Rendezvous of Mobile Agents without Agreement on Local Orientation

Jérémie Chalopin and Shantanu Das

LIF, CNRS & Aix Marseille Université,
39 rue Joliot Curie, 13453 Marseille cedex 13, France
jeremie.chalopin@lif.univ-mrs.fr, shantanu.das@acm.org

Abstract. The exploration of a connected graph by multiple mobile agents has been previously studied under different conditions. A fundamental coordination problem in this context is the gathering of all agents at a single node, called the *Rendezvous* problem. To allow deterministic exploration, it is usually assumed that the edges incident to a node are locally ordered according to a fixed function called local orientation. We show that having a fixed local orientation is not necessary for solving rendezvous; Two or more agents having possibly distinct local orientation functions can rendezvous in all instances where rendezvous is solvable under a common local orientation function. This result is surprising and extends the known characterization of solvable instances for rendezvous and leader election in anonymous networks. On one hand, our model is more general than the anonymous port-to-port network model and on the other hand it is less powerful than qualitative model of Barrière et al. [4,9] where the agents have distinct labels. Our results hold even in the simplest model of communication using identical tokens and in fact, we show that using two tokens per agent is necessary and sufficient for solving the problem.

Keywords: Distributed Coordination, Mobile Agents, Rendezvous, Synchronization, Anonymous Networks, Incomparable Labels.

1 Introduction

Consider an undirected connected graph G that is explored by some mobile entities (called agents) that move along the edges of G. Such a setting occurs in many practical scenarios such as network monitoring, search and rescue operations, intruder detection and exploration of unknown territories by robots. The objective of the agents may be to gather information, search for some resource or make periodic visits to the vertices of G. When there are multiple autonomous agents operating in such an environment, it may be sometimes necessary to gather all the agents at a single location, for example to exchange information or for assignment of duties to the agents. This is a fundamental problem in distributed coordination, known as the *Rendezvous* problem and there is a long history of research on achieving rendezvous in graphs using either deterministic or probabilistic means (see [1] for a survey). This paper focusses on the deterministic

S. Abramsky et al. (Eds.): ICALP 2010, Part II, LNCS 6199, pp. 515–526, 2010.

setting. Notice that if the nodes of the graph G have distinct labels (and the agents can perceive these labels), then rendezvous can be achieved trivially by moving to a predetermined location. The more challenging problem is to rendezvous in a graph where the nodes do not have distinct identifiers. This problem relates to the more general question of what can be computed in a distributed network of anonymous processors [2,3,16].

As an example, consider the case of agents moving on an asynchronous ring, starting from distinct nodes. Clearly, rendezvous is not possible here if the agents keep following each other on the cycle forever. To facilitate rendezvous in such situations, each agent can be equipped with a marking device called the token (or pebble)[1] which can be used to mark the starting location of the agent [5]. Even with the capability of marking nodes, it is not always possible to deterministically solve rendezvous. We are interested in algorithms that achieve rendezvous whenever it is solvable, and otherwise detect and report the fact that is not solvable for the given instance. Such an algorithm is said to solve the *Rendezvous with Detect* problem.

Solving the rendezvous problem requires the agent to explore the graph G. In order to allow the agents to navigate through the graph G, it is usually assumed that the edges incident at any node v are locally ordered as $1, 2, \ldots d$ where d is the degree of v. While local orientation (or port numbering) is usually introduced to ensure navigability, it is inadvertently assumed that the agents agree on the local orientation, and a common order on the port numbers is then used to break symmetries. In this paper, we show that with or without agreement on the local orientation, rendezvous can be solved in the same distributed environments. In general, the edges of G may be assigned labels from some set Γ which is unknown to the algorithm designer and thus there may not be a predetermined order on the set of labels. Each agent, depending on its perceptions of the labels it encounters, may choose some arbitrary order on the set of labels. We define the rendezvous problem in this setting as "*Rendezvous without (agreement on) common port numbering*" or, **RVwA** to distinguish it from the usual setting where all agents agree on a common port numbering. The latter problem, which is a special case of the former, would be called "*Rendezvous with common port numbering*" or, **RVCP**.

Notice that the traversal path followed by the agent would vary depending on the port numbering used by that agent. Thus, it is no longer possible to determine the actions of an agent based only on the starting location of the agent in the graph. This is an important distinction between the two models (i.e. with or without common port numbering). In the former model, each agent can determine the so called *view* [16] of the other agents based on its traversal path and one common strategy for rendezvous is to compare the views of the agents and elect a leader. Such a strategy is no longer possible in our model and we require completely different techniques for solving the problem.

[1] The tokens can not be used for identifying an agent, since all agents carry identical tokens.

Our model is similar to the qualitative model of computation [4,9] where the incident edges at a node have distinct labels but the agents do not agree on a total order on these labels (i.e. the labels are said to *incomparable*). However, the model considered in the above papers also assume the agents to be distinct (i.e. distinguishable) and gives them the ability to write messages on public *whiteboards* available at each node. We, on the other hand, consider the much weaker model where the agents are indistinguishable from each other and have to rendezvous in an anonymous network using only tokens to mark the nodes (and no explicit communication). In fact, we show that the problem of **RVwA** is solvable in exactly those instances where rendezvous with common port numbering is possible. In contrast, in the qualitative model of Barrière et al. [4,9] the class of solvable instances is much larger.

Our Results: We present an algorithm for solving rendezvous of mobile agents in arbitrary graphs when there is no agreement on a common port numbering. Our algorithm solves the problem whenever a deterministic solution is possible and otherwise detects the impossibility of rendezvous. This gives us a characterization of the solvable instances for the **RVwA** problem. Surprisingly, it turns out that this characterization coincides with that for the conventional version of the problem (**RVCP**) where a common port numbering is assumed. In other words, our result shows that for solving rendezvous, the assumption of a common port numbering is not necessary (it is sufficient that the incident edges are distinguishable).

The algorithm presented in this paper uses exactly two tokens per agent and we show that two tokens are necessary for solving the problem in the absence of other capabilities (e.g. whiteboards or distant communication). This contrasts nicely with the simpler problem of **RVCP**, where a single token per agent is known to be necessary and sufficient (e.g. see [12]). In fact, our algorithm uses two tokens because we need to synchronize between the actions of the agents[2]. We believe our synchronization technique, using tokens, will be useful for solving other problems in asynchronous settings.

Related Results: Our work is related to a long line of research on computability in anonymous environments 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.* [3] and by Yamashita and Kameda [16] among others. In the message-passing model with port-to-port communication, each vertex of the graph is associated with a single agent; the agents are stationary and can send and receive messages over the incident edges to communicate with neighbors. The leader election problem in message-passing systems is equivalent to the rendezvous problem in the mobile agent model [7].

[2] Note that for synchronous environments, our algorithm can be adapted to use only one token.

In the deterministic setting, initial solutions for rendezvous were for synchronous environments with the prior knowledge of the topology [18]. An interesting line of research is determining what minimum capabilites allow the agents to solve rendezvous. For instance, Fraigniaud and Pelc [13] consider agents with small memory while Klasing et al. [14] consider robots with no memory of previous steps (though these robots can see at each step, the complete graph and the current location of the other robots). Recently, Czyzowicz et al. [10] considered a model where the agents have unique identifiers, but they have no way to communicate with each other (they cannot leave messages or tokens on the nodes).

2 Definitions and Notations

The Model: The environment is represented by a simple undirected connected graph $G = (V(G), E(G))$ and a set $\mathcal{E}$ of mobile agents that are located in the nodes of G. The initial placement of the agents is denoted by the function $p : \mathcal{E} \to V(G)$. We denote such a distributed mobile environment by $(G, \mathcal{E}, p)$ or by (G, χ_p) where χ_p is a vertex-labelling of G such that $\chi_p(v) = 1$ if there exists an agent a such that $p(a) = v$, and $\chi_p(v) = 0$ otherwise.

In order to enable navigation of the agents in the graph, at each node $v \in V(G)$, the edges incident to v are distinguishable to any agent arriving at v. For each agent $a \in \mathcal{E}$, there is a bijective function $\delta_{(a,v)} : \{(v, u) \in E(G) : u \in V(G)\} \to \{1, 2, \ldots d(v)\}$ which assigns unique labels to the edges incident at node v (where $d(v)$ is the degree of v). In the literature, it is generally assumed that all agents have a common local orientation function: in that case, there exists a port numbering δ such that for each agent a, $\delta_a = \delta$. The nodes of G do not have visible identities by which the agents can identify them. Each agent carries some identical tokens. Each node v of G is equipped with a stack and any agent that is located at v can insert tokens or remove tokens from the stack. Access to the stack at each node, is in fair mutual exclusion Each time an agent gains access to the stack, the agent can see the number of tokens in the stack, and depending on this number, it may put a token, remove a token, or do nothing. Initially no node contains any token and each agent has exactly two tokens. The system is asynchronous; an agents may start at the any time and every action it performs may take an unpredictable (but finite) amount of time. The agents have no means of direct communication with each other. An agent can see another agent only when they are both located at the same node (and not when they are traversing the same edge). All agents start in *active* state, but during the algorithm an agent a may become *passive* and return to its homebase $p(a)$ to *sleep*. Any other agent located at the same node can see a *sleeping* agent and can initiate a process to wake-up such a *sleeping* agent. Note that the agents can distinguish a *sleeping* agent a from any other agent that may be waiting (actively) at the node v.

We also assume that initially each agent knows n, the size of G. We will explain in Section 5, how we can weaken this hypothesis. The agents have no

other prior knowledge about the environment (e.g. the topology, the number of agents, are both unknown a priori).

Graph Coverings and Automorphisms: We now present some definitions and results related to coverings of directed graphs (or multigraphs, since we allow multiple arcs). A directed graph(digraph) $D = (V(D), A(D), s, t)$ possibly having parallel arcs and self-loops, is defined by a set $V(D)$ of vertices, a set $A(D)$ of arcs and by two maps s and t that assign to each arc two elements of $V(D)$: a source and a target. A *symmetric* digraph D is a digraph endowed with a symmetry, i.e. an involution $Sym : A(D) \rightarrow A(D)$ such that for every $a \in A(D), s(a) = t(Sym(a))$. We now define the notion of graph coverings, borrowing the terminology of Boldi and Vigna [6]. A *covering projection* is a homomorphism φ from a digraph D to a digraph D' satisfying the following: For each arc a' of $A(D')$ and for each vertex v of $V(D)$ such that $\varphi(v) = v' = t(a')$ (resp. $\varphi(v) = v' = s(a')$) there exists a unique arc a in $A(D)$ such that $t(a) = v$ (resp. $s(a) = v$) and $\varphi(a) = a'$. If a covering projection $\varphi : D \rightarrow D'$ exists, D is said to be a *covering* of D' via φ and D' is called the base of φ. For symmetric digraphs D, D', φ is a *symmetric covering* if $\forall a \in A(D), \varphi(Sym(a)) = Sym(\varphi(a))$. A digraph D is *symmetric-covering-minimal* if there does not exist any graph D' not isomorphic to D such that D is a symmetric covering of D'.

We consider labelled (di)graphs: a (di)graph G whose vertices are labelled over L will be denoted by (G, λ), where λ: $V(G) \rightarrow L$ is the labelling function[3]. We will consider homomorphisms which preserve the labelling on the vertices. Given a connected undirected graph $G = (V(G), E(G))$, we can associate it with a symmetric strongly connected digraph denoted by $Dir(G)$ and defined as follows: $V(Dir(G)) = V(G)$ and for each edge $\{u, v\} \in E(G)$, there exist two arcs $a_{(u,v)}, a_{(v,u)} \in A(Dir(G))$ such that $s(a_{(u,v)}) = t(a_{(v,u)}) = u$, $t(a_{(u,v)}) = s(a_{(v,u)}) = v$ and $Sym(a_{(u,v)}) = a_{(v,u)}$. A distributed mobile environment $(G, \mathcal{E}, p)$ can be represented by the symmetric labelled digraph $(Dir(G), \chi_p)$.

For simple connected undirected graph, Yamashita and Kameda [16] introduced the concept of views which we present below:

Definition 1. *Given a labelled graph (G, λ) with a port numbering δ, the* view *of a node v is the infinite rooted tree denoted by $T_G(v)$ defined as follows. The root of $T_G(v)$ represents the node v and for each neighbor u_i of v, there is a vertex x_i in $T_G(v)$ (labelled by $\lambda(u_i)$) and an edge from the root to x_i with the same labels as the edge from v to u_i in (G, δ). The subtree of $T_G(v)$ rooted at x_i is again the view $T_G(u_i)$ of node u_i.*

It is known that from the view of any vertex v truncated to depth $2n$, one can construct a labelled digraph (D, μ_D) such that $(Dir(G), \delta)$ is a symmetric covering of (D, μ_D) (See e.g. [3]). The digraph (D, μ_D) corresponds to the *quotient graph* in Yamashita and Kameda's work [16].

[3] Note that this labelling is not necessarily injective, i.e. two vertices may have the same label.

For any labelled undirected graph (G, λ), two vertices $v, v' \in V(G)$ are *similar* (we write $v \sim v'$) if there exists a label-preserving automorphism σ of $(Dir(G), \lambda)$ such that $v = \sigma(v')$. The relation $\sim$ is an equivalence relation over the vertices of G; the equivalence class of $v \in V(G)$ is denoted by $[v]$. For two disjoint subsets of vertices $V, V' \subseteq V(G)$, $G[V]$ denotes the subgraph of G induced by the vertices in V and $G[V, V']$ denotes the bipartite subgraph of G defined as follows. The vertices of $G[V, V']$ are $V \cup V'$ and there is an edge $\{v, v'\}$ in $G[V, V']$ if and only if $v \in V, v' \in V'$ and $\{v, v'\} \in E(G)$.

3 Characterization for Rendezvous

We present the following theorem that relates the various existing characterizations [6,16,17] of distributed mobile environments where the **RVCP** problem can be solved. Note that some of these characterizations were initially given for the problem of leader election in message passing systems.

Theorem 1 ([6,16,17]). *For any distributed mobile environment $(G, \mathcal{E}, p)$, the following statements are equivalent:*

1. *For any common port-numbering function δ, Rendezvous with common port-numbering can be solved in $(G, \mathcal{E}, p, \delta)$;*
2. *For any port-numbering function δ, all vertices of (G, χ_p, δ) have a different view;*
3. *There is no partition $V_1, V_2, \ldots V_k$ of $V(G)$ with $k \in [1, |V(G)| - 1]$ such that for any distinct $i, j \in [1, k]$, the following conditions hold:*
   - *(i) $G[V_i]$ is d-regular for some d, and if d is odd, it contains a perfect matching,*
   - *(ii) $G[V_i, V_j]$ is regular.*
4. *$(Dir(G), \chi_p)$ is symmetric-covering-minimal.*

Since the **RVCP** problem is a special case of the **RVwA** problem, the conditions of Theorem 1 are also necessary for solving rendezvous when the agents do not agree on the local orientation. The following theorem is the main result of our paper; it shows that when each agent is supplied with a sufficient number of (identical) tokens, **RVwA** can be solved in $(G, \mathcal{E}, p)$ if and only if **RVCP** can be solved in $(G, \mathcal{E}, p)$. We prove the theorem by providing an algorithm that achieves this result.

Theorem 2. *There exists an algorithm for rendezvous without common port-numbering in a distributed mobile environment $(G, \mathcal{E}, p)$ if and only if $(Dir(G), \chi_p)$ is symmetric-covering-minimal.*

4 Our Rendezvous Algorithm

We present an algorithm for achieving rendezvous in the absence of a common port numbering (see Algorithm 1). The algorithm has an initialization round and multiple rounds of marking where the agents collectively try to break the

symmetry among the vertices of the graph, until a single vertex can be chosen as the rendezvous location. Figure 1 illustrates the execution of the algorithm on a small toy-graph.

Initialization and Map Construction: At the beginning of the algorithm, each agent a puts a token on its homebase and constructs its view up to depth $2n$ (recall that the agent knows n, the number of vertices of G). Since the other agents may not have started at the same time, agent a may reach a homebase containing a *sleeping* agent. In that case agent a wakes-up the sleeping agent and waits until this agent puts a token on the node.

Algorithm 1. RDV (n)

```
Put a token on your own homebase ;
(D, μ) := ConstructMap(n) ;
/* Each agent computes (D, μ) such that (Dir(G), χ_p) covers (D, μ)   */
if (G, χ_p) is not symmetric-covering-minimal then
    Terminate and report that rendezvous cannot be solved in (G, E, p) ;
else
    /* Each agent knows a rooted map of (G, μ) = (G, χ_p)             */
    Synchronize ;
    Construct equivalence partition of (G, μ) ;
    repeat
        H := The minimal class that contains active homebases ;
        /* H defines the set of active agents for this round          */
        Active agents wait for passive agents to return to their homebases;
        if there exists a class C such that |H| > |C| then
            /* We split the class H into two                          */
            C := minimal class such that |H| > |C| ;
            Active agents (i.e. whose homebase ∈ H) try to mark vertices in C ;
            /* The ones that do not succeed become passive            */
            Synchronize ;
            Distinguish active homebases from passive homebases;
            Refine the labelling μ to give fresh labels to passive homebases;
        else if there exists C = [v] such that |H| < |C| then
            /* We split the class C in two.                           */
            C := minimal class such that |H| < |C| ;
            Each active agent mark a vertex in C ;
            Synchronize ;
            Refine the labelling μ to give fresh labels to marked vertices ;
        Re-construct equivalence partition of (G, μ);
    until There is only one vertex in each class ;
    h := The active homebase with minimum label ;
    The agent whose homebase is h, marks h and wakes up all the other agents;
    All other agents go to the only marked vertex, when woken up;
```

As mentioned before, the information contained in the view is sufficient to construct the quotient graph (D, μ) such that $(Dir(G), \chi_p)$ is a symmetric covering of (D, μ). Since each agent initially knows $n = |V(G)|$, it can determine whether (G, χ_p) is symmetric covering minimal or not. In the former case, the agent has obtained a map of (G, χ_p), while in the latter case, the agent reports that rendezvous is not solvable and terminates the algorithm.
Each agent r executes the following steps during the initialization procedure:

(Step A) Agent r puts a token on its homebase.
(Step B) Agent r constructs its view and builds the quotient graph. If $(Dir(G), \chi_p)$ is not minimal, agent r detects it and terminates the algorithm.
(Step C) Agent r puts a second token on its homebase
(Step D) Agent r checks that all agents have put zero or two tokens on their homebases.
(Step E) Agent r removes the two tokens from its homebase
(Step F) Agent r goes to each homebase h, and waits until either there is no token on h, or there is a sleeping agent on h.

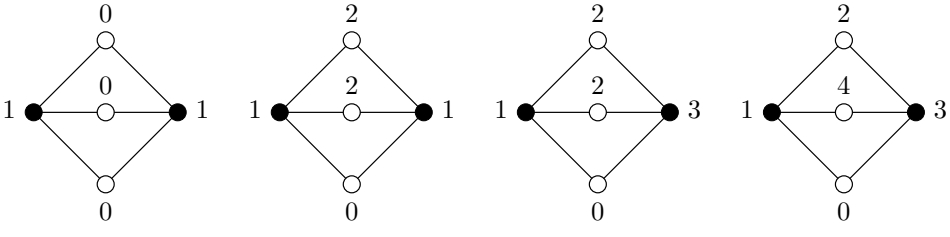

Fig. 1. The different labellings of a graph G computed by the agents during one particular execution of our algorithm on the distributed environment $(G, \mathcal{E}, p)$ represented on the left (black vertices represent homebases)

Constructing an order on equivalence classes: Given any vertex-labelled graph (G, μ), there exists standard procedures for partitioning the vertex set of G to obtain an ordered sequence of equivalence classes that respect the labelling. (See [4,9] for example.) Note that since the agents start with the same labelled graph (G, μ), they compute the same ordered sequence of equivalence classes.

Synchronization Procedure: We now describe the procedure for the marking round. At the beginning of each marking round, the agents choose two classes H and C, where H is the class of active homebases and C is the class of vertices to be marked by the agents during this round. Since the agents agree on the order of the equivalence classes, they also agree on the choice of H and C. (H is the first class in the sequence that contains active homebases and C is the first class whose size is either larger or smaller than that of H).

An agent whose homebase belongs to the class H is active during the current round. Each active agent r executes the following instructions. We suppose that at the beginning of the round, each agent carries its two tokens.

(Step 0) Agent r goes to the homebase of each passive agent (i.e. any homebase $\notin H$) and waits until there is a sleeping agent at that node.

(Step 1) Agent r traverses the graph G using its map, and if it arrives at a node $c \in C$ where there is no token, it puts a token at c and sets its state to SUCCESS. If agent r does not manage to mark any vertex of C (this can happen only when $|C| < |H|$), then it puts a token on any vertex c of C (that has already been marked by another agent). Such an agent sets its state to FAIL.

(Step 2) The agent r does a traversal of the graph and counts the total number of tokens on the vertices $\in C$. If there are strictly less tokens on vertices of C than the number of active agents (because some other agent has not finished the previous step), then agent goes back to its homebase and sleeps until it is woken up by another agent; On wake-up, agent r traverses the graph again to count the number of tokens on the vertices of C. When agent r completes this step, all other agents must have completed the previous step.

(Step 3) The agent r puts the second token on its homebase.

(Step 4) The agent r does a traversal of the graph and each time it arrives on the homebase h of another active agent r', it waits until there is a token or a sleeping agent at h. If there is a sleeping agent at h, agent r wakes it up and waits until there is a token on node h.

(Step 5) The agent goes back to the vertex $c \in C$ where it placed a token during Step 1 and removes a token from c.

(Step 6) The agent r does a traversal of the network. Each time it arrives on a vertex $c' \in C$, it waits until all tokens have been removed from c'.

(Step 7) The agent r removes the token from on its homebase. If the state of the agent is FAIL, then it becomes passive (goes to sleep). Otherwise the agent executes the next step.

(Step 8) The agent r does a traversal of the network. Each time it arrives on the homebase h of an active agent r', it waits until there is no token or there is a sleeping agent at h.

Let us consider two different scenarios.

If $|\mathbf{C}| > |\mathbf{H}|$, then the class C will be partitioned into marked and unmarked vertices. Moreover, each agent knows from the information it has stored on the map at Step 2 which vertices of C has been marked in this round. The agents would relabel the marked vertices to obtain the new labelling μ.

If $|\mathbf{C}| < |\mathbf{H}|$, then there would be two categories of agents: those whose state=SUCCESS and those whose state=FAIL. The former would remain active but the latter would become passive at the end of this procedure. In order to distinguish the active homebases in H from the passive homebases another procedure (explained below) is executed by the active agents.

Distinguishing Active from Passive Homebases: The following procedure is used for partitioning the class H in a round where the class C is smaller than the class of homebases H. The active agents (i.e. those who were successful in the marking process) can put tokens on their homebases to distinguish them. However, in order to avoid conflicts during consecutive rounds, the active agents must first mark vertices of C and not of H. Indeed, when an agent starts this procedure, there may be another agent that is still performing Step 8 of the synchronization procedure. The following steps are followed by an active agent r during this procedure (Note that exactly $|C|$ agents execute this procedure).

(Step 1') Agent r traverses the graph and puts a token on a vertex c of C that contains no token.
(Step 2') The agent r does a traversal of the graph and on each vertex c of C, the agent waits until there is a token on c.
(Step 3') The agent r puts a token on its homebase.
(Step 4') The agent r does a traversal of the graph and at each node $h \in H$, agent r waits until there is a token or there is a sleeping agent. Note that if node h is homebase of an active agent r', then agent r' will eventually complete Step 3' and thus there will be a token at h. Otherwise if h is the homebase of a passive agent r'' then agent r'' will eventually complete the previous procedure and return to its homebase to sleep.
(Step 5') The agent r goes to the node c that it marked in Step 1' and removes the token.
(Step 6') The agent r does a traversal of the network. Each time it arrives on a vertex c' of C, it waits until there is no token on c'.
(Step 7') The agent r removes the token from its homebase.
(Step 8') The agent r does a traversal of the graph and at each node $h \in H$, agent r waits until there is no token at h or there is a sleeping agent at h.

At the end of this procedure each active agent knows which homebases are still active. Hence the agent can relabel the passive homebases to obtain the new labelling μ (i.e. the class H is partitioned into two subclasses).

Refining the Vertex Labelling μ of (G, μ): During the algorithm, each agent keeps track of the labels assigned to the vertices of G by writing them in its local map. Since all active agents agree on the order of equivalence classes, they would agree on the label-assignment.

During each round, some class C_j is split into two sub-classes and vertices in the first subclass are assigned a new label (e.g. the smallest integer not yet used as a label). At the end of a round, the vertex set is repartitioned while respecting the labels assigned to the vertices and after the repartitioning the labelling is refined accordingly. If the new labelling is injective, the algorithm terminates; Otherwise the next round is started.

Correctness of the Algorithm: During our algorithm, the equivalence partitioning is refined in each round using two classes of distinct sizes. We consider now a configuration such that all equivalence classes have the same size. Let q be the maximum number appearing on a vertex of (G, μ); the vertices from G are labelled by numbers in $[0, q]$. Let $V_0, V_1, \ldots, V_q$ be the partition of $V(G)$ defined by μ, i.e., $v \in V_i$ if $\mu(v) = i$. For any distinct $i, j \in [0, q]$, since V_i and V_j are equivalence classes of (G, μ), all vertices in V_i (resp. V_j) have the same number $d_{(i,j)}$ (resp. $d_{(j,i)}$) of neighbors in V_j (resp. V_i). Since the number of edges in $G[V_i, V_j]$ is $|V_i|d_{(i,j)} = |V_j|d_{(j,i)}$ and since $|V_i| = |V_j|$, $d_{(i,j)} = d_{(j,i)}$ and thus $G[V_i, V_j]$ is regular. For any $i \in [0, q]$, since V_i is an equivalence class, $G[V_i]$ is vertex-transitive and thus d_i-regular for some d_i. From Gallai-Edmonds decomposition (see [15]), we know that any vertex-transitive graph of odd degree admits a perfect matching. Consequently, if d_i is odd, we know that $G[V_i]$ admits a perfect matching. Thus, from the characterization of Yamashita and Kameda [17] given in Theorem 1, and since we know that (G, χ_p) is symmetric-covering-minimal, we know that each V_i contains exactly one vertex. In other words, the labelling μ is injective. Thus, the algorithm succeeds in selecting a unique rendezvous location whenever (G, χ_p) is symmetric-covering-minimal.

5 Further Remarks

In this section we justify some of the assumption made in this paper. Based on Angluin's impossibility result [2] (see also [8]), we know that there does not exist any universal algorithm that solves rendezvous with detect. Thus, some prior knowledge about the environment is necessary. We assumed the knowledge of n, the size of the graph; However we can easily adapt our algorithm to use only a bound on the size of the graph instead.

Proposition 1. *If the agents initially know a tight bound $B \in [n, 2n-1]$ on the size of G, the agents can rendezvous in $(G, \mathcal{E}, p)$, or detect it is impossible. Further, if the agents initially know that $(Dir(G), \chi_p)$ is symmetric-covering-minimal then knowledge of an arbitrary bound $B \geq n$ on the size of G is sufficient to solve rendezvous in $(G, \mathcal{E}, p)$.*

Our algorithm presented in Section 4 requires two identical tokens per agent. Under the assumption that each agent knows only the size of G, but has no information on the number of agents, we show that two tokens (per agent) are necessary to solve **RVwA**. However, it remains open to determine if two tokens are still necessary when the agents know completely the environment $(G, \mathcal{E}, p)$.

Proposition 2. *There exists graphs G such that there is no algorithm using one token per agent such that for each distributed mobile environment $(G, \mathcal{E}, p)$, either the agents solve rendezvous in $(G, \mathcal{E}, p)$, or detect that it is impossible.*

Finally, we consider the cost of solving **RVwA** in terms of the number of moves made by the agents. Our algorithm requires an additional $O(n^2 k)$ moves for solving rendezvous without common port numbering, compared to the solution for rendezvous with common port-numbering [12] using tokens.

References

1. Alpern, S., Gal, S.: The Theory of Search Games and Rendezvous. Kluwer, Dordrecht (2003)
2. Angluin, D.: Local and global properties in networks of processors. In: Proc. 12th Symposium on Theory of Computing (STOC 1980), pp. 82–93 (1980)
3. Boldi, P., Codenotti, B., Gemmell, P., Shammah, S., Simon, J., Vigna, S.: Symmetry breaking in anonymous networks: characterizations. In: Proc. 4th Israeli Symp. on Theory of Computing and Systems (ISTCS 1996), pp. 16–26 (1996)
4. Barrière, L., Flocchini, P., Fraigniaud, P., Santoro, N.: Can we elect if we cannot compare? In: Proc. 15th Symp. on Parallel Algorithms and Architectures (SPAA), pp. 324–332 (2003)
5. Baston, V., Gal, S.: Rendezvous search when marks are left at the starting points. Naval Research Logistics 48(8), 722–731 (2001)
6. Boldi, P., Vigna, S.: Fibrations of graphs. Discrete Mathematics 243(1-3), 21–66 (2002)
7. Chalopin, J., Godard, E., Métivier, Y., Ossamy, R.: Mobile agent algorithms versus message passing algorithms. In: Shvartsman, M.M.A.A. (ed.) OPODIS 2006. LNCS, vol. 4305, pp. 187–201. Springer, Heidelberg (2006)
8. Chalopin, J., Godard, E., Métivier, Y., Tel, G.: About the termination detection in the asynchronous message passing model. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 200–211. Springer, Heidelberg (2007)
9. Chalopin, J.: Election and rendezvous with incomparable labels. Theoretical Computer Science 399(1-2), 54–70 (2008)
10. Czyzowicz, J., Labourel, A., Pelc, A.: How to meet asynchronously (almost) everywhere. In: Proc. ACM Symp. on Discrete Algorithms (SODA 2010), pp. 22–30 (2010)
11. Das, S., Flocchini, P., Nayak, A., Kutten, S., Santoro, N.: Map Construction of Unknown Graphs by Multiple Agents. Theoretical Computer Science 385(1-3), 34–48 (2007)
12. Das, S., Flocchini, P., Nayak, A., Santoro, N.: Effective elections for anonymous mobile agents. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288, pp. 732–743. Springer, Heidelberg (2006)
13. 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)
14. Klasing, R., Kosowski, A., Navarra, A.: Taking advantage of symmetries: Gathering of asynchronous oblivious robots on a ring. In: Baker, T.P., Bui, A., Tixeuil, S. (eds.) OPODIS 2008. LNCS, vol. 5401, pp. 446–462. Springer, Heidelberg (2008)
15. Lovász, L., Plummer, M.D.: Matching Theory. Annals of Discrete Mathematics, vol. 29. North-Holland, Amsterdam (1986)
16. 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)
17. Yamashita, M., Kameda, T.: Leader election problem on networks in which processor identity numbers are not distinct. IEEE Transactions on Parallel and Distributed Systems 10(9), 878–887 (1999)
18. 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)