Deterministic rendezvous, treasure hunts and strongly universal exploration sequences

Amnon Ta-Shma * *Uri Zwick* *

Abstract

We obtain several improved solutions for the *deterministic rendezvous* problem in general undirected graphs. Our solutions answer several problems left open in a recent paper by Dessmark *et al.* We also introduce an interesting variant of the rendezvous problem which we call the *deterministic treasure hunt* problem. Both the rendezvous and the treasure hunt problems motivate the study of *universal traversal sequences* and *universal exploration sequences* with some strengthened properties. We call such sequences *strongly universal traversal (exploration) sequences*. We give an explicit construction of strongly universal *exploration* sequences. The existence of strongly universal *traversal* sequences, as well as the solution of the most difficult variant of the deterministic treasure hunt problem, are left as intriguing open problems.

1 Introduction

In the *rendezvous problem* (see Dessmark et al. [DFKP06]) two robots are placed in an unknown environment modeled by a finite, connected, undirected graph. Our goal is to give the two robots *deterministic* sequences of instructions that will ensure that they would eventually meet, no matter on which graph they are placed, and no matter when they are activated. It is, of course, desired that such a meeting would take place as soon as possible. An exact definition of the problem is presented below.

We let n be size of the graph on which the two robots are placed. This size is *not* known to the two robots. As we are after *deterministic* solutions, we have to assume that robot i, where $i = 1, 2$, has a label L_i assigned to it, and $L_1 \neq L_2$. In the absence of such unique labels there is no deterministic way of breaking symmetry and no deterministic solution is possible. We assume that the moves of the robots, after they are activated, are synchronous. An important feature of the problem is that the two robots may be activated at different times, chosen arbitrarily by the adversary. A meeting can take place only when both robots are active.

Dessmark et al. [DFKP06] (see also [DFP03] and [KP04]) presented a deterministic solution of the rendezvous problem which guarantees a meeting of the two robots after a number of steps which is polynomial in n, the size of the graph, ℓ, the length of the shorter of the two labels, and τ, the difference between their activation times. (As the robots can only meet when they are both active, only steps taken after the activation of the second robot are counted.) More specifically, the bound on the number of steps that they obtain is $\tilde{O}(n^5\sqrt{\tau\ell}+n^{10}\ell)$. Note that τ may be much larger then n^{10}, in which case the first term in the bound dominates the second. Dessmark et al. [DFKP06] ask whether it is possible to obtain a polynomial bound that is independent of τ. We answer their question in the affirmative by presenting a deterministic solution that guarantees a rendezvous within $\tilde{O}(n^5\ell)$ time units after the activation of the second robot. In addition to being independent of τ, our solution is much more efficient than the solution of Dessmark et al. [DFKP06], even if the two robots are activated at essentially the same time and τ is small. When the unknown graph in which the two robots are places has maximum degree d, our solution guarantees a rendezvous after at most $\tilde{O}(d^2n^3\ell)$ steps. If the graph is a simple d-regular graph, the number of steps is further reduced to $\tilde{O}(dn^3\ell)$.

Kowalski and Malinowski [KM06] have recently presented a different deterministic solution to the rendezvous problem that guarantees a meeting after at most $\tilde{O}(n^{15} + \ell^3)$ steps. We were not aware of their results when we wrote the first version of our paper. The number of steps performed by their solution is again independent of τ, the difference between the activation time of the two robots, and they are therefore the first to answer the open problem of [DFKP06]. Their solution, however, is less efficient than our solution, no matter what the relation between n and ℓ is, and, more importantly, it only works in a model in which *backtracking* is allowed. Our fastest solution does *not* use backtracking.

*School of Computer Science, Tel Aviv University, Tel Aviv 69978,Israel. E–mail: {amnon,zwick}@tau.ac.il. Amnon thanks the Israel Science Foundation, the Binational Science Foundation and the EU Integrated Project QAP for their support. Uri's research was supported by a GIF grant.

Our fastest solution and the solutions of Dessmark et al. [DFKP06] and [KM06] rely on the existence of *universal traversal sequences*, introduced by Aleliunas et al. [**?**]. The existence of universal traversal sequences is proved using a probabilistic argument. Thus, even though the deterministic solutions mentioned above require only a polynomial number of *steps*, it is not currently known how to compute these steps deterministically in polynomial *time*. A natural question, raised again by Dessmark et al. [DFKP06], is whether there is a solution in which the number of steps and the computation required to determine them are both polynomial. We answer this question in the affirmative, when backtracking is allowed, by using explicit constructions of universal *exploration* sequences.

One of the motivations for the introduction of universal traversal sequences by Aleliunas et al. [**?**] was an attempt to obtain a deterministic log-space algorithm for the *s-t* connectivity problem in undirected graphs. Such an algorithm was recently obtained by Reingold [**?**], thus resolving this major open problem. Reingold [**?**] log-space algorithm for the undirected *s-t* connectivity problem does not provide universal traversal sequences for general graphs. It does provide, however, a log-space, and hence polynomial time, construction of *universal exploration sequences*, a closely related notion previously introduced by Koucký [**?**].

Using the explicit, polynomial-time constructible, universal exploration sequences of Reingold [**?**], we obtain a polynomial-time constructible deterministic algorithm for a natural version of the rendezvous problem in which backtracking is allowed.

In the *treasure hunt* problem a single *robot* is supposed to locate a *treasure* placed in an unknown location in an unknown environment, modeled again by a finite, connected, undirected graph. The problem can be easily solved, using universal traversal sequences, if the treasure is present in the graph when the robot is activated. A much more difficult case is when the robot starts roaming the graph *before* the treasure is placed in the graph, and yet we would like the number of steps that the robot makes from the time the treasure is placed until the treasure is found to be polynomial in n, the number of vertices in the graph, no matter how long the robot has already been running.

The treasure hunt problem corresponds to a version of the rendezvous problem in which one of the robots is completely passive. It may seem, at first sight, that the treasure hunt problem is easier than the rendezvous problem. This, however, is *not* the case.

Loosely speaking, a universal traversal sequence for n-vertex graphs is a predetermined sequence of instructions that when executed on *any* n-vertex graph, from *any* starting vertex, define a walk that visits all the vertices of the graph. A neat probabilistic argument was used by Aleliunas et al. [**?**] to show that there exist universal traversal sequences for n-vertex graphs of length $\tilde{O}(n^5)$. In fact, almost all sequences of this length are universal traversal sequences!

The study of the treasure hunt problem naturally leads to the following problem: Is there a fixed polynomial $p(\cdot)$, such that for every $n \geq 1$ there is a sequence S_n of instructions of length $p(n)$, such that for every $1 \leq k \leq n$, every contiguous subsequence of S_n of length $p(k)$ is a universal traversal sequence for k-vertex graphs? We call such sequences *strongly universal traversal sequences* and we call $p(\cdot)$ their *cover time*. A similar notion was defined in [KM06]. Such sequences, if they exist, would provide very simple solutions to both the rendezvous and the treasure hunt problems. Unfortunately, the existence of such sequences, when no backtracking is allowed, remains an intriguing open problem. We can, however, show the existence of sequences that satisfy the above requirements when the condition $1 \leq k \leq n$ is relaxed to $c\sqrt{\log n} \leq k \leq n$, for some constant c. Such sequences are used in our most efficient rendezvous algorithms.

We can also show the existence of strong universal *exploration* sequences with small polynomial cover times. We remark that the standard probabilistic arguments used to show the existence of (non-strong) universal traversal and exploration sequences cannot be used to prove the existence of *strongly* universal sequences, with *any* cover time. Our construction uses (non-strong) universal exploration sequences as black-boxes, and is efficient otherwise. In particular, when using Reingold's explicit constructions of polynomially long universal exploration sequences, we get an *explicit* construction of strongly universal exploration sequences, and hence *explicit* solutions of the treasure hunt and the rendezvous problems, when backtracking is allowed.

2 The rendezvous problem

We now give the formal definition of the *rendezvous* problem. We in fact define two versions of the problem, with and without *backtracking*. The definitions given are essentially identical to the ones used by Dessmark et al. [DFKP06] and Kowalski and Malinowski [KM06]. The rendezvous solution of [DFKP06] does not require backtracking while the one given by [KM06] uses backtracking in an essential way. We present solutions for both versions of the problem.

Two *robots* are placed in an unknown environment modeled by a finite, connected, undirected graph $G = (V, E)$. We assume that $|V| = n$. The size of the

environment, i.e., the number of vertices in the graph is *not* known to the robots. (The knowledge of n makes the problem easy to solve.) The edges incident on a vertex $u \in V$ are numbered $1, 2, \ldots, \deg(u)$, in a predetermined manner, where $\deg(u)$ is the degree of u. In general, the numbering is not assumed to be consistent, i.e., an edge $(u, v) \in E$ may be the i-th edge of u but the j-th edge of v, where $i \neq j$.

When a robot is in a vertex $u \in V$ it is told the degree $\deg(u)$ of u. All vertices of the same degree are, however, indistinguishable. A robot is not allowed to leave tokens or markers at the vertices that it visits. At each time unit a robot is allowed to either traverse an edge, or stay in place. When the robot is at a vertex u it may ask to traverse the i-th edge $(u, v) \in E$ of u, where $1 \leq i \leq \deg(u)$. The robot then finds itself at v, the other endpoint of this edge. As explained, the i-th edge of u is the j-th edge of v, for some $1 \leq j \leq \deg(v)$. In general $j \neq i$. In the first (easier) variant of the model, the robot is told the index j of the edge it traversed to enter v. This allows the robot to return to u at the next step, if it so chooses. We call this variant of the problem the *rendezvous problem with backtracking*. In the second (more difficult) variant of the model, the robot finds itself at v without knowing which edge it used to get there. We call this variant of the problem the *general rendezvous problem*, or simply the *rendezvous problem*.

Our goal is to have the two robots meet. A robot is, of course, unaware of the whereabouts of the other robot, even if it is at a relatively short distance from it in the graph. The two robots notice each other only when they are both active and are at the same vertex at the same time. In particular, the two robots may traverse the same edge, in opposite directions, and still miss each other.

The rendezvous problem has a trivial solution if *randomization* is allowed. Each robot simply performs a random walk on the graph. The two robots will then meet, with high probability, after a polynomial number of steps in the size of the graph (see, e.g., Coppersmith et al. [**?**]).

We assume in this paper that the two robots move synchronously. For an asynchronous version of the model, see De Marco et al. [DGK$^+$06]. The two robots are not necessarily activated, however, at the same time, and they notice each other only when they are both active. (The simultaneous activation version of the problem is again much easier.) Each robot may keep a count of the number of time units that have elapsed since it was activated. Even though the robots move synchronously, they have no access to a common clock that gives an *absolute* time.

We let τ denote the number of time units that separate the activation times of the two robots. Obtaining a solution of the rendezvous problem that guarantees a rendezvous in a number of steps that is independent of τ is one of the problems left open by Dessmark et al. [DFKP06]. As mentioned, the problem was recently solved by Kowaliski and Malinowski [KM06] when backtracking is allowed. We provide a solution for the harder version of the problem in which backtracking is *not* allowed.

If the two robots are completely identical, the rendezvous problem cannot be solved deterministically. Suppose, for example, that G is a ring on n vertices and that the edges are labeled so that out of every vertex, edge 1 goes clockwise, while edge 2 goes anti-clockwise. If the two robots start the same time at different vertices and follow the same instructions, they would never meet! We need, therefore, a way of breaking the symmetry. This is done, as already mentioned, by assigning robot i a label L_i, where $L_1 \neq L_2$. Each robot knows its own label, but not the label of the robot it is supposed to meet. (It just knows that it has a different label.) If the robots know each other's labels, the robot with the smaller label can replace its label by 0, and the other robot can replace its label by 1, thus obtaining a bound on the number of steps which is independent of ℓ. Furthermore, in such a case, the problem can be reduced to the treasure hunt problem, as the robot with the smaller label can play the role of the treasure. However, the treasure hunt problem is not as easy as expected. While it is easy to obtain a polynomial bound for the problem that depends on both n and τ, finding a bound that is independent of τ seems to be much harder. We give such a bound for the version of the problem with backtracking, and leave the general case, in which backtracking in not allowed, as a major open problem.

Formally, a deterministic solution for the general rendezvous problem is a deterministic algorithm that computes a function $f : \mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathbb{Z}^+ \to \mathbb{Z}^+$, where for every $L \geq 0$, $d \geq 1$ and $t \geq 0$ we have $0 \leq f(L, d, t) \leq d$. This function defines the walk carried out by a robot with label L as follows: at the t-th time unit since activation, when at a vertex of degree d, stay in place if $f(L, d, t) = 0$, or use edge number $f(L, d, t)$, otherwise.

A deterministic solution for the rendezvous problem with backtracking is a deterministic algorithm that computes a function $f : \mathbb{Z}^+ \times \mathbb{Z}^+ \times (\mathbb{Z}^+)^* \to \mathbb{Z}^+$, where for every $L \geq 0$, $d \geq 1$ and $T \in (\mathbb{Z}^+)^*$ we have $0 \leq f(L, d, T) \leq d$. This function defines the walk carried out by a robot with label L as follows: at the t-th time unit since activation, when at a vertex of degree d, if the sequence of edge numbers assigned to the edges that were used to enter the vertices at the

previous time units is $T \in (\mathbb{Z}^+)^t$ (place a 0 in T in all time units in which the robot did not move), stay in place if $f(L, d, T) = 0$, or use edge number $f(L, d, T)$, otherwise. In our solutions that use this model, the function f depends on T only through its last element.

Throughout most of this paper we shall assume that the graph G in which the robots are placed is a d-regular graph, for some $d \geq 3$. This simplifies the presentation without sacrificing any important details. It is easy to extend the solutions given for the d-regular graphs to general graphs using ideas that appear in [DFKP06] (and [KM06]). More details would appear in the full version of the paper.

3 The treasure hunt problem

The *treasure hunt* problem is the variant of the *rendezvous* problem in which the robots are assigned the labels 0 and 1 and robot 0, the *treasure*, cannot move. As in the rendezvous problem, the treasure and the robot looking for it are not necessarily activated at the same time. The hard case of the problem is when the robot is activated before the treasure. Obtaining a deterministic polynomial bound for this problem which is independent of τ, when no backtracking is allowed, is an intriguing open problem. We present an explicit deterministic polynomial solution that does use backtracking.

4 Universal and strongly universal traversal sequences

Let $G = (V, E)$ be a d-regular graph. All graphs in this paper are assumed to be connected. A sequence $\sigma_1 \ldots \sigma_k \in \{1, \ldots, d\}^k$ defines a walk $v_0, v_1, \ldots, v_k$ in G, that starts at $v_0 \in V$ as follows: For $1 \leq i \leq k$, let v_i be the neighbor of v_{i-1} reached using the σ_i-th edge of v_{i-1}. A walk $v_0, v_1, \ldots, v_k$ is said to *cover* a graph G if it visits every vertex of G at least once.

Definition 4.1. (Universal Traversal Sequences (UTSs)) *A sequence $\sigma \in \{1, \ldots, d\}^\ell$ is a* universal traversal sequence *(UTS) for connected d-regular graphs of size at most n if for every such graph G, any numbering of its edges, and any starting vertex v_0 in G, the walk defined by σ in G covers G.*

Aleliunas et al. [**?**] showed that a random sequence of length $O(d^2 n^3 \log n)$ is, with high probability, a universal traversal sequence for d-regular graphs of size at most n. The d-regular graphs here are *not* assumed to be simple.

We introduce a two-fold generalization of universal traversal sequences. First, we would like to have a single *infinite* sequence that can be used to cover all graphs of all sizes. Second, and more importantly, we would like any *suffix* of this infinite sequence to have the same properties.

Definition 4.2. (Strongly Universal Traversal Sequences (SUTSs)) *A possibly infinite sequence $\sigma = \sigma_1 \sigma_2 \ldots$, where $\sigma_i \in \{1, \ldots, d\}$, is a* strongly *universal traversal sequence (SUTS) for connected d-regular graphs with cover time $p(\cdot)$, if for any $n \geq 1$, any contiguous subsequence of σ of length $p(n)$ is a UTS for connected d-regular graphs of size n.*

Note that if there is a fixed polynomial $p(\cdot)$ such that for every $n \geq 1$, there is a SUTS U_n of length n with cover time $p(\cdot)$, then $U_1 U_2 U_4 U_8 \ldots$ is an infinite SUTS with cover time $cp(\cdot)$, for some $c > 0$.

We do not know whether SUTS exist, and this is the main open problem raised by our work. We can however show, using the standard probabilistic argument, the existence of slightly weaker SUTS which are sufficient for solving the rendezvous problem, but not the treasure hunt problem.

Definition 4.3. *A sequence $\sigma \in \{1, \ldots, d\}^\ell$ is a* strongly universal traversal sequence *(SUTS) for connected d-regular graphs of size $[n', n'']$ with cover time $p(\cdot)$, if σ is of length at least $p(n'')$ and for any integer n, $n' \leq n \leq n''$, any contiguous subsequence of σ of length $p(n)$ is a UTS for connected d-regular graphs of size n. For brevity, we say that such a sequence is an $[n', n'']$-SUTS.*

We now claim:

Lemma 4.1. *For every n, there is an $[\log n, n]$-SUTS for d-regular graphs with cover time $p(k) = O(d^2 k^3 \log k)$.*

Proof: Aleliunas et al. [**?**] showed, by bounding the *cover* time of random walks on graphs, that the probability that a random sequence of length $O(d^2 n^3 \log n)$ over the alphabet $\{1, 2, \ldots, d\}$ is *not* a UTS for d-regular graphs of size n, is at most 2^{-n^2}. By the union bound, the probability that such a random sequence is not a $[\log n, n]$-SUTS, is at most $O(n \cdot d^2 n^3 \log n) \cdot 2^{-\log^2 n} < 1$. (Note that there are less than n graph sizes, only $O(d^2 n^3 \log n)$ starting points, and the probability that a specific subsequence is not a UTS for graphs of size s is at most 2^{-s^2}.) Thus, the probability that such a random sequence is a $[\log n, n]$-SUTS is positive, which shows that sequences with the required properties do exist. □

The proof given actually proves the existence of $[c\sqrt{\log n}, n]$-SUTS with cover time $p(k) = O(d^2 k^3 \log k)$,

for some $c > 0$. We do not need this slightly stronger result.

Kowalski and Malinowski [KM06] define *universal cover walks* which are essentially equivalent to our SUTSs, and *almost universal cover walks* which are essentially equivalent to our $[n', n'']$-SUTSs. They prove a lemma that is essntially equivalent to Lemma 4.1.

It is worthwhile noting that the standard probabilistic algorithm *cannot* be used to obtain SUTSs that satisfy the conditions of Definition 4.2. In particular, note that a random sequence of length $\Omega(n)$ would contain, with very high probability, $\Omega(\log_d n)$ consecutive 1's, and it is easy to see that such a subsequence is *not* a UTS even for d-regular graphs of *constant* size.

It is tempting to try and prove the existence of SUTSs using the Lovász *local lemma* (see Erdös and Lovász [**?**] or Alon and Spencer [**?**]). Unfortunately, all our attempts to prove the existence of SUTSs using the local lemma failed.

5 Deterministic rendezvous solutions

5.1 An informal discussion A natural idea is to have each robot run a sequence of UTSs for graphs of increasing sizes, e.g., the infinite sequence $U_1U_2U_4 \ldots U_{2^k} \ldots$, where U_n is a UTS for graphs of size n.

The sequence U_n is guaranteed to cover every graph G of size n, from any starting point. A robot running U_n is thus guaranteed of meeting the other robot, if the other robot is *stationary* (at least for long enough). The sequence $U_1U_2U_4 \ldots U_{2^k} \ldots$ provides, therefore, a solution for the (easy) version of the treasure hunt problem in which the treasure is placed in the graph *before* the robot is activated. Note that if the treasure is placed in a graph of size n *after* the robot is activated, then the robot may have already started implementing UTS for graphs that are much larger than n, and there is no bound that depends only on n on the number of steps that the robot would make before it finds the treasure.

If both robots implement the sequence $U_1U_2U_4 \ldots U_{2^k} \ldots$, then we have no guarantee that the two robots would meet. Part of the problem, as explained in the introduction, is the need to break the symmetry. Assume, therefore, that one of the two robots is assigned the label 0 and the other the label 1. The case of longer labels will be dealt with later.

A natural idea now is to have robot 0 follow the infinite sequence

$$0^{|U_1|}U_1\ 0^{|U_2|}U_2\ 0^{|U_4|}U_4\ \ldots\ 0^{|U_{2^k}|}U_{2^k}\ \ldots$$

and robot 1 follow the 'opposite' sequence

$$U_1 0^{|U_1|}\ U_2 0^{|U_2|}\ U_4 0^{|U_4|}\ \ldots\ U_{2^k} 0^{|U_{2^k}|}\ \ldots\ .$$

This would indeed provide a solution to the rendezvous problem, provided that the two robots are activated at the *same* time. It is not difficult to modify this solution so that it would work when the difference between the activation times of the two robots is polynomial in the size of the graph. (Basically, we repeat each block U_{2^i} twice, see next section.) We say, in this case, that the two robots are roughly *aligned*.

We are thus left with the case in which the difference between the activation time of the two robots is large, relative to the size of the graph in which the two robots are placed. To solve this case, we introduce idle periods not only before or after the completion of a full UTS U_{2^k}, but also between every two adjacent steps in such a sequence.

More specifically, a robot implementing a UTS U_i 'rests' for f_i time units after each step that it makes. A natural choice would be to have $f_i = |U_i|$. Let us say that a robot is in *block* i if it is walking according to U_i. (We assume here that i is a power of 2.) The total of time units required for implementing block i is therefore $|U_i|f_i$.

The crucial parameter used in the analysis of our rendezvous solutions is the index k of the block in which the robot who started first is at the time the second robot starts implementing its block n, where n is the size of the graph. (For simplicity, we assumed here that n is a power of 2). If $f_k > |U_n|f_n$ then the second robot has enough time to implement the sequence U_n, including all the idle steps added to it, while the first robot 'sleeps' and the two robots would meet.

If $f_k \leq |U_n|f_n$ then we have an upper bound on the difference between the starting times of the two robots and a meeting would take place as the two robots are roughly aligned.

5.2 A basic solution of the rendezvous problem We now describe in more detail our basic solution to the rendezvous problem. We again assume that the two robots are assigned the labels 0 and 1. The extension of the solution to the case of general labels is fairly straightforward.

We begin by introducing some notation. For any sequence σ and a bit $b \in \{0, 1\}$, let

$$\sigma^b \ = \ \begin{cases} \sigma & \text{if } b = 1, \\ 0^{|\sigma|} & \text{if } b = 0. \end{cases}$$

In other words, σ^1 is just σ, while σ^0 is a sequence of 0's whose length is equal to the length of σ. Also let

$$\sigma^{m_1 \ldots m_k} \ = \ \sigma^{m_1}\sigma^{m_2} \ldots \sigma^{m_k}\ .$$

Let

$$D_k(\sigma_1 \ldots \sigma_m) \ = \ \sigma_1 0^k \sigma_2 0^k \ldots 0^k \sigma_m 0^k\ .$$

In other words, the sequence $D_k(\sigma)$ is obtained from the sequence σ by inserting an idle period of length k after each step of σ. Finally, for $\ell \in \{0,1\}$ we let $\bar{\ell} = 1 - \ell$.

Let U_n be a UTS for graphs of size at most n, and of length $u_n = |U_n| = \Theta(n^c)$, for some $c \geq 1$. The sequence that a robot with label $\ell \in \{0,1\}$ runs is:

$$D_{u_1}((U_1U_1)^{\ell\bar{\ell}})\, D_{u_2}((U_2U_2)^{\ell\bar{\ell}}) \cdots D_{u_{2^k}}((U_{2^k}U_{2^k})^{\ell\bar{\ell}}) \cdots$$

We let $W_i = D_{u_i}(U_i)$, $B_i^{(0)} = 0^{|W_i|}0^{|W_i|}W_iW_i$ and $B_i^{(1)} = W_iW_i0^{|W_i|}0^{|W_i|}$. We also let $w_i = |W_i|$ and $b_i = |B_i^{(0)}| = |B_i^{(1)}|$. The sequence used by robot 0 is therefore $B_1^0B_2^0B_4^0\ldots$ and sequence used by robot 1 is therefore $B_1^1B_2^1B_4^1\ldots$. We call $B_i^{(\ell)}$ the i-th block of robot ℓ. We call the subsequences W_iW_i and $0^{|W_i|}0^{|W_i|}$ *chunks*. Each block is therefore composed of two chunks.

We let $w_i = |W_i|$ and $b_i = |B_i^{(0)}| = |B_i^{(1)}|$. Note that $w_i = u_i^2 = \Theta(n^{2c})$ and $b_i = 4w_i = 4u_i^2 = \Theta(n^{2c})$. We make the technical assumption that $4u_n \leq u_{2n}$, for every $n = 2^i$.[1] It follows that $16w_n \leq w_{2n}$ and $16b_n \leq b_{2n}$, for every $n = 2^i$. In particular, we get that for every $j \geq 1$, $\sum_{i=0}^{j-1} b_{2^i} < \frac{1}{15}b_{2^j}$. As a consequence, we get the following useful property: If one of the robots is activated when the other robot is at block B_k, then by the time the second robot starts executing block B_{2k}, the block that follows B_k, the first robot has finished much less than a fourth of this block. We are now ready to prove:

THEOREM 5.1. *If for every $k \geq 1$, U_k is a UTS for graphs of size k, and $|U_k| = \Theta(k^c)$, then the two robots meet after at most $\Theta(n^{4c})$ steps after the activation of the second robot, where n is the size of the graph in which the two robots are placed.*

Proof: Let n be the size of the graph on which the two robots are placed. We assume that n is a power of 2. (Otherwise, take the smallest power of 2 larger than n.) Let k be the index of the block in which the first robot to be activated is in at the time the second robot to be activated reaches block n. Clearly $k \geq n$. We consider two cases:

Case 1: $u_k \geq b_n$. As the first robot 'sleeps' for u_k units of time after each step that it takes, the second robot implements a full copy of W_n while the first robot is stationary and the two robots are guaranteed to meet. This case is depicted in Figure 5.2. (It is assumed in the figure that the second robot has label 0. The other case is similar.)

[1] This essentially corresponds to the assumption that $u_n = |U_n| = \Omega(n^2)$. A slightly modified version of the algorithm works without this assumption.

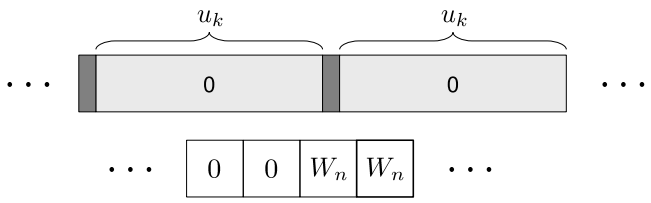

Figure 1: Case 1 in the proof of Theorem 5.1 ($u_k \geq b_n$).

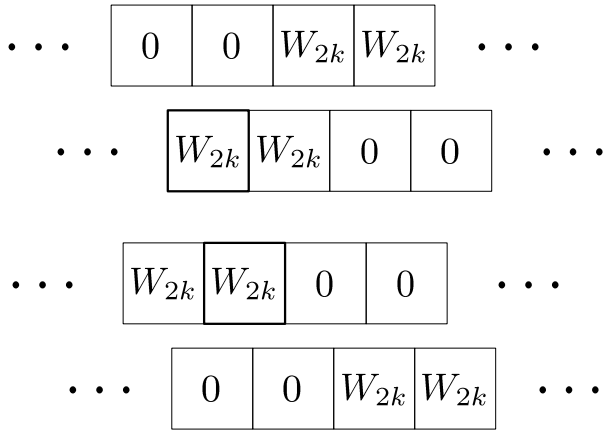

Figure 2: Case 2 in the proof of Theorem 5.1.

Case 2: $u_k < b_n$. As $u_k = \Theta(k^c)$ and $b_n = \Theta(n^{2c})$, it follows that $k \leq O(n^2)$. The second robot to be activated thus finishes executing its k-th block, and starts executing his $2k$-th block, after $O(k^{2c}) = O(n^{4c})$ time units. When this happens, the first robot, as discussed above, is still executing the first forth of its $2k$-th block. The situation is therefore as depicted in Figure 5.2. (In the upper figure the first robot has label 0 while in the lower figure the first robot has label 1.) The two robots are therefore guaranteed to meet before the $2k$-th block is over. □

5.3 An Improved solution To improve the time bound we need to make several changes. First, we replace each UTS U_n with a $[\log n, n]$-SUTS S_n for d-regular graphs with cover time $p(k) = \tilde{O}(k^c)$ and length $|S_n| = p(n)$ (see Lemma 4.1). To be specific, assume that $|S_n| = p(n) = O(n^c \log^b n)$ for some constants b and c. Second, we reduce the delay time performed after each step from $f_k = |S_k|$ to $f_k = (\log k)^c(\log\log k)^{2c+b}$ and we change the delay scheme. Instead of waiting f_k time units after each step in block k, the robot waits j time units after every j steps of the block, for j being a power of two up to f_k. More specifically, after the i'th step in block k the robot waits for r_i time units,

where r_i is the largest power of two for which $r_i|i$ and $r_i \leq f_k$. We still assume the labels are 0 and 1.

Before we proceed, we note that if we look at a robot that is in block k, then for every $m \leq f_k$, every time interval of length $O(m \log m)$ contains in it a continuous waiting period of length at least m. To see that notice that every m steps contain a point where the robot waits at least m time (because $m \leq f_k$). Furthermore, either executing these steps contains a waiting period larger than m, or else executing the m steps takes $O(m \log m)$ time, for every i a power of two, $m/2^i$ times we wait 2^i time.

We are now ready to prove:

THEOREM 5.2. *The two robots meet after $\tilde{O}(n^c)$ steps, where n is the size of the graph.*

Proof: As before, we let k be the block in which the first robot is in, when the second robot reaches block n. We also let T_n be the time needed to execute the n'th block $\widetilde{D}_{f_n}((S_n S_n)^{\ell\bar{\ell}})$, where $\widetilde{D}$ is the modified delay scheme. Notice that $T_n = O(|S_n| f_n) = \tilde{O}(n^c)$.

We now have several cases.

- Assume $f_k \geq |S_{n \log^2 n}|$. Let us look at the time when the second robot starts the active part of block $n \log^2 n$, and let us denote $m = |S_{n \log^2 n}| = O(n^c \log^{2c} n \log^b n)$. During that period the first robot has a continuous waiting period that lasts at least $\frac{m}{\log m}$ time (because $m \leq f_k$). However, $\frac{m}{\log m} = O(n^c \log^{2c-1} n \log^b n) \geq O(n^c \log^b n) \cdot f_{n \log^2 n} = p(n) f_{n \log^2 n}$, and within that period the second robot executes at least $p(n)$ steps. As the sequence the second robot runs is a *strong* UTS, the robot covers the whole graph and the two robots meet. We can therefore assume that $(\log k)^c (\log \log k)^{2c+b} \leq n^c (\log n)^{2c+b}$ and in particular $\log k \leq n$ and $k \leq 2^n$.
- Also, if $k \leq 10n \log n$ then the same analysis applies as before, and the two robots meet within time $\tilde{O}(n^c)$.

Thus we can assume $10n \log n \leq k \leq 2^n$.

Let t be such that $t - t_2 = T_{1..n \log n}$, i.e., the time when the second robot starts executing block $2n \log n$ (notice that $t - t_2 = \tilde{O}(n^c)$). As $k \geq 10n \log n$, the block lengths of the first robot are much larger than those of the second robot. Thus, we either get a long period where the first robot is active, or a long period where it is passive (and we might need to wait for a short while for such a period, if the first robot is close to a block end). As the second robot keeps changing between active and passive states of length $T_{n \log n}$, we must quickly get to a situation where one robot is active and the other is passive for $T_{n \log n}$ time.

If the first robot is passive during the execution of $S_{n \log n}$ we are fine. Otherwise, the first robot is active and the second is passive for $T_{n \log n} = O(n^c \log^{c+b} n)$ time. Now, because $k \leq 2^n$, within any $O(p(n) \log(n)) = O(T_{n \log n})$ time units, the first robot does at least $p(n)$ steps (the condition $k \leq 2^n$ is important here, because it guarantees the robot never waits for more than $n^c \log^b n$ time). Also, as $n \geq \log k$ and because S_k is a $[\log n, n]$ SUTS, it does not matter where the first robot is in his sequence, and it covers the whole graph. In particular the two robots meet. □

5.4 Dealing with arbitrary labels We modify the labels. We start with a label $\ell = \ell_1 \ldots \ell_c$. Following Dessmark et al. we let $\ell' = M(\ell) = \ell_1 \ell_1 \ldots \ell_c \ell_c 01 \in \{0,1\}^{2(c+1)}$. This ensures that even if ℓ is a prefix of ℓ', $M(\ell)$ is not a prefix of $M(\ell')$. We now describe the sequence each robot runs. The n'th block in that sequence (for n a power of two) for a robot with label ℓ is: $\widetilde{D}_{f_n}((S_n S_n)^{M(\ell)})$. We claim that:

THEOREM 5.3. *The two robots meet after $\tilde{O}(l \cdot n^c)$ time, where n is the size of the graph, and l is the length of the shorter label.*

Proof: The proof follows the previous ones. We call the robot who started executing block n first, *the first robot* (note that this time it may be that the robot who was activated later becomes the first robot). The other robot is called the second robot. We let k be the block length of the first robot, when the second robot starts block n. If $k > 2^n$ then the same argument as before applies. This is because that argument did not use the labels, and also did not rely on the number of chunks in a block. The same applies to the case where $k \geq 10n \log n$. Thus, we can focus on the case $k \leq 10n \log n$.

Assume $k \leq n' = 10n \log n$. We now redefine first and second. We call the robot who started executing block n' first, the first robot, and the other the second robot. Now, if the first robot gets to run a whole chunk of block n' before the second robot starts block n', then the same argument as before applies. Otherwise, the offset between the two robots, when the second robot starts block n', is less than one chunk. Now, a robot with label ℓ, runs $(S_m S_m)^{M(\ell)}$. As $M(\ell_1)$,$M(\ell_2)$ are not prefixes of each other, there is an index $1 \leq i \leq 2(l+1)$, such that they differ at index i. As the chunk lengths are now equal (because they run the same block) the offset between the robots is kept, and is less than a

chunk length. In particular, one robot is asleep while the other is awake for a whole execution of S_m. The two robots meet! □

6 Universal and strongly universal exploration sequences

Let $G = (V, E)$ be a d-regular graph. As we saw in Section 4, a sequence $\sigma_1\sigma_2\cdots\sigma_k \in \{1, 2, \ldots, d\}^k$ and a starting vertex $v_0 \in V$, defines a walk $v_0, v_1, \ldots, v_k$ in G. In a similar way, a sequence $\tau_1\tau_2\cdots\tau_k \in \{0, 1, \ldots, d-1\}^k$ and a starting *edge* $e_0 = (v_{-1}, v_0) \in E$ define a walk $v_{-1}, v_0, \ldots, v_k$ as follows: For $1 \le i \le k$, if (v_{i-1}, v_i) is the s-th edge of v_i, let $e_i = (v_i, v_{i+1})$ be the $(s+\tau_i)$-th edge of v_i, where we assume here that the edges of v_i are numbered $0, 1, \ldots, d-1$, and that $s + \tau_i$ is computed modulo d.

Definition 6.1. (Universal Exploration Sequences (UESs)) *A sequence $\tau_1\tau_2\cdots\tau_\ell \in \{0, 1, \ldots, d-1\}^\ell$ is a* universal exploration sequence *for d-regular graphs of size at most n if for every connected d-regular graph $G = (V, E)$ on at most n vertices, any numbering of its edges, and any starting edge $(v_{-1}, v_0) \in E$, the walk obtained visits all the vertices of the graph.*

The existence of UES of length $O(d^2n^3 \log n)$ for d-regular graphs of size at most n can be shown using the same probabilistic argument used to show the existence of such UTS. However, while we do not have *explicit* polynomial-size UTS, Reingold [**?**] obtains an explicit construction of polynomial-size UES:

Theorem 6.1. ([**?**]) *There exists a constant $c \ge 1$ such that for every $d \ge 3$ and $n \ge 1$, a UES of length $O(n^c)$ for d-regular graphs of size at most n can be constructed, deterministically, in polynomial time.*

Reingold's explicit UESs can be easily used to turn the deterministic solutions for the rendezvous problem presented in the previous section into explicit solutions, in the variant of the model in which a robot is told which edge it used to enter a vertex. Note that the knowledge of this edge is needed to trace the walk defined by the UES. We note, however, that the explicit solutions obtained are much less efficient than the non-explicit solutions, as the constant c in Reingold's construction is large. As a natural analog of SUTS, we can define:

Definition 6.2. (Strongly Universal Exploration Sequences (SUESs)) *A possibly infinite sequence $\tau = \tau_1\tau_2\ldots$, where $\tau_i \in \{0, 1, \ldots, d-1\}$, is a* strongly *universal exploration sequence (SUES) for d-regular graphs with cover time $p(\cdot)$, if for any $n \ge 1$, any contiguous subsequence of τ of length $p(n)$ is a UES for d-regular graphs of size n.*

While we cannot show the existence of strongly universal *traversal* sequences (SUTSs), even non-explicitly, the main Theorem of this section shows that strongly universal *exploration* sequences (SUESs) do exist and they can be constructed deterministically in polynomial time.

Theorem 6.2. *If for every $n \ge 1$ there are UES of length $O(n^c)$ for d-regular graphs of size at most n, then there is an infinite SUES for d-regular graphs with cover time $O(n^{2c})$. Furthermore, if the UESs can be constructed deterministically in polynomial time, then so can the SUES.*

We now start working towards proving the theorem. The crucial property of exploration sequences used in the proof of Theorem 6.2 is that walks defined by exploration sequences can be *reversed*. For $\tau = \tau_1\tau_2\cdots\tau_k \in \{0, 1, \ldots, d-1\}^k$, we let $\tau^{-1} = \tau_k^{-1}\tau_{k-1}^{-1}\cdots\tau_1^{-1}$, where $\tau_i^{-1} = d - \tau_i$. It is not difficult to check that a walk defined by an exploration sequence τ can be backtracked by executing the sequence $0\tau^{-1}0$. Note that if $e_0, e_1, \ldots, e_k$ is the sequence of edges defined by τ, starting with e_0, then executing $0\tau^{-1}0$, starting with e_k defines the sequence $e_k, \overleftarrow{e}_k, \overleftarrow{e}_{k-1}, \ldots, \overleftarrow{e}_0, e_0$, where $\overleftarrow{e}$ is the *reverse* of edge e, i.e. if $e = (u, v)$, then $\overleftarrow{e} = (v, u)$. Also, it is not difficult to see that if τ is a universal exploration sequence for graphs of size at most n, then so is τ^{-1}.

Let U_n be a sequence of length n which is a universal exploration sequence for d-regular graphs of size at most cn^α, for some $c > 0$ and $0 < \alpha < 1$. [2] We are interested in sequences U_n only when n is a power of 2. We may assume that for every $k = 2^i$ and $n = 2^j$, where $i < j$, U_k is a prefix of U_n. If this condition does not hold, we can replace the sequence U_n by the sequence $\bar{U}_n = U_1U_1U_2U_4\ldots U_{n/2}$. (Recall that n is assumed to be a power of 2. The sequence U_1 is used twice to ensure that $|\bar{U}_n| = n$.)

We now define recursively a sequence S_n of *strongly* universal exploration sequences. We begin with $S_1 = U_1$. Assume that $U_n = u_1u_2\ldots u_n$ and that $n \ge 2$. Define,

$$S_n = u_1\, S_{r_1} 0 S_{r_1}^{-1} 0\; u_2\, S_{r_2} 0 S_{r_2}^{-1} 0\; u_3\; \cdots \\ u_i\, S_{r_i} 0 S_{r_i}^{-1} 0\; u_{i+1}\; \cdots\; u_{n-1}\, S_{r_{n-1}} 0 S_{r_{n-1}}^{-1} 0\; u_n\ ,$$

[2] It is more convenient here to let U_n be a universal sequence of *length* n, rather than a universal sequence for graphs of size n.

where for every $1 \le i < n$, we let $r_i = 2^j$ where $2^{2j} \mid i$ but $2^{2(j+1)} \not\mid i$. Note that as $n = 2^j$, for some $j \ge 1$, for every $k = 2^i$, where $i < j$, the sequence S_k is a prefix of S_n. Also, for every $1 \le i < n$, we have $r_i = r_{n-i}$. The sequence S_n^{-1} thus differs from S_n only in elements that originate from U_n and in the alignment of the 0's:

$$S_n^{-1} \;=\; u_n^{-1}\, 0 S_{r_1} 0 S_{r_1}^{-1}\; u_{n-1}^{-1}\, 0 S_{r_2} 0 S_{r_2}^{-1}\; u_{n-2}^{-1} \;\cdots$$
$$u_{i+1}^{-1}\, 0 S_{r_{n-i}} 0 S_{r_{n-i}}^{-1}\; u_i^{-1} \;\cdots\; u_2^{-1}\, 0 S_{r_{n-1}} 0 S_{r_{n-1}}^{-1}\, u_1^{-1}\,.$$

For every real number $x \ge 1$, we let $\lfloor\!\lfloor x \rfloor\!\rfloor = \max\{2^i \mid 2^i \le x\ ,\ i \in \mathbb{Z}^+\}$ be the largest power of 2 less than or equal to x. For every $x \ge 1$ we have $\lfloor\!\lfloor x \rfloor\!\rfloor \ge x/2$. Note that if i is a power of 2, then $r_i = \lfloor\!\lfloor \sqrt{i} \rfloor\!\rfloor$. In particular, if $n = 2^j$, where $j \ge 1$, then $r_{n/2} = \lfloor\!\lfloor \sqrt{n/2} \rfloor\!\rfloor$. Thus, the first half of S_n is equal to $S_{n/2} S_{\lfloor\!\lfloor \sqrt{n/2} \rfloor\!\rfloor} 0$, and thus ends with a copy of $S_{\lfloor\!\lfloor \sqrt{n/2} \rfloor\!\rfloor}$, followed by a 0. Similarly, the second half of S_n starts with a copy of $S^{-1}_{\lfloor\!\lfloor \sqrt{n/2} \rfloor\!\rfloor}$. (Note that S_n is of even length.) We next bound the length of S_n.

LEMMA 6.1. *For every $n = 2^j$, where $j \ge 1$, we have $|S_n| < 20n$.*

Proof: Let $s_n = |S_n|$. It is not difficult to see that

$$|S_{2^i}| \;=\; |U_{2^i}| + (|U_{2^i}| - 1) \cdot 2(|S_1| + 1) + \sum_{j=1}^{i/2-1} \left\lfloor \frac{|U_{2^i}| - 1}{2^{2j}} \right\rfloor 2(|S_{2^j}| - |S_{2^{j-1}}|)\,,$$

and hence $s_{2^i} \le 2^i \left(5 + 2\sum_{j=1}^{i/2-1} \frac{s_{2^j} - s_{2^{j-1}}}{2^{2j}}\right)$. It is not difficult to check that $s_1 = 1$, $s_2 = 5$, $s_4 = 15$, $s_8 = 31$ and $s_{16} = 87$. The claim that $s_n \le 20n$ for every $n \ge 16$ then follows using simple induction. (In fact, $\lim_{n\to\infty} s_n/n = 8.61741\ldots$.) □

The sequence S_n possesses the following interesting combinatorial property:

LEMMA 6.2. *Let k and $n \ge 2k^2$ be powers of 2. Then, every subsequence T of S_n or S_n^{-1} of length $s_{2k^2} + 1 = |S_{2k^2}| + 1 \le 40k^2$ contains, as a contiguous subsequence, a full of S_k or S_k^{-1}.*

Proof: We prove the claim by induction on n. If $n = 2k^2$ then the claim is vacuously satisfied as S_n and S_n^{-1} are too short to contain a subsequence of length $s_{2k^2} + 1 = s_n + 1$.

Assume, therefore, that the claim holds for every $m = 2^{j'}$ that satisfies $2k^2 \le m < n = 2^j$. We show that it also holds for n. Let T be a subsequence of S_n of length $s_{2k^2} + 1$. Essentially the same argument works if T is a subsequence of such length of S_n^{-1}. We consider the following cases:

Case 1: T is completely contained in a subsequence S_m or S_m^{-1} of S_n, for some $m < n$.

The claim then follows immediately from the induction hypothesis.

Case 2: T is completely contained in a subsequence $S_m 0 S_m^{-1}$ of S_n, for some $m < n$.

In this case, $T = T'0T''$, where T' is a suffix of S_m and T'' is a prefix of S_m^{-1}. Either $|T'| \ge \frac{1}{2} s_{2k^2}$ or $|T''| \ge \frac{1}{2} s_{2k^2}$. Assume, for concreteness, that $|T''| \ge \frac{1}{2} s_{2k^2}$. The other case is analogous. As T'' is a prefix of S_m^{-1}, and $|T''| \ge \frac{1}{2} s_{2k^2}$, it follows that $m \ge 2k^2$. Now, S_k^{-1} is almost a prefix of S_m^{-1}, in the sense that they differ only in symbols that originate directly from S_m. In particular, a prefix of S_m^{-1} of length $\frac{1}{2} s_{2k^2}$, half the length of S_{2k^2}, ends with a full copy of S_k, followed by 0.

Case 3: T contains a symbol u_ℓ of S_n that originates from U_n.

In this case, $T = T' u_\ell T''$. Again, we either have $|T'| \ge \frac{1}{2} s_{2k^2}$ or $|T''| \ge \frac{1}{2} s_{2k^2}$. Assume again, for concreteness, that $|T''| \ge \frac{1}{2} s_{2k^2}$. The other case is analogous. Let

$$S_{n,\ell} = u_\ell\, S_{r_\ell} 0 S_{r_\ell}^{-1} 0\, u_{\ell+1} \cdots u_{n-1}\, S_{r_{n-1}} 0 S_{r_{n-1}}^{-1} 0\, u_n$$

be the suffix of S_n that starts with the symbol u_ℓ that originates from the ℓ-th symbol of U_n. We claim that the prefix of $S_{n,\ell}$ of length $\frac{1}{2} s_{2k^2}$ contains a copy of S_k. Let $\ell' = \lceil \ell/k^2 \rceil k^2$ be the first index after ℓ which is divisible by k^2. Clearly $r_{\ell'} \ge k$ and hence S_k is a prefix of $S_{r_{\ell'}}$. Thus, $S' = u_\ell\, S_{r_\ell} 0 S_{r_\ell}^{-1} 0 \cdots u_{\ell'} S_k$ is a prefix of $S_{m,\ell}$ which ends with a full copy of S_k. As for every $\ell \le i < \ell'$ we have $r_i = r_{i \bmod k^2}$, we get that S' is contained in the first half of S_{2k^2}, and hence $|S'| \le \frac{1}{2} s_{2k^2}$, as required. □

We are now ready to prove Theorem 6.2:

Proof: We let $E_n = S_{bn^{2c}}$ for some constant b that will be specified later. By Lemma 6.2 we know that for every m, any subsequence of size m of E_n contains as a contiguous subsequence a full copy of $S_{m'}$ or $S_{m'}^{-1}$ for $m' = \Omega(\sqrt{m})$. W.l.o.g., let us assume it contains $S_{m'}$.

We now look at the recursive definition of $S_{m'}$ and ignore all the recursive calls of S_j (for $j < m'$) and their inverses. We can ignore these parts because S_j^{-1} reverses the actions of S_j. We are left with $U_{m'} = u_1, \ldots, u_{m'}$. However, $U_{m'}$ is a UES for graphs of size $n = \Omega((m')^\alpha)$. In particular, any subsequence of E_n of size m covers all graphs of size n.

We get that the cover time is $p(n) = m = O((m')^2) = O(n^{2/\alpha}) = O(n^{2c})$ as desired. The constant b in the definition of E_n is the constant hiding in the $O(\cdot)$ notation (i.e., $E_n = S_{p(n)}$). The length of E_n is that of $S_{p(n)}$. Lemma 6.1 tells us that this is at most $O(p(n)) = O(n^{2c})$. □

This, in particular, gives (an explicit or non-explicit) solution to the treasure hunt problem: all the robot has to do is to run the SUES. The adversary decides when to put the treasure, but then the subsequence of length $p(n)$ starting at this point is a UES and the robot finds the treasure.

7 Concluding remarks and open problems

We obtained improved deterministic solutions for the rendezvous problem that are independent of τ, the difference between the activation times of the two robots. Furthermore, we get close to the length of a UTS. With backtracking, we obtain a polynomial time, *explicit* solution.

The technique used in the paper raises the question whether there exist *strongly* UTSs. We define *the treasure hunt problem* which is the variant of the problem where one robot is static and always stays in the place where it is put. Strong UTS exist iff the treasure hunt problem has a solution that is independent of τ. Standard probabilistic arguments used to show the existence of (non-strong) universal traversal and exploration sequences cannot be used to prove the existence of *strongly* universal sequences, with *any* cover time. We can, however, show an explicit construction of strong universal *exploration* sequences. We do not know wether strong universal *traversal* sequences exist. We believe this last question is very natural and deserves further study.

Acknowledgements

We thank Oded Goldreich for a discussion that led to the definition of the treasure hunt problem.

References

[DFKP06] A. Dessmark, P. Fraigniaud, D. Kowalski, and A. Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 2006. Online publication June 19, 2006.

[DFP03] A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. In *Proc. of 11th ESA*, pages 184–195, 2003.

[DGK+06] G. De Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, and U. Vaccaro. Asynchronous deterministic rendezvous in graphs. *Theoretical Computer Science*, 355(3):315–326, 2006.

[KM06] D.R. Kowalski and A. Malinowski. How to meet in anonymous network. In *Proc. of 13th SIROCCO*, pages 44–58, 2006.

[KP04] D.R. Kowalski and A. Pelc. Polynomial deterministic rendezvous in arbitrary graphs. In *Proc. of 15th ISAAC*, pages 644–656, 2004.