Rendezvous of Mobile Agents in Directed Graphs

Jérémie Chalopin[1], Shantanu Das[1], and Peter Widmayer[2]

[1] LIF, CNRS & Aix-Marseille University, France
jeremie.chalopin@lif.univ-mrs.fr, shantanu.das@acm.org
[2] Institute of Theoretical Computer Science, ETH Zürich, Switzerland
widmayer@inf.ethz.ch

Abstract. We study the problem of gathering at the same location two mobile agents that are dispersed in an unknown and unlabeled environment. This problem called Rendezvous, is a fundamental task in distributed coordination among autonomous entities. Most previous studies on the subject model the environment as an undirected graph and the solution techniques rely heavily on the fact that an agent can backtrack on any edge it traverses. However, such an assumption may not hold for certain scenarios, for instance a road network containing one-way streets. Thus, we consider the case of strongly connected directed graphs and present the first deterministic solution for rendezvous of two anonymous (identical) agents moving in such a digraph. Our algorithm achieves rendezvous with detection for any solvable instance of the problem, without any prior knowledge about the digraph, not even its size.

Keywords: Distributed Algorithm, Directed Graph, Leader Election, Rendezvous, Anonymous Networks, Mobile Agents, Graph Exploration.

1 Introduction

One of the fundamental problems in distributed coordination is the task of gathering together two autonomous entities that are dispersed in a unknown environment. The problem, called *rendezvous* problem, has been studied both for robots moving in a terrain or software agents moving in a network. In the former case, the environment is a bounded (or unbounded) region of the infinite two-dimensional plane. While in the latter case, the environment is modeled as a graph where the two agents are initially located at distinct nodes of the graph. When the two dispersed entities can not communicate from a distance, solving rendezvous is essential for an exchange of information or for achieving even the simplest form of coordination between the mobile entities. The rendezvous problem belongs to the class of symmetry-breaking problems (e.g. leader election is another such problem) that are central to study of computability in distributed systems. The importance of the problem is evident from the large volume of literature [10, 11, 14–17, 19] dedicated to solving the problem under various conditions and restrictions.

N.A. Lynch and A.A. Shvartsman (Eds.): DISC 2010, LNCS 6343, pp. 282–296, 2010.

For rendezvous in graphs, almost all previous results were restricted to undirected graphs. However, in many scenarios, it is natural to model the environment as a directed graph. For instance, in a communication network such as the internet, it may be possible to communicate in only one direction along some of the channels. Another example of a directed graph is a road network containing some one way streets.

To the best of our knowledge, there are no known results on rendezvous of agents in arbitrary and unknown directed graphs. Moreover, the solutions for the undirected graphs do not carry over to the case of digraphs, as they heavily rely on the ability of the agents to backtrack on a traveled path.

In this paper, we show how two identical agents can rendezvous in an unknown digraph. The agents are allowed to move along the arcs of the digraph from the tail end to head end. We assume the digraph to be strongly connected so that any node of the digraph can be reached. An agent located at any node v of the digraph can choose one of the several arcs outgoing from v and this choice is made by a deterministic algorithm which specifies the moves of the agents. The idea is to design an algorithm that when executed by each agent ensures that after a finite number of moves, the agents terminate in exactly one node of the digraph. In the most general setting, the nodes of the graph are anonymous (i.e. the agents can not distinguish between nodes of the same degree) and the agents themselves do not have any unique identifiers either. In this setting, rendezvous can not be solved in arbitrary graphs, using deterministic algorithms. We thus focus on the problem of "rendezvous with detect" where the agents rendezvous if and only if it is deterministically possible for the given instance, and otherwise stop and report that problem is not solvable. Solving rendezvous usually requires the agents to completely explore the digraph. Exploration of unknown and unlabeled graphs (or digraphs) is difficult unless the agents are allowed to mark the nodes. As in several previous papers on the topic [3, 4, 9, 11, 13], we allow the agents to mark nodes of the digraphs by writing on whiteboards available at each node.

Related Works: The problems of rendezvous and leader election has been extensively studied from the point of view of computability in unknown and unlabeled graphs, starting from the work of Angluin [2]. Many researchers have focussed on characterizing the conditions under which distributed coordination problems (such as leader election) can be solved in the absence of unique identifiers for the participating entities. Characterizations of the solvable instances for leader election in *message passing systems* have been provided by Boldi *et al.* [7] and by Yamashita and Kameda [18] among others. The rendezvous problem which falls under the same class of symmetry breaking problems, has been solved under various different assumptions such as distinct labels for the agents, sense of direction information, or under a synchronous setting (e.g. [4, 10, 16, 17, 19]). The problem has been studied for specific topologies such as rings [11], tori [17], trees [15] as well as in the most general case of an unlabeled terrain [14]. The idea of solving rendezvous by marking the starting locations with tokens was first proposed by Baston and Gal [5].

Most of the above results are for undirected graphs. Boldi and Vigna [7, 8] considered directed graphs and they introduced the concept of *fibrations* to illustrate certain properties of digraphs. For instance, they showed that deterministic leader election is solvable only in *fibration-free* digraphs.

For directed graphs, even exploration using deterministic algorithms is relatively difficult compared to undirected graphs. The problem of exploring and constructing a map of an unknown strongly connected digraph has been studied previously, from the point of view of minimizing the number of arc traversals (also called the "moves" complexity). For exploration of node-labelled digraphs, the best known algorithm [1] has a cost of $O(d^{logd} * m)$ moves for a digraph G having deficiency d (i.e. d arcs are needed to make G Eulerian). Another optimization criteria is minimizing the amount of memory required by an agent to explore a directed graph (See e.g. [13])

When the nodes of the digraph are not labelled, it is necessary to mark the nodes so that they can be recognized on subsequent visits. Exploration of unlabeled and unknown digraphs using pebbles to mark vertices, has been studied by Bender et al. [6]. A related problem of searching for a black hole has been recently studied for directed graphs (see [9]).

Our Results: We study the rendezvous problem for two identical agents starting from arbitrary locations in strongly connected directed graphs in the absence of unique identifiers. We present an algorithm for solving *Rendezvous with Detect* in any strongly connected digraph. Our results give a characterization of the solvable instances for rendezvous in this setting. Thus, our result generalizes previous results on rendezvous in undirected graphs to the more general case of directed graphs. The algorithm considered in this paper requires $O(m.n)$ moves for graphs of n nodes and m edges, whereas for undirected graphs, the same problem can be solved in $O(m)$ moves. Our algorithm is universal and works for digraphs of arbitrary size and topology (i.e. no prior knowledge about the digraph is required).

2 Definitions and Properties

2.1 Our Model

The environment where the agents are operating is represented by a strongly connected directed graph $G = (V, A)$ where V is the set of nodes and A is the set of arcs of G. We denote by n and m, the number of nodes and the number of arcs of G, respectively. There are two identical mobile agents located in two distinct nodes of the digraph G. The agents execute the same algorithm and start from the same initial state (though not necessarily at the same time). The agents do not have any prior knowledge of the graph G, not even n, the size of G. Every action performed by an agent takes a finite but otherwise unpredictable amount of time. The node from where an agent a starts the algorithm (i.e. the initial location) is called the *homebase* of agent a, denoted by $h(a)$. The objective of the agents is to solve rendezvous i.e. meet at any unspecified node

of the digraph G. We denote an instance of the rendezvous problem by (G, χ_p) where $\chi_p : V \to \{0, 1\}$ is a node-labeling of G such that $\chi_p(v) = 1$ if there exists an agent a such that $h(a) = v$, and $\chi_p(v) = 0$ otherwise.

The agents can traverse the arcs of G only from the tail end to the head end (but not the other way). Since G is strongly connected each node has at least one incoming and at least one outgoing arc. We denote by $d_{in}(v)$ (resp. $d_{out}(v)$) the number of incoming (resp. outgoing) arcs at v. The arcs going out from a node v are locally oriented i.e. they are labelled as $1, 2, \ldots, d_{out}(v)$. Similarly, the arcs incoming at a node v are labelled as $1, 2, \ldots, d_{in}(v)$ and an agent arriving at node v by an arc e, knows the label of e at v. Note that each arc $e = (u, v)$ of G has two labels, one at the tail end u and the other at the head-end v. The arc labeling of G (sometimes called the port numbering) is specified by $\lambda : A(G) \to \mathbb{N}^2$. We call this the Duplex arc-labeling model. Note that this model corresponds to the *Port-to-Port*(PP) model in the message passing system, as defined by Yamashita and Kameda [18].

We will also consider the more general case of Simplex arc-labeling where only outgoing arcs at a node v are labelled, but not the incoming arcs. In other words, an agent arriving at node v does not know through which arc it arrived. In this case, the arc labeling of G is specified by $\lambda : A(G) \to \mathbb{N}$.This model corresponds to the *Port-to-Mailbox*(PM) model in the message passing system of Yamashita and Kameda [18]. The difference between the two models and the effect it has on the computations, is further discussed in Section 4.

Each node $v \in G$ has a whiteboard and any agent visiting node v can read or write to that whiteboard. We use label(v) to denote the contents of the whiteboard of the node v. Initially all whiteboards are empty, so label$(v)= \phi$, for each $v \in V(G)$. Whenever the two agents are in the same node, they see each other and stop.

2.2 Coverings of Digraphs

For the definitions given below, we consider multigraphs, i.e. digraphs that possibly contain parallel arcs and self loops. Such a multigraph is denoted by $D = (V(D), A(D), s, t)$ where $V(D)$ is a set of vertices, $A(D)$ is a set of arcs, and s and t are two functions that assign to each arc two elements of $V(D)$: a source and a target. A path between two vertices u and v in D is a sequence of arcs $a_1, a_2, \ldots a_p$ such that $s(a_1) = u, \forall 1 \leq i \leq p-1, t(a_i) = s(a_{i+1})$ and $t(a_p) = v$.

A *homomorphism* γ between the multigraph D and the multigraph D' is a mapping γ: $V(D) \cup A(D) \to V(D') \cup A(D')$ such that for each arc $a \in A(D)$, $\gamma(s(a)) = s(\gamma(a))$ and $\gamma(t(a)) = t(\gamma(a))$. An homomorphism γ is an *isomorphism* if γ is bijective. We now define the notion of graph coverings, borrowing the terminology of Boldi and Vigna[8].

Definition 1. *A* covering projection *is a homomorphism φ from D to D' satisfying the following: (i) For each arc a' of $A(D')$ and for each node v of $V(D)$ such that $\varphi(v) = v' = t(a')$ there exists a unique arc a in $A(D)$ such that $t(a) = v$*

and $\varphi(a) = a'$. (ii) For each arc a' of $A(D')$ and for each node v of $V(D)$ such that $\varphi(v) = v' = s(a')$ there exists a unique arc a in $A(D)$ such that $s(a) = v$ and $\varphi(a) = a'$.

If the homomorphism φ satisfies only property (i) above, it is called a *fibration*, whereas if it satisfies only property (ii), it is called an *opfibration*. If a covering projection $\varphi : D \to D'$ exists, D is said to be a *covering* of D' via φ and D' is called the base of φ. The notions of coverings extend to labelled digraphs in an obvious way: the homomorphisms must preserve the labeling. Throughout the paper we will consider digraphs where the vertices and arcs are labelled with labels from a recursive label set L (The labeling of D is denoted by μ_D). We will consider homomorphisms which preserve these labelings.

Definition 2. *A labeled digraph (D, μ_D) is said to be* covering-minimal *if there does not exist any labeled multigraph $(D', \mu_{D'})$, where D' is not isomorphic to D, such that (D, μ_D) is a covering of $(D', \mu_{D'})$ via a label-preserving covering projection.*

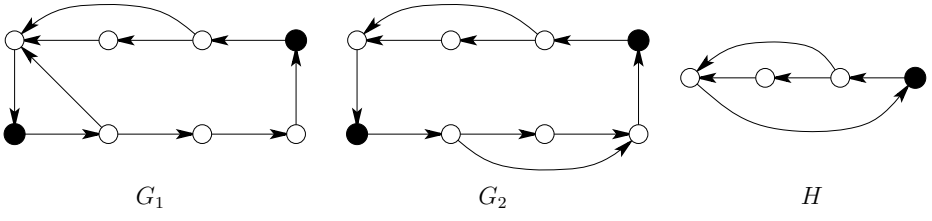

Fig. 1. There exists an opfibration from digraph G_1 to H and a covering projection from digraph G_2 to H. So, G_2 covers H, while G_1 is covering minimal. (The arc labeling is not shown in the figure and the node-labeling is illustrated by colors black and white.)

2.3 Impossibility Result

We now present a characterization of labelled digraphs where it is possible to solve rendezvous of mobile agents. First let us consider some known computability results for digraphs.

Lemma 1 ([8]). *Given a digraph $G(V, A)$ and an arbitrary (non-injective) labeling μ_G on G, if there exists a label-preserving covering projection from the labelled digraph (G, μ_G) to a labeled multigraph (H, μ_H), such that $|V(H)| < |V(G)|$ then it is impossible to solve leader election using message-passing in (G, μ_G).*

Lemma 2. *For a directed graph G, a port numbering λ on the arcs of G, and a bicoloring $\chi_p : V \to \{0, 1\}$ of the nodes of G, if there exists an algorithm for solving rendezvous of mobile agents that start from nodes of same color in G, then there is an algorithm for solving leader election by message passing in (G, λ, χ_p).*

The above result follows trivially from the fact that a mobile agent algorithm can be simulated by a message-passing algorithm where the messages are simply an encoding of the agent's states and their algorithm (see [3] for details). Once rendezvous is solved on the digraph G, there is exactly one node that contains all the agents and this node can be the designated leader.

Based on the above results, it follows that rendezvous is not possible in a labelled digraph (G, λ, χ_p) which is not covering minimal. In the next section we present an algorithm the solves rendezvous for all strongly connected digraphs that are covering minimal. Thus, we have the following result:

Theorem 1. *Given any strongly connected directed graph G, a port numbering λ on the arcs of G, and node labeling χ_p that denotes the initial placement of two identical agents in G, Rendezvous is solvable if and only if (G, λ, χ_p) is covering minimal.*

The above result holds irrespective of whether the port numbering λ is a simplex or duplex arc-labeling. In the next section, we present an algorithm for rendezvous assuming duplex arc-labeling (i.e. when both incoming and outgoing arcs are locally oriented). In section 4, we will show how to extend the algorithm for the case of simplex arc-labeling.

3 Rendezvous Algorithm

A usual strategy for deterministic rendezvous in unknown graphs is the following. Each agent marks its initial location, explores the graphs and builds a labelled map of the graph. From the map and initial position of the agents in the map, the agents can compute a unique location (if it exists, i.e. if the map is asymmetric) and both agents move to that location.

Exploration of a digraph is relatively easy if each agent is provided with a unique marker distinct from that of the other agent (for instance, suppose that one agent has a spray-can of *blue* paint and the other agent has a spray-can of *red* paint). An agent can explore the digraph G by following a depth-first strategy and writing on each visited node using the distinct marker (so that it can recognize it on later visits). Such an algorithm for exploring a digraph is presented as Algorithm Simple-Explore (See Algorithm 1). During the algorithm, the agent maintains a counter that is incremented whenever a new node is visited. The (counter,marker) pair provides a unique label for each node v that is visited by the agent. It is easy to see that the algorithm succeeds in building a map of any strongly connected digraph G.

Lemma 3. *On executing Algorithm Simple-Explore using a unique marker, on any strongly connected digraph G, the agent outputs a map of the digraph.*

Lemma 4. *Algorithm Simple-Explore requires $O(m \cdot n)$ moves by an agent.*

In general, the agents would not have access to distinct markers at the beginning of the algorithm. We now show how two identical agents may obtain distinct

Algorithm 1. SimpleExplore(MARKER)

```
MAP ← φ; Count ← 1;
Write (MARKER, Count) on current node;
Increment Count;
Add current node to MAP;
while ∃ a reachable node in MAP that has unexplored arcs do
    Go to the closest reachable node u that has unexplored arcs;
    Choose the unexplored arc e with smallest label;
    Traverse arc e = (u, v) to reach node v;
    if v is marked with MARKER, (i.e. v ∈ MAP) then
        add arc e to MAP;
    else
        Write (MARKER,Count) on node v;
        Increment Count;
        Add node v and arc e to MAP;
Return MAP;
```

markers for exploring the digraph. Each agent starts by exploring the digraph and whenever it arrives at ith node in its traversal, it simply writes the integer i on the node. The agent a maintains a partial map M_a containing the nodes and arcs visited so far by the agent (The map M_a is a subgraph of G). The problem is that when the agent arrives at an already labelled node, it can not determine whether v was marked by itself (i.e. $v \in M_a$) or node v was marked by the other agent (i.e. $v \notin M_a$). At this stage the agent does not know how to update its map M_a. So, the agent a executes a *checking* procedure (to be described later), in order to determine if node v was marked by another agent. At the end of the checking procedure, either the agent a realizes that the node v is marked by the other agent, or, the agent a is able to come back to a known node u in its map M_a. In the latter case, the agent continues with the traversal without adding the node v to its map. While in the former case, the agent would have detected an asymmetry i.e. a difference in the maps of the two agents. Thus, in this case, the agent starts a new round of exploration (i.e. it executes Algorithm 1) using its current map M_a as the unique marker to mark nodes during the exploration. We present below a high level description of the algorithm. A more detailed description (pseudocode) is given in Algorithm 2. We need the following definition.

Definition 3. *For any two nodes $u, u' \in G$, (u, u') is called a pseudo-arc if there is a (not necessarily simple) directed cycle $(u, v_0, v_1, \ldots, v_t, u' = u_0, u_1, \ldots, u_t = u)$ for some $t \geq 0$, such that label(u_i)=label(v_i), $0 \leq i \leq t$, $\lambda(v_i, v_{i+1}) = \lambda(u_i, u_{i+1})$, $0 \leq i < t$, and $\lambda(u_t, v_0) = \lambda(v_t, u_0)$. We say that an agent traversed the pseudo-arc (u, u') whenever it traverses the path $(u, v_0, v_1, \ldots, v_t, u')$.*

Notice that if $u_i = v_i$, $0 \leq i \leq t$ in the above definition, then the pseudo-arc corresponds to an actual arc in $A(G)$. On the other hand if $t = 0$ i.e. $u = u'$, then the pseudo-arc corresponds to a pair of symmetric arcs between u and v_0.

Algorithm DirRDV

Round I: The agent executes the following steps:

1. The agent executes an exploration procedure during which it traverses the graph, marks the nodes that it visits with a counter (unless the node is already marked) and adds any node that it marks, to its local map (the map is a labelled subgraph of G containing the nodes marked by the agent). The exploration is interrupted whenever either the agent reaches an already marked node v or there are no more unexplored arcs that can be reached from the current node.
2. If the agent has reached an already marked node v, the agent executes procedure *Check-Path* at current node v using its current map, in order to determine if the node v belongs to its map. If procedure Check-Path returns false then node v does not belong to its map and the agent has detected an asymmetry in the digraph. In this case the agent executes round two of the algorithm. Otherwise if Check-Path returns true, the agent would have returned to a known node u (as shown later) and the agent continues the exploration from node u.
3. If there are no unexplored arcs incident to any reachable node in the map, then the agent terminates the exploration and executes procedure Check-MAP. If the procedure returns true then the current instance is not covering minimal and thus, the agent declares that rendezvous is not possible. If procedure Check-MAP returns false then the agent has detected an asymmetry in the digraph. In this case the agent executes round two of the algorithm.

During Round-I, whenever the agent arrives at an unmarked node v, it writes on the whiteboard the label of v which consists of the round number, value of the counter, the current map, and the number of unexplored arcs incident to v. The partial map M_a maintained by the agent a contains each node marked by this agent (along with its label) and each explored arc which the agent used to reach an unmarked node (such an arc is called *discovery arc*). For any explored arc that leads to a marked node, the agent adds a *pseudo-arc* to its map.

Round II: Let M_a be the map obtained by an agent a at the end of first round. The agent executes algorithm Simple-Explore using M_a as the marker. Let (G, μ_G) be the output of the procedure and λ be the arc labeling part of μ_G. The agent can obtain the labeling χ_p by assigning $\chi_p(v) := 1$ for any node v whose label has counter value of 1; $\chi_p(u) = 0$ for any other node u. If (G, λ, χ_p) is covering minimal then the agent computes a unique rendezvous location (that depends only on G, λ and χ_p) and moves to that location using its map. Otherwise the agent declares the problem is not solvable and terminates.

Procedure Check-Path: This procedure is executed at a marked node v, by an agent a that reached v through the arc $e = (u, v)$. Let the current map of agent a be M which is a labelled subgraph of G. If the label of v does not appear in M then the procedure returns false. Otherwise there is a node u' in M that

Algorithm 2. DirRDV

```
/* Algorithm for rendezvous with detect, for two agents.          */
Write HOME on the starting node;
begin
    Round := 1; // Begin 1st Round
    MAP := φ;
    Count := 1;
    Write visited(Round, MAP, Count) on current node;
    Increment Count;
    Add current node to MAP;
    while ∃ a reachable node in MAP that has unexplored arcs do
        Go to the closest reachable node u that has unexplored arcs;
        Choose the unexplored arc e with smallest label;
        Mark e as explored at node u;
        Traverse arc e to reach node v;
        if v is already marked then
            Result := execute Check-Path (MAP, u, e, label(v));
            if Result = true, (i.e. ∃u0 ∈ MAP s.t. label(v)=label(u0)) then
                Add pseudo-arc (u, u0) to MAP;
                Write the new MAP at node u0;
            else
                Go to next round;
        else
            Add node v and arc e to MAP;
            Write visited(Round, MAP, Count) on node v;
            Increment Count;
    Result := Execute Check-Map(MAP);
    if Result = true then
        return "Rendezvous is not possible" ;
    else
        Go to next round;
end
begin
    Round := 2; // Begin 2nd Round
    MARKER:= (Round, MAP);
    Map2 := Simple-Explore (MARKER);
    v := RV-point(Map2);
    Go to node v using Map2;
end
```

has the same label as v. In this case there exists a directed path P from u' to u in M. The agent tries to traverse, from the current node v, the path having a same label-sequence as path P. If such a path does not exist or the agent detects a discrepancy between the path it is traversing and the path P in M, then we know that node v was marked by the other agent and thus, $v \notin M$. In this case, the procedure returns false. Otherwise the procedure returns true and this implies either node u' is same as node v (i.e. arc e completes a cycle in M)

or u' and v are symmetric nodes in G. In either of these cases, the agent has returned to the known node u' in its map, at the end of the checking procedure.

Procedure Check-Map: The agent executes this procedure to check the accuracy of the map that it obtained. The agent first writes the current map on the whiteboard of each node that it originally marked. Then the agent traverses the path corresponding to each pseudo arc in the map and compares the label of each node with the corresponding node in its map. If there is a mismatch, the procedure returns false (i.e. the agent has detected an asymmetry in the digraph). Otherwise the procedure returns true at the end of the traversal. (In this case the map is the base of a covering on the labelled graph (G, λ, χ_p) as shown later.)

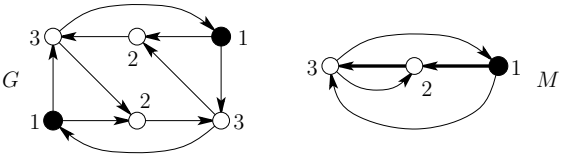

Fig. 2. The labelled digraph M is the map computed by the agents executing our algorithm on G. The thin arcs in M are the pseudo-arcs and black nodes are homebases.

Lemma 5. *If procedure Check-Path returns true for the arc (u, v), then the agent has returned to a known node u_0 in its current map and it has discovered a new pseudo-arc (u, u_0).*

Proof. Let v be the current node and $e = (u, v)$ be the last arc traversed by the agent (let's call it agent a), at the start of the procedure. The agent a is trying to determine whether node v belongs to its current map M_a. Note that $u \in M_a$ and there exists a node $u_0 \in M_a$ such that $label(u_0) = label(v)$ (otherwise the procedure would have returned false). If v was marked by agent a then $v = u_0$ (since the agent would not mark two nodes with the same label). In this case the agent returns to node u_0 at the end of the procedure after traversing the path $P = (u_0, u)$ followed by arc e.

Otherwise node v was marked by the other agent (say agent b), i.e. $v \neq u_0$. At the end of the procedure the agent arrives at a node w such that label(w)=label(v) (otherwise the procedure would return false). Note that either $w = v$ or $w = u_0$, since there are at most two nodes in the digraph having the same label. Suppose $w = v$ and let us consider the directed path P' traversed by the agent during the procedure. This path has the same labels as path P above and ends at the same node (v) as path P. There can not be two distinct incoming arcs with the same label at any node[1]. Thus the paths P and P' are the same, i.e. $u_0 = v$; A contradiction! Thus, the node $w \neq v$, which implies that $w = u_0$. So, the agent would finally arrive at node $u_0 \in M_a$ in either case. □

[1] Recall that we assume duplex arc-labeling.

Lemma 6. *If an agent reaches round two of the algorithm* **DirRDV***, then the two agents have distinct maps at the end of round one.*

Proof. An agent reaches round two, if during round one, either an execution of procedure Check-Path returns false or an execution of Check-MAP returns false. Suppose an agent a executes procedure Check-Path at node v after traversing arc $e = (u, v)$ and the procedure returns false. This implies that node $v \notin M_a$, the current map of agent a. Let M_b be the current map of the other agent. We know $v \in M_b$. If label(v) does not exist in M_a, then $M_a \neq M_b \bigcup H$ for any (possibly empty) subgraph H of G. Thus, even if agent b adds more nodes and arcs to its map, its map would never be same as M_a. Now let us consider the other case when one of the nodes $u_0 \in M_a$ has the same label as node $v \in M_b$. In this case there is no directed path starting from v that has the same sequence of labels as the path $P = (u_0, u) \in M_a$. The agent b can not mark any other node w to its map such that label(w)=label(v). Thus, the map of agent b would never contain any path corresponding to the path P in M_a. Thus the map M_a would be distinct from the map obtained by agent b at the end of its execution of round one of the algorithm.

Let us consider the other scenario when the procedure check-MAP returns false. In this case, either the map M_a is not strongly connected or the maps written on some of the nodes do not correspond to the map carried by the agent. In the latter case it is easy to see that the maps carried by the two agents are distinct. In the former case, notice that (G, λ, χ_p) is not a covering of M_a (since G is strongly connected and M_a is not). This implies that the map of the other agent (which is node disjoint from M_a) can not be an exact copy of M_a. □

The algorithm terminates unsuccessfully if the procedure CheckMAP returns true. This happens when the map constructed by the agent at the end of its exploration (i.e. when there are no more unexplored arcs) is a strongly connected digraph M such that the map written on any node that the agent visits is identical. In this case we can show that the original labelled digraph (G, λ, χ_p) is a covering of M.

Lemma 7. *If the procedure Check-MAP returns true, then the labeled digraph* (G, λ, χ_p) *is not covering minimal.*

Proof. Omitted.

Theorem 2. *For any strongly connected digraph G and a duplex arc-labeling λ on G, algorithm* **DirRDV** *solves the rendezvous of two agents placed initially according to χ_p, if (G, λ, χ_p) is covering minimal and otherwise detects that rendezvous is not solvable.*

Proof. The algorithm executed by an agent may terminate after round one only if the procedure CheckMAP returns true. In this case, we know that (G, λ, χ_p) is not covering minimal as proved in Lemma 7. In all other cases, both the agents reach round two of the algorithm. Thus, due to Lemma 6, the two agents have

distinct maps at the end of round one. The marker used by each agent consists of the round number and its map at the end of round one; This marker is distinct from the marker used by the other agent and also distinct from the markers used by both agents in the first round. Thus, due to Lemma 3 each agent successfully builds the map of (G, λ, χ_p) and if (G, λ, χ_p) is covering minimal, the agents can determine a unique location to meet. □

Theorem 3. *Algorithm Rendezvous requires $O(m.n)$ moves by the agents in total.*

Proof. The exploration part of the algorithm requires $O(m.n)$ moves by the agents (to explore each unexplored arc, $e = (u, v)$ the agent has to traverse at most n arcs to reach the source u of the arc). There are at most m calls to the procedure *Check-Path* and each execution of *Check-Path* requires $O(n)$ moves by an agent. The procedure Check-MAP is executed at most once during the algorithm and this requires $O(m.n)$ moves (since the size of the traversal path in the procedure is at most $m \cdot n$). Finally the second round of the algorithm has the same complexity as procedure *Simple-Explore*, which requires $O(m.n)$ moves as shown previously. □

4 Rendezvous without Incoming Arc Labels

In the previous section, we assumed the presence of local orientation on both the set of incoming arcs and the set of outgoing arcs incident to any node of the digraph. Having local orientation on the outgoing arcs is necessary for navigability in the digraph. In this section we consider digraphs G with simplex arc-labeling λ where incoming arcs at a node are not labelled. In fact, in this model, the agent can not even determine the in-degree of a node. The difference between the two models is the following. In the duplex arc-labeling model, any fibration on (G, λ, χ_p) was also an opfibration and vice versa. However, in the simplex arc-labeling model, there could exist an opfibration φ from (G, λ, χ_p) to a smaller labeled digraph even if (G, λ, χ_p) is covering minimal (and fibration-free). In this case there exists a node $v \in G$ such that two arcs e, f from distinct nodes u, u' lead to the node v such that $label(u) = label(u')$ and $\lambda(e) = \lambda(f)$. For example, see Figure 3 where the node labelled 4 has two similar incoming arcs (from two distinct nodes, both labelled with 2).

One consequence of the above observation is that Lemma 5 does not hold anymore. In other words, after executing procedure Check-Path, if the procedure returns true, then the agent may not have returned to a known node in its map. The agent is either back at a node u_0 in its own map or it is in some symmetric node v_0 in the map of the other agent. The agent was no means of determining which case is true. To solve this problem, we take the following approach. The agent continues with the exploration assuming that it is at node u_0. In this case the agent may have swapped places with the other agent or both agent may be in the same map. As the two subgraphs corresponding to the two partial maps are symmetric, it does not matter which part the agent is in. In fact we can

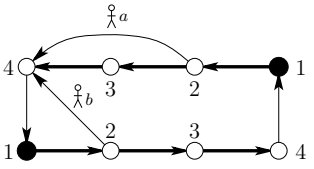

Fig. 3. An execution of the algorithm on a digraph G without incoming local orientation

show that the same algorithm from the previous section works also for the case of simplex arc labeling.

Theorem 4. *For any strongly connected digraph G, and a simplex arc-labeling λ on G, algorithm* **DirRDV** *solves the rendezvous of two agents placed initially according to χ_p, if (G, λ, χ_p) is covering minimal and otherwise detects that rendezvous is not solvable.*

Proof. During the exploration in round one of the algorithm, each agent maintains a map of the subgraph explored by it. Let M_a and M_b be the partial maps of the two agents a and b respectively. If the two maps are not symmetric, the agents will detect the asymmetry (during an execution of Check-Path or CheckMAP). Notice that Lemma 6 still holds. So, if the agents go to Round-2, they would have distinct maps and thus, they will successfully solve the rendezvous problem. Further Lemma 7 also continues to hold and if (G, λ, χ_p) not covering minimal, the agents detect that rendezvous is not possible.

The only difference from the previous section occurs when the maps of the two agents are identical but there exists a node v of G which has two incoming arcs e, f having the same label $x = \lambda(e) = \lambda(f)$. Suppose node $v \in M_a$, then there exists a node $v' \in M_b$ with identical label as node v. (For an illustration, see Figure 3 where the nodes labeled 4 correspond to v and v'). In this case, (G, λ, χ_p) is still covering minimal and we show below that the two agents indeed succeed in solving rendezvous for this instance.

The agent that is first to arrive at node v through either arc e or arc f, would add an arc labeled x incident to v in its map and write the new map at v. The other agent that arrives later to v through the other arc, would notice that the map at v already contains an arc labeled x incoming at the node v. This inconsistency would be detected when this agent executes Check-Path for the new arc that it traversed. Thus this agent would execute Round-2 of the algorithm using a map that does not contain the arcs e or f (such a map is distinct from the map of the other agent). Now consider the other agent which added the arc labeled x to its map. Note that the map written at v' does not contain such an arc. Thus, when the agent executes CheckMAP at the end of Round-1, it will arrive at node v' and detect the inconsistency in the maps. So, the procedure CheckMAP would return false and the agent would execute Round-2 of the algorithm using a map which contains one of the arcs e or f. Thus, the two agents execute round-2 using distinct maps (i.e. distinct markers) and they would succeed in solving rendezvous. □

5 Conclusions

In this paper, we considered the rendezvous of two agents in strongly connected digraphs. If the digraph G is not strongly connected, it is not always possible to explore the graph in general and thus, rendezvous is not possible. For any strongly connected digraph G, the algorithm we presented achieves the rendezvous of two agents whenever it is deterministically possible, i.e. whenever the labeled digraph (G, λ, χ_p) is covering minimal (or fibration-free). The rendezvous problem requires breaking the symmetry between two identical agents and a generalization of this problem, called the *Gathering* problem, is to gather together $k > 2$ agents at a single node of G. To solve the gathering of $k > 2$ agents we need to extend our solution by generalizing the technique of mapping paths in the graph using pseudo-arcs. A straightforward generalization of the algorithm would involve replacing each pseudo-arc traversal with the traversal of a cycle in the map, repeated $r = \gcd(1, 2, \ldots k)$ times (instead of just twice as in our algorithm), in order to ensure that the agent returns to the correct subgraph. However, this will blowup the cost of the algorithm exponentially. So, an interesting question to be answered by future research is whether $k > 2$ agents can gather in anonymous digraphs such that the number of moves is polynomial in the parameters n, m and k. For undirected graphs, it is already known to be possible using $O(m \cdot k)$ moves.

References

1. Albers, S., Henzinger, M.R.: Exploring Unknown Environments. SIAM Journal on Computing 29(4), 1164–1188 (2000)
2. Angluin, D.: Local and global properties in networks of processors. In: Proc. of 12th Symposium on Theory of Computing (STOC), pp. 82–93 (1980)
3. Barrière, L., Flocchini, P., Fraigniaud, P., Santoro, N.: Can we elect if we cannot compare? In: Proc. 15th ACM Symp. on Parallel Algorithms and Architectures (SPAA'03), pp. 200–209 (2003)
4. Barrière, L., Flocchini, P., Fraigniaud, P., Santoro, N.: Election and rendezvous in fully anonymous networks with sense of direction. Theory of Computing Systems 40(2), 143–162 (2007)
5. Baston, V., Gal, S.: Rendezvous search when marks are left at the starting points. Naval Research Logistics 48(8), 722–731 (2001)
6. Bender, M., Fernandez, A., Ron, D., Sahai, A., Vadhan, S.: The power of a pebble: Exploring and mapping directed graphs. In: Proc. 30th ACM Symp. on Theory of Computing (STOC), pp. 269–287 (1998)
7. Boldi, P., Vigna, S.: An effective characterization of computability in anonymous networks. In: Welch, J.L. (ed.) DISC 2001. LNCS, vol. 2180, pp. 33–47. Springer, Heidelberg (2001)
8. Boldi, P., Vigna, S.: Fibrations of graphs. Discrete Math. 243, 21–66 (2002)
9. Czyzowicz, J., Dobrev, S., Kralovic, R., Miklík, S., Pardubská, D.: Black Hole Search in Directed Graphs. In: Kutten, S., Žerovnik, J. (eds.) SIROCCO 2009. LNCS, vol. 5869, pp. 182–194. Springer, Heidelberg (2010)
10. Dessmark, A., Fraigniaud, P., Kowalski, D., Pelc, A.: Deterministic rendezvous in graphs. Algorithmica 46, 69–96 (2006)

11. Dobrev, S., Flocchini, P., Prencipe, G., Santoro, N.: Multiple agents rendezvous in a ring in spite of a black hole. In: Papatriantafilou, M., Hunel, P. (eds.) OPODIS 2003. LNCS, vol. 3144, pp. 34–46. Springer, Heidelberg (2004)
12. Dobrev, S., Flocchini, P., Kralovic, R., Santoro, N.: Exploring a dangerous unknown graph using tokens. In: Proc. of 5th IFIP International Conference on Theoretical Computer Science, TCS (2006)
13. Fraigniaud, P., Ilcinkas, D.: Digraph exploration with little memory. In: Diekert, V., Habib, M. (eds.) STACS 2004. LNCS, vol. 2996, pp. 246–257. Springer, Heidelberg (2004)
14. Czyzowicz, J., Labourel, A., Pelc, A.: How to meet asynchronously (almost) everywhere. In: Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA (2010)
15. Fraigniaud, P., Pelc, A.: Deterministic Rendezvous in Trees with Little Memory. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 242–256. Springer, Heidelberg (2008)
16. Kowalski, D.R., Malinowski, A.: How to meet in anonymous network. Theoretical Computer Science 399(1-2), 141–156 (2008)
17. Kranakis, E., Krizanc, D., Markou, E.: Mobile agent rendezvous in a synchronous torus. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, pp. 653–664. Springer, Heidelberg (2006)
18. Yamashita, M., Kameda, T.: Computing on anonymous networks: Part I–Characterizing the solvable cases. IEEE Transactions on Parallel and Distributed Systems 7(1), 69–89 (1996)
19. 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)