Discrete Applied Mathematics 159 (2011) 896–923

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Deterministic symmetric rendezvous with tokens in a synchronous torus[☆]

Evangelos Kranakis[a], Danny Krizanc[b], Euripides Markou[c,*]

[a] School of Computer Science, Carleton University, Ottawa, Ontario, Canada
[b] Department of Mathematics, Wesleyan University, Middletown, CT 06459, USA
[c] Department of Computer Science & Biomedical Informatics, University of Central Greece, Lamia, Greece

A R T I C L E   I N F O

Article history:
Received 18 September 2008
Received in revised form 7 August 2010
Accepted 27 January 2011
Available online 24 February 2011

Keywords:
Mobile agent
Rendezvous
Rendezvous with detection
Tokens
Torus
Synchronous

A B S T R A C T

In the rendezvous problem, the goal for two mobile agents is to meet whenever this is possible. In the rendezvous with detection problem, an additional goal for the agents is to detect the impossibility of a rendezvous (e.g., due to symmetrical initial positions of the agents) and stop. We consider the rendezvous problem with and without detection for identical anonymous mobile agents (i.e., running the same deterministic algorithm) with tokens in an anonymous synchronous torus with a sense of direction, and show that there is a striking *computational difference* between one and more tokens. Specifically, we show that (1) two agents with a constant number of unmovable tokens, or with one movable token each, cannot rendezvous in an $n \times n$ torus if they have $o(\log n)$ memory, while they can solve the rendezvous with detection problem in an $n \times m$ torus as long as they have one unmovable token and $O(\log n + \log m)$ memory; in contrast, (2) when two agents have two movable tokens each then the rendezvous problem (respectively, rendezvous with detection problem) is solvable with constant memory in an arbitrary $n \times m$ (respectively, $n \times n$) torus; and finally, (3) two agents with three movable tokens each and constant memory can solve the rendezvous with detection problem in an $n \times m$ torus. This is the first publication in the literature that studies tradeoffs between the number of tokens, memory and knowledge the agents need in order to meet in a torus.

1. Introduction

We study the following problem: how should two mobile agents move along the nodes of a network so as to ensure that they meet or rendezvous at a node or an edge?

The problem is well studied for several settings. When the nodes of the network are uniquely numbered, solving the rendezvous problem is easy (the two agents can move to a node with a specific label). However, even in that case, the agents need enough memory in order to remember and distinguish node labels. Symmetry in the rendezvous problem is usually broken by using randomized algorithms or by having the mobile agents use different deterministic algorithms. (See the surveys by Alpern [1,2], as well as the book by Alpern and Gal [4]). Yu and Yung [20] prove that the rendezvous problem cannot be solved on a general graph as long as the mobile agents use the same deterministic algorithm. While Baston and Gal [7] mark the starting points of the agents, they still rely on randomized algorithms or different deterministic algorithms

[☆] A preliminary version of this paper appeared under the title 'Mobile Agent Rendezvous in a Synchronous Torus' in the proceedings of the 7th Latin American Theoretical Informatics Symposium (LATIN' 06), March 2006, Valdivia, Chile, LNCS 3887, pp. 653–664.

[*] Corresponding author. Tel.: +30 22310 66718; fax: +30 2231066939.
E-mail addresses: kranakis@scs.carleton.ca (E. Kranakis), dkrizanc@caucus.cs.wesleyan.edu (D. Krizanc), emarkou@ucg.gr (E. Markou).

0166-218X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2011.01.020

to solve the rendezvous problem. Anderson and Fekete [5] and Alpern and Baston [3] study the problem in two-dimensional lattices, again having the mobile agents use different strategies. Chester and Tutuncu [8] study the problem in a labeled line while Howard [16] studies the rendezvous problem on the interval and the circle. Han et al. [15] improve the lower and upper bounds for the symmetric rendezvous value on the line.

Research has focused on the power, memory and knowledge the agents need to rendezvous in a network. In particular, what is the 'weakest' possible condition which makes rendezvous possible? For example, Yu and Yung [20] have considered attaching unique identifiers to the agents, while Dessmark et al. [11] added unbounded memory; note that having different identities allows each agent to execute a different algorithm. Other researchers [6,12] have given the agents the ability to leave notes in each node they visit. De Marco et al. [10] study the rendezvous problem in arbitrary asynchronous networks for two agents which have unique identifiers, and they solve the problem when an upper bound on the number of nodes of the network is known. The problem is left open when no upper bound on the number of nodes is known.

In another approach, each agent has a stationary token placed at the initial position of the agent. This model is much less powerful than having distinct identities or than having the ability to write in every node. Assuming that the agents have enough memory, the tokens can be used to break symmetries. This is the approach introduced in [18] and studied in [17,13,9,14] for the ring topology. In particular, the authors proved in [17] that two agents with one unmovable token each in a synchronous, n-node oriented ring need at least $\Omega(\log\log n)$ memory in order to make a rendezvous with detection. They also proved that if the token is movable then a rendezvous without detection is possible with constant memory.

We keep here the same model as in [9,13,14,17,18] with the exception of the underlying graph topology. Specifically, we study here the following scenario: there are two identical anonymous agents running the same deterministic algorithm in an anonymous and synchronous oriented torus. As this is one of the weakest models which has appeared in the literature, we would like to study more general topologies, and the selection of the torus is a step towards arbitrary topologies. In particular, we are interested in answering the following questions. What memory do the agents need to solve the rendezvous problem using unmovable tokens? What is the situation if they can move the tokens? What is the tradeoff between memory and the number of tokens?

1.1. Model and terminology

Our model consists of two anonymous and identical mobile agents that are placed in an anonymous, synchronous and oriented torus. The torus consists of n rings, and each of these rings consists of m nodes. Since the torus is oriented, we can say that it consists of n vertical rings. A horizontal ring of the torus consists of n nodes, while a vertical ring consists of m nodes. We call such a torus an $n \times m$ torus. The mobile agents share a common orientation of the torus; i.e., they agree on any direction (clockwise vertical or horizontal). Each mobile agent owns a number of identical tokens; i.e., all tokens are indistinguishable. A token or an agent at a given node is visible to all agents on the same node, but is not visible to any other agents. The agents follow the same deterministic algorithm, and begin execution at the same time and being at the same initial state.

At any single time unit, the mobile agent occupies a node of the torus and may (1) stay there or move to an adjacent node, (2) detect the presence of one or more tokens at the node it is occupying, and (3) release/take one or more tokens to/from the node it is occupying. We call a token *movable* if it can be moved by any mobile agent to any node of the network;, otherwise, we call the token *unmovable* in the sense that, once released, it can occupy only the node in which it has been released.

More formally, we consider a mobile agent as a finite Moore automaton[1] $\mathcal{A} = (X, Y, \mathcal{S}, \delta, \lambda, S_0)$, where $X \subseteq \mathcal{D} \times \mathcal{C}_v \times \mathcal{C}_{\mathrm{MA}}$, $Y \subseteq \mathcal{D} \times \{\texttt{drop}, \texttt{take}\}$, $\mathcal{S}$ is a set of $\sigma \geq 2$ states among which there is a specified state S_0 called the *initial* state, $\delta : \mathcal{S} \times X \to \mathcal{S}$, and $\lambda : \mathcal{S} \to Y$. $\mathcal{D}$ is the set of possible directions that an agent could follow in the torus. Since the torus is oriented, the direction port labels are globally consistent. We assume labels *up, down, left, right*. Therefore, $\mathcal{D} = \{\texttt{up}, \texttt{down}, \texttt{left}, \texttt{right}, \texttt{stay}\}$ (stay represents the situation where the agent does not move). $\mathcal{C}_v = \{\texttt{agent}, \texttt{token}, \texttt{empty}\}$ is the set of possible configurations of a node (if there is an agent and a token in a node then its configuration is agent). Finally, $\mathcal{C}_{\mathrm{MA}} = \{\texttt{token}, \texttt{no} - \texttt{token}\}$ is the set of possible configurations of the agent according to whether it carries a token or not.

Initially, the agent is at some node u_0 at the initial state $S_0 \in \mathcal{S}$. S_0 determines an action (drop token or nothing) and a direction from which the agent leaves u_0, $\lambda(S_0) \in Y$. When incoming to a node v, the behavior of the agent is as follows. It reads the direction i of the port from which it entered v, the configuration $c_v \in \mathcal{C}_v$ of node v (i.e., whether there is a token or an agent in v), and of course the configuration $c_{\mathrm{MA}} \in \mathcal{C}_{\mathrm{MA}}$ of the agent itself (i.e., whether the agent carries a token or not). The triple $(i, c_v, c_{\mathrm{MA}}) \in X$ is an input symbol that causes the transition from state S to state $S' = \delta(S, (i, c_v, c_{\mathrm{MA}}))$. S' determines an action (such as release or take a token or nothing) and a port direction $\lambda(S')$, from which the agent leaves v. The agent continues moving in this way, possibly infinitely.

We assume that the memory required by an agent is at least proportional to the number of bits required to encode its states, which we take to be $\Theta(\log(|\mathcal{S}|))$ bits. The agents know that there are two of them and they also know the number of tokens they have. Memory permitting, an agent can count the number of nodes between tokens, or the total number of nodes of the torus, etc. Since the agents are identical, they face the same limitations on their knowledge of the network.

[1] The first known algorithm designed for graph exploration by a mobile agent, modeled as a finite automaton, was introduced by Shannon [19] in 1951.