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 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 two robots are placed in an unknown environment. The goal is to give the two robots *deterministic* sequences of instructions that ensure that they would eventually meet, no matter on which graph they are placed, and no matter when they are activated.

Formally, we model the environment by a finite, connected, undirected graph $G = (V, E)$. 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. The adversary chooses the environment, and also when to activate each robot, and where to place a robot when it is activated. In addition, the adversary assigns each robot a unique label that is important for symmetry breaking.[1]

An active robot may traverse the graph. It does so by computing at each time step a deterministic function from the information known to it, to an *instruction*. The information known to a robot includes: the robot's label, the number of time units that elapsed since the robot was activated and the degree of its current vertex.[2] An *instruction* for a robot currently in vertex $u \in V$, is a value from the set $\{0, 1, \ldots, \deg(u)\}$, with the interpretation that 0 means that the robot stays in u, while a value $0 < i \leq \deg(u)$ means that the robot traverses the i'th edge leaving u. If a robot traverses the edge $(u, v) \in E$ leaving u, then the robot finds itself at v, the other endpoint of this edge.

A solution to the *rendezvous problem* is an algorithm such that for any adversary (i.e., any environment, any two different labels, any activation times and any two initial vertices) the two robots eventually meet. We assume the two robots move synchronously.[3] The two robots meet only when they

[*]School of Computer Science, Tel Aviv University, Tel Aviv 69978,Israel. E–mail: {amnon,zwick}@tau.ac.il. Amnon thanks the Israel Science Foundation Grant No. 1090/10. Uri's research was supported by a GIF grant. A preliminary version of the paper appeared in SODA 2007.

[1]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!

[2]Notice that the robots do not know the size of the graph. The knowledge of the graph size makes the problem easy to solve.

[3]For an asynchronous version of the model, see De Marco et al. [DGK+06].

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. As the robots can only meet when they are both active, the *cost* of a solution on a specific instance set by the adversary, is the number of time units that elapse from the activation of the second robot to the robots' rendezvous.

Dessmark et al. [DFKP06] (see also [DFP03] and [KP04]) presented a 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. 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 $n^{10}\ell$, 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.

1.1 Backtracking. Kowalski and Malinowski [KM08] presented a deterministic solution to the rendezvous problem that guarantees a meeting after at most $\tilde{O}(n^{15} + \ell^3)$ steps. The number of steps performed by their solution is independent of τ. However, the solution crucially uses *backtracking*, as we explain now.

As we said before, the adversary chooses the environment. I.e., it chooses the graph size, n, the graph $G = (V, E)$ itself, and also the edge labels. In general, the labeling 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 $i \neq j$, and in fact some graphs do not have any consistent labeling at all (e.g., the 3–cycle).

The solution Kowalski and Malinowski present, assumes that when a robot enters a node v after traversing an edge $e = (u, v)$, it is told the index j of the edge e at v. In particular, the robot may now choose the instruction j, causing it to backtrack to v. Formally, the information known to the robot now includes:

- The robot's label L,
- The number of time units T that elapsed since the robot was activated,
- The degree of the current vertex d, and,
- A sequence $j_1, \ldots, j_T$, such that the edge $e_t = (v_t, u_t)$ traversed at time t, has index j_t at vertex u_t.

As before the robot's move is determined by a deterministic function $f(L; T; d; j_1, \ldots, j_T) \in \{0, 1, \ldots, d\}$. We call this variant of the problem the *rendezvous problem with backtracking*.

1.2 Our results. A central result of our work is a deterministic solution that guarantees a rendezvous within $\tilde{O}(n^5\ell)$ time units after the activation of the second robot. When the unknown graph in which the two robots are placed 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 $\tilde{O}(dn^3\ell)$. In addition to being independent of τ, our solution is more efficient than previous solutions for all parameters and even if τ is small. More importantly, our solution (including the fastest one) does *not* use backtracking.

In addition, we introduce the *treasure hunt* problem. In this 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, 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. The second main result in the paper is an efficient algorithm for solving the treasure hunt problem when backtracking is allowed. We do not know if an efficient solution is possible when backtracking is not allowed.

1.3 Traversal sequences and non-backtracking algorithms. Aleliunas et al. [AKL$^+$79] showed a random walk, with high probability, covers all vertices of a graph within polynomial time. Indeed, 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 (in the size of the graph) number of steps, see, e.g., [Ald91, CTW93].

The random walk approach can be de-randomized as follows. Let $G = (V, E)$ be a d-regular graph. A sequence $\sigma_1 \dots \sigma_k \in \{1, \dots, d\}^k$ defines a walk $v_0, v_1, \dots, v_k$ in G, that starts at $v_0 \in V$, where v_i is the σ_i-th neighbor of v_{i-1}. A walk $v_0, v_1, \dots, v_k$ is said to *cover* a graph G if it visits every vertex of G at least once. A universal traversal sequence [AKL$^+$79] for n-vertex graphs is a predetermined sequence of instructions that when executed on *any* n-vertex graph, from *any* starting vertex, defines a walk that visits all the vertices of the graph. Formally,

Definition 1.1. (Universal Traversal Sequences (UTSs)) *A sequence $\sigma \in \{1, \dots, d\}^\ell$ 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.*

A neat probabilistic argument was used by Aleliunas et al. [AKL$^+$79] 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!

Indeed, the treasure hunt problem, with the additional guarantee that the treasure is present in the graph when the robot is activated, can be simply solved by letting the robot follow the sequence $U_1 U_2 U_4 \dots U_{2^k} \dots$, where U_i is a UTS for graphs of size i. A similar strategy solves the rendezvous problem when the robots know the graph size, or when the robots are simultaneously activated.

Now, consider the treasure hunt problem when the robot starts roaming the graph *before* the treasure is placed in the graph. Assume the robot employs the same strategy as before, namely, it follows the sequence $U_1 U_2 U_4 \dots U_{2^k} \dots$. In this case, the adversary may wait with placing the treasure until the robot is inside some sequence U_k, for k that is much larger than the actual graph size n. The UTS property of U_k is valid only when starting at the beginning of the sequence, not when started from the middle. Thus, it seems we have to wait until the next sequence U_{2k} starts, which incurs a $poly(k)$ loss, which for a large k is super-polynomial in n.

We are thus lead 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?

Definition 1.2. (Strongly Universal Traversal Sequences (SUTSs)) *Let $p(\cdot)$ be a polynomial. A possibly infinite sequence $\sigma = \sigma_1 \sigma_2 \dots$, where $\sigma_i \in \{1, \dots, d\}$, is a* strongly *universal traversal sequence (SUTS) with cover time $p(\cdot)$ for connected d-regular graphs, if for any $k \geq 1$, any contiguous subsequence of σ of length $p(k)$ is a UTS for connected d-regular graphs of size k.*

Note that if for every $n \geq 1$, there is a SUTS S_n of length n with cover time $p(k) \geq k$, then $S_1 S_2 S_4 S_8 \dots$ is an infinite SUTS with cover time $2p(\cdot)$.

SUTSs, if exist, would solve both the rendezvous and the treasure hunt problems. We do not know, however, whether SUTSs exist, and this is the main open problem raised by our work. Instead, we can show using the standard probabilistic method the existence of slightly weaker SUTSs which are sufficient for solving the rendezvous problem, but not the treasure hunt problem.

DEFINITION 1.3. *A sequence $\sigma \in \{1, \ldots, d\}^{\ell}$ is a* strongly universal traversal sequence *(SUTS) with cover time $p(\cdot)$ for connected d-regular graphs of size $[n', n'']$, 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.*

LEMMA 1.1. *For every n, there is an $[\log n, n]$-SUTS with cover time $p(k) = O(d^2 k^3 \log k)$ for d-regular graphs.*

Proof: Aleliunas et al. [AKL$^+$79] 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 $O(n \cdot d^2 n^3 \log n) \cdot 2^{-\log^2 n} < 1$, this is because there are less than n graph sizes, only $O(d^2 n^3 \log n)$ places to start a subsequence, 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.[4] □

Kowalski and Malinowski [KM08] 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 essentially equivalent to Lemma 1.1.

It is worthwhile noting that the standard probabilistic method *cannot* be used to obtain SUTSs that satisfy the conditions of Definition 1.2. 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 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 [EL75] or Alon and Spencer [AS00]). Unfortunately, all our attempts to prove the existence of SUTSs using the local lemma failed.

1.4 Exploration sequences and backtracking algorithms. One of the motivations for the introduction of universal traversal sequences by Aleliunas et al. [AKL$^+$79] 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 [Rei08], thus resolving this major open problem. Reingold [Rei08] 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 [Kou02]. Roughly speaking, exploration sequences can replace traversal sequences when backtracking is allowed.

We analogously define strong universal exploration sequences (see Section 3) and show the existence of strong universal *exploration* sequences with polynomial cover times. Furthermore, our construction uses (non-strong) universal exploration sequences as black-boxes, and is efficient otherwise. In particular, when using Reingold's explicit construction of polynomially long universal exploration sequences [Rei08], we get an *explicit* construction of strongly universal exploration sequences. This is used in our solution of the treasure hunt problem when backtracking is allowed.

[4] 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.

1.5 Further remarks. All graphs in this paper are assumed to be connected.

In the discussion so far we often assumed that the graph G in which the robots are placed is a d-regular graph, for some $d \geq 3$. We now explain why this simplifying assumption is immaterial.

Our constructions compose universal traversal and exploration sequences. The only property of these sequences used in the analysis is that the n'th sequence is good for all graphs of size at most n and regular degree n. The instruction set of these sequences include 0 for staying at the current vertex, and $i \in \{1, \ldots, n\}$ for traversing the i'th edge. We enforce a simple rule. If a robot at vertex u with $deg(u) < n$ gets an instruction $i > \deg(u)$, then it simply stays in place. We can think of this as adding $n - \deg(u)$ self loops to u, in particular making the new degree of u exactly n.

To see why this works, suppose G is the graph in which the two robots walk, and that G has n vertices and varying degrees $\deg(u) \leq n$. Let G' be the graph G with $n - \deg(u)$ self loops added at each vertex u. G' is a graph on n vertices with regular degree n. In particular, the universal sequences are good for G' and the construction works.

So far we were only interested in the *existence* of a deterministic function that guarantees fast rendezvous, and we ignored the complexity of the function f itself. Indeed, our non-backtracking algorithms (as well as the solutions of Dessmark et al. [DFKP06] and [KM08]) rely on the existence of *universal traversal sequences*. As we said before, the existence of universal traversal sequences is proved using the probabilistic method. 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*. This lead Dessmark et al. [DFKP06] to ask whether there is a solution in which the number of steps and the computation required to determine them are both polynomial. When backtracking is allowed, we answer this last question in the affirmative, using our explicit construction of universal exploration sequences. We do not have such a construction when backtracking is not allowed.

Finally, we remark that our exploration sequences were improved and shortened by Xin [Xin07].

2 Deterministic rendezvous solutions

2.1 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 $U_1 U_2 U_4 \ldots U_{2^k} \ldots$, 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 $U_1 U_2 U_4 \ldots U_{2^k} \ldots$ 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 $U_1 U_2 U_4 \ldots U_{2^k} \ldots$, 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.

The next idea we explore 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 possible 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. We explain the solution in Section 2.2. 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, let us say a robot is in *block* i if it is walking according to U_i. (We assume here that i is a power of 2.) A robot in block i 'rests' for f_i time units after each instruction of U_i that it makes. The total number of time units required for implementing block i is therefore $|U_i|(f_i + 1)$.

Suppose the two robots run an n-vertex graph. The crucial parameter used in the analysis of our rendezvous solutions is the index $\kappa = \kappa(n)$ of the block in which the robot who started first is at the time the second robot starts implementing its block n. (For simplicity, we assumed here that n is a power of 2). If $f_\kappa > |U_n|(f_n + 1)$ 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 $f_\kappa \leq |U_n|(f_n + 1)$ 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.

2.2 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, and will be described later on.

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:

$$D_{u_1-1}((U_1U_1)^{\ell\bar{\ell}})\, D_{u_2-1}((U_2U_2)^{\ell\bar{\ell}}) \cdots D_{u_{2^k}-1}((U_{2^k}U_{2^k})^{\ell\bar{\ell}}) \cdots$$

We let $W_i = D_{u_i-1}(U_i)$, $B_i^{(0)} = 0^{|W_i|}0^{|W_i|}W_iW_i$ and $B_i^{(1)} = W_iW_i0^{|W_i|}0^{|W_i|}$. The sequence used by robot 0 is $B_1^{(0)}B_2^{(0)}B_4^{(0)}\ldots$ and the sequence used by robot 1 is $B_1^{(1)}B_2^{(1)}B_4^{(1)}\ldots$. We call $B_i^{(\ell)}$ the i-th block of robot ℓ (even though this is a slight abuse of notation: i is always a power of two, and B_i is the $\log i$-th block). 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 $w_n = u_n^2 = \Theta(n^{2c})$ and $b_n = 4w_n = 4u_n^2 = \Theta(n^{2c})$. We make the technical assumption that $4u_n \leq u_{2n}$, for every $n = 2^i$.[5] It follows that $16w_n \leq w_{2n}$

[5] This essentially corresponds to the assumption that $u_n = |U_n| = \Omega(n^2)$ that we can assume without loss of generality.

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_ℓ, then by the time the second robot starts executing block $B_{2\ell}$, the block that follows B_ℓ, the first robot has finished much less than a fourth of this block. We are now ready to prove:

THEOREM 2.1. *If for every $k \geq 1$, U_k is a UTS for graphs of size k, and $|U_k| = \Theta(k^c)$, $c \geq 2$, 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 κ 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 $\kappa \geq n$. We consider two cases:

Case 1: $u_\kappa \geq b_n$. As the first robot 'sleeps' for u_κ 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 2.2. (It is assumed in the figure that the second robot has label 0. The other case is similar.)

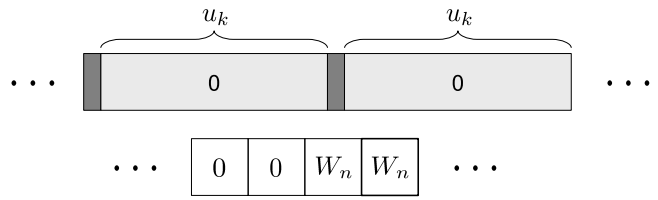

Figure 1: Case 1 in the proof of Theorem 2.1 ($u_\kappa \geq b_n$).

Case 2: $u_\kappa < b_n$. As $u_\kappa = \Theta(\kappa^c)$ and $b_n = \Theta(n^{2c})$, it follows that $\kappa \leq O(n^2)$. The second robot to be activated thus finishes executing its κ-th block, and starts executing his 2κ-th block, after $O(\kappa^{2c}) = O(n^{4c})$ time units. When this happens, the first robot, as discussed in the paragraph just before Theorem 2.1, is still executing the first forth of its 2κ-th block. The situation is therefore as depicted in Figures 2.2 and 2.2. In the first figure the first robot has label 0 while in the second figure the first robot has label 1. In both cases one player is asleep while the other player executes a whole copy of $W_{2\kappa}$. The two robots are therefore guaranteed to meet before the 2κ-th block is over.

□

2.3 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 $p(k) = O(k^c)$ and length $|S_n| = p(n)$ (see Lemma 1.1). Second, we reduce the delay time performed after each step from $f_k = |S_k|$ to $f_k = (\log k)^c (\log\log k)^{2c}$ 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. Formally, let

$$\widetilde{D}_k(\sigma_1 \dots \sigma_m) \;=\; \sigma_1 0^{r_1} \sigma_2 0^{r_2} \dots \sigma_m 0^{r_m}.$$

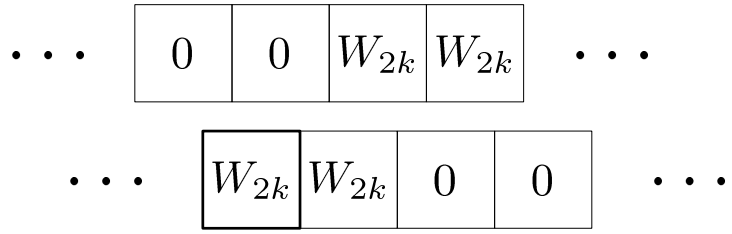

Figure 2: Case 2 in the proof of Theorem 2.1. The first robot has label 0.

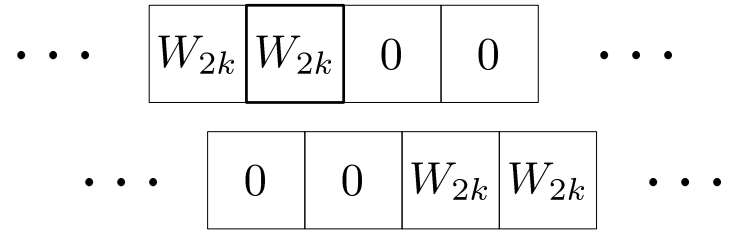

Figure 3: Case 2 in the proof of Theorem 2.1. The first robot has label 1.

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

$$\widetilde{D}_1((S_1 S_1)^{\ell\bar{\ell}})\, \widetilde{D}_2((S_2 S_2)^{\ell\bar{\ell}}) \cdots \widetilde{D}_{2^k}((S_{2^k} S_{2^k})^{\ell\bar{\ell}}) \cdots$$

Before we proceed, we prove the following useful claim:

Claim 1. *If a robot that is in block k and m is a power of two, $8 \leq m \leq f_k$, then every time interval of length $t = 2m \log m$ contains in it a continuous waiting period of length at least m.*

Proof. Assume not. Fix t consecutive steps of the robot while in block k that do not contain a waiting period of length m. The t steps include some original steps $\sigma_{q+1}, \ldots, \sigma_{q+\ell}$ interleaved in between idle steps. There are at most $\lceil \frac{\ell}{2^i} \rceil$ waiting periods of 2^i time in the length t sequence, for $2^i = 1, 2, 4, 8, \ldots, \frac{m}{2}$, and in particular the sequence length is at most $\ell + \lceil \frac{\ell}{2^0} \rceil + 2\lceil \frac{\ell}{2^1} \rceil + \ldots + 2^i \lceil \frac{\ell}{2^i} \rceil + \ldots + \frac{m}{2} \lceil \frac{\ell}{m/2} \rceil \leq \ell(1 + \log m) + m$. Thus, $t = 2m \log m \leq \ell(1 + \log m) + m$. This implies $\ell \geq m + 1$. In particular, there exists at least one instruction σ_{q+i} (for $i < \ell$) after which there is a waiting period of length m. A contradiction.

We are now ready to prove:

Theorem 2.2. *The two robots meet after $\tilde{O}(n^c)$ steps, where n is the size of the graph.*

Proof: Let κ 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 $\widetilde{D}_n((S_n S_n)^{\ell\bar{\ell}})$, 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 $f_\kappa \geq |S_{n\log^2 n}|$. In this case the robots are not aligned and the first robot starts much earlier than the second one. Let us look at the time when the second robot starts the active part of block $n\log^2 n$, and let us denote $m = |S_{n\log^2 n}| = p(n\log^2 n) = O(n^c \log^{2c} n)$. During that period the first robot has a continuous waiting period that lasts at least $\frac{m}{2\log m}$ time (because $m \leq f_\kappa$). However, $\frac{m}{2\log m} = O(n^c \log^{2c-1} n) \geq O(n^c) \cdot f_{n\log^2 n} = p(n) f_{n\log^2 n}$, 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 $f_\kappa = (\log\kappa)^c (\log\log\kappa)^{2c} \leq n^c (\log n)^{2c}$ and in particular $\log\kappa \leq n$ and $\kappa \leq 2^n$.

- Also, if $\kappa \leq 10n\log^3 n$ then the robots are roughly aligned and the same analysis applies as in case 2 of Theorem 2.1, and the two robots meet within time $\tilde{O}(n^c)$.

- Thus we can assume $10n\log^3 n \leq \kappa \leq 2^n$.

  Let t be the time when the second robot starts executing block $2n\log^3 n$ (notice that this happens within $O(T_{n\log^3 n}) = \tilde{O}(n^c)$ time from the time the second robot is placed in the graph). As $\kappa \geq 10n\log^3 n$, 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 $T_{n\log^3 n}$, we get to a situation where one robot is active and the other is passive for $T_{n\log^3 n}$ time.

  - If the first robot is passive during the execution of $S_{n\log^3 n}$ we are fine.
  - Otherwise, the first robot is active and the second is passive for $T_{n\log^3 n}$ time. Now, within any $T_{n\log^3 n} \geq p(n\log^3 n) \geq p(n\log^2 n)\log p(n\log^2 n)$ time units, the first robot does at least $p(n\log^2 n) \geq p(n)$ steps (the condition $\kappa \leq 2^n$ is important here, because it guarantees that $f_\kappa \leq n^c \log^{2c} n = p(n\log^2 n)$). Also, as $n \geq \log\kappa$ and because S_κ is a $[\log\kappa, \kappa]$ 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.

□

2.4 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. For a bit b, let $\widetilde{D}_{n,b} = \widetilde{D}_n((S_nS_n)^{b\bar{b}})$. For a sequence $\bar{b} = b_1, \ldots, b_k$, let $\widetilde{D}_{n,\bar{b}} = \widetilde{D}_{n,b_1} \ldots \widetilde{D}_{n,b_k}$. For a robot with label ℓ, the n'th block in the sequence (for n a power of two) is: $\widetilde{D}_{n,M(\ell)}$.

THEOREM 2.3. *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 κ be the block length of the first robot, when the second robot starts block n. If $\kappa > 2^n$ 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 $10n\log^3 n \leq \kappa \leq 2^n$. Thus, we can focus on the case $\kappa \leq 10n\log^3 n$.

Assume $\kappa \leq n' = 10n \log^3 n$, i.e., the robots are almost aligned. 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 first robot chunks are much larger than the second robot chunks, and so one robot is asleep and the other is awake for the duration of a whole chunk of the second robot, and as before the two robots meet.

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! □

3 Universal and strongly universal exploration sequences

Let $G = (V, E)$ be a d-regular graph. As we saw in Section 1.3, a sequence $\sigma_1 \sigma_2 \cdots \sigma_k \in \{0, 1, 2, \ldots, d\}^k$ and a starting vertex $v_0 \in V$ define 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 $1 \leq i \leq k$, 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 3.1. (Universal Exploration Sequences (UXSs)) *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 UXS of length $O(d^2 n^3 \log n)$ 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 [Rei08] obtains an explicit construction of polynomial-size UXS:

Theorem 3.1. ([Rei08]) *There exists a constant $c \geq 1$ such that for every $d \geq 3$ and $n \geq 1$, a UXS of length $O(n^c)$ for d-regular graphs of size at most n can be constructed, deterministically, in polynomial time.*

Reingold's explicit UXSs can be easily used to turn our basic deterministic solution for the rendezvous problem presented in the previous section into an explicit solution, 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 UXS. We note, however, that the explicit solution obtained is much less efficient than the non-explicit solution, as the constant c in Reingold's construction is large.

As a natural analog of SUTS, we can define:

Definition 3.2. (Strongly Universal Exploration Sequences (SUXSs)) *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 (SUXS) for d-regular graphs with cover time $p(\cdot)$, if for any $n \geq 1$, any contiguous subsequence of τ of length $p(n)$ is a UXS 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 (SUXSs) do exist and they can be constructed deterministically in polynomial time.

THEOREM 3.2. *If for every $n \geq 1$ there are UXS of length $O(n^c)$ for d-regular graphs of size at most n, then there is an infinite SUXS for d-regular graphs with cover time $O(n^{2c})$. Furthermore, if the UXSs can be constructed deterministically in polynomial time, then so can the SUXS.*

3.1 The sequence. The crucial property of exploration sequences used in the proof of Theorem 3.2 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}.

In this section it is more convenient to let U_n be a universal sequence of *length* n, rather than a universal sequence for graphs of size n. Let U_n be a sequence of length n which is a universal exploration sequence for d-regular graphs of size at most γn^α, for some $\gamma > 0$ and $0 < \alpha < 1$. 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. The recursion base is $S_1 = U_1$. Assume that $U_n = u_1u_2\ldots u_n$ and that $n \geq 2$. Define,

$$S_n = u_1\, S_{r_1} 0 S_{r_1}^{-1} 0\; u_2\, S_{r_2} 0 S_{r_2}^{-1} 0\; u_3\; \ldots\; u_i\, S_{r_i} 0 S_{r_i}^{-1} 0\; u_{i+1}\; \cdots\; u_{n-1}\, S_{r_{n-1}} 0 S_{r_{n-1}}^{-1} 0\; u_n \tag{3.1}$$

where for every $1 \leq i < n$, we let $r_i = 2^j$ be the largest power of 2 where $2^{2j} \mid i$ but $2^{2(j+1)} \not| i$ (and so r_i is at most $\sqrt{i}$ but is much smaller if i has only few 2 factors).

Before we begin the proof of Theorem 3.2 we show the sequence S_n is *short*.

LEMMA 3.1. *For every $n = 2^j$, where $j \geq 1$, we have $|S_n| < 20n$.*

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

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

We remind the reader that $|U_\ell| = \ell$. It follows that

$$s_{2^i} \leq 2^i \left(5 + 2 \sum_{j=1}^{i/2-1} \frac{s_{2^j} - s_{2^{j-1}}}{2^{2j}} \right).$$

The values $s_1, \ldots, s_8$ can be computed directly from the definition. Doing that gives $s_1 = 1$, $s_2 = 5$, $s_4 = 15$, $s_8 = 31$ and $s_{16} = 87$. The claim that $s_n < 20n$ for every $n \geq 16$ then follows using induction. (In fact, $\lim_{n\to\infty} s_n/n = 8.61741\ldots$.) □

3.2 S_n **is strong.** The key lemma we are using in the analysis is:

LEMMA 3.2. *Let k and $n \geq 2k^2$ be powers of 2. Then, every subsequence T of S_n or S_n^{-1} of length $s_{2k^2}+1 \leq 40k^2$ contains, as a contiguous subsequence, a full copy of S_k or S_k^{-1}.*

The proof of this lemma uses some nice combinatorial properties of the recursion. Before we give a proof to the lemma, we show that it suffices for proving that S_n is strong (Theorem 3.2).

Proof: (of Theorem 3.2) Suppose there are UXS of length $p(n) = O(n^c)$ for d–regular graphs of length at most n. Let $E_n = S_{2p^2(n)}$. The infinite sequence is $E = E_1E_2E_4E_8 \dots E_{2^k} \dots$. Then, by Lemma 3.2, any subsequence of E of length $O(p^2(n))$ contains in it a full copy of $S_{p(n)}$ or $S_{p(n)}^{-1}$. W.l.o.g., let us assume it contains $S_{p(n)}$.

We now look at the recursive definition of $S_{p(n)}$ and ignore all the recursive calls of S_j (for $j < p(n)$) and their inverses. We can ignore these parts because the sequence S_j^{-1} reverses the action of S_j. We are left with $U_{p(n)} = u_1, \dots, u_{p(n)}$. However, $U_{p(n)}$ is a UXS for graphs of size n. In particular, any subsequence of E of size $O(p^2(n)) = O(n^{2c})$ covers all graphs of size n. □

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 SUXS. The adversary decides when to put the treasure, but then the subsequence of length $p(n)$ starting at this point is a UXS and the robot finds the treasure.

3.3 The combinatorial properties of S_n. We now turn to proving Lemma 3.2 which essentially claims that in any *sub*-sequence of length s_{2k^2} there is a full copy of S_k or S_k^{-1}. For the proof of this lemma we use only the recursive definition of S_n. I.e., the claim is true for any sequence $u_1, \dots, u_n$, and we do not use the fact that U_n is a UXS.

We first determine what is S_n^{-1}. We have:

$$S_n^{-1} = u_n^{-1}\,0S_{r_{n-1}}0S_{r_{n-1}}^{-1}\;u_{n-1}^{-1}\,0S_{r_{n-2}}0S_{r_{n-2}}^{-1}\;u_{n-2}^{-1}\cdots u_{i+1}^{-1}\,0S_{r_i}0S_{r_i}^{-1}\;u_i^{-1}\;\cdots\;u_2^{-1}\,0S_{r_1}0S_{r_1}^{-1}\,u_1^{-1} \tag{3.2}$$

$$= u_n^{-1}\,0S_{r_1}0S_{r_1}^{-1}\;u_{n-1}^{-1}\,0S_{r_2}0S_{r_2}^{-1}\;u_{n-2}^{-1}\cdots u_{i+1}^{-1}\,0S_{r_{n-i}}0S_{r_{n-i}}^{-1}\;u_i^{-1}\;\cdots\;u_2^{-1}\,0S_{r_{n-1}}0S_{r_{n-1}}^{-1}\,u_1^{-1}. \tag{3.3}$$

where we used the claim that $r_i = r_{n-i}$ that we soon state and prove. 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.

The fact $r_i = r_{n-i}$ is one of several combinatorial properties of the construction that we now state.

CLAIM 2. *Let n be a power of 2.*

1. *If $k|n$ then the sequence S_k is a prefix of S_n.*
2. *For every $1 \leq i < n$, we have $r_i = r_{n-i}$.*
3. *For any k and i, if $r_i|k$ then $r_i = r_{i \bmod k^2}$.*
4. *Denote $\llbracket x \rrbracket = \max\{2^i \mid 2^i \leq x\ ,\ i \in \mathbb{Z}^+\}$, i.e., $\llbracket x \rrbracket$ is the largest power of 2 less than or equal to x. Then, the first half of S_n is equal to $S_{n/2}S_{\llbracket\sqrt{n/2}\rrbracket}0$. Similarly, the second half of S_n starts with a copy of $S^{-1}_{\llbracket\sqrt{n/2}\rrbracket}$.*

We defer the proof of the claim to later on, and we turn to the proof of Lemma 3.2.

Proof: (of Lemma 3.2) Fix k. 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 $2k^2 \leq m < n = 2^j$. 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} (because Equation (3.2) shows S_n^{-1} is similar to S_n). 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 $|T'| \geq \frac{1}{2}s_{2k^2}$ or $|T''| \geq \frac{1}{2}s_{2k^2}$. As a suffix of S_m is the same as a prefix of S_m^{-1} we focus on just one case. We assume for concreteness that $|T''| \geq \frac{1}{2}s_{2k^2}$.

As T'' is a prefix of S_m^{-1}, and $|T''| \geq \frac{1}{2}s_{2k^2}$, it follows that $m \geq 2k^2$. 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 (this follows from Equation (3.2) and Claim 2(1)). 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 (by Claim 2(4)).

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 $|T'| \geq \frac{1}{2}s_{2k^2}$ or $|T''| \geq \frac{1}{2}s_{2k^2}$. Assume again, for concreteness, that $|T''| \geq \frac{1}{2}s_{2k^2}$. The other case is analogous. Let

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

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 $r_{\ell'} \geq k$ and $r_i | k$ for every $\ell \leq i < \ell'$. Also, S_k is a prefix of $S_{r_{\ell'}}$ (by Claim 2(1)) and so $S' = u_\ell \, S_{r_\ell} 0 S_{r_\ell}^{-1} 0 \cdots u_{\ell'} S_k$ is a prefix of $S_{n,\ell}$ which ends with a full copy of S_k.

For every $\ell \leq i < \ell'$ we have $r_i = r_{i \bmod k^2}$ (by Claim 2(3)), and so up to changing the symbols u_i the sequence S' is contained in the first of half of S_{2k^2} (which also ends with S_k), and hence the length of $|S'|$ is at most $\frac{1}{2}s_{2k^2}$, as required. □

Finally, we turn to proving the claim:

Proof. (of Claim 2)

1. By inspecting Equation (3.1) and because U_k is a prefix of U_n.

2. n is a power of 2. Now, a number Δ that is a power of 2 that divides $1 \leq i \leq n$ must also divide n. Hence $\Delta | i$ iff $\Delta | n - i$. It follows that the largest power of two that divides i is also the largest power of 2 that divides $n - i$ and so $r_i = r_{n-i}$.

3. First, clearly for any k a power of 2 and any i, $r_{i \bmod k} \leq r_i$. We know, however, that k is large enough, i.e., $r_i \mid k$, and so $r_i^2 \mid i$ and $r_i^2 \mid k^2$ and it follows that $r_i^2 \mid i \bmod k^2$. Thus, $r_i \leq r_{i \bmod k^2}$ and therefore $r_i = r_{i \bmod k^2}$.

4. We already saw that $S_{n/2}$ is a prefix of S_n. By inspecting Equation (3.1) we see that in fact the first half of S_n is $S_{n/2} S_{r_{n/2}} 0$. If $n/2$ is a power of 2, $n/2 = 2^i$, then $r_{n/2} = 2^{\lfloor i/2 \rfloor} = \llbracket 2^{i/2} \rrbracket = \llbracket \sqrt{n/2} \rrbracket$, which completes the first assertion in item (4). The second assertion is similarly proved.

4 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 deserves further study.

Acknowledgements

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

References

[AKL+79] R. Aleliunas, R.M. Karp, R.J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *FOCS*, pages 218–223, 1979.

[Ald91] D.J. Aldous. Meeting times for independent Markov chains. *Stochastic Processes and their applications*, 38(2):185–193, 1991.

[AS00] N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley, 2000.

[CTW93] D. Coppersmith, P. Tetali, and P. Winkler. Collisions among random walks on a graph. *SIAM Journal on Discrete Mathematics*, 6(3):363–374, 1993.

[DFKP06] A. Dessmark, P. Fraigniaud, D. Kowalski, and A. Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 46(1):69–96, 2006.

[DFP03] A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. In *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.

[EL75] P. Erdös and L. Lovasz. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal et al., editors, *Infinite and Finite Sets*, volume 11 of *Colloq. Math. Soc. Janos Bolyai*, pages 609–627. North-Holland, 1975.

[KM08] D. R. Kowalski and A. Malinowski. How to meet in anonymous network. *Theoretical Computer Science*, 399(1-2):141–156, 2008.

[Kou02] M. Koucký. Universal traversal sequences with backtracking. *Journal of Computer and System Sciences*, 65(4):717–726, 2002.

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

[Rei08] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

[Xin07] Qin Xin. Faster treasure hunt and better strongly universal exploration sequences. In *Proceedings of the 18th international conference on Algorithms and computation*, pages 549–560. Springer-Verlag, 2007.