Rendezvous of Mobile Agents in Unknown Graphs with Faulty Links

Jérémie Chalopin[1⋆], Shantanu Das[2], and Nicola Santoro[3]

[1] LaBRI Université Bordeaux 1, Talence, France,
chalopin@labri.fr,
[2] School of Information Technology and Engineering, University of Ottawa, Canada,
shantdas@site.uottawa.ca,
[3] School of Computer Science, Carleton University, Canada,
santoro@scs.carleton.ca

Abstract. A group of identical mobile agents moving asynchronously among the nodes of an anonymous network have to gather together in a single node of the graph. This problem known as the (asynchronous anonymous multi-agent) rendezvous problem has been studied extensively but only for networks that are safe or fault-free. In this paper, we consider the case when some of the edges in the network are dangerous or faulty such that any agent travelling along one of these edges would be destroyed. The objective is to minimize the number of agents that are destroyed and achieve rendezvous of all the surviving agents. We determine under what conditions this is possible and present algorithms for achieving rendezvous in such cases. Our algorithms are for arbitrary networks with an arbitrary number of dangerous channels; thus our model is a generalization of the case where all the dangerous channels lead to single node, called the *Black Hole*. We do not assume prior knowledge of the network topology; In fact, we show that knowledge of only a "tight" bound on the network size is sufficient for solving the problem, whenever it is solvable.

1 Introduction

1.1 The Problem

Consider a networked environment, modelled as a simple connected graph, in which operate a set of mobile computational entities, called agents or robots. A central problem in such systems is the so called *rendezvous* (or *gathering*) problem which requires the agents to meet together in a single node of the network. This problem has been extensively studied in the literature (see, for example, [1, 3, 10, 15, 17, 19, 23]) under a variety of models with different assumptions on the identity of the network nodes and/or of the agents (anonymous or distinct labels), the existence of timing bounds on the agents' actions (synchronous or asynchronous), the intercommunication mechanisms (whiteboards

⋆ Supported in part by grant ANR-06-SETI-015-03 awarded by the Agence Nationale de la Recherche.

or pebbles/tokens), the amount and type of memory, etc. In spite of their widely different models, the existing studies on the rendezvous problems share the common assumption that the environment where the agents operate is safe.

Unlike previous studies on the rendezvous problem, we consider the case when the environment where the rendezvous must take place, is not safe. In our model, some of the edges in the graph are harmful for the agents; specifically, any agent that attempts to traverse any such an edge (from either direction) simply disappears, without leaving any trace. The location of the unsafe links are initially unknown to the agents; we only assume that the unsafe links do not disconnect the network. Notice that if all the edges incident to a node u are unsafe, then node u can never be reached by any agent and is equivalent to a *black hole*, i.e., a node that destroys any incoming agent (e.g., [8, 9, 11–13, 18]). In other words, the black hole model is just a specific case of the model considered in this paper.

We investigate the problem in a very weak (and thus computationally difficult) setting: the network nodes do not have distinct identities (i.e., the network is anonymous), the agents are identical, and their actions (computations and movements) take a finite but otherwise unpredictable amount of time.

The only previous result for rendezvous in faulty networks was in the case of the ring network containing two unsafe links leading to a single node—the black hole [13]. Our investigation is thus a generalization of these studies to networks of arbitrary topology that contain faults at arbitrary locations.

1.2 Our Results

In this paper we provide a full characterization of the rendezvous problem of asynchronous anonymous agents in anonymous networks with unsafe links. Assuming that the safe part of the network is connected, any port on a safe node which leads to an unsafe part of the network, is called a faulty link. We present the following results in this paper:

- We first show that, if there are τ unsafe links in the network and k agents, then it is not possible, in general, for k' agents to rendezvous if $k' > k - \tau$.
- We then prove that the rendezvous of $k-\tau$ agents is deterministically possible only when the network is *covering minimal*. Even in this case, rendezvous is not possible if the agents do not know the size of the network or at least a tight upper bound. In fact, we prove that a loose upper bound $n \leq B_n \leq 2n$ is not sufficient.
- We then show that this result is tight. In fact, we present an algorithm, RDV that requires only the knowledge of a tight upper bound B on the number of nodes n, such that $n \leq B < 2n$. This algorithm allows rendezvous of $k-\tau$ agents in networks where such a rendezvous is possible; the rendezvous occurs with explicit termination for each surviving agent.
- The total number of moves made by the agents during the execution of algorithm RDV is $O(m(m+k))$ where m is the number of edges in G. We prove that this cost is optimal; in fact, we show that solving rendezvous of

$k - \tau$ agents in networks where it is solvable, requires at least $\Omega(m(m+k))$ moves, even when the network topology is known a priori.
- Finally we show that, there exists no *effective* algorithm for *maximal* rendezvous, i.e. there does not exist an algorithm that when executed on any arbitrary network achieves the rendezvous of as many agents as deterministically possible on that network.

Due to the limitations of space, the proofs of some lemmas and theorems have been omitted; These can be found in the full paper.

1.3 Related Work

The problem of *Rendezvous* has been extensively studied mostly using randomized methods (see [1] for a survey). Among deterministic solutions to rendezvous, Yu and Yung [23] and Dessmark et al. [10] presented algorithms for agents with distinct labels. In the anonymous setting, the problem has been studied under different models (synchronous or asynchronous), using either whiteboards [3] or, pebbles/tokens [19]. Some of the recent studies have focussed on minimizing the memory required by the agents for rendezvous([15, 17]).

Most of these solutions are designed for anonymous graphs (i.e. graphs where nodes do not have distinct identities) which present the most challenging (i.e. computationally difficult) situations. The issue of computability in anonymous graphs, have been studied by many authors including Angluin [2], Yamashita and Kameda [22], Mazurkiewicz [20], Sakamoto [21], and Boldi and Vigna [4]. Most of these studies have concentrated on the problem of symmetry-breaking or leader election which is in fact, closely related (and sometimes equivalent [7]) to the rendezvous problem for mobile agents. However, all the above results are restricted to safe or fault-free networks.

Recently attention has focused on designing mobile agent protocols for networks which are faulty, in particular, where there is a black hole, that is a harmful network site that destroys any visiting agent. The research on such networks have concentrated on locating the black hole. In asynchronous systems, this has been studied under two different methods—using whiteboards [11, 12] or using tokens [14] to mark edges. The objective here is minimizing the number of agents that fall into the black hole and the number of moves. In the case of synchronous agents, the objective is to minimize the time taken by the surviving agents to locate the black hole [9, 18]. The general case of multiple black holes has been considered only by Cooper et al. [8]. All these problems assume that the team of agents start from the same node, i.e. they are co-located. When the agents start from distinct nodes, it is very difficult to gather the agents while avoiding the black hole nodes. This has been studied earlier only in the case of ring networks containing a single black hole, by Dobrev et al. [13], where the authors give solutions to rendezvous and near-gathering assuming the knowledge of topology and the size of the network.

2 The Model and Definitions

2.1 The Model

The environment is modelled by the tuple $(G, \xi, p, \lambda, \eta)$ where G is an undirected connected graph, ξ is a set of agents and p specifies the initial placement of the agents in the graph G (i.e. $\forall A \in \xi, p(A) = v : v \in V(G)$). The number of nodes is denoted by $n = |V(G)|$ and the number of agents is denoted by $k = |\xi|$. The agents can move from one node to its adjacent node by traversing the edge connecting them. The edges incident to a node v are locally oriented i.e. they are labelled as $1, 2, \ldots, d(v)$, where $d(v)$ is the degree of node v. Notice that each edge $e = (u, v)$ has two labels, one for the link or port at node u and another for the link at node v. The edge labelling of the graph G is specified by $\lambda = \{\lambda_v : v \in V\}$, where for each vertex u, $\lambda_u : \{(u, v) \in E : v \in V\} \rightarrow \{1, 2, 3, \ldots, d(u)\}$ defines the labelling on its incident edges. For any edge (u, v) we use $\lambda(u, v)$ to denote the pair $(\lambda_u(u, v), \lambda_v(u, v))$.

The function $\eta : E(G) \rightarrow \{0, 1\}$ denotes which edges are safe/faulty. An edge $e \in E(G)$ is safe if $\eta(e) = 1$ and faulty otherwise. The faults are permanent, so any edge that is faulty at the start of the algorithm remains so until the end and no new faulty edge appears during the execution of the algorithm.

The node from where an agent A starts the algorithm (i.e. the initial location) is called the *homebase* of agent A. The agents are all identical (i.e. they do not have distinct names or labels) and they execute the same algorithm. An agent may enter the system at any time and at any location, and on entry, an agent immediately starts its individual execution of the algorithm. The system is totally asynchronous, such that every action performed by an agent takes a finite but otherwise unpredictable amount of time. As in previous papers on the subject, we assume that the agents communicate by reading and writing information on public whiteboards locally available at the nodes of the network. Thus, each node $v \in G$ has a whiteboard (which is a shared region of its memory) and any agent visiting node v can read or write to the whiteboard. Access to the whiteboard is restricted by fair mutual exclusion, so that, at most one agent can access the whiteboard of a node at the same time, and any requesting agent will be granted access within finite time. An agent that is granted access to the whiteboard at node v, is allowed to complete its activity at that node before relinquishing access to the whiteboard (i.e. access control is non preemptive).

Note that it is not necessary for two agents A and B traversing the same edge $e = (u, v)$ of the graph, to arrive at node v in the same order in which they left node u. However, using the whiteboards at the nodes, it is easy to implement a first-in first-out (FIFO) strategy such that agents traversing an edge can be assumed to have reached their destination in order (i.e. an agent cannot overtake another while traversing an edge). For the rest of this paper, we shall assume this FIFO property; this will simplify the description of our algorithms.

2.2 Directed Graphs and Coverings

In this section, we present some definitions and results related to directed graphs and their coverings, which we use to characterize those network where rendezvous is possible. A directed graph(digraph) $D = (V(D), A(D), s_D, t_D)$ 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_D and t_D that assign to each arc two elements of $V(D)$: a source and a target (in general, the subscripts will be omitted). A digraph D is strongly connected if for all vertices $u, v \in V(D)$, there exists a path between u and v. A *symmetric* digraph D is a digraph endowed with a symmetry, that is, an involution $Sym : A(D) \to A(D)$ such that for every $a \in A(D), s(a) = t(Sym(a))$. A bidirectional network can be represented by a strongly connected symmetric digraph, where each edge of the network is represented by a pair of symmetric arcs. In this paper, we consider digraphs where the vertices and the arcs are labelled with labels from a recursive label set L and such digraphs will be denoted by (D, μ_D), where $\mu_D: V(D) \cup A(D) \to L$ is the labelling function. In general, the label on an arc a would be a pair (x, y) and the labelling μ_D should satisfy the property that if $\mu_D(a) = (x, y)$ then $\mu_D(Sym(a)) = (y, x)$, for every arc $a \in D$.

A digraph homomorphism γ between the digraph D and the digraph D' is a mapping $\gamma: V(D) \cup A(D) \to V(D') \cup A(D')$ such that if u, v are vertices of D and a is an arc such that $u = s(a)$ and $v = t(a)$ then $\gamma(u) = s(\gamma(a))$ and $\gamma(v) = t(\gamma(a))$. A homomorphism from (D, μ_D) to (D', μ'_D) is a digraph homomorphism from D to D' which preserves the labelling, i.e., such that $\mu'_D(\gamma(x)) = \mu_D(x)$ for every $x \in V(D) \cup A(D)$.

We now define the notion of graph coverings, borrowing the terminology of Boldi and Vigna[5]. 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 vertex 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 vertex 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 $\varphi(a) = a'$.

The fibre over a vertex v' (resp. an arc a') of D' is the set $\varphi^{-1}(v')$ of vertices of D (resp. the set $\varphi^{-1}(a')$ of arcs of D).

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. A symmetric digraph D is a *symmetric covering* of a symmetric digraph D' via a homomorphism φ if D is a covering of D' via φ such that $\forall a \in A(D), \varphi(Sym(a)) = Sym(\varphi(a))$. 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'.

The notions of coverings extend to labelled digraphs in an obvious way: the homomorphisms must preserve the labelling. Given a labelled symmetric digraph (H, μ_H), the minimum base of (H, μ_H) is defined to be the labelled digraph (D, μ_D) such that (i) (H, μ_H) is a symmetric covering of (D, μ_D) and (ii) (D, μ_D) is symmetric covering minimal.

The following results on digraph coverings were proved in [5].

Property 1. Given two non-empty strongly connected digraphs D, D', each covering projection φ from D to D' is surjective; moreover, all the fibres have the same cardinality. This cardinality is called the number of sheets of the covering.

Property 2. If the digraph (H, μ_H) is a covering of (D, μ_D) via φ, then any execution of an algorithm $\mathcal{P}$ on (D, μ_D) can be lifted up to an execution on (H, μ_H), such that at the end of the execution, for any $v \in V(H)$, v would be in the same state as $\varphi(v)$.

2.3 Definitions and Properties

Given any deterministic (distributed) algorithm $\mathcal{P}$ and a network $(G, \xi, p, \lambda, \eta)$, the order in which the various actions are performed by the agents defines an *execution* of the algorithm on the network $(G, \xi, p, \lambda, \eta)$. We define the *synchronous* execution of an algorithm $\mathcal{P}$ to be the particular execution where all agents start executing at exactly the same time and every action taken by any agent takes exactly one unit of time.

We define the *extended-view* of the network $(G, \xi, p, \lambda, \eta)$ as the labelled digraph (H, μ_H) such that, H consists of two disjoint vertex sets V_1 and V_2 and a set of arcs $\mathcal{A}$ as defined below:

- $V_1 = V(G)$;
- $\mu_H(v) = |\{A \in \xi : p(A) = v\}|$, $\forall v \in V_1$;
- For every safe edge $e = (u, v) \in E(G)$, there are two arcs $a_1, a_2 \in \mathcal{A}$ such that $s(a_1) = t(a_2) = u$, $s(a_2) = t(a_1) = v$, and $\mu_H(a_1) = (\lambda_u(e), \lambda_v(e))$, $\mu_H(a_2) = (\lambda_v(e), \lambda_u(e))$.
- For every faulty edge $e = (u, v)$, there are vertices u' and $v' \in V_2$ with $\mu_H(u') = \mu_H(v') = -1$ and arcs $(u, u'), (u', u), (v, v')$ and $(v', v) \in \mathcal{A}$ with labels $(\lambda_e(u), 0)$, $(0, \lambda_e(u))$, $(\lambda_e(v), 0)$, and $(0, \lambda_e(u))$ respectively;

Here, the vertices in V_1 represent the (safe) nodes of the network and the vertices in V_2 represent (imaginary) Black-Holes. The label on a *safe* vertex v denotes the number of agents that started from the corresponding node, whereas the label on a *black-hole* vertex is always -1. Intuitively, the *extended-view* can be thought of as a canonical representation of the network.

The following results follow from the definition of the *extended-view* of a network and the Properties 1 and 2.

Lemma 1. *For any deterministic algorithm $\mathcal{P}$, a synchronous execution of $\mathcal{P}$ on the network (G, ξ, p, λ, η) is equivalent to a synchronous execution of algorithm $\mathcal{P}$ on the extended-view (H, μ_H), such that the final state of any node in G is exactly same as the state of the corresponding vertex in H.*

Lemma 2. *If the extended view of two networks have same minimum-base (D, μ_D) then all nodes in the two networks which belong to the pre-image of a vertex $v \in D$ would always be in the same state, during a synchronous execution of any algorithm $\mathcal{P}$.*