



**HAL**  
open science

# Deterministic Treasure Hunt in the Plane with Angular Hints

Sébastien Bouchard, Yoann Dieudonné, Andrzej Pelc, Franck Petit

► **To cite this version:**

Sébastien Bouchard, Yoann Dieudonné, Andrzej Pelc, Franck Petit. Deterministic Treasure Hunt in the Plane with Angular Hints. 29th International Symposium on Algorithms and Computation, ISAAC 2018, Dec 2018, Jiaoxi Township, Taiwan. pp.48:1–48:13, 10.4230/LIPIcs.ISAAC.2018.48 . hal-01970990

**HAL Id: hal-01970990**

<https://hal.sorbonne-universite.fr/hal-01970990v1>

Submitted on 6 Jan 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# 1 Deterministic Treasure Hunt in the Plane with 2 Angular Hints

3 **Sébastien Bouchard**

4 Sorbonne Université, CNRS, INRIA, LIP6, F-75005 Paris, France

5 sebastien.bouchard@lip6.fr

6 **Yoann Dieudonné**

7 Laboratoire MIS, Université de Picardie Jules Verne, Amiens, France

8 yoann.dieudonne@u-picardie.fr

9 **Andrzej Pelc**<sup>1</sup>

10 Département d'informatique, Université du Québec en Outaouais, Gatineau, Canada

11 pelc@uqo.ca

12 **Franck Petit**<sup>2</sup>

13 Sorbonne Université, CNRS, INRIA, LIP6, F-75005 Paris, France

14 franck.petit@lip6.fr

---

## 15 — Abstract —

16 A mobile agent equipped with a compass and a measure of length has to find an inert treasure in  
17 the Euclidean plane. Both the agent and the treasure are modeled as points. In the beginning,  
18 the agent is at a distance at most  $D > 0$  from the treasure, but knows neither the distance nor any  
19 bound on it. Finding the treasure means getting at distance at most 1 from it. The agent makes  
20 a series of moves. Each of them consists in moving straight in a chosen direction at a chosen  
21 distance. In the beginning and after each move the agent gets a hint consisting of a positive  
22 angle smaller than  $2\pi$  whose vertex is at the current position of the agent and within which the  
23 treasure is contained. We investigate the problem of how these hints permit the agent to lower  
24 the cost of finding the treasure, using a deterministic algorithm, where the cost is the worst-case  
25 total length of the agent's trajectory. It is well known that without any hint the optimal (worst  
26 case) cost is  $\Theta(D^2)$ . We show that if all angles given as hints are at most  $\pi$ , then the cost can  
27 be lowered to  $O(D)$ , which is optimal. If all angles are at most  $\beta$ , where  $\beta < 2\pi$  is a constant  
28 unknown to the agent, then the cost is at most  $O(D^{2-\epsilon})$ , for some  $\epsilon > 0$ . For both these positive  
29 results we present deterministic algorithms achieving the above costs. Finally, if angles given as  
30 hints can be arbitrary, smaller than  $2\pi$ , then we show that cost  $\Theta(D^2)$  cannot be beaten.

31 **2012 ACM Subject Classification** F.2.2 Nonnumerical Algorithms and Problems; G.2.1 Com-  
32 binatorics

33 **Keywords and phrases** treasure hunt, deterministic algorithm, mobile agent, hint, plane

34 **Digital Object Identifier** 10.4230/LIPIcs.CVIT.2016.23

---

<sup>1</sup> This work was supported in part by NSERC discovery grant 8136 – 2013 and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

<sup>2</sup> This work was performed within Project ESTATE (Ref. ANR-16-CE25-0009-03), supported by French state funds managed by the ANR (Agence Nationale de la Recherche).



## 35 1 Introduction

36 **Motivation.** A tourist visiting an unknown town wants to find her way to the train station  
 37 or a skier lost on a slope wants to get back to the hotel. Luckily, there are many people  
 38 that can help. However, often they are not sure of the exact direction: when asked about  
 39 it, they make a vague gesture with the arm swinging around the direction to the target,  
 40 accompanying the hint with the words “somewhere there”. In fact, they show an angle  
 41 containing the target. Can such vague hints help the lost traveller to find the way to the  
 42 target? The aim of the present paper is to answer this question.

43 **The model and problem formulation.** A mobile agent equipped with a compass and  
 44 a measure of length has to find an inert treasure in the Euclidean plane. Both the agent  
 45 and the treasure are modeled as points. In the beginning, the agent is at a distance at most  
 46  $D > 0$  from the treasure, but knows neither the distance nor any bound on it. Finding the  
 47 treasure means getting at distance at most 1 from it. In applications, from such a distance  
 48 the treasure can be seen. The agent makes a series of moves. Each of them consists in  
 49 moving straight in a chosen direction at a chosen distance. In the beginning and after each  
 50 move the agent gets a hint consisting of a positive angle smaller than  $2\pi$  whose vertex is at  
 51 the current position of the agent and within which the treasure is contained. We investigate  
 52 the problem of how these hints permit the agent to lower the cost of finding the treasure,  
 53 using a deterministic algorithm, where the cost is the worst-case total length of the agent’s  
 54 trajectory. It is well known that the optimal cost of treasure hunt without hints is  $\Theta(D^2)$ .  
 55 (The algorithm of cost  $O(D^2)$  is to trace a spiral with jump 1 starting at the initial position  
 56 of the agent, and the lower bound  $\Omega(D^2)$  follows from Proposition 5.1 which establishes this  
 57 lower bound even assuming arbitrarily large angles smaller than  $2\pi$  given as hints.)

58 **Our results.** We show that if all angles given as hints are at most  $\pi$ , then the cost of  
 59 treasure hunt can be lowered to  $O(D)$ , which is optimal. Our real challenge here is in the  
 60 fact that hints can be angles of size *exactly*  $\pi$ , in which case the design of a trajectory always  
 61 leading to the treasure, while being cost-efficient in terms of traveled distance, is far from  
 62 obvious.

63 If all angles are at most  $\beta$ , where  $\beta < 2\pi$  is a constant unknown to the agent, then we  
 64 prove that the cost is at most  $O(D^{2-\epsilon})$ , for some  $\epsilon > 0$ . Finally, we show that arbitrary  
 65 angles smaller than  $2\pi$  given as hints cannot be of significant help: using such hints the cost  
 66  $\Theta(D^2)$  cannot be beaten.

67 For both our positive results we present deterministic algorithms achieving the above  
 68 costs. Both algorithms work in phases “assuming” that the treasure is contained in increasing  
 69 squares centered at the initial position of the agent. The common principle behind both  
 70 algorithms is to move the agent to strategically chosen points in the current square, depending  
 71 on previously obtained hints, and sometimes perform exhaustive search of small rectangles  
 72 from these points, in order to guarantee that the treasure is not there. This is done in such  
 73 a way that, in a given phase, obtained hints together with small rectangles exhaustively  
 74 searched, eliminate a sufficient area of the square assumed in the phase to eventually permit  
 75 finding the treasure.

76 In both algorithms, the points to which the agent travels and where it gets hints are  
 77 chosen in a natural way, although very differently in each of the algorithms. The main  
 78 difficulty is to prove that the distance travelled by the agent is within the promised cost. In  
 79 the case of the first algorithm, it is possible to cheaply exclude large areas not containing the  
 80 treasure, and thus find the treasure asymptotically optimally. For the second algorithm, the  
 81 agent eliminates smaller areas at each time, due to less precise hints, and thus finding the  
 82 treasure costs more.

83 Due to lack of space, the details of one of the algorithms and proofs of several results are  
 84 in the Appendix that is the full version of the paper.

85 **Related work.** The problem of treasure hunt, i.e., searching for an inert target by one  
 86 or more mobile agents was investigated under many different scenarios. The environment  
 87 where the treasure is hidden may be a graph or a plane, and the search may be deterministic  
 88 or randomized. An early paper [4] showed that the best competitive ratio for deterministic  
 89 treasure hunt on a line is 9. In [8] the authors generalized this problem, considering a  
 90 model where, in addition to travel length, the cost includes a payment for every turn of the  
 91 agent. The book [2] surveys both the search for a fixed target and the related rendezvous  
 92 problem, where the target and the finder are both mobile and their role is symmetric: they  
 93 both cooperate to meet. This book is concerned mostly with randomized search strategies.  
 94 Randomized treasure hunt strategies for star search, where the target is on one of  $m$  rays, are  
 95 considered in [13]. In [17, 19] the authors study relations between the problems of treasure  
 96 hunt and rendezvous in graphs. The authors of [3] study the task of finding a fixed point  
 97 on the line and in the grid, and initiate the study of the task of searching for an unknown  
 98 line in the plane. This research is continued, e.g., in [12, 16]. In [18] the authors concentrate  
 99 on game-theoretic aspects of the situation where multiple selfish pursuers compete to find a  
 100 target, e.g., in a ring. The main result of [15] is an optimal algorithm to sweep a plane in  
 101 order to locate an unknown fixed target, where locating means to get the agent originating  
 102 at point  $O$  to a point  $P$  such that the target is in the segment  $OP$ . In [10] the authors  
 103 consider the generalization of the search problem in the plane to the case of several searchers.  
 104 Collective treasure hunt in the grid by several agents with bounded memory is investigated  
 105 in [9, 14]. In [5], treasure hunt with randomly faulty hints is considered in tree networks. By  
 106 contrast, the survey [7] and the book [6] consider pursuit-evasion games, mostly on graphs,  
 107 where pursuers try to catch a fugitive target trying to escape.

## 108 2 Preliminaries

109 Since for  $D \leq 1$  treasure hunt is solved immediately, in the sequel we assume  $D > 1$ . Since  
 110 the agent has a compass, it can establish an orthogonal coordinate system with point  $O$   
 111 with coordinates  $(0, 0)$  at its starting position, the  $x$ -axis going East-West and the  $y$ -axis  
 112 going North-South. Lines parallel to the  $x$ -axis will be called horizontal, and lines parallel to  
 113 the  $y$ -axis will be called vertical. When the agent at a current point  $a$  decides to go to a  
 114 previously computed point  $b$  (using a straight line), we describe this move simply as “Go  
 115 to  $b$ ”. A hint given to the agent currently located at point  $a$  is formally described as an  
 116 ordered pair  $(P_1, P_2)$  of half-lines originating at  $a$  such that the angle clockwise from  $P_1$  to  
 117  $P_2$  (including  $P_1$  and  $P_2$ ) contains the treasure.

118 The line containing points  $A$  and  $B$  is denoted by  $(AB)$ . A segment with extremities  $A$   
 119 and  $B$  is denoted by  $[AB]$  and its length is denoted  $|AB|$ . Throughout the paper, a polygon  
 120 is defined as a closed polygon (i.e., together with the boundary). For a polygon  $S$ , we will  
 121 denote by  $\mathcal{B}(S)$  (resp.  $\mathcal{I}(S)$ ) the boundary of  $S$  (resp. the interior of  $S$ , i.e., the set  $S \setminus \mathcal{B}(S)$ ).  
 122 A rectangle is defined as a non-degenerate rectangle, i.e., with all sides of strictly positive  
 123 length. A rectangle with vertices  $A, B, C, D$  (in clockwise order) is denoted simply by  $ABCD$ .  
 124 A rectangle is *straight* if one of its sides is vertical.

125 In our algorithms we use the following procedure `RectangleScan( $R$ )` whose aim is to  
 126 traverse a closed rectangle  $R$  (composed of the boundary and interior) with known coordinates,  
 127 so that the agent initially situated at some point of  $R$  gets at distance at most 1 from every  
 128 point of it and returns to the starting point. We describe the procedure for a straight  
 129 rectangle whose vertical side is not shorter than the horizontal side. The modification of

130 the procedure for arbitrarily positioned rectangles is straightforward. Let the vertices of the  
 131 rectangle  $R$  be  $A, B, C$  and  $D$ , where  $A$  is the North-West vertex and the others are listed  
 132 clockwise. Let  $a$  be the point at which the agent starts the procedure.

133 The idea of the procedure is to go to vertex  $A$ , then make a snake-like movement in  
 134 which consecutive vertical segments are separated by a distance 1, and then go back to point  
 135  $a$ . The agent ignores all hints gotten during the execution of the procedure. Suppose that  
 136 the horizontal side of  $R$  has length  $m$  and the vertical side has length  $n$ , with  $n \geq m$ . Let  
 137  $k = \lfloor m \rfloor$ . Let  $a_0, a_1, \dots, a_k$  be points on the North horizontal side of the rectangle, such  
 138 that  $a_0 = A$  and the distance between consecutive points is 1. Let  $b_0, b_1, \dots, b_k$  be points  
 139 on the South horizontal side of the rectangle, such that  $b_0 = D$  and the distance between  
 140 consecutive points is 1.

141 The pseudocode of procedure `RectangleScan( $R$ )` is given in Algorithm 1.

---

**Algorithm 1** Procedure `RectangleScan( $R$ )`

---

```

1: if  $k$  is odd then
2:   for  $i = 0$  to  $k - 1$  step 2 do
3:     Go to  $a_i$ ; Go to  $b_i$ ;
4:     Go to  $b_{i+1}$ ; Go to  $a_{i+1}$ 
5:   end for
6:   Go to  $a$ 
7: else
8:   for  $i = 0$  to  $k - 2$  step 2 do
9:     Go to  $a_i$ ; Go to  $b_i$ ;
10:    Go to  $b_{i+1}$ ; Go to  $a_{i+1}$ 
11:  end for
12:  Go to  $a_k$ ; Go to  $b_k$ 
13:  Go to  $a$ 
14: end if

```

---

142 ► **Proposition 2.1.** For every point  $p$  of the rectangle  $R$ , the agent is at distance at most  
 143 1 from  $p$  at some time of the execution of Procedure `RectangleScan( $R$ )`. The cost of the  
 144 procedure is at most  $5n \cdot \max(m, 2)$ , where  $n \geq m$  are the lengths of the sides of the rectangle.

### 145 **3 Angles at most $\pi$**

146 In this section we consider the case when all angles given as hints are at most  $\pi$ . Without  
 147 loss of generality we can assume that they are all equal to  $\pi$ , completing any smaller angle to  
 148  $\pi$  in an arbitrary way: this makes the situation even harder for the agent, as hints become  
 149 less precise. For such hints we show Algorithm `TreasureHunt1` that finds the treasure at  
 150 cost  $O(D)$ . This is of course optimal, as the treasure can be at any point at distance at most  
 151  $D$  from the starting point of the agent.

152 For angles of size  $\pi$ , every hint is in fact a half-plane whose boundary line  $L$  contains the  
 153 current location of the agent. For simplicity, we will code such a hint as  $(L, right)$  or  $(L, left)$ ,  
 154 whenever the line  $L$  is not horizontal, depending on whether the indicated half-plane is to  
 155 the right (i.e., East) or to the left (i.e., West) of  $L$ . For any non-horizontal line  $L$  this is  
 156 non-ambiguous. Likewise, when  $L$  is horizontal, we will code a hint as  $(L, up)$  or  $(L, down)$ ,  
 157 depending on whether the indicated half-plane is up (i.e., North) from  $L$  or down (i.e., South)  
 158 from  $L$ .

159 In view of the work on  $\phi$ -self-approaching curves (cf. [1]) we first note that there is a big  
 160 difference of difficulty between obtaining our result in the case when angles given as hints  
 161 are *strictly smaller* than  $\pi$  and when they are *at most*  $\pi$ , as we assume. A  $\phi$ -self-approaching  
 162 curve is a planar oriented curve such that, for each point  $B$  on the curve, the rest of the curve  
 163 lies inside a wedge of angle  $\phi$  with apex in  $B$ . In [1], the authors prove the following property

164 of these curves: for every  $\phi < \pi$  there exists a constant  $c(\phi)$  such that the length of any  
 165  $\phi$ -self-approaching curve is at most  $c(\phi)$  times the distance  $D$  between its endpoints. Hence,  
 166 for angles  $\phi$  strictly smaller than  $\pi$ , our result could possibly be derived from the existing  
 167 literature: roughly speaking, the agent should follow a trajectory corresponding to any  
 168  $\phi$ -self-approaching curve to find the treasure at a cost linear in  $D$ . Even then, transforming  
 169 the continuous scenario of self-approaching curves to our discrete scenario presents some  
 170 difficulties. However, the crucial problem is this: the result of [1] holds only when  $\phi < \pi$   
 171 (the authors also emphasize that for each  $\phi \geq \pi$ , the property is false), and thus the above  
 172 derivation is no longer possible for our purpose when  $\phi = \pi$ . Actually, this is the real difficulty  
 173 of our problem: handling angles equal to  $\pi$ , i.e., half-planes.

174 We further observe that a rather straightforward treasure hunt algorithm of cost  $O(D \log D)$ ,  
 175 for hints being angles of size  $\pi$ , can be obtained using an immediate corollary of a theorem  
 176 proven in [11] by Grünbaum: each line passing through the centroid of a convex polygon cuts  
 177 the polygon into two convex polygons with areas differing by a factor of at most  $\frac{5}{4}$ . Suppose  
 178 for simplicity that  $D$  is known. Starting from the square of side length  $2D$ , centered at the  
 179 initial position of the agent, this permits to reduce the search area from  $P$  to at most  $\frac{5P}{9}$  in a  
 180 single move. Hence, after  $O(\log D)$  moves, the search area is small enough to be exhaustively  
 181 searched by procedure `RectangleScan` at cost  $O(D)$ . However, the cost of each move during  
 182 the reduction is not under control and can be only bounded by a constant multiple of  $D$ ,  
 183 thus giving the total cost bound  $O(D \log D)$ . By contrast, our algorithm controls both the  
 184 remaining search area and the cost incurred in each move, yielding the optimal cost  $O(D)$ .

185 The high-level idea of our Algorithm `TreasureHunt1` is the following. The agent acts in  
 186 phases  $j = 1, 2, 3, \dots$  where in each phase  $j$  the agent “supposes” that the treasure is in a  
 187 straight square  $R_j$  centered at the initial position of the agent, and of side length  $2^j$ . When  
 188 executing a phase  $j$ , the agent successively moves to distinct points with the aim of using  
 189 the hints at these points to narrow the search area that initially corresponds to  $R_j$ . In our  
 190 algorithm, this narrowing is made in such a way that the remaining search area is always  
 191 a straight rectangle. Often this straight rectangle is a strict superset of the intersection of  
 192 all hints that the agent was given previously. This would seem to be a waste, as we are  
 193 searching some areas that have been previously excluded. However, this loss is compensated  
 194 by the ease of searching description and subsequent analysis of the algorithm, due to the fact  
 195 that, at each stage, the search area is very regular.

196 During a phase, the agent proceeds to successive reductions of the search area by moving  
 197 to distinct locations, until it obtains a rectangular search area that is small enough to be  
 198 searched directly at low cost using procedure `RectangleScan`. In our algorithm, such a final  
 199 execution of `RectangleScan` in a phase is triggered as soon as the rectangle has a side smaller  
 200 than 4. If the treasure is not found by the end of this execution of procedure `RectangleScan`,  
 201 the agent learns that the treasure cannot be in the supposed straight square  $R_j$  and starts  
 202 the next phase from scratch by forgetting all previously received hints. This forgetting again  
 203 simplifies subsequent analysis. The algorithm terminates at the latest by the end of phase  
 204  $j_0 = \lceil \log_2 D \rceil + 1$ , in which the supposed straight square  $R_{j_0}$  is large enough to contain the  
 205 treasure. Hence, if the cost of a phase  $j$  is linear in  $2^j$ , then the cost of the overall solution is  
 206 linear in the distance  $D$ .

207 In order to give the reader deeper insights in the reasons why our solution is valid and  
 208 has linear cost, we need to give more precise explanations on how the search area is reduced  
 209 during a given phase  $j \geq 2$  (when  $j = 1$ , the agent makes no reduction and directly scans  
 210 the small search area using procedure `RectangleScan`). Suppose that in phase  $j \geq 2$  the  
 211 agent is at the center  $p$  of a search area corresponding to a straight rectangle  $R$ , every side of

## 23:6 Deterministic Treasure Hunt in the Plane with Angular Hints

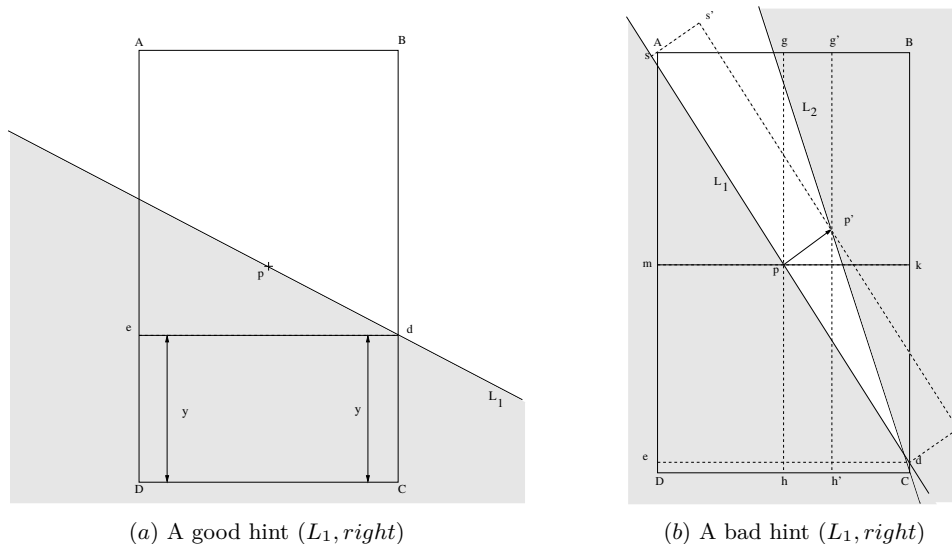
212 which has length between 4 and  $2^j$  (note that this is the case at the beginning of the phase),  
 213 and denote by  $A, B, C$  and  $D$  the vertices of  $R$  starting from the top left corner and going  
 214 clockwise. In order to reduce rectangle  $R$ , the agent uses the hint at point  $p$ . The obtained  
 215 hint denoted by  $(L_1, x_1)$  can be of two types: either a *good* hint or a *bad* hint. A good hint is  
 216 a hint whose line  $L_1$  divides one of the sides of  $R$  into two segments such that the length  $y$   
 217 of the smaller one is at least 1. A bad hint is a hint that is not good.

218 If the received hint  $(L_1, x_1)$  is good, then the agent narrows the search area to a rectangle  
 219  $R' \subset R$  having the following three properties:

- 220 1.  $R \setminus R'$  does not contain the treasure.
- 221 2. The difference between the perimeters of  $R$  and  $R'$  is  $2y \geq 2$ .
- 222 3. The distance from  $p$  to the center of  $R'$  is exactly  $\frac{y}{2}$ .

223 and then moves to the center of  $R'$ .

224 An illustration of such a reduction is depicted in Figure 1(a). The reduced search area  
 225  $R'$  is the rectangle  $ABde$ .



■ **Figure 1** In Figure (a) the agent received a good hint  $(L_1, right)$  at the point  $p$  of a rectangular search area  $ABCD$ . In Figure (b) it received a bad hint  $(L_1, right)$  at the point  $p$  and hence it moved to point  $p'$  and got a hint  $(L_2, left)$ . In both figures the excluded half-planes are shaded.

226 If the agent receives a bad hint, say  $(L_1, right)$ , at the center of a rectangular search area  
 227  $R$ , we cannot apply the same method as the one used for a good hint: this is the reason for  
 228 the distinction between good and bad hints. If we applied the same method as before, we  
 229 could obtain a rectangular search area  $R'$  such that the difference between the perimeters of  
 230  $R$  and  $R'$  is at least  $2y$ . However, in the context of a bad hint, the difference  $2y$  may be very  
 231 small (even null), and hence there is no significant reduction of the search area. In order to  
 232 tackle this problem, when getting a bad hint at the center  $p$  of  $R$ , the agent moves to another  
 233 point  $p'$  which is situated in the half-plane  $(L_1, right)$  at distance 2 from  $p$ , perpendicularly  
 234 to  $L_1$ . This point  $p'$  is chosen in such a way that, regardless of what is the second hint, we  
 235 can ensure that two important properties described below are satisfied.

236 The first property is that by combining the two hints, the agent can decrease the search  
 237 area to a rectangle  $R' \subset R$  whose perimeter is smaller by 2 compared to the perimeter of  $R$ , as



238 it is the case for a good hint, and such that  $R \setminus R'$  does not contain the treasure. This decrease  
 239 follows either directly from the pair of hints, or indirectly after having scanned some relatively  
 240 small rectangles using procedure `RectangleScan`. In the example depicted in Fig. 1 (b),  
 241 after getting the second hint ( $L_2, left$ ), the agent executes procedure `RectangleScan(ss'd'd)`  
 242 followed by `RectangleScan(gg'h'h)` and moves to the center of the new search area  $R'$  that  
 243 is the rectangle  $Agpm$ . Note that the part of  $R'$  not excluded by the two hints and by  
 244 the procedure `RectangleScan` executed in rectangles  $ss'd'd$  and  $gg'h'h$  is only the small  
 245 quadrilateral bounded by line  $L_2$  and the segments  $[AB]$ ,  $[s'd']$  and  $[gh]$ . However, in order  
 246 to preserve the homogeneity of the process, we consider the entire new search area  $R'$  which  
 247 is a straight rectangle whose perimeter is smaller by at least 2, compared to that from  $R$ .  
 248 This follows from the fact that no side of  $R$  has length smaller than 4. The agent finally  
 249 moves to the center of  $R'$ .

250 The second property is that all of this (i.e., the move from  $p$  to  $p'$ , the possible scans  
 251 of small rectangles and finally the move to the center of  $R'$ ) is done at a cost linear in the  
 252 difference of perimeters of  $R$  and  $R'$ , as shown in the Appendix. The two properties together  
 253 ensure that, even with bad hints, the agent manages to reduce the search area in a significant  
 254 way and at a small cost. So, regardless of whether hints are good or not, we can show that  
 255 the cost of phase  $j$  is in  $\mathcal{O}(2^j)$  and the treasure is found during this phase if the initial square  
 256 is large enough. The difficulty of the solution is in showing that the moves prescribed by our  
 257 algorithm in the case of bad hints guarantee the two above properties, and thus ensure the  
 258 correctness of the algorithm and the cost linear in  $D$ .

259 The details of the algorithm and its analysis are in the Appendix.

260 ► **Theorem 1.** *Consider an agent  $A$  and a treasure located at distance at most  $D$  from the*  
 261 *initial position of  $A$ . By executing Algorithm `TreasureHunt1`, agent  $A$  finds the treasure*  
 262 *after having traveled a distance  $\mathcal{O}(D)$ .*

## 263 4 Angles bounded by $\beta < 2\pi$

264 In this section we consider the case when all hints are angles upper-bounded by some constant  
 265  $\beta < 2\pi$ , unknown to the agent. The main result of this section is Algorithm `TreasureHunt2`  
 266 whose cost is at most  $\mathcal{O}(D^{2-\epsilon})$ , for some  $\epsilon > 0$ . For a hint  $(P_1, P_2)$  we denote by  $\overline{(P_1, P_2)}$   
 267 the complement of  $(P_1, P_2)$ .

### 268 4.1 High level idea

269 In Algorithm `TreasureHunt2`, similarly as in the previous algorithm, the agent acts in phases  
 270  $j = 1, 2, 3, \dots$ , where in each phase  $j$  the agent “supposes” that the treasure is in the straight  
 271 square centered at its initial position and of side length  $2^j$ . The intended goal is to search  
 272 each supposed square at relatively low cost, and to ensure the discovery of the treasure by  
 273 the time the agent finishes the first phase for which the initial supposed square contains the  
 274 treasure. However, the similarity with the previous solution ends there: indeed, the hints  
 275 that may now be less precise do not allow us to use the same strategy within a given phase.  
 276 Hence we adopt a different approach that we outline below and that uses the following notion  
 277 of tiling. Given a square  $S$  with side of length  $x > 0$ ,  $Tiling(i)$  of  $S$ , for any non-negative  
 278 integer  $i$ , is the partition of square  $S$  into  $4^i$  squares with side of length  $\frac{x}{2^i}$ . Each of these  
 279 squares, called *tiles*, is closed, i.e., contains its border, and hence neighboring tiles overlap in  
 280 the common border.

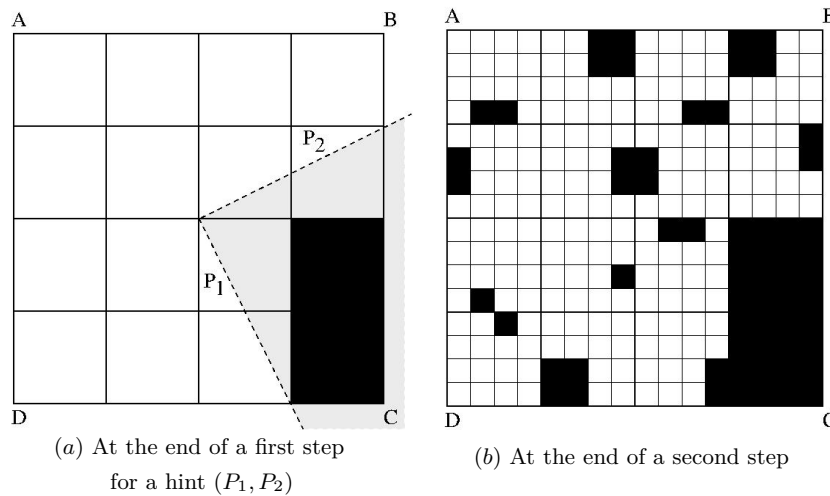
281 Let us consider a simpler situation in which the angle of every hint  $(P_1, P_2)$  is always  
 282 equal to the bound  $\beta$ : the general case, when the angles may vary while being at most  $\beta$ ,



283 adds a level of technical complexity that is unnecessary to understand the intuition. In the  
 284 considered situation, the angle of each excluded zone  $\overline{(P_1, P_2)}$  is always the same as well. The  
 285 following property holds in this case: there exists an integer  $i_\beta$  such that for every square  $S$   
 286 and every hint  $(P_1, P_2)$  given at the center of  $S$ , at least one tile of  $Tiling(i_\beta)$  of  $S$  belongs  
 287 to the excluded zone  $\overline{(P_1, P_2)}$ .

288 In phase  $j$ , the agent performs  $k$  steps: we will indicate later how the value of  $k$  should  
 289 be chosen. At the beginning of the phase, the entire square  $S$  is white. In the first step, the  
 290 agent gets a hint  $(P_1, P_2)$  at the center of  $S$ . By the above property, we know that  $\overline{(P_1, P_2)}$   
 291 contains at least one tile of  $Tiling(i_\beta)$  of  $S$ , and we have the guarantee that such a tile  
 292 cannot contain the treasure. All points of all tiles included in  $\overline{(P_1, P_2)}$  are painted black in  
 293 the first step. This operation does not require any move, as painting is performed in the  
 294 memory of the agent. As a result, at the end of the first step, each tile of  $Tiling(i_\beta)$  of  $S$  is  
 295 either black or white, in the following precise sense: a black tile is a tile all of whose points  
 296 are black, and a white tile is a tile all of whose interior points are white.

297 In the second step, the agent repeats the painting procedure at a finer level. More  
 298 precisely, the agent moves to the center of each white tile  $t$  of  $Tiling(i_\beta)$  of  $S$ . When it gets  
 299 a hint at the center of a white tile  $t$ , there is at least one tile of  $Tiling(i_\beta)$  of  $t$  that can be  
 300 excluded. As in the first step, all points of these excluded tiles are painted black. Note that a  
 301 tile of  $Tiling(i_\beta)$  of  $t$  is actually a tile of  $Tiling(2i_\beta)$  of  $S$ . Moreover, each tile of  $Tiling(i_\beta)$   
 302 of  $S$  is made of exactly  $4^{i_\beta}$  tiles of  $Tiling(2i_\beta)$  of  $S$ . Hence, as depicted in Figure 2, the  
 303 property we obtain at the end of the second step is as follows: each tile of  $Tiling(2i_\beta)$  of  $S$   
 is either black or white.



■ **Figure 2** White and black tiles at the end of the first and the second step of a phase, for square  $S = ABCD$  and  $i_\beta = 2$ .

304 In the next steps, the agent applies a similar process at increasingly finer levels of tiling.  
 305 More precisely, in step  $2 < s \leq k$ , the agent moves to the center of each white tile of  
 306  $Tiling((s - 1)i_\beta)$  of  $S$  and gets a hint that allows it to paint black at least one tile of  
 307  $Tiling(s \cdot i_\beta)$  of  $S$ . At the end of step  $s$ , each tile of  $Tiling(s \cdot i_\beta)$  of  $S$  is either black or white.  
 308 We can show that at each step  $s$  the agent paints black at least  $\frac{1}{4^{i_\beta}}$ th of the area of  $S$  that is  
 309 white at the beginning of step  $s$ .

310 After step  $k$ , each tile of  $Tiling(k \cdot i_\beta)$  of  $S$  is either black or white. These steps permit  
 311 the agent to exclude some area without having to search it directly, while keeping some  
 312

313 regularity of the shape of the black area. The agent paints black a smaller area than excluded  
 314 by the hints but a more regular one. This regularity enables in turn the next process in  
 315 the area remaining white. Indeed, the agent subsequently executes a brute-force searching  
 316 that consists in moving to each white tile of  $Tiling(k \cdot i_\beta)$  of  $S$  in order to scan it using the  
 317 procedure `RectangleScan`. If, after having scanned all the remaining white tiles, it has not  
 318 found the treasure, the agent repaints white all the square  $S$  and enters the next phase. Thus  
 319 we have the guarantee that the agent finds the treasure by the end of phase  $\lceil \log_2 D \rceil + 1$ , i.e.,  
 320 a phase in which the initial supposed square is large enough to contain the treasure. The  
 321 question is: how much do we have to pay for all of this? In fact, the cost depends on the  
 322 value that is assigned to  $k$  in each phase  $j$ . The value of  $k$  must be large enough so that the  
 323 distance travelled by the agent during the brute-force searching is relatively small. At the  
 324 same time, this value must be small enough so that the the distance travelled during the  $k$   
 325 steps is not too large. A good trade-off can be reached when  $k = \lceil \log_{4^{i_\beta}} \sqrt{2^j} \rceil$ . Indeed, as  
 326 highlighted in the proof of correctness, it is due to this carefully chosen value of  $k$  that we  
 327 can beat the cost  $\Theta(D^2)$  necessary without hints, and get a complexity of  $\mathcal{O}(D^{2-\epsilon})$ , where  $\epsilon$   
 328 is a positive real depending on  $i_\beta$ , and hence depending on the angle  $\beta$ .

## 329 4.2 Algorithm and analysis

330 In this subsection we describe our algorithm in detail, prove its correctness and analyze  
 331 its complexity. In the Appendix we define a function  $index : (0, 2\pi) \rightarrow \mathbb{N}^+$  that has the  
 332 following properties, for any angle  $0 < \alpha < 2\pi$ .

- 333 1. For every square  $S$  and for every hint  $(P_1, P_2)$  of size  $2\pi - \alpha$  obtained at the center of  $S$ ,  
 334 there exists a tile of  $Tiling(index(\alpha))$  of  $S$  included in  $\overline{(P_1, P_2)}$ .
- 335 2. For every angle  $\alpha' < \alpha$ , we have  $index(\alpha) \leq index(\alpha')$ .

336 In the sequel, the integer  $index(\alpha)$  is called the index of  $\alpha$ . Algorithm 2 gives a pseudo-  
 337 code of the main algorithm of this section. It uses the function `Mosaic` described in Algorithm  
 338 3 that is the key technical tool permitting the agent to reduce its search area. The agent  
 339 interrupts the execution of Algorithm 2 as soon as it gets at distance 1 from the treasure, at  
 340 which point it can “see” it and thus treasure hunt stops.

---

### Algorithm 2 TreasureHunt2

---

```

1:  $IndexNew := 1$ 
2:  $i := 1$ 
3: loop
4:   repeat
5:      $IndexOld := IndexNew$ 
6:      $IndexNew := \text{Mosaic}(i, IndexOld)$ 
7:   until  $IndexNew = IndexOld$ 
8:    $i := i + 1$ 
9: end loop

```

---

341 In the following, a square is called black if all its points are black. A square is called  
 342 white if all points of its interior are white. (In a white square, some points of its border may  
 343 be black).

344 ► **Lemma 2.** *For any positive integers  $i$  and  $k$ , consider an agent executing function*  
 345 *`Mosaic`( $i, k$ ) from its initial position  $O$ . Let  $S$  be the straight square centered at  $O$  with side*  
 346 *of length  $2^i$ . For every positive integer  $j \leq \lceil \log_{4^k} \sqrt{2^i} \rceil$ , at the end of the  $j$ -th execution of the*  
 347 *first loop (lines 5 to 20) in `Mosaic`( $i, k$ ), each tile of  $Tiling(jk)$  of  $S$  is either black or white.*

## 23:10 Deterministic Treasure Hunt in the Plane with Angular Hints

348 ▶ **Lemma 3.** *For every positive integers  $i$  and  $k$ , a call to function  $\text{Mosaic}(i, k)$  has cost at*  
 349 *most  $2^i \frac{3 + \log_4 k (4^k - 1)}{2} + 2k + 8$ .*

---

### Algorithm 3 Function $\text{Mosaic}(i, k)$

---

```

1:  $O :=$  the initial position of the agent
2:  $S :=$  the straight square centered at  $O$  with sides of length  $2^i$ 
3: Paint white all points of  $S$ 
4:  $IndexMax := k$ 
5: for  $j = 1$  to  $\lceil \log_{4^k} \sqrt{2^i} \rceil$  do
6:   for all tiles  $t$  of  $\text{Tiling}((j-1)k)$  of  $S$  do
7:     if  $t$  is white then
8:       Go to the center of  $t$ 
9:       Let  $(P_1, P_2)$  be the obtained hint
10:       $k' :=$  index of  $\overline{(P_1, P_2)}$ 
11:      if  $k' > IndexMax$  then
12:         $IndexMax := k'$ 
13:      end if
14:      if  $IndexMax = k$  then
15:        for all tiles  $t'$  of  $\text{Tiling}(k)$  of  $t$  such that  $t' \subset \overline{(P_1, P_2)}$  do
16:          Paint black all points of  $t'$ 
17:        end for
18:      end if
19:    end if
20:  end for
21: end for
22: if  $IndexMax = k$  then
23:   for all tiles  $t$  of  $\text{Tiling}(k(\lceil \log_{4^k} \sqrt{2^i} \rceil))$  of  $S$  do
24:     if  $t$  is white then
25:       Go to the center of  $t$ 
26:       Execute  $\text{RectangleScan}(t)$ 
27:     end if
28:   end for
29: end if
30: Go to  $O$ 
31: return  $IndexMax$ 

```

---

350 Let  $\psi$  be the index of  $2\pi - \beta$ . The next proposition follows from the aforementioned  
 351 properties of the function  $index$ .

352 ▶ **Proposition 4.1.** Let  $(P_1, P_2)$  be any hint. The index of  $\overline{(P_1, P_2)}$  is at most  $\psi$ .

353 Using Lemmas 2, 3 and Proposition 4.1 we prove the final result of this section.

354 ▶ **Theorem 4.** *Consider an agent  $A$  and a treasure located at distance at most  $D$  from the*  
 355 *initial position of  $A$ . By executing Algorithm  $\text{TreasureHunt2}$ , agent  $A$  finds the treasure*  
 356 *after having traveled a distance in  $\mathcal{O}(D^{2-\epsilon})$ , for some  $\epsilon > 0$ .*

357 **Proof.** We will use the following two claims.

358 ▶ **Claim 4.1.** Let  $i \geq 1$  be an integer. The number of executions of the repeat loop in the  
 359  $i$ -th execution of the external loop in Algorithm 2 is bounded by  $\psi$ .

360 **Proof of the claim:** Suppose by contradiction that the claim does not hold for some  $i \geq 1$ .  
 361 So, the number of executions of the repeat loop in the  $i$ -th execution of the external loop  
 362 in Algorithm 2 is at least  $\psi + 1$ . In each of these executions of the repeat loop, the agent  
 363 calls function  $\text{Mosaic}(i, *)$  exactly once. For all  $1 \leq j \leq \psi + 1$  ( $\psi \geq 1$ , by definition of an  
 364 index), denote by  $v_j$  the returned value of function  $\text{Mosaic}(i, *)$  in the  $j$ -th execution of the  
 365 repeat loop in the  $i$ -th execution of the external loop. Note that  $v_1 \neq 1$ : indeed, if  $v_1 = 1$   
 366 the repeat loop would be executed exactly once, which would be a contradiction because it  
 367 is executed at least  $\psi + 1 \geq 2$  times.

368 In view of Algorithm 2 and Proposition 4.1, the returned value of `Mosaic`( $i, *$ ) is a positive  
 369 integer that is at most  $\psi$ . Since  $v_1 \neq 1$ , this implies that  $\psi \geq 2$ . Moreover, for all  $2 \leq j \leq \psi$ ,  
 370 we have  $v_j \geq v_{j-1}$  (cf. lines 5-6 of Algorithm 2 and lines 4, 11-12 of Algorithm 3). Hence,  
 371 there exists an integer  $k \leq \psi$  such that  $v_k = v_{k-1}$ . However, according to Algorithm 2, this  
 372 implies that the number of executions of the repeat loop in the  $i$ -th execution of the external  
 373 loop is at most  $k \leq \psi$ . This is a contradiction which concludes the proof of the claim.  $\star$

374 **► Claim 4.2.** The distance traveled by the agent before variable  $i$  becomes equal to  $\lceil \log_2 D \rceil + 2$   
 375 in the execution of Algorithm 2 is  $\mathcal{O}(D^{2-\epsilon})$ , where  $\epsilon = \frac{1}{2}(1 - \log_4 \psi (4^\psi - 1)) > 0$ .

376 **Proof of the claim:** In view of the fact that the returned value of every call to function  
 377 `Mosaic` in the execution of Algorithm 2 is at most  $\psi$ , it follows that in each call to function  
 378 `Mosaic`( $*, k$ ) the parameter  $k$  is always at most  $\psi$ . Hence, in view of Claim 4.1 and Lemma 3,  
 379 as long as variable  $i$  does not reach the value  $\lceil \log_2 D \rceil + 2$ , the agent traveled a distance at  
 380 most

$$381 \quad \psi \cdot \sum_{i=1}^{\lceil \log_2 D \rceil + 1} 2^i \frac{3 + \log_4 \psi (4^\psi - 1)}{2} + 2\psi + 8 \quad (1)$$

$$382 \quad \leq \psi 2^{(\lceil \log_2 D \rceil + 1)} \frac{3 + \log_4 \psi (4^\psi - 1)}{2} + 2\psi + 9 \quad (2)$$

$$383 \quad \leq \psi 2^{2\psi + 12 + \log_4 \psi (4^\psi - 1)} 2^{(\log_2 D)} \frac{3 + \log_4 \psi (4^\psi - 1)}{2} \quad (3)$$

$$384 \quad = \psi 2^{2\psi + 12 + \log_4 \psi (4^\psi - 1)} D^{2 - \frac{1}{2}(1 - \log_4 \psi (4^\psi - 1))} \quad (4)$$

386 By (4), the total distance traveled by the agent executing Algorithm 2 is  $\mathcal{O}(D^{2-\epsilon})$  where  
 387  $\epsilon = \frac{1}{2}(1 - \log_4 \psi (4^\psi - 1))$ . Since  $\psi$  is a positive integer, we have  $0 < \log_4 \psi (4^\psi - 1) < 1$  and  
 388 hence  $\epsilon > 0$ . This ends the proof of the claim.  $\star$

389 Assume that the theorem is false. As long as variable  $i$  does not reach  $\lceil \log_2 D \rceil + 2$ ,  
 390 the agent cannot find the treasure, as this would contradict Claim 4.2. Thus, in view of  
 391 Claim 4.1, before the time  $\tau$  when variable  $i$  reaches  $\lceil \log_2 D \rceil + 2$  the treasure is not found.  
 392 By Algorithm 2, this implies that during the last call to function `Mosaic` before time  $\tau$ ,  
 393 the function returns a value that is equal to its second input parameter. This implies that  
 394 during this call, the agent has executed lines 23 to 28 of Algorithm 3: more precisely, there is  
 395 some integer  $x$  such that from each white tile  $t$  of `Tiling`( $x$ ) of the straight square  $S$  that is  
 396 centered at the initial position of the agent and that has sides of length  $2^{\lceil \log_2 D \rceil + 1}$ , the agent  
 397 has executed function `RectangleScan`( $t$ ). Hence, at the end of the execution of lines 23  
 398 to 28, the agent has seen all points of each white tile of `Tiling`( $x$ ) of  $S$ . Moreover, in view  
 399 of Lemma 2, we know that the tiles that are not white, in `Tiling`( $x$ ) of  $S$ , are necessarily  
 400 black. Given a black tile  $\sigma$  of `Tiling`( $x$ ), each point of  $\sigma$  is black, which, in view of lines 15  
 401 to 17 of Algorithm 3, implies that  $\sigma$  cannot contain the treasure. Since square  $S$  necessarily  
 402 contains the treasure, it follows that the agent must find the treasure by the end of the last  
 403 execution of function `Mosaic` before time  $\tau$ . As a consequence, the agent stops the execution  
 404 of Algorithm 2 before assigning  $\lceil \log_2 D \rceil + 2$  to variable  $i$  and thus, we get a contradiction  
 405 with the definition of time  $\tau$ , which proves the theorem.  $\blacktriangleleft$

## 406 5 Arbitrary angles

407 We finally observe that if hints can be arbitrary angles smaller than  $2\pi$  then the treasure  
 408 hunt cost  $\Theta(D^2)$  cannot be improved in the worst case.

409 **► Proposition 5.1.** If hints can be arbitrary angles smaller than  $2\pi$  then the optimal cost of  
 410 treasure hunt for a treasure at distance at most  $D$  from the starting point of the agent is  
 411  $\Omega(D^2)$ .

412 **6 Conclusion**

413 For hints that are angles at most  $\pi$  we gave a treasure hunt algorithm with optimal cost  
 414 linear in  $D$ . For larger angles we showed a separation between the case where angles are  
 415 bounded away from  $2\pi$ , when we designed an algorithm with cost strictly subquadratic in  $D$ ,  
 416 and the case where angles have arbitrary values smaller than  $2\pi$ , when we showed a quadratic  
 417 lower bound on the cost. The optimal cost of treasure hunt with large angles bounded away  
 418 from  $2\pi$  remains open. In particular, the following questions seem intriguing. Is the optimal  
 419 cost linear in  $D$  in this case, or is it possible to prove a super-linear lower bound on it? Does  
 420 the order of magnitude of this optimal cost depend on the bound  $\pi < \beta < 2\pi$  on the angles  
 421 given as hints?

422 — **References** —

- 423 **1** O. Aichholzer, F. Aurenhammer, C. Icking, R. Klein, E. Langetepe and G. Rote, Generalized  
 424 self-approaching curves, *Discrete Applied Mathematics* 109 (2001), 3-24
- 425 **2** S.Alpern and S.Gal, *The Theory of Search Games and Rendezvous*, Kluwer Academic Public-  
 426 ations, 2003.
- 427 **3** R. Baeza-Yates, J. Culberson, and J. Rawlins, Searching the plane, *Information and Compu-  
 428 tation* 106 (1993), 234-252.
- 429 **4** A. Beck, D.J. Newman, Yet more on the linear search problem, *Israel J. Math.* 8 (1970),  
 430 419-429.
- 431 **5** L. Boczkowski, A. Korman, Y. Rodeh, Searching on trees with noisy memory,  
 432 arXiv:1611.01403v2 [cs.DS] (2017).
- 433 **6** A. Bonato and R.J. Nowakowski, *The Game of Cops and Robbers on Graphs*, American  
 434 Mathematical Society, 2011.
- 435 **7** T.H. Chung, G.A. Hollinger and V. Isler, Search and pursuit-evasion in mobile robotics: A  
 436 survey, *Auton. Robot* 31 (2011), 299-316.
- 437 **8** E. Demaine, S. Fekete, S. Gal, Online searching with turn cost, *Theoretical Computer Science*  
 438 361 (2006), 342 D 355.
- 439 **9** Y. Emek, T. Langner, D. Stolz, J. Uitto, R. Wattenhofer, How many ants does it take to find  
 440 the food? *Theoretical Computer Science* 608 (2015), 255-267.
- 441 **10** G. M. Fricke, J. P. Hecker, A. D. Griego, L. T. Tran and Melanie E. Moses, A Distributed  
 442 Deterministic Spiral Search Algorithm for Swarms, *Proc. IEEE/RSJ International Conference  
 443 on Intelligent Robots and Systems (IROS 2016)*, 4430-4436.
- 444 **11** B. Grünbaum, Partitions of mass-distributions and convex bodies by hyperplanes, *Pacific J.  
 445 Math.* 10 (1960), 1257-1261
- 446 **12** A. Jez and J. Lopuszanski, On the two-dimensional cow search problem, *Information Processing  
 447 Letters* 109 (2009), 543 - 547.
- 448 **13** M.Y. Kao, J.H. Reif, S.R. Tate, Searching in an unknown environment: an optimal randomized  
 449 algorithm for the cow-path problem, *Information and Computation* 131 (1996), 63-79.
- 450 **14** B. Keller, T. Langner, J. Uitto, R. Wattenhofer, Overcoming obstacles with ants, *Proc. 19th  
 451 International Conference on Principles of Distributed Systems (OPODIS 2015)*, 1-17.
- 452 **15** E. Langetepe, On the Optimality of Spiral Search, *Proc. 21st Ann. ACM-SIAM Symp. Disc.  
 453 Algor. (SODA 2010)*, 1-12.
- 454 **16** E. Langetepe, Searching for an axis-parallel shoreline, *Theoretical Computer Science* 447  
 455 (2012), 85-99.
- 456 **17** A. Miller, A. Pelc, Tradeoffs between cost and information for rendezvous and treasure hunt,  
 457 *Journal of Parallel and Distributed Computing* 83 (2015), 159-167.
- 458 **18** K. Spieser and E. Frazzoli, The Cow-Path Game: A Competitive Vehicle Routing Problem,  
 459 *Proc. 51st IEEE Conference on Decision and Control* (2012), 6513 - 6520.
- 460 **19** A. Ta-Shma and U. Zwick, Deterministic rendezvous, treasure hunts and strongly universal  
 461 exploration sequences. *ACM Transactions on Algorithms* 10 (2014), 12:1-12:15.