International Journal of Computer Applications (0975 - 8887)
Volume 143 - No.11, June 2016

A Target Searching Scenario in Undirected Acyclic Graph

A. B. Sagar
National Informatics Center,
Alirajpur Dt, M.P.,
India

Avinash Pathak
National Informatics Center,
Tikamgarh Dt, M.P.,
India

ABSTRACT

In a generic problem in search theory we have some metric search space and two players - a target T and a searcher S. Mostly T is static in the space according to some specified probability distribution or in some cases dynamic, and S starts its search at some arbitrary start point. The usual goal will be to design a strategy that minimizes the expected time for S to find T. S knows the search space but has no other information and T could be a fully or partially informed target. In some versions, S has visibility characteristics allowing it to see a small distance 'δ' from its location. This kind of exploration problems relate to various contexts, such as robot motion planning in hazardous or inaccessible terrain, maintaining security of large networks, and searching, indexing, and analyzing digital data in the Internet [1] [2] [3]. One of the earliest examples of such problems is the linear search problem, proposed by Bellman [4] and Beck [5]. Here, the search space is an infinite line, with the searcher initially at an arbitrary origin, and the target located at an unknown point on the line, at distance d from the origin. The objective is to minimize the worst-case ratio of the distance traveled by the searcher over d. Different scenario (with cycles) is considered in cops and robbers problem [6] where the cops and robbers take alternate turns in movements and the question usually posed is how many pursuers are necessary to ensure the eventual capture of all the robbers. The Isaac's Princess and Monster problem [7] is also a related problem but with different scenario. The Lost Cow problem [8] is stated as a short-sighted cow following along an infinite fence and wants to find the gate. The lost cow problem is limited in the idea that the target is static. In case of one dimensional space, if a moving target wants to escape, then it can do so by constantly moving away from the searcher. And in case of a cyclic graph, a fully informed target can move around in cycles and never be found by the searcher. The present study stands out in that it studies the problem in case of undirected acyclic graph which was not studied before. But the proposed search space offers an interesting environment to both target and the searcher. To the target, it provides an opportunity to move towards the searcher without being found; and to the searcher it provides an easier search space by removing the case of cyclic movement of the target. Sleator and Tarjan [9] suggested evaluating the performance of an online algorithm using competitive analysis. This paper proposes an online algorithm for the provided search space and also attempts to find upper and lower bounds of the competitive ratio.

General Terms

Searching graph

Keywords

Target Searching Problem, Online Search, Uninformed Search

1. BACKGROUND

Let s and t be two points in a graph G. Consider a searcher at s that is searching for the point t in G. If the robot has the complete information of G and also knows the exact location of t, then the searcher can choose a path inside G to move from s to t. The choice of a path depends on optimization criteria for the given problem. In many situations, it is expected that the searcher follows the shortest path from s to t inside G. In some other situations, the searcher may be asked to follow a path from s to t inside G which has the minimum number of edges. There are known efficient sequential algorithms for computing these types of paths from a starting point s to the target point t inside a known region G [10]. Thus, the searcher can compute an optimal path, depending upon the optimization criteria, and then follow the path from s to t. Such path planning algorithms are called $offline\ \ algorithms$ for target searching problems in a known environment [11]. Consider a natural scenario when the searcher does not have a complete knowledge of the search space a priori, and also does not know the location of the target t, but can recognize the target. In such a situation, the searcher is asked to reach t from an arbitrary starting position s. The problem here is to design an efficient algorithm which a searcher can use to search for the target t. This is called an online problem for target searching in an unknown environment, and the algorithms for such online problems are known as $online\ algorithms$. Observe that any such algorithm is $online$ in the sense that decisions must be made instantaneously. If a searcher is asked to explore or see all points of an unknown environment rather than searching for a particular target t, the problem is known as an online exploration problem, and algorithms for such problems are known as $online\ exploration\ algorithms$. One of the difficulties in working with incomplete information or with only local information is that the path cannot be pre-planned and therefore, its global optimality can hardly be achieved. Instead, one can judge an online algorithm based on its performance with respect to other existing or theoretically feasible algorithms, or how 'reasonable' they are with respect to 'human watchmen'. As is the case with other online problems [25] [9], the efficiency of online algorithms for searching and exploration algorithms is generally measured using their competitive ratios [1].

2. INTRODUCTION & LITERATURE REVIEW

The problem of a searcher traversing to find a target is an important computational problem with several applications in various contexts. This class of problems usually have a searcher that must locate a target lying at some unknown location in the environment. A general objective is to devise efficient strategies that allow the searcher to locate the target as quickly as possible. One of the earliest examples of such problems is the linear search problem, proposed by Bellman [4] and Beck [5]. Here, the search space is an infinite line, with the searcher initially at an arbitrary origin, and the target located at an unknown point on the line, at distance d from the origin. The objective is to minimize the worst-case ratio of the distance traveled by the searcher over d. Exploration problems appear in various contexts, such as robot motion planning in hazardous or inaccessible terrain, maintaining security of large networks, and searching, indexing, and analyzing digital data in the internet [1] [2] [3]. This problem is also closely relates to search games [13] [14]. A good example is looking for a lost key on a road when we do not know in which direction the key is and at what distance. This linear-search problem is also called the star search or ray search problem, which is known informally as the "m-lane cow-path problem". In this setting, there is a set of m semi-infinite rays with a common origin O, and a searcher (cow) initially at an origin O. The target (pasture) is located at distance d from O, however the searcher is oblivious of the ray on which the target lies. A strategy is an algorithm that specifies how the searcher traverses the rays and aims to minimize the worst-case distance traveled, again normalized by the optimal distance d. This simple problem has important applications to robot navigation, artificial intelligence, and operations research [15],[16],[17],[18],[19],[20],[21],[22]. This is because it can be applied to cases that need efficient allocation of resources for multiple tasks. A different application is the design of efficient interruptible algorithms. This is an AI problem with surprising connections to the ray-search problem [22][23]. In this paper the authors look for a good search strategy by balancing theoretical quality with practical applicability.
Searching problems are central to almost all areas of computer science. Several variants of searching problems come up in the study of data structures, database applications, computational geometry, and artificial intelligence. Owing to the importance of searching problems, many variants of simple searching have been studied by several researchers, including searching in unknown environments [24] [25], and searching in the presence of errors [26] [27]. The present paper is not about shortest path but about the strategies and completeness. Linear line was considered in lost cow problem. Different scenario (with cycles) are considered in cops and robbers problem. But the kind of scenario which the authors presented here is not considered by any of the researchers before. Here there is no limitation on the movement of target. A target can move at any speed to any vertex. There is also a consideration of the scenario when a target may not be on the edge or vertex but in the space in-between. The cops and robbers problem [6] and the lost cow problem [8] have some similarity to the present problem and can provide some insight into some of the issues in the stated problem. In cops and robbers problem, the cops and robbers take alternate turns in movements. This feature/limitation is not there in the present problem. In cops and robbers problem, the question usually posed is how many pursuers are necessary to ensure the eventual capture of all the evaders. But in the present problem, the question is what strategy is certain of finding the target. The Isaac's Princess and Monster Problem [7] is different scenario. The lost cow problem is stated as a short-sighted cow following along an infinite fence and wants to find the gate. This makes a convenient one-dimensional planning problem. If the location of the gate is given, then the cow can reach it by traveling directly. The lost cow problem is limited in the idea that the target is static. Even if the target is dynamic it can only avoid the searcher if it is moving away from the origin. But in the present problem, it is possible to miss the searcher even when the target is moving towards the origin [see Fig: 4]. In the hunter-rabbit problem, it is easy to see that a single hunter can catch the rabbit simply by guessing its location in the next turn. However, if the hunter's strategy is deterministic, knowing it, the rabbit would never get caught. Similarly, the hunter could always catch the rabbit in a single move if he knew its strategy. But there are several differences between existing problems and the stated problem. Lost cow is linear problem with static target where as the stated problem is non-linear with dynamic target. The oil search problem is a progressing solution and there is no change in the depth of the oil which implies a static target. In the literature, the names pursuer-evader, cop-robber, monster-princess, hunter-rabbit, and sheriff-thief have been used somewhat synonymously. The authors adopt the searcher-target term for it emphasizes the discrete nature of the problem.

3. PROBLEM DEFINITION

The basic model of the search space for online graph exploration is as follows. The undirected connected graph is G = (V, E). The authors assume that the vertices are labeled so that the searcher is able to distinguish them. Each edge e = (u, v) $\in$ E has a non-negative real weight $|e|$, also called the length or the cost of the edge. To simplify the study the authors assume that the length or cost of each edge is uniform. For traversing an edge, the searcher has to pay the respective edge cost. The searcher travels with constant maximum speed of 1. The searcher knows the topology of the environment (graph) but does not know the location of the target. The problem of a searcher S trying to find a target T in an unknown search space G with no a priori knowledge about the location of the target, falls into the category of online algorithms. Though the term 'visibility' was used in many of the works reviewed, the authors would like to use the term 'detectability'. S has a 'detectability zone' which is the range within which it can identify the target. T also has a 'zone' within which it can be detected in the environment. Depending on the environment of the search space, these zones can vary in size - giving possibilities to several scenarios. For simplicity, the authors will consider that the zone size is constant. Targets and searchers are of two types – static and dynamic. Let T_S, T_D, S_S and S_D denote static target, dynamic target, static searcher and dynamic searcher respectively. Combinations such as static target & dynamic searchers, dynamic target & static searchers, and dynamic target & dynamic searchers, make interesting search problems. Also the search space G may or may not have paths. If G is pathless and target is static, then the best strategy for the searchers is to flood G or do a zig-zag movement ultimately locating target.
Let $find(S, T, G)$ be the polymorphic function defining the strategy of a searcher S for locating the target T in search space G. The various forms of the function are given below:

$find(S_d, T_s, G_{np})$ — S is dynamic, T is static, G has no paths

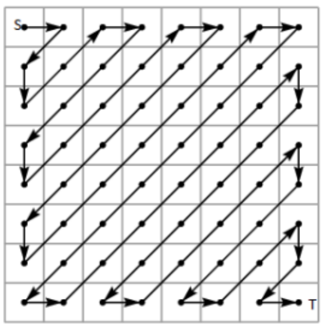

Fig. 1. Zig-Zag Search

$find(S_d, T_s, G)$ — S is dynamic, T is static, G has paths

$find(S_d, T_d, G_{np})$ — S is dynamic, T is dynamic, G has no paths

$find(S_d, T_d, G)$ — S is dynamic, T is dynamic, G has paths

$find(S_s, T_d, G_{np})$ — S is static, T is dynamic, G has no paths

$find(S_s, T_d, G)$ — S is static, T is dynamic, G has paths

If S was successful in locating T, then $find(S, T, G) = 1$, otherwise, 0. We know that $find(S_d, T_s, G_{np})$ will always be 1 because entire search space is explored. But when G has paths, a zig-zag search or flooding cannot not guarantee to find the target. Assuming that the searcher has complete or partial information of the location of the target will turn the search into a heuristic search. But the authors assume that the searcher has no information about the location of the target in the search space while the target may have full or partial information about the strategy of the searcher.
Cost/length of the edges is uniform. Movements of searcher or target are linear. They cannot jump over any edges or vertices, that is, movement can only happen through adjacent edges. If the target is constantly moving, then a simple strategy of the searcher is to get on the target's trail and follow it. But this only works for "slower and continuously moving" targets. But in the stated problem, the target may move with any speed, even with a speed greater than that of searcher, and hence the searcher cannot follow the trail. As the target may move back to the old edges to escape from the searcher, strategy of marking edges as visited/not-visited does not help. So, the problem at hand is to design a complete scheme to effectively find the target with minimum cost.
Sleator and Tarjan [9] suggested evaluating the performance of an online algorithm using competitive analysis. In a competitive analysis, an online algorithm A is compared to an optimal offline algorithm. An optimal offline algorithm knows the location of the target in advance and can find it with minimum traversal cost. Given a target location σ in our search space, if CA(σ) denotes the cost incurred by A and C$_{OPT}$(σ) denotes the cost incurred by an optimal offline algorithm OPT. The algorithm A can be called c-competitive if there exists a constant a such that

$$C_A(\sigma) \leq C_{OPT}(\sigma) + a$$

for all target locations σ. Here the authors assume that A is a deterministic online algorithm. The factor c is also called the competitive ratio of A.

4. TARGET SEARCHING STRATEGY

A strategy is an algorithm that specifies how a searcher traverses the search space in order to have the highest probability of locating the target with minimum moves and in shortest time. Let e_1, e_2, e_3, .. e_n be the edges of the graph G, and let c_1, c_3, c_3, etc. be the corresponding costs of searching the paths. If the total cost of searching the whole graph is C_G, and $i, j \in G$, then,

$$C_G = \sum_{i=1}^{n} 2c_i - \sum_{j \in G} c_j$$

Where j represent the edges which were not traversed twice.

If $P(T_s)$ represents the probability of success of $find(S_d, T_s, G)$ and $P(T_d)$ represents the probability of success of $find(S_d, T_d, G)$, then

$P(T_s)$ = 1 and $P(T_d) \leq 1$

$P(T_d) \leq P(T_s)$

Let X be a deterministic strategy for a graph on-line search problem. For the problems we consider here it usually suffices to model a searching strategy as a sequence of edges numbers, i.e., $X = (x_1, x_2, x_3, ...)$ with $x_k > 0$ and $0 \leq k < n$ where n is the total number of edges of graph G. In the beginning the position of the searcher is a point O on the graph; it has to find a target t that is located somewhere on its left or right edges. In a one dimensional space, if a searcher knows that a target is n steps away either to the left or to the right, it could use the following algorithm: go n steps to the left, and if the target was not found, turn back and go $2n$ steps to the right. It is not hard to see that this is the best it can do. However, the value of n is unknown to the searcher. Therefore, when going to the left (or right) the searcher has no way of judging when it is safe to assume that the target is located to the right (or left) from its search starting point. Obviously for the searcher, blindly betting on one of the directions is a bad idea in the worst case. So, the algorithm that solves this kind of problem explores both the paths alternatingly.

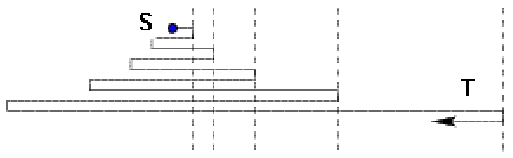

Fig. 2. Searching on a line

But if the search space is a two dimensional and the target is dynamic, as in fig(3), then an iterative deepening search on the alternate sides could not guarantee that the searcher will locate the target. This is because a dynamic target can find a path to escape from the searcher as shown in the fig(4). If the edges are numbered, and a searcher's traversal sequence is (0,1,2,3,4,5,6,7,8,9), then the target can use the following sequence (9,9,9,9,9,7,8,6,5,5) and escape from the searcher successfully. In the above scenario, had the search pattern of the searcher has been (0,2,4,3,6,8,7,5,9), then the target would not have escaped from the searcher. So, it seems that randomization of the search sequence of the searcher has overcome this problem. But this may not necessarily work all the time and

Algorithm 1 Alternate edges search for a static target

```
1: $Edges\_Searched\ ES[\,] \leftarrow \{\}$
2: $Edges\_Remaining\ ER[\,] \leftarrow \{e_1, e_2, ...e_n\}$
3: $Next\_Edge\ NE \leftarrow \{\}$
4: procedure SEARCH ($Graph\ G$)
5:     while ER Not empty do
6:         $remove(next\ edge\ e\ from\ ER[\,])$
7:         $add(edge\ e\ to\ NE)$
8:         $search(e)$
9:         if target is found then
10:            Return(find() = 1)
11:        end if
12:        $add\ e\ to\ ES[\,]$
13:    end while
14:    Return(find() = 0)
15: end procedure
```

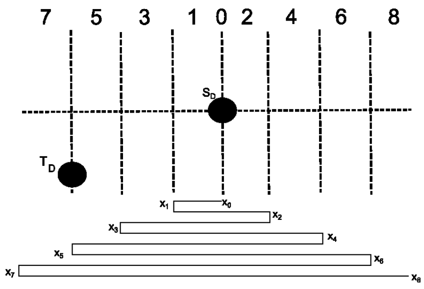

Fig. 3. Searching in the graph

is subject to the traversal sequence of the target. The target may, by chance, have a sequence that helps it bypass the searcher. And in the case of a fully informed target, randomization cannot really help because there will always be a sequence for the target to make up a counter sequence for escape.

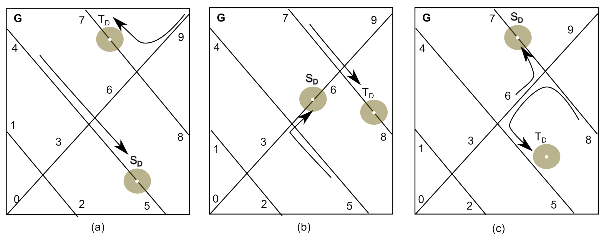

Fig. 4. Searcher missing a target

Because of this possibility, the authors categorize targets into three types.

Oblivious target: The oblivious target is not an informed target. It has no idea about the search sequence of the searcher. So the target generates its complete traversal sequence in advance, before the searcher begins its search.

Adaptive online target: This target may not have full information about the total search strategy of the searcher. It observes the searcher's most recent move, guesses the next move, and then adaptively makes its own move. This target will plan its each move adaptively, i.e., without knowing the random choices made by the searcher on the present or any future move.

Adaptive offline target: This target also generates a request sequence adaptively. This target makes the next move based on the searcher's most recent moves, but serves them optimally at the end. As it can be understood, this target is so strong, that randomization does not work against it. So, this is charged with the optimum sequence for any of the searcher's search sequence.

The competitiveness of a searcher's randomized online algorithm is measured by its performance against any oblivious target. For deterministic algorithms, adaptive adversaries are not more powerful than the oblivious ones since the algorithm's moves can be foreseen. But for randomized algorithms, it is worth introducing the a-competitiveness against adaptive adversaries. The authors call the a-competitive algorithms as a-competitive against any oblivious adversary for contrast.

5. THE COMPETITIVE ANALYSIS

A standard technique to measure the quality of online algorithms is competitive analysis [12], which compares the outcome of an algorithm with an optimal offline solution.

Let A be a deterministic algorithm for the present search problem. For any target t at distance $dist(t)$ from the origin, searcher travels a fixed distance according to algorithm A, which we denote $cost(A,t)$, to find the target. The authors say that algorithm A has competitive ratio c if, for all target positions t,

$$cost(A,t) \leq c\,.\,dist(t) + a$$

where c and a are constants that are independent of the target position t.

Let d be the distance of the target from the origin. The number of steps taken during the alternate search will be as below:

Target – Steps
$1 \rightarrow 2 + d$
$2 \rightarrow 2 + 6 + d$
$3 \rightarrow 2 + 6 + 10 + d$
$4 \rightarrow 2 + 6 + 10 + 14 + d$
$5 \rightarrow 2 + 6 + 10 + 14 + 18 + d$

i.e.2(1+3+5+7+9....)

Let us assume that target is located on the right side of the proposed graph from the origin. Assume that $2^{k-1} < d \leq 2^{k+1}$ for some k.
Then the total distance traveled during the alternative search is $(2.1 + 2.|-2| + 2.4 + 2.|-8| + ... + 2.2^{k-1} + 2.|-2^k| + d = 2.2^{k+1} + d)$
If the location of t is known a priori, then it is a straight walk of length d from the origin to t.
So, the competitive ratio of the alternate search in the proposed undirected acyclic graph is

$$\begin{aligned}\frac{(2.2^{k+1}+d)}{d} &= 1 + \frac{2.2^{k+1}}{d} \\ &= 1 + \frac{2.2^{k+1}}{2^{k-1}} \\ &= 9\end{aligned}$$

If algorithm R is a randomized algorithm, then the distance traveled by the searcher to find a particular target is no longer fixed.

Instead, $cost(R, g)$ is a random variable, and the authors define the competitive ratio by the expected value of this random variable. In other words, algorithm R has competitive ratio c if, for all target positions g,

$$E[cost(R, g)] \leq c \cdot dist(g) + d$$

where c and d are constants as before. In particular, if an algorithm for the present search problem has competitive ratio c, then for any target that is distance n from the origin, the expected distance that the searcher has to travel in order to find the target is at most cn plus some small constant.
Let the randomized algorithm has a geometric ratio $r > 1$, where r is a constant that is fixed for the duration of the algorithm to make sure that it will not include the searched edges repeatedly in its search sequence. If the target is dynamic, then it has to include the searched edges repeatedly, so the authors assume that $r \geq 1$. For ease of reference, assume that the n edges are labeled with integers $0, 1, ..., n - 1$.

Algorithm 2 Random edge search algorithm

```
 1: $\sigma \leftarrow A\, random\, sequence\, of\, \{0, 1, 2, 3, ...n - 1\}$
 2: $\in \leftarrow A\, random\, real\, value\, from\, [0, 1]$
 3: $d \leftarrow r^{\in}$
 4: $p \leftarrow 0$
 5: procedure RandomSearch($Graph\ G$)
 6:     Repeat
 7:     Explore path $\sigma(p)$ upto distance $d$
 8:     If target not found then return to origin
 9:     $d \leftarrow d.r$
10:     $p \leftarrow (p + 1) mod\ n$
11:     Until goal found
12: end procedure
```

However, the use of randomization is very limited; randomization is needed only at the very beginning of the search, in order to pick a random sequence and a random "initial search distance". The algorithm never needs access to a random number generator once the search has begun. From the literature, it can be derived that the competitive ratio of this random edge search algorithm is

$$R(r, n) = 1 + \frac{2}{n} \cdot \frac{1+r+r^2+..+r^{n-1}}{ln\ (r)}$$

The authors assume that adding an extra searcher is imposes high cost. But if it was considered that the cost issue is to be ignored then, a complete solution for finding the target can be easily achieved with a minimum cost and minimal time.
Against an adaptive adversary, as the power of randomization is severely limited, the complete solution is to adopt multiple searchers. So, m different searchers can be employed to search for the target. However, if they are all searching randomly, the authors do not know which of the searchers will terminate successfully on a given target. Also there is a possibility that none of them may succeed in finding the target. It is because there is still a little possibility for a target to escape as shown in fig 6.
So let us discuss the best exploration strategies for the searchers. It would not be the best strategy for the searchers to start off at one end and scan through to the other end because the target may be located on the farther end, resulting in the worst case.
Starting in the middle and searching alternatively in sequence also does not seem to be the best strategy because while the searchers move to the other side, the target may move to the already explored edges.

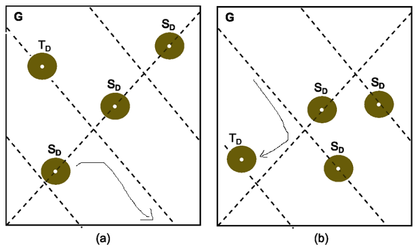

Fig. 5. Multiple searchers

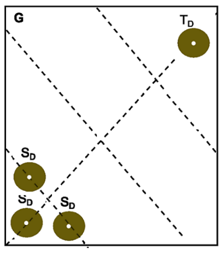

Fig. 6. Multiple searchers

6. REFERENCES

[1] P. Berman. On-line searching and navigation. In Online Algorithms: The State of the Art, volume 1442 of Lecture Notes in Computer Science, pages 232-241. Springer, 1998.

[2] L. Gasieniec and T. Radzik. Memory efcient anonymous graph exploration. In Proceedings of WG, volume 5344 of LNCS, pages 14-29, 2008.

[3] N. Rao, S. Kareti, W. Shi, and S. Iyengar. Robot navigation in unknown terrains: Introductory survey of nonheuristic algorithms. Report ORNL/TM-12410, Oak Ridge Nat. Lab., 1993.

[4] R. Bellman. An optimal search problem. SIAM Review, 5:274, 1963.

[5] A. Beck. On the linear search problem. Naval Research Logistics, 2:221-228, 1964.

[6] M. Aigner and M. Fromme. A game of cops and robbers. Discrete Applied Math, 8:1-12, 1984.

[7] R. Isaacs, Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization, John Wiley & Sons, New York (1965), PP 349-350.

[8] LaValle, Steven M. Planning algorithms. Cambridge university press, 2006, pg 555.

[9] Sleator, Daniel D., and Robert E. Tarjan. "Amortized efficiency of list update and paging rules." Communications of the ACM 28.2 (1985): 202-208.

[10] S.K. Ghosh, Visibility Algorithms in the Plane, Cambridge University Press, Cambridge, United Kingdom, 2007.

[11] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Boston, MA, 1991.

[12] A. Borodin and R. El-Yaniv. Online Computation and Competitive Analysis. Cambridge University Press, 1998.

[13] Gal, S.: Search Games. Academic Press (1980)

[14] Alpern, S., Gal, S.: The Theory of Search Games and Rendezvous. Kluwer Academic Publishers (2003)

[15] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. Information and Cmputation, 106:234-244, 1993.

[16] E.D. Demaine, S.P. Fekete, and S. Gal. Online searching with turn cost. Theoretical Computer Science, 361:342-355, 2006.

[17] P. Jaillet and M. Stafford. Online searching. Opes. Res., 49:234-244, 1993.

[18] M-Y. Kao and M.L. Littman. Algorithms for informed cows. In Proceedings of the AAAI 1997 Workshop on Online Search, 1997

[19] M-Y. Kao, Y. Ma, M. Sipser, and Y.L. Yin. Optimal constructions of hybrid algorithms. Journal of Algorithms, 29(1):142-164, 1998.

[20] A. Lopez-Ortiz and S. Schuierer. The ultimate strategy to search on m rays. Theoretical Computer Science, 261(2):267-295, 2001.

[21] S. Schuierer. Lower bounds in online geometric searching. Computational Geometry: Theory and Applications, 18(1):37-53, 2001.

[22] D.S. Bernstein, L. Finkelstein, and S. Zilberstein. Contract algorithms and robots on rays: unifying two scheduling problems. In Proceedings of the 18^{th} International Joint Conference on Artificial Intelligence (IJCAI), pages 1211-1217, 2003.

[23] Angelopoulos, Spyros, Alejandro Lopez-Ortiz, and Konstantinos Panagiotou. “Multi-target ray searching problems.” Theoretical Computer Science (2014).

[24] Baeza-Yates, R. A., Culberson, J. C., and Rawlins, G. J. E. (1993), Searching in the plane, Inform. and Comput. 16, 234-252.

[25] Fiat, A., Foster, D. P., Karloff, H., Rabani, Y., Ravid, Y., and Vishwanathan, S. (1991), Competitive algorithms for layered graph traversal, in “Proceedings, 32nd IEEE Symposium on Foundations of Computer Science,” pp. 288-297.

[26] Aslam, J. A., and Dhagat, A. (1991), Searching in the presence of linearly bounded errors, in “Proceedings, 23rd ACM Symposium on Theory of Computing,” pp. 486-493.

[27] Rivest, R. L., Meyer, A. R., Kleitman, D. J., Winklmann, K., and Spencer, J. (1980), Coping with errors in binary search procedures, J. Comput. System Sci. 20, 396-404.