Asynchronous deterministic rendezvous in graphs

Gianluca De Marco* Luisa Gargano* Evangelos Kranakis† Danny Krizanc‡
Andrzej Pelc§ Ugo Vaccaro*

Abstract

Two mobile agents (robots) having distinct labels and located in nodes of an unknown anonymous connected graph, have to meet. We consider the asynchronous version of this well-studied rendezvous problem and we seek fast deterministic algorithms for it. Since in the asynchronous setting meeting at a node, which is normally required in rendezvous, is in general impossible, we relax the demand by allowing meeting of the agents inside an edge as well. The measure of performance of a rendezvous algorithm is its *cost*: for a given initial location of agents in a graph, this is the number of edge traversals of both agents until rendezvous is achieved. If agents are initially situated at a distance D in an infinite line, we show a rendezvous algorithm with cost $O(D|L_{min}|^2)$ when D is known and $O((D+|L_{max}|)^3)$ if D is unknown, where $|L_{min}|$ and $|L_{max}|$ are the lengths of the shorter and longer label of the agents, respectively. These results still hold for the case of the ring of unknown size but then we also give an optimal algorithm of cost $O(n|L_{min}|)$, if the size n of the ring is known, and of cost $O(n|L_{max}|)$, if it is unknown. For arbitrary graphs, we show that rendezvous is feasible if an upper bound on the size of the graph is known and we give an optimal algorithm of cost $O(D|L_{min}|)$ if the topology of the graph and the initial positions are known to agents.

1 Introduction

1.1 The problem

Two mobile agents (robots) initially located in nodes of a network, modeled as an undirected connected graph, have to meet. This task is known in the literature as the rendezvous problem in graphs. It was mostly studied in the synchronous setting and meeting was required at a node. In this paper we study the asynchronous version of the rendezvous problem. In this setting meeting at a node may be impossible even in the two-node graph, as the adversary can desynchronize the agents and make them visit nodes at different times. Thus we have to relax the requirement and allow agents to meet either in a node or inside an edge. Such a definition of meeting is natural,

*Dipartimento di Informatica e Applicazioni, Università di Salerno, 84081 Baronissi (SA), Italy. E-mail: {demarco,lg,uv}@dia.unisa.it

†School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6, Canada. Research supported in part by NSERC and MITACS. E-mail: kranakis@scs.carleton.ca

‡Computer Science Group, Mathematics Department, Wesleyan University, Middletown, CT 06459 USA. E-mail: dkrizanc@wesleyan.edu

§Département d'informatique, Université du Québec en Outaouais, Gatineau, Québec J8X 3X7, Canada. Research supported in part by NSERC grant OGP 0008136 and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais. E-mail: pelc@uqo.ca

e.g., when agents are robots traveling in a labirynth. We seek efficient deterministic algorithms to solve this asynchronous rendezvous problem.

If nodes of the graph are labeled then agents can decide to meet at a predetermined node and the rendezvous problem reduces to graph exploration. However, in many applications, when rendezvous is needed in a network of unknown topology, such unique labeling of nodes may be unavailable, or agents may be unable to perceive such labels. Hence it is important to design rendezvous algorithms for agents operating in *anonymous* graphs, i.e., graphs without unique labeling of nodes. Clearly, the agents have to be able to *locally* distinguish ports at a node: otherwise, an agent may even be unable to visit all neighbors of a node of degree 3 (after visiting the second neighbor, the agent cannot distinguish the port leading to the first visited neighbor from that leading to the unvisited one). Consequently, agents initially located at two nodes of degree 3, might never be able to meet. Hence we make a natural assumption that all ports at a node are locally labeled 1,...,d, where d is the degree of the node. No coherence between those local labelings is assumed. When an agent leaves a node, it is aware of the port number by which it leaves and when it enters a node, it is aware of the entry port number and of the degree of the node. Unless otherwise stated, we do not assume any knowledge of the topology of the graph or of its size. Likewise, agents are unaware of the distance separating them.

1.2 The model

The network. The network is modeled as an undirected connected graph. Since we allow meetings inside an edge, we have to avoid unwanted crossings. Thus, for simplicity, we consider an embedding of the underlying graph in the three-dimensional Euclidean space, with nodes of the graph being points of the space and edges being pairwise disjoint line segments joining them. For any graph such an embedding exists. Mobile agents are modeled as points moving inside this embedding.

Adversarial decisions, definition of rendezvous and its cost. An agent, currently located at a node, does not know the other endpoints of yet unexplored incident edges. If the agent decides to traverse such a new edge, the choice of the actual edge belongs to the adversary, as we are interested in the worst-case performance. This choice given to the adversary captures the fact that the topology and the orientation of the network are unknown to agents. Clearly, sometimes an agent may decide to traverse an already known edge, e.g., when it traverses an edge, goes back and then traverses it again. An algorithm for agent with label L depends on L and causes the agent to make the following decision at any node of the graph: either take a specific already explored incident edge, or take a yet unexplored incident edge (in which case the choice of the edge is made by the adversary).

There is another important choice given to the adversary, this one capturing the asynchronous characteristics of the process. When the agent, situated at a node v at time t_0 has to traverse an edge modeled as a segment $[v, w]$, the adversary performs the following choice. It selects a time point $t_1 > t_0$ and any continuous function $f : [t_0, t_1] \longrightarrow [v, w]$. This function models the actual movement of the agent inside the line segment $[v, w]$ in the time period $[t_0, t_1]$. Hence this movement can be at arbitrary speed, the agent may go back and forth, as long as it does not leave the segment and the movement is continuous. We say that at time $t \in [t_0, t_1]$ the agent is in point $f(t) \in [v, w]$. Moreover, the adversary chooses the starting time of the agent. Hence an agent's trajectory is represented by the concatenation of the functions chosen by the adversary

for consecutive edges that the agent traverses. (Recall that the choice of the edge incident to a current node also belongs to the adversary, whenever the edge is yet unexplored.)

For a given algorithm, given starting points of agents and a given sequence of adversarial decisions in an embedding of a graph G, a rendezvous occurs if both agents are at the same point at the same time. We say that rendezvous is *feasible* in a given graph, if there exists an algorithm for agents such that for any embedding of the graph, any starting points and any sequences of adversarial decisions, the rendezvous does occur. The *cost* of rendezvous is defined as the worst-case number of edge traversals by both agents (the last partial traversal counted as a complete one for both agents), where the worst case is taken over all decisions of the adversary.

Labels and local knowledge. If agents are identical, i.e., they do not have distinct identifiers, and execute the same algorithm, then deterministic rendezvous is impossible even in the oriented ring: the adversary will make the agents move in the same direction at the same speed, thus they will never meet. Hence we assume that agents have distinct identifiers, called labels, which are two different nonempty binary strings, and that every agent knows its own label.

If *both* agents knew *both* labels, the problem could be again reduced to that of graph exploration: the agent with smaller label does not move, and the other agent searches the graph until it finds it. (This strategy is sometimes called "wait for mommy".) However, the assumption that agents know each other may often be unrealistic: agents may be created in different parts of the network in a distributed fashion, oblivious of each other. Hence we assume that each agent knows its own label but does not know the label of the other. The only initial input of a (deterministic) rendezvous algorithm executed by an agent is the agent's label. During the execution of the algorithm, an agent learns the local port number by which it enters a node and the degree of the node.

Notation. L_{min} denotes the shorter label and L_{max} the longer one, with ties broken arbitrarily. If L is a label, $|L|$ denotes its length. n denotes the number of nodes in the graph, and D the distance between initial positions of the agents.

1.3 Our results

We first look at the case of rendezvous in an infinite line. This case is important, as the results for the infinite line carry over to the case of the ring of unknown size and the cost depends on the initial distance between agents rather than on the size of the ring. For agents initially situated at a distance D in an infinite line, we show a rendezvous algorithm with cost $O(D|L_{min}|^2)$ when D is known and $O((D + |L_{max}|)^3)$ if D is unknown, where $|L_{min}|$ and $|L_{max}|$ are the lengths of the shorter and longer label of the agents, respectively. These results still hold for the case of the ring (even of unknown size) but then we also give an algorithm of cost $O(n|L_{min}|)$ (and this is optimal), if the size n of the ring is known, and of cost $O(n|L_{max}|)$, if it is unknown. In both these algorithms the knowledge of the initial distance D between agents is not assumed, and for D of the order of n, their complexity is better than that of infinite line algorithms. On the other hand, for small D and small labels of agents, the opposite is true. For arbitrary graphs, we show that rendezvous is feasible if an upper bound on the size of the graph is known, and we give an optimal algorithm of cost $O(D|L_{min}|)$ if the topology of the graph and the initial positions are known to agents.

It should be noted that asynchronous rendezvous techniques significantly differ from the synchronous case. An important ingredient in synchronous rendezvous algorithms in graphs is

the insertion of idle periods for each of the agents (depending on its label) during which the other agent walks in the graph and has the chance of meeting the standing agent. This is, of course, impossible in the asynchronous setting, as the adversary controls waiting time of the agents. Instead, the algorithm should be designed to force the agents to move on the same path in opposite directions sufficiently far to guarantee meeting. This is complicated by the fact that agents do not have any sense of direction and do not know each other's labels which serve as the algorithm's parameters.

1.4 Related work

The rendezvous problem has been introduced in [24]. The vast literature on rendezvous (see the book [4] for a complete discussion and more references) can be divided into two classes: papers considering the geometric scenario (rendezvous in the line, see, e.g., [11, 12, 19], or in the plane, see, e.g., [9, 10]), and those discussing rendezvous in graphs, e.g., [2, 5]. Most of the papers, e.g., [2, 3, 7, 11, 20] consider the probabilistic scenario: inputs and/or rendezvous strategies are random. In [20] randomized rendezvous strategies are applied to study self-stabilized token management schemes. Randomized rendezvous strategies use random walks in graphs, which have been widely studied and applied also, e.g., in graph traversing [1], on-line algorithms [14] and estimating volumes of convex bodies [17]. A natural extension of the rendezvous problem is that of gathering [18, 20, 23, 25], when more than 2 agents have to meet in one location.

Deterministic rendezvous with anonymous agents working in unlabeled graphs but equipped with tokens used to mark nodes was considered e.g., in [22]. In [26] the authors considered rendezvous of many agents with unique labels. In [16, 21] deterministic rendezvous in graphs with labeled agents was considered. However, in all the above papers, the synchronous setting was assumed. While asynchronous rendezvous under geometric scenarios has been studied, e.g., in [18], the present paper is, to the best of our knowledge, the first to consider deterministic asynchronous rendezvous in graphs. Due to space limitations, some proofs are moved to the Appendix.

2 Rendezvous in the infinite line

We first observe that a simple modification of the (synchronous) rendezvous algorithm proposed in [16] can be used to perform asynchronous rendezvous in an n-node tree in time $O(n)$. Trees have a convenient feature from the point of view of rendezvous. Every tree has either a *central node*, defined as the unique node minimizing the distance from the farthest leaf, or a *central edge*, defined as the edge joining the only two such nodes. This suggests the following natural rendezvous algorithm: explore the tree, find the central node or the central edge, and try to meet there. Each agent can explore the tree by DFS, keeping a stack for used port numbers. At the end of the exploration, the agent has a map of the tree, can identify the central node or the central edge, and can find its way either to the central node or to one endpoint of the central edge, in the latter case knowing which port corresponds to the central edge. In the first case, rendezvous is accomplished after the other agent gets to the central node. In the second case, both agents traverse the central edge once and they have to meet by the time the later agent performs this traversal.

However, the above method uses the possibility of exploring the entire tree in order to construct a map of it. This is impossible, e.g., in the case of the infinite line.

In this section we consider the case when the two agents are initially situated in an infinite line at distance D. While infinite graphs are not our primary interest, the case of the infinite line is important because the obtained results carry over to a ring of arbitrary unknown size and do not depend on this size.

2.1 Distance D known to agents

In this section, we present a rendezvous algorithm with cost $O(D|L_{min}|^2)$. The algorithm assumes that the agents know D and it is formulated for an agent with label L. Each agent has an initial local orientation left-right and these orientations of both agents may be the same or different.

Algorithm Rendezvous-in-Infinite-Line(D)
The algorithm consists of two parts: Label Transformation and Label Execution.
Label Transformation. The Label Transformation part takes the label L of an agent and produces the label L^* consisting of a string of $|L|$ zeros, followed by a 1 and then followed by the string L.
Label Execution. For a given agent, we define the execution of the i-th bit of L^* as performing $2iD$ steps left, $(4i+1)D$ steps right and $(2i+1)D$ steps left (resp. $2iD$ steps right, $(4i+1)D$ steps left and $(2i+1)D$ steps right) if $L^*(i) = 0$ (resp. if $L^*(i) = 1$), according to the agent's initial local orientation. For an agent with label L, the Label Execution part consists of consecutive executions of all bits of L^* from left to right. The algorithm stops when rendezvous is achieved.

The following fact is straightforward.

Fact 2.1.

1. *The label execution of L^* requires $\sum_{i=1}^{|L^*|}(8i+2) = O(|L^*|^2) = O(|L|^2)$ steps;*
2. *For any labels L_1, L_2, if $L_1 \neq L_2$ then none of the L_1^*, L_2^* is a prefix of the other;*
3. *If $|L_1| > |L_2|$ then $|L_1^*| > |L_2^*|$.*

The execution of the p-th bit of L^* can be divided into three segments: the *first segment* consists of the first $2pD$ steps, the *second segment* is formed by the next $(4p+1)D$ steps and finally the *last segment* consists of the last $(2p+1)D$ steps.

Let $s_H(p)$ be the time when agent H finishes executing the p-th bit of its transformed label.

Correctness and analysis

Fix a global left-right orientation of the line. Below we use the terms "left" and "right" according to this fixed orientation.

We define the direction of an agent H on bit position p as left (resp. right) if the execution of the first segment of p for H consists of $2pD$ consecutive steps to the left (resp. right) from its initial position. It is clear that depending on their local orientations and on the values of the p-th bit of their transformed labels, the agents may have either the same or different directions on a given bit position. When the directions of the agents differ on a given bit position, two situations are possible during the execution of the *first* segment of p, depending on the agents' initial positions: either in the first step the agents move approaching each other (in this case, we

say that the directions are *convergent*), or they move receding farther from each other (in this case, we say that the directions are *divergent*).

Lemma 2.1. *If the agents have different directions on a given bit position p, they meet by time* $\min\{s_A(p), s_B(p)\}$.

Proof. Two cases may happen: either (a) the directions are convergent or (b) they are divergent.

Case (a). Let X be the agent that first completes the execution of the first segment of the p-th bit of its transformed label. At this time, if Y is already executing the p-th bit, then they have to meet because they are at distance D and move approaching each other for more than D steps. If Y is still executing some previous bit $q \leq p-1$, then it can be at a distance of at most $(2(p-1)+1)D = (2p-1)D$ steps from its initial position (in some direction), while the first segment of the pth bit executed by X carries it at distance $2pD$, starting towards the initial position of Y. Again, recalling that their initial positions are at distance D, they have to meet.

Case (b). Let X be the agent that first completes the execution of the second segment of the p-th bit of its transformed label. During the second segment, agent X moves $(2p+1)D$ steps from its initial position, towards Y's initial position. As in the previous case, if Y is still executing some bit $q \leq p-1$ when X completes the last segment, then Y can be at most $(2(p-1)+1)D = (2p-1)D$ steps (in any direction) from its initial position: the agents have to meet.

Assume now that at the time τ when X completes the second segment of the p-th bit, Y is already executing the p-th bit. In this case, we can observe that during the execution of the second segment of their p-th bit, the two agents move approaching each other. It follows that at time τ, agent Y can be as far from its initial position as permitted by the execution of the first segment of p, i.e., at most $2pD$ steps in some direction. Recalling that X and Y's initial positions are D steps apart one from the other, and that X moves $(2p+1)D$ steps from its initial position towards Y's initial position, agents have to meet by time $\min\{s_X(p), s_Y(p)\}$. □

Theorem 2.1 (Correctness). *Let q be the length of the shortest tranformed label. Agents must meet by time* $\min\{s_A(q), s_B(q)\}$.

Proof. In view of Lemma 2.1, it is enough to show that there exists a bit position $1 \leq p \leq q$ on which the agents have different directions. In fact, if the orientations of the agents differ, then $p = 1$; otherwise, in view of part 2 of Fact 2.1, there is a bit position $d \leq q$ where the transformed labels differ. In this case, $p = d$. □

Theorem 2.2 (Analysis). *Assume the agents are initially at distance D and D is known to both of them. Then the cost of Algorithm Rendezvous-in-Infinite-Line(D) is $O(D|L_{min}|^2)$.*

Proof. Each agent runs the algorithm proceeding from left to right on the transformed label, executing bit by bit. By Theorem 2.1 we have that the agents must meet by time $\min\{s_A(d), s_B(d)\}$, where d is the first bit position where their transformed labels differ. By parts 1 and 3 of Fact 2.1, $d = O(|L_{min}|^2)$. Executing every bit takes $O(D)$ steps. Hence, the agents must meet after $O(D|L_{min}|^2)$ steps. □

2.2 Distance D unknown to agents

In this section, we present a rendezvous algorithm with cost $O((D + L_{max})^3)$. The algorithm does not assume the knowledge of D. It is formulated for an agent with label L. Each agent has an initial local orientation left-right.

Algorithm Rendezvous-in-Infinite-Line
The algorithm consists of two parts: Label Transformation and Label Execution.
Label Transformation. The Label Transformation part takes the label L of an agent and produces the label L^* in the following way.
Step 1. Produce label L' as follows: insert a new bit after every bit of L, alternating 0 and 1 (e.g., if L is the string 110, we obtain the new string 101100).
Step 2. Produce label L'' by adding the pattern 11110 at the end of L'.
Step 3. Finally, label L^*, called the *transformed label* of the agent, is obtained as an infinite concatenation of copies of L''.
Label Execution. For a given agent, we define the execution of the i-th bit of L^* as performing i^2 steps left, $2i^2 + i$ steps right and $i^2 + i$ steps left (resp. i^2 steps right, $2i^2 + i$ steps left and $i^2 + i$ steps right), if $L^*(i) = 0$ (resp. if $L^*(i) = 1$) according to the agent's initial local orientation. For an agent with label L, the Label Execution part consists of consecutive executions of all bits of L^*, from left to right, until rendezvous is achieved.

As in the previous subsection, the execution of the p-th bit of L^* can be divided into three segments: the first p^2 steps form the *first segment* of p, the next $2p^2 + p$ form the *second segment* of p and the last $p^2 + p$ steps of p form the *last segment*.

Denote by $L^*(a, b)$ the substring of L^* contained between bit positions a and b, extremities included. The following fact is straightforward.

Fact 2.2.

1. *L^* does not contain the substring* 0000*;*
2. *The only place in L^* where the substring* 11110 *occurs is at the end of a copy of L';*
3. *For any bit position b of L^*, the execution of $L^*(1, b)$ requires $\sum_{i=1}^{b}(4i^2+2i) = O(b^3)$ steps;*
4. *For any labels L_1, L_2, if $L_1 \neq L_2$ then $L'_1 \neq L'_2$;*

Correctness and analysis

Directions of agents on a given bit position are defined analogously as in Section 2.1. Also the notions of convergent and divergent directions on a given bit position are similar.

Lemma 2.2. *Let X and Y be the two agents. For any $p > 1$, there exists a $q > p$ such that $q - p = O(|L_{max}|)$ and the direction of X on $L^*_X(q)$ is different from that of Y on $L^*_Y(q)$.*

Proof. Assume X be the agent with the larger label and Y the other agent. Let r be the largest integer and s be the smallest integer, $p < r < s$, such that $L^*_X(r, s)$ includes two occurrences of the substring 11110. Since for every q, with $r \leq q \leq s$, we have $q - p \leq 2 \cdot |L''_{max}| = O(|L_{max}|)$, it is enough to show that there is a $q \in \{r, \ldots, s\}$, such that the agents have different directions on q.

We can assume that the agents have the same direction on the bit positions where $L^*_X(r,s)$ contains the two substrings 11110, otherwise there is nothing to prove. In view of part 1 of Fact 2.2, the only way for this to happen is that both agents have the same orientation and $L^*_X(r,r+4) = L^*_Y(r,r+4) = L^*_X(s-4,s) = L^*_Y(s-4,s) = 11110$.

From part 2 of Fact 2.2, this implies that $L^*_X(r+5,s-5)$ and $L^*_Y(r+5,s-5)$ correspond to L'_X and L'_Y respectively. In view of part 4 of Fact 2.2, there must be a bit position d, $r+5 \leq d \leq s-5$ where L'_X and L'_Y differ. The lemma holds for $q = d$. □

As before, the integer $s_H(p)$ denotes the time when agent H finishes executing the p-th bit of its transformed label.

Lemma 2.3. *If the agents have different directions on a given bit position $p \geq D$, they meet by time* $\min\{s_A(p), s_B(p)\}$.

Proof. Two cases may happen: either (a) the directions are convergent or (b) they are divergent.

Case (a). Let X be the agent that first completes the execution of the first segment of the p-th bit of its transformed label. At this time, if Y is already executing the p-th bit, then they have to meet because they are at distance D and move approaching each other for more than D steps. If Y is still executing some previous bit $q \leq p-1$, then it can be at a distance of at most $(p-1)^2 + (p-1)$ steps, in some direction, from its initial position, while the first segment of the pth bit executed by X carries it at distance p^2, starting towards the initial position of Y. Again, recalling that their initial positions are at distance $D \leq p$, they have to meet.

Case (b). Let X be the agent that first completes the execution of the second segment of the p-th bit of its transformed label. During the second segment, agent X moves $p^2 + p$ steps, from its initial position, towards Y's initial position. As in the previous case, if Y is still executing some bit $q \leq p-1$ when X completes the second segment, then Y can be at most $(p-1)^2+(p-1)$ steps (in any direction) from its initial position: the agents have to meet.
Assume now that at the time τ when X completes the second segment of the p-th bit, Y is already executing the p-th bit. In this case, we can observe that during the execution of the second segment of their p-th bit, the two agents move approaching each other. It follows that at time τ, agent Y can be as far from its initial position as permitted by the execution of the first segment of p, i.e., at most p^2 steps in some direction. Recalling that Y is initially $D \leq p$ steps distant from X, they have to meet by time $\min\{s_X(p), s_Y(p)\}$. □

Theorem 2.3 (Correctness and analysis). *Assume the agents are initially at distance D. Algorithm Rendezvous-in-Infinite-Line achieves the rendezvous in $O((D + |L_{max}|)^3)$ steps.*

Proof. Each agent runs the algorithm proceeding from left to right on the transformed label, executing bit by bit. By Lemma 2.3 the agents must meet by time $\min\{s_A(d), s_B(d)\}$, where $d \geq D$ is the first bit position on which they have different directions. By Lemma 2.2, there exists a bit position $d \geq D$ on which the agents have different directions. This proves the correctness of the algorithm.

Moreover, Lemma 2.2 also guarantees that $d = D + O(|L_{max}|) = O(D + |L_{max}|)$. Hence, from part 3 of Fact 2.2, the cost of the algorithm is $\sum_{i=1}^{d}(4i^2 + 2i) = O((D + |L_{max}|)^3)$.

□

3 Optimal rendezvous in the ring

Our results for the infinite line carry over to the case when agents are situated in a ring of unknown size n. However, for the ring there is no danger of "infinite escape" by going in divergent directions: in this case the agents can still meet "on the other side" of the ring. Consequently, for the ring we get rendezvous algorithms whose cost depends on n (which is also an upper bound on the initial distance between agents) and on the size of the labels. The knowledge of D is not assumed and for D of the order of n the bound on the cost of rendezvous is better than that previously established.

We present a rendezvous algorithm with cost of optimal order of magnitude $O(n|L_{min}|)$, working on an arbitrary unoriented ring of known size and an algorithm with cost $O(n|L_{max}|)$, when the size of the ring is unknown. Since the ring is unoriented, each of the agents has a local right/left orientation and these orientations for the two agents may differ.

Algorithm Rendezvous-in-Ring
The algorithm consists of two parts: Label Transformation and Label Execution. The Label Transformation part takes the label L of an agent and produces the label L^* in the following way. First produce label L' consisting of a string of $|L|$ zeros, followed by a 1 and then followed by the string L. The label L^*, called the *transformed label* of the agent, is obtained by replacing in L' each 0 by 01 and each 1 by 10.
The Label Execution part is divided into phases numbered 1,2,... For a given agent, we define the execution of bit 0 (resp. 1) in phase a as performing 3^a steps left (resp. right), according to the agent's local orientation. For an agent with label L, phase a consists of consecutive executions of all bits of L^* from left to right. Since the agents do not know the size of the ring, the number of phases is unbounded. The algorithm stops when rendezvous is achieved.

The following fact is straightforward.

Fact 3.1.

1. *$|L^*| = O(|L|)$;*
2. *For any labels L_1, L_2, if $L_1 \neq L_2$ then none of the L_1^*, L_2^* is a prefix of the other;*
3. *There are no more than two consecutive equal bits in L^*;*
4. *If $|L_1| > |L_2|$ then $|L_1^*| > |L_2^*|$.*

Correctness and analysis

In order to show the correctness of Algorithm Rendezvous-in-Ring, we need to prove that for any size of the ring, any initial position of the agents and any behavior of the adversary, the agents will eventually meet. Let n be the size of the ring and $x = \lceil \log_3 n \rceil + 1$. Consider agents A and B. Let L_A and L_B be the labels of agents A and B, respectively. Let P be the longest common prefix of L_A^* and L_B^*. For the b-th bit of L_A^* , we denote by $L_A^*(b)$ the value of this bit. Similarly for $L_B^*(b)$. If the b-th bit of L_A^* is still in P then we use notation $P(b)$ to denote $L_A^*(b) = L_B^*(b)$. For an agent H and bit position b, let $t'_H(b)$ be the time when agent H starts executing the b-th bit of its transformed label in phase x. Analogously, let $t''_H(b)$ be the time when agent H finishes executing the b-th bit of its transformed label in phase x.

The following fact is straightforward.

Fact 3.2. *If during a time interval $[t_1, t_2]$ one of the agents does at least $c+n$ consecutive steps in one direction and the other agent does at most c steps in this direction, for any $c \geq 0$, then they must meet by time t_2.*

We first prove two lemmas.

Lemma 3.1. *Let X be the agent that first starts the execution of phase x and let Y be the other agent (i.e. $t'_X(1) \leq t'_Y(1)$).*

1. *If the agents don't meet by time $\min\{t''_X(1), t''_Y(1)\}$, then $t'_Y(1) < t''_X(1)$.*
2. *In the case when the orientations of the agents differ, they meet by time $\min\{t''_X(1), t''_Y(1)\}$.*

The next lemma shows that if agents have the same orientation then they are "almost" synchronized in phase x on the common prefix P.

Lemma 3.2. *Suppose that both agents have the same orientation. Let P be the longest common prefix of L^*_A and L^*_B. Consider the b-th bit of P. Let X_b be the agent that first starts executing this bit in phase x and let Y_b be the other agent (i.e., $t'_{X_b}(b) \leq t'_{Y_b}(b)$). Then $t'_{Y_b}(b) \leq t''_{X_b}(b)$, unless the agents meet by time $t''_{X_b}(b+1)$.*

The following result establishes the correctness of Algorithm Rendezvous-in-Ring.

Theorem 3.1. *Let n be the size of the ring. Let d be the first position where the transformed labels of the agents differ. Agents must meet by time $\min\{t''_A(d), t''_B(d)\}$.*

We finally establish the complexity of Algorithm Rendezvous-in-Ring.

Theorem 3.2. *The cost of Algorithm Rendezvous-in-Ring is $O(n|L_{max}|)$, for the n-node ring. Moreover, if the size n of the ring is known to the agents, then Algorithm Rendezvous-in-Ring can be modified to have cost $O(n|L_{min}|)$, which is optimal.*

Proof. It follows from Theorem 3.1 that by the meeting time none of the agents completed phase x. Hence the cost of the algorithm is $O(n(|L_A| + |L_B|)) = O(n|L_{max}|)$.

Suppose that both agents know n. Then they can start executing Algorithm Rendezvous-in-Ring at the beginning of phase x. By the proof of Theorem 3.1, agents must meet by time $\min\{t''_A(d), t''_B(d)\}$, where d is the first position where their transformed labels differ. Clearly, $d \leq |L_{min}|$. Hence both agents execute at most $|L_{min}|$ bits before meeting, which implies that the cost is $O(n|L_{min}|)$. The optimality of this cost follows from a result in [16], applied to the case when the distance between agents is $\Theta(n)$. □

4 Rendezvous in arbitrary graphs

We start with the following observation.

Proposition 4.1. *If a map of the graph with labeled ports and indicated initial positions of agents is available to both of them then deterministic asynchronous rendezvous can be done at cost $O(D|L_{min}|)$, which is optimal.*

Proof. Each agent computes the distance D between them and finds the lexicographically smallest path of length D from its own position to the position of the other agent (paths are viewed as sequences of port numbers). Thus both agents identify the same cycle of length $2D$ on which their both initial positions are situated. (Notice that the cycle need not be simple, some edges may be repeated, it may even degenerate to one path considered in both directions.) Then agents apply the modified version of Algorithm Rendezvous-in-Ring to this cycle. The size of the cycle is known, so rendezvous can be achieved at cost $O(D|L_{min}|)$. □

We conclude this section with a feasibility result in the case when an upper bound on the size of the graph is known to agents.

Theorem 4.1. *Suppose that a bound M on the number of nodes in the graph is known to both agents. Then deterministic asynchronous rendezvous is feasible.*

5 Conclusion

The results that we presented for the asynchronous deterministic rendezvous in graphs contribute to understanding the feasibility and complexity of this problem which seems far more complex than its synchronous counterpart. In fact, it remains open if asynchronous deterministic rendezvous is at all feasible in arbitrary graphs of unknown size. Our solution heavily uses the knowledge of the upper bound on the size.

As far as complexity is concerned, the optimal cost of rendezvous remains open even in an infinite line. Our algorithm for known D has time complexity $O(D|L_{min}|^2)$, while the lower bound, following from [16], is $\Omega(D|L_{min}|)$. In the case of the n-node ring, we established upper bounds $O(n|L_{min}|)$ and $O(n|L_{max}|)$, for the known and unknown size, respectively. Is rendezvous with cost $O(n|L_{min}|)$ possible for the ring of unknown size? Finally, our algorithm showing feasibility of rendezvous for arbitrary graphs with known upper bound M on the size is not efficient. Is there a rendezvous algorithm polynomial in the bound M and in the lengths of the agents' labels?

References

[1] R. Aleliunas, R.M. Karp, R.J. Lipton, L. Lovász, and C. Rackoff, Random walks, universal traversal sequences, and the complexity of maze problems, Proc. FOCS'1979, 218-223.

[2] S. Alpern, The rendezvous search problem, SIAM J. on Control and Optimization 33 (1995), 673-683.

[3] S. Alpern, Rendezvous search on labelled networks, Naval Reaserch Logistics 49 (2002), 256-274.

[4] S. Alpern and S. Gal, The theory of search games and rendezvous. Int. Series in Operations research and Management Science, Kluwer Academic Publisher, 2002.

[5] J. Alpern, V. Baston, and S. Essegaier, Rendezvous search on a graph, Journal of Applied Probability 36 (1999), 223-231.

[6] S. Alpern and S. Gal, Rendezvous search on the line with distinguishable players, SIAM J. on Control and Optimization 33 (1995), 1270-1276.

[7] E. Anderson and R. Weber, The rendezvous problem on discrete locations, Journal of Applied Probability 28 (1990), 839-851.

[8] E. Anderson and S. Essegaier, Rendezvous search on the line with indistinguishable players, SIAM J. on Control and Optimization 33 (1995), 1637-1642.

[9] E. Anderson and S. Fekete, Asymmetric rendezvous on the plane, Proc. 14th Annual ACM Symp. on Computational Geometry, 1998.

[10] E. Anderson and S. Fekete, Two-dimensional rendezvous search, Operations Research 49 (2001), 107-118.

[11] V. Baston and S. Gal, Rendezvous on the line when the players' initial distance is given by an unknown probability distribution, SIAM J. on Control and Optimization 36 (1998), 1880-1889.

[12] V. Baston and S. Gal, Rendezvous search when marks are left at the starting points, Naval Res. Log. 48 (2001), 722-731.

[13] S.A. Cook and P. McKenzie, Problems complete for deterministic logarithmic space, Journal of Algorithms 8 (5) (1987), 385-394.

[14] D. Coppersmith,, P. Doyle, P. Raghavan, and M. Snir, Random walks on weighted graphs, and applications to on-line algorithms, Proc. STOC'1990, 369-378.

[15] D. Coppersmith, P. Tetali, and P. Winkler, Collisions among random walks on a graph, SIAM J. on Discrete Math. 6 (1993), 363-374.

[16] A. Dessmark, P. Fraigniaud, and A. Pelc, Deterministic rendezvous in graphs, Proc. 11th European Symposium on Algorithms (ESA'2003), LNCS 2832, 184-195.

[17] M. Dyer, A. Frieze, and R. Kannan, A random polynomial time algorithm for estimating volumes of convex bodies, Proc. 21st Annual ACM Symposium on Theory of Computing (STOC'1989), 375-381.

[18] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Gathering of asynchronous oblivious robots with limited visibility, Proc. STACS'2001, LNCS 2010, 247-258.

[19] S. Gal, Rendezvous search on the line, Operations Research 47 (1999), 974-976.

[20] A. Israeli and M. Jalfon, Token management schemes and random walks yield self stabilizing mutual exclusion, Proc. PODC'1990, 119-131.

[21] D. Kowalski, A. Pelc, Polynomial deterministic rendezvous in arbitrary graphs, Proc. 15th Annual Symposium on Algorithms and Computation (ISAAC'2004), December 2004, Hong Kong.

[22] E. Kranakis, D. Krizanc, N. Santoro and C. Sawchuk, Mobile agent rendezvous in a ring, Proc. 23rd International Conference on Distributed Computing Systems (ICDCS'2003), 592-599.

[23] W. Lim and S. Alpern, Minimax rendezvous on the line, SIAM J. on Control and Optimization 34 (1996), 1650-1665.

[24] T. Schelling, The strategy of conflict, Oxford University Press, Oxford, 1960.

[25] L. Thomas, Finding your kids when they are lost, Journal on Operational Res. Soc. 43 (1992), 637-639.

[26] X. Yu and M. Yung, Agent rendezvous: a dynamic symmetry-breaking problem, Proc. International Colloquium on Automata, Languages, and Programming (ICALP'1996), LNCS 1099, 610-621.

Appendix

Proofs of Section 3

Lemma 3.1. *Let X be the agent that first starts the execution of phase x and let Y be the other agent (i.e. $t'_X(1) \leq t'_Y(1)$).*

1. If the agents don't meet by time $\min\{t''_X(1), t''_Y(1)\}$*, then* $t'_Y(1) < t''_X(1)$.

2. In the case when the orientations of the agents differ, they meet by time $\min\{t''_X(1), t''_Y(1)\}$.

Proof. 1. Let v be the position on the ring of agent Y at time $t'_X(1)$. Without loss of generality, assume that the execution of the first bit by agent X is clockwise. Since X executes phase x, the execution of the first bit consists in making 3^x steps clockwise. Suppose, for the purpose of contradiction, that $t'_Y(1) \geq t''_X(1)$. That is, agent Y still executes some phase $< x$ at time $t''_X(1)$, or has just finished it. In particular, $\min\{t''_X(1), t''_Y(1)\} = t''_X(1)$. zeros and the In view of part 3 of Fact 3.1, agent Y could move at most $2 \cdot 3^{x-1}$ steps clockwise from v during the time segment $[t'_X(1), \min\{t''_X(1), t''_Y(1)\}]$. The clockwise distance between agents is less than n. Hence, in order to show that the agents meet by time $t''_X(1)$, it is enough to show that $3^x \geq 2 \cdot 3^{x-1} + n$. This inequality follows from the definition of x.
2. Suppose that the orientations of the agents differ. Assume also that $t'_Y(1) < t''_X(1)$, otherwise the conclusion follows from part 1. Recall that the first bit executed in any phase is 0 for both agents (the string L^* starts with a 0 for any L). Again we assume without loss of generality that the execution of the first bit by agent X is clockwise. Observe that agent Y moves *counterclockwise* in the time segment $[t'_Y(1), \min\{t''_X(1), t''_Y(1)\}]$. If $t''_Y(1) \leq t''_X(1)$ then in the time segment $[t'_Y(1), \min\{t''_X(1), t''_Y(1)\}]$ agent Y does at least n steps counterclockwise while agent A moves only clockwise. Hence, we get the conclusion by Fact 3.2, for $c = 0$. If $t''_Y(1) > t''_X(1)$ then in the time segment $[t'_X(1), \min\{t''_X(1), t''_Y(1)\}]$ agent Y does at most $2 \cdot 3^{x-1}$ steps clockwise from v, by the same argument as in part 1. The clockwise distance between agents is less than n. Agents will meet at the latest by the time when X does $2 \cdot 3^{x-1} + n$ steps clockwise. Again we use the inequality $3^x \geq 2 \cdot 3^{x-1} + n$ to show that this must happen by time $t''_X(1) = \min\{t''_X(1), t''_Y(1)\}$. □

Lemma 3.2. *Suppose that both agents have the same orientation. Let P be the longest common prefix of L^*_A and L^*_B. Consider the b-th bit of P. Let X_b be the agent that first starts executing this bit in phase x and let Y_b be the other agent (i.e., $t'_{X_b}(b) \leq t'_{Y_b}(b)$). Then $t'_{Y_b}(b) \leq t''_{X_b}(b)$, unless the agents meet by time $t''_{X_b}(b+1)$.*

Proof. Notice that b cannot be the last position of L^*_A because of part 2 of Fact 3.1, hence $t''_{X_b}(b+1)$ is well defined. The proof is by induction on b. If $b = 1$ then the lemma follows from the first part of Lemma 3.1 (recall that this part did not depend on orientations of the agents). Assume that $b > 1$ and suppose by the inductive assumption that the lemma holds for $b - 1$.

Claim. If agents do not meet by time $t''_{X_b}(b)$, then $t'_{Y_b}(b-1) \leq t''_{X_b}(b-1)$.

In order to prove the Claim suppose that agents do not meet by time $t''_{X_b}(b)$ and $t'_{Y_b}(b-1) > t''_{X_b}(b-1)$. This implies $t'_{Y_b}(b-1) > t'_{X_b}(b-1)$. Hence, $X_b = X_{b-1}$. However, by the inductive hypothesis $t'_{Y_{b-1}}(b-1) \leq t'_{X_{b-1}}(b-1)$, since we supposed that the agents do not meet by time $t''_{X_b}(b) = t''_{X_{b-1}}(b)$. This is a contradiction.

Consider two cases.

Case 1. $P(b-1) \neq P(b)$

Suppose that $t'_{Y_b}(b) > t''_{X_b}(b)$. In view of the Claim and of Fact 3.2 applied to the time interval $[t'_{X_b}(b), t''_{X_b}(b)]$ agents must meet by time $t''_{X_b}(b)$.

Case 2. $P(b-1) = P(b)$

Notice that in Case 2, b cannot be the last position of P because P cannot end with two equal consecutive bits. In view of part 3 of Fact 3.1, $P(b+1) \neq P(b)$.

Suppose that $t'_{Y_b}(b) > t''_{X_b}(b)$. Then at time $t''_{X_b}(b)$ agent Y_b has not yet started executing the b-th bit. In view of the Claim, agent Y_b must start executing the $(b-1)$-th bit by time $t'_{X_b}(b) \geq t''_{X_b}(b-1)$, unless they met by time $t''_{X_b}(b)$. Hence, in the time segment $[t''_{X_b}(b), t''_{X_b}(b+1)]$ agent X_b does 3^x consecutive steps in one direction, while in the time segment $[t''_{X_b}(b), t''_{Y_b}(b)]$ agent Y_b does at least 3^x consecutive steps in the opposite direction. This implies that they must meet by time $\min\{t''_{X_b}(b+1), t''_{Y_b}(b)\} \leq t''_{X_b}(b+1)$. □

Theorem 3.1. *Let n be the size of the ring. Let d be the first position where the transformed labels of the agents differ. Agents must meet by time* $\min\{t''_A(d), t''_B(d)\}$.

Proof. Recall that for an agent H and bit position b, $t'_H(b)$ is the time when agent H starts executing the b-th bit of its transformed label in phase $x = \lceil \log n \rceil + 1$. Analogously, $t''_H(b)$ is the time when agent H finishes executing the b-th bit of its transformed label in phase x.

If the orientations of the agents differ, then the theorem follows from Lemma 3.1. Hence, suppose that agents have the same orientation. Since d is the first position where the transformed labels of the agents differ, there exists exactly one agent in whose transformed label the $(d-1)$-th and d-th bits are the same. Without loss of generality let us assume it is agent A, i.e., $L^*_A(d-1) = L^*_A(d)$. By part 3 of Fact 3.1, $L^*_A(d-1) \neq L^*_A(d-2) = L^*_B(d-2)$. Assume also that on the two equal bits $L^*_A(d-1)$ and $L^*_A(d)$ agent A moves clockwise. Let $X \in \{A, B\}$ be the first agent that starts executing the $(d-1)$-th bit and let Y be the other agent, i.e., $t'_X(d-1) \leq t'_Y(d-1)$.

Claim. We have

$$t'_X(d-1) \geq t''_X(d-2) \geq t'_Y(d-2),$$

unless agents meet by time $t''_X(d-1)$.

Observe that the first inequality is obvious, while the second follows from the Claim in the proof of Lemma 3.2.

To complete the proof of the theorem consider the following two cases.

Case 1. $X = A$.

In view of the Claim, agent B is in the process of executing the $(d-2)$-th bit at time $t'_A(d-1)$, unless agents meet by time $t''_A(d-1)$. Suppose they do not meet by this time. In the time segment $[t'_A(d-1), t''_B(d)]$ agent B either does at most 3^x steps clockwise or it does at least 3^x consecutive steps counterclockwise. In the time segment $[t'_A(d-1), t''_A(d)]$, agent A does $2 \cdot 3^x$ consecutive steps clockwise. Hence agents have to meet by time $\min\{t''_A(d), t''_B(d)\}$. It follows that in case 1 agents must meet by time $\max\{t''_A(d-1), \min\{t''_A(d), t''_B(d)\}\}$.

Two subcases are now possible. If $t''_A(d-1) \leq t''_B(d)$ then the agents meet by time $\min\{t''_A(d), t''_B(d)\}$. Suppose $t''_A(d-1) > t''_B(d)$. Then in the time segment $[t'_A(d-1), t''_A(d-1)]$, agent A does 3^x steps clockwise, while agent B has done at least 3^x consecutive steps counterclockwise in the time segment $[t'_A(d-1), t''_B(d)]$. In view of the inequality $t''_A(d-1) > t''_B(d)$ they have to meet by time $t''_B(d) = \min\{t''_A(d), t''_B(d)\}$.

Case 2. $X = B$.

In view of the Claim, agent A is in the process of executing the $(d-2)$-th bit at time $t'_B(d-1)$, unless agents meet by time $t''_B(d-1)$. Suppose they do not meet by this time. In the time segment $[t'_B(d-1), t''_A(d)]$ agent A either does at most 3^x steps counterclockwise or does at least $2 \cdot 3^x$ consecutive steps clockwise. In the time segment $[t'_B(d-1), t''_B(d)]$, agent B does 3^x consecutive steps clockwise and 3^x consecutive steps counterclockwise. Hence agents have to meet by time $\min\{t''_A(d), t''_B(d)\}$. It follows that in case 2 agents must meet by time $\max\{t''_B(d-1), \min\{t''_A(d), t''_B(d)\}\}$.

As in the previous case, if $t''_B(d-1) \leq t''_A(d)$ then the agents meet by time $\min\{t''_A(d), t''_B(d)\}$. When $t''_B(d-1) > t''_A(d)$, proceeding analogously to the Case 1 we can conclude that agents have to meet by time $t''_A(d) = \min\{t''_A(d), t''_B(d)\}$.

Hence, in both cases, agents have to meet by time $\min\{t''_A(d), t''_B(d)\}$. □

Proofs of Section 4

Theorem 4.1. *Suppose that a bound M on the number of nodes in the graph is known to both agents. Then deterministic asynchronous rendezvous is feasible.*

Proof. Consider all pairs (G, s), where G is a graph with at most M nodes and s is a node of the graph G. For each such graph consider all possible labelings $(1, ..., d)$ of ports at each node of degree d. Define an *input* to be any pair (G, s), where G is a graph with a port labeling and s is a node of the graph G.

For any input choose a *traversal* of the graph G from the starting node s, i.e., a sequence S of port numbers such that an agent starting at s and following this sequence of port numbers (exit port, entry port, exit port, entry port,...) traverses all edges of G. For each traversal S, let S^* be the concatenation of S and S', where S' is the reverse of the sequence S (i.e., the sequence S read from end to start). Let T be the sequence resulting from concatenating sequences S^* corresponding to chosen traversals for all inputs, putting 0 between two such consecutive sequences as a separation marker.

We define the execution of T on a graph G starting from node s as follows. The agent starts at s and follows consecutive port numbers indicated by the sequence T, disregarding zeroes. It may happen that at some point the execution is impossible for one of the two reasons:

either the next term of T does not correspond to the current entry port (the agent left a node by port a, traversed an edge and enters the next node by port b but the next term of T is different from b), or the next term of T indicates a port number larger than the degree of the current node, hence a nonexisting port. In both these situations we say that the traversal is *aborted*. Instead of continuing it, the agent goes back to s (returning via the same path indicated by port numbers) and resumes execution of T from the next term 0 (i.e., executing the part of T corresponding to the traversal of the next input). This is done until the entire sequence T is executed in this manner.

Let $f : N \longrightarrow N$ be a function such that $f(x+1) > f(x) \cdot |T|$, where $|T|$ is the length of the sequence T. An example of such a function is $f(x) = (|T|+1)^x$. Let $num(L)$ denote the integer whose binary representation is the sequence L. The algorithm for an agent with label L starting from node s in a graph G with port labeling is the following.

- Execute $f(num(L))$ times the sequence T and stop.

It remains to show that this algorithm performed by both agents in a given graph G_0 guarantees rendezvous. Suppose that agents have labels L_1 and L_2, with $num(L_1) > num(L_2)$. Call the agent with label L_i the ith agent. We have $f(num(L_1)) > f(num(L_2)) \cdot |T|$. Consider a time segment corresponding to one execution of the sequence T by the first agent. During this execution the agent must traverse all edges of graph G_0 because a part of the sequence T corresponds to the input composed of G_0 with the given port labeling and of the starting point of this agent. Hence either rendezvous is achieved or the second agent must have executed at least one nonzero term of T during this time segment. After $f(num(L_1)) - 1$ executions of the sequence T by the first agent one of two events must happen. Either rendezvous is achieved or the second agent must have executed at least $f(num(L_1)) - 1$ nonzero terms. Suppose that the second event takes place (otherwise we are done). However $f(num(L_1)) - 1 \geq f(num(L_2)) \cdot |T|$. Hence after $f(num(L_1)) - 1$ executions of the sequence T by the first agent, the second agents must have completed all its $f(num(L_2))$ executions of the sequence T. Hence it must have stopped. Now the first agent has the last execution of the sequence T remaining, while the second agent does not move any more. Since during this last execution the first agent must traverse all edges of graph G_0, rendezvous must occur. □