HAL
open science

Analysis of Deterministic Tracking of Multiple Objects using a Binary Sensor Network

Yann Busnel, Leonardo Querzoni, Roberto Baldoni, Marin Bertier, Anne-Marie Kermarrec

To cite this version:

Yann Busnel, Leonardo Querzoni, Roberto Baldoni, Marin Bertier, Anne-Marie Kermarrec. Analysis of Deterministic Tracking of Multiple Objects using a Binary Sensor Network. ACM Transactions on Sensor Networks, 2011, 8, pp.0. inria-00590873

HAL Id: inria-00590873

https://inria.hal.science/inria-00590873

Submitted on 5 May 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Analysis of Deterministic Tracking of Multiple Objects using a Binary Sensor Network

YANN BUSNEL
LINA / Université de Nantes – France
LEONARDO QUERZONI and ROBERTO BALDONI
MIDLAB / Università di Roma "La Sapienza" – Italy
MARIN BERTIER
IRISA / INSA Rennes – France
and
ANNE-MARIE KERMARREC
INRIA Rennes - Bretagne Atlantique – France

Let consider a set of anonymous moving objects to be tracked in a binary sensor network. This paper studies the problem of associating deterministically a track revealed by the sensor network with the trajectory of an unique anonymous object, namely the *Multiple Object Tracking and Identification* (MOTI) problem. In our model, the network is represented by a sparse connected graph where each vertex represents a binary sensor and there is an edge between two sensors if an object can pass from one sensed region to another one without activating any other sensor. The difficulty of MOTI lies in the fact that the trajectories of two or more objects can be so close that the corresponding tracks on the sensor network can no longer be distinguished (track merging), thus confusing the deterministic association between an object trajectory and a track.

The paper presents several results. We first show that MOTI cannot be solved on a general graph of ideal binary sensors even by an omniscient external observer if all the objects can freely move on the graph. Then, we describe some restrictions that can be imposed *a priori* either on the graph, on the object movements or both, to make the MOTI problem always solvable. In the case of absence of an omniscient observer, we show how our results can lead to the definition of distributed algorithms that are able to detect when the system is in a state where MOTI becomes unsolvable.

Categories and Subject Descriptors: F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Binary Sensor Network, Distributed algorithm, Impossibility and Characterization, Passive Trajectory Tracking

1. INTRODUCTION

Context and motivations. "*Tracking the movements of anonymous objects and associating each object trajectory with an unique identifier*" is a basic problem in many applicative contexts such as surveillance [Oh and Sastry 2005], rescue, traffic monitoring [Bloisi et al. 2007], pursuit evasion games, *etc.* Tracking objects with a sensor network is a challenging task. Potential inaccuracy of sensors (*e.g.* [Aslam et al. 2003; Lazos et al. 2009]) and the complexity of the localization subsystem computability (*e.g.* [Aspnes et al. 2004]) significantly complicate tracking initiation, maintenance and termination of object trajectories leading to false detection and missing observations.

Even without considering false detections and missing observations, the problem of associating an unique identifier to a track corresponding to the trajectory of one of many objects is difficult due to the potential merging of tracks. Tracking a unique object moving among an ideal setting could appear trivial but following many anonymous objects becomes challenging, owing to that merging tracks may confuse the latter association if sensors do not have adequate capabilities [Liu et al. 2003]. In other words, two tracks may become so close to each other that they become indistinguishable, and then impossible to identify after splitting again. Deciding, after a track merging, what is a relevant association among multiple hypothesis is usually achieved by looking at the behaviors of tracks before the merging happened [Liu et al. 2003; Kulathumani et al. 2009]. In the following, we refer to this one-to-one mapping between tracks and object trajectories as the *Multiple Object Tracking and Identification* (MOTI) problem.

We investigate MOTI by using a binary sensor network, which means that each sensor reports only a binary value indicating if an object is in its sensing region or not. Focusing on such minimalist and simple devices is motivated by the fact that we want to study the essence of MOTI's solvability without looking at the powerfulness of sensors capabilities. As in [Oh and Sastry 2005], in our model, the sensor network is represented by a sparse connected graph, namely the *passage connectivity graph* (PCG). Each vertex in this graph represents a binary sensor and there is an edge between two sensors u and v only if an object can pass from the sensing region of u to the one of v without activating any other remaining sensor.

Contributions. In this paper, we show that it is impossible to solve the MOTI problem in a generic graph. To make the impossibility as strong as possible, the impossibility is proved considering an omniscient observer that has a complete knowledge of the state of the graph and under ideal conditions for object tracking such as perfect binary sensors (*i.e.* no false detections or missing observations), ideal coverage of the sensing areas (*i.e.* disjoint sensing regions), so that at each instant of time one object activates only one sensor and each sensor is activated by at most one object.

We prove that the impossibility of solving MOTI is a structural problem that depends on the topology of the underlying PCG and on the number of moving objects. Then, we describe some restrictions that can be imposed *a priori* either on the graph, on the object movements, or on both, to make MOTI always solvable. More specifically, we show that if the passage connectivity graph is acyclic, MOTI can always be solved. Also, if the graph contains cycles with a length greater than ℓ, we prove that MOTI can be solved only if the maximum number of objects that can move concurrently is less than $\lceil \frac{\ell}{2} \rceil$.

Finally, we propose a characterization of MOTI solvability based on the state of sensors belonging to cycles of the PCG. The interest of this characterization is twofold: *(i)* it leads to a computationally light algorithm run by the observer ensuring MOTI solvability and *(ii)* this characterization brings to the realization of distributed algorithms run by each sensor, thus removing the need of the omniscient observer. From this point of view, the choice of modeling the system with a PCG and reasoning on it is functional to the design of these solution as it let us easily translate theoretical results in algorithms.

Roadmap. Section 2 discusses some related works and Section 3 introduces the system model. Section 4 defines the Multiple Object Tracking and Identification (MOTI) problem and shows that this problem is impossible to solve in our setting. Section 5 gives

two characterizations of MOTI solvability, which are used in Section 6 to introduce some promising classes of systems where MOTI can always be solved. However, previous characterizations are expensive to compute. Then, Section 7 presents a MOTI characterization involving only cycles in PCG. The algorithms derived by the latter MOTI characterization are presented in Section 8 (a centralized algorithm runs by the omniscient observer) and 9 (a fully distributed algorithm runs by each sensor) respectively. Finally, Section 10 concludes the paper.

2. RELATED WORK

Tracking mobile objects through sensors is useful in a large spectrum of applications [Han et al. 2004]. It is treated in the literature from various perspectives, but most of the works are concerned with the problem of correctly tracking one [Aslam et al. 2003] or more [Singh et al. 2007] objects in a network of binary sensors considering noise and false detections under various constraints (*e.g.* noise levels, power consumption [Gui and Mohapatra 2004], limited computational or network resources [Liu et al. 2003], *etc.*). The problem of associating measurements from sensors to the appropriate tracks, especially when missing reports, unknown targets and false reports are present, has been tackled in the past [Reid 1979] using approaches for extracting the most probable hypothesis from a set containing multiple hypotheses all compatible with the actual observations.

In our work, we take a more theoretical approach to the problem considering a setting characterized by a network of *ideal* sensors (no noise, no false detection, no limitation on communication) and show that, in the general case, it is impossible to correctly associate sensed tracks to moving objects. This result is a consequence of the possibility of track merging and splitting during the observation period [Liu et al. 2003]: once two tracks are merged due to the excessive proximity of two objects, it is impossible to deterministically maintain the identity of the two objects as soon as their tracks split.

Issues related to track merging and splitting are common to every setting where two or more objects can move freely. However, in order to provide the reader with proofs of our statements, in this paper, we limit the problem analysis to a specific environment where object movements are partially constrained, and that can be abstracted as a *passage connectivity graph* [Oh and Sastry 2005]. Note that, beside the simplicity of this model, it still perfectly maps indoor applications like tracking the movements of people inside a building. Also our model includes passive object tracking [Mao et al. 2009; Viani et al. 2009]. In such tracking, the object is assumed to be *clean* (and thus anonymous), *i.e.*, there is no any equipment carried by the target and the tracking procedure is considered to be passive.

Recently, Crespi, Cybenko and Jiang [Crespi et al. 2008] proposed a theoretical analysis of trackability. They modeled the evolution of a nondeterministic automata whose state transitions can be observed. They explored the issue of how fast the number of possible tracks compatible with the observations grows. Their work shows that the worst case growth speed is either polynomial or exponential indicating a sort of phase transition in tracking accuracy for different settings. The problem of deciding if objects moving concurrently in an environment can be accurately tracked though a binary sensor network (i.e. the main problem attacked in this paper) can be seen as a special case of the problem investigated in [Crespi et al. 2008]: it should be checked if the set of hypothetical state evolutions that can be associated to the sequence of observations done through the sensors

is always constituted by a single hypothesis; if this is the case, the MOTI problem can be solved. However, investigating this simplified scenario characterized by strong assumptions (i.e. perfect sensing devices, partially constrained movements, etc.) has a specific interest on its own, because it let us define constraints and algorithms (both centralized and distributed) that can be used a starting point to design and deploy systems where moving object can be deterministically tracked.

Last but not least, the study of tracking objects on binary sensors networks has gained a lot of attention in the very recent years. As an example, [Wang et al. 2008] studies the problem of object tracking in the presence of imperfect binary sensing and [Shrivastava et al. 2009] explores the limit of spatial resolution that can be achieved in tracking an objects within a two-dimensional field of binary proximity sensors.

3. SYSTEM MODEL

We consider a system composed of a set of generic objects moving in an environment where sensors can detect their presence. Such an environment can be modeled as a *Passage Connectivity Graph* $PCG(V, E)$ where the set of vertices V represents binary sensors; it exists an edge $e_{i,j} \in E$ linking two vertices $v_i, v_j \in V$ if and only if (*iff*) it is possible for an object to move from the position where it is detected by sensor v_i to the one where it is detected by sensor v_j without activating, during the movement, any other sensor. We assume that, in the considered environment, movements are always possible in both directions. Therefore, the PCG is undirected. Moreover, we assume that there is always one single way to move between two positions: only an edge can connect two distinct vertices. For each vertex v_i, a special edge $e_{i,i}$ exists in E representing the possibility for an object to remain in the same position.

The set of moving objects $\{o_1, \ldots, o_x\}$ is denoted by $\mathbb{O}$. Time is represented as a discrete and infinite sequence $T = [t_0, t_1, \ldots]$ of instants starting at t_0. At every time instant, each object o occupies a position represented by a vertex $v_i \in V$. The position of an object o at a specific time instant t is given by the function $loc : T \times \mathbb{O} \rightarrow V$ that returns the corresponding vertex v_i. When an object, positioned on a vertex v_i, decides to move to a new position, it can choose as its destination any vertex v_j such that $e_{i,j} \in E$; the set of possible destinations is returned by the function $adj : \mathbb{G} \times V \rightarrow \mathcal{P}(V)$, where $\mathbb{G}$ is the set of all graphs. The movement of an object o at a specific time instant t is given by the function $mov : T \times \mathbb{O} \rightarrow E$ that returns the edge $e_{i,j}$ used for the movement where $v_i = loc_t(o)$ and $v_j \in adj_{PCG}(v_i)$.

We assume that, regardless of nodes positions and movements, the two following conditions always hold in our system: $\forall t \in T, \forall o_i, o_j \in \mathbb{O}$, *(i)* $loc_t(o_i) \neq loc_t(o_j)$ (*i.e.* two distinct objects cannot be located on the same vertex at the same time) and *(ii)* $mov_t(o_i) \neq mov_t(o_j)$ (*i.e.* two distinct objects cannot move on the same edge at the same time). Without these constraints, obviously, the problem is trivially impossible. Note that making these assumptions hold in a real environment can be rather difficult due to the wide area covered by some sensors or to the impossibility of constraining the number of objects moving between the areas covered by different sensors; however, these consideration strengthen the results of impossibility showed in the next section.

Each object $o \in \mathbb{O}$ moves in the system describing a trajectory. The *trajectory* described by object o between $t_i, t_j \in T$ (with $j > i$) is defined as $P_{t_i,t_j,o} = [v^{t_i}, \ldots, v^{t_j}]$ where $v^{t_k} = loc_{t_k}(o), k \in [i, j]$. For instance, Figure 1 shows 3 objects moving on a simple

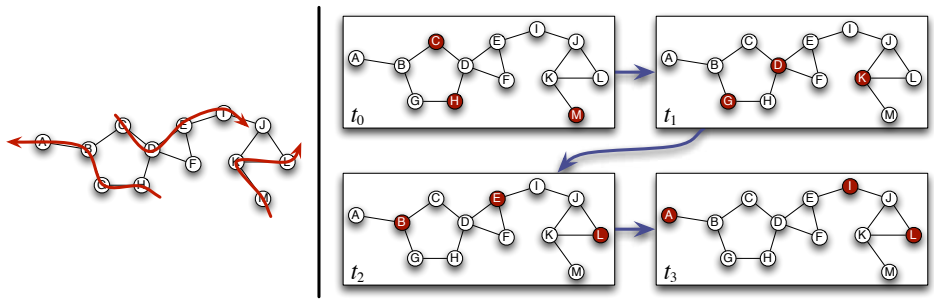

Fig. 1. Example of real and observed trajectories of 3 object o_1, o_2, o_3 in the interval $[t_0, t_3]$: $P_{t_0,t_3,o_1} = [C, D, E, I]$; $P_{t_0,t_3,o_2} = [H, G, B, A]$; $P_{t_0,t_3,o_3} = [M, K, L, L]$.

PCG. Their trajectories are presented in the caption. Note that an object can remain at the same place for a certain period of time, as o_3 between t_2 and t_3.

Given t_i and t_j, we define the *global trajectory* as the set containing all object trajectories. The global trajectory is denoted as $P_{t_i,t_j} = \{P_{t_i,t_j,o}\}_{o \in \mathbb{O}}$ or simply P if the time interval is precisely defined by the context. A specific global trajectory P_{t_i,t_j} is an element of $\mathbb{P}_{t_i,t_j}$, the set containing all possible global trajectories described by the objects in $\mathbb{O}$ on PCG during the period $[t_i, t_j]$.

3.1 System State

The state of the system at time t is described by the state of each sensor (object detected or not) at that time. This state is represented as a vector of boolean values, one for each vertex in PCG.

Definition 3.1 *State Vector*. A state vector at time t, denoted S_t, is a vector of size $|V|$ where:

$$\forall v \in V, S_t[v] = \begin{cases} 1 & \text{if } \exists\, o \in \mathbb{O} : loc_t(o) = v \\ 0 & \text{otherwise} \end{cases}$$

For example, in Figure 1, we have

$$S_{t_0} = [\ \underset{\substack{\uparrow \\ A}}{0}\ ,\ \underset{\substack{\uparrow \\ B}}{0}\ ,\ \underset{\substack{\uparrow \\ C}}{1}\ ,\ \underset{\substack{\uparrow \\ D}}{0}\ ,\ \underset{\substack{\uparrow \\ E}}{0}\ ,\ \underset{\substack{\uparrow \\ F}}{0}\ ,\ \underset{\substack{\uparrow \\ G}}{0}\ ,\ \underset{\substack{\uparrow \\ H}}{1}\ ,\ \underset{\substack{\uparrow \\ I}}{0}\ ,\ \underset{\substack{\uparrow \\ J}}{0}\ ,\ \underset{\substack{\uparrow \\ K}}{0}\ ,\ \underset{\substack{\uparrow \\ L}}{0}\ ,\ \underset{\substack{\uparrow \\ M}}{1}\ ].$$

In the following, we denote as $\mathbb{S}$ the set of all possible state vectors with respect to x objects in a given PCG. We obviously have $\binom{|V|}{x}$ different state vectors.

3.2 The Observer

The system has an observer that is able to read, at any time t, the state vector S_t. The aim of the observer is to identify objects and trace their trajectories over time. Given a state vector S_t, the observer must identify a function tag to assign a unique identifier $\overline{o} \in \overline{\mathbb{O}}$ to each vertex v such that $S_t[v] = 1$ (and a predefined value $\perp$ to all the other vertices), with the only constraint that two vertices cannot share a same identifier (with the exception of $\perp$). For instance, in Figure 1, as $S_{t_2} = [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0]$ and $\overline{\mathbb{O}} \subset \mathbb{N}$, we

can have $tag(S_{t_2}) = [\perp, 3, \perp, \perp, 2, \perp, \perp, \perp, \perp, \perp, \perp, 1, \perp]$. Each identifier represents an object identified by the observer. For the sake of convenience, we introduce the function $\overline{loc} : T \times \overline{\mathbb{O}} \to V$ that returns the vertex v that was tagged by $\overline{o} \in \overline{\mathbb{O}}$ at time $t \in T$.

Given an object identified by the observer, and its corresponding tag $\overline{o}$, we can define the *observed trajectory* between two time instants t_i, t_j as $\overline{P}_{t_i,t_j,\overline{o}} = [v^{t_i}, \ldots, v^{t_j}]$ where $v^{t_k} = \overline{loc}_{t_k}(\overline{o}), k \in [i, j]$. Effectively, we have to introduce the *observed global trajectory* $\overline{P}_{t_i,t_j} = \left\{\overline{P}_{t_i,t_j,\overline{o}}\right\}_{\overline{o}\in\overline{\mathbb{O}}}$, corresponding to the set of all observed trajectories perceived by the observer. Obviously, $\overline{P}_{t_i,t_j}$ belongs to $\mathbb{P}_{t_i,t_j}$.

4. THE PROBLEM OF IDENTIFYING OBJECTS AND TRACKING THEIR TRAJECTORIES

4.1 The MOTI Problem

Let us consider an interval of time $[t_i, t_j]$. The *Multiple Object Tracking and Identification* (MOTI) problem consists in defining a function tag such that the following condition holds: $\forall o \in \mathbb{O},\ \exists \overline{o} \in \overline{\mathbb{O}} : P_{t_i,t_j,o} = \overline{P}_{t_i,t_j,\overline{o}}$. Globally speaking, we can simply define MOTI as the following condition:

$$P_{t_i,t_j} = \overline{P}_{t_i,t_j}$$

That means that the set of observed trajectories is exactly the same that the real ones. Given that trajectories (both real and observed) are a consequence of object locations, at a finer level of granularity, we have:

$$\text{MOTI is solved iff } \forall o \in \mathbb{O},\ \exists \overline{o} \in \overline{\mathbb{O}},\ \forall t \in [t_i, t_j] : \overline{loc}_t(\overline{o}) = loc_t(o)$$

The difficulty of this problem comes from the fact that there could be situations in which the observer is not able to distinguish the trajectories of two or more objects. It is obvious that if $|\mathbb{O}| = 1$, the problem is always solvable and trivial.

4.2 An impossibility result

In this section we show how the MOTI problem cannot be solved in the general case. The proof consists in a first theorem that states the conditions under which MOTI is not solvable, and second one where we prove that there exists at least one scenario where these conditions hold.

This first theorem formalizes an obvious intuition: if the observer can deduce from a single set of observations two different trajectories for a same object, it is impossible to deterministically decide which one is correct.

THEOREM 4.1 MOTI UNSOLVABILITY. *Given an interval of time $[t_i, t_j]$, a $PCG(V, E)$, a set $\mathbb{O}$ of x objects, MOTI cannot be solved iff*

$$\forall tag, \exists P, P' \in \mathbb{P}_{t_i,t_j} :\ P \neq P' \wedge \overline{P} = \overline{P'}.$$

where $\overline{P}$ and $\overline{P'}$ are obtained by the observer using any function tag *from real global trajectories P and P'.*

PROOF. Let us consider an interval of time $[t_i, t_j]$, a $PCG(V, E)$ and a set $\mathbb{O}$ of x objects.

—Given any tag function, assume that (1) $\exists P, P' \in \mathbb{P}_{t_i,t_j} : P \neq P' \wedge \overline{P} = \overline{P'}$ and (2) MOTI can be solved. Given that MOTI can be solved we have $P = \overline{P}$ and that $P' = \overline{P'}$ (*cf.* MOTI definition in section 4). But, we have $P \neq P'$ and $\overline{P} = \overline{P'}$. Therefore, there is a contradiction.

—Let prove now that if MOTI cannot be solved $\Longrightarrow \forall tag, \exists P, P' \in \mathbb{P}_{t_i,t_j} : P \neq P' \wedge \overline{P} = \overline{P'}$. Let prove the contrapositive (i.e. *modus tollens*) of this proposition.
Consider $\exists tag, \forall P, P' \in \mathbb{P}_{t_i,t_j} : P \neq P' \Longrightarrow \overline{P} \neq \overline{P'}$ $(P = P' \vee \overline{P} \neq \overline{P'})$.
Let us call $map : \mathbb{P}_{t_i,t_j} \rightarrow \mathbb{P}_{t_i,t_j}$ a function which associates the real trajectory to the observed one. Each bijective map function verifies the following expression: $\forall P, P' \in \mathbb{P}_{t_i,t_j} : P \neq P' \Longrightarrow map(P) \neq map(P') \Longrightarrow \overline{P} \neq \overline{P'}$ as $map(\cdot)$ corresponds to the observed trajectories. Let map be the identity function. We have: $\forall P \in \mathbb{P}_{t_i,t_j} : P = map(P) = \overline{P}$. Then, by definition of Section 4, MOTI can be solved using this specific tag function. Thus, by contrapositive, we have:
MOTI cannot be solved $\Longrightarrow \forall tag, \exists P, P' \in \mathbb{P}_{t_i,t_j} : P \neq P' \wedge \overline{P} = \overline{P'}$.

We then obtain the equivalence, *i.e.* a characterization of the MOTI unsolvability. □

COROLLARY 4.2 MOTI SOLVABILITY. *Given an interval of time* $[t_i, t_j]$, *a* $PCG(V, E)$, *a set* $\mathbb{O}$ *of* x *objects and a* tag *function. MOTI can be solved if, and only if,*

$$\forall P, P' \in \mathbb{P}_{t_i,t_j} : P \neq P' \Longrightarrow \overline{P} \neq \overline{P'}.$$

Consequently, *it is impossible to solve MOTI* if, and only if, there exist at least two global trajectories which respect the condition of Theorem 4.1. Therefore, we can state that:

THEOREM 4.3 MOTI IMPOSSIBILITY. *Given the system model presented in Section 3, MOTI cannot be solved.*

PROOF. Consider a 4 vertex PCG with a 4 edge loop around its vertices, as shown in Figure 2.*a*. Consider two objects moving in this PCG and a time interval constituted by two consecutive time instants $[t_0, t_1]$ with $t_1 = t_0 + 1$. Consider the two global trajectories $P, P' \in \mathbb{P}_{t_0,t_1}$, presented in Figure 2.*b* and 2.*c*. They are defined as:

$$\left\{ \begin{array}{l} P = \left\{ \begin{array}{l} [A \ , \ B] \\ [D \ , \ C] \end{array} \right\} \\ P' = \left\{ \begin{array}{l} [A \ , \ C] \\ [D \ , \ B] \end{array} \right\} \end{array} \right.$$

Obviously, we have $P \neq P'$.

According to the system model and the observer capabilities, the information known by the observer for both case P and P', is presented in Figure 2.*d*. We assume that t_0 is the initial time. Then, the observer does not have any other information available about the system than the two following state vectors (which are the same for both P and P'):

$$S_{t_0} = \left[\, 1, 0, 0, 1 \,\right] \text{ and } S_{t_1} = \left[\, 0, 1, 1, 0 \,\right]$$

As the relation tag is a function (a single output for a given input element), it exists a unique tagging for a state, according to the corresponding state vector and previous state available. Thus, assume that:

$$tag(S_{t_0}) = \left[\, 1, \perp, \perp, 2 \,\right].$$

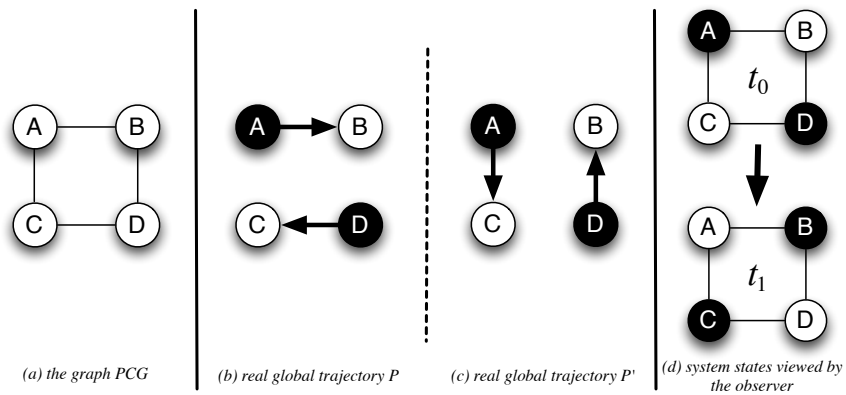

Fig. 2. A simple example in which the observer cannot track precisely the trajectories of objects: For a PCG presented in (a) with two objects, two global trajectories P and P', presented in (b) and (c) respectively, can occur and the observer is not able to distinguish them with only the information presented on (d) if objects have moved following trajectory P or P'.

According to $tag(S_{t_0})$ and S_{t_1}, only two values are possible for $tag(S_{t_1})$:

$$(1)\ tag(S_{t_1}) = \left[\ \perp, 1, 2, \perp\ \right] \text{ or } (2)\ tag(S_{t_1}) = \left[\ \perp, 2, 1, \perp\ \right].$$

For each of these cases, $tag(S_{t_1})$ is the same for the observation of P and P'. So, given a function tag, $\overline{P} = \overline{P'}$. Then, due to Theorem 4.1, MOTI cannot be solved. □

In order to better understand this result, let's take a closer look at each case:

Case (1). In this case, the observed global trajectory computed by the observer is:

$$\overline{P} = \overline{P'} = \left\{ \begin{array}{l} [A\ ,\ B] \\ [D\ ,\ C] \end{array} \right\}.$$

Then, we have $P = \overline{P} = \overline{P'} \neq P'$ and MOTI is not solvable.

Case (2). By symmetry, in this case, the observed global trajectory computed by the observer is:

$$\overline{P} = \overline{P'} = \left\{ \begin{array}{l} [A\ ,\ C] \\ [D\ ,\ B] \end{array} \right\}.$$

Then, we have $P' = \overline{P'} = \overline{P} \neq P$ and MOTI is not solvable.

So, it exists at least one case in which MOTI cannot be solved.

5. MOTI SOLVABILITY

We have proved that, in the general case, MOTI cannot be solved. Yet, we can identify some constraints that can be added to the system in order to solve it. Before delving into the details about how the system model can be constrained to make MOTI solvable, we need to introduce additional notations.

5.1 Characterization of safe/unsafe states and movements

Given an object o and the trajectory it describes on the PCG as time goes, we can identify every single movement.

Definition 5.1 *Movement*. Let $o \in \mathbb{O}$ be an object moving in the system represented by $PCG(V, E)$. For each time instant $t \in T$, we define its movement as $m_{t,o} = P_{t,t+1,o}$.

Definition 5.2 *Movement Set*. Consider a system represented by $PCG(V, E)$ where objects belonging to $\mathbb{O}$ can move. For each time instant $t \in T$, we define the movement set as $M_t = P_{t,t+1}$.

The movement set is defined, for each time instant t, as the set of all movements done by objects; therefore, it represents how the system "evolves" right after the specific time t.

If we now consider the system state at a specific time t, we can identify all the possible movements that objects are able to do. Each possible combination of these movements corresponds to a different movement set. All these movement sets are defined on the basis of the position of objects on the graph, *i.e.* given a state vector, we can define all the possible movements. From this point of view, movement sets are not necessarily tied to time, as they can be considered as the sets of *possible* movements that objects can do if, at a certain time, they are located on a specific subset of vertices. This idea leads us to the definition of the *State Graph*, which is a graph representing possible system states (in terms of state vectors) and possible movements sets linking them.

Definition 5.3 *State Graph*. Let $PCG(V, E)$ be a graph representing the environment where x objects can move. The corresponding state graph is defined as $SG(\mathbb{S}, \mathbb{M})$, where $\mathbb{S}$ is the set of all possible state vectors and $\mathbb{M}$ is the set of all possible movement sets.

Figure 3 shows an instance of a state graph, depicted on the right side, when considering two moving objects and the 4 vertex PCG introduced above, depicted on the left side. In this example, each arrow corresponds to a *possible* movement. For instance, from S_5 to S_4, we have only one possible movement :

$$M = \{[B, D]; [D, C]\}.$$

Now we can define which edges and which vertices of the state graph should be considered as *unsafe* with respect to the solvability of MOTI.

Definition 5.4 *Unsafe Movement*. Consider a system represented by $PCG(V, E)$ where objects belonging to $\mathbb{O}$ can move, and the corresponding state graph $SG(\mathbb{S}, \mathbb{M})$. Consider two states $S, S' \in \mathbb{S}$ such that it exists a movement set $M \in \mathbb{M}$ that links these two states ($S \xrightarrow{M} S'$). M is *unsafe* iff $\exists M' \in \mathbb{M}$ such that $S \xrightarrow{M'} S'$ and $M \neq M'$.

We consider unsafe all movement sets linking two system states that are linked by at least two movement sets. The idea behind this definition is that an observer can not distinguish which movement set has really occurred between all the possible unsafe movement sets linking the same system states (because the trajectories it observes are only built using system states). The presence of unsafe movements in a state graph makes MOTI problem impossible to solve.

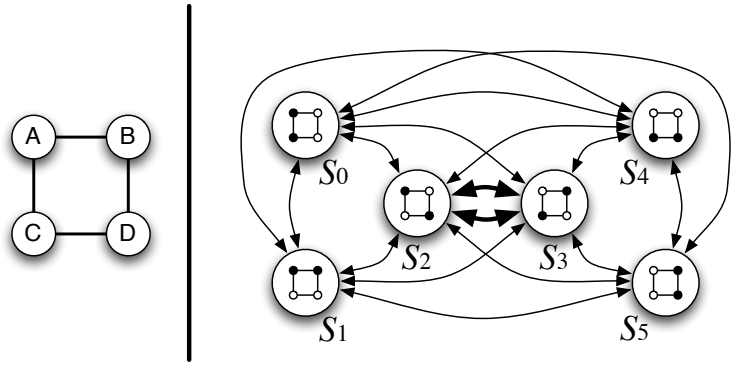

Fig. 3. Example of a state graph for the left 4 vertex PCG including 2 objects.

Similarly, we can define what an unsafe state is.

Definition 5.5 *Unsafe state*. Consider a system represented by $PCG(V, E)$ where objects belonging to $\mathbb{O}$ can move, and the corresponding state graph $SG(\mathbb{S}, \mathbb{M})$.
A state $S \in \mathbb{S}$ is *unsafe* iff $\exists M \in \mathbb{M}, \exists S' \in \mathbb{S}$ such that $S \xrightarrow{M} S'$ and M is unsafe.

Considering again the state graph depicted in Figure 3, states S_2 and S_3 are both unsafe because there are two distinct edges (two unsafe movement sets) linking them (bold arrows). All the other states (and, by the way, all the other movements sets) are safe.

5.2 Characterizing MOTI solvability

On the basis of these definitions, we revise Theorem 4.2 and propose two different definitions of MOTI solvability. First, we assume that the global trajectory is known (*e.g.* when analyzing the behaviour of the system *a posteriori*).

THEOREM 5.6 P-SOLVABILITY. *Let $SG(\mathbb{S}, \mathbb{M})$ the state graph defined over a $PCG(V, E)$ representing the system where objects in $\mathbb{O}$ can move. Consider a specific trajectory $P \in \mathbb{P}_{t_i,t_j}$: MOTI can be solved iff $\forall t \in [t_i, t_{j-1}],\ M_t$ is safe.*

PROOF. Consider a specific trajectory $P \in \mathbb{P}_{t_i,t_j}$.

—Assume that $\exists t \in [t_i, t_{j-1}]$ such that M_t is *unsafe*. Let us call $P^* = P_{t,t+1}$ the sub-trajectory of the considered P_{t_i,t_j} such that $M_t = P^*$. Due to Definition 5.4, it exists at least one different unsafe movement M' in $\mathbb{M}$ between the two same system states S, S' such that $M' = P'^*$. Obviously, $P^* \neq P'^*$ but they share the same initial and final state vectors S_t and S_{t+1}. Given that the relation tag available to the observer is a function, it exists a unique tagging for these states. Then, we have:

$$\exists P^*, P'^* \in \mathbb{P}_{t,t+1} : P^* \neq P'^* \wedge \overline{P^*} = \overline{P'^*}.$$

Therefore, due to Theorem 4.1, MOTI is unsolvable for P^* and P'^*. Therefore, given that P^* is a sub-trajectory of P_{t_i,t_j}, MOTI is unsolvable for P_{t_i,t_j}.

—Assume here that $\forall t \in [t_i, t_{j-1}],\ M_t$ is *safe*. Then, we have: $\forall t \in [t_i, t_{j-1}], \nexists\, M \neq M_t$ such that $S_t \xrightarrow{M} S_{t+1}$. Then, $\forall t \in [t_i, t_{j-1}], \exists! P \in \mathbb{P}_{t,t+1}$ from S_t to S_{t+1} and, as

it is unique, this P is the sub-trajectory between time t and $t+1$ of the considered P_{t_i,t_j}. So, $\forall P' \in \mathbb{P}_{t,t+1}$ such that $P \neq P'$, the initial (or respectively the final) system state of P' is not equal to the initial (or respectively the final) system state of P. Then, consider a bijection function map as introduced in the proof of Theorem 4.1. We have: $\forall P' \in \mathbb{P}_{t,t+1} : P \neq P' \Longrightarrow map(P) \neq map(P')$. If map is the identity function, we have: $\forall P' \in \mathbb{P}_{t,t+1} : P \neq P' \Longrightarrow \overline{P} \neq \overline{P'}$.

Therefore, due to Corollary 4.2 MOTI can be solved. □

Even though characterizing MOTI solvability with respect to a specific global trajectory is useful for all those systems where we want to decide at any point of time if MOTI is solvable or not, it is also possible to define a set of cases where MOTI is always solvable. This new characterization generalizes Theorem 5.6 with all possible trajectories that can occur in the system:

THEOREM 5.7 $\mathbb{P}$-SOLVABILITY. *Let $SG(\mathbb{S},\mathbb{M})$ the state graph defined over a graph $PCG(V,E)$ representing the system where objects in $\mathbb{O}$ can move.*
$\forall P \in \mathbb{P}_{t_i,t_j}$: *MOTI can be solved iff $\forall S \in \mathbb{S},\ S$ is safe.*

PROOF. Assume that $\forall P \in \mathbb{P}_{t_i,t_j}$ MOTI can be solved. Then, $\forall P \in \mathbb{P}_{t_i,t_j}, \forall t \in [t_i, t_{j-1}]$: M_t is safe (Theorem 5.6) and $\forall P \in \mathbb{P}_{t_i,t_j}, \forall t \in [t_i, t_{j-1}]$: S_t is safe (Definition 5.5). Given that each system state can occur as an initial state of a trajectory, we have:

$$\forall S \in \mathbb{S}, \exists P \in \mathbb{P}_{t_i,t_j} : S_{t_i} = S.$$

Then, given that all system states are safe for all possible trajectories, we have that $\forall S \in \mathbb{S},\ S$ is safe.

Assume now that $\forall S \in \mathbb{S},\ S$ is safe. Due to Definition 5.5, we have that $\forall M \in \mathbb{M}, M$ is safe. It follows that $\forall P \in \mathbb{P}_{t_i,t_j}, \forall t \in [t_i, t_{j-1}], M$ is safe. Therefore due to Theorem 5.6, MOTI can be solved. □

We then obtain two definitions of MOTI solvability, according to the granularity required by the context. The latter can be very different depending to the knowledge of the designer and the aim of the observation (a priori or online treatment).

Both are used in the following for characterizing MOTI solvability with a specific or generic point of view.

6. A SUFFICIENT CONDITION FOR MAKING MOTI $\mathbb{P}$-SOLVABLE

In this section, we show how MOTI can be solved by adding some specific constraints on the system model. More specifically, we show how the problem becomes easily solvable if we assume that object movements are limited in some way.

As we previously explained, the state graph associated to a system may present one or more unsafe states that make MOTI unsolvable. One way to avoid the presence of unsafe states in the state graph, is to remove some of the unsafe movements sets such that the remaining movements sets are all safe. From a practical point of view, this means limiting object movements in the environment, modifying the environment itself or constraining their actions. Let think about a traffic control application where vehicles can move from one road to another through barriers, this can be realized simply by closing temporary some barriers in case of huge growth of traffic density. Therefore, that means deleting the

corresponding edges between two vertices of PCG and modifying accordingly the state graph.

Basically, the only cause of the MOTI non solvability in a system is the presence of cycles in the graph (*cf.* Corollary 6.2): when two or more objects move inside a cycle, it might be impossible to distinguish their trajectory based on the sole observation of the state vectors. This problem can be avoided by limiting the number of objects in $\mathbb{O}$ that can move concurrently below a specific constant k, which strictly depends from some characteristics of the PCG. Note that if $k = 1$, MOTI is trivially always solvable because this scenario is equivalent to the one where a single moving object is present. In this case, we can guarantee that MOTI is solvable for all the possible trajectories P such that at most k objects move concurrently at each time unit. The set of all these trajectories is a subset of $\mathbb{P}$ and will be denoted as $\mathbb{P}^k$.

THEOREM 6.1. *Let $k \leq |\mathbb{O}|$ be the maximum number of objects that can move concurrently, $\ell > 1$ be an integer and $PCG(V, E)$ be a graph which does not contain cycles of length $1 < l < \ell$. $\forall P \in \mathbb{P}^k$: MOTI is P-solvable iff $\lceil \frac{\ell}{2} \rceil > k$.*

PROOF. We first prove that if $k < \lceil \frac{\ell}{2} \rceil \Rightarrow \forall P \in \mathbb{P}^k$, P does not contain any unsafe movement. The proof is done by induction on the number n of objects moving concurrently in the system at each step. In the following, given S and S' two state vectors, we refer to *diff*(S, S') to denote the number of vertices in PCG whose state changes between S and S'. Note that *diff*(S, S') is always an even number because for each object that moves, two vertices change their state in the state vector.

Base step on n. Consider the State Graph construction algorithm reported in Appendix A.1. At the first iterative step of that algorithm, edges labeled with the movement of a single object are added to the edge-free state graph, linking all the possible couples of state vectors S, S' such that *diff*$(S, S') = 2$ and it exists an edge in PCG linking the two vertices that changed their state. For each of these couples (S, S'), a single edge is added as only one object in the system can do the movement associated to the two vertices whose state changes between S and S'. Therefore, for a system where only one object at a time can move (*i.e.* $n = 1$), the resulting state graph does not contain any unsafe movements.

Induction hypothesis. Assume that if n objects move concurrently, none of the possible trajectories $P \in \mathbb{P}^n$ contains an unsafe movement.

Induction step on n. Now, consider the case where $n + 1$ objects move concurrently. We want to show that, apart from this change, no unsafe movement is added to SG. More specifically, we want to prove that $\forall S, S' \in SG$ such that $2 \leq$*diff*$(S, S') \leq 2(n + 1)$, if we can add an edge in SG between S and S' labelled with $n + 1$ concurrent movements, then there cannot exist another edge between them labelled with $n + 1$ or less movements.

Let us first consider the case where *diff*$(S, S') = 2$. Assume that the two vertices changing their state between S and S' are labelled v_1 and v_{n+2}. An edge labelled with $n + 1$ movements can be added in SG between S and S' only if it exists in PCG the path $p = v_1, v_2, \ldots, v_{n+1}, v_{n+2}$ and $\forall v \in \{v_1, \ldots, v_{n+1}\}, S[v] = 1$. Now, we show that the only possible movement bringing the system from S to S' is the one where each object located in v_i, with $i \in [1, n + 1]$, moves to v_{i+1}. Assume, without loss of generality, that there is another possible movement that does not involve objects located on vertices $v_2, \ldots, v_{n+1}$. Figure 4 represents this case where the object located on v_1 must move to a different subgraph PCG_A of PCG and an object located on subgraph PCG_B must move to node v_{n+2}. If the only path connecting vertices in PCG_A with vertices in PCG_B is p (it

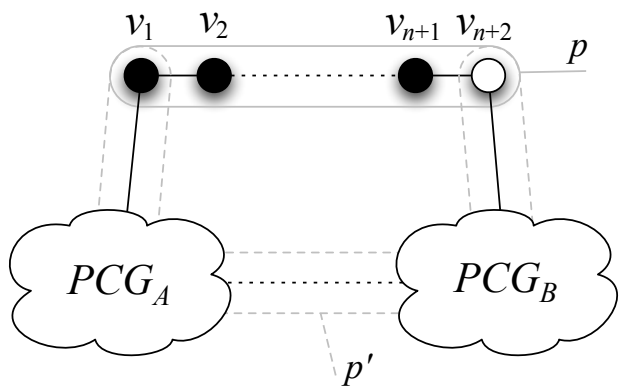

Fig. 4. Representation of the PCG which illustrates the possible existing paths connecting v_1 and v_{n+2}.

does not exist a link p' as in Figure 4), then $|\{v \in PCG_A : S'[v] = 1\}| > |\{v \in PCG_A : S[v] = 1\}|$ (and respectively, $|\{v \in PCG_B : S'[v] = 1\}| < |\{v \in PCG_B : S[v] = 1\}|$). This implies that $\exists v \in PCG_A \cup PCG_B : S[v] \neq S'[v] \Rightarrow$*diff*$(S, S') > 2$, which is impossible due to our initial hypothesis. On the contrary, if there is a path $p' \neq p$ connecting PCG_A to PCG_B, then p is part of a cycle $c \subset G$. The length of c is, by assumption, at least ℓ. In order to have *diff*$(S, S') = 2$, in S, there must be an object located on all vertices in c but v_{n+2} and all the objects located on these vertices, with the exception of those located on $v_2, \ldots, v_{n+1}$, must be moved (otherwise *diff*$(S, S') > 2$). But this means that m $(\geq \ell - n)$ objects will move concurrently. Given that $\lceil \frac{\ell}{2} \rceil > k \geq n + 1$, we have $m > 2(n + 1) - n > n + 1$, *i.e.* every other edges connecting S to S' in SG must be labelled with more than $n + 1$ movements.

Now, consider the case where *diff*$(S, S') = 2(x + 1)$ with $x \leq n$. In this case, there are $n+1$ distinct objects moving on $x+1$ distinct paths, each characterized by the presence of an object on each vertex but the last one (as p in Figure 4). The same reasoning shown for the previous case can be applied to every single path, considering that each of these paths contains strictly less than $n + 1$ objects.

Secondly, we prove by contradiction that if $\forall P \in \mathbb{P}^k$, MOTI is P-solvable $\Rightarrow \lceil \frac{\ell}{2} \rceil > k$. Assume for the moment that $k \geq \lceil \frac{\ell}{2} \rceil$. Consider the smallest cycle c in PCG constituted by vertices $v_1, \ldots, v_\ell$. Now consider, without loss of generality, a state vector S where $\forall i \in \{2 \cdot k - 1 | k \in [1, \lceil \frac{\ell}{2} \rceil]\} : S[v_i] = 1$ and $\forall i \in \{2 \cdot k | k \in [1, \lfloor \frac{\ell}{2} \rfloor]\} : S[v_i] = 0$. Consider also the state vector S' that is identical to S but where $\forall i \in \{2 \cdot k - 1 | k \in [1, \lfloor \frac{\ell}{2} \rfloor]\} : S[v_i] = 0$ and $\forall i \in \{2 \cdot k | k \in [1, \lfloor \frac{\ell}{2} \rfloor]\} : S[v_i] = 1$ (Note that, if ℓ is odd then $S'[v_\ell]$ remains unchanged and equals to 1). Then the states of all vertices – but the last in case of ℓ odd – indexed from 1 to ℓ are inverted. Such two state vectors are in SG with certainty because we are assuming that $k \geq \lceil \frac{\ell}{2} \rceil$. Now consider the following movements $M = \{[v_i, v_{i+1 \mod \ell}]\}_{i \in \{2 \cdot k - 1 | k \in [1, \lfloor \frac{\ell}{2} \rfloor]\}}$ and $M' = \{[v_i, v_{i-1 \mod \ell}]\}_{i \in \{2 \cdot k - 1 | k \in [1, \lceil \frac{\ell}{2} \rceil]\}}$. Both movements link S to S' in SG and are labeled with lower or equals concurrent moves than k. Therefore, at least an unsafe movement exists. This is in contradiction with the initial assumption that $\forall P \in \mathbb{P}^k$, MOTI is P-

solvable. □

In a nutshell, there are two possible methods to guarantee that the conditions at the basis of Theorem 6.1 always hold. The first method requires to choose, as the PCG, a topology characterized by cycles of length strictly larger than $2 \cdot x$. As a consequence, MOTI is $\mathbb{P}$-solvable in any system characterized by an acyclic graph as presented in the following Corollary 6.2. The second method requires to limit the number of objects that can move concurrently in the system.

COROLLARY 6.2. *MOTI is $\mathbb{P}$-solvable in any system characterized by a $PCG(V, E)$ that is acyclic.*

PROOF. We can consider PCG as a graph with a cycle of infinite length. Due to Theorem 6.1, MOTI is $\mathbb{P}$-solvable as long as no more than an infinite number of objects move concurrently in the system. Given that $\mathbb{O}$ is finite, the $\mathbb{P}$-solvability is always guaranteed. □

7. FROM REASONING ON STATE GRAPH TO REASONING ON PCG

Previous characterizations exploited the notion of state graph, however getting the full knowledge of the entire state graph can be computationally very expensive[1]. Moreover, calculating the state graph needs the complete knowledge of the system. At contrary, reasoning directly on the PCG opens the way to distributed solutions that relies only on local knowledge (*cf.* Section 9). Therefore we need to introduce a new MOTI characterization based on a more tractable data structure, namely the PCG.

First, this section introduces (Section 7.1) a characterization of an unsafe state based both on cycles present in the PCG and on the placement of objects on top of cycle's vertices (Theorem 7.1). This theorem leads to an interesting corollary that can be practically used as a sufficient condition for defining if a state is safe, *i.e.* if each cycle of the PCG contains at least two adjacent vertices sensing no objects. This practical condition is used in Section 8.2 to design a centralized algorithm, run by the observer, to circumvent MOTI impossibility and in Section 9 to derive a distributed algorithm to detect unsafe states.

Secondly, this section presents (Section 7.2) a characterization of unsafe movements based both on the presence of cycles in the PCG and on the placement of objects in cycle's vertices (Theorem 7.5). This theorem is used in Section 8.2 to derive an algorithm, run by the observer, that after detecting an unsafe state, can help the system to avoid an unsafe movement by constraining the movements of $k - 1$ objects.

Notations. Hereafter, we denote as c a generic cycle in a PCG and ℓ_c its length in terms of edges. Also, when needed, we consider that the vertices of a cycle c are labeled following a clockwise order. As an example, Figure 5 depicts a cycle with $\ell_c = 13$ where nodes are labelled from v_1 to $v_{\ell_c} = v_{13}$. Moreover, the notation $c \subseteq G$ means *c is a cycle in the graph G*.

Finally we introduce the notion of *unoccupied vertices'set of a cycle c*, for a given state vector S, denoted $\xi_c(S)$. This set is defined as follows:

$$\xi_c(S) = \{i \in [1, \ell_c] | S[v_i] = 0\}$$

[1] As an example, Appendix A.1 proposes an algorithm to efficiently compute the state graph whose complexity is nevertheless $O\left(\binom{|V|}{x}^3\right)$.

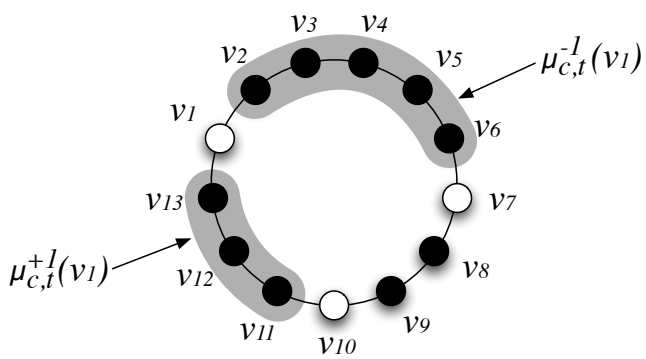

Fig. 5. Unoccupied vertices set and object sequence.

As an example in Figure 5, $\xi_c(S)$ is equal to $\{1, 7, 10\}$.

7.1 Safe State Characterization based on PCG

THEOREM 7.1 UNSAFE STATE CHARACTERIZATION. *Let PCG be a graph and $S_t \in \mathbb{S}$ a state vector.*

$$S_t \text{ is unsafe} \iff (\exists c \subseteq PCG, \forall v \in c, S_t[v] = 0 \Rightarrow \forall v' \in Adj_c(v), S_t[v'] = 1)$$

PROOF. OF THEOREM 7.1: ($\Longleftarrow$) Assume that the ℓ_c vertices of cycle c are numbered from v_1 to v_{ℓ_c} in the clockwise order (see Figure 6). For simplicity, we present first the case where $|\xi_c(S_t)| = 1$, then the generic one where $|\xi_c(S_t)| = l$.

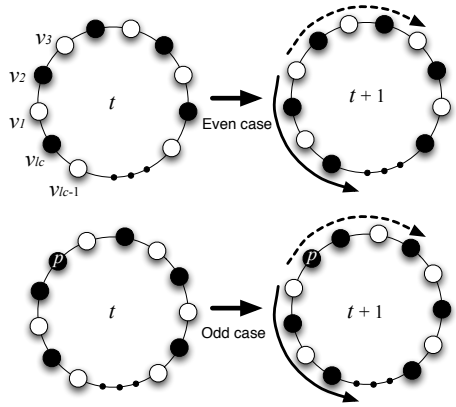

Fig. 6. In this scenario with a half occupied ℓ_c vertex cycle, in the absence of a way to identify objects, it is impossible to discern between the two movements (dashed vs. solid).

$|\xi_c(S_t)| = 1$: Then, $\exists!\ k \in [1, \ell_c]$ such that $S_t[v_k] = 0$. Therefore, we have the following two states S_t and S_{t+1}:

$$\begin{array}{rl} & \quad\ \ k \ \ k+1 \\ & \quad\ \ \downarrow \quad\ \downarrow \\ S_t = & [1 \ \cdots \ 1 \ 0 \quad 1 \quad 1 \ \cdots \ 1] \\ \text{and } S_{t+1} = & [1 \ \cdots \ 1 \ 1 \quad 0 \quad 1 \ \cdots \ 1] \end{array}$$

Then there exist two movement sets $M, M' \in \mathbb{P}_{t,t+1}$ such that:

—M and M' contain the same set of movements for all the objects that are not located in c;

—M contains the movement of a single object located in c that move counterclockwise from v_{k+1} to v_k;

—M' contains the movements of all objects located in c that move clockwise.

Then there exist at least two distinct movements from S_t to S_{t+1}. These movements are unsafe by Definition 5.4 and S_t is consequently unsafe by Definition 5.5.

$|\xi_c(S_t)| = l$ with $l \leq \left\lfloor \frac{\ell_c}{2} \right\rfloor$. This case is a generalization of the previous one. Figure 6 shows the extreme case for odd and even cases that still satisfy the theorem hypotheses. In S_t, we have $\forall i \in \xi_c, S_t[i] = 0$. Let consider S_{t+1} such that $\forall i \in [2, \ell_c], S_{t+1}[v_i] = S_t[v_{i-1}]$ and $S_{t+1}[v_1] = S_t[v_{\ell_c}]$. It follows that $\xi_c(S_{t+1}) = \{i+1 \mod \ell_c | i \in \xi_c(S)\}$. Then, there exist two movement sets $M, M' \in \mathbb{P}_{t,t+1}$ such that :

—M and M' contain the same set of movements for all the objects that are not located in c;

—M contains the movements of $|\xi_c(S_t)|$ objects located in c that move counterclockwise (corresponding to objects located on $v_{i+1}, \forall i \in \xi_c(S_t)$);

—M' contains the movements of all objects located in c that move clockwise.

Then there exist at least two distinct movement sets from S_t to S_{t+1}. These movements are unsafe by Definition 5.4 and S_t is consequently unsafe by Definition 5.5.

($\Longrightarrow$) Let's prove the contrapositive of the theorem claim.

$$(\forall c \subseteq PCG, \exists v \in c, S_t[v] = 0 \wedge \exists v' \in Adj_c(v), S_t[v'] = 0) \Rightarrow S_t \text{ is safe}$$

which is equivalent to

$$(\forall c \subseteq PCG, \exists v \in c, \exists v' \in Adj_c(v), S_t[v] = S_t[v'] = 0) \Rightarrow S_t \text{ is safe}$$

Therefore, assume to have a state S_t such that the following predicate (a) is true:

$$\forall c \subseteq PCG, \exists v \in c, \exists v' \in Adj_c(v), S_t[v] = S_t[v'] = 0 \qquad \text{(a)}$$

We want to show that all movements starting form S_t are safe. We have to recall here that, as stated in the system model, an object cannot move in one single step further than an adjacent vertex of its current location. Consider a portion of cycle c, which by hypothesis respects the assumption *(a)*, represented on the top of Figure 7. Starting from this state, only four possible evolutions of these two vertices can happen, presented from left to right on the bottom part of the same figure. Let's consider each of these cases separately.

Case 1. Consider the system evolves from S_t to case 1 of S_{t+1} as depicted in Figure 7. Neither v_i and v_j are occupied. According to the system model, no object can move through both these two vertices between t and $t+1$. Then, the edge connecting these vertices in the PCG can be considered as inexistent between t and $t+1$, as far as object movements are concerned. Thus, the cycle c can be considered as an acyclic path for this specific time step.

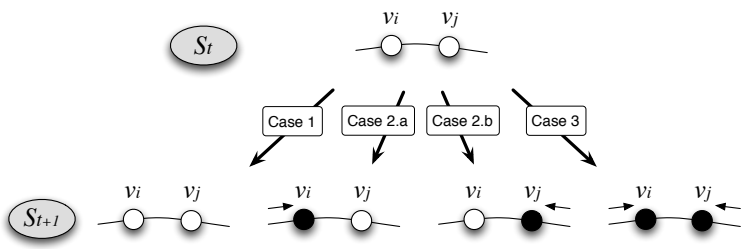

Fig. 7. Possible system evolutions of a sub-cycle from the state S_t to the state S_{t+1}.

Case 2.a. Consider the system evolves from S_t to case 2.a of S_{t+1} as depicted in Figure 7. v_i is occupied. Given that an object cannot pass through several vertices with a single movement, the object located on v_i necessarily arrived from the left edge of the picture. Then, the edge connecting these vertices in the PCG can be considered, again, as inexistent between t and $t+1$, as far as object movements are concerned. Thus, the cycle c can be considered as an acyclic path for this specific time step.

Case 2.b. This case is the same as case 2.a with inverting v_i and v_j. Then, the object located on v_j necessarily arrived from the right edge of the picture.

Case 3. Consider the system evolves from S_t to case 3 of S_{t+1} as depicted in Figure 7. Both v_i and v_j are occupied. The object located on v_i at time $t+1$ arrived from the left edge of the picture while the one located on v_j arrived from the right edge of the picture. This is the only possibility given that, in a single movement, two objects cannot traverse a same edge. Then, the edge connecting these vertices in the PCG can be considered, again, as inexistent between t and $t+1$, as far as object movements are concerned. Thus, the cycle c can be considered as an acyclic path for this specific time step.

This last reasoning can be applied on all the cycle of PCG as the assumption (a) considers any cycle of the PCG. Then, all cycles can be considered as acyclic between t and $t+1$, as far as object movements are concerned. Therefore, according to Corollary 6.2, S is safe. $\square$

The previous theorem implies that if each unoccupied vertex of a cycle in PCG is surrounded by occupied vertices, then it exists an unsafe movement starting from this state. For instance, in Figure 5, any element of the set of unoccupied vertices (*i.e.* $\{v_1, v_7, v_{10}\}$) is surrounded by occupied vertices. Further, we can cite at least two unsafe movements:

$$P = \left\{ \begin{array}{c} [v_2 \ , \ v_1] \\ [v_3 \ , \ v_2] \\ [v_8 \ , \ v_7] \\ [v_{11} \ , \ v_{10}] \end{array} \right\} \quad \text{and} \quad P' = \left\{ \begin{array}{c} [v_3 \ , \ v_4] \\ [v_4 \ , \ v_5] \\ [v_5 \ , \ v_6] \\ [v_6 \ , \ v_7] \\ [v_8 \ , \ v_9] \\ [v_9 \ , \ v_{10}] \\ [v_{11} \ , \ v_{12}] \\ [v_{12} \ , \ v_{13}] \\ [v_{13} \ , \ v_1] \end{array} \right\}$$

This observation brings to the following characterization of a safe state:

COROLLARY 7.2. *Given a state vector S, S is safe iff in each cycle c of PCG there is at least two adjacent vertices that do not host objects.*

PROOF. The proof derives directly from the negation of the claim of Theorem 7.1 i.e., S is safe $\Leftrightarrow \forall c \in PCG, \exists v \in V$ such that $S[v] = 0 \wedge \exists v' \in Adj_c(v), S[v'] = 0$. $\square$

Therefore, if we consider a PCG including the cycle depicted in Figure 5, the state of the PCG is unsafe.

7.2 Unsafe Movement Characterization based on PCG

MOTI becomes unsolvable if an unsafe movement leads the system into an unsafe state. Therefore, this section studies the problem of characterizing unsafe movements with respect to PCG when the system is in a certain state S. Thanks to Theorem 7.5, the problem of safe and unsafe movements can be studied only considering cycles in PCG.

We first need to formalize in a cycle the length of the sequence of adjacent vertices that host objects with respect to the position of a given vertex v in the same sequence. Therefore, for each direction of the cycle c with respect to v we have two possibly different values. Formally, we have:

Definition 7.3 *Clockwise Object Sequence*. The *clockwise object sequence from v* on a cycle c at time t, denoted $\mu^{-1}_{c,t}(v)$, is the length of the clockwise sequence of occupied vertices in c, starting from v (no included).

Definition 7.4 *Counterclockwise Object Sequence*. The *counterclockwise object sequence from v* on a cycle c at time t, denoted $\mu^{+1}_{c,t}(v)$, is the length of the counterclockwise sequence of occupied vertices in c, starting from v (not included).

As an example in Figure 5, if we consider the cycle c and vertex v_1 at time t, $\mu^{+1}_{c,t}(v_1)$ is equal to 3 and $\mu^{-1}_{c,t}(v_1)$ to 5. Before getting into the formal theorem characterizing when a movement is safe or not, for clarity of the reader, we provide an intuition, an informal explanation of the statement and an introductory example.

Theorem 7.5's intuition. The characterization of safe and unsafe object movements in a cycle c depends on the rotation of objects in c. A movement that brings the system from S_t to S_{t+1} is safe if the movement is constituted by some objects that move clockwise and some other moving counterclockwise. The different direction of object movements makes the whole movement unique (i.e., it cannot exist another movement that brings the system in the same state S_{t+1}). On the contrary, if the movement is formed by all objects moving in the same direction (either clockwise or counterclockwise) then this one is unsafe (because S_{t+1} could be reached also moving the objects in the opposite direction). Figure 8 shows an example of safe movement from S_t to S_{t+1} where some objects move clockwise and some other move counterclockwise. Figure 9 shows an example of unsafe movements where S_{t+1} can be reached from S_t through two distinct movement sets. The first one moves objects clockwise, the second one moves objects counterclockwise.

Theorem 7.5's informal explanation of the statement. To check if all objects move along the same direction or not, we observe how unoccupied vertices "virtually" move between two consecutive states S_t and S_{t+1} inside a cycle. These "virtual" movements are due to the real movements done by objects in c. For example, Figure 8 shows that the "virtual" movement of the unoccupied vertex v_1 at S_t into v_2 at S_{t+1} is due to the real movement of the object hosted by v_2 at S_t into v_1 at S_{t+1}. The "virtual" movement of v_7 at S_t in v_5

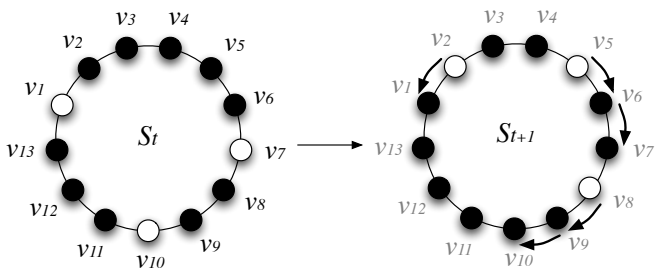

Fig. 8. Example of a safe movement from the state S_t to a state S_{t+1}.

at S_{t+1} is due to the real movement of the objects hosted by both v_5 and v_6 in v_6 and v_7 respectively.

Combining the notion of virtual movements and the one of clockwise/counterclockwise object sequences, we notice that a movement is unsafe if the distance between *each* unoccupied vertex in S_t and S_{t+1} is lesser than or equal to either its counterclockwise object sequence or its clockwise object sequence at time t. A movement is safe if the distance between *at least one* unoccupied vertex v in S_t and S_{t+1} (due to the v's virtual movement) is greater than either v's counterclockwise object sequence or v's clockwise object sequence at time t. The existence of such v is actually the proof that some objects moved clockwise and some other counterclockwise.

Introductory example. Figure 8 shows a state that can be reached from the state of Figure 5 due to a safe movement. If we consider the informal explanation of the theorem and the two object sequences for each unoccupied vertex depicted in Figure 8, we have:

(i) Clockwise: v_1 virtually moves to v_5, v_7 to v_8 and v_{10} to v_2. Then, concerning v_1, we have that its virtual movement in the cycle, from t to $t+1$, is of $(5-1)$ positions while $\mu^{-1}_{c,t}(v_1)$ is equal to 5. On v_7, we have $(8-7) \leq \mu^{-1}_{c,t}(v_7) = 2$ and on v_{10}, we have $(2-10 \mod 13) > \mu^{-1}_{c,t}(v_{10}) = 3$. Therefore, it exists at least one unoccupied vertex (i.e, v_{10}) which has virtually moved further than its clockwise object sequence.

(ii) Counterclockwise: v_1 virtually moves to v_8, v_7 to v_2 and v_{10} to v_5. Then, concerning v_1, we have $(1-8 \mod 13)\mu^{+1}_{c,t}(v_1) = 3$. On v_{10}, we have $(10-5) > \mu^{+1}_{c,t}(v_{10}) = 2$ and on v_7, we have $(7-2) \geq \mu^{+1}_{c,t}(v_7) = 5$. Therefore, it exists at least one unoccupied vertex (here, v_1 and v_{10}) which has virtually moved further than its counterclockwise object sequence. Therefore, following the theorem informal explanation, the movement is safe.

Consider now the unsafe movements shown in Figure 9 that bring to the same state S_{t+1}. We observe that, clockwise, v_1 virtually moves to v_4, v_{10} to v_{13} and v_7 to v_9. Then, regarding v_1, $(4-1) \leq \mu^{-1}_{c,t}(v_1) = 5$, regarding v_7, $(9-7) \leq \mu^{-1}_{c,t}(v_7) = 2$ and regarding v_{10}, $(13-10) \leq \mu^{-1}_{c,t}(v_{10}) = 3$. Thus, all unoccupied vertices have moved lesser than their respective clockwise object sequences. Then, the condition of the theorem is verified, and the movement is unsafe.

More formally:

THEOREM 7.5 UNSAFE MOVEMENT CHARACTERIZATION. *Let $PCG \in \mathbb{G}$ and S_t the state vector at time t, such that S_t is unsafe. Let $c \subseteq G$ a cycle such that c respects the*

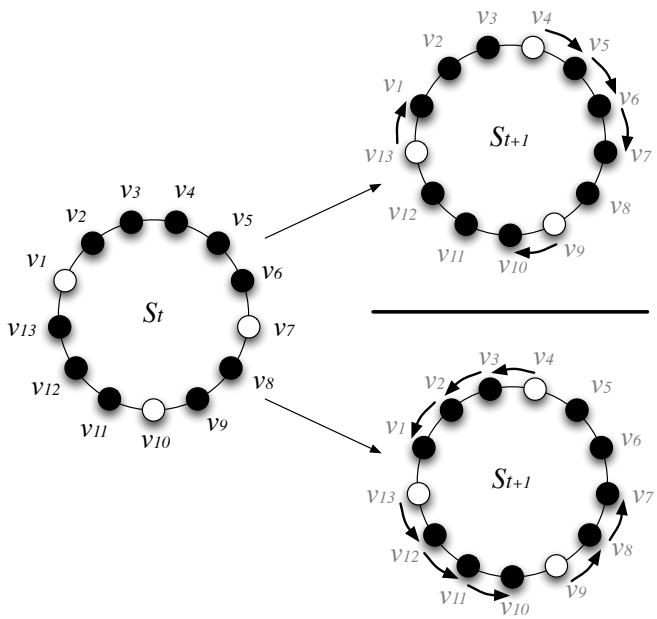

Fig. 9. Example of unsafe movements from the state S_t to a state S_{t+1}.

condition of Theorem 7.1 and no object has entered or leaved c during M_t.
$\exists k \in \{-1, 1\}, \forall i \in \xi_c(S_t), \exists j \in [1, \mu^k_{c,t}(v_i)], (i + k \cdot j \mod \ell_c) \in \xi_c(S_{t+1}) \iff M_t$ *is unsafe*

PROOF. Let consider that the hypothesis are verified for a cycle c in PCG. As no object has entered or leaved c during $[t, t+1]$, we have $|\xi_c(S_t)| = |\xi_c(S_{t+1})|$.
($\Longrightarrow$) Assume the following predicate is true: $(\exists k \in \{-1, 1\}, \forall v_i \in c, S_t[v_i] = 0 \Rightarrow \exists j \in [1, \mu^k_{c,t}(v_i)], S_{t+1}[v_{i+k\cdot j \mod \ell_c}] = 0)$. Let $J^k_{c,t}$ be the vector containing, for each empty vertex v_i of c, the value of j picked in $[1, \mu^k_{c,t}(v_i)]$ if $S_t[v_i] = 0$, and $\perp$ otherwise.

Consider the following movement sets $M, M' \in \mathbb{P}_{t,t+1}$:

—M and M' contain the same set of movements for all the objects that are not located in c;

—M contains the movements of $\sum_{i \in \xi_c(S_t)} J^k_{c,t}[v_i]$ objects located in c that move counter-k-wise (where counter-k-wise means clockwise if $k = +1$ and counterclockwise if $k = -1$);

—M' contains the movements of $\ell_c - \sum_{i \in \xi_c(S_t)} J^k_{c,t}[v_i]$ objects located in c that move k-wise (where k-wise means clockwise if $k = -1$ and counterclockwise if $k = +1$).

Then, M and M' lead to the same set $\xi_c(S_{t+1})$. Thus, M and M' correspond to two different movements from S_t to S_{t+1}. So, it exists at least one movement set (either M or M'), different from M_t, between S_t and S_{t+1}. Then, by Definition 5.4, M_t is unsafe.

($\Longleftarrow$) Let us prove the contrapositive of the claim. Assume that the following predicate is true

$$\forall k \in \{-1, 1\}, \exists v_i \in c, \forall j \in [1, \mu^k_{c,t}(v_i)], S_t[v_i] = 0 \wedge S_{t+1}[v_{i+k\cdot j[\ell_c]}] = 1. \quad (1)$$

This hypothesis means that it exists at least one unoccupied vertex which "virtually" moves further than both its clockwise and counterclockwise object sequences.

From this hypothesis, we have to show that M_t is safe. In order to simplify the reading of this proof, assume that any calculus on the label of vertices of cycle c is done, from now, modulo ℓ_c.

If $\forall v_i \in c, S_t[v_i] = 1$, given that $|\xi_t| = |\xi_{t+1}|$, then $\forall v_i \in c, S_{t+1}[v_i] = 1$. But, this outcome is contradictory because from the hypothesis $\exists v_i \in c$ such that $S_t[v_i] = 0$. Thus, $|\xi_t| = |\xi_{t+1}| \neq 0$.

Assume that for a vertex v_i, $S_{t+1}[v_i] = 0$. From the system model definition (see Section 3), an object cannot move more than one vertex distance between two consecutive time steps. Then, no object has moved through v_i between t and $t+1$. Based on this unoccupied vertex between two consecutive time, it is always possible for the observer to extrapolate deterministically the movement of objects in c. So, M_t is safe.

Thus, assume that $\forall i \in \xi_c(S_t), S_{t+1}[v_i] = 1$, otherwise, using the same method, M_t must be safe. Given this assumption and Hypothesis 1, it is possible to infer that $\exists i \in \xi_c(S_t), \forall j \in [-\mu^{-1}_{c,t}(v_i), \mu^{+1}_{c,t}(v_i)], S_{t+1}[v_{i+j}] = 1$. Moreover, it is trivial that $\forall i \in \xi_c(S_t), (i + \mu^{+1}_{c,t}(v_i) + 1) \in \xi_c(S_t)$ and $(i - \mu^{-1}_{c,t}(v_i) - 1) \in \xi_c(S_t)$, given Definitions 7.3 and 7.4. The vertices labeled by these two naturals are then occupied at time $t+1$, but unoccupied at time t. So, $\exists i \in \xi_c(S_t), \forall j \in [-\mu^{-1}_{c,t}(v_i) - 1, \mu^{+1}_{c,t}(v_i) + 1], S_{t+1}[v_{i+j}] = 1$.

In that case, as v_i is occupied at time $t+1$, assume, without loss of generality, by the symmetry of this statement, that an object has moved in counterclockwise from v_{i-1} to v_i. Then, as $S_{t+1}(v_{i-1}) = 1$, one another object has necessarily moved from v_{i-2} to v_{i-1}, and so forth until $v_{i-\mu^{-1}_{c,t}(v_i)}$. So, an object has moved from $v_{i-\mu^{-1}_{c,t}(v_i)-1}$ to $v_{i-\mu^{-1}_{c,t}(v_i)}$ given the assumption $S_{t+1}[v_{i-\mu^{-1}_{c,t}(v_i)}] = 1$. Or, $(i - \mu^{-1}_{c,t}(v_i) - 1) \in \xi_c(S_t)$. Then, obviously, no object has moved from $v_{i-\mu^{-1}_{c,t}(v_i)-1}$ to $v_{-\mu^{-1}_{c,t}(v_i)}$ between t and $t+1$. This movement is then impossible. The condition $\exists i \in \xi_c(S_t), \forall j \in [-\mu^{-1}_{c,t}(v_i) - 1, \mu^{+1}_{c,t}(v_i) + 1], S_{t+1}[v_{i+j}] = 1$ is then never verified. From that, by contradiction, $\exists i \in \xi_c(S_t), S_{t+1}[v_i] = 0$.

Thus, as it exists at least one unoccupied vertex in c at time t and $t+1$, using the same method than above, the movement M_t is safe. $\square$

The previous theorem leads to an operative corollary that can be used to prevent unsafe movements. The following corollary leverages the observation that a necessary condition for a movement to be unsafe is that this movement makes all unoccupied vertices virtually move between S_t and S_{t+1}. For instance, in Figure 5, page 15, if at least one element of the set of unoccupied vertices (*i.e.* $\{v_1, v_7, v_{10}\}$) remains unoccupied after a movement, then this movement is safe. Formally:

COROLLARY 7.6. *Given a PCG and a state vector S_t, such that S_t is unsafe, consider $c \subseteq G$ a cycle such that c respects the condition of Theorem 7.1. If less than $|\xi_c(S_t)|$ objects move in this cycle between t and $t+1$, then the movement is safe.*

PROOF. By Theorem 7.5, an unsafe movement must verify the following condition:

$$\exists k \in \{-1, 1\}, \forall i \in \xi_c(S_t), \exists j \in [1, \mu^k_{c,t}(v_i)], (i + k \cdot j \mod \ell_c) \in \xi_c(S_{t+1}).$$

This implies that given a direction of movement (either clockwise or counterclockwise), the location of *all* unoccupied vertices virtually move between t and $t+1$. ($j \in [1, \mu^k_{c,t}(v_i)]$). Thus, if only less than $|\xi_c(S_t)|$ objects are allowed to move, it exists at least one unoccupied vertex at time t, which will remain unoccupied at time $t+1$, therefore the previous condition cannot be verified and the claim follows. □

Note that this corollary gives us a practical rule we can use at a specific point in time to avoid an unsafe movement. This can be obtained by blocking a number of objects located in the PCG's cycles that is always lower than the value k obtainable applying Theorem 6.1.

8. CONSTRAINING NODE MOVEMENTS ON-LINE BY AN ACTIVE OBSERVER

This section presents two algorithms that can be run by an observer in order to solve MOTI. The first is based on the state graph, the second on PCG. The structure of the algorithms is the same: the observer first verifies if the system is in a safe state. In the affirmative case, the observer allows any movement of objects. On the contrary, *i.e.* if the system is in an unsafe state, the observer can act on the system by limiting the movement of some objects, based on the result of either Theorem 6.1, or Corollary 7.6.

8.1 An algorithm based on the State Graph

Each time the observer receives a state vector S_t, it executes Algorithm 1:

Algorithm 1:

Data: a state vector S_t, the state graph SG
1 **if** S_t *is unsafe in* SG **then**
2 $\quad k \leftarrow \lceil \frac{\ell}{2} \rceil - 1$;
3 $\quad Q \leftarrow$ Select $|\mathbb{O}| - k$ objects in $\mathbb{O}$; `% Theorem 6.1`
4 $\quad$ Block the movement of every object $o \in Q$ for the next time step;

Note that the selection done at line 3 can be performed using any selection strategy.

This algorithm checks at each time if the next movement leads to an unsafe state using SG to verify if the current state is safe. If it is not, according to the Theorem 6.1, this algorithm blocks movements of all but k objects in the system.

8.2 An algorithm based on PCG

The following algorithm improves the previous one by removing the need for the observer to calculate the SG, and by reasoning only on cycles of the PCG and the state vector S_t.

Each time the observer receives a state vector S_t, it executes Algorithm 2.

To verify if S_t is safe, the observer has to check that each cycle in PCG satisfies the assumption of Corollary 7.2 (line 4). This means that all objects in this cycle can move freely. On the contrary, if the state S is unsafe (line 7), the observer leaves at most $|\xi_c(S_t)| - 1$ objects in the cycle to move between time t and $t+1$, according to Corollary 7.6. Therefore, MOTI remains solvable for this specific cycle. Then, as all cycles are considered in this algorithm, MOTI can be solved in the interval $[t_i, t_j]$.

Algorithm 2:

```
Data: a PCG, a state vector S_t
ℓ ← 2;
repeat
    foreach cycle c ∈ PCG with length ℓ do
        if ∃v ∈ c, ∃v' ∈ Adj_c(v) st S_t[v] = S_t[v'] = 0        % Corollary 7.2
        then
            continue;
        else
            k ← |ξ_c(S_t)|;                                      % the state is unsafe
            Let only k − 1 objects do a free movement in c;      % Corollary 7.6
    ℓ++;
until there are no cycles with length equals or greater than ℓ in the PCG ;
```

More refined algorithms that block as few objects as possible while keeping MOTI solvability can be envisaged, but this is out of the scope of this paper.

9. DISTRIBUTED UNSAFE STATE DETECTION

Based on Corollary 7.2 we can derive a distributed algorithm to detect the presence of unsafe states where this detection does not rely on an omniscient observer but is done locally at each sensor. Note that this algorithm does not allow to solve MOTI unless binary sensors are connected to actuators used to block unsafe movements by modifying (temporarily) the PCG. Such actuators can, for example, close a barrier on a road, thus avoiding a potential unsafe movement. This actually corresponds to break a cycle in the PCG.

Assume that each sensor s runs the same code and each sensor is aware of a sub-graph of the PCG including only its neighbors. This graph is a star, denoted $STAR_s(V', E')$, where $s \in V'$ is the vertex in the center of the star with vertex degree equal to $|V'| - 1$. $STAR_s$ *is a subgraph of* PCG. We also assume that the speed of messages is much higher than the one of objects. The algorithm works as follows:

—A sensor s notifies any change of its state to its neighbors by sending to each sensor $s' \in STAR_s$, a message $UPDATE(state, s)$.

—Upon the arrival of a message $UPDATE(state, s')$ at a sensor s,
  —s updates the state of its neighbor s';
  —for any path s', s, s'' in $STAR_s$ such that this path has at least two vertices hosting an object, s sends a message $PROBE(seq, s)$ to either s' or s''. This message includes a sequence number and the identifier of the originator of this probe.

—Upon the arrival of a message $PROBE(seq, id)$ from a neighbor s' at a sensor s, the following cases are possible:
  —$id \neq s$: for any node s'' belonging to the path s', s, s'' in $STAR_s$ such that this path does not contain two consecutive unoccupied nodes (Corollary 7.2), s relays $PROBE(seq, id)$ to s'';
  —$id = s$: the state is unsafe. Indeed, a $PROBE$ message sent by s has been received by s itself, this means that s belongs to a cycle of the PCG and along this cycle there are not two consecutive vertices that do not host objects;

Note that all the sensors in the cycle detect the unsafe state as all of them initiate sending their own probe message. The drawback of this simple algorithm is the number of exchanged messages. This number can be traded with the knowledge of PCG owned by a sensor. Of course the larger the knowledge, the lesser the load. For example, if we consider that each sensor knows the entire PCG, we can simply run the previous algorithm by assuming that each sensor s works on a star graph $STAR_s^*$ obtained from $STAR_s$ by suppressing the vertices in $STAR_s$ that do not belong to a cycle in PCG. Note that if a sensor does not belong to any cycle in PCG, its star graph is a singleton graph. In this way, all sensors not belonging to any cycle do not participate to the unsafe state detection algorithm and, thus, do not generate and relay useless $PROBE$ messages. Also $UPDATE$ messages are exchanged among sensors involved in at least one cycle in PCG.

Further distributed algorithms that save more messages can be envisaged but this is out of the scope of this paper.

10. CONCLUSION

Automatically tracking object movements by means of sensors has strong practical impacts in several industrial fields. As such it has been tackled in various different setting since the seventies.

In this paper, we have considered a set of anonymous objects moving on the top of a binary sensor network (represented by a generic graph – Passage Connectivity Graph) and studied the problem of deterministically associating a trajectory of a single object with a track revealed by the sensor network, namely the Multiple Object Tracking and Identification (MOTI) problem. This problem can be considered as a specific instance of the more general problem of deciding how many trajectory hypotheses can be generated by a set of observations done through sensors [Crespi et al. 2008]: in this case we were interested in all those settings where the set of hypotheses always contains a single trajectory for each moving object. We have shown that solving MOTI is impossible even in a very favorable environment with ideal sensors, in which objects activate exactly one sensor at a time, there are no false object detection and there is the presence of an omniscient observer that knows the state of the entire graph.

Following this impossibility result, we have investigated which constraints could be added to the environment to make such associations possible. More specifically, we have shown that MOTI can be solved either if the graph is acyclic or the length of the smallest cycle in the graph is strictly greater than the double of the number of objects that can concurrently move in the system. This leaves the opportunity to modify the topology of the passage connectivity graph like for example in an indoor scenario. Once the maximum number of moving objects is known, our results mainly impact the deployment phase of a sensors network.

If we consider the issue of solving MOTI on-the-fly (*i.e.* without imposing constraints in the deployment phase), we also provided a characterization of MOTI solvability based on the state of sensors belonging to cycles of PCG. Using this characterization, an omniscient observer can assess on-the-fly the risk of violating MOTI (safeness of the system) and then, take the decision of blocking some specific objects to eliminate that risk. A corollary to this characterization led to a condition for detecting the safety of the system based on the state of adjacent sensors in cycles of a passage connectivity graph. This local condition leads to the design of some distributed algorithms that can be run by each sensor, avoiding

the need of the omniscient observer for detecting risks of violating MOTI.

Several open questions have still to be considered and are left for future work. For instance, a challenging aim consists in refining our algorithms in order to block as few objects as possible while keeping MOTI solvability, and characterizing the optimal complexity according to different heuristics. Alternatively, it would be possible to envisage scenarios, and consequently algorithms working on them, where objects can be moved in order to solve unsafe conditions, or where further sensors can be deployed to increase the ability of the system to distinguish independent trajectories. Furthermore, distributed algorithm, cheaper with respect to the number of messages used, can be can be designed in order to make their execution more efficient in settings where energy consumption is an issue.

Acknowledgments

This work is partially supported by the European Network of Excellence RESIST and by the European Project SM4ALL. The authors are indebted with the reviewers for their comments and suggestions that greatly improved presentation and content of the paper. A short and preliminary version of this paper [Busnel et al. 2008] appeared at DCOSS '08 (International Conference on Distributed Computing in Sensor Systems).

REFERENCES

ASLAM, J., BUTLER, Z., CONSTANTIN, F., CRESPI, V., CYBENKO, G., AND RUS, D. 2003. Tracking a moving object with a binary sensor network. In *Proceedings of the 1st International Conference on Embedded networked sensor systems (SenSys '03)*. ACM, Los Angeles, CA, USA, 150–161.

ASPNES, J., GOLDENGERG, D., AND YANG, Y. R. 2004. On the computational complexity of sensor network localization. In *Proceedings of the 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensor '04)*. Springer, Turku, Finland, 32–44.

BLOISI, D., IOCCHI, L., LEONE, G. R., PIGLIACAMPO, R., TOMBOLINI, L., AND NOVELLI, L. 2007. A distributed vision system for boat traffic monitoring in the venice grand canal. In *Proceedings of the 2nd International Conference on Computer Vision Theory and Applications (VISAPP '07)*. INSTICC, Barcelona, Spain, 549–556.

BUSNEL, Y., QUERZONI, L., BALDONI, R., BERTIER, M., AND KERMARREC, A.-M. 2008. On the deterministic tracking of moving objects with a binary sensor network. In *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '08)*. Springer, Santorini Island, Greece, 46–59.

CRESPI, V., CYBENKO, G., AND JIANG, G. 2008. The theory of trackability with applications to sensor networks. *ACM Transactions on Sensor Networks 4,* 3 (May), 1–42.

FLOYD, R. W. 1962. Algorithm 97: Shortest path. *Communications of the ACM 5,* 6 (June), 345.

GUI, C. AND MOHAPATRA, P. 2004. Power conservation and quality of surveillance in target tracking sensor networks. In *Proceedings of the 10th ACM International Conference on Mobile Computing and Networking (MobiCom '04)*. ACM, Philadelphia, PA, USA, 129–143.

HAN, M., XU, W., TAO, H., AND GONG, Y. 2004. An algorithm for multiple object trajectory tracking. In *Proceedings of the 4th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '04)*. IEEE Press, Washington, DC, USA, I–864–I–871.

KULATHUMANI, V., ARORA, A., SRIDHARAN, M., AND DEMIRBAS, M. 2009. Trail: A distance-sensitive sensor network service for distributed object tracking. *ACM Transactions on Sensor Networks 5,* 2, 1–40.

LAZOS, L., POOVENDRAN, R., AND RITCEY, J. A. 2009. Analytic evaluation of target detection in heterogeneous wireless sensor networks. *ACM Transactions on Sensor Networks 5,* 2, 1–38.

LIU, J., LIU, J., REICH, J., CHEUNG, P., AND ZHAO, F. 2003. Distributed group management for track initiation and maintenance in target localization applications. In *Proceedings of the 2nd ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '03)*. Springer, Palo Alto, CA, USA, 113–128.

MAO, X., LI, X.-Y., SHEN, X., AND CHEN, F. 2009. ilight: device-free passive tracking by wireless sensor networks. In *Proceedings of the 7th International Conference on Embedded networked sensor systems (SenSys '09)*. ACM, Berkeley, CA, USA, 315–316.

OH, S. AND SASTRY, S. 2005. Tracking on a graph. In *Proceedings of the 4th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '05)*. IEEE Press, Los Angeles, CA, USA, 195–202.

REID, D. B. 1979. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control 24,* 6, 843–854.

SHRIVASTAVA, N., MUDUMBAI, R., MADHOW, U., AND SURI, S. 2009. Target tracking with binary proximity sensors. *ACM Transactions on Sensor Networks 5,* 4, 1–33.

SINGH, J., MADHOW, U., KUMAR, R., SURI, S., AND CAGLEY, R. 2007. Tracking multiple targets using binary proximity sensors. In *Proceedings of the 6th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '07)*. ACM, Cambridge, MA, USA, 529–538.

VIANI, F., MARTINELLI, M., IORIATTI, L., LIZZI, L., OLIVERI, G., ROCCA, P., AND MASSA, A. 2009. Real-time indoor localization and tracking of passive targets by means of wireless sensor networks. In *IEEE Antennas and Propagation Society International Symposium (APSURSI 2009)*. IEEE, Charleston, SC, USA, 1–4.

WANG, Z., BULUT, E., AND SZYMANSKI, B. K. 2008. Distributed target tracking with imperfect binary sensor networks. In *2nd Annual Conference of the International Technology Alliance*. ITA, London, UK, 307–308.

WARSHALL, S. 1962. A theorem on boolean matrices. *Journal of the ACM 9,* 1 (January), 11–12.

A. APPENDIX

A.1 An algorithm for computing SG

In this appendix, we propose a state graph construction algorithm, based on the conditional transitive closure of a one-movement only state graph. This condition ensures that no inconsistent movement will be included in the state graph generated by the algorithm.

Algorithm 3: State graph construction

Data: a PCG, the number of moving object x
Result: the associated state graph $SG(\mathbb{S}, \mathbb{M})$

1 $\mathbb{S} \leftarrow \{S \in \{0,1\}^{|V|} | \sum_{v \in V} S[v] = x\}$;
2 $\mathbb{M} \leftarrow \varnothing$;
3 **foreach** $S \in \mathbb{S}$ **do**
4 **foreach** $S' \in \mathbb{S} \backslash \{S\}$ **do**
5 **if** $\exists!(v, v') \in V2$ *such that* $S[v] = S'[v'] = 1$ *and* $S[v'] = S'[v] = 0$ **then**
6 $\mathbb{M} \leftarrow \mathbb{M} \cup \{S \xrightarrow{\{[v-v']\}} S'\}$;
7 **foreach** $S \in \mathbb{S}$ **do**
8 **foreach** $S' \in \mathbb{S} \backslash \{S\}$ **do**
9 **foreach** $S'' \in \mathbb{S} \backslash \{S\}$ **do**
10 **if** $(S' \rightarrow S \in \mathbb{M}) \wedge (S \rightarrow S'' \in \mathbb{M})$
11 **and** $(\forall S' \rightarrow S'' \in \mathbb{M},\ e_{S',S''} \neq e_{S',S} \cup e_{S,S''})$
12 **and** $\forall [v', v_1] \in e_{S',S},\ \forall [v_2, v''] \in e_{S,S''},\ v_1 \neq v_2$ **then**
13 $\mathbb{M} \leftarrow \mathbb{M} \cup \{S' \xrightarrow{e_{S',S} \cup e_{S,S''}} S''\}$;
14 **if** PCG *is undirected* **then**
15 merge symmetric edges in $\mathbb{M}$;
16 **return** $(\mathbb{S}, \mathbb{M})$;

This protocol is composed of two main parts. The first part begin by creating an empty edge set and a complete vertices set. Then, starting from line 3 to line 6, the algorithm puts in the state graph all one-object-movement edges. Secondly, from line 7 to line 13, following the same mechanism as the one used in the Floyd-Warshall algorithm for transitive closure computation, three nested loops compute iteratively all possible movements with any number of concurrent movements. Finally, the last lines, from 14 to 15, merge symmetric movement edges in order to return an undirected state graph.

The termination of this protocol is trivial as it is composed only of nested loops of finite length. Moreover, the complexity of this algorithm is obviously $O(|\mathbb{S}|^3)$ as it contains three nested loops on $\mathbb{S}$.

The correctness of this algorithm is based on the correctness of the Floyd-Warshall algorithm [Floyd 1962; Warshall 1962]. The condition of line 12 only ensures that no inconsistent movement can be included in the state graph, and given that new edges in SG are generated from consistent movements, no existing movement can be ignored.

Received: 12-18-2008 – Revised: 08-05-2010