Searching with Increasing Speeds*

Leszek Gąsieniec, Shuji Kijima, and Jie Min

Abstract

In the classical search problem on the line or in higher dimension one is asked to find the shortest (and often the fastest) route to be adopted by a robot R from the starting point s towards the target point t located at unknown location and distance D. It is usually assumed that robot R moves with a fixed unit speed 1. It is well known that one can adopt a "zig-zag" strategy based on the exponential expansion, which allows to reach the target located on the line in time $\leq 9D$, and this bound is tight. The problem was also studied in two dimensions where the competitive factor is known to be $O(D)$.

In this paper we study an alteration of the search problem in which robot R starts moving with the initial speed 1. However, during search it can encounter a point or a sequence of points enabling faster and faster movement. The main goal is to adopt the route which allows R to reach the target t as quickly as possible. We study two variants of the considered search problem: (1) with the *global knowledge* and (2) with the *local knowledge*. In variant (1) robot R knows *a priori* the location of all intermediate points as well as their expulsion speeds. In this variant we study the complexity of computing optimal search trajectories. In variant (2) the relevant information about points in P is acquired by R gradually, i.e., while moving along the adopted trajectory. Here the focus is on the competitive factor of the solution, i.e., the ratio between the solutions computed in variants (2) and (1). We also consider two types of search spaces with points distributed on the line and subsequently with points distributed in two-dimensional space.

1 Introduction

Search problems refer to frequently considered combinatorial (structural or algorithmic) problems within and across multiple fields including operations research, computing, mathematics and others. The search problem in the form studied in this paper was originally posed more than a half-century ago by Bellman [8] who asked: *"A hiker is lost in a forest which size is not known to her. What is the best path to adopt to escape the forest?"*

In more general terms, search problems deal with either single or multiple searchers looking for a hidden object referred to as *target*, with the ultimate goal of minimising the time required to accomplish the task. Numerous variants of the problem have been considered reflecting on different search spaces (e.g., a geometric setting vs. a graph), whether the target is fixed or mobile, whether the search space is stable, or if the target is a point or a collection of points, a curve or a closed non-zero volume region. Another separation line refers to

*This work was partially done while the first author visited Kyushu University and is supported in part by the Networks Sciences and Technologies (NeST) initiative in the School of EEECS, University of Liverpool.

deterministic versus randomized search strategies, and whether the searchers have access to extra tools supporting navigation, see [4, 5, 6, 9, 10, 18, 20, 25, 26].

The search on the line has been analysed in detail by Baeza-Yates et al. in [4] under the name of the *cow-path problem*. This seminal work prompted further work on on different variants of the search problem including extensions [5, 6, 18, 21, 22, 26, 28]. In addition to the line, Baeza-Yates, et al. [4, 5] studied the cow-path problem on co-centred w infinite rays, and proposed a deterministic algorithm with the name *linear spiral search*. The case with $w = 1$ is trivial, and for $w = 2$ (the case with infinite line) the algorithm always finds the target in time at most $9D$, where D is the time needed to move from the starting point s to the target t. They also provided the lower bound argument showing optimality of their solution up to lower order terms. In the same work, the authors considered also a system of rays with $w > 2$ showing an optimal (up to lower order terms) result of $\left(1 + 2\frac{w^w}{(w-1)^{w-1}}\right) \cdot D$ time bound to find the target using a deterministic search strategy.
In [6] Baeza-Yates and Schott examined also other variants of the cow-path problem. They observed that if D is known in advance, the search on the line requires time $3D$ in the worst case. They also studied scenarios with two or more robots having uniform speeds. They show that if robots are able to communicate at arbitrary distance, the total distance $2D$ must be travelled to find the destination, and $4D$ if the two robots must reach the destination. Baeza-Yates and Schott showed also that the total distance travelled when no communication is present, and both robots must reach the target is also $9D$, the same time it would take a single robot. A similar study, however with an arbitrary number of robots can be found in [11] where the authors study also the case with different speeds. More tight analysis for two robots with different speeds was subsequently published in [7]. The case with multiple speeds was also studied in the context of patrolling linear environments first by Czyzowicz *et al* in [14] and in the follow up work of Kawamura and Kobayashi in [23]. Another interesting study on robots with different speeds can be found in [13] where the authors distinguish between moving and searching speeds.

In terms of probabilistic approach Kao, et al. [22] examined the first randomized algorithm for the cow-path problem and, for the case of $w = 2$ rays, he obtained an optimal randomized $4.59112 \cdot D$ bound for the search time. They also provided a bound for $w > 2$ paths, where they conjecture their this approach to be an optimal randomized strategy.

In this paper we consider the search problem in which the robot can increase its speed by visiting specific points in space. This work apart from having an intrinsic combinatorial value can be also seen as a simplification of the *gravity assist* concept [27] used in space exploration. Also there is some parallel to sharing schemes with different vehicle types, e.g., city bikes combined with electric cars and others.

1.1 The Model and The Search Problem

In this work we consider search by a single robot R either on the infinite line or in two-dimensional (Euclidean) space. The robot has a zero-visibility radius, moves freely and starts the exploration with the uniform speed 1. The search space is populated by n points from set $P = \{p_1, \ldots, p_n\}$, with the starting point $s = p_\sigma$ and the target $t = p_\tau$, for some integer $1 \leq \sigma \neq \tau \leq n$. Similarly to the past work in this area we assume that the points in P have integer coordinates. This is to avoid dealing with infinitesimal moves and the

assumption about non-zero visibility radius of R. Each point $p_i \in P$ has the associated *expulsion speed* v_i. More precisely, robot R always leaves p_i with the speed v_i if the speed it entered p_i was smaller or equal. Please note that R can only go faster, i.e., visiting a node with a smaller expulsion speed does not affect the current speed of R. For the completeness we also assume that $v_\sigma = 1$ and $v_\tau = +\infty$.

The *main task* for robot R is to compute (and subsequently adopt) the fastest route from the starting point s to the target/destination point t taking advantage of the increasing expulsion speeds of points in P visited on the way to t. We study two variants of the considered search problem: (1) with the *global knowledge* and (2) with the *local knowledge*. In variant (1) robot R knows *a priori* the location of all points in P as well as their expulsion speeds. In this variant we study the complexity of computing optimal search trajectories. In variant (2) the relevant information about points in P is acquired by R gradually, i.e., while moving along the adopted trajectory. Here the focus is on the competitive ratio of the solution, i.e., the ratio between the solution computed in variant (2) and the optimal solution from (1). We also consider two types of search spaces with points distributed on the line and subsequently with points distributed in two-dimensional space.

1.2 Our Contribution

The following results constitute the contribution of this paper.

On the line In *variant (1)* with full knowledge we show that the line of points can be processed in time $O(n)$ to find the fastest route from any point in P to t. The algorithm is based on the known solution for the range queries. In fact after $O(n)-$time preprocessing one can query any point on the line for the shortest route to t in time $O(\log n)$. In *variant (2)* with local knowledge we show that the trajectory based on the classical "zig-zag" strategy always admits a competitive factor 9. I.e., consistently with the classical version of the search problem where it is known that one cannot reduce this constant in the worst case.

In 2D space In *variant (1)* we observe that one can process P with Dijkstra's algorithm in time $O(n^2)$ to compute the fastest route from any point in P to target t. After this preprocessing one can query any point on the plane for the shortest route to t in time $Q(n)$, where $Q(n) = \text{polylog}(n)$ refers to query time for the nearest point in additively weighted Voronoi diagrams of size n. Using this result we show that if there are at most k different expulsion speeds one can process the points in P in time $O(k \cdot n \cdot \text{polylog}(n))$. In *variant (2)* we show that the spiral strategy admits the asymptotically optimal competitive ratio $O(D)$.

2 Search on the line

In this section we assume that the moves of robot R are limited to an infinite (integer) line $\mathcal{L}$ which contains all points in P offering different expulsion speeds. In section 2.1 we consider the search problem in the *full-knowledge* model where we show how to find the fastest route to from the starting point s to the target t in the optimal time $O(n)$. We also comment on querying arbitrary points on the line. Later in section 2.2 we show that the classical zig-zag strategy [5] provides 9-competitive solution also when R is allowed to increase its speed throughout the searching process.

2.1 *Variant (1) - with full knowledge*

Recall that in this variant robot R is fully aware of its own starting position s, the content of P including offered speeds and the location of target t.

Example In order to build some intuition we first consider a simple informal example, see figure Figure 1, where $s = p_4$ is the starting point with the initial speed $v_4 = 1$, each point p_i offers the relevant expulsion speed v_i, and the arcs indicate the consecutive steps (during which R moves with strictly increasing expulsion speeds) on the optimal (fastest) path towards target $t = p_8$.

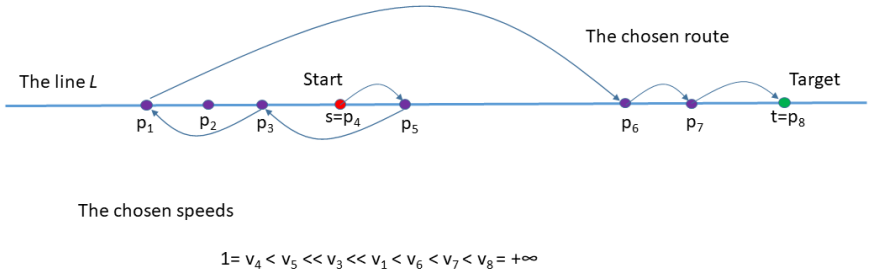

Figure 1: Solution example

In the example above the robot must have a very good reason to turn back (as it is initially moving towards the target) after visiting p_5. In other words it must be more beneficial for the robot to visit (and to adopt the expulsion speeds of) points p_3 and p_1, rather than going directly from p_5 to p_6. This can happen when points p_1, p_3, and p_5 are relatively close to each other and $v_5 \ll v_3 \ll v_1$. And the total time needed to move with speed v_5 from p_5 to p_3, then with speed v_3 from p_3 to p_1, and finally with speed v_1 towards p_6 is smaller than going directly from p_5 to p_6 with speed v_5. The adopted route has to be faster also from a more direct route $p_5 \to p_3 \to p_6$. This example indicates also that robot R changes the direction on its walk only in points with higher expulsion speeds.

Motivated by this example one can summarise the properties of the optimal (fastest) walk to be adopted by the robot as follows.

1. While visiting point p_i robot R adopts the expulsion speed v_i iff v_i is higher than the current speed of R. This reflects the assumption that at any time R moves with the highest possible speed encountered so far.

2. Robot R changes the direction on its walk only at points with higher speeds. I.e., changing direction without increasing speed always results in suboptimal solution as one could construct a faster route by turning a bit earlier.

3. It is enough to compute for each point $p_i \in P$ the closest to the left and to the right points $HL(p_i)$ and $HR(p_i)$ with higher expulsion speeds than v_i. According to properties 1 and 2 these are the only points in which the speed and possibly the direction of the walk change.

Let a *walk* be a direct move from $p_i \in P$ to $p_j \in P$ with the expulsion speed v_i (walks are denoted by arcs in the example). Thanks to property 3 one can conclude that in search for the optimal solution (from any point in P to t) instead of dealing with a quadratic number of walks, it is enough to consider at most $2n$ walks connecting any $p_i \in P$ with the corresponding $HL(p_i)$ and $HR(p_i)$.

Nearest larger neighbour Note that all these walks can be determined in time $O(n)$ by swiping (with the help of a single stack) the line of points once in each direction. During each swipe, e.g., from left to right, while processing each point p_i we assume inductively that on the top of the stack we have the expulsion speeds of p_{i-1}, and below of $HL(p_{i-1})$, and below of $HL(HL(p_{i-1}))$, etc. In order to find $HL(p_i)$ we keep removing points from the stack until we find a point with a higher expulsion speed than v_i. We set this point as $HL(p_i)$ and to maintain the invariant we push p_i on the top of the stack. The process explained above is a known solution to the classical *nearest larger neighbour* problem [3].

Note that the directed graph solely based on points in $p_i \in P$ and the respective walks p_i to $HL(p_i)$ and to $HL(p_i)$ is acyclic, I.e., the walks always lead towards higher speeds. Thus the points in P can be sorted topologically (starting from the target t) in time $O(n)$. Finally, one can visit these points one by one in the computed order to determine the fastest route between any point $p_i \in P$ and target t. Such route is computed instantly on the basis of the fastest routes already computed for $HL(p_i)$ and $HR(p_i)$ as the fastest route from p_i to t has to visit first one of these points. The following theorem holds.

Theorem 1 *For the collection P of n points on the line one can compute the fastest route between any point p_i and target t in the optimal time $O(n)$.*

Having computed optimal routes towards t for all points in P one can also compute for any point $p \in \mathcal{L}$ the closest point (to the left and to the right) in P by simple binary search in time $O(\log n)$. This allows to compute the fastest route from any point p to target t in time $O(\log n)$.

2.2 *Variant (2) - with local knowledge*

Recall that in this case, the robot only knows its initial speed 1 and is not aware of neither the location of other points nor the expulsion speeds available in them. In what follows we show that the classical "zig-zag" strategy $Z_{\mathcal{L}}$ can be adopted here with the same 9-competitive guarantee as in the cow-path problem.

Given an instance I of the considered search problem. Let the route $S \equiv (s = p_\sigma \xrightarrow{v_\sigma=1} p_{i_2} \xrightarrow{v_{i_2}} \dots p_{i_l} \xrightarrow{v_{i_l}} t = p_\tau)$ be the optimal solution of I. The points chosen to this solution are called *critical points*. This solution is based on l potentially overlapping segments on the line L. The first segment defined by critical points p_σ, p_{i_2} is traversed with the speed $v_\sigma = 1$. The next $l-2$ segments based on points $p_{i_j}, p_{i_{j+1}}$ are traversed with the speeds v_{i_j} respectively, for all $j = 2, \dots, l-1$. The last segment based on points p_{i_l}, p_τ is traversed with the speed v_{i_l}. Let d_j be the total distance traversed from p_σ to p_{i_j}, for all $j = 2, \dots, l$, and d_{l+1} referring to the total distance traversed on the way to target $t = p_\tau$. In addition let D_j be the absolute (Euclidean) distance between p_σ and all critical points included in the

optimal solution S. Note that the lengths of l segments defined above can be expressed as $d_j - d_{j-1}$, for all $j = 2, \ldots, l$, where $d_1 = 0$. And finally, the respective traversal times on the considered segments are: $\frac{d_2}{v_\sigma}$ on segment (p_σ, p_{i_2}), $\frac{d_{j+1}-d_j}{v_{i_j}}$ on segments $(p_{i_j}, p_{i_{j+1}})$, for all $j = 2, \ldots, l$, including $\frac{d_{l+1}-d_l}{v_{i_l}}$ on segment (p_{i_l}, p_τ).

Lemma 2 *Given a traversal path $U = (p_{i_1} \xrightarrow{v_{i_1}} p_{i_2} \xrightarrow{v_{i_2}} ... \xrightarrow{v_{i_{k-1}}} p_{i_k})$ with strictly increasing expulsion speeds and the traversal time $T(U)$. And another traversal path with the same points $U' = (p_{i_1} \xrightarrow{v'_{i_1}} p_{i_2} \xrightarrow{v'_{i_2}} ... \xrightarrow{v'_{i_{k-1}}} p_{i_k})$ with the traversal time $T(U')$, where $v_{i_j} \leq v'_{i_j}$, for all $j = 1, \ldots, k$. Then $T(U') \leq T(U)$.*

Proof. The thesis of the lemma follows directly from the fact that all segments are shared by U and U', and each of them is traversed not slower in U'. □

Lemma 3 *If the order of critical points used in the optimal solution $S \equiv (s = p_\sigma \xrightarrow{v_\sigma=1} p_{i_2} \xrightarrow{v_{i_2}} \ldots p_{i_l} \xrightarrow{v_{i_l}} t = p_\tau)$ corresponds to the first occurrences of these points on the zigzag path $Z_\mathcal{L}$, the traversal time admitted by $Z_\mathcal{L}$ is 9-competitive.*

Proof. Let P_Z be the actual path that robot R adopted on the way to target t, i.e., the relevant prefix of $Z_\mathcal{L}$. Also, let d'_j be the length of prefix of P_Z until the first encounter of p_{i_j} and v'_{i_j} be the expulsion speed associated with the segment of P_Z connecting p_{i_j} and $p_{i_{j+1}}$. This segment is of length $d'_{i_{j+1}} - d'_{i_j}$. Thus the total traversal time to target t along consecutive segments of $Z_\mathcal{L}$ is

$$T(P_Z) = \sum_{j=1}^{l} \frac{d'_{i_{j+1}} - d'_{i_j}}{v'_{i_j}}$$

Note that the speed used by robot R between consecutive critical points can be the same as in the optimal solution, or it may be faster as due to taking wider swings (on $Z_\mathcal{L}$) robot R can pick some faster expulsion speeds earlier at non-critical points visited on the way. Thus the (average) speeds adopted between critical points satisfy $v_{i_j} \leq v'_{i_j}$. And, the competitive ratio of the "zig-zag" strategy can be expressed as:

$$r = \frac{T(P_Z)}{T(S)} = \frac{\sum_{j=1}^{l} \frac{d'_{i_{j+1}} - d'_{i_j}}{v'_{i_j}}}{\sum_{j=1}^{l} \frac{d_{i_{j+1}} - d_{i_j}}{v_{i_j}}} \leq \frac{\sum_{j=1}^{l} \frac{d'_{i_{j+1}} - d'_{i_j}}{v_{i_j}}}{\sum_{j=1}^{l} \frac{d_{i_{j+1}} - d_{i_j}}{v_{i_j}}}$$
$$= \frac{\frac{d'_{i_l}}{v_{i_l}} + \sum_{j=1}^{l-1} d'_{i_j}(\frac{1}{v_{i_{j-1}}} - \frac{1}{v_{i_j}})}{\frac{d_{i_l}}{v_{i_l}} + \sum_{j=1}^{l-1} d_{i_j}(\frac{1}{v_{i_{j-1}}} - \frac{1}{v_{i_j}})}.$$

Now knowing that $d'_{i_j} \geq D_{i_j}$ and using the fact from [5] that the distance walked along $Z_\mathcal{L}$ towards each critical point p_{i_j} is $d'_{i_j} \leq 9D_{i_j}$, for each $j = 1, \ldots, l$, we can estimate the competitive ratio

$$r \leq \frac{\frac{9D_{i_l}}{v_{i_l}} + \sum_{j=1}^{l-1} 9D_{i_j}(\frac{1}{v_{i_{j-1}}} - \frac{1}{v_{i_j}})}{\frac{D_{i_l}}{v_{i_l}} + \sum_{j=1}^{l-1} D_{i_j}(\frac{1}{v_{i_{j-1}}} - \frac{1}{v_{i_j}})} = 9$$

□

We conclude with the following theorem.

Theorem 4 *The traversal time admitted by $Z_{\mathcal{L}}$ is 9-competitive.*

Proof. We already know, see Lemma 3, that the 9-competitive ratio is secured if the order in which the critical points are visited in the optimal solution is the same as their first occurrences in $Z_{\mathcal{L}}$. However, if this order is altered certain critical points will be approached (and segments in between traversed) with faster speeds than in the optimal solution S. This observation combined with Lemma 2 admit the thesis of the theorem.

□

3 Search on 2d plane

In this section we consider the search problem in 2d Euclidean plane Π. Similarly to the case on the line we study first the variant with the full knowledge and later focus on the case where robot R has only local knowledge. Also in this section we assume that the points in P have integer coordinates.

3.1 *Variant (1) - with global knowledge*

The task of finding the optimal route in 2d-plane is to some extent similar to the case on the line. Namely, one can construct a $DAG = (P, A)$ with a collection A of directed edges (arcs) $p_i \to p_j$, for all $p_i, p_j \in P$ with $v_i < v_j$. Each arc $p_i \to p_j$ has the associated weight representing the time needed to traverse from p_i to p_j with the expulsion speed v_i available in p_i. The size of DAG is quadratic in $|P| = n$, thus one can solve the search problem by finding all shortest (fastest) paths from points in P to target t in time $O(n^2)$.

While in the case on the line we managed to reduce the size of such DAG to $O(n)$, in 2d-plane the challenge is steeper due to greater freedom of movement of robot R. In addition, after computing all shortest paths in DAG further queries on arbitrary points (outside of P) for the fastest routes towards target t remain non-trivial. To counterpart, one can use the concept of *additively weighted Voronoi diagrams*, see, e.g., [17], based on points in P where each $p_i \in P$ has weight w_i which reflects the time required to move from p_i to t in DAG. This type of diagram partitions the whole plane into n cells $C_1, \ldots, C_n$, where cell C_i contains all points p for which the value $|(p, p_i)| + w_i$ is minimised w.r.t. all $i = 1, \ldots, n$. It is known, that one can compute additively weighted Voronoi diagrams on n points in time $O(n \log n)$ [17]. One can also enhance such diagrams in time $O(n \cdot \text{polylog} n)$ to enable a (randomised) algorithm finding the closest (among n) weighted point in time $\mathcal{Q}(n) = \text{polylog}(n)$, see the work of Karavelas and Yvinec [19] based on the ideas from [15]. While this complexity is not as good as the basic query time $O(\log n)$ available for unweighed points, see the classical algorithm of Kirkpatrick for planar point location [24], it still allows

us to construct a faster solution to the search problem if there is a relatively small number $k \ll n$ of distinct expulsion speeds $v_1^* \geq \cdots \geq v_k^*$ present in the system.

The invariant The improved construction of all fastest paths (to target t) operates in k rounds and is based on the following invariant. On the conclusion of round i we compute the fastest routes to t for any point with the expulsion speed at least as fast as v_i^*. Let $P_i = \{p_{i_1}, \ldots, p_{i_m}\} \subset P$ be the set of all such points with the traversal times $T_{i_1}, \ldots, T_{i_m}$ respectively. Note that these times are computed for good, i.e., they never change, as in further rounds we only add points with strictly smaller expulsion speeds. During round $i+1$ for each point $p \in P_{i+1} \setminus P_i$ we need to determine whether robot should go directly to t or should be relayed via some point in P_i with a higher expulsion speed. The time of moving directly to target t can be computed easily, however, choosing the right relay point in P_i is more complex.

The best relay node In the solution we use additively weighted Voronoi diagrams in which times $T_{i_1}, \ldots, T_{i_m}$ will determine (after proper rescaling) the weights of points in P_i. Recall that in additively weighted Voronoi diagrams the closest point is chosen according to the (Euclidean) distance to the point added to its weight. In our problem we try to minimise the sum of times needed to walk from p to a point $p_{i_j} \in P_i$ and its weight T_{i_j} which is $\frac{dist(p,p_{i_j})}{v_{i+1}^*} + T_{i_j}$. Since the first term is not referring to the Euclidean distance we can multiply both terms by v_{i+1}^* to obtain $dist(p, p_{i_j}) + T_{i_j} \cdot v_{i+1}^*$. This rescaling applied for each point in P_i does not change the selection of the fastest route to t via points in P_i, while it allows to use additively weighted Voronoi diagrams to speed up the search for the best relay node in P_i. Thus if we construct an additively weighted Voronoi diagram for points in P_i with weights $T_{i_1} \cdot v_{i+1}^*, \ldots, T_{i_m} \cdot v_{i+1}^*$, we can find for any $p \in P_{i+1} \setminus P_i$ the best relay node in P_i in polylogarithmic time.

The following theorem holds.

Theorem 5 *If the number of distinct speeds is limited to $k \ll n$ one can find all fastest routes to target t in time $O(n \cdot k \cdot \text{polylog}(n))$.*

Proof. The algorithm works in k rounds. During each round one needs to construct an additively weighted Voronoi diagram which is enhanced to answer the closest point queries. The total cost of such construction is $k \cdot O(n \cdot \text{polylog}(n))$. In addition, every point in P is queried exactly once during the search for the best relay node. This give the total complexity $k \cdot O(n \cdot \text{polylog}(n)) + O(n \cdot \text{polylog}(n)) = O(n \cdot k \cdot \text{polylog}(n))$. □

After computing all fastest paths from points in P to target t one can compute one more (enhanced) additively weighted Voronoi diagram to provide the fastest route queries for any point in Π in time $Q(n) = \text{polylog}(n)$.

3.1.1 Search on a 2d-grid

In the last part of this section we show that if all points in P are located on a relatively small grid with at least one dimension limited to size $g \ll n$ (e.g., the grid has g rows) and the robot is allowed to use only edges of the grid one can find all fastest routes to target t in time $O(g \cdot n \log n)$.

Dynamic nearest larger neighbour In this model we also use the solution to the *nearest larger neighbour* problem on the line. However this time we adopt the dynamic version in which one can ask queries at arbitrary points, remove and add values in time $O(\log n)$, where n is the cap on the number of values currently stored. The three operations can be implemented with the help of a balanced binary search tree, in which all values are kept in the leaves and each internal node contains the largest value stored in the respective subtree.

Also here the construction is done by considering distinct expulsion speeds in decreasing order $v_1^* \geq \cdots \geq v_k^*$, for some $k \leq n$. In fact we use the same notation, division into rounds and a similar invariant. In particular, we assume that on the conclusion of round i the fastest routes from all points in P_i to t are already computed and they never change. In addition we assume that the points from P_i are processed in the relevant rows for the nearest larger neighbour queries according to their expulsion speeds.

During round $i+1$ we consider points from $P_{i+1} \setminus P_i$ in an arbitrary order. Let $p \in P_{i+1} \setminus P_i$ where p belongs to some column c in the grid. In order to compute the fastest route from p to t we first compute the fastest direct route (without relay nodes) in constant time. This route needs to be compared with the best route via some relay node in P_i. In order to find the best relay node we query each row at column c for the nearest largest value, i.e., to find the closest points p_l and p_r, to the left and right respectively, with larger expulsion speeds for which the fastest routes are already computed in earlier rounds. These are the only relay points in this row which need to be considered as going directly (i.e., not visiting any other relay points in this or some other rows which are considered separately) to any other relay point in this row will always result in slower solution. Thus the fastest route from p to target t can be computed by examining at most $2g$ nodes which can be done in time $O(g \cdot \log n)$. And when the fastest route from p is finally computed, we insert p to the *nearest larger neighbour* solution in the relevant row in time $O(\log n)$.

Finally, since the cost of inclusion (finding the fastest route) of each node in P is bounded by time $O(g \cdot \log n)$ we conclude with the following theorem.

Theorem 6 *If the points from P are distributed in a grid with one dimension limited to $g \ll n$ and robot R can move only along edges of the grid, all fastest routes towards target t can be computed in time $O(g \cdot n \log n)$.*

3.2 *Variant (2) - with local knowledge*

It is well known that in the classical search problem in 2d space the competitive ratio of search process is $\Omega(D)$ as on the way to target t located at an unspecified distance D robot R needs to visit all (discrete, with integer coordinates) points within a ball of radius D centred in s. Since there are $\Omega(D^2)$ integral points in such ball and the fastest route is of length D the competitive ratio follows.

In this section we show that analogously to the classical search the spiral strategy admits also in this case $O(D)$-competitive solution w.r.t. the fastest route from the starting point s to target t. Since we adopted the model with the integral points we will use a simplification of the spiral shape formed of borders b_i of increasing in size boxes B_i, where $B_0 = \{s\}$ with $s = (x_\sigma, y_\sigma)$, and for $i \geq 1$ box B_i contains all points $u = (x, y)$, such that $|x - x_\sigma|, |y - y_\sigma| \leq i$. The border b_i is defined as $B_i \setminus B_{i-1}$, it has a square shape and it contains exactly $8 \cdot i$ integral points, for any $i \geq 0$. The spiral strategy instructs robot R to search through the

consecutive (with increasing i) borders b_i, and to adopt faster expulsion speeds as soon as they are encountered.

The proof of $O(D)$-competitiveness is done in two steps. We first relocate points in set P such that the fastest solution S' for the new locations of points is at least as fast as S, which is the fastest solution for the original location of points in P. We later show that the spiral based solution is $O(D)-$competitive with respect to S', so in turn it is also $O(D)-$competitive with respect to S.

Let $D' \leq D$ be the index of the border to which target t belongs to, i.e., $t \in b_{D'}$ and let $S \equiv (s = p_{i_1} \xrightarrow{v_\sigma=1} p_{i_2} \xrightarrow{v_{i_2}} \dots p_{i_l} \xrightarrow{v_{i_l}} t)$ be the fastest route from s to t. We construct a different arrangement (with alternative locations) of points in the solution S in which if $p_{i_j} \in S$ belongs to border b_j, for any $j < D'$, it is moved to the location $(x_\sigma, y_{\sigma+j})$ on the vertical line originating in s. All other points in S including target t are moved to the location $(x_\sigma, y_{\sigma+D'})$.

Let $S' \equiv (s = p_{i'_1} \xrightarrow{v_{i'_1}=1} p_{i'_2} \xrightarrow{v_{i'_2}} \dots p_{i'_{m-1}} \xrightarrow{v_{i'_{m-1}}} p_{i'_m} = t)$ be the fastest route from s to t on the newly formed line. We point out here that the time complexity $T(S')$ of the solution S' is not worse than the time complexity $T(S)$, in other words $T(S') \leq T(S)$. And this happens because the distance between any pair of points in S can be only reduced during the relocation process.

Finally, we show that the time complexity T_s of our spiral strategy applied to points in P (before rearrangement) is only $O(D)$ multiplicative factor away from $T(S')$. In the analysis, we bound T_s from above by T_s^* referring to the time complexity of a "lazy" strategy in which while searching border b_i robot R uses the fastest expulsion speed $v^*(i-1)$ encountered earlier in box B_{i-1}, and the fastest expulsion speed found in b_i (if larger than $v^*(i-1)$) will be used only in border b_{i+1} and later (until finally substituted by a higher expulsion speed).

In what follows we show that $\frac{T_s^*}{T(S')} = O(D')$. Note that $T(S') = \sum_{j=2}^{m} \frac{i'_j - i'_{j-1}}{v_{i'_{j-1}}}$. On the other hand in the lazy strategy robot R will search $i'_2 - i'_1 + \mathbf{1}$ the most central borders with the speed $v_{i'_1}$, then it will search through $i'_j - i'_{j-1}$ borders with speed $v_{i'_{j-s}}$, for any $j = 3, \dots m$, and finally the last $i'_m - i'_{m-1}$ borders with speed $v_{i'_{m-1}}$. The size of each border is not larger than $8 \cdot D$, thus we can estimate T_s^* from above by $\sum_{j=2}^{m} \frac{(i'_j - i'_{j-1}) \cdot 8D'}{v_{i'_{j-1}}} + \frac{8}{v_{i'_1}}$. Comparing the two complexities we note that in the summations every term in T_s^* is larger at most $8D'$ times, where $D' \leq D$. The only extra (positive) cost in T_s^* refers to the term $\frac{8}{v_{i'_1}}$. However, since robot R has to walk at least distance 1 along P with the speed $v_s = v_{i'_1}$ we obtain a good amortisation and finally conclude with the following theorem.

Theorem 7 *The spiral search strategy admits asymptotically optimal solution with $O(D)$-competitive factor.*

4 Conclusion

In this paper we considered the search problem with increasing speeds for models with local and global knowledge. Several problems remain open. This includes computation of more accurate asymptotic bound (beyond *Big-O*) notation of the competitive factor in the solution

based on the spiral strategy. Another unanswered question refers to faster computation of the best routes from points in P to target t when the number of distinct expulsion speeds can be linear in n. Finally, one could also consider the case with points in P moving along known or unknown trajectories.

Acknowledgements

The authors would like to thank Jurek Czyzowicz for early discussions on the studied problem and the anonymous reviewers for a number of corrections and suggestions which helped us to improve the presentation.

References

[1] S. Alpern, V. Baston, and S. Essegaier. Rendezvous search on a graph. *J. Applied Probability* **36(1)**, pp. 223–231, 1999.

[2] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous*. Kluwer Academic Publishing, Dordrecht, 2003.

[3] T. Asano, S. Bereg, and D.G. Kirkpatrick. Finding nearest larger neighbors. *Efficient Algorithms*, 2009, pp. 249–260.

[4] R.A. Baeza-Yates , J.C. Culberson, and G.J.E. Rawlins. Searching with uncertainty. *Proc. SWAT 88: 1st Scandinavian workshop on algorithm theory*, **318**, pp. 176–189, 1988.

[5] R.A. Baeza-Yates , J.C. Culberson, and G.J.E. Rawlins. Searching in the plane. *Information and Computation* **106 (2)**, pp. 234–252, 1993.

[6] R.A. Baeza-Yates and R. Schott. Parallel searching in the plane. *Computational Geometric Theory and Applications* **5(3)**, pp. 143–154, 1995.

[7] E. Bampas, J. Czyzowicz, L. Gąsieniec, D. Ilcinkas, R. Klasing, T. Kociumaka, and D. Pająk, Linear Search by a Pair of Distinct-Speed Robots. *SIROCCO 2016*, pp. 195-211.

[8] R. Bellman. Minimization problem. *Bull. AMS* **62(3)**, p. 270, 1956.

[9] M.A. Bender, A. Fernández, D. Ron, A. Sahai, and S.P. Vadhan. The power of a pebble: Exploring and mapping directed graphs. *STOC'98*, pp. 269–278.

[10] P. Bose, J.-L. De Carufel, and S. Durocher. Revisiting the problem of searching a line. *ESA 2013*, pp. 205–216.

[11] M. Chrobak, L. Gąsieniec, T. Gorry, and R. Martin: Group Search on the Line. *SOFSEM 2015*, pp. 164-176.

[12] A. Collins, J. Czyzowicz, L. Gąsieniec, A. Labourel. Tell me where I am so I can meet you sooner. *ICALP 2010*, pp. 502–514.

[13] J. Czyzowicz, L. Gąsieniec, K. Georgiou, E. Kranakis, F MacQuarrie, The Beachcombers' Problem: Walking and searching with mobile robots. *Theoretical Computer Science* **608**, pp. 201–218, 2015.

[14] J. Czyzowicz, L. Gąsieniec, A. Kosowski, E. Kranakis, Boundary Patrolling by Mobile Agents with Distinct Maximal Speeds, *ESA 2011*, pp. 701–712.

[15] O. Devillers, Improved incremental randomized Delaunay triangulation. *Symposium on Computational Geometry* pp. 106–115, 1998.

[16] Y. Dieudonné, A. Pelc. Anonymous meeting in networks. *SODA 2013*, pp. 737–747.

[17] S. Fortune, A Sweepline Algorithm for Voronoi Diagrams. *Algorithmica* **2**, pp. 153–174, 1987.

[18] S.K. Ghosh and R. Klein. Online algorithms for searching and exploration in the plane. *Computer Science Review* **4(4)**, pp. 189–201, 2010.

[19] M.I. Karavelas and M. Yvinec, Dynamic Additively Weighted Voronoi Diagrams in 2D. *ESA'02*, pp. 586–598.

[20] M. Hammar, B.J. Nilsson, and S. Schuierer. Parallel searching on m rays. *Comput. Geom.* **18(3)**, pp. 125–139, 2001.

[21] A. Jeż, and J. Łopuszański. On the two-dimensional cow search problem. *Information Processing Letters* **131(11)**, pp. 543–547, 2009.

[22] 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* **109(1)**, pp. 63–79, 1996.

[23] A. Kawamura and Y. Kobayashi, Fence patrolling by mobile agents with distinct speeds. *Distributed Computing* **28(2)**, pp. 147–154, 2015.

[24] D.G. Kirkpatrick, Optimal Search in Planar Subdivisions. *SIAM J. Computing* **12(1)** pp. 28–35, 1983.

[25] E. Koutsoupias, C.H. Papadimitriou, and M. Yannakakis. Searching a fixed graph. *ICALP'96*, pp. 280–289.

[26] H. Li and K P. Chong. Search on lines and graphs. *Proc. 48th IEEE Conference on Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. (CDC/CCC 2009)* **109(11)**, pp. 5780–5785.

[27] D. Shortt, Gravity assist. *www.planetary.org*, September 27, 2013.

[28] T. Temple and E. Frazzoli. Whittle-indexability of the cow path problem. *American Control Conference (ACC), 2010*, pp. 4152–4158.