How to Meet at a Node of Any Connected Graph

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

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

Abstract

Two mobile agents have to meet at the same node of a connected graph with unlabeled nodes. This intensely researched task is known as rendezvous. The adversary assigns the agents different starting nodes in the graph and different integer labels from a set $\{1, \ldots, L\}$. Time is slotted in synchronous rounds. The adversary wakes up the agents in possibly different rounds. After wakeup, the agents move as follows. In each round, an agent can either stay idle or move to an adjacent node. Each agent knows its label but not the label of the other agent, and agents have no a priori information about the graph. They do not know L. They execute the same deterministic algorithm whose parameter is the agent's label. The time of a rendezvous algorithm is the worst-case number of rounds since the wakeup of the earlier agent till the meeting.

In most of the results concerning rendezvous in graphs, the graph is finite and rendezvous relies on the exploration of the entire graph. Thus the time of rendezvous depends on the size of the graph. This approach is inefficient for very large graphs, and cannot be used for infinite graphs. For such graphs it is natural to seek rendezvous algorithms whose time depends on the initial distance D between the agents. In this paper we adopt this approach and consider rendezvous in arbitrary connected graphs with nodes of finite degrees, and whose set of nodes is finite or countably infinite. Our main result is the first deterministic rendezvous algorithm working under this general scenario.

For any node v and any positive integer r, let $P(v, r)$ be the number of paths of length r in the graph, starting at node v. For any instance of the rendezvous problem where agents start at nodes v_1 and v_2 at distance D, let $P(v_1, v_2, D) = \max(P(v_1, D), P(v_2, D))$. It is well known that, for example in trees, $\Omega(D + P(v_1, v_2, D) + \log L)$ is a lower bound on rendezvous time for such an instance. The time of our algorithm, working for arbitrary connected graphs of finite degrees, is polynomial in this lower bound.

As an application we solve the problem of approach for synchronous agents in terrains in the plane, in time polynomial in $\log L$ and in the initial distance between the agents in the terrain.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Distributed algorithms

Keywords and phrases Algorithm, graph, rendezvous, mobile agent, terrain

Digital Object Identifier 10.4230/LIPIcs.DISC.2022.11

Funding *Andrzej Pelc*: Research supported by NSERC discovery grant 2018-03899 and by the Research Chair in Distributed Computing at the Université du Québec en Outaouais.

1 Introduction

Two mobile agents have to meet at the same node of a connected graph. This intensely researched task is known as *rendezvous* and has numerous applications. In computer networks, such as the internet, software agents navigate in the network and the purpose of meeting may be to share data collected from distributed databases and to plan further actions based on these data. If the network models a labyrinth, or corridors in a contaminated mine, mobile robots circulating in the network may have to meet to coordinate maintenance or decontamination tasks. Finally, people may want to meet in an unknown mall or park whose alleys are links of a network and crossings are its nodes.

In most of the results concerning rendezvous in graphs, the graph is finite and rendezvous relies on the exploration of the entire graph. Thus the time of rendezvous depends on the size of the graph. This approach is inefficient for very large graphs, and cannot be used for infinite graphs. For such graphs it is natural to seek rendezvous algorithms whose time depends on the initial distance D between the agents. In this paper we adopt this approach and consider deterministic rendezvous in arbitrary connected graphs with nodes of finite degrees, and whose set of nodes is finite or countably infinite.

We also consider the problem of *approach* for synchronous[1] agents in terrains in the plane. This is the task of getting two agents with vision radius 1 navigating in a *terrain* (see the definition below) to see each other, i.e., to reach positions w_1 and w_2 in the terrain, such that the segment $[w_1, w_2]$ is of length at most 1 and is included in the terrain.

1.1 The model

Graphs. We consider arbitrary simple connected graphs with nodes of finite degrees, and whose set of nodes is finite or countably infinite. Nodes of the graph are unlabeled and ports at each node of degree d are arbitrarily labeled $0, 1, \dots, d-1$. There is no coherence between port numbering at different nodes. There are two agents to which the adversary assigns different starting nodes in the graph and different integer labels from a set $\{1, \dots, L\}$. Time is slotted in synchronous rounds. The adversary wakes up the agents in possibly different rounds. After wakeup, the agents move as follows. In each round, an agent can either stay idle or move to an adjacent node. An agent makes a move by choosing a port number at its current node. When entering the adjacent node corresponding to the chosen edge the agent learns the port of entry and the degree of this node. When agents cross each other in an edge, traversing it simultaneously in opposite directions, they do not even notice this fact. Agents cannot mark the visited nodes in any way. We assume that the memory of the agents is unbounded: from the computational point of view they are modeled as Turing machines. Each agent knows its label but not the label of the other agent, and agents have no a priori information about the graph. They do not know L. They execute the same deterministic algorithm whose parameter is the agent's label. The time of a rendezvous algorithm is the worst-case number of rounds since the wakeup of the earlier agent till the meeting. The meeting can occur before the wake-up of the later agent.

We will use the following terminology. A *walk* in a graph is any sequence $(v_0, v_1, \dots, v_k)$, such that $\{v_i, v_{i+1}\}$ is an edge, for any $i < k$. A *path* in a graph is any walk $(v_0, v_1, \dots, v_k)$ such that $v_{i+2} \neq v_i$ for any $i < k-1$. In other words, paths are walks with no immediate backtrack. Since an agent learns the entry port number upon visiting a node, it can avoid backtracking and thus it can travel only using paths.

Terrains. We consider mobile agents modeled by points moving in subsets of the Euclidean plane E^2 and equipped with compasses showing cardinal directions N, E, S, W, with a common unit of length, with clocks ticking at the same rate, and with vision of radius 1. This means that at any point u in the terrain, the agent sees all points v, such that the segment $[u, v]$ has length at most 1 and is included in the terrain.

Consider a finite or countably infinite family $\{O_1, O_2, \dots\}$ of pairwise disjoint closed convex subsets of the Euclidean plane E^2, called *obstacles*. For any real $\epsilon > 0$, such a family is called *ϵ-scattered* if all distances between obstacles are at least ϵ. A *terrain* is a subset of

[1] By this we mean that clocks of the agents tick at the same rate, and when they move, they travel at the fixed speed 1; see section 1.1.

the Euclidean plane which is the complement of the union of a ϵ-scattered family of obstacles, for some $\epsilon > 0$.[2] Hence any terrain is an open connected subset of the Euclidean plane. A terrain that is the complement of the union of a ϵ-scattered family of obstacles is itself called an ϵ-scattered terrain.

Agents wake up at distinct points p_1 and p_2 of the terrain at possibly different times, chosen by the adversary. The distance D in the terrain between points p_1 and p_2 is defined as the infimum of lengths of all polygonal lines between points p_1 and p_2, included in the terrain. The adversary also assigns different integer labels from a set $\{1, \ldots, L\}$ to the agents. The clock of every agent starts at its wake-up time. Each agent knows its own label but not the label of the other agent, and both of them know a common positive real $\epsilon \leq D$, such that the terrain is ϵ-scattered. As before, the memory of the agents is unlimited. Each agent executes a sequence of *actions*. An action can be either waiting at the current point for a chosen time t, or moving along a segment I of length $x \leq 1$ in a chosen direction dir, so that I is included in the terrain. Notice that, since the vision radius of the agents is 1, an agent can check the latter condition before the move. Whenever agents move, they move at the same fixed speed 1. Recall that they may start at different times and that one agent may move while the other waits. The approach is defined as the moment when the agents see each other for the first time. The time of an approach algorithm is the worst-case time since the wakeup of the earlier agent till the approach.

1.2 The lower bounds

We mention two well-known lower bounds on time, one for rendezvous in graphs and one for approach in the terrain.

Graphs. For any node v and any positive integer r, let $P(v, r)$ be the number of paths of length r in the graph, starting at node v. For any instance of the rendezvous problem where agents start at nodes v_1 and v_2 at distance D, let $P(v_1, v_2, D) = \max(P(v_1, D), P(v_2, D))$. It is well known that, for example in trees, $\Omega(D + P(v_1, v_2, D) + \log L)$ is a lower bound on rendezvous time for such an instance. Indeed, the lower bound $\Omega(D)$ is obvious, the lower bound $\Omega(\log L)$ follows from [14] (even in the two-node tree and even for simultaneous start) and the lower bound $\Omega(P(v_1, v_2, D))$ follows from the fact that the adversary can delay one of the agents and place it at the last node at distance D visited by the other agent.

Terrains. For any instance of the approach problem in a terrain T, with agents starting at points p_1, p_2, let D be the distance between p_1 and p_2 in T. Clearly, $\Omega(D)$ is a lower bound on the time of approach, as agents have speed 1. On the other hand, the lower bound $\Omega(\log L)$ holds even in the empty plane and follows from [14]. Hence we get the lower bound $\Omega(D + \log L)$.

1.3 Our results

Our main result is the solution of the feasibility problem of rendezvous in connected graphs under the above described general scenario. We design a rendezvous algorithm working for arbitrary connected graphs with nodes of finite degrees, and whose set of nodes is finite or countably infinite. Its execution time is polynomial in the above mentioned lower bound $\Omega(D + P(v_1, v_2, D) + \log L)$ for the rendezvous problem.

[2] Notice that obstacles do not need to be bounded. In general, disjoint closed unbounded sets in the plane may have distance 0 (such as a curve and its asymptote) but this possibility is precluded in terrains by the fact that the family of obstacles is ϵ-scattered.

This result should be compared to four sets of previous results about rendezvous in graphs, known in the literature. In [14, 19, 25], the authors considered rendezvous under the same scenario but only for finite graphs. The method adopted in these papers crucially relies on the finiteness of the graph, as it requires the exploration of the entire graph. Hence it cannot be applied to infinite graphs, and even in very large finite graphs it is inefficient. In [11], the authors considered some infinite graphs, such as the line and infinite multidimensional grids. They designed almost optimal rendezvous algorithms but they used two very strong assumptions: first, they assumed that the agents know their position in the graph, and second, they assumed simultaneous start. Hence, their results are far from our generality. In [7], the authors considered only trees (finite or infinite), and, in the unoriented case, only regular trees. The regularity assumption was important, as they relied on knowing the size of any ball of a given radius in the tree. (As explained in [7], the regularity assumption could be weakened to assuming that the size of any ball of given radius is bounded and that both agent know a common bound on this size). Hence again, while the complexity of algorithms in [7] is better than ours, their results are far from our generality. Finally, in [13], the authors designed a rendezvous algorithm working in arbitrary connected graphs (finite or infinite), but worked under the asynchronous scenario. In this scenario, meeting at a node cannot be guaranteed, and hence rendezvous conditions are relaxed to allow meeting inside an edge. Moreover, the algorithm from [13] is very inefficient, in particular, its worst-case cost is exponential in L.

To the best of our knowledge, we propose the first algorithm guaranteeing a meeting at a node in arbitrary connected graphs with nodes of finite degrees when each agent knows only its own label. This general result is possible by applying a new way of organizing activity and waiting periods of the agents. These periods are decided according to bits of (transformed) labels of agents. However, while in previous papers, activity meant either exploring the entire finite graph [25] or exploring a ball in infinite trees [7], in the present paper activity means traversing a single path. The second change consists in allocating rapidly increasing periods of time devoted to processing consecutive paths. The agent uses the time allocated to a given path π first traversing it, then waiting at the other end of it, and then traversing back the path π. So even in its activity period the agent spends a long time staying idle. These crucial changes (the "path-by-path" approach and long waiting periods at the end of each path) made it possible to get rid of the assumption of finiteness of the graph in [25] and of the regularity of the tree in [7].

As an application of this new method we solve the problem of approach for synchronous agents in terrains in the plane, in time polynomial in $\log L$ and in the initial distance D in the terrain between the agents. Hence in this scenario, the execution time of our algorithm is polynomial in the lower bound $\Omega(D + \log L)$ for the approach problem.

This result should be compared to five previous results about approach, known in the literature. In [11], the authors designed an almost optimal algorithm for approach but their algorithm had rather limited scope. First, it worked only for the obstacle-free plane, and second, it used two strong assumptions: that the agents know their position in the plane, and that they start simultaneously. In [13], the authors designed an algorithm for approach working for even more general subsets of the Euclidean plane than we do, and working for the asynchronous scenario but the worst-case cost of their algorithm was exponential in L. In [4], the authors designed an almost optimal algorithm for asynchronous approach in the obstacle-free plane, under a strong assumption that each agent knows its position in the plane. In [15], the authors designed a polynomial algorithm for approach of agents with possibly different steady speeds, but their algorithm worked only for the obstacle-free plane, hence had significantly more limited scope. Finally, in [8], the authors strengthened the result from [15] by designing an algorithm for approach working for the asynchronous scenario at cost polynomial in $D + \log L$ but, again, their algorithm worked only for the obstacle-free plane.

1.4 Related Work

Results closest to the present paper were discussed in Section 1.3. In the present section we mention other related work. Rendezvous was studied both in the randomized and deterministic settings. An excellent survey of randomized rendezvous can be found in [2], cf. also [1, 5]. Deterministic rendezvous in networks is overviewed in [23, 24]. Rendezvous was also studied in geometric settings, such as the interval of the real line, e.g., [5, 6], and the plane, e.g., [3, 9, 12]. The task of meeting for more than two agents, called *gathering*, was investigated, e.g., in [16, 18, 20].

In the deterministic setting, investigations were mostly focused on the feasibility and time complexity of synchronous rendezvous in networks. In most of the literature concerning rendezvous in networks, nodes of the network are assumed to be unlabeled and marking nodes by agents is not allowed. In this case, anonymous agents cannot meet in many symmetric networks, e.g., in oriented rings, if they start simultaneously. The reason for this is the symmetry of the initial configuration. In order to make the task feasible, symmetry is usually broken by assigning the agents distinct labels and assuming that each agent knows only its own label. This is the same scenario as in the present paper and in the previously mentioned papers [7, 14, 19, 25]. Some authors studied a weaker scenario in which agents, as well as nodes, are anonymous. Gathering many anonymous agents in unlabeled networks was the subject of [16]. In this weak scenario, not all initial configurations of agents are possible to gather, and the authors of [16] characterized all such configurations. Stronger scenarios were also investigated. The authors of [22] studied the time of rendezvous in labeled networks, in the context of algorithms with advice.

In the asynchronous model, an adversary controls the speed of agents. Asynchronous rendezvous and approach in the plane was studied in [8, 10, 18], and asynchronous rendezvous in graphs was introduced in [21] and later investigated in [4, 17].

2 Rendezvous in connected graphs

2.1 The algorithm

We first introduce some notation and terminology.

For any label $\ell \in \{1, \dots, L\}$ we define the *transformed label* $Trans(\ell)$ as follows. Let $(c_1, \dots, c_s)$ be the binary representation of ℓ (with $c_1 = 1$). First we define the binary sequence $Trans_1(\ell)$ as follows. We replace each bit 1 by 10, each bit 0 by 01 and add bits 11 at the end. The obtained sequence is of length $2s + 2$ and has the property that if we start with two different labels then none of the obtained sequences can be a prefix of the other (cf. [14]). In order to get $Trans_2(\ell)$ we replace in $Trans_1(\ell)$ each bit 1 by 10 and each bit 0 by 01. The resulting sequence $Trans_2(\ell)$ has length $4s + 4$. Notice that since binary representations of labels may have different lengths, the same is true for the resulting sequences $Trans_2(\ell)$. However, they have the property that if $\ell_1 \neq \ell_2$ then there exists an index j, such that the jth bit of $Trans_2(\ell_1)$ is 1 and the jth bit of $Trans_2(\ell_2)$ is 0. (Hence, not only $Trans_2(\ell_1)$ and $Trans_2(\ell_2)$ differ in some bit but we can guarantee that in some position the bits are 1 and 0 and in some other position the bits are 0 and 1). Finally, we obtain $Trans(\ell)$ from $Trans_2(\ell)$ by adding to it the prefix 011. The resulting sequence of length $4s + 7$ still has the previous two properties (being prefix-free and guaranteeing that if $\ell_1 \neq \ell_2$ then there exists an index j, such that the jth bit of $Trans_2(\ell_1)$ is 1 and the jth bit of $Trans_2(\ell_2)$ is 0) and moreover it has the third property that the segment consisting of bits with indices 2,3,4 is the only segment of three consecutive bits 1 in $Trans(\ell)$.

As an example consider label $\ell = 9$. Then the binary representation of ℓ is 1001. We have $Trans_1(\ell) = (1001011011)$, $Trans_2(\ell) = (10010110011010011010)$, and $Trans(\ell) = (01110010110011010011010)$.

We define the infinite binary sequence $Tape(\ell)$ as the concatenation of infinitely many copies of $Trans(\ell)$. We will call $Tape(\ell)$ the tape of the agent with label ℓ. The i-th copy of $Trans(\ell)$ is called the i-th segment of $Tape(\ell)$.

Any path $\pi = (v_0, v_1, \ldots, v_k)$ of length k starting at node v_0 is coded as the sequence of port numbers $(p_0, \ldots, p_{k-1})$ such that port p_i at node v_i leads to node v_{i+1}. We will often identify a path with its code. The length k of path π is denoted by $|\pi|$. We denote by $rev(\pi)$ the reverse path $(v_k, v_{k-1}, \ldots, v_0)$ coded by the sequence of port numbers $(q_k, q_{k-1}, \ldots, q_1)$, such that port q_j at node v_j leads to node v_{j-1}. Since an agent learns the entry port upon entering a node, an agent that has traversed the path π learns (the code of) the path $rev(\pi)$.

For any node v, we define the infinite sequence of all finite paths $(\pi_1, \pi_2, \ldots)$ starting at node v and ordered as follows: every path of smaller length precedes every path of larger length, and paths of a given length are ordered lexicographically by their codes. Since in our algorithm all paths of smaller lengths are traversed by the agents before all paths of larger length, when an agent has finished processing all paths of length i, it knows (the codes of) all paths of length $i + 1$ because when an agent is at the end of a path of length i, it sees the degree of the final node.

The high-level idea of the Algorithm RV, guaranteeing rendezvous in any connected graph with nodes of finite degrees, is the following. The algorithm is executed by an agent with label ℓ starting at a node v. We assign rapidly increasing time periods a_i (in our solution, the integers a_i increase quadratically) to process consecutive bits b_i of $Tape(\ell)$ of the agent. If the bit b_i is in the j-th segment of $Tape(\ell)$, then its processing concerns the path π_j starting at v, in the following way:

- if $b_i = 1$ then the agent traverses path π_j, waits $a_i - 2|\pi_j|$ rounds at the end of it, and traverses path $rev(\pi_j)$;
- if $b_i = 0$ then the agent waits a_i rounds.

Note that the agent starts and ends processing each bit of $Tape(\ell)$ at its starting node v. We will show that rendezvous must occur by the time when one of the agents processes all bits of its $Tape(\ell)$ corresponding to the lexicographically smallest among shortest paths from its initial position to the initial position of the other agent.

Below we give the pseudo-code of the algorithm. For any positive integer i we define $a_i = 3i^2$. The algorithm is interrupted as soon as the agents meet.

▶ Remark. To show that the formulation of the algorithm is correct, we need to prove that if b_i is the i-th bit of $Tape(\ell)$ located in the j-th segment of $Tape(\ell)$ then $a_i \geq 2|\pi_j|$. Indeed, we have $a_i = 3i^2 \geq 2i \geq 2j \geq 2|\pi_j|$.

2.2 Correctness and complexity

In this section we prove the correctness of Algorithm RV and establish its complexity. For an instance of the rendezvous problem, where agents A_1 and A_2 start at nodes v_1 and v_2 respectively, we define the critical segment of A_1 as follows. Let π be the lexicographically

Algorithm 1 Algorithm `RV`.

```
for $i = 1, 2, \ldots$ do
    if $b_i$ is the $i$-th bit of $Tape(\ell)$ located in the $j$-th segment of $Tape(\ell)$ then
        if $b_i = 1$ then
            traverse path $\pi_j$;
            wait $a_i - 2|\pi_j|$ rounds;
            traverse path $rev(\pi_j)$;
        else
            wait $a_i$ rounds;
```

smallest among shortest paths from v_1 to v_2. We call path π the critical path of the agent. The critical segment of the tape of agent A_1 is the segment of its tape assigned to path π. The critical segment of the tape of agent A_2 is defined similarly.

The correctness of Algorithm `RV` follows from the two following lemmas.

▶ **Lemma 1.** *Suppose that the agents start executing Algorithm* `RV` *simultaneously. Then they meet by the end of the execution of the critical segment of the agent that starts its critical segment first.*

Proof. Let A_1 be the agent that starts its critical segment earlier and let A_2 be the other agent. If both agents start their critical segments simultaneously, we call A_1 and A_2 arbitrarily. Let T be the round in which A_1 starts its critical segment. Consider two cases.

Case 1. Agent A_2 starts some segment in round T. For $i = 1, 2$, let σ_i be the segment that agent A_i starts in round T. There exists an index j such that the j-th bit of agent A_1 is 1, the j-th bit of agent A_2 is 0, and these bits are in segments σ_1 and σ_2, respectively. Hence, in the first part of the execution of its j-th bit, agent A_1 traverses its critical path, while agent A_2 waits at its other end. Thus the agents meet at the starting node of A_2.

Case 2. Agent A_2 does not start any segment in round T. Let σ_2 be the segment which agent A_2 is processing in round T. Let (d_1, d_2, d_3, d_4) be the four bits that agent A_2 executes starting in round T, i.e., while agent A_1 executes the first four bits of its critical segment. (d_1, d_2, d_3, d_4) are not the first four bits of the segment σ_2 of agent A_2. If all bits d_2, d_3, d_4 are within segment σ_2 then at least one of them must be 0, as there cannot be three consecutive bits 1 (apart from positions 2,3,4 in a segment). If some of these bits are within the segment τ following σ_2, then one of them must be 0, since every segment starts with a 0. Hence, in any case, there exists an index j such that the j-th bit of agent A_1 is 1 and the j-th bit of agent A_2 is 0. Hence, in the first part of the execution of its j-th bit, agent A_1 traverses its critical path, while agent A_2 waits at its other end. Thus the agents meet at the starting node of A_2. ◀

▶ **Lemma 2.** *Suppose that the agents do not start executing Algorithm* `RV` *simultaneously. Then they meet by the end of the execution of the critical segment of the agent that starts executing Algorithm* `RV` *earlier.*

Proof. Let A_1 be the agent that starts the execution of Algorithm `RV` earlier, and let A_2 be the other agent. Let T be the round in which A_1 starts its critical segment and let $k-1$ be the index of the first bit in this segment. Hence agent A_1 starts the execution of the second bit of its critical segment (which is equal to 1) in round $T' = T + a_{k-1}$. Let δ be the delay in the start of execution of agent A_2 w.r.t. the start of agent A_1. We have following two cases.

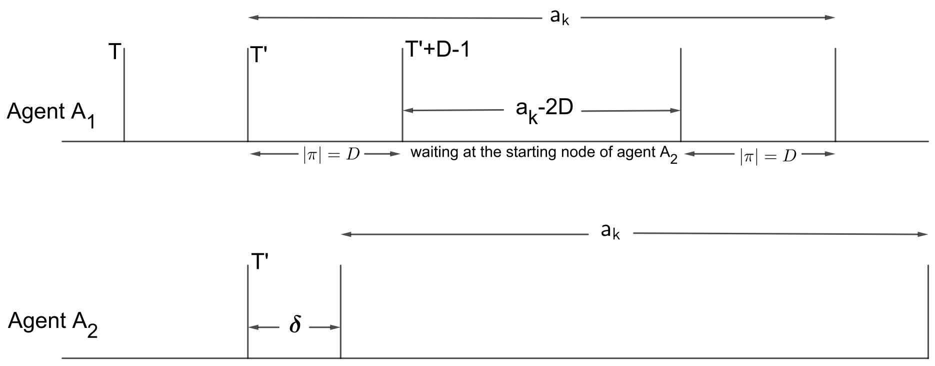

Figure 1 An illustration for the proof of Lemma 2: when $\delta \leq D$, agent A_1 meets agent A_2 during the execution of the k-th bit of agent A_1.

- $\delta \leq D$ **:** Let us consider the time interval $[T', T' + D - 1]$ (cf. Fig. 1). During this time interval agent A_1 traverses its critical path π (of length D) by processing its k-th bit. Since $\delta \leq D$, the execution of the $k-th$ bit of agent A_2 starts in the time interval $[T', T'+D-1]$. If the k-th bit of A_2 is 0, then we are done, because agent A_1 meets agent A_2 at the starting node of A_2 during the execution of this bit of agent A_2. Otherwise, the k-th bit of A_2 is 1 and we have the following two possibilities. If the k-th bit of A_2 is not the first bit 1 of some segment σ_2, then at least one among the $(k+1)$-th and $(k+2)$-th bits of A_2 is 0. Since the k-th, the $(k+1)$-th and the $(k+2)$-th bits of A_1 are 1, the agents meet at the starting node of A_2 during the execution of the first bit 0 of A_2 following its k-th bit. Now consider the case when the k-th bit of A_2 is the first bit 1 of some segment σ_2. The start of the execution of segment σ_2 by agent A_2 is delayed by at most D with respect to the start of the execution of the critical segment by agent A_1. Hence, for some index m, the m-th bit of agent A_1 is 1, the m-th bit of agent A_2 is 0 and the delay between the executions of these bits is most D. Hence the agents meet at the starting node of A_2, during the executions of their m-th bits.

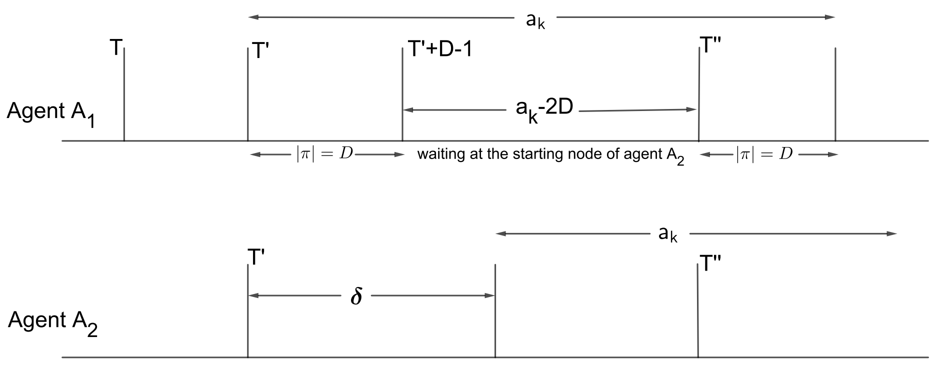

Figure 2 An illustration for the proof of Lemma 2: when $\delta > D$ and $\delta \leq T'' - T'$, agent A_1 meets agent A_2 during the execution of the k-th bit of agent A_1.

- $\delta > D$ **:** Let j be the index of the bit which agent A_2 is processing in round $T' + D$. Since $\delta > D$, we have $j < k$.
  Consider the processing of the k-th bit of agent A_1. Since this bit is 1, agent A_1 first traverses its critical path π (of length D), then waits $a_k - 2D$ rounds at the other end of π (which is the starting node of A_2), and finally traverses the reverse path $rev(\pi)$. We will use the following claim.

  ▷ Claim. $a_k - 2D > a_j$.

  To prove the claim notice that $a_j \le a_{k-1}$, since $j < k$. Since $1 \le D \le k$, we have

  $$a_k - a_j \ge a_k - a_{k-1} = 3k^2 - 3(k-1)^2 = 6k - 3 \ge 6D - 3 > 2D,$$

  which proves the claim.
  Let $T'' = T' + a_k - D - 1$. Consider the time interval $I = [T' + D, T'']$ which is the time interval during which agent A_1 waits at the starting node of A_2. If $\delta \le T'' - T'$ then the start of the execution of the k-th bit of A_2 happens during the time interval I, hence the agents meet at the starting node of A_2 (cf. Fig. 2). If $\delta > T'' - T'$ then during the last round of interval I agent A_2 executes some j-th bit, for $j < k$. By the claim, the start of the execution of the j-th bit of agent A_2 happens during the waiting period of agent A_1 (cf. Fig. 3). Hence the agents meet at the starting node of A_2 during time interval I.

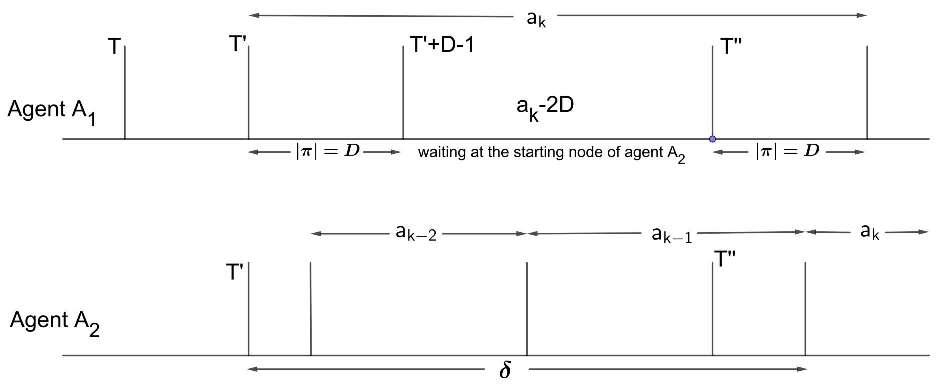

Figure 3 An illustration for the proof of Lemma 2: when $\delta > D$ and $\delta > T'' - T'$, agent A_1 meets agent A_2 during the execution of the k-th bit of agent A_1. ◀

We are now ready to prove the main result of this section.

▶ **Theorem 3.** *Algorithm* RV *guarantees rendezvous of agents starting at nodes v_1 and v_2 at distance D in an arbitrary connected graph (finite or infinite) in time polynomial in $D + P(v_1, v_2, D) + \log L$.*

Proof. Let A be the agent that started the execution of the algorithm first, and in the case of simultaneous start, let it be the agent that started its critical segment first. Let v be the starting node of A. By Lemmas 1 and 2 we know that the agents will meet by the end of the execution of the critical segment of agent A. Let P be the number of paths of length at most D starting at v. We have

$$P = \sum_{i=1}^{D} P(v,i) \leq D \cdot P(v,D).$$

The number of segments till the critical segment of A is at most P. Each segment contains at most $c \log L$ bits, for some constant c. Hence the number B of all bits until the end of the critical segment is at most $cP \log L$. The execution time of the algorithm is at most

$$\sum_{j=1}^{B} a_j = \sum_{j=1}^{B} 3j^2 = \frac{B(B+1)(2B+1)}{2} \in O(B^3).$$

On the other hand we have

$$O(B^3) \subseteq O((P \log L)^3) \subseteq O((D \cdot P(v,D) \cdot \log L)^3) \subseteq O((D \cdot P(v_1,v_2,D) \cdot \log L)^3),$$

which is polynomial in $D + P(v_1,v_2,D) + \log L$. ◀

3 Approach in terrains

In this section we use the path-by-path method from Algorithm RV to solve the problem of approach for synchronous agents in terrains, in time polynomial in $\log L$ and in the initial distance D between the agents in the terrain. We do it in two steps. First, we modify Algorithm RV to obtain an efficient rendezvous algorithm for arbitrary connected subgraphs of the infinite oriented grid, and then we derive an algorithm for approach from this modified algorithm.

We consider the infinite oriented grid $\mathbb{Z} \times \mathbb{Z}$. Every node is adjacent to the four nodes at distance 1 from it in directions North, East, South, West. Ports at nodes of the grid are denoted N, E, S, W, according to the orientation. We define a *shape* as any connected subgraph of this grid. Mobile agents navigate in a fixed shape along its edges. An agent located at a current node of the shape knows which of the ports correspond to edges in the shape. The agents have the same characteristics as described for general connected graphs. Our first aim is to design a rendezvous algorithm working in arbitrary shapes in time polynomial in $\log L$ and in the initial distance D between the agents in the shape.

If we were not concerned with the efficiency, we could directly use Algorithm RV, as shapes are connected graphs. However, this would not give us the desired complexity polynomial in $\log L$ and in D. Indeed, recall that Algorithm RV guarantees rendezvous of agents starting at nodes v_1 and v_2 at distance D in time polynomial in $P(v_1,v_2,D) + \log L$. For shapes (in fact even for the empty grid) the number of paths of length D between two nodes at distance D may be exponential in D. (For example, the number of paths of length $2a$ between nodes (x,y) and $(x+a,y+a)$ which are at distance $2a$ in the grid is $\binom{2a}{a}$).

Luckily, we can significantly reduce the number of processed paths using the orientation of the grid. As in Algorithm RV, paths in shapes are ordered so that every path of smaller length precedes every path of larger length, and paths of a given length are ordered lexicographically by their codes which are sequences of ports N, E, S, W ordered $N < E < S < W$. The key change is to avoid processing all paths in the shape. We do it by associating to each node w in the shape at distance i from the starting node v of the agent, a single specific path of length i called *canonical*. This is the lexicographically smallest of all paths of length i from v to w in the shape. Since all paths of smaller lengths are traversed by the agents before all paths of larger length, when an agent has finished processing all canonical paths of length i, it knows (the codes of) all canonical paths of length $i+1$. Indeed, the canonical path of length $i+1$ from v to w has a prefix of length i which is the canonical path from v to a neighbor w' of w at distance i from v in the shape. Thus the agent can determine this canonical path to w before starting its traversal.

This is the only change we make in Algorithm RV: the sequence $(\pi_1, \pi_2, \dots)$ is now the sequence of all canonical paths in the shape, starting at v, ordered as above, and the rest of the algorithm is as before.[3] The resulting rendezvous algorithm, working for arbitrary shapes, is called Algorithm Shape-RV. Let $C(v,i)$ be the number of canonical paths of length i in the shape, starting at node v. By definition, $C(v,i)$ is equal to the number of nodes in the shape at distance i from v, and since shapes are subgraphs of the grid, we have that $C(v,i)$ is in $O(i^2)$. The same analysis as for Algorithm RV proves the following lemma.

▶ **Lemma 4.** *Algorithm* RV-Shape *guarantees rendezvous of agents starting at nodes v_1 and v_2 at distance D in an arbitrary shape in time polynomial in $D + \log L$. Rendezvous occurs at the starting node of one of the agents.*

We are now ready to make the second step in our design of the algorithm for approach in terrains. Let us first consider agents operating in the same shape, as above, but in a slightly changed model. Instead of operating in synchronous rounds (as is usually the case for the graph setting) we allow the agents to start with any positive delay δ, not necessarily integer. This means that the later agent may start while the earlier agent is traversing an edge. We use the same algorithm as above, i.e, Algorithm RV-Shape. It is easy to see that Lemma 4 still holds in this slightly more general situation. Indeed, the proof of Lemma 1 does not need any change (since it deals with the case $\delta = 0$) and the proof of Lemma 2 requires only minimal changes, replacing rounds by points in time. (For example, the time interval $[T', T' + D - 1]$ which meant a segment of D rounds in the proof of Lemma 2 would have to be replaced by the time interval $[T', T' + D]$ meant as a time segment of length D between two points in time).

It is well known [13, 15] that the problem of approach in the empty plane (without obstacles) can be reduced to that of rendezvous in an infinite oriented grid. For completeness we sketch this reduction below. For any point v in the plane, consider the infinite oriented grid G_v defined as the following graph embedded in the plane. One of the nodes of G_v is v and every node u is adjacent to 4 nodes at Euclidean distance 1 from it, and located North, East, South and West from node u. Ports at every node are labeled N, E, S, W, in the obvious way.

Any rendezvous algorithm in the grid G_v (whose aim is to bring two agents starting at arbitrary nodes of the grid with arbitrary delay to the same node at the same time) can be transformed in an approach algorithm in the empty plane as follows. Let A be any rendezvous algorithm for G_v. Algorithm A can be executed in the grid G_w, for any point w in the plane. Consider two agents in the plane starting respectively from point v and from another point w in the plane, with some delay δ. Let v' be the node of G_v closest to point w. We will say that v and v' are *companions*. If there are more than one closest nodes, we pick one of them arbitrarily. Notice that v' is at distance at most $\sqrt{2}/2 < 1$ from w. Let α be the vector $v'w$. Execute algorithm A on the grid G_v with agents starting at nodes v and v' with delay δ. Let u be the node of G_v in which these agents meet at some time t. The transformed algorithm A^* for approach in the plane works as follows: execute the

[3] Notice that we could not apply this method for arbitrary connected graphs. In an anonymous graph it is impossible to tell if two paths with given codes end up at the same node or not. This shows the power of the grid orientation.

same algorithm A but with one agent starting at v and traveling in G_v and the other agent starting at w and traveling in G_w, so that the starting time of the agent starting at w is the same as the starting time of the agent starting at v' in the execution of A in G_v; the starting time of the agent starting at v does not change. In time t the agent starting at v and traveling in G_v will be at point p, as previously. The agent starting at w and traveling in G_w will get to some point q at time t. Clearly, p is a node of G_v, q is a node of G_w, points p and q are companions and $q = p + \alpha$. Hence both agents will be at distance less than 1 at time t, which means that they accomplish approach in the (empty) plane. Notice that the transformed algorithm A^* has the same time as algorithm A.

Unfortunately, the above transformation does not guarantee approach even if there is one closed convex obstacle in the plane. Indeed, this obstacle could be positioned in such a way that the segment $[p, q]$ intersects it, and hence although agents get close to each other, they cannot see each other. This is why we need a preprocessing for the approach algorithm and we need to carefully choose the length of edges of the grids depending on the given parameter ϵ, such that the terrain is ϵ-scattered.

We will use the following geometric observation.

▶ **Lemma 5.** *Let v be any point in an ϵ-scattered terrain. Let $\xi = \min(1, \epsilon)$. Then there exists a point v' in the terrain at distance $\xi/3$ from v, such that the distance between v' and any obstacle is at least $\xi/3$.*

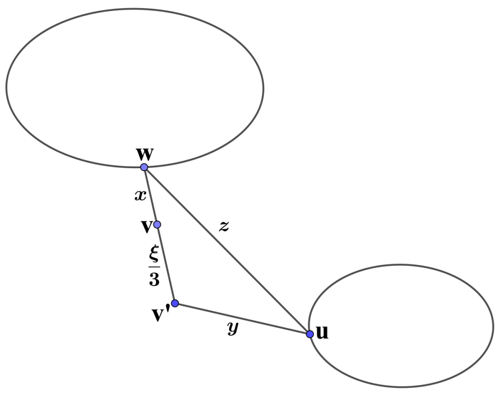

Figure 4 An illustration for the proof of Lemma 5.

Proof. If the distance between v and any obstacle is at least $\xi/3$ then we can take $v' = v$. Hence suppose that the distance between v and the closest obstacle O is $x < \xi/3$. Let w be the point in O such that the distance between v and w is x (cf. Fig. 4).

Let v' be the (unique) point in the terrain in the line vw at distance $\xi/3$ from v. (The existence of such a point follows from the fact that the terrain is ϵ-scattered, and the unicity follows from the definition of x.) Clearly, v' is at distance larger than $\xi/3$ from O. Consider any other obstacle O'. It is enough to show that v' is at distance at least $\xi/3$ from O'. Let u be the point of O' closest to v' and let y be the distance between u and v'. Let the distance between w and u be z. Consider the triangle $wv'u$. We have $x + \xi/3 + y \geq z \geq \epsilon \geq \xi$. Hence $y \geq \xi/3$. ◀

The preprocessing part of our algorithm for approach executed by agent A is the following Procedure `Away`(ϵ) which takes as parameter a positive real ϵ such that the terrain is ϵ-scattered. Recall that both agents get the same ϵ as input. Procedure `Away`(ϵ) works as follows. Let $\xi = \min(1, \epsilon)$ and let $\lambda = \xi/3$. Let v be the starting point of agent A. If the agent does not see any obstacles at distance less than λ then it defines $v' = v$ and terminates the procedure. Otherwise, the agent defines w to be the closest point in the closest obstacle. Then it goes along the line vw away from the point w to the point v' at distance λ from v. Since the terrain is ϵ-scattered, the point v' is in the terrain. This concludes the procedure. By Lemma 5, point v' is at distance at least λ from any obstacle.

Consider the grid $H_{v'}$ which is defined similarly as the above grid $G_{v'}$, with the only exception that adjacent nodes are at distance λ instead of distance 1. Define the shape $S_{v'}$ as the subgraph of $H_{v'}$ induced by nodes that are points of the terrain. Now the Algorithm `Approach` executed by agent A can be succinctly described as follows. Execute Procedure `Away`(ϵ) to get to point v' and then execute Algorithm `RV-Shape` in the shape $S_{v'}$.

The following theorem proves the correctness and establishes the complexity of Algorithm `Approach`.

▶ **Theorem 6.** *Algorithm* `Approach` *accomplishes approach of arbitrary agents in any ϵ-scattered terrain in time polynomial in $D + \log L$, where D is the distance between the starting points of the agents in the terrain.*

Proof. Consider two agents A and B, starting from points v and w respectively. Let v' and w' be the points which the agents A and B reach after executing the Procedure `Away`. Algorithm `RV-Shape` is now executed by agent A in shape $S_{v'}$ and by agent B in shape $S_{w'}$. First suppose that $S_{v'} = S_{w'}$. In this case Lemma 4 guarantees that agents will meet either at v' or at w'. Without loss of generality, suppose that they meet at w'. Now suppose that $S_{v'} \neq S_{w'}$. Hence the agents operate in different shapes. Let w^* be the companion node of w'. The point w^* is a node in the shape $S_{v'}$. Lemmas 1 and 2 can still be used to guarantee that when agent A gets to node w^* during the execution of its critical segment (where the critical path is now from w' to w^*) then agent B is at point w'. This is because the considerations in the proofs of Lemmas 1 and 2 do not depend on which paths agent B is traversing in the execution of Algorithm `RV-Shape` (these paths may depend on the shape in which it operates) but only on the waiting times at its starting node which are independent of the shape.

In view of Lemma 5, there is no obstacle at distance less than λ from point w'. The companion w^* of w' is at distance smaller than λ from w'. Hence the segment w^*w' is contained in the terrain and has length smaller than 1, and thus the agents can see each other. This proves the correctness of Algorithm `Approach`.

As for complexity, first observe that the duration of Procedure `Away` is at most $\lambda \leq D$ (because agents were given a common $\epsilon \leq D$ such that the terrain is ϵ-scattered, and $\lambda \leq \epsilon$). Notice that since the shape $S_{v'}$ is the subgraph of $H_{v'}$ induced by nodes that are points of the terrain, the distance between any two nodes of the shape (defined as the distance in the graph) is at most twice the distance D between these points in the terrain. In view of Lemma 4, we conclude that the execution time of Algorithm `RV-Shape` (which is the second part of Algorithm `Approach`) is polynomial in $D + \log L$. Hence the execution time of the entire Algorithm `Approach` is also polynomial in $D + \log L$. ◀

4 Conclusion

We presented two deterministic algorithms: one for rendezvous in arbitrary connected graphs with nodes of finite degrees (whose set of nodes is finite or infinite) and the other for approach in terrains in the plane. For the rendezvous algorithm the scope is the most general possible. Indeed, rendezvous is obviously impossible in disconnected graphs, and if a graph has a node of infinite degree then there is no rendezvous algorithm guaranteed to always finish in any finite time T, even if agents start from adjacent nodes. The remaining open problem concerns complexity. Our algorithm works in time polynomial in $D + P(v_1, v_2, D) + \log L$ and we did not try to optimize the degree of this polynomial. The problem of designing a rendezvous algorithm working for arbitrary connected graphs in optimal time remains open. This is a challenging problem: it remains unsolved even for finite graphs, despite two decades of intense research.

It is important to note that techniques used in the literature for rendezvous in finite graphs do not seem to have an easy extension to the case of infinite graphs. For example, the idea of trying increasing guesses on the size n of the graph and running known algorithms for finite graphs does not seem to work in infinite graphs for the following reason. Rendezvous algorithms for graphs with size bounded by n rely on exploration of the entire graph using sequences $UXS(n)$ (i.e., Universal Exploration Sequences of port numbers for graphs of size at most n), cf. [25]. Such a sequence guarantees visiting all nodes of a graph of size at most n but does not give any guarantee of visiting some ball in an infinite graph. This is due to the anonymity of the graphs: an agent cannot detect if it enters a loop instead of visiting all nodes at distance r. Existing rendezvous algorithms for finite graphs rely on one agent waiting at its starting node and the other catching it by visiting this node, which could not be guaranteed for any guess. Another possibility would be to exhaustively explore a ball of a given radius r (for increasing values of r) by traversing all paths of length r (coded as sequences of port numbers) starting at the starting node of the agent. However, in arbitrary infinite graphs, there is no upper bound on the number of such paths, and thus it would be impossible to determine a sufficiently long waiting period for agent A at its starting node to guarantee that the other agent B certainly catches it. These difficulties forced us to invent the "path-by-path" technique used in this paper.

For approach, three problems remain open: the first concerns the scope, the second concerns information available to the agents, and the third concerns complexity. Our algorithm works for ϵ-scattered terrains and its time is polynomial in $D + \log L$, while $\Omega(D + \log L)$ is the lower bound on the complexity of approach. This invites three questions. The first concerns the generality of the environment. While our class of terrains is fairly large, it is natural to ask if there exists an algorithm for approach working in arbitrary open connected subsets of the plane, with similar complexity. An algorithm for approach working in all such subsets could be easily obtained by a modification of the result from [13] but its complexity is prohibitive: it is exponential in L. The second problem concerns the information available to the agents. We assumed that each agent knows its own label, and both of them know a common real $\epsilon \leq D$, such that the terrain is ϵ-scattered. The first assumption is necessary to break symmetry: anonymous agents walking at the same speed cannot meet deterministically even in the empty plane, if they start simultaneously. However, we may ask if agents can meet in ϵ-scattered terrains without any other knowledge, with complexity similar to that of our algorithm. The third problem concerns optimal complexity of approach. What is the optimal time of approach even only in our setting of ϵ-scattered terrains? This is not known even for the plane without any obstacles. In this case, the best known algorithm has time $O(D^2 \log L)$ (folklore) and the best known lower bound is $\Omega(D^2 + D \log L)$ and follows from [14].

References

1 Steve Alpern. The rendezvous search problem. *SIAM Journal on Control and Optimization*, 33(3):673–683, 1995. doi:10.1137/S0363012993249195.

2 Steve Alpern and Shmuel Gal. *The theory of search games and rendezvous*, volume 55 of *International series in operations research and management science*. Kluwer, 2003.

3 Edward J. Anderson and Sándor P. Fekete. Two dimensional rendezvous search. *Operations Research*, 49(1):107–118, 2001. doi:10.1287/opre.49.1.107.11191.

4 Evangelos Bampas, Jurek Czyzowicz, Leszek Gasieniec, David Ilcinkas, and Arnaud Labourel. Almost optimal asynchronous rendezvous in infinite multidimensional grids. In *Proc. 24th International Symposium on Distributed Computing (DISC 2010)*, volume 6343, pages 297–311. Springer, 2010. doi:10.1007/978-3-642-15763-9_28.

5 Vic Baston and Shmuel Gal. Rendezvous on the line when the players' initial distance is given by an unknown probability distribution. *SIAM Journal on Control and Optimization*, 36(6):1880–1889, 1998. doi:10.1137/S0363012996314130.

6 Vic Baston and Shmuel Gal. Rendezvous search when marks are left at the starting points. *Naval Research Logistics (NRL)*, 48(8):722–731, 2001. doi:10.1002/nav.1044.

7 Subhash Bhagat and Andrzej Pelc. Deterministic rendezvous in infinite trees. *CoRR*, abs/2203.05160, 2022. doi:10.48550/arXiv.2203.05160.

8 Sébastien Bouchard, Marjorie Bournat, Yoann Dieudonné, Swan Dubois, and Franck Petit. Asynchronous approach in the plane: a deterministic polynomial algorithm. *Distributed Computing*, 32(4):317–337, 2019. doi:10.1007/s00446-018-0338-2.

9 Sébastien Bouchard, Yoann Dieudonné, Andrzej Pelc, and Franck Petit. Almost universal anonymous rendezvous in the plane. In *Proc. 32nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2020)*, pages 117–127. ACM, 2020. doi:10.1145/3350755.3400283.

10 Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed computing by mobile robots: Gathering. *SIAM Journal on Computing*, 41(4):829–879, 2012. doi:10.1137/100796534.

11 Andrew Collins, Jurek Czyzowicz, Leszek Gasieniec, Adrian Kosowski, and Russell A. Martin. Synchronous rendezvous for location-aware agents. In *Proc. 25th International Symposium on Distributed Computing (DISC 2011)*, volume 6950, pages 447–459. Springer, 2011. doi:10.1007/978-3-642-24100-0_42.

12 Jurek Czyzowicz, Leszek Gasieniec, Ryan Killick, and Evangelos Kranakis. Symmetry breaking in the plane: Rendezvous by robots with unknown attributes. In *Proc. 2019 ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pages 4–13. ACM, 2019. doi:10.1145/3293611.3331608.

13 Jurek Czyzowicz, Andrzej Pelc, and Arnaud Labourel. How to meet asynchronously (almost) everywhere. *ACM Trans. Algorithms*, 8(4):37:1–37:14, 2012. doi:10.1145/2344422.2344427.

14 Anders Dessmark, Pierre Fraigniaud, Dariusz R. Kowalski, and Andrzej Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 46(1):69–96, 2006. doi:10.1007/s00453-006-0074-2.

15 Yoann Dieudonné and Andrzej Pelc. Deterministic polynomial approach in the plane. *Distributed Computing*, 28(2):111–129, 2015. doi:10.1007/s00446-014-0216-5.

16 Yoann Dieudonné and Andrzej Pelc. Anonymous meeting in networks. *Algorithmica*, 74(2):908–946, 2016. doi:10.1007/s00453-015-9982-0.

17 Yoann Dieudonné, Andrzej Pelc, and Vincent Villain. How to meet asynchronously at polynomial cost. *SIAM Journal on Computing*, 44(3):844–867, 2015. doi:10.1137/130931990.

18 Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Gathering of asynchronous robots with limited visibility. *Theor. Comput. Sci.*, 337(1-3):147–168, 2005. doi:10.1016/j.tcs.2005.01.001.

19 Dariusz R. Kowalski and Adam Malinowski. How to meet in anonymous network. *Theor. Comput. Sci.*, 399(1-2):141–156, 2008. doi:10.1016/j.tcs.2008.02.010.

20 Wei Shi Lim and Steve Alpern. Minimax rendezvous on the line. *SIAM Journal on Control and Optimization*, 34(5):1650–1665, 1996. doi:10.1137/S036301299427816X.

21 Gianluca De Marco, Luisa Gargano, Evangelos Kranakis, Danny Krizanc, Andrzej Pelc, and Ugo Vaccaro. Asynchronous deterministic rendezvous in graphs. *Theor. Comput. Sci.*, 355(3):315–326, 2006. doi:10.1016/j.tcs.2005.12.016.

22 Avery Miller and Andrzej Pelc. Tradeoffs between cost and information for rendezvous and treasure hunt. *J. Parallel Distributed Comput.*, 83:159–167, 2015. `doi:10.1016/j.jpdc.2015.06.004`.

23 Andrzej Pelc. Deterministic rendezvous in networks: A comprehensive survey. *Networks*, 59(3):331–347, 2012. `doi:10.1002/net.21453`.

24 Andrzej Pelc. Deterministic rendezvous algorithms. In Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors, *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*, pages 423–454. Springer, 2019. `doi:10.1007/978-3-030-11072-7_17`.

25 Amnon Ta-Shma and Uri Zwick. Deterministic rendezvous, treasure hunts, and strongly universal exploration sequences. *ACM Trans. Algorithms*, 10(3):12:1–12:15, 2014. `doi:10.1145/2601068`.