Beachcombing on Strips and Islands[⋆]

Evangelos Bampas[1], Jurek Czyzowicz[2], David Ilcinkas[1], and Ralf Klasing[1]

[1] LaBRI, CNRS and University of Bordeaux, France
{evangelos.bampas,david.ilcinkas,ralf.klasing}@labri.fr
[2] Département d'informatique, Université du Québec en Outaouais, Canada
Jurek.Czyzowicz@uqo.ca

Abstract. A group of mobile robots (beachcombers) have to search collectively every point of a given domain. At any given moment, each robot can be in *walking mode* or in *searching mode*. It is assumed that each robot's maximum allowed searching speed is strictly smaller than its maximum allowed walking speed. A point of the domain is searched if at least one of the robots visits it in searching mode. The Beachcombers' Problem consists in developing efficient *schedules* (algorithms) for the robots which collectively search all the points of the given domain as fast as possible.

We first consider the *online* Beachcombers' Problem, where the robots are initially collocated at the origin of a semi-infinite line. It is sought to design a schedule A with maximum *speed* S, defined as $S = \inf_\ell \frac{\ell}{t_A(\ell)}$, where $t_A(\ell)$ denotes the time when the search of the segment $[0, \ell]$ is completed under A. We consider a *discrete* and a *continuous* version of the problem, depending on whether the infimum is taken over $\ell \in \mathbb{N}^*$ or $\ell \geq 1$. We prove that the `LeapFrog` algorithm, which was proposed in [Czyzowicz et al., SIROCCO 2014, LNCS 8576, pp. 23–36 (2014)], is in fact optimal in the discrete case. This settles in the affirmative a conjecture from that paper. We also show how to extend this result to the more general continuous online setting.

For the *offline* version of the Beachcombers' Problem, we consider the *single-source* Beachcombers' Problem on the cycle, as well as the *multi-source* Beachcombers' Problem on the cycle and on the finite segment. For the *single-source* Beachcombers' Problem on the cycle, we show that the structure of the optimal solutions is identical to the structure of the optimal solutions to the two-source Beachcombers' Problem on a finite segment. In consequence, by using results from [Czyzowicz et al., ALGOSENSORS 2014, LNCS 8847, pp. 3–21 (2014)], we prove that the single-source Beachcombers' Problem on the cycle is NP-hard, and we derive approximation algorithms for the problem. For the *multi-source* variant of the Beachcombers' Problem on the cycle and on the finite segment, we obtain efficient approximation algorithms.

[⋆] Part of this work was done while Jurek Czyzowicz was visiting the LaBRI as a guest professor of the University of Bordeaux. This work was partially funded by the ANR project DISPLEXITY (ANR-11-BS02-014). This study has been carried out in the frame of "the Investments for the future" Programme IdEx Bordeaux – CPU (ANR-10-IDEX-03-02).

One important contribution of our work is that, in all variants of the offline Beachcombers' Problem that we discuss, we allow the robots to *change direction of movement* and search points of the domain on both sides of their respective starting positions. This represents a significant generalization compared to the model considered in [Czyzowicz et al., ALGOSENSORS 2014, LNCS 8847, pp. 3–21 (2014)], in which each robot had a fixed direction of movement that was specified as part of the solution to the problem. We manage to prove that changes of direction do not help the robots achieve optimality.

1 Introduction

A group of n mobile robots have to explore collectively a given one-dimensional domain. The robots may be initially collocated or dispersed in the domain. At every moment of time, a robot can be either in *walking mode* or in *searching mode.* A robot in walking mode traverses the domain with a speed not exceeding its maximal *walking speed.* A robot in searching state can travel using at most its maximal *searching speed*, which is strictly smaller than its walking speed, reflecting the fact that a searching activity is more time-consuming. Different robots may have distinct maximal walking and searching speeds. A robot can change mode, speed, and direction of movement instantaneously. There is no communication between the robots during the execution of the algorithm. In the Beachcombers' Problem, the goal is to design a schedule for the movement of all robots so that the domain is *searched* as fast as possible. A domain is said to be searched under a given schedule, if every point of the domain is visited by at least one robot in searching mode.

As pointed out in [11], where the Beachcombers' Problem was introduced, there are numerous examples in quite diverse domains in which exploration using *two-speed* robots arises as a natural model for the underlying processes. For example, *foraging* or *harvesting* a field may take longer than inadvertent walking. In computer science, *web page indexing* or *code inspection* require a more involved investigation. A common feature of these examples is that the activity of searching, or other action to be performed on the territory, takes more time than casual territory traversal. The analogy to *beachcombers* has been introduced in [11] to bring out that, e.g., a beachcomber looking for things of value performs a meticulous search of the beach, which takes significantly more time than simply walking from one point of the beach to another. Further motivation for the two-speed model can be found in [11, 12].

Preliminaries and notation. We consider searching schedules using two-speed robots in the following one-dimensional geometric domains: the cycle of a known circumference L, the finite straight line segment of a known length L, and on the semi-infinite line $[0; +\infty)$. The efficiency of the search in the first two cases is expressed in terms of the time t_f when the search of the cycle is completed or, equivalently, the speed L/t_f of the process. However, in the latter case, the schedule efficiency is better expressed by the speed of the search, represented by

$\inf_\ell \frac{\ell}{t_A(\ell)}$ where $t_A(\ell)$ denotes the time when the search of the segment $[0;\ell]$ is completed. In the *discrete* version of the problem, the infimum $\inf_\ell \frac{\ell}{t_A(\ell)}$ is over $\ell \in \mathbb{N}^*$. On the other hand, in the *continuous* setting, the infimum $\inf_\ell \frac{\ell}{t_A(\ell)}$ is taken over $\ell \geq 1$.

A *schedule* for the robots is defined by a strictly increasing sequence of times $t_0, t_1, \cdots$, as well as, for every robot i and every interval $[t_j, t_{j+1}]$, for $j \geq 0$, a *mode* (walking or searching), a *speed* (respecting the maximum speed of the chosen mode), and a *direction* of movement. A schedule is *correct* if, for every point p of the domain, there exists a time moment at which p is visited by a robot in searching mode. For any fixed robot i, we refer to the individual schedule of robot i as the *trajectory* of robot i. Clearly, this sequence of intervals is finite in the offline case and infinite in the online setting.

Observe that while the model allows to use any speed not exceeding the maximal speed given for the robot's mode, we can restrict consideration only to using its maximal searching and walking speeds. Also notice that any searching schedule may be converted to another one, which has the property that all sub-segments which were being searched have pairwise disjoint interiors. Therefore, when looking for the optimal searching schedule, it is sufficient to restrict consideration to schedules whose searched sub-segments may only intersect at their endpoints.

Previous work. The Beachcombers' Problem was introduced and studied in [11]. An optimal (offline) algorithm was presented for the problem in which all robots are initially located on one endpoint of a finite segment of known length. Furthermore, a 2-competitive (online) algorithm was presented for the case where all robots are initally collocated on the origin of a semi-infinite line. In [12], the Beachcombers' Problem was studied for the case of more than one starting positions on a finite segment of known length. For a fixed number $t \geq 2$ of starting positions, the t-source Beachcombers' Problem on a finite segment asks to find t starting points on the segment, an assignment of the robots to the starting points, and a search schedule which concludes the search of the finite segment as quickly as possible. It was shown in [12] that this problem is NP-hard for $t = 2$, even when all robots have the same walking speed, that the optimal solution can be computed efficiently when all robots have the same searching speed, and that there exist a deterministic approximation algorithm for $t = 2$ and a randomized approximation algorithm for general t.

Our contributions. In Section 2, we study the online Beachcombers' Problem on the semi-line $[0; +\infty)$. We prove that the `LeapFrog` algorithm, which was proposed in [11], is in fact optimal in the discrete case. This settles in the affirmative a conjecture from [11]. We also show how to extend this result to the more general continuous online setting.

As regards the offline Beachcombers' Problem, we consider in Section 3 the *single-source* Beachcombers' Problem on the cycle. We show that the structure of the optimal solutions to the single-source Beachcombers' Problem on a cycle is identical to the structure of the optimal solutions to the two-source Beachcombers' Problem on a finite segment, as defined in [12]. This implies that the results from [12] for the case of two distinct sources are carried over to the case of the cycle, yielding an NP-hardness result as well as the existence of efficient approximation algorithms for the problem. In particular, the NP-hardness of the single-source Beachcombers' Problem on the cycle seems at first somewhat surprising, in view of the existence of an efficient algorithm generating optimal schedules for the single-source problem on a finite segment.

Furthermore, in Section 4, we explain how to modify the arguments from Section 3 so as to obtain approximation algorithms for the *multi-source* variant of the Beachcombers' Problem on the cycle and on the finite segment. Our results for the cycle topology provide a partial answer to an open question posed in [11] and [12], concerning the study of the problem in different domain topologies.

One further important contribution of our work is that, in all variants of the offline Beachcombers' Problem that we discuss, we allow the robots to *change direction of movement* and search points of the domain on both sides of their respective starting positions. This represents a significant generalization compared to the model considered in [12], in which each robot had a fixed direction of movement that was specified as part of the solution to the problem. On an intuitive level, allowing the robots to zigzag should not result in a faster schedule. However, no proof of this intuition had been found until now. We manage to *prove* that changes of direction do not help the robots achieve optimality.

Due to lack of space, proofs are omitted.

Related work. Searching and exploration have been studied in numerous papers considering graphs or geometric environments (e.g. [1, 4, 5, 7, 8, 14, 17, 18, 16, 21]). The performance of the searching or exploration is typically expressed by the trajectory length or the time used by the mobile agent.

Many searching and exploration algorithms are studied in the *online* setting, i.e., the target position or sometimes other parameters of the environment are *a priori* unknown (cf. [2, 3, 9, 14, 16, 19, 20]). Efficiency of such algorithms is typically measured by the *competitive ratio*, i.e., the ratio of the time spent by the online algorithm with respect to the time of the optimal offline algorithm.

Most of the papers studying searching and exploration concern single robots. Sets of collaborating mobile robots were studied, e.g., in [10, 15, 22, 23]. Tradeoffs between the number of robots and the time of exploration were derived in [19].

The majority of the research on mobile robots concerns robots having the same mobile speed. Robots with distinct speeds were considered in the context of sensor energy efficiency [25], for designing fast converging population protocols [6], and for patrolling the boundary of an environment [13, 24].

2 The Online Beachcombers' Problem on the Semi-Line

In this section, we consider two variants of the online beachcombers' problem on the semi-line. The first one corresponds to the online problem presented in [11].

Definition 1 (Discrete Online Beachcombers' problem). *Given n robots with walking speeds w_i and searching speeds $s_i < w_i$, for $1 \leq i \leq n$, initially collocated at the origin of a semi-line $[0; +\infty)$, the problem consists in finding a correct schedule for this semi-line. The discrete online speed of a schedule A is defined as $\inf_{\ell \in \mathbb{N}^*} \frac{\ell}{t_A(\ell)}$, where $t_A(\ell)$ denotes the time when the search of the segment $[0; \ell]$ is completed.*

Definition 2 (Continuous Online Beachcombers' problem). *Given n robots with walking speeds w_i and searching speeds $s_i < w_i$, for $1 \leq i \leq n$, initially collocated at the origin of a semi-line $[0; +\infty)$, the problem consists in finding a correct schedule for this semi-line. The continuous online speed of a schedule A is defined as $\inf_{\ell \geq 1} \frac{\ell}{t_A(\ell)}$, where $t_A(\ell)$ denotes the time when the search of the segment $[0; \ell]$ is completed.*

The idea of the `LeapFrog` algorithm is to make all sufficiently fast robots, forming the so-called swarm of the algorithm, meet at some regular intervals. For this purpose, each robot of the swarm is assigned a specific fraction of such regular interval that it has to search (the robot walks the rest of the interval). For each robot, the assigned searching subinterval is calculated as a function of the walking and searching speeds of all the robots participating in the swarm. The robots repeat the same behavior in each interval, always all of them meeting at its extremities. Although all robots are always used in the optimal offline algorithm presented in [11], some robots whose walking speeds are too slow (informally, not larger than the average speed of the swarm) may not participate in the swarm and thus may be never used in the online `LeapFrog` algorithm.

The main purpose of this section is to prove the optimality of the Algorithm `LeapFrog` described in [11]. Our first step toward this goal is to restrict ourselves to particular schedules, which are much simpler to analyze but are nevertheless at least as efficient (in terms of online speeds) as general ones. The following simple lemma holds both for the discrete and the continuous cases.

Lemma 1. *For every correct schedule S, there exists a correct schedule S' whose both online speeds are not smaller than the respective ones of S, and such that every moving agent always moves in the initial direction at the full speed permitted by its current mode. Moreover, the interiors of the segments searched by the different robots do not overlap.*

Let LF be the discrete online speed of Algorithm `LeapFrog`. Before proving that Algorithm `LeapFrog` is optimal in terms of *discrete* online speed, we prove the slightly weaker result that no correct schedule can have a *continuous* online speed larger than LF.

Lemma 2. *The* continuous *online speed of any correct schedule is at most* LF*.*

Theorem 1. *The* discrete *online speed of any correct schedule is at most* LF*.*

Concerning the continuous online speed metrics, it is possible to obtain a slightly more precise result than the one of Lemma 2.

Lemma 3. *If there are at least two robots and Algorithm* `LeapFrog` *uses all the robots, then any continuous online speed of any correct schedule is less than* LF.

It turns out that simple variations of Algorithm `LeapFrog` can match the bounds given in Lemmas 2 and 3.

In Algorithm `LeapFrog`, all agents participating in the swarm are synchronized at every integer point, that is, they all arrive at the same time at every integer point. For any positive integer N, we denote by $\texttt{LeapFrog}_N$ the variant of `LeapFrog` for which the agents participating in the swarm synchronize every $1/N$ units of distance, instead of every unit as in the original Algorithm `LeapFrog`. It is easy to check that the continuous online speed of Algorithm $\texttt{LeapFrog}_N$ tends to LF as N tends toward infinity. The family of algorithms $\{\texttt{LeapFrog}_N\}_{N\in\mathbb{N}^*}$ is thus optimal in the case when there at least two robots and Algorithm `LeapFrog` uses all the robots (cf. Lemma 3).

If there is only one robot, then the only reasonable algorithm is the one in which the single robot always searches at its maximal speed. This algorithm is in fact Algorithm `LeapFrog`, and its continuous online speed is equal, in this special case, to its discrete online speed LF. Lemma 2 thus shows that Algorithm `LeapFrog` is optimal also in this case.

The remaining case is when there are at least two robots, but the swarm of Algorithm `LeapFrog` does not use all the robots. In this particular case, we consider the following adaptation $\texttt{LeapFrog}'$ of Algorithm `LeapFrog`. Let r, with searching speed s, be some robot not participating in the swarm in Algorithm `LeapFrog`. In our adaptation $\texttt{LeapFrog}'$, this robot r searches the semi-line from its beginning at its maximum searching speed s during $1/\text{LF}$ time units before stopping forever. Let p be the point at which r stops. All the robots of the swarm walk at the walking speed of the slowest walker among them until reaching point p. (Note that this walking speed is larger than LF by construction of the swarm.) At this point, all swarm robots execute Algorithm $\texttt{LeapFrog}_N$ as if p was the origin of the semi-line, with N defined as follows. The integer N is chosen sufficiently large so that, at any time at least $1/\text{LF}$, the swarm has always searched one segment of length $1/N$ ahead of the normal Algorithm $\texttt{LeapFrog}_N$. One can prove that the continuous online speed of Algorithm $\texttt{LeapFrog}'$ is equal to LF, which is optimal by Lemma 2.

3 Single-Source Beachcombers on the Cycle

The purpose of this section is to show that the structure of the optimal solutions to the offline Beachcombers' Problem on the cycle is identical to the structure of the optimal solutions to the two-source Beachcombers' Problem on a finite segment, as defined in [12]. This implies that the results from [12] for the case of two distinct sources are carried over to the case of the cycle, even if the agents are allowed to zigzag. The (offline) single-source Beachcombers' Problem on the cycle is defined as follows:

Definition 3 (BPC – **Beachcombers' Problem on the Cycle).** *Consider a cycle C_L of circumference L and n robots $r_1, r_2, \ldots, r_n$, initially placed at point*

0 *of the cycle, each robot* r_i *having searching speed* s_i *and walking speed* w_i*, such that* $s_i < w_i$*. The Beachcombers' Problem consists in finding an efficient correct searching schedule* $\mathcal{A}$ *of* C_L*. The speed* $S_{\mathcal{A}}$ *of the solution to the Beachcombers' Problem equals* $S_{\mathcal{A}} = L/t_f$*, where* t_f *is the finishing time of* $\mathcal{A}$*.*

The (offline) t-source Beachcombers' Problem on the segment was defined in [12] as follows:

Definition 4 (t-SBP – t-Source Beachcombers' Problem [12]). *Consider an interval* $I_L = [0, L]$ *and* n *robots* $r_1, \ldots, r_n$*, each robot* r_i *having searching speed* s_i *and walking speed* w_i*, such that* $s_i < w_i$*. The* t*-Source Beachcombers' Problem consists in finding an efficient correct searching schedule* $\mathcal{A}$ *of* I_L*, in which the robots are divided into at most* t *groups with each group being initially placed on a particular point of the segment (the source) and having a fixed direction of movement. The speed* $S_{\mathcal{A}}$ *of the solution to the Beachcombers' Problem equals* $S_{\mathcal{A}} = L/t_f$*, where* t_f *is the finishing time of* $\mathcal{A}$*.*

Note that the model of [12] precludes by definition any change of direction of movement for the robots, since each group of robots has a fixed direction of movement which is specified as part of the solution to the t-SBP problem. On the other hand, in our model for BPC, no such restriction is imposed but we are able to *prove* that changing directions does not help the robots.

In the following propositions and lemmas, we will refer to schedules for BPC, unless it is explicitly stated otherwise.

Proposition 1. *For every correct schedule* $\mathcal{S}$*, there exists a correct schedule* $\mathcal{S}'$ *whose completion time is not greater than that of* $\mathcal{S}$ *and which additionally satisfies the following properties:*

1. *Every pair of arcs searched by the robots under* $\mathcal{S}'$ *have disjoint interiors.*
2. *During every time interval of* $\mathcal{S}'$*, every robot* i *is either stopped or it moves at the maximum speed* w_i *or* s_i*, according to its chosen mode during that interval.*

In view of Proposition 1, we will assume in the following that the trajectory of each robot i is characterized by a sequence of arcs $(\mathcal{A}_{i,j})_{0 \le j \le \sigma_i}$ and, for each arc, a mode (searching or walking) and a direction (clockwise or counterclockwise), such that in each arc the robot is moving at the maximum allowed speed. Note that an arc $\mathcal{A}_{i,j}$ may correspond to one or more consecutive time intervals of the schedule.

Lemma 4. *For every correct schedule* $\mathcal{S}$*, there exists a correct schedule* $\mathcal{S}'$ *whose completion time is not greater than that of* $\mathcal{S}$ *and in which the trajectory of every robot in* $\mathcal{S}'$ *satisfies the following:*

– *It either stops at the origin at time* 0*, or it searches a sequence of arcs in clockwise (resp. counterclockwise) direction, in order of increasing clockwise (resp. counterclockwise) distance from the origin, and then it either stops or* *it moves counterclockwise (resp. clockwise) to the origin and then searches a sequence of arcs in counterclockwise (resp. clockwise) direction, in order of increasing counterclockwise (resp. clockwise) distance from the origin.*

- *In between arcs that the robot searches clockwise (resp. counterclockwise), it walks clockwise (resp. counterclockwise) straight from the end of the last searched arc to the beginning of the next one.*
- *The robot stops at the moment when it searches a non-empty arc for the last time.*
- *Traversing the circle clockwise from the origin, we first encounter all the arcs that are searched by the robot in the clockwise direction and, subsequently, we encounter all the arcs that are searched by the robot in the counterclockwise direction.*

Lemma 5. *For every correct schedule $\mathcal{S}$, there exists a correct schedule $\mathcal{S}'$ whose completion time is not greater than that of $\mathcal{S}$ and in which, while moving from the origin in a clockwise direction, one first encounters all the arcs that are searched by some robot moving in clockwise direction under $\mathcal{S}'$, and then one encounters all the arcs searched by some robot moving in counterclockwise direction under $\mathcal{S}'$.*

We call a schedule that satisfies the properties guaranteed by Proposition 1, Lemma 4, and Lemma 5 *normal*:

Definition 5 (Normal schedules). *A schedule is called* normal *if every robot's trajectory is either empty (the robot stops at time* 0*), or it consists of one clockwise or counterclockwise* leg*, as defined below, or it consists of two legs in opposite directions, such that after the first leg the robot returns to the origin by walking at full speed backwards over the first leg.*

A clockwise (resp. counterclockwise) leg *is a part of a robot's trajectory that starts at the origin and consists of searching at full speed a sequence of arcs in order of increasing clockwise (resp. counterclockwise) distance from the origin. In between searched arcs, the robot walks at full speed in the clockwise (resp. counterclockwise) direction from the end of the last searched arc to the beginning of the next one.*

In addition, a normal schedule satisfies the following properties:

1. *Every pair of searched arcs (not necessarily by the same robot) have disjoint interiors.*
2. *For every robot, each of its legs corresponds to at most one loop around the circle and, if its trajectory has two legs, they do not overlap.*
3. *While moving from the origin in a clockwise direction, one first encounters all the searched arcs that belong to clockwise legs, and then one encounters all the searched arcs that belong to counterclockwise legs.*

It follows from the proofs of Proposition 1, Lemma 4, and Lemma 5 that, for every correct schedule $\mathcal{S}$ that is not normal, there exists a correct normal schedule $\mathcal{S}'$ that has smaller or equal completion time. In other words, we can guarantee all of the properties ensured by Proposition 1, Lemma 4, and Lemma 5 simultaneously. In the following, we will assume normal schedules without loss of generality. In fact, a careful examination of the proofs reveals that, in all cases, the modification of $\mathcal{S}$ to $\mathcal{S}'$ strictly decreases the completion time of at least one robot. This is less obvious in Lemma 5, but it suffices to apply the modification described in the proof for a pair of arcs a, b, such that one of them is the last searched arc in some robot's clockwise leg or the last searched arc in some robot's counterclockwise leg. It is easy to check that if $\mathcal{S}$ does not satisfy the property in the statement of Lemma 5, then there exists at least one such pair of searched arcs. We thus have the following:

Lemma 6. *For every non-normal correct schedule $\mathcal{S}$, there exists a normal correct schedule $\mathcal{S}'$ whose completion time is not greater than that of $\mathcal{S}$ and in which at least one robot requires strictly less time to complete its trajectory.*

With every fixed normal schedule $\mathcal{S}$, we associate the corresponding partition of the circle into pairwise interior-disjoint arcs, each of which is searched by a single robot that is moving in the same direction over a continuous time interval. In view of Lemma 4, we may assume that the origin is not in the interior of any of the arcs.

Definition 6. *Let $\mathcal{S}$ be a normal schedule. We denote by $\mathcal{A}^+_{\mathcal{S}}$ (resp. $\mathcal{A}^-_{\mathcal{S}}$) the set of searched arcs that belong to clockwise (resp. counterclockwise) legs of robots. For $a, b \in \mathcal{A}^+_{\mathcal{S}} \cup \mathcal{A}^-_{\mathcal{S}}$, we write $a \prec b$ if a clockwise traversal starting from the origin encounters arc a before arc b.*

For the purpose of stating the next lemma, given a normal schedule $\mathcal{S}$ with completion time T, we will denote by $\mathcal{I}(\mathcal{S})$ the inclusion-maximal set of searched arcs that satisfies the following property: Each arc in $\mathcal{I}(\mathcal{S})$ is searched by a robot that stops strictly earlier than T and $\bigcup_{I \in \mathcal{I}(\mathcal{S})} I$ is a continuous arc that contains the origin. We will denote by $R(\mathcal{S})$ the number of distinct robots that search the arcs in $\mathcal{I}(\mathcal{S})$.

Lemma 7. *Let $\mathcal{S}$ be a normal correct schedule with completion time T, such that there exists $\epsilon > 0$ and at least one robot that stops at time $T - \epsilon$. Then, there exists a normal correct schedule $\mathcal{S}'$ with completion time at most T and $R(\mathcal{S}') > R(\mathcal{S})$.*

Repeated applications of Lemma 7 yield Corollary 1, from which Corollaries 2 and 3 follow immediately:

Corollary 1. *In every optimal and normal schedule, all robots terminate their trajectories simultaneously.*

Corollary 2. *Every optimal schedule is normal.*

Corollary 3. *In every optimal schedule, the trajectory of each robot contains at least one leg.*

We are now ready to further restrict the structure of optimal schedules. We first show that each robot searches only one arc per leg (Lemma 8), then that there are no *crossing robots* (Lemma 9, cf. Definition 7), and then that each robot performs only one leg (Lemma 10).

Lemma 8. *In every optimal schedule, each leg of the trajectory of each robot contains exactly one searched arc.*

Definition 7 (Crossing robots). *Let $\mathcal{S}$ be a normal schedule. We say that a pair of robots i, j* cross *under $\mathcal{S}$ if robot i searches arcs $a_i^+ \in \mathcal{A}_\mathcal{S}^+$ and $a_i^- \in \mathcal{A}_\mathcal{S}^-$, robot j searches arcs $a_j^+ \in \mathcal{A}_\mathcal{S}^+$ and $a_j^- \in \mathcal{A}_\mathcal{S}^-$, and $a_i^+ \prec a_j^+ \prec a_i^- \prec a_j^-$.*

Lemma 9. *No optimal schedule contains a pair of crossing robots.*

Lemma 10. *In every optimal schedule, the trajectory of each robot contains exactly one leg.*

Lemma 10 is the main technical tool for connecting the optimal schedules for BPC instances to the optimal schedules for 2-SBP instances.

Lemma 11. *Let $\mathcal{I}$ be an instance of* 2-SBP *on an interval of length L and let $\mathcal{J}$ be an instance of* BPC *with the same set of robots on a circle of circumference L. The completion time of the optimal schedule is the same in both instances.*

We obtain now, as immediate corollaries of Lemma 11 and the results in [12], the NP-hardness of BPC, as well as the existence of deterministic and randomized approximation algorithms for BPC.

Theorem 2. BPC *is* NP*-hard, even when all robots have the same walking speed.*

Theorem 3. BPC *admits a 0.5568-approximation algorithm that runs in $\mathcal{O}(n \log n)$ time.*

Theorem 4. BPC *instances in which all robots have the same search speed can be solved optimally in time $\mathcal{O}(n \log n)$.*

Theorem 5. BPC *admits a randomized algorithm which achieves an expected approximation ratio of $\frac{3}{4}$, needs $\mathcal{O}(n)$ random bits, and runs in $\mathcal{O}(n \log n)$ time.*

4 Multi-Source Beachcombers on the Line and Cycle

We now leverage our techniques from the previous section to obtain results for the multi-source version of the beachcombers' problem on the line and on the cycle, while allowing changes of direction as in BPC (Definition 3). We define the problem t-$\mathsf{SBPL_z}$:

Definition 8 (t-$\mathsf{SBPL_z}$ – t-Source Beachcombers' Problem on the Line with zigzags). *Consider a line segment of length L and n robots $r_1, \ldots, r_n$, each robot r_i having searching speed s_i and walking speed $w_i > s_i$. Find an efficient correct searching schedule $\mathcal{A}$ of the segment, in which the robots are divided into at most t groups with each group being initially placed on a particular point of the segment (the source). The speed $S_{\mathcal{A}}$ of the solution equals $S_{\mathcal{A}} = L/t_f$, where t_f is the finishing time of $\mathcal{A}$.*

Similarly, we define t-$\mathsf{SBPC_z}$ (t-Source Beachcombers' Problem on the Cycle with zigzags), where, instead of a segment of length L, the robots have to search a cycle of circumference L. Note that, in contrast to t-SBP (Definition 4), the robots are allowed to change direction of movement and, in particular, to search segments on both sides of their respective starting points.

By following the arguments for BPC from Section 3 and modifying the proofs as necessary, we can prove that, for a fixed choice of starting points and a fixed allocation of the robots to those starting points, the equivalents of Lemma 6 and Corollaries 1–3 hold for t-$\mathsf{SBPL_z}$ as well, with the only difference that a *normal schedule* for t-$\mathsf{SBPL_z}$ is defined as follows:

Definition 9 (Normal schedules for t-$\mathsf{SBPL_z}$). *Given a fixed choice of t starting points and a fixed allocation of the robots to those starting points, a schedule for t-$\mathsf{SBPL_z}$ is called* normal *if every robot's trajectory is either empty (the robot stops at time 0), or it consists of one leftward or rightward* leg, *as defined below, or it consists of two legs in opposite directions, such that after the first leg the robot returns to the origin by walking at full speed backwards over the first leg.*

A rightward (resp. leftward) leg *is a part of a robot's trajectory that starts at its assigned source and consists of searching at full speed a sequence of segments toward the right (resp. left) in order of increasing distance from the source. In between searched segments, the robot walks at full speed to the right (resp. left) from the end of the last searched arc to the beginning of the next one.*

In addition, a normal schedule satisfies the following properties:

1. *Every pair of searched segments (not necessarily by the same robot) have disjoint interiors.*
2. *The given segment of length L is partitioned into t regions, such that each region is associated with exactly one starting point, and the robots originating from the associated starting point are confined within that region.*

Subsequently, we obtain the following lemma, which corresponds to Lemma 10.

Lemma 12. *Given a fixed choice of t starting points and a fixed allocation of the robots to those starting points, in every optimal t-$\mathsf{SBPL_z}$ schedule, the trajectory of each robot contains exactly one leg.*

The following Lemma connects the optimal schedules for t-$\mathsf{SBPL_z}$ instances to the optimal schedules for $2t$-SBP instances.

Lemma 13. *Let $\mathcal{I}$ be an instance of $2t$-SBP on a segment of length L and let $\mathcal{J}$ be an instance of t-$\mathsf{SBPL_z}$ with the same set of robots on a segment of length L. The completion time of the optimal schedule is the same in both instances.*

We thus obtain, in view of the results for t-SBP [12], the following results for t-$\mathsf{SBPL_z}$:

Theorem 6. *t-$\mathsf{SBPL_z}$ instances in which all robots have the same search speed can be solved optimally in time $\mathcal{O}(n \log n)$.*

Theorem 7. *t-$\mathsf{SBPL_z}$ admits a randomized algorithm which achieves an expected approximation ratio of $1 - \left(1 - \frac{1}{2t}\right)^{2t}$, needs $\mathcal{O}(n \log t)$ random bits, and runs in $\mathcal{O}(n \log n)$ time.*

Finally, we prove that the optimal solution to a t-$\mathsf{SBPC_z}$ instance with a given swarm on a cycle of circumference L shares its structure with the optimal solution to a t-$\mathsf{SBPL_z}$ instance with the same swarm on a segment of length L. Indeed, the normal schedules for t-$\mathsf{SBPC_z}$ are essentially the same as those for t-$\mathsf{SBPL_z}$, except that the region associated with each starting point is now an arc of the cycle.

Lemma 14. *Given a fixed choice of t starting points and a fixed allocation of the robots to those starting points, in every optimal t-$\mathsf{SBPC_z}$ schedule, the trajectory of each robot contains exactly one leg.*

Lemma 15. *Let $\mathcal{I}$ be an instance of t-$\mathsf{SBPL_z}$ on a segment of length L and let $\mathcal{J}$ be an instance of t-$\mathsf{SBPC_z}$ with the same set of robots on a cycle of circumference L. The completion time of the optimal schedule is the same in both instances.*

In view of Theorems 6 and 7, we obtain the following results for t-$\mathsf{SBPC_z}$:

Theorem 8. *t-$\mathsf{SBPC_z}$ instances in which all robots have the same search speed can be solved optimally in time $\mathcal{O}(n \log n)$.*

Theorem 9. *t-$\mathsf{SBPC_z}$ admits a randomized algorithm which achieves an expected approximation ratio of $1 - \left(1 - \frac{1}{2t}\right)^{2t}$, needs $\mathcal{O}(n \log t)$ random bits, and runs in $\mathcal{O}(n \log n)$ time.*

5 Concluding Remarks

There are several directions in which the study of the search and exploration using two-speed robots may continue. An obvious one is to improve the approximation ratio for the versions of the problem that are NP-hard. In this respect, we should investigate whether zigzags may help to obtain approximate solutions, at least for particular combinations of searching and walking speeds of the robots (note that we know from the present paper that zigzags never help to obtain *optimal* solutions). Another direction is to study the configurations of robots' speeds and/or environments for which optimal solutions can be computed efficiently. Finally, it is worthwhile to consider different and more general search domains, such as non-simple closed or open curves.

References

1. Albers, S., Henzinger, M.R.: Exploring unknown environments. SIAM J. Comput. 29(4), 1164–1188 (2000)
2. Albers, S.: Online algorithms: a survey. Math. Program. 97(1-2), 3–26 (2003)
3. Albers, S., Schmelzer, S.: Online algorithms - what is it worth to know the future? In: Algorithms Unplugged, pp. 361–366. Springer (2011)
4. Alpern, S., Gal, S.: The theory of search games and rendezvous, vol. 55. Kluwer Academic Publishers (2002)
5. Baeza-Yates, R.A., Culberson, J.C., Rawlins, G.J.E.: Searching in the plane. Information and Computation 106, 234–234 (1993)
6. Beauquier, J., Burman, J., Clement, J., Kutten, S.: On utilizing speed in networks of mobile agents. In: ACM SIGACT-SIGOPS 2010, pp. 305–314. ACM (2010)
7. Beck, A.: on the linear search problem. Israel Journal of Mathematics 2(4), 221–228 (1964)
8. Bellman, R.: An optimal search problem. Bull. Am. Math. Soc. p. 270 (1963)
9. Berman, P.: On-line searching and navigation. In: Fiat, A., Woeginger, G. (eds.) Online Algorithms The State of the Art, pp. 232–241. Springer (1998)
10. Chalopin, J., Flocchini, P., Mans, B., Santoro, N.: Network exploration by silent and oblivious robots. In: Graph-Theoretic Concepts in Computer Science, WG 2010, LNCS 6410, pp. 208–219. Springer (2010)
11. Czyzowicz, J., Gasieniec, L., Georgiou, K., Kranakis, E., MacQuarrie, F.: The beachcombers' problem: Walking and searching with mobile robots. In: Structural Information and Communication Complexity, SIROCCO 2014, LNCS 8576, pp. 23–36. Springer (2014)
12. Czyzowicz, J., Gasieniec, L., Georgiou, K., Kranakis, E., MacQuarrie, F.: The multi-source beachcombers' problem. In: Algorithms for Sensor Systems, ALGOSENSORS 2014, Revised Selected Papers, LNCS 8847, pp. 3–21. Springer (2014)
13. Czyzowicz, J., Gasieniec, L., Kosowski, A., Kranakis, E.: Boundary patrolling by mobile agents with distinct maximal speeds. In: Algorithms, ESA 2011, LNCS 6942, pp. 701–712. Springer (2011)
14. Czyzowicz, J., Ilcinkas, D., Labourel, A., Pelc, A.: Worst-case optimal exploration of terrains with obstacles. Inf. Comput. 225, 16–28 (2013)
15. Das, S., Flocchini, P., Kutten, S., Nayak, A., Santoro, N.: Map construction of unknown graphs by multiple agents. Theor. Comput. Sci. 385(1-3), 34–48 (2007)
16. Demaine, E.D., Fekete, S.P., Gal, S.: Online searching with turn cost. Theoretical Computer Science 361(2), 342–355 (2006)
17. Deng, X., Papadimitriou, C.H.: Exploring an unknown graph. In: Foundations of Computer Science, FOCS 1990, pp. 355–361. IEEE (1990)
18. Deng, X., Kameda, T., Papadimitriou, C.H.: How to learn an unknown environment (extended abstract). In: Foundations of Computer Science, FOCS 1991, pp. 298–303. IEEE (1991)
19. Dereniowski, D., Disser, Y., Kosowski, A., Pajak, D., Uznanski, P.: Fast collaborative graph exploration. In: Automata, Languages, and Programming, ICALP 2013, LNCS 7966, pp. 520–532. Springer (2013)
20. Fleischer, R., Kamphans, T., Klein, R., Langetepe, E., Trippen, G.: Competitive online approximation of the optimal search ratio. SIAM J. Comput. 38(3), 881–898 (2008)
21. Fomin, F.V., Thilikos, D.M.: An annotated bibliography on guaranteed graph searching. Theor. Comput. Sci. 399(3), 236–245 (2008)

22. Fraigniaud, P., Gasieniec, L., Kowalski, D.R., Pelc, A.: Collective tree exploration. Networks 48(3), 166–177 (2006)
23. Higashikawa, Y., Katoh, N., Langerman, S., Tanigawa, S.: Online graph exploration algorithms for cycles and trees by multiple searchers. J. Comb. Optim. (2012)
24. Kawamura, A., Kobayashi, Y.: Fence patrolling by mobile agents with distinct speeds. In: Algorithms and Computation, ISAAC 2012, LNCS 7676, pp. 598–608. Springer (2012)
25. Wang, G., Irwin, M.J., Fu, H., Berman, P., Zhang, W., Porta, T.L.: Optimizing sensor movement planning for energy efficiency. ACM Transactions on Sensor Networks 7(4), 33 (2011)