2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Searching for a One-Dimensional Random Walker: Deterministic Strategies with a Time Budget When Crossing is Allowed

Narges Noori, Alessandro Renzaglia and Volkan Isler

***Abstract*— We present deterministic strategies for capturing a target performing a discrete random walk on a discretized line segment. The searcher has a limited time budget. Its goal is to maximize the probability of capturing the target within the budget. A challenging aspect of our model is that the target can cross the searcher without being captured when they take the same edge at the same time in opposite directions. We present a Partially Observable Markov Decision Process (POMDP) approach for finding the optimal search strategy. We also present an efficient approximate solution to the POMDP. The strategies found by this approach reveal structural properties of the efficient search strategies which we exploit to solve the problem efficiently without running the POMDP.**

I. INTRODUCTION

Pursuit evasion games are studied in robotics community for various applications such as finding victims of a disaster and capturing adversarial intruders [1]. In a typical pursuit-evasion game there are two players: the evader and the pursuer. The evader tries to avoid being captured by the pursuer. Meanwhile, the pursuer tries to capture the evader. The main focus of the literature in this area is to design pursuit strategies that guarantee capture in the worst case scenario i.e. when the evader can perform the best possible strategy to avoid capture. An advantage of modeling such applications as pursuit-evasion games is this worst-case guarantee. Even if the evader is not adversarial, a pursuit strategy, if it exists, guarantees capture. However, this generality comes at a cost. Considering the worst case behavior can be too conservative, and there might be instances that a worst case solution may not exist. In other words, the adversary can escape from any deterministic pursuit strategy. For example, consider a simple game which takes place on a path with nodes $[0, 1, \ldots, N]$ and equal length edges. The players move simultaneously along the edges. The pursuer captures the evader when they are on the same node. However, crossing is allowed: If the pursuer is at node i, the evader is at node $i + 1$, and they move toward each other, the evader is not captured (we will justify this model shortly). It is not too difficult to see that for any deterministic pursuer strategy, there is an evader strategy which avoids capture indefinitely. This evader strategy is as follows: It waits until the searcher is at one of the neighboring nodes, and as the pursuer moves onto its current location, it moves to the pursuer's location. See Fig. 1(b).

The authors are with the Department of Computer Science & Engineering, University of Minnesota, Minneapolis, USA. This work was supported by National Science Foundation Awards #1111638 and #0917676. Emails: {noori,arenz,isler}@cs.umn.edu.

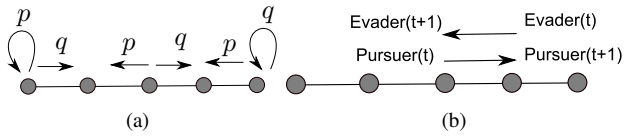

Fig. 1. (a) Target's motion model. Here $p = (1 - q)$. (b) The pursuer will miss the target if they cross each other.

The game mentioned above is inspired by our ongoing work on finding radio-tagged invasive fish in inland waters using an Autonomous Surface Vehicle (ASV) [2]. When the ASV is moving, the noise from the motors interferes with the signal from the fish and makes it very difficult to detect them. Therefore, we resort to visiting a discrete set of locations and stopping the motors when scanning for the fish at these locations. Therefore, the fish will be detected only when the ASV is on a discrete set of points. Often, the fish are likely to be near the shore of a lake and the boat's path will be a simple path. Therefore, the environment can be represented as a discrete line segment as described above.

From the previous discussion, if we model the fish as an adversary, then there is no deterministic strategy which guarantees capture. On the other hand, it is very unlikely that the fish execute an adversarial strategy. Therefore, in this work, we study the problem of finding a target when it is moving according to a stochastic model[1]. In the absence of any other information, a natural movement model is given by the *simple random walk*: the target moves either left or right with equal probability at each time step. We study the search problem on the one-dimensional environment model given above. In this environment, if crossing was not allowed, the best strategy is to simply sweep the line from left to right. However, as we will see below, this strategy is not very efficient when crossing is allowed. It turns out that the optimal solution under the crossing model has not been studied previously in the literature.

Specifically, we study the problem of designing a deterministic search strategy that maximizes the probability of capture when there is a time constraint. For this version we make the following contributions: We first show how to compute the capture probability for a given strategy. This is difficult when crossing is allowed because at any given time the target has non-zero probability of being on either side of the pursuer, and the belief (the distribution of target's possible locations) is not smooth. See Fig. 3 for an example.

[1]Hereafter, we switch the terminology from "evader" to "target" to emphasize the non-adversarial nature of this player's movement.

We then formulate the problem of computing the optimal search strategy as finding the optimal strategy of a Partially Observable Markov Decision Process (POMDP) [3]. We define the state of the pursuer as its location plus the belief about the location of the target. The exact representation of the belief is intractable since there will be exponentially many states. On the other hand, the non-smooth nature of the belief makes it difficult to approximate it with a parametric distribution. To tackle this challenge, we propose a method based on binning the belief in non-uniform bins and then approximating the belief in each bin with a uniform distribution. The width of the bins increase exponentially as the bins get farther away from the current position of the searcher. This incorporates the intuition that the shape of the belief in farther points is not crucial for the decision on whether to move or not. We show that the POMDP can be solved using this representation. However, the solution is still time-consuming. Therefore, we also present an alternative structured solution based on the POMDP solution and show, in simulations and also using our analysis, that its performance is close to the optimal POMDP solution.

Finally, we note that in a parallel submission [4], we study the problem when the searcher has a limited energy budget and the wait action has a different cost than the movement actions (left or right). For this more general version of the problem, computing optimal solutions using the POMDP formulation becomes infeasible due to the large number of states. Therefore, we restrict our attention to a class of randomized strategies and present closed-form solutions.

The paper is organized as follows. In Section II, we present related work. Section III presents the problem formulation. In Section IV, we show how the probability of capture for a given strategy can be computed. The proposed approach for computing the optimal strategy is presented in Section V. Finally, we present concluding remarks in Section VI.

II. Related Work

Various properties of random walks on graphs such as the expected time to visit every node of the graph have been studied extensively [5]. Continuous random walks are also well-studied [6]–[8]. Although random walks have been the subject of many works, none of them has addressed the problem of capturing the random walker restricted to constraints such as limited time budget or energy budget. The proposed pursuit strategy in [9] ensures capturing a random walker by a pursuer who can observe it at all times. In this work, however, the searcher cannot observe the target unless they are on the same node. Here, we devise deterministic strategies while in [4] randomized strategies are studied for capture subject to limited energy and different costs for move and wait actions. In recent work [10], we studied the problem when crossing is not allowed. There, the question is how much the searcher should wait and how much it should move subject to the different costs of these actions. In this work, however, we consider the case that by crossing the searcher misses the target.

III. Problem Statement

In this section, we formalize the search problem. We will use a line segment with discrete locations $[0, 1, \ldots, N]$ as the environment. The target starts from an unknown node, and thus we assume that the initial probability distribution of target's possible locations is uniform $\frac{1}{N+1}$. Afterward, the target performs a *simple random walk* as follows. From location $0 < i < N$, with probability q it moves one unit to the *right*, and with the remaining probability $(1 - q)$ it moves one unit to the *left* (see Fig. 1(a)). We assume that the boundary points 0 and N are reflective: at 0, the target *stays* with probability $(1 - q)$; at N, the target *stays* with probability q.

Note that we could have a non-zero probability of staying at the current position in addition to the left and right motions. However, in this paper, we consider the case that the probability of stay is zero and the target only moves to the right or to the left. We focus on this case because it is more challenging since the belief function is not smooth (Fig. 3). Moreover, throughout the paper, we consider the case that $q = (1 - q) = 0.5$, but our proposed approach works for other values of q as well as for the case that there is a non-zero probability of stay for the target.

The searcher starts from the left-most point i.e. $i = 0$. At each time step, it can move to the right or to the left or stay at its current node. Throughout the paper, we refer to these actions as R, L, S respectively. The searcher's strategy is defined as a sequence of these actions e.g. $R^i S^j L^k$ represents i steps to the right, j stay actions and then k steps to the left.

The searcher captures the target when both of them are on the same node simultaneously. Given a time constraint T, the task is to design the capture strategy $\mathcal{S}$ such that the probability of capture is maximized.

IV. Capture Probability of a Given Strategy

In this section, we show how the probability of capture for a given strategy can be computed. We then use this technique to compare strategies. Let us denote the searcher's location at time t by $s(t)$ and also the target's location by $\mathcal{E}(t)$. Also, $\mathcal{E}_0$ denotes the initial location of the target. The searcher is performing the strategy $\mathcal{S}$ given as a sequence of actions R, L and S. Fig. 2 shows an illustrative example of the location of the searcher for a specific strategy, and the target for a specific random walk path, as a function of time.

We are interested in computing the probability of capturing the target at time t: $P_{capture}(t)$. The capture events are those that the searcher and the target are at the same position at the same time. Thus, we must consider the target's paths that end up at $s(t)$ at time t, i.e. $\mathcal{E}(t) = s(t)$. Counting the events that the target starts at $\mathcal{E}_0$ and reaches $s(t)$ at t is not too difficult. However, we can not simply sum up these events to obtain $P_{capture}$ because they are overlapping. We need to satisfy two extra conditions which make the problem challenging. First, we must only count the events such that the target has not been captured sooner than t. Second, since crossing is allowed, path interactions such as the one marked by arrow

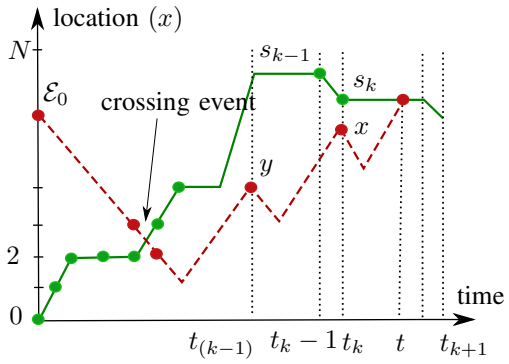

Fig. 2. The position of the players as a function of time. The target's path is shown in dashed lines. The time marked by the arrow is not capture because crossing is allowed.

in Fig. 2 does not count as a capture event. In order to tackle these two challenges, we look at the searcher's path $s(t)$ as a piecewise linear function. That is $s(t)$ is composed of a set of time intervals $[t_k, t_{k+1})$ such that $s(t)$ is constant in $[t_k, t_{k+1})$ and has the value s_k. Note that the intervals are right-open. For example, in Fig. 2 the searcher is at s_{k-1} in the time interval $[t_{k-1}, t_k)$, i.e. from t_{k-1} to $t_k - 1$.

In order to compute $P_{capture}(t)$, we take a divide and conquer approach to recursively compute the target paths that yield capture at time t but are *safe* before t. By *safe* we mean that the target has not been captured before time t. To do so, we consider the intervals before t. Each time-interval acts as the basis in our divide and conquer method. We can overcome the two issues mentioned above as follows. Since in each interval, $s(t)$ is constant, we don't have the crossing events such as the one marked by arrow in Fig. 2. We also can enumerate the target paths that co-locate the target and the searcher for the *first* time at t by counting the target paths that are completely below or above s_i during $[t_i, t_{i+1})$ for $i < k$. This enables us to compute the capture events using recursive equations as follows. Let us introduce the following probabilities:

- $P_{safe}(x, t)$: the probability that the target *safely* arrives at location x at time t. Here *safe* refers to the fact that the target has not been captured before time t.
- $P_{safe}(x_1, t_k, x_2, t_{k+1})$: assuming that the target is at x_1 at time t_k, $P_{safe}(x_1, t_k, x_2, t_{k+1})$ is the probability that it *safely* reaches x_2 at time t_{k+1}. Here, *safe* refers to the fact that the target is staying above or below s_k during $[t_k, t_{k+1})$. Note that this function is different from the previous one in the sense that it counts the safe events in a single interval $[t_k, t_{k+1})$.
- $F(x_1, x_2, t)$: the probability that the target starts at location x_1 and for the first time reaches at x_2 after t time steps (first passage probability).
- $G(x_1, x_2, t)$: the probability that the target starts at location x_1 and reaches at x_2 after t time steps (not necessarily for the first time).

Recall that our goal is to compute the probability of capturing the target at time t i.e. $P_{capture}(t)$. First, we consider the time interval $[t_k, t_{k+1})$ that t belongs to. See Fig. 2. During this time interval the searcher is at s_k. At time t_k the target can be anywhere except s_k. Let x be the target's position at time t_k. The capture events can then be described as follows: The target safely arrives at x at time t_k i.e. $P_{safe}(x, t_k)$ and then, from x, it reaches s_k for the first time after $t - t_k$ time steps:

$$P_{capture}(t) = \sum_x P_{safe}(x, t_k) F(x, s_k, t - t_k) \quad (1)$$

The safe events $P_{safe}(x, t_k)$ can be obtained from the following recursive equations:

$$P_{safe}(x, t_k) = \sum_y P_{safe}(y, t_{k-1}, x, t_k) P_{safe}(y, t_{k-1}) \quad (2)$$

The probability function $P_{safe}(x_1, t_k, x_2, t_{k+1})$ is obtained from:

$$P_{safe}(x_1, t_k, x_2, t_{k+1}) = G(x_1, x_2, t_{k+1} - t_k) - \sum_{t'=0}^{t_{k+1}-t_k} (F(x_1, s_k, t') G(s_k, x_2, t_{k+1} - t_k - t')) \quad (3)$$

Equation (3) counts the events that the target arrives at x_2 starting from x_1 while not crossing s_k by excluding the events that cross s_k (second term in r.h.s. of (3)) from the total number of events (first term in r.h.s. of (3)). Finally, we refer the interested reader to [11] for computing $G(x_1, x_2, t)$ and $F(x_1, x_2, t)$.

V. The Proposed Search Strategy

We now address the problem of designing an efficient search strategy. We consider two cases: 1) T is enough for at least two sweeps of the line i.e. $T > 2N$; 2) T is less than the time required for two sweeps i.e. $T \leq 2N$. In the first case, we present intuitive strategies and show that they guarantee a high probability of capture. In the second case, we provide a POMDP-based approach and propose strategies that are close to the POMDP solution.

A. When Time is Enough for Two Sweeps

In this case, the searcher has enough time for at least two sweeps $T > 2N$. One intuitive strategy is to sweep the whole line twice. However, if the searcher moves all the time, the parity of its distance to the target will change only when the target does a stay action at one of the boundary points (Fig. 1(a)). This is because at other points the target moves one step to the right or to the left, and thus its distance to the searcher changes by two. Therefore, we add a wait step in order to increase the probability of capture in the events that the target starts from an odd node. We call this strategy the *Sweep* strategy: the searcher moves all the way to the right, waits for one step at the last point N and then moves back to the left toward 0 ($R^N S L^N ...$). We also present a second strategy which we call *StopAndGo*: the searcher moves for one step, then waits for one step and so on ($RSRS...$). The probability of capture for these

two strategies is computed exactly for various values of N using the analysis in Section IV. These strategies achieve a very high probability of capture: 0.95 and 0.90 for Sweep and StopAndGo respectively. Since the highest possible performance for the probability of capture is one, we pick these two strategies as our best strategies for $T > 2N$.

B. When Time is Not Enough for Two Sweeps

In this case, the time constraint is less than the time required for two sweeps $T \leq 2N$. Therefore, the pursuer cannot perform the strategies presented in Section V-A. In the following, in order to find the search strategy that maximizes the probability of capture, we first formulate the problem as a Markov Decision Process (MDP). We then investigate the solutions found by this MDP.

C. MDP Formulation

Let us first briefly review the MDP setup. An MDP is composed of four components [12]: 1) The set of states; 2) The set of actions that the agent (searcher) can perform; 3) The state transition probability function upon performing each action; 4) The reward obtained after each state transition.

Suppose that we define the states of the MDP as $(\mathcal{P}, \mathcal{E})$ where $\mathcal{P}$ is the position of the searcher and $\mathcal{E}$ is the position of the target. Apparently, we cannot use this setup for the states of MDP since the location of the target is not observable to the searcher. The searcher's only observation is that it hasn't captured the target yet. Therefore, our problem is in fact a Partially Observable Markov Decision Processes (POMDP). However, we can convert it to an MDP by defining the states as the searcher's belief about the target's location [3]. The goal is to find the policy that maximizes the reward collected by the searcher upon execution of the policy. The following are the sets which define our MDP.

• States: Each state has two parts: the current position of the searcher, and the belief of the searcher about the position of the target. Let us denote the searcher's belief by $B = [p_0, p_1, ..., p_N]$ where p_i is the probability that the target is at i. We also have the capture state which is the terminal state.

• Actions: The set of actions that the searcher can perform in each state. These actions are moving to the left, to the right and staying at the current node.

• Reward: We set the reward for all state transitions to zero except the transition to the capture state which is one. Thus, the policy that maximizes the value of states is in fact the one that maximizes the probability of capture.

The exact representation of the searcher's belief, $B = [p_0, p_1, ..., p_N]$, will lead to exponential number of states which is impossible to tackle. One method is to approximate the belief by a specific function which can be represented by a small number of parameters as in [10] for the case that crossing is not allowed. However, this approach cannot be applied to the crossing case directly, because here the belief is not a smooth function. A sample of the searcher's belief is shown in Fig. 3.

Intuitively, the shape of the belief at farther points is less important for the searcher in picking the best action at its current position. This is because by the time that the searcher reaches those points, the shape of the belief will change. Therefore, we represent the belief by bins with exponentially increasing width as follows. Each bin starts at $c \pm 2^i, 0 \leq i \leq \log(N)$. The approximate belief in each bin is uniform which we compute as follows. We first compute the cumulative belief in each bin. Then we take the average of this cumulative value in the corresponding bin. Finally, we assign the closest discretization level to this average as the bin value. An example is shown in Fig. 3.

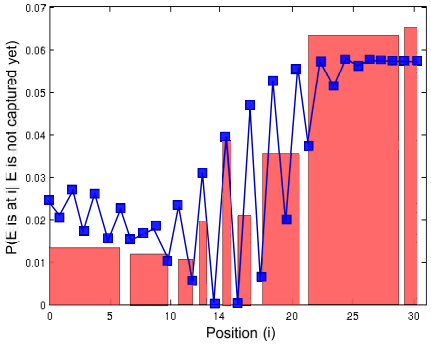

Fig. 3. The actual belief (blue) and its approximation using bins (the red).

D. MDP Strategies

In this section, we first present the proposed strategies for $T \leq \frac{3}{2}N$ and then we consider $\frac{3}{2}N < T < 2N$.

We implemented our MDP method by using INRA MDP MATLAB Toolbox [13]. The simulations are done on Dell Poweredge 6950 with $14GB$ memory. We used 50 levels for discretizing each bin. Thus, if there are n bins, the number of possible states would be 50^n.

E. When Time is less than 1.5 sweeps

The strategies found by MDP for $N = 20$ and $N = 31$ are shown in Table I and Table II respectively. A first interesting result is that, as shown in these tables, the MDP solutions have a common property: the stay actions are uniformly distributed among the right actions. In other words, the strategies are of the form $(R^k S)^m$. Let us refer to this class of strategies as the uniform strategies. Table I and Table II also present the best uniform strategy for the same values of N and T which are obtained by changing the number of rights k in each group of $(R^k S)$ and computing the probability of capture using the analysis in Section IV. From these tables, the uniform strategy is very close to the MDP solutions. There are two exceptions in which the uniform strategy outperforms the MDP one: for $N = 31$ and $T = 35, 44$. This is an artifact of the approximation error of the MDP.

• *What is the best k in $(R^k S)^m$?*

As shown in Tables I and II, in some uniform strategies there are extra right actions at the end of the strategy e.g. for $N = 20$, $T = 14$ the best uniform strategy is

	$T=9$	$T=14$	$T=19$	$T=24$	$T=29$
MDP strategy	RSR^2SR^3S	$(RS)^2R^3S(R^2S)^2$	$(R^3S)^4R^2S$	$R^2SR^3S(R^2S)^3(R^3S)^2$	$R^2SR^3S(R^2S)^3$ $(RS)^2R^3SR^2SRS$
MDP capture probability	0.2966	0.4201	0.5711	0.6991	0.8020
Uniform strategy	$(R^3S)^2R$	$(R^3S)^3R^2$	$(R^5S)^3R$	$(R^3S)^6$	$(R^2S)^9S^2$
Uniform capture probability	0.2971	0.4294	0.5642	0.7095	0.8031

TABLE I

THE MDP SOLUTIONS FOR $N = 20$.

	$T=9$	$T=14$	$T=24$	$T=35$	$T=44$
MDP strategy	R^3SR^4S	$(R^2S)^3R^4S$	$R^3SR^5SR^3SR^6SR^2S$	$R^{30}S^5$	$RS^3R^{19}SR^3SR^2S(RS)^2R^2S^3RS^3$
MDP capture probability	0.1978	0.2830	0.4625	0.58	0.675
Uniform strategy	$(R^3S)^2R$	$(R^3S)^3R^2$	$(R^3S)^6$	$(R^4S)^7$	$(R^2S)^{14}R^2$
Uniform capture probability	0.1949	0.2818	0.4656	0.6638	0.787

TABLE II

THE MDP SOLUTIONS FOR $N = 31$.

$(R^3S)^3R^2$. In order to avoid the effect of these extra rights on the performance, we compare uniform strategies which are multiples of (R^kS) groups, i.e. when $\mathcal{S} = (R^kS)^m$ and $N = km$. Fig. 4-top-right shows this comparison for the following strategies: $(RS)^{35}$ and $N = 35$, $(R^2S)^{17}$ and $N = 34$, $(R^3S)^{12}$ and $N = 36$, $(R^4S)^9$ and $N = 36$, $(R^5S)^7$ and $N = 35$, $(R^6S)^6$ and $N = 36$, $(R^7S)^5$ and $N = 35$. A similar comparison is shown in Fig. 4-top-left for values of N around 20. This comparison suggests that the performance of $(R^2S)^m$ is comparable to the other uniform strategies and sometimes it is the best. Thus, we pick $(R^2S)^m$ as our candidate strategy.

• *What if there are more than one stop actions in the uniform strategies?*

The next question that arises is how does $(R^2S)^m$ perform with respect to other strategies. We investigate this question by comparing $(R^2S)^m$ with the Uniform strategies that have more than one stop i.e. $(R^kS^n)^m$.

Fig. 4-middle-left shows the comparison of $(R^2S)^m$ with $(R^kS^n)^m$ for $N = 40$ and $T = 60$ obtained from simulation. Note that in these strategies we have left actions when the searcher arrives at the last point. According to this figure, the performance degrades as the number of stops in $(R^kS^n)^m$ increases, and $(R^2S)^m$ is better than any other.

Fig. 4-middle-left suggests that there is an optimal value for the number of rights k in $(R^kS)^m$. Fig. 4-middle-right depicts the performance of uniform strategies with only one stop and different number of rights $(R^kS)^m$. The result here also confirms that $(R^2S)^m$ is a good choice among $(R^kS)^m$.

• *What if the distribution of stays among right actions is non-uniform?*

Fig. 4-bottom-left shows the comparison of $(R^2S)^m$ with the non-uniform strategy $R^{16}S^{10}R^8S^2R^4SR^2S$, the sweep strategy R^NSL^{T-N-1}, and the MDP solution. For the comparison we use the analysis presented in Section IV. The $(R^2S)^m$ outperforms these strategies. Note that the capture probability of the MDP solution is lower than $(R^2S)^m$ due to the approximation error in belief representation.

F. When Time is greater than 1.5 sweeps

Let us now investigate the case that $1.5N \leq T < 2N$. We compare four strategies. The first one, is the Sweep strategy we introduced in Section V-A. Then, we define the following strategies:

- $(R^2S)^{N/2}(L^2S)^m$: repeat the pattern R^2S until the searcher is at node N, then L^2S for the rest.
- $(R^3S)^kL^m$: repeat the pattern (R^3S) until the searcher is at node N, then move back to the left.
- RightLeftRight: move to the right for $\frac{3N}{4}$, then turn back and move to the left for $\frac{N}{2}$, and then move to the right for the rest of time-steps.

As shown in Fig. 4-bottom-right for $N = 72$, the performance of $(R^2S)^{N/2}(L^2S)^m$ is better for $T < 120$, and then afterwards the Sweep strategy becomes better. It is worth emphasizing that both are above 0.81 after time 120. We observed the same behavior for $N = 30$ and $N = 54$. Thus, we declare both of them as our candidate strategies for the case that $1.7N < T$. When $1.5N < T < 1.7N$, we consider $(R^2S)^{N/2}(L^2S)^m$ as our best strategy.

VI. CONCLUSION

In this paper, we studied deterministic strategies for capturing a random walker on a line segment. We presented a POMDP approach and we incorporated belief binning as a way to tackle the very large dimension of the belief space. We gave the solutions found by this POMDP for manageable problem sizes. These solutions led us to a specific class of strategies where the stay actions are uniformly distributed among the move actions. We then investigated the number of stays and rights and showed that two rights followed by a single stop action (or two left actions and a stop action when the time budget is large) has good performance comparable to the optimal solution. A summary of the best strategies found for different cases is depicted in Table III.

A question we left open in this work is whether there are theoretical bounds on the performance of the proposed (two move and one stop) strategy. We are also working on extending our strategies to two dimensional environments.

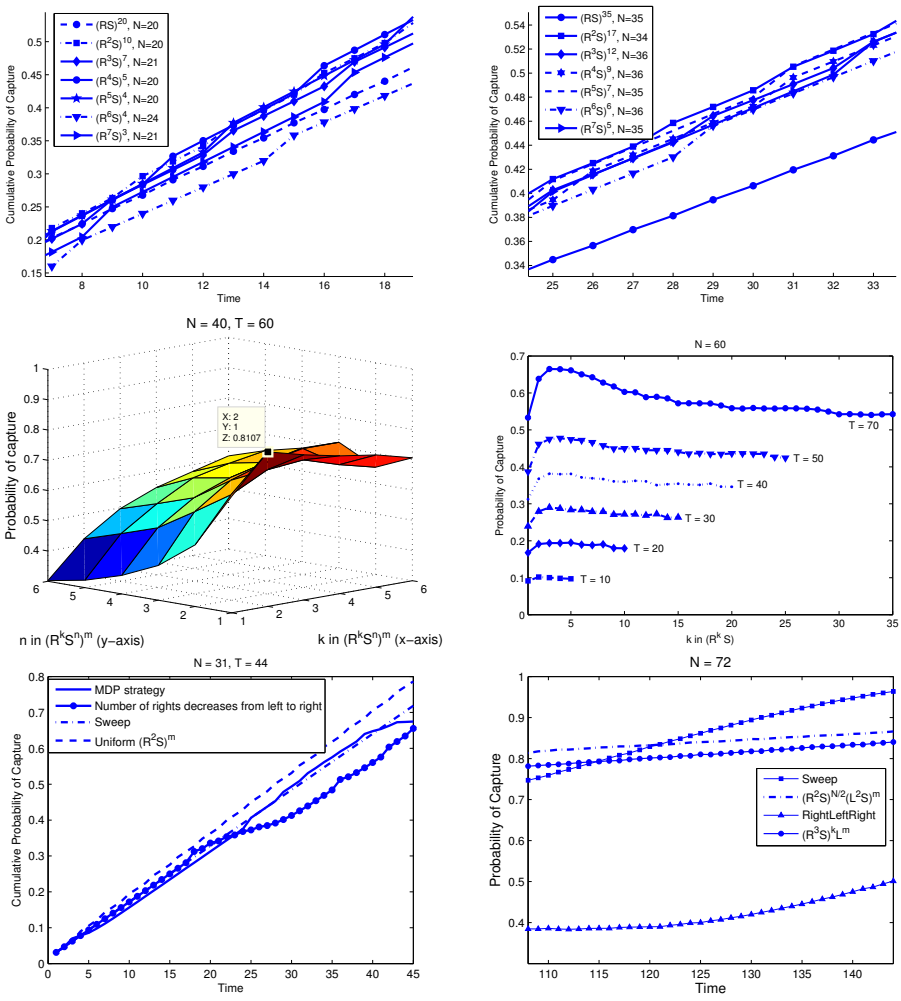

Fig. 4. **Top)** Comparison of uniform strategies $(R^kS)^m$. Here $N = km$. Left) N around 20. Right) N around 35. **Middle)** Left) Uniform strategies with more than one stop. $(R^2S)^m$ is superior to the others. Right) Performance of $(R^kS)^m$ for $N = 60$ and different values of T compared in simulation. **Bottom)** Left) Comparison of $(R^2S)^m$ with non-uniform strategies. Right) Comparison of four strategies when $1.5N < T$ for $N = 72$.

	Best Strategy
$2N < T$	Sweep
$1.7N < T \leq 2N$	Sweep
$1.5N < T \leq 1.7N$	$(R^2S)^{N/2}(L^2S)^m$
$T \leq 1.5N$	$(R^2S)^m$

TABLE III

SUMMARY OF THE PROPOSED BEST STRATEGIES FOR DIFFERENT CASES.

REFERENCES

[1] T. Chung, G. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robots*, vol. 31, no. 4, pp. 299–316, 2011.

[2] P. Tokekar, D. Bhadauria, A. Studenski, and V. Isler, "A robotic system for monitoring carp in minnesota lakes," *Journal of Field Robotics*, vol. 27, no. 6, pp. 779–789, 2010.

[3] S. Thrun, W. Burgard, D. Fox *et al.*, *Probabilistic robotics*. MIT press Cambridge, MA, 2005, vol. 1.

[4] A. Renzaglia, N. Noori, and V. Isler, "Searching for a one-dimensional random walker: Randomized strategy with energy budget," in *IEEE Conference on Intelligent Robots and Systems*, 2013.

[5] L. Lovász, "Random walks on graphs: A survey," *Combinatorics, Paul erdos is eighty*, vol. 2, no. 1, pp. 1–46, 1993.

[6] S. Redner, *A Guide to First-Passage Processes*, 1st ed. Cambridge: University Press, 2001.

[7] P. Krapivsky and S. Redner, "Kinetics of a diffusive capture process: lamb besieged by a pride of lions," *Journal of Physics A: Mathematical and General*, vol. 29, no. 17, p. 5347, 1999.

[8] A. Gabel, S. Majumdar, N. Panduranga, and S. Redner, "Can a lamb reach a haven before being eaten by diffusing lions?" *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2012, no. 05, p. P05011, 2012.

[9] S. Kopparty and C. V. Ravishankar, "A framework for pursuit evasion games in rn," *Information Processing Letters*, vol. 96, no. 3, pp. 114–122, 2005.

[10] N. Noori, P. Plonski, A. Renzaglia, P. Tokekar, J. Vander Hook, and V. Isler, "Long-term search through energy efficiency and harvesting," Department of Computer Science & Engineering, University of Minnesota, Tech. Rep. 13-001, 2013. [Online]. Available: http://www.cs.umn.edu/research/technical_reports/view/13-001

[11] N. Noori, A. Renzaglia, and V. Isler, "Searching for a one-dimensional random walker: Deterministic strategies with a time budget when crossing is allowed," Department of Computer Science & Engineering, University of Minnesota, Tech. Rep. 13-009, 2013. [Online]. Available: http://www.cs.umn.edu/research/technical_reports/view/13-009

[12] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning Series, 1998.

[13] "Mdp matlab toolbox," http://www.inra.fr/mia/T/MDPtoolbox/, accessed November, 2012.