Multi-target Ray Searching Problems

Spyros Angelopoulos[1], Alejandro López-Ortiz[2], and Konstantinos Panagiotou[3]

[1] CNRS-LIP6, Pierre and Marie Curie University, Paris, France.
[2] David R. Cheriton School of Computer Science, University of Waterloo, Canada.
[3] Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Abstract. We consider the problem of exploring m concurrent rays using a single searcher. The rays are disjoint with the exception of a single common point, and in each ray a potential target may be located. The objective is to design efficient search strategies for locating t targets (with $t \leq m$). This setting generalizes the extensively studied *ray search* (or *star search*) problem, in which the searcher seeks a single target. In addition, it is motivated by applications such as the interleaved execution of heuristic algorithms, when it is required that a certain number of heuristics have to successfully terminate.
We apply two different measures for evaluating the efficiency of the search strategy. The first measure is the standard metric in the context of ray-search problems, and compares the total search cost to the cost of an optimal algorithm that has full information on the targets. We present a strategy that achieves optimal competitive ratio under this metric. The second measure is based on a weakening of the optimal cost as proposed by Kirkpatrick [ESA 2009] and McGregor *et al.* [ESA 2009]. For this model, we present an asymptotically optimal strategy which is within a multiplicative factor of $\Theta(\log(m-t))$ from the optimal search cost. Interestingly, our strategy incorporates three fundamental search paradigms, namely uniform search, doubling and hyperbolic dovetailing. Moreover, for both measures, our results demonstrate that the problem of locating t targets in m rays is essentially as difficult as the problem of locating a single target in $m-(t-1)$ rays.

1 Introduction

Searching for a target is a common task in everyday life, and, unsurprisingly, an important computational problem with numerous applications in various contexts. This class of problems involves a *searcher* that must locate a *target* which lies at some unknown point in the environment. The natural 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 independently by Beck [2]. Here, the environment consists of an infinite line, with the searcher initially at some point designated as the 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.

A natural generalization of the linear-search problem is the *star search* or *ray search* problem, which is also known, informally, as the *m-lane cow-path problem*. In this setting, we are given a set of m semi-infinite rays (lanes), all with a common origin O, and a searcher (cow) initially placed at the 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 the objective is to minimize the worst-case distance traveled, again normalized by the optimal distance d.

This deceivably simple problem has important applications to robot navigation, artificial intelligence, and operations research (see e.g. [5, 12, 16, 19, 1, 10, 6, 11] for some illustrative examples). This is due to the fact that it can be applied in settings in which we seek efficient allocation of resources to multiple tasks. For instance, consider the setting in which m different randomized heuristics (of the *Las Vegas* type) can be employed to solve a problem. However, we do not know in advance which of the heuristics will terminate successfully on a given input. How should we distribute the processing time to the different heuristics (assuming that we can interleave the execution of the heuristics)? This is an example of a setting that is often encountered in the construction of (deterministic or randomized) *hybrid* algorithms [12]. A different application is the design of efficient *interruptible* algorithms, namely algorithms that can return meaningful solutions even if interrupted during their execution. This is a fundamental problem in artificial intelligence, with surprising connections to the ray-search problem [5].

To our knowledge, with the exception of [18], all previous work on ray-search (and related) problems has focused on the case in which the searcher must locate *a single* target. However, a natural generalization of the problem involves the setting in which multiple targets may exist, and the searcher's objective is to locate t different targets. For instance, consider the setting in which a hybrid algorithm can execute m different heuristics, as described earlier. It may be the case that we require that not just a single, but rather several heuristic algorithms successfully terminate and return a solution. This is often desired in situations in which we do not have strong guarantees on the quality of the solution that each heuristic returns. A typical example is SAT-solvers that invoke such hybrid algorithms (also known as *algorithm portfolios* [9]): here, we do not know in advance which heuristic is the most appropriate for any given input. The objective of this paper is thus to initiate the study of ray-search problems in the setting where the strategy must guarantee that a certain number of targets are located as efficiently as possible. We also expect that the generalization to multiple targets will prove an interesting topic of study in other contexts of search and exploration problems, which so far have focused almost exclusively on the case of a single target.

Models and performance measures Throughout the paper we consider the setting in which up to m potential targets are placed in the m rays, with at most one target per ray. More specifically, we will denote by $\lambda_1 \geq \ldots \geq \lambda_m$ the distances of the targets, in non-increasing order. We allow the possibility that $\lambda_i = \infty$ for some values $i \in [1, m]$. In other words, there may be rays with no target located on them. We will denote by $\Lambda = \{\lambda_1, \dots, \lambda_m\}$ the multiset of all target distances, and we make the standard assumption $\lambda_m \geq 1$. Note that without this assumption, i.e., if the distances can become arbitrarily small, no search algorithm can be competitive (see, e.g., [6]). We seek efficient strategies for locating $t \leq m$ targets, where we assume that t is provided as input to the algorithm, and is thus known. We emphasize that the searcher can move from ray to ray only by passing over the origin, and that the search terminates when the t-th target is reached.

In order to evaluate the performance of a strategy, one needs to compare the *search cost* incurred by the algorithm (that has no information on the set Λ) to the cost of an *ideal* algorithm that has a certain amount of information concerning the targets. The worst-case ratio of these costs gives rise to the so-called *competitive ratio*, since search problems are often considered online problems in the literature of search and exploration.

We distinguish between two concrete models. First, we consider the classical model in which the ideal algorithm has *complete information* on the placement of targets, that is, the algorithm knows not only Λ, but also the specific ray on which the target at distance λ_i can be found. In this model, the cost of the optimal algorithm is easy to evaluate, and equals $2\sum_{i=m-t+2}^{m} \lambda_i + \lambda_{m-t+1}$. In other words, the optimal strategy locates the t targets closest to the origin (the factor of 2 is due to the searcher returning back to the origin). On the other hand, more recently, a different approach in defining a less powerful (and in a sense, more realistic) ideal algorithm was proposed independently by Kirkpatrick [14] and McGregor, Onak and Panigrahy [18]. In their setting, the ideal algorithm has only *partial information* about the locations of the targets. More specifically, we allow the ideal algorithm knowledge of the set Λ, but not of the exact mapping of distances λ_i to the rays. The implication is that the ideal algorithm itself will be associated with an *intrinsic* search cost, which is the worst-case cost of a search strategy that has knowledge of Λ. Following the notation of Kirkpatrick, we will denote by $\xi_t(\Lambda)$ the intrinsic (i.e., "optimal") cost for locating t targets. We shall omit the subscript "t", whenever it is clear from the context.

Contribution of this paper In this work we provide optimal strategies for locating $t \leq m$ targets in the m-ray setting. In Section 2 we focus on the complete information model for the ideal algorithm. We show that the worst-case competitive ratio for locating t targets is the same as the worst-case competitive ratio for locating a single target in $m - (t-1)$ rays, and we obtain a tight analytical expression for the optimal competitive ratio (c.f. Theorem 1).

Section 3 addresses the problem under the partial information model, and contains the main results of this work. First, we provide an analytic expression of the intrinsic cost, given the set of target distances Λ (c.f. Theorem 2). Next, we present a strategy based on combination of doubling and hyperbolic search that yields a $\Theta(\log m)$-competitive algorithm (c.f. Theorem 3). Last, we use the previous strategy as a subroutine so as to obtain an optimal algorithm of competitive ratio $\Theta(\log(m-t))$ (Theorem 4 and Theorem 5). Interestingly, this optimal strategy incorporates three fundamental search paradigms, namely uniform search, doubling and hyperbolic dovetailing. Similar to our results concerning the full-information model, we can interpret this result as a reduction between the problems of locating a single and multiple targets. Namely, the optimal competitive ratio for finding t targets in m rays is determined by the optimal strategy for locating *one* target in $m-(t-1)$ rays. Observe that, as discussed above, while the competitive ratio is the same the strategy itself is quite different from the $m-(t-1)$ ray case.

Related work Ray-search problems have a long and exciting history of research. We review some representative results, with the observation that the vast majority apply to the complete information model for the ideal algorithm. For the linear search problem, Beck and Newman [3] first showed an optimal competitive ratio of 9. The generalization to the m-ray ray-search problem was first studied by Gal [8] and later by Baeza-Yates *et al.* [1]. Both works proposed a round-robin strategy of exponentially increasing lengths that achieves optimal competitive ratio (see also the discussion of Jaillet and Stafford [10]). The above results are obtained by means of deterministic strategies; however, it is known that randomization can help improve the competitive ratio. In particular, Kao *et al.* [13] gave an optimal randomized algorithm for linear search, a result that was extended by Kao *et al.* [12] to the m-ray problem (under the restrictive assumption of round robin strategies). Other variants include the setting in which the searchers incur some turn cost when they switch direction (studied by Demaine *et al.* [6]), the case of multiple searchers (López-Ortiz and Schuierer [17]) and the average-case analysis of linear search (due to Kao and Littman [11]). Typically, round-robin strategies based on iterative deepening yield optimal or near-optimal algorithms, and similar ideas lead to efficient search algorithms in more general settings and environments (see the results of Koutsoupias *et al.* [15] and Fleischer *et al.* [7]).

In contrast, the study of the partial information model is much more recent. Kirkpatrick [14] addressed both deterministic and randomized algorithms under this framework. For both cases he presented optimal strategies based on a searching technique named *hyperbolic dovetailing*, since in each round a ray is searched to distance inversely proportional to its rank. The (optimal) competitive ratio of both deterministic and randomized strategies based on hyperbolic search is shown to be $\Theta(\log m)$. Independently, McGregor *et al.* [18] studied the setting in which there is a target in each ray, and the objective is to locate as many as possible at a cost close to the intrinsic cost. Their results provide randomized algorithms for locating $k-\tilde{O}(k^{5/6})$ targets at a cost no more than $(1+o(1))$ times the intrinsic cost for locating k of them.

2 Ray search in the full-information model

For the case of a single target and m concurrent rays, it is known that optimal strategies can be found in the class of the so-called *geometric* or *exponential* strategies (see, e.g., [8]). In this class of strategies, the searcher performs a round-robin exploration of rays with distances forming a geometric sequence (i.e., of the form $b^0, b^1, b^2, \ldots$, for some appropriate choice of the base $b > 1$). We show that similar geometric strategies lead to optimal multi-target search algorithms. Proofs are omitted for space reasons.

Lemma 1. *There is a geometric strategy for searching t targets in m rays with competitive ratio at most $1 + 2\frac{b^{m-(t-1)}}{b-1}$, where b is the base of the strategy.*

Theorem 1. *The competitive ratio of the geometric strategy of Lemma 1 is minimized for $b = \frac{m-(t-1)}{m-t}$, for which it is equal to $1 + 2\frac{(m-(t-1))^{m-(t-1)}}{(m-t)^{m-t}}$. Moreover, this is optimal, i.e., there is no algorithm with a smaller competitive ratio.*

We emphasize that the competitive ratio of Theorem 1 is the same as the competitive ratio of searching a single target in $m - t + 1$ rays [8].

3 Ray-search in the partial-information model

In this section we study deterministic algorithms for ray-search in the partial-information model for the ideal algorithm (as discussed in the introduction). Recall that the m rays are associated with a (multi)set Λ of target distances $\Lambda = \{\lambda_1, \ldots \lambda_m\}$ (with $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_m$), and the objective is to locate t targets. In order to facilitate the exposition of our results, we focus on a slightly different cost formulation; namely, we assume that the searcher incurs cost only the first time it traverses a previously unexplored segment of a certain ray (for instance, we do not charge the searcher for returning to the origin). In other words, the total search cost is the sum of the maximum distances traversed on each ray. For the sake of completeness, we note that, our results can be extended to the m-ray search problem under the "standard" cost formulation.

As mentioned in Section 1, we assume that the multiset Λ is not known to the (online) search strategy, but is known, in contrast, to the ideal algorithm. A *presentation* of Λ is a specific assignment of distances in Λ to target locations in the rays, which is unknown both to the online strategy *and* to the ideal algorithm. Given Λ, we denote by $\xi_t(\Lambda)$ the *intrinsic cost* of the ideal algorithm for locating at least t targets, namely the minimum worst-case search cost of a strategy that knows Λ. The special case of $t = 1$ was treated in [14, 18].

3.1 Intrinsic cost of multi-target search

We begin by evaluating the intrinsic cost in the case where we search for $t \geq 1$ targets.

Theorem 2. *The intrinsic cost for locating t targets in a presentation with associated distance set Λ is*

$$\xi_t(\Lambda) = \min_{1 \leq i_1 < \ldots < i_t \leq m} \sum_{j=1}^{t} i_j \cdot \mu_{i_j}, \tag{1}$$

where $\mu_{i_j} = \lambda_{i_j} - \lambda_{i_{j+1}}$, for $j < t$, and $\mu_{i_t} = \lambda_{i_t}$.

Proof. First, we will lower-bound the intrinsic cost of any search strategy A which succeeds for all possible presentations of Λ. At each point in time, the strategy has explored each ray to some distance: in particular, suppose without loss of generality, that A has searched rays $1, \ldots, m$ to distances $d_1, \ldots d_m$, in non-increasing order, i.e., $d_1 \geq d_2 \ldots \geq d_m$. We then can claim the following property concerning A and the set of distances $\{d_i : i \in [1, m]\}$: there must exist indices $1 \leq i_1 < \ldots < i_t \leq m$, such that $d_{i_j} \geq \lambda_{i_j}$ $(1 \leq j \leq t)$, otherwise strategy A will have not located t targets for at least one presentation of Λ. It follows then that the overall search cost of strategy A has to be at least $i_1\lambda_{i_1} + (i_2 - i_1)\lambda_{i_2} + \ldots + (i_t - i_{t-1})\lambda_{i_t}$. Note that this is equal to $\sum_{j=1}^{t} i_j \cdot \mu_{i_j}$, where the μ_i's are as in the statement of the lemma.

On the other hand, we can upper-bound the intrinsic cost by considering the following strategy that works in t phases. Fix indices $1 \leq i_1 < i_2 \ldots < i_t \leq m$. In phase t, the strategy searches rays $1, \ldots i_t$ up to depth $\lambda_{i_t} = \mu_{i_t}$. Let N_t denote the set of rays on which no target was located in phase t. In phase $t-1$, the strategy will search all rays in N_t up to an additional length of $\lambda_{i_{t-1}} - \lambda_{i_t} = \mu_{i_{t-1}}$. More general, if N_j denotes the set of rays for which no target was located during phase j, then in phase $j-1$ the strategy will search all rays in N_j up to an additional distance of $\lambda_{i_{j-1}} - \lambda_{i_j} = \mu_{i_{j-1}}$. We terminate when t targets have been located (which may happen before we reach the end of phase 1). Note that this strategy will always locate at least one target per phase, since in phase j it searches i_j rays up to distances λ_{i_j}, hence its cost is upper bounded by $\sum_{j=1}^{t} i_j \cdot \mu_{i_j}$. □

Given two sets of target distances Λ and Λ', we say that Λ *dominates* Λ' (denoted by $\Lambda \succ \Lambda'$) if $\lambda_i \geq \lambda'_i$. The following is an immediate corollary of Theorem 2.

Corollary 1. *If $\Lambda \succ \Lambda'$, then $\xi_t(\Lambda) \geq \xi_t(\Lambda')$, for any t.*

3.2 A $O(\log m)$-competitive algorithm

In this section we present a search strategy for locating t targets that achieves competitive ratio $O(\log m)$. This strategy, which we call Adaptive Hyperbolic Search is based on a combination of hyperbolic search and doubling, and will be used as subroutine in the construction of an optimal algorithm in Section 3.3.

Before we present our algorithm, let us describe briefly the hyperbolic dovetailing algorithm in [14, 18] for locating a single target. The algorithm begins with assigning unique ranks to the rays, which are integers in $[1, m]$, and by initializing a counter c to the value 1. It then proceeds in iterations, where the ray with rank i is searched up to distance c/i. If no target was found this way, then c is increased by 1 at the beginning of the next iteration.

There are (at least) two natural ways one could attempt to extend this algorithm to the case where we are interested in finding $t > 1$ targets. On the one hand, we could simply choose to never change the rank of rays, even after a target is located on some ray. On the other hand, we could behave "aggressively", and update the ranks immediately after a target was located (according to some chosen rule). However, it turns out that both ways lead to extremely ineffective algorithms of competitive ratio $\Omega(t)$.

Our algorithm (see the pseudocode below) strikes a balance between the above two extremes. Initially, as in the classical hyperbolic search, it begins by assigning unique ranks to the rays, and by initializing a counter c to 1. However, the execution of our algorithm is divided into *epochs*, where each epoch in turn consists of two *phases* (the boolean variable $firstphase$ in the statement of the algorithm determines whether we are in the first phase or not). During the first phase of each given epoch, the algorithm searches, for all i, the ray of rank i (denoted by r_i in the pseudocode) to a distance of c/i, i.e., it performs a hyperbolic search according to rank. The phase terminates when a target is discovered, at which point the second phase begins; this phase proceeds until iteration $\overline{c} \leftarrow 2c$, and again consists of hyperbolic search according to rank (in what follows we call *iteration* j the execution of lines 3–33 when c has value j). Targets found during this phase do not affect the rank. However, at iteration $\overline{c}$ the ranks of the rays are updated (lines 27–33), by removing rays on which targets are found.

Algorithm 1: Adaptive hyperbolic search

```
1  $T \leftarrow 0$ , $c \leftarrow 1$, $\overline{c} \leftarrow 0$ , $firstphase \leftarrow$ true
2  for $i = 1$ to $m$ do
3      $r_i \leftarrow i$
4      $found_i \leftarrow$ false
5  end
6  repeat
7      while firstphase=true or (firstphase=false and $c < \overline{c}$) do
8          for $i = 1$ to $m$ do
9              if $found_i = false$ then
10                 search ray $r_i$ up to distance $\frac{c}{i}$
11                 if target found at ray $r_i$ then
12                     $found_i \leftarrow$ true
13                     $T \leftarrow T + 1$
14                     if $T = t$ then break
15                     if firstphase=true then
16                         $firstphase \leftarrow$ false
17                         $\overline{c} \leftarrow 2c$
18
19
20
21         end
22         $c \leftarrow c + 1$
23     end
24     $firstphase \leftarrow$ true , $count \leftarrow 1$
25     for $i = 1$ to $m$ do
26         if $found_i$= false then
27             $r_i \leftarrow count$
28             count $\leftarrow$ count+1
29
30     end
31 until $T = t$
```

Analysis Denote by R_i the set of rays that acquire rank equal to i during the execution of the algorithm. Note that if a ray is assigned rank i during some epoch, and rank $j \neq i$ in some subsequent epoch, it cannot be assigned rank i again in the future. This observation allows us to define the search cost on a ray l for the interval in which the rank of l is i, which we denote by $C^i(l)$. We also denote by $C(R_i) = \sum_{l \in R_i} C^i(l)$ the overall search cost for rays of rank i. With this notation, the cost of our algorithm can be written as

$$ALG = \sum_{i=1}^{m} C(R_i). \tag{2}$$

Moreover, we will use the notation c^* to denote the value of c when the algorithm terminates, i.e., the last iteration. The next lemma bounds the value of $C(R_i)$.

Lemma 2. *For any* $1 \leq i \leq m$, $C(R_i) \leq \frac{3c^*}{i}$.

Proof. Note that every time the algorithm performs a search on a ray $l \in R_i$, it contributes to the cost $C(R_i)$ in two possible ways: i) By searching the ray the *first* time after it is assigned rank i (in other words, when line 12 is executed immediately after l acquires rank equal to i), and ii) in all remaining cases, i.e., subsequent iterations in the same epoch in which line 12 is executed for l. Let $C_1^i(l)$ and $C_2^i(l)$ denote the above two contributions to $C^i(l)$. Clearly,

$$C(R_i) = \sum_{l \in R_i} (C_1^i(l) + C_2^i(l)). \tag{3}$$

We first bound the cost incurred by $C_2^i(l)$. Let e denote the total number of epochs in the execution of the algorithm, and let $c_1, \ldots, c_e$ denote the value of c at the end of the corresponding epoch. (In particular, we have $c_e = c^*$). Let $1 \leq j < e$ be any epoch, and let us denote by l the ray of rank i the jth epoch, and by l' the ray of rank i in the $(j+1)$th epoch. At the end of the jth epoch, l is searched to distance c_j/i. Moreover, note that the *second* time l' is searched in the $(j+1)$st epoch, it had been already searched down to distance $\frac{c_j}{i}$. Therefore, the accumulated cost for searching the rays of rank i over all epochs is bounded by $\frac{c^*}{i}$, i.e., $\sum_{l \in R_i} C_2^i(l) \leq \frac{c^*}{i}$.

It remains to bound the cost $C_1^i = \sum_{l \in R_i} C_1^i(l)$. To this end, let $l_1, l_2, \ldots, l_e$ denote the rays in R_i, where l_k had rank i in epoch k. Note that there is a unique ray that had rank i in a particular epoch. Moreover, ray l_k contributes a cost of at most $\frac{c_{k-1}}{i}$ to C_1^i (where we use the convention $c_0 = 0$). Note also that the definition of the algorithm implies that $c_{k+1} \geq 2c_k$, as the first phase in the epoch ends when c attains the value $2c_k$, due to line 19 in the statement of the algorithm. We conclude that $c_k \leq \frac{c^*}{2^{k-1}}$, and thus, $C_1^i = \sum_{k=1}^{j} C_1^i(l_k) \leq \sum_{k=1}^{j} \frac{1}{2^{k-1}} \frac{c^*}{i} \leq \frac{2c^*}{i}$. □

The next lemma relates the intrinsic complexity with the cost of the algorithm.

Lemma 3. *For any* $t \geq 2$, $\xi_t(\Lambda) \geq \frac{c^*}{5}$.

Proof. Set $c' = \lfloor c^*/3 \rfloor$. We can assume, without loss of generality, that $c^* \geq 5$. In order to prove the statement, let us first assume that the iterations c' and c^* occurred in the same epoch of the execution of the algorithm. Then no target is found during iterations up to c' which also belong in the epoch of c', as otherwise it would be that $c^* \leq 2c' < c^*$, a contradiction.

Observe that at the end of an epoch, i.e., at lines 28–33, each possible rank between 1 and $m - T$ is assigned to a unique ray (recall that T is the number of targets that were discovered up to the current iteration). Consequently, if in the first iteration of the succeeding epoch no additional target is found, then for each $1 \leq i \leq m - T$ there is a ray that has been searched up to distance c/i. In our case in particular, since at most $t - 1$ targets were found at the beginning of the last epoch (i.e., the epoch of c^*), for every j with $1 \leq j \leq m - (t - 1)$, there exists a ray that has been searched unsuccessfully up to distance c'/j. This implies that if we set

$$\Lambda' = \{\lambda'_1, \ldots, \lambda'_m\} = \left\{ c', \frac{c'}{2}, \frac{c'}{3}, \ldots, \frac{c'}{m-(t-1)}, 0, \ldots, 0 \right\},$$

then $\Lambda \succ \Lambda'$. Corollary 1 implies that $\xi_t(\Lambda) \geq \xi_t(\Lambda') = \xi_1\big(\{\lambda'_1, \ldots, \lambda'_{m-(t-1)}\}\big)$, where the equality follows easily from Theorem 2. We conclude the proof in this case by observing that $\xi_1 \left(\{\lambda'_1, \ldots, \lambda'_{m-(t-1)}\} \right) = \min_{1 \leq i \leq m-(t-1)} i\lambda'_i = c'$.

It remains to consider the case where iterations c' and c^* occurred in different epochs. We may further assume that in the epoch of iteration c', at least one target was discovered at some iteration smaller than or equal to c' which also belongs in the epoch of c', as otherwise the same argument as in the previous case would apply. Let $c'' \geq c'$ be the first iteration of the epoch that succeeds the epoch of iteration c'. We shall denote, in the remainder, this epoch as the *current* epoch. Note that $c'' \leq 2c' < c^*$. Suppose that $\ell \geq 0$ targets are discovered during iteration c'', and let us denote by $i_1 < i_2 < \cdots < i_\ell$ the ranks of the rays on which they were discovered, and by $d_1, \ldots, d_\ell$ their corresponding distances. We claim that

$$d_j \geq \frac{c''}{i_j + (t - \ell - 1)} \quad \text{for all} \quad 1 \leq j \leq \ell. \tag{4}$$

To show this, note that the number of targets that were found in all previous epochs (i.e., before the current epoch) is at most $t - \ell - 1$. Therefore, the ray with rank i_j in the current epoch had rank at most $i_j + (t - \ell - 1)$ in the previous epoch; this follows immediately from the rank update in lines 28–33 of the algorithm.

The definition of the i_j's implies that no target is found on all other rays that are searched in iteration c''. Thus, for all $i \in [1, m - (t - \ell - 1)] \setminus \{i_1, \ldots, i_\ell\}$ there is a ray that is searched to distance c''/i (this occurs on the ray of rank i in the current epoch). By putting this together with (4), we infer that for all i as above, there is a distinct target at distance at least c''/i, and that there are ℓ targets at distances at least $c''/(i_j + (t - \ell - 1))$. In what follows we shall exploit this to construct a lower bound on the intrinsic cost of Λ. Define

$$a_j = \begin{cases} j + (t - \ell - 1), & \text{if } j \in \{i_1, \ldots, i_\ell\} \\ j, & \text{otherwise} \end{cases}.$$

The previous discussion implies that for all $1 \leq j \leq m-(t-\ell-1)$, there is a distinct ray with a target at distance at least c''/a_j. Let b_j denote the value of the jth element in the increasingly sorted sequence of $\{a_i\}_{1 \leq i \leq m-(t-\ell-1)}$. Note that $b_1 \leq \ell+1$, since the pigeonhole principle implies that one of the values $[1, \ell+1]$ is not contained in $\{i_1, \ldots, i_\ell\}$. Similarly we can argue that $b_j \leq \ell + j$. We now define

$$\Lambda'' = \{\lambda''_1, \ldots, \lambda''_m\} = \left\{ \frac{c''}{\ell+1}, \frac{c''}{\ell+2}, \ldots, \frac{c''}{m-t+1+2\ell}, 0, \ldots, 0 \right\},$$

where the number of 0's is $t-\ell-1$. Then, $\Lambda \succ \Lambda''$, and thus $\xi_t(\Lambda) \geq \xi_{\ell+1}\left(\{\lambda''_1, \ldots, \lambda''_{m-(t-\ell-1)}\}\right)$. By applying Theorem 2 we infer that

$$\xi_t(\Lambda) \geq \min_{i_1 < \cdots < i_{\ell+1}} \left\{ \sum_{j=1}^{\ell} i_j (\lambda''_{i_j} - \lambda''_{i_{j+1}}) + i_{\ell+1} \lambda''_{i_{\ell+1}} \right\}. \tag{5}$$

Let $(i^*_j)_{1 \leq j \leq \ell+1}$ denote any choice of the i_j's that minimizes the above expression. Suppose that for all $1 \leq j \leq \ell+1$ we have that $i^*_j \leq \ell+1$. Then (5) simplifies to $\sum_{i=1}^{\ell+1} c''/(\ell+i) \geq c''/2$. On the other hand, suppose that there is a $1 \leq k \leq \ell+1$ such that $i^*_k \geq \ell+1$. Then, the bound in (5) is due to the monotonicity of the i^*_j's at least

$$i^*_{\ell+1} \lambda''_{i^*_{\ell+1}} = i^*_{\ell+1} \frac{c''}{\ell + i^*_{\ell+1}} \overset{(i^*_{\ell+1} \geq \ell)}{\geq} \frac{c''}{2}.$$

Since $c'' = \lfloor c/3 \rfloor$, the proof is completed. □

Lemma 2, Lemma 3 and (3) imply the main result of this section.

Theorem 3. *The adaptive hyperbolic-search algorithm locates t targets in m rays with associated distance set Λ at a search cost of at most $15 \log m \cdot \xi_t(\Lambda)$.*

3.3 An asymptotically optimal multi-target search algorithm

We now describe an optimal algorithm for locating t targets. We begin with a lower bound which demonstrates that the problem is at least as hard as searching a single target in $m-t+1$ rays. The proof is omitted for space reasons.

Theorem 4. *For the m ray problem in the partial information model, there exists a distance set Λ such that every deterministic algorithm that successfully locates t targets incurs a cost at least $\Omega(\log(m-t)) \cdot \xi_t(\Lambda)$, for at least one presentation of Λ.*

Let s be such that $t = m - s$. Note that if $s \geq m/2$, then $t \leq m/2$, which in turn implies that the adaptive hyperbolic search of Section 3.2 is asymptotically optimal, as follows from Theorem 3 and Theorem 4. Therefore, we shall focus only on the case $s \leq m/2$.

The optimal algorithm, which we call *Hybrid*, consists of two phases. In the first phase we perform a *uniform* search, i.e., we search all rays in the same fixed order, increasing by a unit the distance up to which we search rays in each iteration. Once a target is located in one of the rays, we effectively discard that ray, without affecting the ordering of the remaining rays. This phase continues until $m-2s$ targets have been located, at which point the algorithm switches to the adaptive hyperbolic search algorithm (Algorithm 1). In this second phase, we search for s more targets in the remaining $2s$ rays on which the uniform search did not locate any targets.

Theorem 5. *Let $s \leq m/2$. Algorithm Hybrid locates $t = m-s$ targets at a total cost $O(\log s) \cdot \xi_t(\Lambda)$, for any presentation with associated distance set Λ.*

Proof. Let C_1, C_2 denote the search costs incurred by the first and second phase, respectively. We will show that $C_1 \leq 2 \cdot \xi_t(\Lambda)$, and $C_2 = O(\log s) \cdot \xi_t(\Lambda)$, which will be sufficient to prove the theorem.

Consider first the uniform-search phase. From construction, the first target will be located after the uniform search incurs a cost of at most $m\lambda_m$; the second target will be located at an overall cost of at most $m\lambda_m + (m-1)(\lambda_{m-1} - \lambda_m)$; and more generally, by the time the l-th target is discovered, the uniform-search phase has not incurred cost more than $m\lambda_m + \sum_{i=1}^{l-1}(m-i)(\lambda_{m-i} - \lambda_{m-i+1})$. Therefore, the overall cost of Phase 1 is at most

$$C_1 \leq m\lambda_m + \sum_{j=1}^{m-2s-1}(m-j)(\lambda_{m-j} - \lambda_{m-j+1}). \tag{6}$$

On the other hand, from Theorem 2, we know that there exist $1 \leq i_1 < \ldots < i_t \leq m$ such that $\xi_t(\Lambda) \geq i_t\lambda_{i_t} + \sum_{j=1}^{t-1} i_j(\lambda_{i_j} - \lambda_{i_{j+1}})$. Since $i_t > i_{t-1}$, and $\lambda_{i_t} \geq \lambda_m$ we deduce that $\xi_t(\Lambda) \geq i_t\lambda_m + i_{t-1}(\lambda_{i_{t-1}} - \lambda_m) + \sum_{j=2}^{t-1} i_{t-j}(\lambda_{i_{t-j}} - \lambda_{i_{t-j+1}})$, from which it follows that

$$\xi_t(\Lambda) \geq i_t\lambda_m + \sum_{j=1}^{t-1} i_{t-j}(\lambda_{m-j} - \lambda_{m-j+1}). \tag{7}$$

Since the indices i_j assume different values, we know that $i_t \geq t = m-s$. More generally, we have that $i_{t-j} \geq t-j = m-s-j$, for all $j \in [0, m-2s-1]$. Thus,

$$\frac{m-j}{i_{t-j}} \leq \frac{m-j}{m-s-j} \leq 2 \qquad \text{for all } j \leq m-2s. \tag{8}$$

Combining (6), (7) and (8) we conclude that $C_1 \leq 2 \cdot \xi_t(\Lambda)$.

It remains to argue that $C_2 = O(\log s) \cdot \xi_t(\Lambda)$. Let M denote the subset of rays that are searched in phase 2, and let Λ_M denote the subset of Λ induced by the set M. By applying Theorem 3, we infer that $C_2 = O(\log(2s)) \cdot \xi_s(\Lambda_M)$. However, $\xi_s(\Lambda_M) \leq \xi_t(\Lambda)$. This is because even if the distances of the targets for the $m-2s$ rays involved in Phase 1 are revealed to the optimal (i.e., ideal) algorithm, such an algorithm would still have to locate s more targets among the rays in M. Thus, it follows that $C_2 = O(\log s) \cdot \xi_t(\Lambda)$, which is also the overall complexity of the search algorithm. □

4 Conclusions

Several problems remain to be studied in this setting, among them the optimal expected search cost of t targets as well as the case where only an arbitrary subset of size $s \leq t$ of the targets is being sought.

References

1. R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Information and Cmputation*, 106:234–244, 1993.
2. A. Beck. On the linear search problem. *Naval Research Logistics*, 2:221–228, 1964.
3. A. Beck and D.J. Newman. Yet more on the linear search problem. *Israel J. of Math.*, 8:419–429, 1970.
4. R. Bellman. An optimal search problem. *SIAM Review*, 5:274, 1963.
5. D.S. Bernstein, L. Finkelstein, and S. Zilberstein. Contract algorithms and robots on rays: unifying two scheduling problems. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1211–1217, 2003.
6. E.D. Demaine, S.P. Fekete, and S. Gal. Online searching with turn cost. *Theoretical Computer Science*, 361:342–355, 2006.
7. R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen. Competitive online approximation of the optimal search ratio. *SIAM Journal on Computing*, 38(3):881–898, 2008.
8. S. Gal. Minimax solutions for linear search problems. *SIAM J. on Applied Math.*, 27:17–30, 1974.
9. C.P. Gomes and B. Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1–2):43–62, 2001.
10. P. Jaillet and M. Stafford. Online searching. *Opes. Res.*, 49:234–244, 1993.
11. M-Y. Kao and M.L. Littman. Algorithms for informed cows. In *Proceedings of the AAAI 1997 Workshop on Online Search*, 1997.
12. M-Y. Kao, Y. Ma, M. Sipser, and Y.L. Yin. Optimal constructions of hybrid algorithms. *Journal of Algorithms*, 29(1):142–164, 1998.
13. M-Y. Kao, J.H. Reif, and S.R. Tate. Searching in an unknown environment:an optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–80, 1996.
14. D. G. Kirkpatrick. Hyperbolic dovetailing. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA)*, pages 616–627, 2009.
15. E. Koutsoupias, C.H. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proc. of the 23rd Int. Colloq. on Automata, Languages and Programming (ICALP)*, pages 280–289, 1996.
16. A. López-Ortiz and S. Schuierer. The ultimate strategy to search on m rays. *Theoretical Computer Science*, 261(2):267–295, 2001.
17. A. López-Ortiz and S. Schuierer. On-line parallel heuristics, processor scheduling and robot searching under the competitive framework. *Theoretical Computer Science*, 310(1–3):527–537, 2004.
18. A. McGregor, K. Onak, and R. Panigrahy. The oil searching problem. In *Proc. of the 17th European Symposium on Algorithms (ESA)*, pages 504–515, 2009.
19. S. Schuierer. Lower bounds in online geometric searching. *Computational Geometry: Theory and Applications*, 18(1):37–53, 2001.