Asynchronous Deterministic Rendezvous on the Line*

Grzegorz Stachowiak

Institute of Computer Science, University of Wrocław
Joliot-Curie 15, 50-383 Wrocław, Poland
gst@cs.uni.wroc.pl

Abstract. We study the rendezvous problem in the asynchronous setting in the graph of infinite line following the model introduced in [13]. We formulate general lemmas about deterministic rendezvous algorithms in this setting which characterize the algorithms in which the agents have the shortest routes. We also improve rendezvous algorithms in the infinite line which formulated in [13]. Two agents have distinct labels $L_{\min}, L_{\max}$ and $|L_{\min}| \leq |L_{\max}|$. When the initial distance D between the agents is known, our algorithm has cost $D|L_{\min}|^2$ which is an improvement in the constant. If the initial distance is unknown we give an algorithm of cost $O(D \log^2 D + D \log D|L_{\max}| + D|L_{\min}|^2 + |L_{\max}||L_{\min}| \log |L_{\min}|)$ which is an asymptotic improvement.

1 Introduction

Two mobile agents (robots) are initially located in a network being an undirected connected graph. Their task is to meet somewhere in the graph. This is known in the literature as the *rendezvous problem*. Papers studying rendezvous problem in the synchronous setting require agents to meet in a node. In the asynchronous setting the adversary can make the agents visit nodes at different times, so it is assumed that the meeting can occur either in a node or inside an edge. In this paper we follow the model introduced in [13].

In this model vertices are not labeled, but agents can distinguish edges adjacent to a node. We assume, that the ports of a node are locally labeled $1, 2, 3, \ldots, d$, where d is the degree of the node. An agent currently located in a node knows only the local labeling corresponding to this node. No coherence between these local labelings is assumed. When an agent traverses an edge, it knows both the label of the port by which it leaves and the port it enters a node and the degrees of the nodes. In general version of the problem we do not assume any knowledge of the topology of the graph, its size and the initial positions of the agents.

If agents are identical and execute the same program, then deterministic rendezvous is in in general impossible. Particularly in the graph of infinite line the adversary can make the agents move in the same direction at the same speed. Hence we assume, that the agents have unique identifiers, called labels, which are distinct binary strings, and every agent knows its own label. We also assume, that the agent knows nothing about the label of the other agent. The only initial input of a (deterministic) rendezvous algorithm

* Supported by MNiSW grants N206 001 31/0436, 2006–2008 and N N206 1723 33, 2007–2010.

M. Nielsen et al. (Eds.): SOFSEM 2009, LNCS 5404, pp. 497–508, 2009.

executed by an agent is the agent's label. During the execution of the algorithm the agent learns the local port number by which it enters a node. If L is a label, $|L|$ denotes its length. When there are two agents, by $L_{\min}$ we denote the shorter label and by $L_{\max}$ the longer label. The distance between the initial positions of the agents is D.

To analyze our algorithms we can consider an adversary. In general version of the problem, the adversary can choose the topology of yet unexplored part of the graph. We consider asynchronous algorithms, so when the agent situated in a node v_0 at a time t_0 has to traverse a segment $[v_0, v_1]$, the adversary chooses $t_1 > t_0$ and the continuous function $f : [t_0, t_1] \to [v_0, v_1]$ with $f(t_0) = v_0, f(t_1) = v_1$. This function defines the actual movement of an agent inside the segment $[v_0, v_1]$. Some authors assume that the agent can go back and forth inside a segment, but since this assumption does not give agents additional capability to avoid each other, we assume that f is monotone. The agent can move with an arbitrary speed. We say that in time $t \in [t_0, t_1]$ the agent is in point $f(t) \in [v_0, v_1]$.

As we already mentioned, the rendezvous occurs when both agents are at the same point at the same time. The *cost* of the rendezvous is defined as the worst-case number of edge traversals by both agents (the last partial edge traversal is counted as a complete one for both agents), where the worst-case is taken over all choices of labels and decisions of the adversary.

In [13] the choice of the starting times for the agents is also left to the adversary. We assume, that both agents start at the moment $t = 0$. Starting at different times can be described by the constant function f for the agent starting later in the time before it starts moving. Starting points of the agents are chosen by the adversary.

The rendezvous problem was introduced in [22]. The problem of the rendezvous on the line attracted very much attention [4,6,10,11,12,13,16,17,24]. Other considered scenarios were rendezvous on the plane [7,8]) and in graphs [1,3,13]. Most papers consider probabilistic scenario e.g. [1,2,9,10,11,17,18,24], where inputs or rendezvous algorithms are random. A natural extension of the rendezvous problem is that of gathering [15,18,21,23] where many agents should meet in one location.

Deterministic rendezvous of anonymous agents able to mark nodes in unlabeled graphs was considered in [20]. In [13,14,19,25] deterministic rendezvous in graphs with labeled agents was considered. In almost all these papers synchronous setting was assumed. The only exception was paper [13] in which rendezvous in graphs in the asynchronous setting was introduced. Asynchronous rendezvous under geometric scenario was studied in [15].

We can perform the rendezvous in an n-node tree in time $O(n)$ (see [13]). Every tree has either a central node or a central edge. Agents can first explore the tree by DFS and then meet in the central node or edge. The above method can be applied on a finite line, but is not feasible in the infinite line.

In paper [13] two algorithms for the infinite line are described. One assumes knowledge of D by both agents and has cost $O(D|L_{\min}|^2)$. The other does not assume the knowledge of D and has cost $O(D^3 + |L_{\max}|^3)$. In that paper also the rendezvous problem on a n-node ring was concerned. An optimal $O(n|L_{\min}|)$ rendezvous algorithm for known n and $O(n|L_{\max}|)$ algorithm for unknown n were found. Since these algorithms for the ring are almost optimal we concentrate on the graph of infinite line.

In section 2 we give general theorems stating that we can reduce the lengths of the routes of the agents when they contain subroutes called lightnings. The routes without lightnings are either ascending or unimodal. In section 3 we introduce a general class of skeleton algorithms used in the further sections. In section 4 we describe an algorithm for known D. It has the same asymptotic cost $O(D|L_{\min}|^2)$ as in [13], but the constant hidden behind the big O is eight times better than in [13]. In sections 5,6 and 7 we describe the algorithms for unknown D. The algorithm for known $|L|$ has cost $O(D|L|^2)$ which is the same as for known D. The algorithm for unknown $|L|$ (and D) has cost $O(D\log^2 D + D\log D|L_{\max}| + D|L_{\min}|^2 + |L_{\max}||L_{\min}|\log|L_{\min}|)$ which is asymptotically better than in [13].

2 General Algorithms

We consider an agent starting from vertex v of the infinite line. The agent assigns an *orientation* to the line choosing direction *right* according to the first edge it follows and *left* to the opposite direction. Then the agent tags the vertices of the line. Vertex v has tag 0, the vertex k steps to the right gets tag k and the vertex k steps to the left gets tag $-k$. This tagging can be then arbitrarily extended to all possible positions of the agent (i.e. to all "points" inside edges). Thus we view the tagging to be a continuous mapping from the infinite line to $\mathbb{R}$.

The agent moves along some *route* depending on agent's label and possibly on the initial distance D between the agents (if they know D). This route can be expressed by a sequence of integers $(x_1, x_2, x_3, \ldots)$ such that $x_{2i-1} > x_{2i} < x_{2i+1}$. The agent moving along such a route, first goes right to the vertex tagged x_1 (so $x_1 > 0$), then left to the vertex of tag x_2, then right to the vertex x_3 and so on. A *segment* $[a, b]$ of the line is the subgraph of the line consisting of all vertices between vertices tagged a and b including these vertices, and edges between them. The route $(x_1, x_2, x_3, \ldots)$ can be viewed as the sum of subsequent segments: $[0, x_1], [x_1, x_2], [x_2, x_3], \ldots$.

Now let us consider two agents starting from vertices u and v. Each agent has a label that determines its route. For arbitrary routes, not necessarily defined by a rendezvous algorithm, we have two possibilities. The first possibility is that both agents meet no matter of what pace they follow their routes. This possibility should be the case when we indeed use a rendezvous algorithm. In such a case we say that two routes *meet*. The other possibility is that there is a way to follow both routes, in which the agents do not meet. If this is possible, we say that two routes *miss*. We say that a route contains *lightning* $x_i, x_{i+1}, x_{i+2}, x_{i+3}$, if $x_{i+1}, x_{i+2} \in [x_i, x_{i+3}]$. We can transform a route containing a lightning into *reduced route* described by the sequence $(x_1, x_2, \ldots, x_i, x_{i+3}, \ldots)$ (just skipping x_{i+1}, x_{i+2}). This operation is called the *reduction of a lightning*.

Lemma 1. *Assume, two agents in the infinite line start from vertices u and v. Assume, the route V of the agent starting from v contains a lightning and V' is the reduced route. The route V and the route U starting from u miss if and only if V' and U miss.*

Proof. Figure 1 illustrates the Proof. □

This lemma can also be formulated in an equivalent way: the routes starting from u and v meet if and only if the reduced route starting from v and the route starting from u

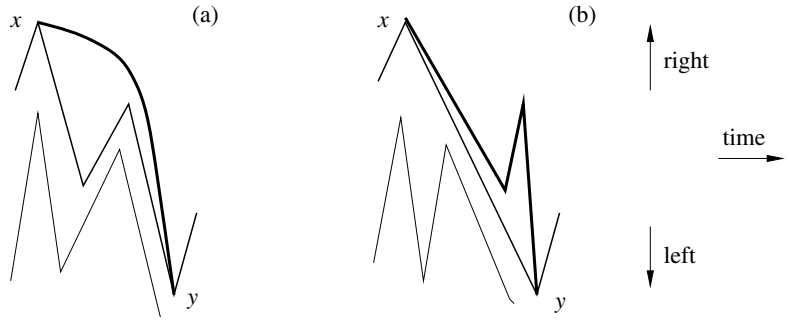

Fig. 1. Proof of Lemma 1. A lightning connecting x and y can be replaced with segment $[x, y]$ (a) or vice versa (b) and the two routes (thicker and thinner) all the time miss.

meet. This has some consequences for rendezvous algorithms on the infinite line. In a rendezvous algorithm we require, that any two routes assigned to different labels meet, no matter what orientations of the line are chosen by the agents.

Theorem 1. *Assume, that in a rendezvous algorithm a route containing a lightning is assigned to some label L. If we replace this route by the reduced route, then the algorithm remains a rendezvous algorithm.*

For an arbitrary rendezvous algorithm, we can reduce its cost reducing all lightnings in its routes. We should say how these maximally reduced routes look like. We call a route $(x_1, x_2, x_3, \ldots)$ *ascending* if sequences $x_1, x_3, x_5, \ldots$ and $0, -x_2, -x_4, \ldots$ are ascending. A sequence of numbers $a_1, a_2, \ldots, a_s$ is *unimodal* if and only if there is m such that $a_m = \max a_i$ and $a_{i-1} < a_i$ for $i \leq m$ and $a_i > a_{i+1}$ for $i > m$ (it is possible that $a_m = a_{m+1}$). We call a **route** $(x_1, x_2, x_3, \ldots)$ *unimodal* if the following conditions hold

- sequences $x_1, x_3, x_5, \ldots$ and $0, -x_2, -x_4, -x_6, \ldots$ are unimodal (and thus finite),
- if $x_{2m'-1} = \max\{x_{2i-1}\}$ and $-x_{2m''} = \max\{-x_{2i}\}$, then $|(2m'-1)-2m''| = 1$.

Theorem 2. *A route $(x_1, x_2, x_3, \ldots)$ does not contain lightnings if and only if it is either ascending or unimodal.*

Proof. An ascending route obviously does not contain lightnings. Assume, there is a lightning $x_i, x_{i+1}, x_{i+2}, x_{i+3}$ on a unimodal route for an odd i. We have $x_{2m'-1} \geq x_i \geq x_{i+2}$ and $-x_{2m''} \geq -x_{i+3} \geq -x_{i+1}$, which contradicts the condition $|(2m' - 1) - 2m''| = 1$. The case of even i is analogous.

Now we show, that if a route does not contain lightnings, then it is either ascending or unimodal. Let x_m be the first element of the sequence $(x_1, x_2, x_3, \ldots)$ for which $x_m \geq x_{m+2}$ for an odd m or $x_m \leq x_{m+2}$ for an even m. If such an index m does not exist, then the route is ascending.

Without the loss of generality we assume, that m is odd and $x_m \geq x_{m+2}$. For an even m the proof is symmetric. Since $x_m \geq x_{m+2}$, then $x_{m+1} < x_{m+3}$, otherwise $x_m, x_{m+1}, x_{m+2}, x_{m+3}$ form a lightning. Since $x_{m+1} < x_{m+3}$, then also $x_{m+2} > x_{m+4}$, otherwise $x_{m+1}, x_{m+2}, x_{m+3}, x_{m+4}$ form a lightning. Since $x_{m+2} > x_{m+4}$,

then also $x_{m+3} < x_{m+5}$, otherwise $x_{m+2}, x_{m+3}, x_{m+4}, x_{m+5}$ form a lightning. And so on until the end of the sequence. So the only local maxima of odd indexed sequence can be x_m and x_{m+2}, and the only local minimum of the even indexed sequence is x_{m+1}. Thus the route is unimodal. □

3 Skeleton Algorithms

We introduce a general family of *skeleton algorithms* on the infinite line. All rendezvous algorithms constructed in this paper are based on algorithms from this family. In the construction of such an algorithm we have a string of positive integers $S = (s_1, s_2, s_3, \ldots)$ called *skeleton*. The label $L \in \{0,1\}^*$ of an agent is transformed into another string[1] $L^* = (l_1, l_2, l_3, \ldots) \in \{-1,1\}^*$ of length equal to the length the skeleton (can be infinite).

The algorithm builds the route of an agent from the segments which it traverses. The first segment is $[0, l_1 \cdot s_1]$ (so we define $s_0 = 0$), and the i-th segment is $[l_{i-1} \cdot s_{i-1}, l_i \cdot s_i]$. If the skeleton $S = (s_1, s_2, \ldots, s_m)$ is finite, then the $m+1$-st segment is $[l_m \cdot s_m, 0]$ (thus $s_{m+1} = 0$). Because $l_1 \cdot s_1$ and x_1 are both positive, we should have $l_1 = 1$.

We are particularly interested in skeletons being either ascending ($s_{i-1} < s_i$ for all i) or unimodal sequences. The routes defined by such skeletons are either ascending or unimodal.

To analyze skeleton algorithms with ascending skeletons we introduce the function $W : \mathbb{N} \to \mathbb{R}$. The value of $W(x)$ is the maximum length of a route defined by a skeleton algorithm till the moment, the distance of an agent from vertex 0 becomes equal x for the first time. Obviously $W(x)$ is a strictly ascending function. We can formulate the following obvious fact.

Fact 1. $W(x) \leq x + 2 \sum_{s_i < x} s_i$.

When the skeleton S is ascending we define $\Delta s_i = s_i - s_{i-1}$. We can formulate the following lemma about skeleton algorithms with ascending skeletons.

Lemma 2. *Let the skeleton $S = (s_1, s_2, s_3, \ldots)$ be an ascending sequence. Let us have two agents starting from vertices u and v in distance D and having different labels L, K transformed into strings $L^* = (l_1, l_2, l_3, \ldots)$ and $K^* = (k_1, k_2, k_3, \ldots)$. There are four pairs $(\lambda, \kappa) \in \{-1,1\}^2$. If for any pair (λ, κ) there exists i such that $(l_i, k_i) = (\lambda, \kappa)$ and $\Delta s_i \geq D$, then these agents always meet. Let j be the smallest index such that for any pair (λ, κ) there is $i \leq j$ fulfilling $(l_i, k_i) = (\lambda, \kappa)$ and $\Delta s_i \geq D$. The total length of the routes of both agents till the meeting is not bigger than $2W(s_j)$.*

Proof. We consider the tagging of the infinite line according to the agent starting from v. We assume, u has tag D. The case of tag $-D$ is symmetric. From the premise of the lemma we conclude that there exists an index i for which the following conditions hold

- $s_i - s_{i-1} \geq D$,
- the i-th segment of the route starting in v ends in the vertex tagged s_i,
- the i-th segment of the route starting in u ends in the vertex tagged $D - s_i$.

[1] $\{-1,1\}^*$ denotes here the set of all finite or infinite strings of elements in $\{-1,1\}$.

Without loss of generality we can assume, that the agent starting from v is the first agent completing i-th segment of its route and it occurs in moment t. The agent starting from u is in moment t inside segment $[D-s_i, D+s_{i-1}]$. Thus in moment t the agent starting from u is in point of tag smaller or equal, than the point the agent starting from v is in. The starting point u has tag bigger, than the starting point v. Because of continuity of the routes the agents meet somewhere between the moments 0 and t. They meet until they complete j-th segments of their routes. This gives the estimation of the cost. □

We can also formulate a very similar lemma for unimodal skeletons.

Lemma 3. *Let the skeleton $S = (s_1, s_2, s_3, \ldots)$ be a unimodal sequence. Let us have two agents starting from vertices u and v in distance D and having different labels L, K transformed into strings $L^* = (l_1, l_2, l_3, \ldots)$ and $K^* = (k_1, k_2, k_3, \ldots)$. If for any pair $(\lambda, \kappa) \in \{-1, 1\}^2$ there exists i such that $(l_i, k_i) = (\lambda, \kappa)$ and also $s_i - s_{i-1} \geq D$ or $s_i - s_{i+1} \geq D$, then the routes of these agents meet.*

Proof. The case of $s_i - s_{i-1} \geq D$ is a repetition of the proof of the previous lemma.

An unimodal skeleton has to be finite. The case $s_i - s_{i+1} \geq D$ is a repetition of the proof of the previous lemma, if we reverse the time. We remind that the last segment returns the agent to its starting vertex. □

4 Known D

In this section we consider two agents initially situated in the infinite line in distance D. Unlike in the further sections this distance is known to both agents. In [13] an algorithm of cost $O(D|L_{\min}|^2)$ was presented. We can express the cost of a rendezvous algorithm more precisely, than in terms of the big O. We say, that the cost is at most $\sim f(D, L)$ when it is bounded from above by a function $g(D, L)$ such that[2] $g(D, L) \sim f(D, L)$.

The algorithm for known D described in [13] has cost $\sim 8D|L_{\min}|^2$ because of Fact 2.1 from [13]. In this section we concentrate on improving the leading constant in front of $D|L_{\min}|^2$. We present a rendezvous algorithm of cost $\sim D|L_{\min}|^2$. Our construction is based on skeleton algorithms with unimodal skeletons.

First we define $\mathcal{B}_k$ to be the set of all strings $(l_1, l_2, \cdots, l_{2k}) : l_i \in \{-1, 1\}$ whose exactly k elements l_i are 1's including $l_1 = 1$. We have $|\mathcal{B}_k| = \frac{1}{2}\binom{2k}{k}$.

Lemma 4. *If L^* and K^* are different sequences in $\mathcal{B}_k$, then for any pair $(\lambda, \kappa) \in \{-1, 1\}^2$ an index i exists such that $(l_i, k_i) = (\lambda, \kappa)$.*

Let $k = k(r')$ be the smallest k, for which $|\mathcal{B}_k| = \frac{1}{2}\binom{2k}{k} \geq 2 \cdot 2^{2r'}$. There exists a mapping $\varphi_{r'}$ assigning a unique element $L_1 \in \mathcal{B}_k$ to each label L of length $r = 2r'$ or $r = 2r' - 1$. The rendezvous algorithm is presented in Figure 2.

Fact 2. *The total length of any route for label of length r is at most $\sim \frac{1}{2}Dr^2$.*

[2] We follow the notation $f(D, L) \sim g(D, L)$ equivalent to $\forall \epsilon > 0 : |f(D, L)/g(D, L) - 1| < \epsilon$ for $D > D_\epsilon, L > L_\epsilon$.

1. For label L of length r let $r' = \lceil r/2 \rceil$ and $k = k(r')$. Let $k' = k'(r')$ be the smallest integer such that $2k' - \log_{1.1} k' - 1 \geq 2k$.
2. The unimodal skeleton $S = (D, 2D, 3D, \ldots, (k'-1)D, k'D, k'D, (k'-1)D, \ldots, D)$.
3. Let $I = \{i \in [1, 2k'] : \exists_{\iota \in \mathbb{Z}}\ i = \lceil 1.1^{\iota} \rceil\} \cup \{k'+1\}$. Let $J = \{1, 2, \ldots, 2k'\} \setminus I$.
4. We define the sequence $L^* = (l_1, l_2, \ldots, l_{2k'}) \in \{-1, 1\}^*$ assigned to L. Let $L_J = \varphi_{r'}(L)$. We put the string L_J as elements of l_j for $j \in J$ in unchanged order (if $|J| > 2k$, then $l_j = 1$ for a couple of last indices $j \in J$). If $i \in I$ then $l_i = -l_{i-1}$.

Fig. 2. Skeleton algorithm for known D

The analysis of this algorithm is split into two cases: the values r' for both labels equal or different. For equal values the rendezvous is assured by entries of L^* corresponding to the index set J. The entries in the set I are responsible for the rendezvous, if these values are different.

Lemma 5. *Any two routes corresponding to labels L_1, L_2 of lengths $r_1, r_2 : r_1 \leq r_2$ such that $\lceil r_1/2 \rceil = \lceil r_2/2 \rceil = r'$ meet. The cost in this case is at most $\sim Dr_1^2$.*

Proof. We can apply Lemmas 3 and 4 considering only indices in J, so the routes meet. The cost of the algorithm is at most the total length of both routes i.e. $\sim Dr_1^2$. □

Fact 3. *If $r_1' < r_2'$, then $k(r_1') < k(r_2')$.*

Lemma 6. *Any two routes corresponding to labels L_1, L_2 of lengths r_1, r_2 such that $r' = \lceil r_1/2 \rceil < \lceil r_2/2 \rceil = r''$ meet. The cost in this case is at most $\sim Dr_1^2$.*

Proof. If $r' < r''$, then $k' = k'(2r') < k'(2r'') = k''$. Let $S = (D, 2D, 3D, \ldots, (k''-1)D, k''D, k''D, (k''-1)D, \ldots, D)$ be the skeleton for L_2. The indices $\{1, 2, \ldots, 2k''\}$ are divided into sets I and J. We consider the smallest $i \in I$ such that $k'+1 < i$. Such an index i exists, because $k'+1 < k''+1 \in I$. Note, that $i \leq \lceil 1.1(k'+1) \rceil$.

We consider the tagging of the infinite line by the agent of label L_2. Without loss of generality we can assume, that the agent labeled L_1 starts from vertex D.

The agent labeled L_2 has the vertex of tag $(i-1)D$ in the i-th segment of its route, because $l_i \neq l_{i-1}$. Assume, it gets to this vertex in moment t. The agent of label L_1 is in moment t in the segment $[D - k'D, D + k'D]$ and $k' \leq (i-2)$. So in moment t the agent of label L_1 is in point of smaller tag, than the agent labeled L_2. The tag of the starting point of the agent labeled L_1 is bigger than for the agent labeled L_2. Due to continuity of the routes they have to meet in some moment earlier than t.

The length of the route of agent of label L_2 until the end of segment i is at most $\sim 1.1^2 \frac{1}{4} Dr_1^2$. The total length of the route of the agent labeled L_1 is at most $\sim (\frac{1}{2}) Dr_1^2$. Thus the cost of the algorithm is at most $\sim (\frac{1}{2} + 1.1^2 \frac{1}{4}) Dr_1^2 < Dr_1^2$. □

Finally we summarize Lemmas 5 and 6 as the main theorem.

Theorem 3. *The algorithm from this section is a rendezvous algorithm on the infinite line for known distance D of cost at most $\sim D|L_{\min}|^2$.*

5 Unknown D, Fixed $|L|$

Now we construct a rendezvous algorithm in the case when $r = |L|$ is fixed and the agents do not know the distance D. This algorithm has cost $O(D|L|^2)$. In this section we also try to minimize the leading constant in front of $D|L|^2$.

Our algorithm is a skeleton algorithm with an ascending infinite skeleton and is presented in figure 3. In this algorithm we have a parameter a which determines the leading constant in the cost and is chosen later on in this section.

Lemma 7. *Let us consider the routes of two agents labeled L, K. An index j exists, such that for any $(\lambda, \kappa) \in \{-1, 1\}^2$ there is $i : i \le j$ fulfilling $(l_i, k_i) = (\lambda, \kappa)$ and $\Delta s_i \ge D$. If we choose the smallest such an index j, then $s_j \le Dab/(b-1)$. The routes meet and the rendezvous cost is at most $2W(Dab/(b-1))$.*

Proof. Let j be the smallest integer for which $\Delta s_{j-2k+1} > D$. For all $2k$ values $i \in (j-2k, j]$ we have $\Delta s_i \ge \Delta s_{j-2k+1} \ge D$. Because of Lemma 4 for any $(\lambda, \kappa) \in \{-1, 1\}^2$ there is $i \in (j-2k, j]$ such that $(l_i, k_i) = (\lambda, \kappa)$. Since Δs_i is a geometric progression, $\Delta s_j \le Da$ implies $s_j = \sum_{i \le j} \Delta s_i \le Dab/(b-1)$. Due to Lemma 2, the routes meet till any agent gets to the distance $Dab/(b-1)$ from its starting vertex. □

Now we should estimate $W(y)$ for an arbitrary y. Then we estimate $b - 1$.

Lemma 8. *Function $W(y)$ is upper bounded by a function $F(y, a, k) \sim \frac{4yk}{\ln a}$.*

Proof. We have $W(y) < \sum_{i=0}^{\infty} \frac{2y}{b^i} = 4yk \sum_{i=0}^{\infty} \frac{1}{a^{i/2k}} \frac{1}{2k} \sim 4yk \int_0^{\infty} \frac{dx}{a^x} = \frac{4yk}{\ln a}$. □

Fact 4. $b - 1 = a^{1/2k} - 1 = e^{\ln(a)/2k} - 1 \sim 1 + \frac{\ln a}{2k} - 1 = \frac{\ln a}{2k}$.

Finally we choose the parameter a minimizing the upper bound on the rendezvous cost $2W(Dab/(b-1))$. We have $2W\left(\frac{Dab}{b-1}\right) < \frac{16k^2 Da}{\ln^2 a}(1 + o(1))$. In order to minimize the leading constant we minimize $g(a) = \frac{a}{\ln^2 a}$. It is easy to compute, that function $g(a)$ attains its minimum for $a_m = e^2$ and $g(a_m) = e^2/4$.

Theorem 4. *The algorithm described in this section for $a = a_m$ is a rendezvous algorithm for unknown D and fixed $|L|$ of cost at most $\sim e^2 D|L|^2$.*

1. The skeleton is the sequence $s_i = \lceil b^i \rceil$, where $b = a^{1/(2k)}$.
2. We choose the minimal integer k such that $|\mathcal{B}_k| \ge 2^r$. There exists a mapping φ assigning a unique $L'' \in \mathcal{B}_k$ to each $L \in \{0, 1\}^r$.
3. The periodic sequence L^* is constructed by repeating $L'' = \varphi(L)$ infinite number of times.

Fig. 3. Skeleton algorithm for unknown D and fixed $|L|$

6 Superposition of Skeleton Algorithms

In the next section we define a rendezvous algorithm in the case when both D and $|L|$ can vary. Our solution requires combining two skeleton algorithms. In this section we define the *superposition* of two algorithms. From now on we analyze the algorithms only in terms of the big O no longer taking care about the leading constants.

Let us have two skeleton algorithms A_U, A_V with ascending skeletons U and V. They do not need to be rendezvous algorithms i.e. do not need to assure the rendezvous. We define a skeleton algorithm $A_U \circ A_V$ being their *superposition* in figure 4. Assume, that in A_U (A_V) the rendezvous happens till any agent gets to the distance x from its starting vertex. The algorithm $A_U \circ A_V$ assures the rendezvous, till this distance for any agent is cx for some constant c. Thus in the sense of this distance $A_U \circ A_V$ is at most c times worse, than the algorithm A_U (A_V).

Lemma 9. *If $u'_i > 8$, then $u_i \leq u'_i < \frac{19}{3}u_i$. If $v'_i > 8$, then $v_i \leq v'_i < \frac{19}{3}v_i$.*

We denote by $W_U(y), W_V(y)$ and $W(y)$ the maximum cost of the route till the agent's distance from its starting vertex is equal y, when the route is generated by A_U, A_V and $A_U \circ A_V$ respectively.

Lemma 10. *Let $A_U, A_V, A_{V'}$ be skeleton algorithms. Let the labels L, K of two agents be transformed in algorithm A_U into strings $P^* = (p_1, p_2, p_3, \ldots)$ and $R^* = (r_1, r_2, r_3, \ldots)$ respectively. Let j be the smallest index, such that for any $(\lambda, \kappa) \in \{-1, 1\}^2$ there is an integer $i : i \leq j$ fulfilling $(p_i, r_i) = (\lambda, \kappa)$ and $\Delta u_i \geq D$. The route generated by algorithm $A_U \circ A_V$ for L and the route generated algorithm $A_U \circ A_{V'}$ for K meet, till any agent goes to the distance u'_j from its starting vertex. The same holds, when the routes are generated by $A_V \circ A_U$ and $A_{V'} \circ A_U$ respectively.*

Proof. We have $\Delta s_{i'} \geq \Delta u_i$ for $s_{i'} = u'_i$. Thus the premise of Lemma 2 is true and the rendezvous occurs till any agent gets to the distance u'_j from its starting vertex. □

1. Let $U = (u_1, u_2, \ldots)$ and $V = (v_1, v_2, \ldots)$ be the skeletons of A_U and A_V respectively. We define $X_0 = (0, 2]$ and $X_k = (2^{2k-1}, 2^{2k+1}]$ for $k > 0$.
2. We transform the skeleton $U = (u_1, u_2, \ldots)$ into the sequence $U' = (u'_1, u'_2, \ldots)$:
   $i \leftarrow 1, k \leftarrow 0$ and $(a, b] \leftarrow X_k$
   while i does not exceed length(U)
   if $\Delta u_i \leq b - a$ **then** $u'_i \leftarrow a + \Delta u_i, a \leftarrow u'_i, i \leftarrow i + 1$
   else $k \leftarrow k + 2, (a, b] \leftarrow X_k$.
3. We transform skeleton $V = (v_1, v_2, \ldots)$ into sequence $V' = (v'_1, v'_2, \ldots)$ using almost the same subroutine as transforming U into U'. The only difference is initializing $k \leftarrow 1$.
4. The ascending skeleton S of $A_U \circ A_V$ is the result of merging sequences U', V'.
5. Let $P^* = (p_1, p_2, \ldots)$ and $Q^* = (q_1, q_2, \ldots)$ be the strings belonging to $\{-1, 1\}^*$ that are assigned to L in algorithms A_U and A_V respectively. In the j-th segment of the route defined by $A_U \circ A_V$, the agent goes to the vertex tagged $p_i \cdot u'_i$ if $s_j = u'_i$, or to the vertex $q_i \cdot v'_i$ when $s_j = v'_i$.

Fig. 4. Superposition $A_U \circ A_V$ of skeleton algorithms with ascending skeletons

1. Let $l = l(L)$ be such that $2^{l-1} \leq |L| < 2^l$. Let L_1 be the concatenation: $L0^{2^l-1-|L|}1$. Let A_U be the algorithm described in Figure 3 for label L_1.
2. We divide the set $\mathbb{N}$ of all indices into disjoint sets of subsequent indices $I_1, I_2, I_3, \ldots$ such that the set I_k has $2k$ elements. So: $I_1 = \{1, 2\}$,$I_2 = \{3, 4, 5, 6\}$,$I_3 = \{7, 8, 9, 10, 11, 12\}$,....
3. For each $i \in I_k$ let $\Delta v_i = 2^k$. This defines the skeleton V.
4. Let r_k be the largest integer, such that $2^{r_k} \leq \frac{1}{2}\binom{2k}{k}$. There is a mapping φ_k assigning each nonnegative integer smaller than 2^{r_k} a unique string from $\mathcal{B}_k$.
5. Let $l_k = \min\{l, 2^{r_k} - 1\}$, $Q_k = \varphi_k(l_k)$. Let Q^* be the concatenation $Q_1Q_2Q_3Q_4 \ldots$.
6. The algorithm A_V has skeleton V for which the string $Q^* \in \{-1, 1\}^*$ is applied.
7. The rendezvous algorithm is the superposition $A_U \circ A_V$

Fig. 5. Algorithm for unknown D and $|L|$

Lemma 11. $W(y) \leq \frac{19}{3}W_U(y) + \frac{19}{3}W_V(y) + O(y)$.

Proof. Let $j = \min\{i : s_i \geq y\}$ and $s_j \in X_k$. We can divide the sections of the route generated by algorithm $A_U \circ A_V$ into four sets.

- Sections for $s_i \leq 8$. Altogether they have length not bigger than 64.
- Sections for $s_{i-1} \notin X_l, s_i \in X_l$. Their length is at most $2\sum_{l=0}^{k} 2^{2l+1} = O(y)$.
- Sections whose length is Δs_i. Their total length is at most $\sum_{i=0}^{j} \Delta s_i = O(y)$.
- Sections $[-u'_{i-1}, u'_i]$ or $[u'_{i-1}, -u'_i]$ corresponding to sections $[-u_{i-1}, u_i]$ or $[u_{i-1}, -u_i]$ in algorithm A_U. Their total length is not bigger than $\frac{19}{3}W_U(y)$.
- Sections like in the previous set but for algorithm A_V. They have total length not bigger than $\frac{19}{3}W_V(y)$. □

7 Unknown D and $|L|$

In this section we present an efficient rendezvous algorithm on the infinite line when neither $|L|$ nor D are known. Our algorithm is a superposition $A_U \circ A_V$ of two skeleton algorithms A_U and A_V and is described in figure 5. Algorithm A_U is an efficient rendezvous method when the $|L_{\min}|$ and $|L_{\max}|$ are the same or close, otherwise it does not assure, that the agents meet. Algorithm A_V assures the rendezvous only when $|L_{\min}|$ and $|L_{\max}|$ differ substantially.

Lemma 12. *If $l(L) = l(K)$, then the routes generated by $A_U \circ A_V$ meet till any agent gets to the distance $O(D|L|)$ from its vertex 0.*

Proof. Because of the the analysis of the algorithm from section 5 (Lemma 7 and Fact 4) the routes in A_U meet till any agent gets to the distance $O(Dab/(b-1)) = O(D|L|)$ from its vertex 0. Due to Lemma 10 the same holds for $A_U \circ A_V$. □

Lemma 13. *If $l(L) < l(K)$, then the routes generated by $A_U \circ A_V$ meet till any of the agents goes to the distance $O\left(|L| \log |L| + D \log D\right)$ from its starting vertex.*

Proof. Let the strings $Q, R \in \{-1,1\}^*$ be assigned in A_V to the agent labels L, K respectively. Let k be the smallest positive integer for which $l(L) < 2^{r_k} - 1$. We have $2k \sim r_k \sim \log|L|$. Let k' be the smallest integer $k' \geq k$ such that $2^{k'} \geq D$. Note, that $k' = O(\max\{\log|L|, \log D\})$. Let $j = \sum_{i=1}^{k'} 2i$ be the last index in $I_{k'}$. For each $(\lambda, \kappa) \in \{-1,1\}^2$ there exists $i \leq j$, that $(q_i, r_i) = (\lambda, \kappa)$ and $\Delta v_i = 2^{k'} \geq D$. Because of Lemma 10 the rendezvous occurs till the distance of any agent from its vertex 0 is $O(v_j)$. Note that $v_j = \sum_{i=1}^{k'} 2i2^i = O(k'2^{k'})$. We can estimate that

$$O(v_j) = O(k'2^{k'}) = O\left(|L|\log|L| + D\log D\right). \qquad \square$$

Lemma 14. $W_U(y) = O(y|L|), W_V(y) = O(y\log y)$

Theorem 5. *The algorithm $A_U \circ A_V$ is a rendezvous algorithm. The rendezvous cost is $O(D\log^2 D + D\log D|L_{\max}| + D|L_{\min}|^2 + |L_{\max}||L_{\min}|\log|L_{\min}|)$.*

Proof. Assume, we apply the algorithm for two labels $L, K : |L| \leq |K|$. The fact that $A_U \circ A_V$ is a rendezvous algorithm follows from Lemmas 12 and 13. The cost in case $l(L) = l(K)$ can be estimated using Lemmas 11,12 and 14. The agents meet till their distances from their starting vertices are $y = O(D|L|)$. So

$$2W(y) \leq \frac{38}{3}W_U(y) + \frac{38}{3}W_V(y) + O(y) = O(D|L|^2 + D|L|\log D)$$

The cost in case $l(L) < l(K)$ is estimated by Lemmas 11,13,14. The rendezvous occurs till the distance between any agent and its staring vertex is $y = O(|L|\log|L| + D\log D)$. Denote by W_L, W_K the values of W for algorithms defined by labels L, K.

$$\begin{aligned} W_L(y) + W_K(y) &\leq \frac{38}{3}W_{U(K)}(y) + \frac{38}{3}W_V(y) + O(y) \\ &= O(|K||L|\log|L| + |K|D\log D + D\log^2 D) \end{aligned}$$

The maximum cost is the maximum of the bounds on the costs $2W(y)$ and $W_L(y) + W_K(y)$ i.e. $O(D\log^2 D + |K|D\log D + D|L|^2 + |K||L|\log|L|)$. $\square$

8 Conclusions and Open Problems

It is unclear what the lower bound on rendezvous cost is, even in the case when D is known. The author supposes that it is $\Omega(D|L_{\min}|^2)$. Another question is the lower bound for unknown D. We can construct an algorithm very similar to that in [13] in which the cost is $O(D^3 + |L_{\min}|^3)$. The cost of our algorithm from this paper depends on $|L_{\max}|$. Can we get an efficient algorithm of cost not depending on $|L_{\max}|$?

References

1. Alpern, S.: The rendezvous search problem. SIAM J. Control Optim. 33, 673–683 (1995)
2. Alpern, S.: Rendezvous search on labelled networks. Naval Res. Logist. 49, 256–274 (2002)
3. Alpern, J., Baston, V., Essegaier, S.: Rendezvous search on a graph. J. Appl. Probab. 36, 223–231 (1999)

4. Alpern, S., Gal, S.: Rendezvous search on the line with distinguishable players. SIAM J. Control Optim. 33, 1270–1276 (1995)
5. Alpern, S., Gal, S.: The theory of search games and rendezvous. In: International Series in Operations Research and Management Science, vol. 55. Kluwer Academic Publishers, Dordrecht (2002)
6. Anderson, E., Essegaier, S.: Rendezvous search on the line with indistinguishable players. SIAM J. Control Optim. 33, 1637–1642 (1995)
7. Anderson, E., Fekete, S.: Asymmetric rendezvous on the plane. In: Proc. Symp. on Computational Geometry 1998, pp. 365–373 (1998)
8. Anderson, E., Fekete, S.: Two-dimensional rendezvous search. Oper. Res. 49, 107–118 (2001)
9. Anderson, E., Weber, R.: The rendezvous problem on discrete locations. J. Appl. Probab. 28, 839–851 (1990)
10. Baston, V.J.: Two rendezvous search problems on the line. Naval Res. Logist. 46, 335–340 (1999)
11. Baston, V., Gal, S.: Rendezvous on the line when the players' initial distance is given by an unknown probability distribution. SIAM J. Control Optim. 36, 1880–1889 (1998)
12. Baston, V., Gal, S.: Rendezvous search when marks are left at the starting points. Naval Res. Logist. 48, 722–731 (2001)
13. De Marco, G., Gargano, L., Kranakis, E., Krizanc, D., Pelc, A., Vaccaro, U.: Asynchronous deterministic rendezvous in graphs. Theor. Comput. Sci. 355, 315–326 (2006)
14. Dessmark, A., Fraigniaud, P., Pelc, A.: Deterministic rendezvous in graphs. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 184–195. Springer, Heidelberg (2003)
15. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of asynchronous oblivious robots with limited visibility. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 247–258. Springer, Heidelberg (2001)
16. Gal, S.: Rendezvous search on the line. Oper. Res. 47, 974–976 (1999)
17. Han, Q., Du, D., Vera, J., Zuluaga, L.F.: Improved bounds for the symmetric rendezvous value on the line. In: SODA 2007, pp. 69–78 (2007)
18. Israeli, A., Jalfon, M.: Token management schemes and random walks yield self stabilizing mutual exclusion. In: Proc. PODC 1990, pp. 119–131 (1990)
19. Kowalski, D.R., Pelc, A.: Polynomial deterministic rendezvous in arbitrary graphs. In: Fleischer, R., Trippen, G. (eds.) ISAAC 2004. LNCS, vol. 3341, pp. 644–656. Springer, Heidelberg (2004)
20. Kranakis, E., Krizanc, D., Santoro, N., Sawchuk, C.: Mobile agent rendezvous in a ring. In: Proc. ICDCS 2003, Providence, RI, USA, pp. 592–599 (2003)
21. Lim, W., Alpern, S.: Minimax rendezvous on the line. SIAM J. Control Optim. 34, 1650–1665 (1996)
22. Schelling, T.: The Strategy of Conflict. Oxford University Press, Oxford (1960)
23. Thomas, L.: Finding your kids when they are lost. J. Oper. Res. Soc. 43, 637–639 (1992)
24. Uthaisombut, P.: Symmetric rendezvous search on the line using moving patterns with different lengths. Department of Computer Science, University of Pittsburgh (2006)
25. Yu, X., Yung, M.: Agent rendezvous: a dynamic symmetry-breaking problem. In: Meyer auf der Heide, F., Monien, B. (eds.) ICALP 1996. LNCS, vol. 1099, pp. 610–621. Springer, Heidelberg (1996)