in \Theta(n)$ so the size of the configuration space is $\Theta(n^{b+1})$.

If the number of branches is also $\Theta(n)$ then the size of the configuration space becomes non-polynomial. The goal of the pursuer is to find a simple path from $(0, 0, ..., 0, \text{root})$ to any of the following configurations $(\max\{c_1\}, \max\{c_2\}, ..., \max\{c_b\}, *)$ in this state space. The edges linking these configurations depend on the speed advantage $s_p$ that we allow the pursuer, which can range from 1 to $2(n-1)$. The more speed we allow, the denser the connectivity of the state space. Therefore, given a certain pursuer speed, a clearing strategy is possible when there is a path connecting the starting state to one of the goal states. When $s_p = 2(n-1)$, for example, there are direct edges connecting them, so the pursuer can clear the star-shaped tree in one time unit. Clearly, if we restrict the number of branches $b$ to be fixed, the size of the state space becomes polynomial and typical graph-searching algorithms can give us a clearing strategy. It is not immediately obvious, however, if in the case of general trees restricting the branching factor puts CLEAR in $P$. One of the things that we can say about these trees is the following:

*Lemma 1:* Let $T_h$ be a full tree of height $h$, with each node having fixed degree $b$. Then the minimum pursuer speed required to clear $T_h$ satisfies $h \leq s_p \leq 2bh$.

*Proof:* Let $s_h$ be the minimum speed required, with $s_0 = 0$ and $s_1 = 1$. We can extend an optimal clearing strategy on $T_{h-1}$ just by allowing $2b$ additional speed on the pursuer. Whenever the optimal strategy for $T_{h-1}$ visits the leafs of $T_{h-1}$ a $2b$ speed increase will allow it to clear the leafs of $T_h$ as well. So, $s_h \leq s_{h-1} + 2b$. Also, $T_h$ cannot be cleared with speed $s_{h-1}$, which can be seen from full binary trees of height 2,3 and 4, so $s_h > s_{h-1}$. Recursion over h gives us the result. ■

A linear-time clearing strategy for full trees of fixed degree is shown in Alg. 1. The pursuer must traverse the path produced by that algorithm with speed $s_p = 2bh$. Essentially, the path starts from the root, clears a new set of leafs, and goes back to the root. Each time a node is visited all its children are re-cleared, which prevents recontamination. At each return to the root the set of cleared nodes is augmented by at least $b$ leafs, so the algorithm terminates in linear time.

Other variants of CLEAR are also computationally intractable, even for the case of stars. For instance, computing the minimum time in which a star is clearable by a pursuer with speed $s_p$, or computing the minimum speed at which a star is clearable at a given time $t$. This is shown in the following theorems.

*Definition 3:* Let MINTIME-CLEAR$(S_n, s_p, v_0) = t^*$ be the optimization problem of computing the minimum time $t^*$ at which a star-shaped tree $S_n$ is clearable by a single

**Algorithm 1** CLEAR-FULL-TREE(root)

```
Let U be the nodes that are parents of leafs, ordered from left to right
for each node u in U do
    if u has not been visited then
        Let P be the path from root to u
        for $i = 1...|P| - 1$ do
            clear the children of node P[i] except P[i+1]
            go to P[i+1]
        end for
        for $i = |P|...2$ do
            go to P[i-1]
            clear the children of node P[i-1] except P[i]
        end for
    end if
    mark u as visited
end for
```

pursuer with speed $s_p$ who starts at node $v_0$.

*Definition 4:* Let MINSPEED-CLEAR$(S_n, t, v_0) = s_p^*$ be the optimization problem of computing the minimum speed $s_p^*$ with which a single pursuer can clear a star-shaped tree $S_n$ in time $t$, starting at node $v_0$.

*Definition 5:* Let RANGE-CLEAR$(S_n, s_p^{max}, t^{max}, v_0)$ be the decision problem that returns 'yes' if and only if the star $S_n$ can be cleared within time $t^{max}$ by a pursuer who has speed at most $s_p^{max}$.

*Theorem 2:* MINTIME-CLEAR$(S_n, s_p, v_0) = t^*$ and MINSPEED-CLEAR$(S_n, t, v_0) = s_p^*$ are NP-hard.

*Proof:* The exact same reduction from 3PARTITION as presented in Theorem 1 can be used in this case too. We observe that in that construction $t^* = n - \max\{x_i - 1\}/(2(B-3))$ is the minimum time at which the star can be cleared given the pursuer speed $2(B-3)$, and conversely, the minimum speed at which the star is clearable in the given time $n - \max\{x_i - 1\}/(2(B-3))$ is $s_p^* = 2(B-3)$. ■

*Theorem 3:* RANGE-CLEAR$(S_n, s_p^{max}, t^{max}, v_0)$ is NP-complete.

*Proof:* Via a reduction from the decision version of MINSPEED-CLEAR$(S_n, t, v_0) = s_p^*$, which is NP-complete. To answer that decision problem we can apply RANGE-CLEAR$(S_n, s, t, \text{root}(S_n))$ for all possible speeds $s$. The same polynomial-time-verifiable certificate that was used in Theorem 1 is valid here, too. ■

*Definition 6:* Let MINSPEED-CLEAR$(S_n, v_0) = s_p^*$ be the optimization problem[3] of computing the minimum speed $s_p^*$ with which a single pursuer can clear a star-shaped tree $S_n$, starting at node $v_0$.

*Theorem 4:* MINSPEED-CLEAR$(S_n, v_0) = s_p^*$ is NP-hard.

*Proof:* The reduction presented in Theorem 1 shows that a 3-partition exists if and only if the star can be cleared with a certain speed at a certain time. It does not guarantee that $2(B-3)$ is the minimum speed at which the star can be cleared. For example, a speed of $2\max x_i$ can also guarantee clearing of the star. We want to modify that reduction so that $s_p = 2(B-3)$ is indeed the minimum speed at which the modified star can be cleared.

Consider the following modification to the star presented in the proof of Theorem 1: we add $s_p^2$ new branches to that star, each containing $B-2$ nodes, including the root. A pursuer with speed $s_p = 2(B-3)$ can clear each of these new branches, starting and ending at the root, in exactly one time unit. The old portion of the tree can also be cleared with that speed.

We claim that the new star cannot be cleared with speed less than $s_p$. This is because the first attempt to clear one of the branches of length $B-2$ would allow recontamination of the root. At the next time step $3n + s_p^2 - 1$ branches will have an a recontaminated edge. Re-clearing these edges will take time at least twice the number of those edges, by which time the branch we started with is going to be completely recontaminated. ■

*Corollary 1:* The complexity of all the above problems remains true when stars are replaced with trees, and general graphs.

## IV. HEURISTIC SEARCH

The practical significance of these results is that in the scenario where a single pursuer has lost visual contact with the evader and wants to recover line of sight, computing time- or speed-optimal clearing strategies is computationally intractable, except in simple cases. Therefore, search strategies that do not guarantee clearing, but still can find the evader in bounded expected time, may be of interest. In the following section, we discuss a number of deterministic and randomized search heuristics that could be used by the pursuer in the presence of a non-cooperative and non-adversarial evader.

Note that we model the evader motion as a random walk since that provides an effective pessimistic bound for a cooperative evader (i.e. it is not helping, but it is not actively using oracular knowledge to avoid capture). This assumption has three attributes that are desirable for our experiments: first, the evader's motion is independent of the pursuer's plan (non-adversarial); second, his motion is continuous in the sense that he is not allowed to jump or teleport to distant nodes in the graph; and third, due to its lack of predictability an evader performing random walk can be seen as fully non-cooperative[4].

We examine four different pursuer search plans that do not generally provide deterministic guarantees on clearing: depth-first search (DFS), breadth-first search (BFS), shortest path to a random target node (SPRT), and performing random cycles in the graph (RC). In SPRT the pursuer randomly chooses a target node, and follows the shortest path from the current node to it. He repeats this process until the

[3]The difference between Def. 4 and Def. 6 is that the latter does not have a deadline by which the clearing must happen.

[4]A partially cooperative evader would allow some (predictable) determinism in his motion.

evader is found. The RC heuristic involves computing a fundamental set of cycles of the graph, using the algorithm presented in [21], and performing a random walk over those cycles. Decomposing the pursuer's trajectory into cycles is a technique with provably good capture times under a different set of assumptions on the motion of the pursuer and the evader, as shown in [12].

To compare these pursuer plans against each other we restrict the motion of the evader to maximal planar graphs. We generate these graphs on $n$ nodes and $3n-6$ edges randomly by recursive random construction of maximal planar graphs on $n-1$ nodes[5]. At each time step we allow the evader to cross one edge on the planar graph, and the pursuer to cross $s_p$ edges. We ran simulations over planar graphs with 50 to 1000 nodes, and for pursuer speeds 1 to 100. To quantify the performance of each pursuer strategy we do the following: for each pair of parameters we averaged the time it took the pursuer to find the evader over 100 random samples of maximal planar graphs and over 10 different initial positions for the two agents.

In addition, we evaluated each of the four pursuer strategies mentioned above in two settings: one where the pursuer is allowed to move on any edge of the planar graph, and another, where he is restricted to a spanning tree. So, in total we evaluated eight pursuer strategies. It is worth mentioning that applying the random cycles strategy on trees required modification: we modified it so that the pursuer starts from the root, randomly chooses a leaf and visits it, and then returns back to the root. In all eight cases the evader is moving on the planar graph.

The first observation that we can make from what our simulations indicate is that restricting the pursuer's motion to a spanning tree of the original planar graph does not affect the expected capture time asymptotically, as the speed of the pursuer increases. This is illustrated in Fig. 2 which shows the difference between the expected capture times on a spanning tree and the expected capture times on the planar graph. In fact, the same trend is visible in the difference of the standard deviations of capture times, which is illustrated in Fig. 3. What this indicates is that even though spanning trees only include approximately one third of the planar graph's edges, the fact that we are performing node search and we do not allow the evader to hide on edges makes the capture time depend mainly on how quickly each method can cover the nodes of the graph.

Our simulations also indicate that shortest-path to a random target consistently outperforms the other techniques both on spanning trees and on the full planar graph. In fact, in the case of spanning trees, among the 2000 combinations of $(s_p, |V|)$ parameters, SPRT had the lowest mean capture time in $74\%$ of the cases, while DFS was the best method in $18\%$. This is shown in Fig. 4 where we have plotted the heuristic with the lowest mean capture time for each set of parameters that we examined in our simulations. As expected from our previous observation regarding the restriction to a

[5]We used LEDA 6.3 and Python's networkx in our implementation

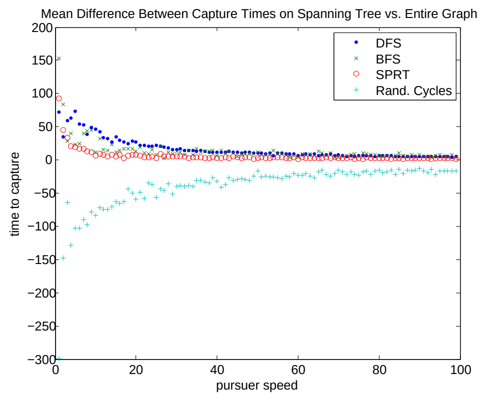

Fig. 2. The difference between mean capture times between restricted and unrestricted pursuer motion. The negative difference for the RC heuristic shows that the tree version of the heuristic performed consistently better than the graph version.

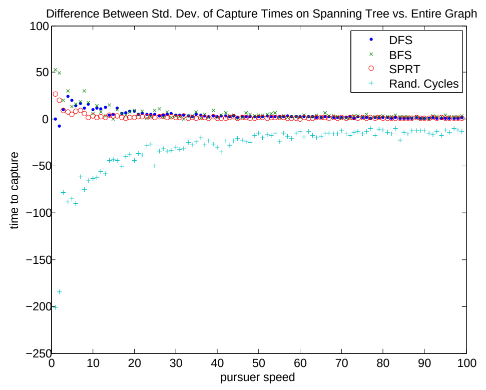

Fig. 3. The difference between standard deviations of capture times between restricted and unrestricted pursuer motion. Again, the negative difference for the RC heuristic shows that the tree version of the heuristic had a lower standard deviation than the graph version.

spanning tree, the distribution of methods with the lowest capture time does not change significantly when we transition from the tree to the graph. When pursuer motion on the graph is allowed SPRT performs best in $54\%$ of the cases, while DFS is best in $45\%$ of the cases, which is shown in Fig. 5. These results indicate that SPRT is a very promising heuristic, so its expected cover time, and meeting time with a random walk needs to be formally analyzed using results similar in flavor to [22].

## V. CONCLUSIONS & FUTURE WORK

In this work we examined the variant of pursuit-evasion games in which the pursuer is faster than the evader, which is an assumption that has been largely neglected by existing literature. Yet, it is a natural assumption in many settings, such as clearing contaminated waters, performing search and rescue missions using aerial vehicles, or recovering visual contact with a non-adversarial evader after losing line of sight due to the structure of the surrounding environment. We proved that the problem of computing the minimum speed at which a single pursuer can clear a graph is NP-hard, and so are many other problems related to it. We

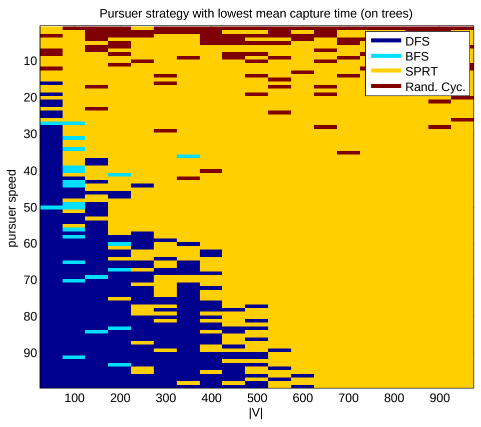

Fig. 4. Each entry with index $(s_p, |V|)$ in this matrix shows the label of the heuristic which had the lowest average capture time in the set of experiments that were performed with speed $s_p$ and node size $|V|$, and restricted the pursuer within the spanning tree. SPRT outperforms the other heuristics for this set of parameters.

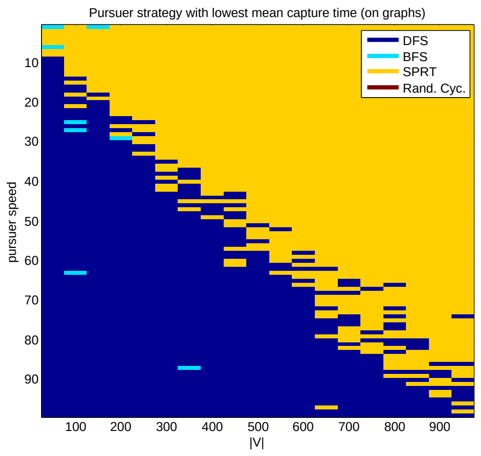

Fig. 5. Each entry with index $(s_p, |V|)$ in this matrix shows the label of the heuristic which had the lowest average capture time in the set of experiments that were performed with speed $s_p$ and node size $|V|$, and allowed the pursuer to move in the planar graph, just like the evader. SPRT and DFS were the heuristics that performed best.

provided logarithmic bounds on the minimum speed required to clear full trees of fixed degree, and we presented a linear-time algorithm that allows a pursuer to clear said trees. Finally, we provided empirical evaluation of search heuristics that enable the pursuer to find a non-adversarial and non-cooperative evader on planar graphs, and we identified a randomized heuristic which seems to outperform classical well-known heuristics. Avenues for future work include designing algorithms for motion in continuous environments with obstacles, in order to clear a diffusive evader. We are also interested in the impact of stronger models of evader behavior, since our eventual goal includes implementation in contexts where the evader behavior is genuinely cooperative.

## ACKNOWLEDGMENT

The authors would like to thank the Natural Sciences and Research Council of Canada (NSERC) as well as the Walter Sumner Foundation for their support.

## REFERENCES

[1] T. Parsons, “Pursuit-evasion in a graph,” in *Theory and Applications of Graphs*, ser. Lecture Notes in Mathematics, 1978, vol. 642, pp. 426–441.

[2] L. J. Guibas, J.-C. Latombe, S. M. Lavalle, D. Lin, and R. Motwani, “Visibility-based pursuit-evasion in a polygonal environment,” in *International Journal of Computational Geometry and Applications*, 1997, pp. 17–30.

[3] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou, “The complexity of searching a graph,” in *Journal of the ACM*, vol. 35, no. 1, Jan 1988, pp. 18–44.

[4] S. Bhattacharya and S. Hutchinson, “Approximation schemes for two-player pursuit evasion games with visibility constraints,” in *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[5] S. Lavalle, C. Becker, and J.-C. Latombe, “Motion strategies for maintaining visibility of a moving target,” in *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA*, 1997, pp. 731–736.

[6] B. Tovar and S. M. LaValle, “Visibility-based Pursuit–Evasion with Bounded Speed,” *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1350–1360, Nov. 2008.

[7] F. V. Fomin and D. M. Thilikos, “An annotated bibliography on guaranteed graph searching,” *Theoretical Computer Science*, vol. 399, no. 3, pp. 236–245, June 2008.

[8] T. H. Chung, G. a. Hollinger, and V. Isler, “Search and pursuit-evasion in mobile robotics,” *Autonomous Robots*, vol. 31, no. 4, pp. 299–316, July 2011.

[9] L. Barrière, P. Fraigniaud, N. Santoro, and D. M. Thilikos, “Searching is not jumping,” in *Graph-Theoretic Concepts in Computer Science, 29th International Workshop, WG*, 2003, pp. 34–45.

[10] R. Borie, C. Tovey, and S. Koenig, “Algorithms and complexity results for pursuit-evasion problems,” in *International Joint Conference on Artifical Intelligence*, 2009, pp. 59–66.

[11] R. Murrieta-Cid, R. Monroy, S. Hutchinson, and J.-P. Laumond, “A Complexity result for the pursuit-evasion game of maintaining visibility of a moving evader,” *IEEE International Conference on Robotics and Automation*, pp. 2657–2664, May 2008.

[12] M. Adler, H. Racke, N. Sivadasan, C. Sohler, and B. Vocking, “Randomized pursuit-evasion in graphs,” in *Proceedings of the International Colloquium on Automata, Languages and Programming*. SIAM, 2002, pp. 901–912.

[13] S. Bhattacharya and S. Hutchinson, “On the Existence of Nash Equilibrium for a Two-player Pursuit–Evasion Game with Visibility Constraints,” *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 831–839, Dec. 2009.

[14] A. Kolling and S. Carpin, “Pursuit-Evasion on Trees by Robot Teams,” *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 32–47, Feb. 2010.

[15] D. Bhadauria and V. Isler, “Capturing an evader in a polygonal environment with obstacles,” in *IJCAI*, 2011, pp. 2054–2059.

[16] M. Vieira, R. Govindan, and G. Sukhatme, “Scalable and practical pursuit-evasion with networked robots,” *Intelligent Service Robotics*, vol. 2, pp. 247–263, 2009.

[17] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias, “Efficient multi-robot search for a moving target,” *Int. Journal of Robotics Research*, vol. 28, no. 2, pp. 201–219, Feb. 2009.

[18] A. Kleiner, A. Kolling, M. Lewis, and K. Sycara, “Hierarchical visibility for guaranteed search in large-scale outdoor terrain,” *Autonomous Agents and Multi-Agent Systems*, vol. 26, no. 1, pp. 1–36, Aug. 2011.

[19] R. Vidal, O. Shakernia, H. Kim, D. Shim, and S. Sastry, “Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation,” *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 662 – 669, oct 2002.

[20] M. R. Garey and D. S. Johnson, “Strong np-completeness results: Motivation, examples, and implications,” *Journal of the ACM*, vol. 25, no. 3, pp. 499–508, July 1978.

[21] K. Paton, “An algorithm for finding a fundamental set of cycles of a graph,” *Commun. ACM*, vol. 12, no. 9, pp. 514–518, Sept. 1969.

[22] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovasz, and C. Rackoff, “Random walks, universal traversal sequences, and the complexity of maze problems,” in *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, 1979, pp. 218–223.