

# Delays Induce an Exponential Memory Gap for Rendezvous in Trees\*

Pierre Fraigniaud  
CNRS and University Paris Diderot  
France  
pierre.fraigniaud@liafa.jussieu.fr

Andrzej Pelc  
Université du Québec en Outaouais  
Canada  
andrzej.pelc@uqo.ca

## ABSTRACT

The aim of rendezvous in a graph is meeting of two mobile agents at some node of an unknown anonymous connected graph. The two identical agents start from arbitrary nodes in the graph and move from node to node with the goal of meeting. In this paper, we focus on rendezvous in trees, and, analogously to the efforts that have been made for solving the exploration problem with compact automata, we study the size of memory of mobile agents that permits to solve the rendezvous problem deterministically.

We first show that if the delay between the starting times of the agents is *arbitrary*, then the lower bound on memory required for rendezvous is  $\Omega(\log n)$  bits, even for the line of length  $n$ . This lower bound meets a previously known upper bound of  $O(\log n)$  bits for rendezvous in arbitrary trees of size at most  $n$ . Our main result is a proof that the amount of memory needed for rendezvous *with simultaneous start* depends essentially on the number  $\ell$  of leaves of the tree, and is exponentially less impacted by the number  $n$  of nodes. Indeed, we present two identical agents with  $O(\log \ell + \log \log n)$  bits of memory that solve the rendezvous problem in all trees with at most  $n$  nodes and at most  $\ell$  leaves. Hence, for the class of trees with polylogarithmically many leaves, there is an exponential gap in minimum memory size needed for rendezvous between the scenario with arbitrary delay and the scenario with delay zero. Moreover, we show that our upper bound is optimal by proving that  $\Omega(\log \ell + \log \log n)$  bits of memory is required for rendezvous, even in the class of trees with degrees bounded by 3.

---

\*Part of this work was done during this author's visit at the Research Chair in Distributed Computing of the Université du Québec en Outaouais. The first author is supported by the ANR projects ALADDIN and PROSE, and by the INRIA project GANG. The second author is supported in part by NSERC discovery grant and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SPAA'10, June 13–15, 2010, Thira, Santorini, Greece.  
Copyright 2010 ACM 978-1-4503-0079-7/10/06 ...\$10.00.

## Categories and Subject Descriptors

G.2.2 [Discrete Mathematics]: Graph Theory—*Network problems*; F.1.0 [Theory of Computation]: Computation by Abstract Devices—*General*

## General Terms

Algorithms, Theory.

## Keywords

rendezvous, exploration, robots, mobile entities, abstract state machine.

## 1. INTRODUCTION

The rendezvous in a network [1, 4] is the following task. Two identical mobile agents, initially located in two nodes of the network, move along links from node to node, and eventually have to get to the same node at the same time. The network is modeled as an undirected connected graph, and agents traverse links in synchronous rounds. In this paper we consider deterministic rendezvous, and seek rendezvous protocols that do not rely on the knowledge of node labels, and can work in anonymous graphs as well (cf. [3]). This assumption is motivated by the fact that, even when nodes are equipped with distinct labels, agents may be unable to perceive them or nodes may refuse to reveal their labels, e.g., due to security reasons. (Note also that if nodes of the graph are labeled using distinct names, then agents can meet at some a priori agreed node, and rendezvous reduces to graph exploration).

Obviously, deterministic rendezvous is not possible if the initial positions of the two agents are symmetric, i.e., if there is an automorphism of the graph that carries one node on the other. Hence, lots of efforts have been dedicated to the study of the feasibility of rendezvous, and to the time required to achieve this task, when feasible. For instance, deterministic rendezvous with agents equipped with tokens used to mark nodes was considered, e.g., in [26]. Deterministic rendezvous of agents equipped with unique labels was discussed in [12, 13, 24]. (In this latter scenario, symmetry is broken by the use of the different labels of agents, and thus rendezvous is sometimes possible even for symmetric initial positions of the agents). Recently, rendezvous using variants of Universal Traversal Sequences was investigated in [30]. Surprisingly though, as opposed to what was done for the graph exploration problem (see, e.g., [10, 18, 23, 28]), or for other tasks such as routing (see, e.g., [16, 17]), very little is known on the amount of memory required by

the agents for achieving rendezvous. Up to our knowledge, the only existing results prior to this work are dedicated to rendezvous in rings and trees. Memory needed for randomized rendezvous in the ring is discussed, e.g., in [25]. Memory needed for deterministic rendezvous in trees is discussed in [19] where it is proved that the minimum memory size guaranteeing rendezvous in all trees of size at most  $n$  is  $\Theta(\log n)$  bits, even if the two agents start at different times, and at least  $\Omega(\log \log n)$  bits, even if the two agents start at the same time.

The aim of this paper is to determine the space complexity of rendezvous in trees.

## 1.1 Our results

We first show that if the delay between the starting times of the agents is arbitrary, then the lower bound on memory required for rendezvous is  $\Omega(\log n)$  bits, even for the line of length  $n$ . This lower bound matches the upper bound from [19] in the case of arbitrary trees.

Our main result is a proof that the amount of memory needed for rendezvous *with simultaneous start* in trees depends essentially on the number  $\ell$  of leaves of the tree, and is exponentially less impacted by the number  $n$  of nodes. Indeed, we show two identical agents with  $O(\log \ell + \log \log n)$  bits of memory that solve the rendezvous problem in all trees with  $n$  nodes and  $\ell$  leaves. Hence, for the class of trees with polylogarithmically many leaves, there is an exponential gap in minimum memory size needed for rendezvous between the scenario with arbitrary delay and the scenario with delay zero.

Moreover, we show that the size  $O(\log \ell + \log \log n)$  of memory is optimal, even in the class of trees with degrees bounded by 3. More precisely, for infinitely many integers  $\ell$ , we show a class of arbitrarily large trees with maximum degree 3 and with  $\ell$  leaves, for which rendezvous requires  $\Omega(\log \ell)$  bits of memory. This lower bound, together with a result from [19] showing that  $\Omega(\log \log n)$  bits of memory are required for rendezvous with simultaneous start in the line of length  $n$ , implies that our upper bound  $O(\log \ell + \log \log n)$  cannot be improved, even for trees with maximum degree 3.

## 1.2 Other related work

The rendezvous problem was first mentioned in [29]. Authors investigating rendezvous (cf. [3] for an extensive survey) considered either the geometric scenario (rendezvous in an interval of the real line, see, e.g., [8, 9, 20]), or in the plane, see, e.g., [6, 7]). Many papers, e.g., [1, 2, 5, 8, 22] study the probabilistic setting: inputs and/or rendezvous strategies are random.

A natural extension of the rendezvous problem is that of gathering [15, 22, 27, 31], when more than 2 agents have to meet in one location. In [32] the authors considered rendezvous of many agents with unique labels. The impact of memory size on the feasibility of the related task of tree exploration, for trees with unlabeled nodes, has been studied in [14, 21]. In [14] the authors showed that no agent can explore with termination all trees of bounded degree and that memory of size  $O(\log^2 n)$  bits is enough to explore all trees of size  $n$  and return to the starting node. In [21] it was shown that the latter task can be accomplished by an agent with memory of size  $O(\log n)$  bits.

## 2. FRAMEWORK AND PRELIMINARIES

We consider trees whose nodes are unlabeled, and edges incident to a node  $v$  have distinct labels in  $\{0, \dots, d-1\}$ , where  $d$  is the degree of  $v$ . Thus every undirected edge  $\{u, v\}$  has two labels, which are called its *port numbers*<sup>1</sup> at  $u$  and at  $v$ . Port numbering is *local*, i.e., there is no relation between port numbers at  $u$  and at  $v$  (we do not assume any sense of direction, of any kind). A pair of distinct nodes  $u, v$  of a tree is called *symmetric* if there exists an automorphism of the tree preserving port numbering, that carries one node on the other. Recall that an automorphism of the tree is a bijection  $f : V \rightarrow V$ , where  $V$  is the set of nodes of the tree, such that for any  $w, w' \in V$ ,  $w$  is adjacent to  $w'$  if and only if  $f(w)$  is adjacent to  $f(w')$ . It preserves port numbering if for any  $w, w' \in V$ , the port number corresponding to edge  $\{w, w'\}$  at node  $w$  is equal to the port number corresponding to edge  $\{f(w), f(w')\}$  at node  $f(w)$ . So  $u$  and  $v$  are symmetric if there exists an automorphism  $f$  preserving port numbering, and such that  $f(u) = v$ .

We consider mobile agents traveling in trees with locally labeled ports. The tree and its size are a priori unknown to the agents. We first define precisely an individual agent. An agent is an abstract state machine  $\mathcal{A} = (S, \pi, \lambda, s_0)$ , where  $S$  is a set of states among which there is a specified state  $s_0$  called the *initial state*,  $\pi : S \times \mathbb{Z}^2 \rightarrow S$ , and  $\lambda : S \rightarrow \mathbb{Z}$ . Initially the agent is at some node  $u_0$  in the initial state  $s_0 \in S$ . The agent performs actions in rounds measured by its internal clock. Each action can be either a move to an adjacent node or a null move resulting in remaining in the currently occupied node. State  $s_0$  determines a natural number  $\lambda(s_0)$ . If  $\lambda(s_0) = -1$  then the agent makes a null move (i.e., remains at  $u_0$ ). If  $\lambda(s_0) \geq 0$  then the agent leaves  $u_0$  by port  $\lambda(s_0)$  modulo the degree of  $u_0$ . When incoming to a node  $v$  in state  $s \in S$ , the behavior of the agent is as follows. It reads the number  $i$  of the port through which it entered  $v$  and the degree  $d$  of  $v$ . The pair  $(i, d) \in \mathbb{Z}^2$  is an input symbol that causes the transition from state  $s$  to state  $s' = \pi(s, (i, d))$ . If the previous move of the agent was null, (i.e., the agent stayed at node  $v$  in state  $s$ ) then the pair  $(-1, d) \in \mathbb{Z}^2$  is the input symbol read by the agent, that causes the transition from state  $s$  to state  $s' = \pi(s, (-1, d))$ . In both cases  $s'$  determines an integer  $\lambda(s')$ , which is either  $-1$ , in which case the agent makes a null move, or a non negative integer indicating a port number by which the agent leaves  $v$  (this port is  $\lambda(s') \bmod d$ ). The agent continues moving in this way, possibly infinitely.

Since we consider the rendezvous problem for identical agents, we assume that agents are copies  $A$  and  $A'$  of the same abstract state machine  $\mathcal{A}$ , starting at two distinct nodes  $v_A$  and  $v_{A'}$ , called the *initial positions*. We will refer to such identical machines as a *pair of agents*. It is assumed that the internal clocks of a pair of agents tick at the same rate. The clock of each agent starts when the agent starts executing its actions. Agents start from their initial position with *delay*  $\theta \geq 0$ , controlled by an adversary. This means that the later agent starts executing its actions  $\theta$  rounds after the first agent. Agents do not know which of them is first and what is the value of  $\theta$ . We seek agents with small memory, measured by the number of states of the corre-

<sup>1</sup>In the absence of port numbers, rendezvous is usually impossible, as the adversary may prevent an agent from taking some edge incident to the current node.

sponding automaton, or equivalently by the number of bits on which these states are encoded. An automaton with  $K$  states requires  $\Theta(\log K)$  bits of memory.

Initial positions forming a symmetric pair of nodes are crucial for our considerations. Indeed, if the initial positions are not a symmetric pair, then there exists a pair of agents that can meet in a given tree, for any delay  $\theta$ , and, if the initial position is symmetric, then meeting is impossible for any pair of agents, for  $\theta = 0$ . We say that a pair of agents solve the rendezvous problem *with arbitrary delay* (resp. *with simultaneous start*) in a class of trees, if, for any tree in this class and for any initial positions that are not symmetric, both agents are eventually in the same node of the tree in the same round, regardless of the starting rounds of the agents (resp. provided that they start in the same round). Hence, in particular, solving the rendezvous problem with simultaneous start means achieving rendezvous whenever it is possible.

Consider any tree  $T$  and the following sequence of trees constructed recursively:  $T_0 = T$ , and  $T_{i+1}$  is the tree obtained from  $T_i$  by removing all its leaves.  $T' = T_j$  for the smallest  $j$  for which  $T_j$  has at most two nodes. If  $T'$  has one node, then this node is called the *central node* of  $T$ . If  $T'$  has two nodes, then the edge joining them is called the *central edge* of  $T$ . A tree  $T$  with port labels is called *symmetric*, if there exists a non-trivial automorphism  $f$  of the tree (i.e., an automorphism  $f$  such that  $f(u) \neq u$ , for some  $u \in V$ ) preserving port numbering. If a tree with port labels has a central node, then it cannot be symmetric. In a non-symmetric tree, every pair of nodes is non-symmetric, hence rendezvous is feasible for all initial positions of agents.

The following statement is an easy consequence of the techniques and results from [21].

**FACT 2.1.** *There exists an agent accomplishing the following task in an arbitrary tree, using  $O(\log m)$  bits of memory in trees with at most  $m$  nodes: it finds the number  $m$  of nodes in the tree, and*

- *if the tree has a central node, then the agent goes to this node, and stops;*
- *if the tree has a central edge but is not symmetric, then, for any initial position, the agent goes to the same extremity of the central edge, and stops; moreover, it knows which port of this extremity corresponds to the central edge;*
- *if the tree is symmetric, then the agent goes to one of the extremities of the central edge; moreover, to whichever extremity it goes, the agent knows which port of this extremity corresponds to the central edge; finally, the number of rounds used to go to this extremity of the central edge differs by at most  $m$ , for any two initial positions of the agent in an  $m$ -node tree.*

In the sequel, the procedure accomplishing the above task will be called Procedure **recognize**.

### 3. RENDEZVOUS WITH ARBITRARY DELAY

It was proved in [19] that rendezvous with arbitrary delay can be accomplished in  $n$ -node trees using  $O(\log n)$  bits of memory. It was also observed that rendezvous requires

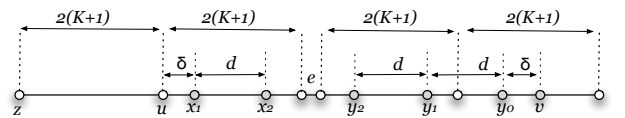
$\Omega(\log n)$  bits of memory in arbitrarily large trees with  $2n+1$  nodes and maximum degree  $n$ . The lower bound examples were trees  $T_n$  consisting of two nodes  $u$  and  $v$  of degree  $n$ , both linked to a common node  $w$ , and to  $n-1$  leaves. However, these trees have linear degree and the reason for the logarithmic memory requirement is simply that agents with smaller memory are incapable of having an output function  $\lambda$  with range of linear size, and thus the adversary can place agents in nodes  $u$  and  $v$  and distribute ports in such a way that none of the agents can ever get to node  $w$ , which makes rendezvous infeasible.

This example leaves open the question if rendezvous with sub-logarithmic memory is possible, e.g., in all trees with constant maximum degree. It turns out that if the delay is arbitrary, this is not the case: rendezvous requires logarithmic memory even for the class of lines.

**THEOREM 3.1.** *Rendezvous with arbitrary delay in the  $n$ -node line requires agents with memory  $\Omega(\log n)$ .*

**PROOF.** Let  $k$  be the number of memory bits of the agent and  $K = 2^k$  be its number of states. Place one agent at some node  $u$  of the infinite line where each edge has the same port number at its two extremities. In any interval of length  $K+1$  there exist two nodes at which the agent is in the same state. Let  $x_1$  be the first node of the trajectory of the agent in which this happens and let  $s$  be the state of the agent at  $x_1$ . Let  $x_2$  be the second node of the trajectory of the agent at which the agent is in state  $s$ . Let  $\delta$  be the distance between  $u$  and  $x_1$  and let  $d$  be the distance between  $x_1$  and  $x_2$ .

We construct the following instance of the rendezvous problem (see Fig. 1). The line is of length  $8(K+1)+1$ . Let  $e$  be the central edge of this line. Assign number 0 to ports leading to edge  $e$  from both its extremities, and assign other port labels so that ports leading to any edge at both its extremities get the same number 0 or 1. (This is equivalent to 2-edge-coloring of the line.) Let  $z$  be the endpoint of the line, for which  $x_1$  is between  $z$  and  $x_2$ . Let  $y_1$  and  $y_2$  be symmetric images of  $x_1$  and  $x_2$ , respectively, according to the axis of symmetry of the line. Let  $y_0$  be the node distinct from  $y_2$ , at distance  $d$  from  $y_1$ . Let  $v$  be the node at distance  $\delta$  from  $y_0$ , such that the vectors  $[x_1, u]$  and  $[y_0, v]$  have opposite directions. The other agent is placed at node  $v$ .



**Figure 1: Construction in the proof of Theorem 3.1.**

Let  $t_1$  be the number of rounds that the agent starting at  $u$  takes to reach  $x_1$  in state  $s$ . Let  $t_2$  be the number of rounds that the agent starting at  $v$  takes to reach  $y_1$  in state  $s$ . Let  $\theta = t_2 - t_1$ . The adversary delays the agent starting at  $u$  by  $\theta$  rounds. Hence the agent starting at  $u$  reaches  $x_1$  at the

<sup>2</sup>Here we say that the agent *reaches* node  $w$  in state  $s$ , if  $s$  is the state of the agent in  $w$  after the application of the transition function; Hence, once the agent has reached  $w$  in state  $s$ , it leaves  $w$  by port  $\lambda(s)$ .

same time  $t$  and in the same state as the agent starting at  $v$  reaches  $y_1$ . The points  $x_1$  and  $y_1$  are symmetric positions, hence rendezvous is impossible after time  $t$ . Before time  $t$  the two agents were on different sides of edge  $e$ , in view of  $\delta + d \leq 2(K + 1)$ , hence rendezvous did not occur, although the initial positions of the agents are not a symmetric pair. The size of the line is  $O(K) = O(2^k)$ , which concludes the proof.  $\square$

Together with the logarithmic upper bound from [19], the result above completely solves the problem of determining the minimum memory of the agents permitting rendezvous with arbitrary delay. Hence in the rest of the paper we concentrate on rendezvous with simultaneous start, thus assuming that the delay  $\theta = 0$ .

## 4. RENDEZVOUS WITH SIMULTANEOUS START

### 4.1 Upper bound

It turns out that the size of memory needed for rendezvous with simultaneous start depends on two parameters of the tree: the number  $n$  of nodes and the number  $\ell$  of leaves. In fact we show that rendezvous in trees with  $n$  nodes and  $\ell$  leaves can be done using only  $O(\log \ell + \log \log n)$  bits of memory. Thus, for trees with polylogarithmically many leaves,  $O(\log \log n)$  bits of memory are enough. In view of Theorem 3.1, this shows an exponential gap in the minimum memory size needed for rendezvous between the scenarios with arbitrary delay and with delay zero.

**THEOREM 4.1.** *There is a pair of identical agents solving rendezvous with simultaneous start in all trees, and using, for any integers  $n$  and  $\ell$ ,  $O(\log \ell + \log \log n)$  bits of memory in trees with at most  $n$  nodes and at most  $\ell$  leaves.*

The rest of the subsection is dedicated to the proof of Theorem 4.1. Let  $T$  be any tree. Let  $T'$  be the *contraction* of  $T$ , that is the tree obtained from  $T$  by replacing every path<sup>3</sup> in  $T$  joining two nodes of degree different from 2 by an edge (the ports of this edge correspond to the ports at both extremities of the contracted path). Notice that if  $T$  has  $\ell$  leaves, then its contraction  $T'$  has at most  $2\ell - 1$  nodes. Our rendezvous algorithm uses Procedure **recognize**, mentioned in Section 2, as a subroutine for solving the problem in the simple case where the contraction tree  $T'$  is non symmetric, i.e., either there is a central node, or there is a central edge and the two edge-labeled trees obtained by removing the central edge in  $T'$  are not isomorphic. (The isomorphism must preserve both the structure of the trees, and the labeling of the edges). Indeed, in a non symmetric case, Fact 2.1 states that two agents performing Procedure **recognize** will eventually identify a single node of  $T'$ , and hence a single node of  $T$  as well, at which they will rendezvous. The difficult and more challenging case is when the contraction tree  $T'$  has a central edge with two non distinguishable extremities, in which case the ability to solve the rendezvous problem depends on the large tree  $T$ , and the initial positions of the two agents. Identifying whether these two positions are symmetric or not in  $T$ , and, in the latter

<sup>3</sup>Here, by *path* we mean a sequence of adjacent nodes of degree 2, all pairwise distinct.

case, achieving rendezvous, is complicated by the constraint that the agents must use sub-logarithmic memory when  $\ell$  is small. The main part of the proof will be dedicated to describing how this task can actually be achieved in a memory efficient manner.

Rendezvous proceeds in two phases. During the first phase, each of the two agents starts by executing procedure **recognize** in  $T$ , ignoring the degree-2 nodes. That is, protocol **recognize** is modified so that whenever an agent enters a degree-2 node through port  $i \in \{0, 1\}$  in some state  $s$ , it will leave that node at the next round by port  $(i + 1) \bmod 2$ , in the same state  $s$ . More precisely, let  $s_0$  be the initial state of an agent executing **recognize**. Our modified agent starts in a state  $s_0^*$ , and leaves the initial node through port 0. The agent proceeds in states  $s_0^*$  until it enters a node of degree different from 2. At such a node, independently of the incoming port number, the agent enters state  $s_0$ . From that point on, the agent performs **recognize** by ignoring degree-2 nodes. We call **recognize-bis** the protocol **recognize** modified in this way. Observe that, in trees with no nodes of degree 2, the two protocols **recognize** and **recognize-bis** perform similarly (only the beginnings of the two protocols slightly differ). Hence, protocols **recognize** and **recognize-bis** perform similarly in  $T'$ . Formally, the following holds.

**CLAIM 4.1.** *The states at nodes of degrees different from 2 of an agent performing **recognize-bis** in  $T$  starting from some node  $u$  are identical to the states of an agent performing **recognize** in  $T'$  starting from node  $v$ , where  $v$  is the first node of degree different from 2 reached when leaving  $u$  through port 0.*

Using this claim, rendezvous in  $T$  is achieved as follows.

- If there is a central node in  $T'$ , then that node is identified by protocol **recognize** (cf. Fact 2.1). Hence, from Claim 4.1, the corresponding node in  $T$  is identified by **recognize-bis**. Rendezvous is achieved by waiting for the other agent at that node.
- Similarly, if there is a central edge in  $T'$ , and the tree  $T'$  is not symmetric, then one extremity of that edge is identified by protocol **recognize** (cf. Fact 2.1). Hence, from Claim 4.1, the corresponding node in  $T$  is identified by **recognize-bis**. Rendezvous is achieved by waiting for the other agent at that node.

We are left with the case in which the contraction tree  $T'$  is symmetric. In this case, according to Fact 2.1, the two agents may end up in two different nodes of  $T$  after having performed **recognize-bis**. These two nodes correspond to the two extremities of the central edge of the symmetric tree  $T'$ . For instance, in the  $n$ -node path with an odd number of edges, the two agents may end up in the two symmetric extremities of the path after having performed **recognize-bis**. Also, in the binomial tree with  $n$ -nodes (cf.[11]), the two agents may end up in the two symmetric roots of the two binomial subtrees of  $T$  with  $n/2$  nodes. Still, we prove that rendezvous is possible with little memory assuming that the two initial positions of the agents were not symmetric in  $T$ . Actually, the first of the two key ingredients in our proof is showing how rendezvous can be achieved in the path using agents with  $O(\log \log n)$  bits of memory.

More precisely, in the lemma below, we consider *blind* agents in paths, that is agents that ignore port labels. More

precisely, when entering a node, such an agent can just distinguish between the incoming edge and the other edge (if any). We use blind agents for an explicit reason: the NSC for rendezvous in path by blind agents can easily be characterized. Specifically, let  $P = (v_1, \dots, v_m)$  be an  $m$ -node path, and consider two identical blind agents initially located at nodes  $v_a$  and  $v_b$ ,  $a < b$ . Rendezvous using blind agents is possible if and only if  $m$  odd, or  $m$  even and  $a - 1 \neq m - b$ .

Of course, a standard agent can simulate the behavior of a blind agent. When applying the lemma below with standard agents, we will make sure that the starting positions  $v_a$  and  $v_b$  are such that rendezvous is achievable even with blind agents.

**LEMMA 4.1.** *There exists a pair of identical blind agents accomplishing rendezvous with simultaneous start in all paths, whenever it is possible, and using  $O(\log \log m)$  bits of memory in paths with at most  $m$  nodes.*

**PROOF.** Let  $P = (v_1, \dots, v_m)$  be an  $m$ -node path, and consider two identical blind agents initially located at nodes  $v_a$  and  $v_b$ ,  $a < b$ . To achieve rendezvous, the two agents perform a sequence of traversals of  $P$ , executed at lower and lower speeds, aiming at eventually meeting each other at some node. More precisely, for an integer  $s \geq 1$ , a traversal of the path is performed at speed  $1/s$ , if the agent remains idle  $s - 1$  rounds before traversing any edge. For instance, traversing  $P$  from  $v_1$  to  $v_m$  at speed  $1/s$  requires  $(m - 1)s$  rounds. Our rendezvous algorithm for the line, called **prime**, performs as follows.

**Begin**  
start in arbitrary direction;  
move at speed 1 until reaching one extremity  
of the path;  
 $p \leftarrow 2$ ;  
**While** no rendezvous **do**  
traverse the entire path twice, at speed  $1/p$ ;  
 $p \leftarrow$  smallest prime larger than  $p$ ;  
**End**

We now prove that, whenever rendezvous is possible for blind agents (i.e., when  $m$  odd, or  $m$  even and  $a - 1 \neq m - b$ ), the two agents meet before the  $p$ th iteration of the loop, for  $p = O(\log n)$ . Let  $p_j$  be the  $j$ th prime number ( $p_1 = 2$ ). Hence the speed of each agent at the  $j$ th execution of the loop is  $1/p_j$ . If rendezvous has not occurred during the  $j$ th execution of the loop, then the two agents have crossed the same edge, say  $e = \{v_c, v_{c+1}\}$ , at the same time  $t$ , in opposite directions. This can occur if, for instance, the agent initially at  $v_a$  moves to node  $v_1$ , traverses twice the path at successive speeds  $p_1, \dots, p_{j-1}$ , and,  $c p_j$  rounds after having eventually started walking at speed  $p_j$ , traverses the edge  $e$  at time  $t$ , while the other agent initially at  $v_b$  moves to  $v_m$ , traverses twice the path at successive speeds  $p_1, \dots, p_{j-1}$ , and,  $(m - c)p_j$  rounds after having eventually started walking at speed  $p_j$ , traverses the same edge  $e$  in the other direction at the same time  $t$ . In fact, there are four cases to consider, depending on the two starting directions of the two agents: towards  $v_1$  or towards  $v_m$ . From these four cases, we get that one of the following four equalities must hold (the first one corresponds to the previously described scenario:  $v_a$  moves towards  $v_1$  while  $v_b$  moves towards  $v_m$ ):

$$\begin{aligned} t &= (a - 1) + 2(m - 1) \sum_{i=1}^{j-1} p_i + c p_j \\ &= (m - b) + 2(m - 1) \sum_{i=1}^{j-1} p_i + (m - c)p_j \end{aligned}$$

or

$$\begin{aligned} t &= (a - 1) + 2(m - 1) \sum_{i=1}^{j-1} p_i + (m - 1)p_j + (m - c)p_j \\ &= (b - 1) + 2(m - 1) \sum_{i=1}^{j-1} p_i + c p_j \end{aligned}$$

or

$$\begin{aligned} t &= (m - a) + 2(m - 1) \sum_{i=1}^{j-1} p_i + (m - c)p_j \\ &= (m - b) + 2(m - 1) \sum_{i=1}^{j-1} p_i + (m - 1)p_j + c p_j \end{aligned}$$

or

$$\begin{aligned} t &= (m - a) + 2(m - 1) \sum_{i=1}^{j-1} p_i + (m - c)p_j \\ &= (b - 1) + 2(m - 1) \sum_{i=1}^{j-1} p_i + c p_j \end{aligned}$$

Therefore we get that

$$p_j \text{ divides } |a - b|, \text{ or } p_j \text{ divides } |m - (a + b) + 1|.$$

As a consequence, since the  $p_i$ 's are primes, we get that if the two agents have not met after the  $j$ th execution of the loop, then

$$\prod_{i \in \mathcal{I}} p_i \text{ divides } |a - b| \quad \text{and} \quad \prod_{i \in \mathcal{J}} p_i \text{ divides } |m - (a + b) + 1|$$

where  $\mathcal{I} \cup \mathcal{J} = \{1, \dots, j\}$ . Therefore, since the  $p_i$ 's are primes,  $\prod_{i=1}^j p_i$  divides  $|a - b| \cdot |m - (a + b) + 1|$ . Hence, if rendezvous is feasible, it must occur at or before the  $j$ th execution of the loop, where  $j$  is the largest index such that  $\prod_{i=1}^j p_i$  divides  $|a - b| \cdot |m - (a + b) + 1|$ . Thus it must occur at or before the  $j$ th execution of the loop, where  $j$  is the largest index such that  $\prod_{i=1}^j p_i \leq m^2$ .

Let  $\pi(x)$  be the number of prime numbers smaller than or equal to  $x$ . On the one hand, we have  $\prod_{i=1}^j p_i \geq 2^{\pi(p_j)}$ . Hence, rendezvous must occur at or before the  $j$ th execution of the loop, where  $j$  is the largest index such that  $2^{\pi(p_j)} \leq m^2$ , i.e.,  $\pi(p_j) \leq 2 \log m$ . On the other hand, from the Prime Number Theorem we get that  $\pi(x) \sim x / \ln(x)$ , i.e.,

$$\lim_{x \rightarrow \infty} \frac{\pi(x)}{x / \ln(x)} = 1.$$

Hence, for  $m$  large enough,  $\pi(x) \geq x / (2 \ln(x))$ . Thus rendezvous must occur at or before the  $j$ th execution of the loop, where  $j$  is the largest index such that  $p_j / \ln p_j \leq 4 \log m$ .

From the above, we get that (1) rendezvous must occur whenever it is feasible, and (2) it occurs at or before the  $j$ th execution of the loop, where  $\log p_j \leq O(\log \log m)$ . Since the next prime  $p$  can be found using  $O(\log p)$  bits, e.g., by exhaustive search, we get that **prime** performs rendezvous using agents with  $O(\log \log m)$  bits of memory.  $\square$

The (blind) agents described in Lemma 4.1 perform a protocol called **prime**. This protocol uses the infinite sequence of prime numbers. We denote by **prime**( $i$ ) the protocol **prime** modified so that it stops after having considered the  $i$ th prime number.

We now come back to our general rendezvous protocol in trees (with port numbers). If the agents detect symmetry of the contraction tree, a second phase begins. Let  $\nu = 2x$  be the number of nodes in the contraction tree  $T'$ . (We have  $\nu$  even since  $T'$  is symmetric with respect to its central edge). We define a (non-simple) path called the *rendezvous path*, denoted by  $P$ , that will be used by the agents to rendezvous using protocol **prime**. To define  $P$ , let  $u$  and  $v$  be the two extremities of the path in  $T$  corresponding to the central edge in  $T'$ . This path is called the *central path*, and is denoted

by  $C$ . Abusing notation,  $C$  will also be used as a shortcut for the instruction: “traverse  $C$ ”.

Let  $\text{BW}$  (for “basic walk”) be the instruction of performing the following actions: leave by port 0, and, perpetually, whenever entering a degree- $d$  node by port  $i \in \{0, \dots, d-1\}$ , leave that node by port  $(i+1) \bmod d$ . Similarly, let  $\text{CBW}$  (for “counter basic walk”), be the instruction of performing the following: leave by the port used to enter the current node at the previous step, and, perpetually, whenever entering a degree- $d$  node by port  $i$ , leave that node by port  $(i-1) \bmod d$ . For  $j \geq 1$ , let  $\text{BW}(j)$  (resp.,  $\text{CBW}(j)$ ) be the instruction to execute  $\text{BW}$  (resp.,  $\text{CBW}$ ) until  $j$  nodes of degree different from 2 have been visited. Let  $B_u$  (resp.,  $B_v$ ) be the path corresponding to the execution of  $\text{BW}(2(\nu-1))$  from  $u$  (resp., from  $v$ ). Note that a node can be visited several times by the walk, and thus neither  $B_u$  nor  $B_v$  are simple. Note also that since  $T'$  has  $\nu$  nodes, it has  $\nu-1$  edges, and thus both  $B_u$  and  $B_v$  are closed paths, i.e., their extremities are  $u$  and  $v$ , respectively. Let  $\overline{B}_u$  (resp.,  $\overline{B}_v$ ) be the path corresponding to the execution of  $\text{CBW}(2(\nu-1))$  from  $u$  (resp., from  $v$ ). We define

$$P = (B_u | C_{u \rightarrow v} | \overline{B}_v | C_{v \rightarrow u})^{5\ell} | (B_u | C_{u \rightarrow v} | \overline{B}_v)$$

where “ $|$ ” denotes the concatenation of paths,  $C_{u \rightarrow v}$  (resp.,  $C_{v \rightarrow u}$ ) denotes the path  $C$  traversed from  $u$  to  $v$  (resp., from  $v$  to  $u$ ), and, for a closed path  $Q$ ,  $Q^\alpha$  denotes  $Q$  concatenated with itself  $\alpha$  times.

The path  $P$  is well defined. Indeed, the sequence

$$B_u | C_{u \rightarrow v} | \overline{B}_v | C_{v \rightarrow u}$$

leads back to node  $u$ . Also, the two extremities of the path are  $u$  and  $v$ . Now, the agents have no clue whether they are standing at  $u$  or at  $v$ . Nevertheless, we have the following.

**CLAIM 4.2.** *Starting from an extremity  $u$  or  $v$  of the central path  $C$ , an agent performing the sequence of instructions*

$$\left( \text{BW}(2(\nu-1)), C, \text{CBW}(2(\nu-1)), C \right)^{5\ell}$$

followed by

$$\text{BW}(2(\nu-1)), C, \text{CBW}(2(\nu-1))$$

traverses the path  $P$  from one of its extremities to the other.

Before establishing the claim, note that instructions  $\text{BW}(2(\nu-1))$  and  $\text{CBW}(2(\nu-1))$  are meaningful, since agents can have counters of size  $O(\log \ell)$  bits, and they know  $\nu$  in view of Fact 2.1. To establish the claim, it suffices to notice that the path  $\overline{P}$  reverse to  $P$  is given by

$$\begin{aligned} \overline{P} &= (B_v | C_{v \rightarrow u} | \overline{B}_u) | (C_{u \rightarrow v} | B_v | C_{v \rightarrow u} | \overline{B}_u)^{5\ell} \\ &= (B_v | C_{v \rightarrow u} | \overline{B}_u | C_{u \rightarrow v})^{5\ell} | (B_v | C_{v \rightarrow u} | \overline{B}_u). \end{aligned}$$

The two agents will use protocol **prime** along the path  $P$  to achieve rendezvous. However, to make sure that rendezvous succeeds, the two agents must not start **prime** simultaneously at the two extremities of  $P$ , in order to break symmetry. Unfortunately, this requirement is not trivial to satisfy. Indeed, Fact 2.1 guarantees that the delay between the times the two agents reach the two extremities of  $C$  (and thus of  $P$  as well) does not exceed  $n$ , but no guarantee can be given for the minimum delay, which could be zero. This is because the delay does not depend on the tree  $T'$ , but on the tree  $T$ . Hence two agents starting simultaneously in  $T$

```

Begin
  for consecutive values  $i \geq 1$  do
    /* try rendezvous */
    for  $j = 0, 1, \dots, 2(\nu-1)$  do
      perform  $\text{BW}(j)$ ;
      perform  $\text{CBW}(j)$ ; /* back to init. position */
      perform prime( $i$ ) on the rendezvous path  $P$ ;
    /* reset */
    go to the other extremity of the central path  $C$ ;
    for  $j = 0, 1, \dots, 2(\nu-1)$  do
      perform  $\text{BW}(j)$ ;
      perform  $\text{CBW}(j)$ ; /* back to init. position */
      return to the original extremity of the
      central path  $C$ ;
End

```

**Figure 2: Second phase of the rendezvous (performed when the contraction tree is symmetric).**

may actually finish the first phase of our protocol (i.e., the execution of **recognize-bis**) at the same time, even if  $T$  is not symmetric, and even if  $T$  is symmetric but the starting positions were not symmetric. The second key ingredient in our proof is a technique guaranteeing eventual desynchronization of the two agents. A high level description of this technique is summarized in Figure 2. We describe this technique in detail below.

The outer loop of the protocol in Figure 2 states how many consecutive prime numbers the protocol will test while performing **prime** along the path  $P$ . Performing **prime**( $i$ ) for successive values of  $i$ , instead of just **prime**, is for avoiding a perpetual execution of **prime** in the case when the two agents started the execution of phase 2 at the same time from the two extremities of  $P$ . For every number  $i \geq 1$  of primes to be used in **prime**, the protocol performs two inner loops. The first one is an attempt to achieve rendezvous along  $P$ , while the second one is used to upper bound the delay between the two agents at the end of the outer loop, in order to guarantee that the next execution of the outer loop will start with a delay between the two agents that does not exceed  $n$ .

During the first inner loop, an agent executing the protocol performs a series of basic walks, of different lengths. For  $j = 0$ , the agent performs nothing. In this case, **prime**( $i$ ) is performed on  $P$  directly. For  $j > 0$ , the agent performs a basic walk in  $T$  to the  $j$ th node of degree different from 2 that it encounters along its walk. When  $j = 2(\nu-1)$ , the basic walk is a complete one, traversing each edge of  $T$  twice. Each  $\text{BW}(j)$  is followed by a  $\text{CBW}(j)$ , so as to come back to the original position at the same extremity of the path  $P$ . Once this is done, the agent performs **prime**( $i$ ) on  $P$ .

The second inner loop aims at resetting the two agents. For this purpose, each agent goes to the other extremity of  $C$ , performs the same sequence of actions as the other agent had performed during its execution of the first inner loop, and returns to its original extremity of  $C$ . This enables resetting the two agents in the following sense.

**CLAIM 4.3.** *Let  $t$  and  $t'$  be the times of arrival of the two agents at the two extremities of  $C$  after the execution of **recognize-bis**. Then the difference between the times the two agents enter each execution of the outer loop of the protocol in Figure 2 is at most  $|t - t'|$ .*

To establish the claim, just notice that, during every execution of the outer loop, the sets of actions performed by

the two agents inside the loop are identical, differing only by their orders.

A consequence of Claim 4.3 is the following lemma.

**LEMMA 4.2.** *Let  $t$  and  $t'$  be the times of arrival of the two agents at the two extremities of  $C$  after the execution of **recognize-bis**. For every  $i$ , the delay between the two agents at the beginning of each execution of **prime**( $i$ ) cannot exceed  $|t - t'| + 18n\ell$ . On the other hand, for every  $i$ , if at the beginning of each execution of **prime**( $i$ ) the delay between the two agents is zero, then their initial positions were symmetric in  $T$ .*

**PROOF.** For  $j \geq 1$ , let  $l_j$  and  $l'_j$  be the lengths (i.e., numbers of edges) of the paths in  $T$  between the  $j$ th and the  $(j + 1)$ th node of degree different from 2 that is met by the two agents, respectively, during their basic walk from their positions at the two extremities of  $C$ . At the  $j$ th iteration of the first inner loop, one agent has traversed  $2 \sum_{a=1}^j \sum_{b=1}^a l_b$  edges during **BW**( $a$ ) and **CBW**( $a$ ) for all  $a = 1, \dots, j$ . The other agent has traversed  $2 \sum_{a=1}^j \sum_{b=1}^a l'_b$  edges during the same **BW**( $a$ ) and **CBW**( $a$ ). Since the number of rounds of **prime**( $i$ ) is the same for both agents, we get that their “desynchronization” is at most:

$$\begin{aligned} |t - t'| + 2 \sum_{a=1}^j \sum_{b=1}^a |l_b - l'_b| &\leq n + 4(\nu - 1) \sum_{b=1}^{2(\nu-1)} |l_b - l'_b| \\ &\leq n + 4(\nu - 1) \sum_{b=1}^{2(\nu-1)} \max\{l_b, l'_b\} \\ &\leq n + 8(\nu - 1)n \\ &\leq 9\nu n \\ &\leq 18n\ell. \end{aligned}$$

This completes the proof of the first statement in the lemma. Assume now that, at the beginning of each of the  $2\nu - 1$  executions of **prime**( $i$ ) in the outer loop, the delay between the two agents is zero. This implies that, for every  $j = 0, \dots, 2(\nu - 1)$  we have

$$t + 2 \sum_{a=1}^j \sum_{b=1}^a l_b = t' + 2 \sum_{a=1}^j \sum_{b=1}^a l'_b.$$

Therefore,  $t = t'$  and  $l_j = l'_j$  for every  $j = 1, \dots, 2(\nu - 1)$ . In other words, the two agents started at the same time, and all segments of degree-2 nodes encountered by the agents were of the same length. Since the contraction tree  $T'$  is symmetric, it follows that the initial positions of the two agents were symmetric in  $T$ .  $\square$

In view of the lemma, at each execution  $i$  of the outer loop, there is an execution  $j$  of **prime**( $i$ ) for which the two agents do not start the second phase at the same time from their respective extremities of  $P$ . Moreover, during this  $j$ th execution of **prime**( $i$ ), the delay between the two agents is at most  $|t - t'| + 18n\ell$ . Hence, from Fact 2.1, this delay  $\delta$  is at most  $19n\ell$ . The length of the rendezvous path  $P$  is at least  $20n\ell$  because  $B_u$  and  $B_v$  are each of length at least  $2n$ . Therefore, at the first time when both agents are simultaneously in the  $j$ th execution of **prime**( $i$ ), they occupy two non symmetric positions in  $P$ : one is at one extremity of  $P$ , and the other is at some node of  $P$  at distance  $\delta > 0$

along  $P$  from the other extremity of  $P$ . Moreover, since the delay  $\delta$  between the two agents is smaller than the length of the path  $P$ , the agent first executing **prime**( $i$ ) has not yet completed the first traversal of  $P$  when the other agent starts **prime**( $i$ ). As a consequence, the two agents act as if **prime**( $i$ ) were executed with both agents starting simultaneously at non symmetric positions in the path. Now, for small values of  $i$ , **prime**( $i$ ) may not achieve rendezvous in  $P$ . However, in view of Lemma 4.1, for some  $i = O(\log n)$ , rendezvous will be completed whenever the initial positions of the agents were not symmetric in  $T$ .

We complete the proof by checking that each agent uses  $O(\log \ell + \log \log n)$  bits of memory. Protocol **recognize-bis** executed in  $T$  consumes the same amount of memory as Protocol **recognize** executed in  $T'$ . Since  $T'$  has at most  $2\ell - 1$  nodes, **recognize-bis** uses  $O(\log \ell)$  bits of memory. During the second phase of the rendezvous, a counter is used for identifying the index  $j$  of the inner loop. Since  $j \leq 2\nu \leq 4\ell$ , this counter uses  $O(\log \ell)$  bits of memory. All executions of **prime** are independent, and performed one after the other. Thus, in view of Lemma 4.1, a total of  $O(\log \log n)$  bits suffice to implement these executions. The index  $i$  of the outer loop grows until it is large enough so that **prime**( $i$ ) achieves rendezvous in a path of length  $O(n\ell)$ . Thus,  $i \leq \log(n\ell)$ , and thus  $O(\log \log(n\ell)) = O(\log \log n)$  bits suffice to encode this index. This completes the proof of Theorem 4.1.

## 4.2 The lower bound $\Omega(\log \ell)$

In this section we prove that rendezvous with simultaneous start in trees with  $\ell$  leaves requires  $\Omega(\log \ell)$  bits of memory even in the class of trees with maximum degree 3. Together with the lower bound of  $\Omega(\log \log n)$  on memory size needed for rendezvous on the  $n$ -node line<sup>1</sup>, established in [19], this result proves that our upper bound  $O(\log \ell + \log \log n)$  from Section 4.1 cannot be improved even for trees of maximum degree 3.

**THEOREM 4.2.** *For infinitely many integers  $\ell$ , there exists an infinite family of trees with  $\ell$  leaves, for which rendezvous with simultaneous start requires  $\Omega(\log \ell)$  bits of memory.*

**PROOF.** Let  $\ell = 2^i$ . Consider a complete binary tree of depth  $i - 1$  (thus with  $2^{i-1}$  leaves). Label ports in this tree by assigning, at each node, label 0 to the port leading to the parent, and labels 1 and 2 to the ports leading to children. This labeling determines a fixed order of all nodes (e.g., using the lexicographic order of the sequences of ports leading from the root to nodes). For any binary vector of length  $2^{i-1}$  modify the tree by attaching to each leaf corresponding to 1 an additional leaf and adding nothing at leaves corresponding to 0. Assign numbers 1 and 0 to ports corresponding to newly created edges (1 at the old leaf which becomes a node of degree 2, and 0 at the new leaf). Clearly there are  $2^{2^{i-1}} = 2^{\ell/2}$  resulting labeled trees. Call them *side trees*.

For any pair of side trees  $T'$  and  $T''$  and for any positive even integer  $m$ , consider the tree  $T$  consisting of side trees  $T'$  and  $T''$  whose roots are joined by a path of length  $m + 1$  (i.e., there are  $m$  added nodes of degree two). Ports at the added nodes of degree two are labeled as follows: both ports

<sup>1</sup>Notice that the lower bound  $\Omega(\log \log n)$  holds for  $n$ -node trees with many leaves as well: it suffices to attach  $\ell/2$  leaves on each extremity of the line and the argument from [19] goes through.

at the central edge have label 0, and ports at both ends of any other edge of the line have the same label 0 or 1. (This corresponds to a 2-edge-coloring of the line). Call any tree resulting from this construction a *two-sided tree*. For any two-sided tree consider initial positions of the agents at nodes  $u$  and  $v$  of the line adjacent to roots of its side trees.

Consider agents with  $k$  bits of memory (thus with  $K = 2^k$  states). A *tour* of a side tree associated with an initial position ( $u$  or  $v$ ) is the part of the trajectory of the agent in this side tree between consecutive visits of the associated initial position. Observe that the maximum duration  $D$  of a tour is smaller than  $K \cdot (3 \cdot 2^{i-1})$ . Indeed, the number of nodes in a side tree is at most  $3 \cdot 2^{i-1} - 1$ , hence the number of possible pairs (state, node of the side tree) is at most  $K \cdot (3 \cdot 2^{i-1} - 1)$ . A tour of longer duration than this value would cause the agent to leave the same node twice in the same state, implying an infinite loop. Such a tour could not come back to the initial position.

For a fixed agent with the set  $S$  of states and a fixed side tree, we define the function  $p : S \rightarrow S$  as follows. Let  $s$  be the state in which the agent starts a tour. Then  $p(s)$  is the state in which the agent finishes the tour. Now we define the function  $q : S \rightarrow S \times \{1, \dots, D\}$ , called the *behavior function*, by the formula  $q(s) = (p(s), t)$ , where  $t$  is the number of rounds to complete the tour when starting in state  $s$ . The number of possible behavior functions is at most  $F = (KD)^K$ . A behavior function depends on the side tree for which it is constructed.

Suppose that  $k \leq \frac{1}{3} \log \ell$ . We have  $D < 3K2^{i-1} = \frac{3}{2}K\ell$ , hence  $KD < \frac{3}{2}K^2\ell$ . Hence we have  $\log K + \log \log(KD) \leq k + \log \log(\frac{3}{2}K^2\ell) \leq k + 2 + \log k + \log \log \ell$ , which is smaller than  $\frac{2}{3} \log \ell$  for sufficiently large  $k$ . It follows that  $K \log(KD) < \ell^{2/3} < \ell/2$ , which implies  $F = (KD)^K < 2^{\ell/2}$ . Thus the number of possible behavior functions is strictly smaller than the total number of side trees. It follows that there are two side trees  $T_1$  and  $T_2$  for which the corresponding behavior functions are equal.

Consider two instances of the rendezvous problem for any length  $m + 1$  of the joining line, where  $m$  is a positive even integer: one in which both side trees are equal to  $T_1$ , and the other for which one side tree is  $T_1$  and the other is  $T_2$ . Rendezvous is impossible in the first instance because in this instance initial positions of the agents form a symmetric pair of nodes. Consider the second instance, in which the initial positions of the agents do not form a symmetric pair. Because of the symmetry of labeling of the joining line, agents cannot meet inside any of the side trees. Indeed, when one of them is in one tree, the other one is in the other tree. Since the behavior function associated with side trees  $T_1$  and  $T_2$  is the same, the agents leave these trees always at the same time and in the same state. Hence they cannot meet on the line, in view of its odd length. This implies that they never meet, in spite of asymmetric initial positions. Hence rendezvous in the second instance requires  $\Omega(\log \ell)$  bits of memory.  $\square$

## 5. REFERENCES

- [1] S. Alpern, The rendezvous search problem, *SIAM J. on Control and Optimization* 33 (1995), 673-683.
- [2] S. Alpern, Rendezvous search on labelled networks, *Naval Reaserch Logistics* 49 (2002), 256-274.
- [3] S. Alpern and S. Gal, The theory of search games and rendezvous. *Int. Series in Operations research and Management Science*, Kluwer Academic Publisher, 2002.
- [4] J. Alpern, V. Baston, and S. Essegaiar, Rendezvous search on a graph, *Journal of Applied Probability* 36 (1999), 223-231.
- [5] E. Anderson and R. Weber, The rendezvous problem on discrete locations, *Journal of Applied Probability* 28 (1990), 839-851.
- [6] E. Anderson and S. Fekete, Asymmetric rendezvous on the plane, *Proc. 14th Annual ACM Symp. on Computational Geometry* (1998), 365-373.
- [7] E. Anderson and S. Fekete, Two-dimensional rendezvous search, *Operations Research* 49 (2001), 107-118.
- [8] 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 Opt.* 36 (1998), 1880-1889.
- [9] V. Baston and S. Gal, Rendezvous search when marks are left at the starting points, *Naval Reaserch Logistics* 48 (2001), 722-731.
- [10] S. A. Cook and C. Rackoff. Space Lower Bounds for Maze Threadability on Restricted Machines. *SIAM J. Comput.* 9 (1980), 636-652.
- [11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms*, McGraw-Hill 1990.
- [12] G. De Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, U. Vaccaro, Asynchronous deterministic rendezvous in graphs, *Theoretical Computer Science* 355 (2006), 315-326.
- [13] A. Dessmark, P. Fraigniaud, D. Kowalski, A. Pelc. Deterministic rendezvous in graphs. *Algorithmica* 46 (2006), 69-96.
- [14] K. Diks, P. Fraigniaud, E. Kranakis, A. Pelc, Tree exploration with little memory, *Journal of Algorithms* 51 (2004), 38-63.
- [15] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Gathering of asynchronous oblivious robots with limited visibility, *Proc. 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2001)*, LNCS 2010, 247-258.
- [16] P. Fraigniaud and C. Gavoille. A Space Lower Bound for Routing in Trees. In *19th Annual Symp. on Theoretical Aspects of Computer Science (STACS 2002)*, Springer LNCS 2285, 65-75.
- [17] P. Fraigniaud and C. Gavoille. Routing in Trees. In *28th Int. Colloquium on Automata, Languages and Programming (ICALP 2001)*, Springer LNCS 2076, 757-772.
- [18] P. Fraigniaud and D. Ilcinkas. Digraphs Exploration with Little Memory. *21st Symp. on Theoretical Aspects of Comp. Science (STACS 2004)*, Springer LNCS 2996, 246-257.
- [19] P. Fraigniaud, A. Pelc, Deterministic rendezvous in trees with little memory, *Proc. 22nd International Symposium on Distributed Computing (DISC 2008)*, Springer LNCS 5218, 242-256.
- [20] S. Gal, Rendezvous search on the line, *Operations Research* 47 (1999), 974-976.
- [21] L. Gasieniec, A. Pelc, T. Radzik, X. Zhang, Tree

- exploration with logarithmic memory, Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), 585-594.
- [22] A. Israeli and M. Jalfon, Token management schemes and random walks yield self stabilizing mutual exclusion, Proc. 9th Annual ACM Symposium on Principles of Distributed Computing (PODC 1990), 119-131.
- [23] M. Koucký, Universal Traversal Sequences with Backtracking, Proc. 16th IEEE Conference on Computational Complexity (2001), 21-26.
- [24] D. Kowalski, A. Malinowski, How to meet in anonymous network, in 13th Int. Colloquium on Structural Information and Comm. Complexity, (SIROCCO 2006), Springer LNCS 4056, 44-58.
- [25] E. Kranakis, D. Krizanc, and P. Morin, Randomized Rendez-Vous with Limited Memory, Proc. 8th Latin American Theoretical Informatics (LATIN 2008), Springer LNCS 4957, 605-616.
- [26] E. Kranakis, D. Krizanc, N. Santoro and C. Sawchuk, Mobile agent rendezvous in a ring, Proc. 23rd Int. Conference on Distributed Computing Systems (ICDCS 2003), IEEE, 592-599.
- [27] W. Lim and S. Alpern, Minimax rendezvous on the line, SIAM J. on Control and Optimization 34 (1996), 1650-1665.
- [28] O. Reingold. Undirected connectivity in log-space. Journal of the ACM 55(2008), 1-24.
- [29] T. Schelling, The strategy of conflict, Oxford University Press, Oxford, 1960.
- [30] A. Ta-Shma and U. Zwick. Deterministic rendezvous, treasure hunts and strongly universal exploration sequences. Proc. 18th ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), 599-608.
- [31] L. Thomas, Finding your kids when they are lost, Journal on Operational Res. Soc. 43 (1992), 637-639.
- [32] 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.