Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Distributed Intrusion Detection for the Security of Industrial Cooperative Robotic Systems ⋆

A. Fagiolini [*] G. Dini [**] A. Bicchi [***]

[*] *DEIM, Faculty of Engineering, Università degli Studi di Palermo, Italy,* fagiolini@unipa.it.
[**] *Department of Computer Engineering, Faculty of Engineering, Università di Pisa, Italy,* gianluca.dini@ing.unipi.it
[***] *Interdep. Research Center "E. Piaggio", Faculty of Engineering, Università di Pisa, Italy, and Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova, Italy,* bicchi@centropiaggio.unipi.it

Abstract: This paper addresses the problem of detecting possible intruders in a group of autonomous robots which coexist in a shared environment and interact with each other according to a set of common rules. We consider intruders as robots which misbehave, i.e. do not follow the rules, because of either spontaneous failures or malicious reprogramming. Our goal is to detect intruders by observing the congruence of their behavior with the social rules as applied to the current state of the overall system. Moreover, in accordance with the fully distributed nature of the problem, the detection itself must be performed by individual robots, based only on local information. We present a general formalism that allows to model uniformly a large variety of multi–robot systems and propose an intrusion detection protocol that is based on a *local monitor* and a suitable set–valued consensus. We apply our method to a system composed of forklifts moving within an industrial scenario setup, and we present performance results of the obtained intrusion detection system by simulation.

The availability of distributed systems gave rise in the late 80s to a profound rethinking of many decision making problems and enabled solutions that were impossible before (see e.g. the works of Bertsekas and Tsitsiklis (1991); Lynch (1996)). A similar trend is now happening in Control and will soon enable a formidable number of new robotic applications. Various distributed control policies have been proposed for formation control, flocking, sensor coverage, and intelligent transportation (see e.g. Bicchi et al. (2008); Figueiredo et al. (2001); Olfati-Saber (2006)). An intrinsic paradigm shift is indeed conveyed, from the idea of a distributed intelligent system as a collection of interacting software processes, to that of a network of physical robots that take information from the environment and act within the environment itself to change it. To this purpose, various theoretical issues are still unsolved, which include, among the most important ones, the definition of standard formalisms to describe all possible *behaviors* of a cooperative robot, and of a mechanism that allows their rapid and systematic translation into corresponding control systems. While standards are common practice in Information Technology, the same need is starting to be perceived also in Robotics at every application level, although we believe that the larger variety and higher complexity of interconnected cyber–physical systems may be the principal causes, explaining why they have not been established yet. Moreover, the larger variety of applications where multi–robot systems will be used mostly involves unattended or possibly hostile scenarios, where malicious attackers can try to tamper with the supervisory system of a (group of) robot(s) so as to obtain deviation in the system's motion (Baras (2007)). Since robots are cyber–physical systems embodying complex, real–time intelligent links between perception and action, the detection of such motion misbehavior is much more complex than that of a selfish or a corrupted agent in a Peer–to–Peer system (see e.g. the RCAR protocol by Dini and Lo Duca (2012)), which makes the *security* problem in robotic system a formidable challenge. We refer to these misbehaving robots as *intruders* that may display uncooperative behavior due to spontaneous failure or malicious reprogramming.

In this paper, we first provide a formalism that consists in a hybrid model capturing the behavior of a general class of robots. According to our formalism, robots have the ability to interact with other neighboring robots based on event–based rules. For the sake of clarity, consider an automated forklift moving in a factory's warehouse. Such a forklift is a system described by a *configuration* vector and by a *dynamics*, including inertial and geometric parameters, that depend on its physical structure, its size and shape, etc. If the forklift is meant to be used within an environment with obstacles and other cooperative forklifts,

⋆ This work has been partially supported by the European Commission with contract number FP7-2010-257649 PLANET, "PLAtform for the deployment and operation of heterogeneous NETworked cooperating objects", and with contract number FP7-2010-257462 HYCON2, "Highly-complex and networked control systems". This work has been also partially supported by the Italian Research Project ROAD (PRIN 2012).

the forklift itself is provided with a supervisory system, that can be implemented as an *automaton* and that allows it to perform a finite number of maneuvers. Each maneuver is represented by a *discrete state* of the automaton and is continuously *decoded* into a suitable control value to be applied to the forklift's actuators. Moreover, a forklift has onboard sensors, such as cameras or infrareds, that determine its *visibility* capacities and provide information that is used to plan its trajectory and prevent collisions with obstacles or other forklifts. To this purpose, its sensor outputs are constantly *encoded* into a finite number of *events*, indicating e.g the presence or absence of another *neighboring* forklift, that may or may not require the forklift to change its current maneuver. As a whole, a cooperative forklift is a complex system, that can be described by the hybrid formalism that we propose in Section 1, where all such components are formally defined.

Secondly, we propose an Intrusion Detection System (IDS) that consists of a distributed protocol which is based on two components: a local monitor and a set–valued consensus algorithm. The local *monitor* reconstructs the information of free and occupied regions in the neighborhood of a target robot, by estimating the events that it should have observed and applying an inversion of the cooperative model at the automaton level. The inversion of the complete hybrid nonlinear cooperative model is only possible for specific systems (see e.g.Millérioux and Daafouz (2007); Sain and Massey (2002)), while our technique is valid for every system in the considered class. The set–valued consensus algorithm allows the robots to reach consensus on the estimated free/occupied regions in the neighborhood of the target (Section 2).

The work is partially based on and extends the paper by Bicchi et al. (2010) by providing a more general formalization of the cooperative model and of the IDS. Although the technique applies to a large variety of mobile robots, we focus here on systems composed of automated forklifts, which find important exploitation in various industrial contests (Section 3). A performance analysis of the obtained system is also provided which shows that the communication among the robots can indeed improve the IDS capability to discover misbehaving robots.

1. COOPERATION PROTOCOLS AND LOCAL MISBEHAVIOR DETECTION

Consider n agents, $\mathcal{A}_1, \ldots, \mathcal{A}_n$, sharing a state-space, or *environment* $\mathcal{Q}$. By *cooperation protocol* $\mathcal{P}$ we mean a formal description of the agents' constitutive elements, i.e. their perceptions and actions, and of the rules used to interconnect these elements. More precisely, $\mathcal{P}$ specifies, for each robot $\mathcal{A}_i$:

- A configuration vector $q_i \in \mathcal{Q}$, where $\mathcal{Q}$ is a configuration space;
- A discrete state $\sigma_i \in \Sigma_i$, where Σ_i is the set of allowed maneuvers;
- A dynamic map f_i describing how the agent's configuration is updated: $\dot{q}_i = f_i(q_i, u_i)$, where u_i is the input vector;
- A controller map g_i that, based on the agent current configuration q_i and on its current discrete state σ_i, returns the control value $u_i = g_i(q_i, \sigma_i)$;

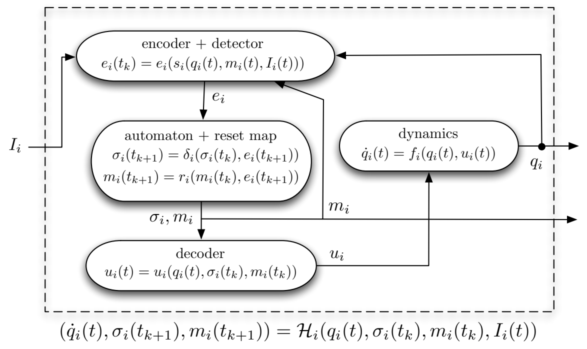
encoder + detector
$e_i(t_k) = e_i(s_i(q_i(t), m_i(t), I_i(t)))$
e_i
automaton + reset map
$\sigma_i(t_{k+1}) = \delta_i(\sigma_i(t_k), e_i(t_{k+1}))$
$m_i(t_{k+1}) = r_i(m_i(t_k), e_i(t_{k+1}))$
dynamics
$\dot{q}_i(t) = f_i(q_i(t), u_i(t))$
I_i
q_i
m_i
σ_i, m_i
decoder
$u_i(t) = u_i(q_i(t), \sigma_i(t_k), m_i(t_k))$
u_i
$(\dot{q}_i(t), \sigma_i(t_{k+1}), m_i(t_{k+1})) = \mathcal{H}_i(q_i(t), \sigma_i(t_k), m_i(t_k), I_i(t))$

Figure 1. Architecture of a protocol–based cooperative physical agent.

- A reset map describing how an agent's mission state variable m_i is updated: $m_i^+ = r_i(m_i, e_i)$;
- A logical event vector $e_i \in \mathbb{B}$, whose components activate based on the presence of other neighboring agents;
- A neighbor detection map d_i that, based on the configurations of other neighboring agents, results in the computation of the logical event vector $e_i = d_i(q_i, v_i)$, where $v_i = (q_{i_1}, \ldots, q_{i_p})$ is a vector formed of the configurations of the agent's neighbors;
- An automaton δ_i that describes how the agent's maneuver σ_i is chosen based on the occurrence of event e_i: ${\sigma_i}^+ = \delta_i(\sigma_i, e_i)$.

Having denoted the history $I_i(\tau)$, for $\tau = 0, \cdots, t$, of $\mathcal{A}_i$'s neighbor configuration set with $\tilde{I}_i(t)$, the agent's behavior is obtained as the solution $\phi_{\mathcal{H}_i}(q_i^0, \sigma_i^0, \tilde{I}_i(t))$ of the dynamic system above where $\tilde{I}_i(t)$ acts as its input. Refer to Fig. 1 for a graphical representation of the dynamic model of $\mathcal{A}_i$.

Let us now consider how an observing agent $\mathcal{A}_h$ can learn whether another agent $\mathcal{A}_i$ is cooperative or not, by measuring the trajectory $\bar{q}_i(t)$ that it has executed, over successive observation periods. It generally holds that $\mathcal{A}_i$'s neighbor configuration set I_i is partially unknown to $\mathcal{A}_h$, since its neighborhood is not entirely included in $\mathcal{A}_h$'s visibility region.

The local monitor consists of a *set–valued observer* $\tilde{\mathcal{H}}_i$ allowing the monitor agent $\mathcal{A}_h$ to compute all possible behaviors $\tilde{q}_i$ of the target agent $\mathcal{A}_i$ based on its measured behavior $\bar{q}_i(t)$, on the set of its known neighbors $\bar{v}_i$, and on the monitor's current visibility map V_h. This also allows the monitoring agent $\mathcal{A}_i$ to *infer* the presence or the absence of neighbors of $\mathcal{A}_i$ (indeed a map $\hat{I}^{h,i}$ of free/occupied regions in $\mathcal{A}_i$'s neighborhood):

$$(\tilde{q}_i, \hat{I}^{h,i}) = \tilde{\mathcal{H}}_i(\bar{q}_i, \bar{v}_i, V_h)\,.$$

The property that makes this approach appealing is that the observer can be *automatically generated* once the cooperation protocol $\mathcal{P}$ is specified. The local monitor is based on the following components:

- A copy of the target agent's dynamics f_i;
- An *uncertain encoder map* $\tilde{d}_i$ describing how an estimated *event vector* $\hat{e}_i$ is computed, based on the measured configurations of the agent $\bar{q}_i$ and of its

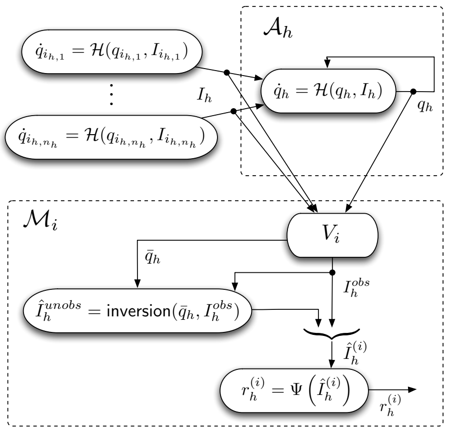
$\mathcal{A}_h$
$\dot{q}_{i_{h,1}} = \mathcal{H}(q_{i_{h,1}}, I_{i_{h,1}})$
$\dot{q}_{i_{h,n_h}} = \mathcal{H}(q_{i_{h,n_h}}, I_{i_{h,n_h}})$
I_h
$\dot{q}_h = \mathcal{H}(q_h, I_h)$
q_h
$\mathcal{M}_i$
V_i
$\bar{q}_h$
I_h^{obs}
$\hat{I}_h^{unobs} = \mathsf{inversion}(\bar{q}_h, I_h^{obs})$
$\hat{I}_h^{(i)}$
$r_h^{(i)} = \Psi\left(\hat{I}_h^{(i)}\right)$
$r_h^{(i)}$

Figure 2. Architecture of a local monitor onboard of agent $\mathcal{A}_h$ that is able to estimate the cooperativeness of an agent $\mathcal{A}_i$ by using only local information.

known neighbors $\bar{v}_i$, and on the current visibility V_i of the observer: $\hat{e}_i = \tilde{d}_i(\bar{q}_i, \bar{v}_i, V_h)$;

- An *uncertain automaton* defined through the nondeterministic map
$$\tilde{\delta}_i(\hat{\sigma}_i, \hat{e}_i) = \{\bar{\sigma} \in \Sigma_i \mid \exists \sigma \in \hat{\sigma}_i \mid \bar{\sigma} = \delta_i(\sigma, \hat{e}_i)\} \ ,$$
describing how the agent's estimated action $\hat{\sigma}_i$ is updated based on the estimated event vector $\hat{e}_i$:
$$\hat{\sigma}_i^+ = \tilde{\delta}_i(\hat{\sigma}_i, \hat{e}_i)\,; \tag{1}$$
- A copy of the controller map g_i.

The predicted behaviors $\tilde{q}_i$ generated by the observer $\tilde{\mathcal{H}}_i$ are then compared with the actual one $\bar{q}_h$. A behavior $\bar{q}_i(t)$ results to be compatible with the model $\mathcal{H}_i$ during an observation period $T = [t_k, t_{k+1})$, if
$$||\bar{q}_i(t) - \tilde{q}_i(t)|| < \varepsilon\,, \text{ for all } t \in [t_k, t_{k+1})\,,$$
where ε is an accuracy parameter, and $||\cdot||$ is the Hausdorff norm. Therefore, if no predicted behavior is compatible then the agent is uncooperative, otherwise the agent is either certainly cooperative or still uncertain. Once a compatible behavior is found, the corresponding map $\hat{I}^{h,i}$, composed of the set of positions where neighboring agents can be, is selected.

2. DISTRIBUTED SET–VALUED CONSENSUS PROTOCOL

Consider m_i observing agents, $\mathcal{A}_{i_1}, \cdots, \mathcal{A}_{i_{m_i}}$, trying to reach an agreed decision on the cooperativeness of a common neighbor $\mathcal{A}_i$. Here we assume that agents are connected through a communication topology described by an *undirected* graph $G(V_G, E_G)$, where V_G is a node set representing the agents and E_G is an edge set representing agents that are within communication range. Recall that a graph is undirected when the fact that $\mathcal{A}_h$ can send a message to $\mathcal{A}_k$, implies also the reverse. We show here how the observing agents can construct a unique global estimate I_i^* of the occupancy map I_i only through one–hop message exchange.

As already stated in the introduction, off–the–shelf solutions for network agreement are inadequate to our context, as they typically work on data represented by real numbers or vectors and use very simple combination rules. As the outputs of local monitors are continuous sets, we need to attack the consensus problem from a more general perspective. In this vein, we consider that the consensus domain is $\mathcal{Q}$ and that each agent participating in the estimation process has a set–valued state $X_h \in 2^{\mathcal{Q}}$, for $h = 1, \cdots, m_i$. We also assume that a generic *merging function* $F : 2^{\mathcal{Q}} \times 2^{\mathcal{Q}} \to 2^{\mathcal{Q}}$ is intended to be used to combine any two states X_h, X_k of two different agents into a new state value $F(X_h, X_k)$. A composed merging function can be introduced as
$$\begin{aligned} F^{(l)} &: 2^{\mathcal{Q}^l} \to 2^{\mathcal{Q}} \\ &\quad (X_1, \cdots, X_l) \mapsto F(\cdots F(X_1, X_2) \cdots, X_l)\,. \end{aligned}$$
A hypothetical centralized process having full knowledge of all agents' initial estimates would be able to compute in one step the following estimate
$$X^* = F^{(m_i)}(X_1(0), \cdots, X_{m_i}(0))\,. \tag{2}$$

If F is both *commutative*, i.e., $F(X_1, X_2) = F(X_2, X_1)$ for all X_1, X_2, and *associative*, i.e., $F(X_1, F(X_2, X_3)) = F(F(X_1, X_2), X_3)$ for all X_1, X_2, X_3, the set–valued estimate X^* is well–defined, since it is independent of the order by which the estimates are processed. We also require that F be *idempotent* if $F(X_1, X_1) = X_1$ for all X_1. Moreover, let $CV_h(p)$ be the set of agents that can transmit a message to $\mathcal{A}_h$ by passing through at most $p-1$ other agents, i.e. $CV_h(p) = \{j \in V_G \mid \text{dist}(i, j) \leq p\}$, where $\text{dist}(h, k)$ is the so–called geodesic distance of $\mathcal{A}_h$ from $\mathcal{A}_k$, i.e. the shortest path length between the two agents. Recall the notion of graph diameter being the maximum distance between any two nodes in the graph, i.e. $\text{diam}(G) = \max_{i,j \in V_G} \text{dist}(i, j)$.

We are now ready to prove the following result, which has a theoretical importance going beyond the scope of this paper, and involving the convergence of a class of set–valued consensus protocol systems:

Theorem 1. (Set–Valued Consensus Protocol). A collection of m_i agents running a consensus protocol described by the dynamic system
$$\begin{cases} X_h(k+1) = F^{(p_h(1))}(X_{h,1}(k), \cdots, X_{h,p_h(1)}(k))\,, \\ \quad X_h(0) = U_h\,, \end{cases}$$
where $p_h(k) = \text{card}(CV_h(k))$, for all agents h, converges to the centralized consensus state in at most $\tilde{n} = \text{diam}(G)$ steps, i.e.,
$$X(\tilde{n}) = \mathbf{1}_{m_i} X^*\,,$$
if F is commutative, associative, and idempotent, and if the communication graph G is connected.

Proof 1. Let us first prove, by induction, that the consensus state of an agent $\mathcal{A}_h$ after k consensus steps is
$$X_h(k) = F^{(p_h(k))}(X_{h,1}(0), \cdots, X_{h,p_h(k)}(0))\,.$$
The property is trivially satisfied after one consensus step:
$$X_h(1) = F^{(p_h(1))}(X_{h,1}(0), \cdots, X_{h,p_h(1)}(0))\,.$$
By assuming that the property holds after k steps, we want to prove its satisfaction after the $(k+1)$–th step. We have:
$$X_h(k+1) = F^{(p_h(1))}(J_1(k), \cdots, J_{p_h(1)}(k))\,, \tag{3}$$

where $J_i(k) = F^{(p_i(k))}(X_{i,1}(0), \cdots, X_{i,p_i(k)}(0))$ by the inductive hypothesis. Moreover, note that the order by which every estimate is processed is irrelevant, thanks to

F's associativity and commutativity, and that multiple occurrence of the same estimate $X_{i,j}(0)$ can be simplified through F's idempotency. Eq. 3 involves the states of all agents $l \in CV_j(k)$ where $j \in CV_h(1)$, whose union gives by definition $CV_h(k+1)$, the set of agents that can send a message to $\mathcal{A}_h$ via a communication path of at most $k+1$ other agents, which proves the property. To finally prove the theorem, it is sufficient to note that, for all $k \geq \tilde{n}$, $CV_h(k) = V_G$ and hence $p_h(k) = m_i$, as the communication graph G is connected. Therefore, we have

$$\begin{aligned} X_h(k) &= F^{(p_h(\tilde{n}))}(X_{h,1}(0), \cdots, X_{h,p_h(\tilde{n})}(0)) = \\ &= F^{(m_i)}(U_1 \cdots, U_{m_i}) = X^* , \end{aligned}$$

for all h and all $k \geq \tilde{n}$, which concludes the proof.

We can now move to the main implication of Theorem 1 as for what it concerns our intrusion detection problem. A hypothetical *centralized observer* receiving all m_i estimates would be able to compute in a single step the following merged estimate

$$I_i^c = \hat{I}_i^{(i_1)} \cap \hat{I}_i^{(i_2)} \cap \cdots \cap \hat{I}_i^{(i_{m_i})} ,$$

where $\cap$ is the set–theoretic intersection. Since the operator $\cap^*$ satisfies the hypotheses of Theorem 1, it is straightforward to show the following:

Corollary 1. (Monitor Agreement Protocol). A set–valued consensus protocol where

- the communication graph G is connected,
- the generic consensus state X_h is initialized with the locally estimated occupancy map, i.e.
$$U_h = \hat{I}_i^h(t_k|t_{k+1}) , \text{ and}$$
- $F^{(p_h(1))}$ is defined through the merging function
$$\begin{aligned} \cap^* \, : \, & 2^{\mathcal{Q}} \times 2^{\mathcal{Q}} \to 2^{\mathcal{Q}} \\ & (X_1, X_2) \mapsto \{x \,|\, \exists x_1 \in X_1 \setminus \emptyset, x_2 \in X_2 \setminus \emptyset \,| \\ & \qquad x = x_1 \cap x_2\} , \end{aligned}$$

converges in finite time to a consensus state $X^* = \mathbf{1_{m_i}} I_i^*$ with $I_i^* = I_i^c$. Moreover, the very same decision on $\mathcal{A}_i$'s cooperativeness computed by the centralized observer is reached in finite time by all agents.

3. APPLICATION

In this section the proposed architecture for cooperative agents and IDS is applied to an industrial scenario, where a number of automated forklifts move within an environment that can be represented as a matrix of cells and macro-cells. Cells are square regions that can be exclusively occupied by a single forklift to prevent collisions, and macro–cells are sequence of cells, representing e.g. corridors or narrow paths, whose access from the forklifts needs to be exclusively handled to prevent deadlocks. Forklifts are assigned with paths, being sequences of adjacent cells or macro-cells, that may intersect. Forklifts are required to travel at a maximum speed v_{max}, if the current cell or macro–cell if free, or decelerate if another forklift is approaching from a path on its right. To detect their neighbors, forklifts are provided with 360–degree cameras with visibility range R_i. More formally, we consider n forklifts that move within a shared environment $\mathcal{Q} = R^2 \times SO(2) \times R$ that is divided into cells that can be accessed with mutual exclusion policy, i.e. only one forklift at most can occupied each cell at a time, to avoid collisions. Each generic forklift $\mathcal{A}_i$ can accelerate, decelerate while going straight, turning left or right. Each forklift is assigned with a path that is a sequence of cells that it has to execute. To avoid deadlock, cells in the path are organized into macro–cells that are sets of adjacent cells. Macro–cells can be accessed with concurrent access policy by multiple forklifts, if they are proceeding toward the same direction, with mutual exclusion policy otherwise. The considered system is an instance of $\mathcal{P}$ (Section 1), whose main elements are the following:

- The dynamic map
$$\begin{aligned} f_i \, : \, & \mathcal{Q} \times \mathcal{U}_i \to T_{\mathcal{Q}} \\ & (q_i, u_i) \mapsto (v_i \cos\theta_i, \, v_i \sin\theta_i, \, \omega_i, \, a_i)^T \, ; \end{aligned}$$
which describes how the continuous state $q_i = (x_i, y_i, \theta_i, v_i)$ is updated, based on its input $u_i = (a_i, \omega_i)$;
- The topology set $T_i = \{\eta_{i,1}(q_i), \eta_{i,2}(q_i)\}$, where: 1) $\eta_{i,1}(q_i)$ is a topology returning the region composed of the (x, y)–coordinates of the next macro–cell it has to access to; 2) $\eta_{i,2}(q_i)$ is a topology returning the region composed of the (x, y)–coordinates of the next cell to which it needs to access to.
- The discrete state set is $\Sigma_i = \{\mathsf{ACC}, \mathsf{DEC}\}$, where ACC indicates that the forklift $\mathcal{A}_i$ is required to accelerate up to the maximum speed, and DEC indicates that it is required to decelerate down to null speed;
- The deterministic automaton is
$$\begin{aligned} \delta_i \, : \, & \Sigma_i \times 2^{E_i} \to \Sigma_i \\ & (\mathsf{ACC}, e^{i,1}) \mapsto \mathsf{ACC} \\ & (\mathsf{ACC}, e^{i,2}), (\mathsf{ACC}, e^{i,3}), (\mathsf{DEC}, e^{i,4}) \mapsto \mathsf{DEC} \, , \\ & (\mathsf{DEC}, e^{i,1}) \mapsto \mathsf{ACC} \\ & (\mathsf{DEC}, e^{i,2}), (\mathsf{DEC}, e^{i,3}), (\mathsf{DEC}, e^{i,4}) \mapsto \mathsf{ACC} \end{aligned}$$
where $e^{i,1}, \cdots, e^{i,4}$ are suitable events whose definition is omitted here for space reasons.

For the sake of clarity, consider the scenario in Fig. 3 including five forklifts that must solve conflicts at cell and macro–cell levels. At cell level, forklifts 00 and 01 need to negotiate the access to the cells 123 and 125, forklifts 01 and 02 need to negotiate the access to the cell 124; forklifts 01 and 04 need to negotiate the access to the cell 125; at macro–cell level, forklifts 00 and 04 need to negotiate the access to the macro–cell $\{125, 165, 205\}$. Let us assume that forklift 00 incorrectly stops, thus causing a deadlock in the system. Fig. 4 shows how the local monitors on the forklifts 01, 02, 03 and 04, can obtain an estimate the occupancy map trying to explain forklift 00's behavior. The figures show that not all four monitors can detect the non-cooperation of forklift 00 without relying on communication, in which case forklift 00 is optimistically considered as correct. The performance of the proposed IDS has been evaluated through execution of a large number of simulations, in which parameters, such as the forklifts initial positions and paths, have been randomly generated. Simulations including no cell or macro–cell conflicts have been discarded. In every simulation, only one forklift was programmed to uncooperatively stop while accessing to the shared cell or macro–cell. The number of local monitors was also varied. For every simulation, we have measured if at least one local monitor has discovered the misbehavior of the uncooperative forklift (this situation is graphically represented by a red circle on the uncooperative forklift).

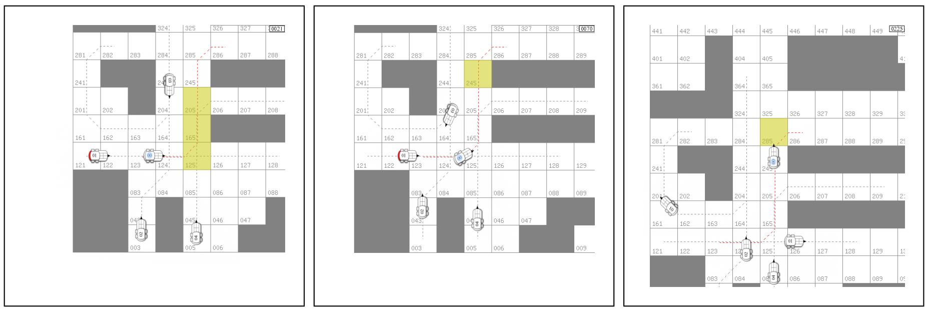

Figure 3. Snapshots from the simulation of five forklifts that cooperatively plan their motions to avoid collisions and deadlocks.

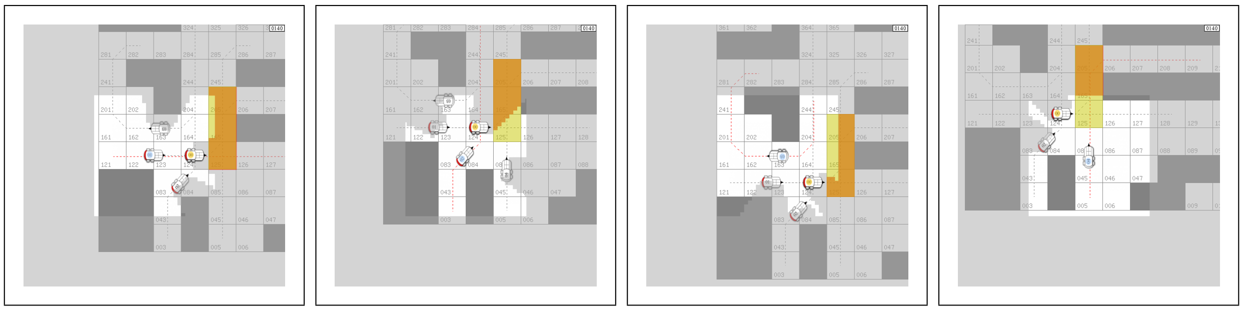

Figure 4. Occupancy maps of the forklift 00 reconstructed by the other forklifts: a blue circle indicates the observing forklift, light gray represents region where a local monitor cannot see, green (orange) represent regions where the absence (presence) of other forklifts is required. The figures show that not all four monitors can detect the uncooperation of forklift 00 without relying on communication.

# monitors	# detections	# false negatives
2	8.1 %	78.3 %
3	13.9 %	81.2 %
4	24.2 %	80.7 %
5	29.9 %	75.9 %
6	33.3 %	76.6 %
> 6	~ 34 %	~ 79 %

(a)

# monitors	# detections	# false negatives
2	23.2 %	66.7 %
3	48.7 %	38.6 %
4	66.7 %	17.7 %
5	75.8 %	14.8 %
6	89.3 %	12.5 %
> 6	~ 93 %	~ 7 %

(b)

Table 1. Performance

By assuming the optimistic approach, implying that an uncertain forklift for which an explanation to its behavior still exists is considered cooperative, we have measured the percentage of false negatives. This is summarized in Table 1–a. The table shows that the number of local monitors affects the effectiveness and reliability of the detection system only for small numbers (< 6). For larger numbers of local monitors, the monitors themselves are even unable to see the misbehavior forklift, which is hided by some other forklifts, and thus cannot effectively participate in the detection. It is also apparent that, without monitor communication, the IDS is ineffective.

We finally show how and in what amount the ability of the detection system is improved by means of communication. Before doing this, we show how local monitors can capitalize on the possibility to share locally estimated occupancy maps by using the proposed *set–valued consensus*. Referring to the example above, Fig. 5, one for each local monitor, show how the occupancy maps are iteratively improved, which finally allows to detect the presence of the uncooperative forklift. In each figure, the first picture on the left represents the estimation that the local monitor has computed based on only locally available information, and the k–th picture (with $k > 1$) shows how the same occupancy map is improved after k steps of the consensus algorithm. By comparing the last picture on the right of the four figures, it is possible to show that all local monitors consent on the absence of an forklift in the neighborhood of forklift 00, which can be consequently categorized as uncooperative. In Table 1–b a comparison of the results obtained without and with communication is reported, by considering the same simulations that were selected in the previous section. It is shown that the system's detection capability is much further improved. It is finally worth observing that the communication costs are related to the quantity of information needed to exchange the estimated occupancy maps; however communication among robots allows the achievement of high misbehavior detection capacities.

4. CONCLUSION

The misbehavior detection problem in systems where robots interact with each other based on event–based rules was addressed. While in fault diagnosis approaches based on model excitation, would require the given motion cooperation protocol to be modified, this is unnecessary according to our technique.

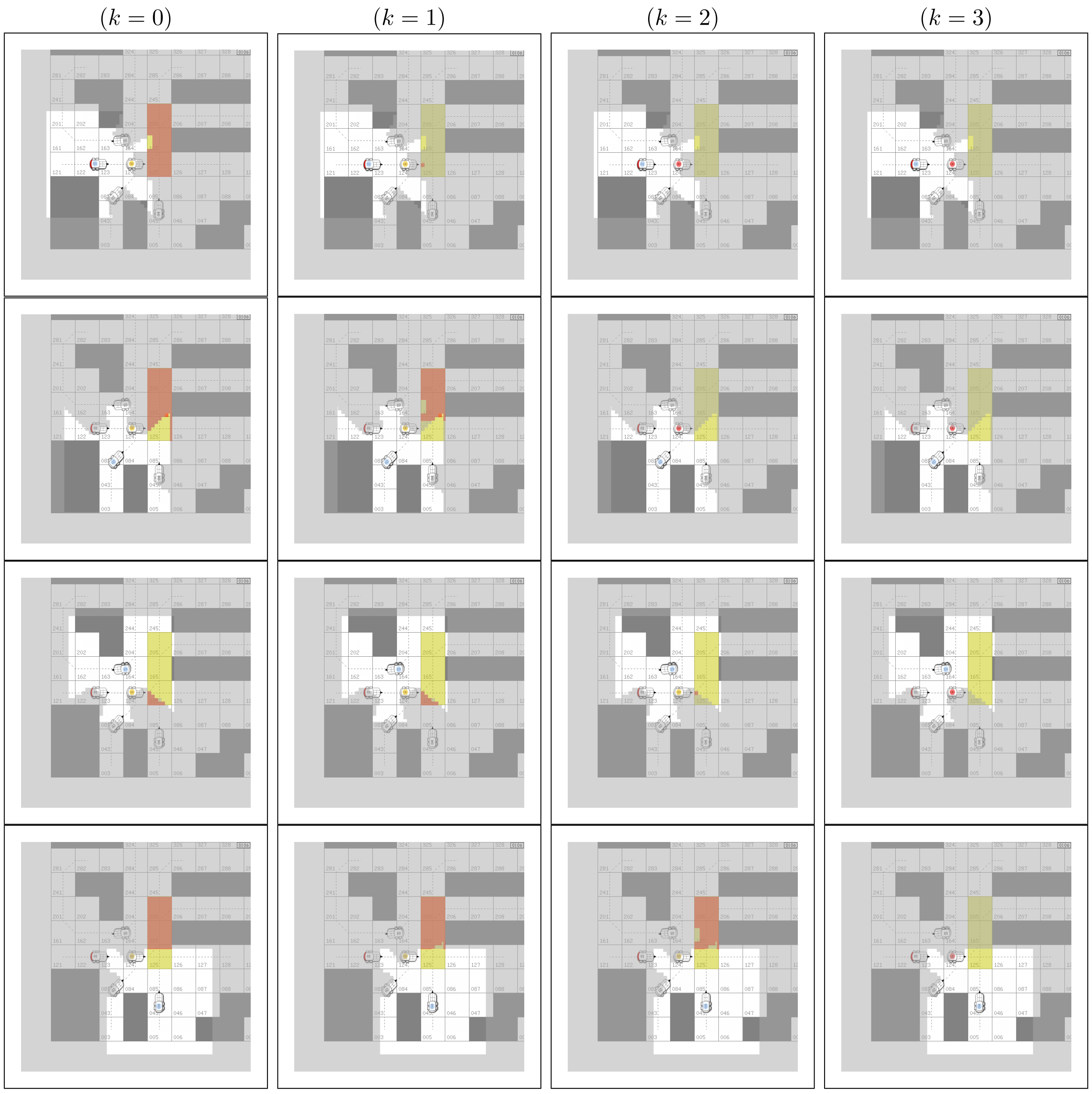

Figure 5. Consensus run of the local monitors: forklift 01 communicates with forklift 02 and 03, forklift 02 communicates with forklift 01 and 04, forklift 03 communicates with forklift 01, forklift 04 communicates with forklift 02.

REFERENCES

Baras, J. (2007). Security and trust for wireless autonomic networks: Systems and control methods. *European Journal of Control*, 13(2-3), 105–133.

Bertsekas, D. and Tsitsiklis, J. (1991). A survey of some aspects of the parallel and distributed iterative algorithms. *Automatica*, 27(1), 3–21.

Bicchi, A., Danesi, A., Dini, G., La Porta, S., Pallottino, L., Savino, I.M., and Schiavi, R. (2008). Heterogeneous wireless multirobot systems. *IEEE Robotics and Automation Magazine*, 15(1), 62–70.

Bicchi, A., Fagiolini, A., and Pallottino, L. (2010). Toward a society of robots: Behavior, misbehavior, and security. *IEEE Robotics & Automation Magazine*, 17(4), 26–36.

Dini, G. and Lo Duca, A. (2012). Towards a reputation–based routing protocol to contrast blackholes in a delay tolerant network. *Ad Hoc Networks*, 10(7), 1167–1178.

Figueiredo, L., Jesus, I., Machado, J., Ferreira, J., and Martins de Carvalho, J. (2001). Towards the development of intelligent transportation systems. *Proc. IEEE Intelligent Transportation Systems*, 1206–1211.

Lynch, N.A. (1996). *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Millérioux, G. and Daafouz, J. (2007). Invertibility and flatness of switched linear discrete-time systems. *Hybrid Systems: Computation and Control*, 714–717.

Olfati-Saber, R. (2006). Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Trans. on Automatic Control*, 51(3), 401–420.

Sain, M. and Massey, J. (2002). Invertibility of linear time-invariant dynamical systems. *IEEE Trans. on Automatic Control*, 14(2), 141–149.