POLITECNICO DI TORINO

Master's Degree in Computer Engineering - Automation and Intelligent Cyber-Physical Systems

Politecnico di Torino

Master's Degree Thesis

Distributed dynamic estimation for cyber-physical systems

Supervisors

Prof. Sophie FOSSON

Prof. Diego REGRUTO

Candidate

Roberto BRAMUCCI

April 2024

Abstract

In recent years, cyber-physical systems are gaining increasing attention due to their versatility of use in various application fields. Their peculiar characteristic of integration between computational aspects and physical processes determines the necessity of addressing new vulnerabilities and security issues. Attacks launched in the cyber domain can have consequences on the physical components, posing serious risks to industrial and daily life activities, such as transportation systems or healthcare. Cyber-physical systems are composed of a set of interconnected devices that take measurements from the physical world through sensors and aim at processing the measured data for some purpose. This thesis examines the problem of estimating the state of cyber-physical systems subjected to sensor attacks, known as secure state estimation, starting with a mathematical analysis of the problem based on the existing literature. Since the cyber-physical systems framework is intrinsically distributed, the measurement acquisition occurs in a decentralized fashion. On the other hand, the computational aspect of problem resolution can be handled in two different ways. Firstly, we focus on the centralized approach, such that the sensor measurements are sent to a single, central computing device, which processes the data with a global perspective. An observer algorithm is designed and examined for this purpose, and its performance is compared with a state-of-the-art algorithm. Subsequently, a decentralized strategy is also employed for computations, as it is considered more suitable for cyber-physical systems due to privacy and fault vulnerabilities arising from the presence of a unique computing device. An innovative observer tailored for solving secure state estimation in this setting is introduced and evaluated. Two example problems are addressed, a theoretical one and a potential real-world application, employing the aforementioned algorithms for numerical simulations. Considering that centralized observers yield superior performance compared to their distributed counterparts, albeit with various practical and security-related drawbacks, this work focuses on the development and evaluation of decentralized algorithms, for which a standard has not yet been established in the literature.

Table of Contents

List of Tables

List of Figures

Acronyms

CPS

Cyber-Physical System

SSE

Secure State Estimation

DSSE

Distributed Secure State Estimation

FC

Fusion Center

IST

Iterative Shrinkage-Thresholding algorithm

DIST

Distributed Iterative Shrinkage-Thresholding algorithm

ETPL

Event-Triggered Projected Luenberger observer

WSN

Wireless Sensor Network

RSS

Received Signal Strength

Chapter 1

Introduction

1.1 Cyber-physical systems

A cyber-physical system (CPS) is an integration of computation with physical processes whose behavior is defined by both cyber and physical parts of the system [1]. It consists of a collection of *computing* devices, *communicating* with one another and *interacting* with the physical world via sensors and actuators in a feedback loop [2]. A cyber-physical system is the intersection of the physical and the cyber parts, therefore the interaction between the physical and the computational components must be analyzed, rather than considering them as separate entities [1]. The development of processing, wireless communication, and sensor technologies has led to the manufacture of low-cost, highly capable components for cyber-physical systems, even though some types of these systems have been used in industry since the 1980s. Due to the versatility of application fields and the potential advantages it offers for society, economy, and the environment, research attention on CPSs has intensified, gaining recognition from academia, industry, and also governments. This research has the potential to have a profound impact on how engineering systems are designed and developed to address societal demands in a number of areas, including energy, environment, and healthcare, as illustrated in [3] and in [4]:

- Energy Systems: sustainable energy generation, transmission, and distribution are the main objectives. Real-time distributed sensing, measurement, and analysis can improve the responsiveness and dependability of electric energy production and distribution.
- Transportation Systems: the primary goal is to address sustainability, efficiency, traffic congestion, and safety through the development of intelligent vehicles, public transportation, and traffic systems.
- Agriculture: the implementation of a wide range of modern agricultural

management strategies and technologies has the goal of increasing accuracy in agriculture.

- Healthcare and Medical Systems: designing and developing medical systems and technologies with improved intelligence, interoperability, efficiency, and reliability.

The aforementioned examples represent only a small fraction of the numerous fields in which these types of systems can be applied. Other possible areas of use are assisted living, process control, avionics, critical infrastructure control, distributed robotics, defense systems, and manufacturing [5]. On the other hand, the distributed nature of these large-scale systems, together with the integration of computing, communication, and control, results in increased vulnerabilities and security weaknesses [6]. Since sensors and actuators are essential parts of control systems, focusing on their security is crucial. Sensors measure critical variables and communicate them to the controllers, which compute the control inputs based on the sensor measurements. The actuators synthesize the control inputs to enable the system to accomplish tasks like tracking and regulation. Along with these, a communication network possibly connected to the Internet is used to enable the exchange of information between sensors, actuators and controllers. Therefore, this connection makes CPS more exposed to attacks that target the physical component, even though they are launched in the cyber domain. As a result, new security problems that differ from those of traditional cyber security have emerged [7]. Malicious attacks can cause major performance degradation and even system failure by jamming communication channels and introducing false data into sensors, modifying their measurements. The security of cyber-physical systems has drawn increasing attention from the control community in an effort to evaluate and minimize the negative effects of such intrusions, since these systems are frequently essential parts of key infrastructures and are becoming more autonomous, i.e. they do not need human intervention to operate [8]. Some of the major attacks on control systems are mentioned in [9] and summarized in Table 1.1.

1.2 Secure state estimation problem

Closely linked to the security issue of CPS is the *secure state estimation* (SSE) problem, whose objective is to reconstruct the system state in the presence of attacks on sensors. Malicious attacks on actuators, which are also possible and are addressed in [7], are not considered here. No information is available regarding the magnitude, statistical description, or temporal evolution of the adversarial attacks; otherwise, standard techniques used to cope with noise could also be applied to the attacks. The unique assumption made on sensor attacks is sparsity, i.e. the

Attack	Year	Target	Type	Consequences
Maroochy	2000	Water services in Queensland, Australia.	Cyber attack to SCADA network.	Evacuation of untreated sewage into storm water drains and local waterways.
Stuxnet	2009	Iranian uranium enrichment plant.	Cyber attack to SCADA systems.	Significant damage to centrifuges.
RQ-170	2011	US RQ-170 unmanned aerial vehicle (UAV).	GPS signal spoofing.	Loss of control of the UAV in Iran.
ABS	2013	Mazda RX7 ABS sensors (test bed).	Non-invasive spoofing attacks for Anti-lock Braking Systems.	Experimental test.
Ukraine	2015	Ukrainian power distribution networks.	Spoofing of control commands through malware.	Outages and lasting damages.
Jeep hack	2015	Jeep car on a highway in St. Louis (USA).	Remote manipulation of Electronic Control Units through cellular connection.	Under control test.

Table 1.1: Examples of real and experimental CPS attacks.

attacker has access to only a small portion of the overall number of sensors. Given the distributed nature of the system, along with the large dimensionality, this assumption is considered realistic. However, the SSE problem is intrinsically a non-polynomial (NP) hard problem since the set of attacked sensors is unknown and combinatorial candidates should be checked [10]. The first possible strategy is brute force search, i.e. try all possible attack configurations. The SSE problem can be formulated as a non-convex ℓ_0 minimization problem, which can be solved by an ℓ_0 decoder. This approach is suited for small systems since the computational complexity grows combinatorially with the number of sensors, leading to excessive memory and time requirements for relatively larger systems. For this reason, Fawzi et al. (2014) [7] propose a computationally efficient relaxation method, which recasts the original non-convex ℓ_0 optimization problem into a convex ℓ_1/ℓ_r problem, by leveraging the sparsity assumption. This relaxed convex problem can be solved in polynomial time by a so-called ℓ_1/ℓ_r decoder. Shoukry and Tabuada (2016) [11] develop gradient-descent algorithms with increased computational efficiency, thanks to the adoption of event-triggered techniques. They can be considered an improvement of compressive sensing techniques where part of the signal is sparse, i.e. the attacks, and the other part is governed by linear dynamics, i.e. the state. The major drawback of these state-of-the-art algorithms is the loss of soundness and completeness guarantees for the convex relaxation methods and the restrictive convergence sufficient conditions for the gradient-descent ones [6], [12]. Moreover, [7] assumes the time invariance of the attack support, whereas [11] considers the knowledge of a tight upper bound of the number of attacks.

1.3 Decentralization: distributed secure state estimation problem

Given the intrinsic distributed nature of CPSs, conventional optimization algorithms, which we refer to as *centralized*, cannot always be applied to solve optimization problems over cyber-physical networks since they require data management by a single entity [13], called Fusion Center (FC): the measurements acquired by distributed sensors are transmitted to a central high-performing computing device, which collects and processes all the data. The recent trend is to remove the fusion center and decentralize not only the acquisition but also the processing of the data, since in CPSs it is typically undesirable, and at times impossible, to collect them at a unique node [13]. The main reasons why distributed processing is typically preferred over the centralized one are [14]:

- a centralized processing architecture is unfeasible for large-scale networks, i.e. networks with a significant number of sensors, in terms of energy utilization and introduction of delays [15], due to the need of long-range communications to transmit data to the FC;
- a centralized processing architecture is not resilient to a failure in the FC, which would stop the overall processing. Decentralization enhances the network's robustness to the presence of a certain number of sensor faults;
- a decentralized processing architecture preserves agent information privacy: each sensor can keep some information private.

Moreover, the distributed approach is also economically advantageous: the usually expensive and high-performance fusion center is removed, and a network of sensors

with limited memory capacity, performing minimal operations and communicating with each other, is employed.
From these considerations, the *distributed secure state estimation* (DSSE) problem arises: a set of nodes is required to collectively track the state of a linear dynamical system using measurements from their own sensors and messages exchanged with neighboring nodes via a communication network [16], in the presence of adversarial attacks. In contrast to the SSE problem, where the FC has the goal of reconstructing the system state in the presence of attacks on some sensors, having access to global information, in the DSSE problem each sensor only relies on local information, i.e. its own measurements and the one-hop neighbors' messages. Therefore, the DSSE is regarded as more complex, and the algorithms used to solve it are usually sub-optimal with respect to the centralized ones.

1.4 Contributions

The objectives of this thesis include the implementation and testing of algorithms for solving state estimation problems in the presence of sparse sensor attacks, along with performance comparison with respect to a state-of-the-art observer. In particular, an innovative algorithm is developed, named Distributed Sparse Observer, for solving the state estimation problem in a distributed setting, consistently with the intrinsic decentralized nature of CPSs.

1.5 Organization

The thesis is organized as follows.
In Chapter 2, the SSE problem is introduced and analyzed, based on current developments in the literature. The Sparse Observer algorithm is implemented to address this type of problem. Then, a decentralized perspective is employed, and the distributed version of the problem is examined. Drawing inspiration from the centralized version, the Distributed Sparse Observer is developed.
In Chapter 3, the algorithms mentioned above are employed to address an illustrative, synthetic SSE problem, with the aim of comparing their respective performances in various cases of interest. First, the Sparse Observer is compared with the Event-Triggered Projected Luenberger observer, which is considered a standard algorithm in this field of study. Then, the same problem is solved using a decentralized approach and the Distributed Sparse Observer is employed for this purpose.
In Chapter 4, the localization problem is introduced, representing a potential real-world application of growing interest in recent times. It can be formulated as a state estimation problem, suitable to be solved by the distributed version of

the observer. As in Chapter 3, numerical simulations are conducted to compare the results obtained under different possible connection topologies and models of sensor attacks.
Lastly, Chapter 5 is dedicated to final considerations, conclusions, and potential future developments.

Chapter 2

Algorithms

2.1 Secure state estimation problem statement

As in [7] and [11], CPSs are modeled as discrete-time linear time-invariant dynamical systems

$$\begin{aligned} x(k+1) &= Ax(k) \\ y(k) &= Cx(k) + a(k) \end{aligned} \tag{2.1}$$

where $x(k) \in \mathbb{R}^n$ is the system state at time $k \in \mathbb{N}$, $y(k) \in \mathbb{R}^q$ are the observed measurements, $a(k) \in \mathbb{R}^q$ is the attack vector, $A \in \mathbb{R}^{n,n}$ is the system matrix and $C \in \mathbb{R}^{q,n}$ is the sensors measurement matrix. Assuming that each sensor i takes a scalar measurement $y_i(k) \in \mathbb{R}$, the attack vector is defined such that the i^{th} component $a_i(k) \neq 0$ if the corresponding sensor $i \in \{1, \dots, q\}$ is attacked; otherwise, if the i^{th} sensor is not attacked, it follows that $a_i(k) = 0$ and the corresponding output $y_i(k)$ is not corrupted.
Two assumptions are made on the attacks: the first is the sparsity of vector $a(k)$, i.e. no more than $s \ll q$ components of $a(k)$ are non-zero, which can be expressed using the ℓ_0-norm as $\|a(k)\|_0 \leq s$. The other assumption is the time invariance of the set of attacked sensors, i.e. the support of the attack vector is constant over time. This hypothesis is considered realistic when the time required for the malicious agent to take control of a node is significant with respect to the time scale of the estimation algorithm [7].

SSE problem. For some $\tau \leq n$, given matrices A and C, the aim is to estimate the initial state $x(0)$ of the plant from the corrupted observations
$y = (y(0)^T, \dots, y(\tau-1)^T)^T \in \mathbb{R}^{q\tau}$, i.e. in the presence of sparse sensor attacks $a(k)$.

After obtaining the τ vectors $y(0), \dots, y(\tau-1)$, the fusion center uses a decoder $\mathcal{D} : \mathbb{R}^{q\tau} \mapsto \mathbb{R}^n$ to estimate the initial state $x(0)$ of the plant.

Definition 1. *s errors are correctable after τ steps by the decoder $\mathcal{D} : \mathbb{R}^{q\tau} \mapsto \mathbb{R}^n$ if for any $x(0) \in \mathbb{R}^n$, any $K \subset \{1, \dots, q\}$ with $|K| \leq s$, and any sequence of vectors $a(0), \dots, a(\tau-1)$ in $\mathbb{R}^q$ such that $supp(a(k)) \subset K$, we have $\mathcal{D}(y(0), \dots, y(\tau-1)) = x(0)$ where $y(k) = CA^k x(0) + a(k)$ for $k = 0, \dots, \tau - 1$.*

Therefore, s errors are correctable after τ steps, i.e. the system is *resilient* against s attacks after τ steps, if there exists a decoder that can correct s errors after τ steps. The following proposition gives a necessary and sufficient condition for s errors to be correctable:

Proposition 1. *s errors are correctable after τ steps if and only if for all $z \in \mathbb{R}^n \backslash \{0\}$, $|supp(Cz) \cup supp(CAz) \cup \dots \cup supp(CA^{\tau-1}z)| > 2s$.*

The proof can be found in [7]. From Proposition 1, it follows that it is not possible to recover the initial state $x(0)$ if matrix 2.2

$$\mathcal{O} = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{\tau-1} \end{pmatrix} \in \mathbb{R}^{q\tau,n} \tag{2.2}$$

with $\tau = n$, i.e. the observability matrix, has rank smaller than n. In other words, if the system is not observable, then it is also not resilient. Another direct consequence of the proposition is that the number of correctable errors is always less than $q/2$, for any τ, i.e. the number of attacked sensors cannot exceed half of the total number of sensors.
The optimal decoder $\mathcal{D}_0 : \mathbb{R}^{q\tau} \mapsto \mathbb{R}^n$ is defined such that $\mathcal{D}_0(y)$ is the solution of the optimization problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n,\, a\in\mathbb{R}^{q\tau}} \quad & \frac{1}{2}\|a\|_0 \\ \text{s.t.} \quad & a = y - \mathcal{O}x \end{aligned} \tag{2.3}$$

where $\|a\|_0$ denotes the number of nonzero components in a. Solving this problem means searching for the sparsest attack vector a consistent with the measured data

$$y = \begin{pmatrix} y(0) \\ \vdots \\ y(\tau-1) \end{pmatrix} = \begin{pmatrix} \mathcal{O} & I \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \tilde{a} \end{pmatrix}$$

where $\mathcal{O}$ is defined in 2.2, $I \in \mathbb{R}^{q\tau,q\tau}$ is the identity matrix, $\tilde{x} = x(0) \in \mathbb{R}^n$ and $\tilde{a} = \begin{pmatrix} a(0) \\ \vdots \\ a(\tau-1) \end{pmatrix} \in \mathbb{R}^{q\tau}$.

Proposition 2. Assume that s errors are correctable after τ steps, i.e. Proposition 1 holds. Then the decoder $\mathcal{D}_0$ corrects s errors.

The system $a = y - \mathcal{O}x$, reformulated as

$$y = \begin{pmatrix} \mathcal{O} & I \end{pmatrix} \begin{pmatrix} x \\ a \end{pmatrix} \tag{2.4}$$

in the variables $x \in \mathbb{R}^n$ and $a \in \mathbb{R}^{q\tau}$, is underdetermined, since matrix $(\mathcal{O}\ I) \in \mathbb{R}^{q\tau, n+q\tau}$, therefore there are $q\tau$ equations, i.e. measurements, in $n + q\tau$ unknowns, i.e. the initial state and the τ attack vectors. Basic linear algebra states that a system like 2.4 has infinitely many solutions (provided that there exists at least one). Therefore, without additional information it is impossible to recover $(x^T,\ a^T)^T$ from y. The information exploited here is the sparsity of the attack vector a. This can be considered as a compressive sensing problem, which consists in reconstructing an s-sparse vector x from $y = Cx$, where $C \in \mathbb{R}^{q,n}$, with $q < n$ [17]: compressive sensing theory states that an s-sparse vector $x \in \mathbb{R}^n$, with $s \ll n$, can be recovered from compressed linear measurements $y = Cx$, $y \in \mathbb{R}^q$, with $q < n$, if C satisfies certain conditions. The drawback of the $\mathcal{D}_0$ decoder is that the optimization problem 2.3 is non-polynomial (NP) hard, so it can only be solved using a combinatorial approach, due to the use of the ℓ_0-norm. Therefore, this approach is unfeasible. One possible strategy consists in replacing the ℓ_0-norm by the ℓ_1-norm, i.e. its best convex approximation [18].
The decoder $\mathcal{D}_1 : \mathbb{R}^{q\tau} \mapsto \mathbb{R}^n$ is defined such that $\mathcal{D}_1(y)$ is the solution of the following convex optimization problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n,\, a \in \mathbb{R}^{q\tau}} \quad & \frac{1}{2}\|a\|_1 \\ \text{s.t.} \quad & a = y - \mathcal{O}x \end{aligned} \tag{2.5}$$

where $\|a\|_1 = \sum_{i=1}^{q} |a_i|$ denotes the ℓ_1-norm of a. Since the ℓ_1-norm is convex, the optimization problem 2.5, known as *partial basis pursuit* (*partial* since only a part of the unknown vector is sparse) or ℓ_1-minimization, is convex and can be efficiently solved. On the other hand, since it is a relaxation of the original ℓ_0 problem, the decoder $\mathcal{D}_1$ is sub-optimal with respect to the optimal decoder $\mathcal{D}_0$.
Finally, problem 2.5 can be formulated as a *Lasso* (also known as *basis pursuit denoising* problem, introduced by Tibshirani (1996) [19]) to consider the possible presence of measurement noise:

$$\min_{x \in \mathbb{R}^n,\, a \in \mathbb{R}^{q\tau}} \quad \frac{1}{2}\left\| y - \begin{pmatrix} \mathcal{O} & I \end{pmatrix} \begin{pmatrix} x \\ a \end{pmatrix} \right\|_2^2 + \lambda \|a\|_1 \tag{2.6}$$

where $\lambda > 0$. This problem formalizes a trade-off between the accuracy with which $\mathcal{O}x + a$ approximates y, and the complexity of the solution, intended as the number of nonzero entries in a. The larger is the regularization parameter λ, the more problem 2.6 is biased towards finding low-complexity solutions, i.e. solutions with many zeros. Actually, problem 2.6 is a partial Lasso, since only a part of the vector to estimate is sparse, i.e. the attack vector, whereas the state is in general non-sparse. A straightforward and easy-to-implement method for solving Lasso problems is the *iterative shrinkage-thresholding* algorithm, proposed by Daubechies et al. in [20].

2.1.1 Iterative Shrinkage-Thresholding algorithm

Given matrix $C \in \mathbb{R}^{q,n}$, a k-sparse vector $\tilde{x} \in \mathbb{R}^n$ (with $k \ll n$), and noisy measurements $y = C\tilde{x} + \eta$, where $\eta \in \mathbb{R}^q$ is the measurement noise, the following "weighted" Lasso problem is considered:

$$\min_{x \in \mathbb{R}^n} \quad \frac{1}{2}\|y - Cx\|_2^2 + \Lambda^T |x| \tag{2.7}$$

where $\Lambda \in \mathbb{R}^n$ is the vector of regularization parameters, $|x| = (|x_1|, \ldots, |x_n|)^T$ and $\Lambda^T|x|$ is a weighted ℓ_1-norm. One of the most common approaches for solving problems like 2.7 is the class of iterative shrinkage-thresholding (IST) algorithms [21].
Given $x \in \mathbb{R}^n$, the component-wise shrinkage-thresholding operator $\mathbb{S}_\Lambda : \mathbb{R}^n \mapsto \mathbb{R}^n$, for any $\Lambda = (\lambda_1, \ldots, \lambda_n)^T \in \mathbb{R}^n_+$, is defined as:

$$\mathbb{S}_{\lambda_i}(x_i) := \begin{cases} x_i - \lambda_i & \text{if } x_i > \lambda_i \\ x_i + \lambda_i & \text{if } x_i < -\lambda_i \\ 0 & \text{if } |x_i| \le \lambda_i \end{cases}$$

Algorithm 1 IST

```
Input: $C$, $y$, $\gamma > 0$, $\Lambda$
Output: $\hat{x}$ = estimate of $\tilde{x}$
Initialization: $x(1) \in \mathbb{R}^n$, e.g. $x(1) = 0$
1: for $t = 1, \ldots, T_{max}$ do
2:     $x(t+1) = \mathbb{S}_{\gamma\Lambda}[x(t) + \gamma C^T(y - Cx(t))]$
3: end for
4: $\hat{x} = x(T_{max})$
```

where $\gamma \in \mathbb{R}_+$ is the learning rate, defined such that $\gamma < \|C\|_2^{-2}$ to guarantee the convergence to the minimum of the Lasso functional. The convergence of the IST algorithm is proven in [20], [21]. The value of T_{max} is determined by the chosen stopping criterion. The IST algorithm is considered a natural extension of a gradient-based method since each iteration consists of a gradient step and a shrinkage-thresholding step. The main advantage of this algorithm is its ease of implementation. On the other hand, it is acknowledged as a slow method. Its iterative structure makes the IST algorithm easily adaptable for dynamic and distributed systems, as further discussed in the following sections.

2.1.2 Sparse Observer

The SSE problem can be stated in an equivalent dynamic perspective [22].

Online SSE problem. For some $\tau \leq n$, given matrices A and C, the aim is to estimate, at any time instant k, the τ-delayed state $x(k-\tau+1)$ from the corrupted measurements $(y(k-\tau+1)^T, \dots, y(k)^T)^T$. If $\tau = 1$, this is an online (not delayed) SSE.

The aim is to estimate the τ-delayed state $x(k-\tau+1)$ and the constant support of the attack vectors, using the last $q\tau$ measurements, such that at each time instant k the new q measurements $y(k)$ are included, whereas the oldest ones are discarded. Therefore, the value of τ represents the extension of the measurements time window. This problem can be solved by developing an online version of the IST algorithm, here referred to as *Sparse Observer*, where:

$$\mathbf{y}(k) = \begin{pmatrix} y(k-\tau+1) \\ \vdots \\ y(k) \end{pmatrix}, \ \mathbf{a}(k) = \begin{pmatrix} a(k-\tau+1) \\ \vdots \\ a(k) \end{pmatrix}, \ \boldsymbol{\eta}(k) = \begin{pmatrix} \eta(k-\tau+1) \\ \vdots \\ \eta(k) \end{pmatrix}$$

Algorithm 2 Sparse Observer

```
Input: $A$, $\mathcal{O}$, $x_0$, $\gamma > 0$, $\Lambda$, $\tau \leq n$
Output: $\hat{z}(k) = \begin{pmatrix} \hat{x}(k-\tau+1) \\ \hat{\mathbf{a}}(k) \end{pmatrix}$ = estimate of $\begin{pmatrix} x(k-\tau+1) \\ \mathbf{a}(k) \end{pmatrix}$
Initialization: $x(1) = x_0 \in \mathbb{R}^n$, $\hat{z}(\tau) = 0 \in \mathbb{R}^{n+q\tau}$, $G = (\mathcal{O}\ I_{q\tau})$
1: for $k = \tau, \dots, T$ do
2:     $\mathbf{y}(k) = \mathcal{O}x(k-\tau+1) + \boldsymbol{\eta}(k) + \mathbf{a}(k)$ ← System Dynamics
3:     $\hat{z}(k+\frac{1}{2}) = \mathbb{S}_{\gamma\Lambda}[\hat{z}(k) + \gamma G^T(\mathbf{y}(k) - G\hat{z}(k))]$ ← IST step
4:     $\hat{\mathbf{a}}(k+1) = \hat{\mathbf{a}}(k+\frac{1}{2})$
5:     $\hat{x}(k+1) = A\hat{x}(k+\frac{1}{2})$ ← State update
6: end for
```

Algorithm 2 is a Luenberger-like observer since it executes one IST step, i.e. the gradient step, followed by the application of the soft-thresholding operator, at each time instant k. Moreover, it exploits the knowledge of the system dynamics, multiplying the state estimate by A. The IST step can be executed one or more times, but the characteristic of the algorithm is that it is not run to convergence to the Lasso solution. At each time instant k, the Lasso problem can be formulated as:

$$\min_{x\in\mathbb{R}^n,\, a\in\mathbb{R}^{q\tau}} \quad \frac{1}{2}\left\|\mathbf{y}(k) - \begin{pmatrix}\mathcal{O} & I\end{pmatrix}\begin{pmatrix}x\\ a\end{pmatrix}\right\|_2^2 + \Lambda^T\left|\begin{pmatrix}x\\ a\end{pmatrix}\right| \tag{2.8}$$

where, since the state is expected to be non-sparse, the first n components of the regularization parameters vector $\Lambda \in \mathbb{R}^{n+q\tau}$ are null.
If τ is set to 1, the time window collapses, and only the last q measurements $y(k)$ are considered at each instant k. Therefore, in this case the Sparse Observer solves a non-delayed version of the SSE problem, i.e. the current state $x(k)$ is estimated at each k.

2.1.3 Event-Triggered Projected Luenberger observer

For completeness, the *Event-Triggered Projected Luenberger* (ETPL) observer is presented in this section. ETPL, proposed by Shoukry and Tabuada in [11], is one of the state-of-the-art algorithms for solving SSE problems in the absence of measurement noise and with the assumption of knowing a tight upper bound on the number of sensors under adversarial attack.

Algorithm 3 Event-Triggered Projected Luenberger Observer

Input: A, C, x_0, s, $\tau \le n$

Output: $\hat{z}(k) = \begin{pmatrix}\hat{x}(k-\tau+1)\\ \hat{E}(k)\end{pmatrix}$ = estimate of $\begin{pmatrix}x(k-\tau+1)\\ E(k)\end{pmatrix}$

Initialization: $x(1) = x_0 \in \mathbb{R}^n$, $\hat{z}(\tau-1) = 0 \in \mathbb{R}^{n+q\tau}$

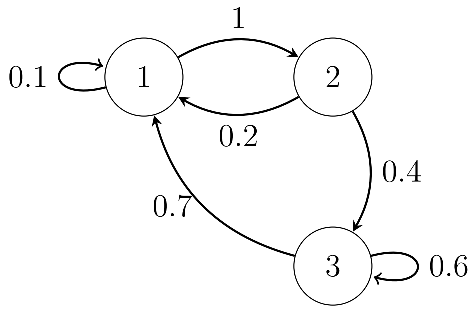
```
1:  for $k = \tau, \dots, T$ do
2:      $x(k+1) = Ax(k)$ ← System Dynamics
3:      $y(k) = Cx(k) + a(k)$
4:      $Y(k) = \mathcal{O}x(k-\tau+1) + E(k)$
5:      $\hat{z}_T(k) = \bar{A}\hat{z}(k-1) + \bar{N}y(k)$ ← Time Update Step
6:      $\hat{z}_\Pi(k) = \Pi(\hat{z}_T(k))$ ← Projection Step
7:      $\hat{z}(k) = \hat{z}_\Pi(k)$
8:      $count = 0$
9:      while $V(\hat{z}_\Pi(k)) \ge (1-\nu)V(\hat{z}_\Pi(k-1))$ & $count < 100$ do
10:         $count = count + 1$
11:         $\hat{z}(k) = \hat{z}_\Pi(k) + L(Y(k) - Q\hat{z}_\Pi(k))$ ← ETPL Update Step
12:         $\hat{z}_\Pi(k) = \Pi(\hat{z}(k))$
13:     end while
14: end for
```

where $\bar{N} = \begin{pmatrix} 0 \\ N_1 \\ \vdots \\ N_q \end{pmatrix}$ and $Y(k)$, $E(k)$, $\bar{A}$, $\mathcal{O}$, $Q = (\mathcal{O}\ I)$, $N_1, \ldots, N_q$ are defined as illustrated in [11]. Π and V, whose definition can be found in [11], are the projection operator and the Lyapunov candidate function, respectively.
The ETPL algorithm implementation considered here differs from that of the paper for the following reasons:

- the system input $u(k)$ is not considered;
- since the restrictive convergence sufficient condition provided in [11] is not satisfied by the systems analyzed in the following chapters, a stopping condition is added to the original one, i.e. the *count* variable. As a consequence, the value of ν is chosen as null, since we have observed that by increasing the value of ν, the while loop condition tends to reduce to the sole stopping condition $count < 100$;
- fixing the Luenberger gain $L = Q^+$, i.e. the Moore-Penrose inverse of Q, the inner loop of the ETPL algorithm in [11] can be replaced with one update step.

2.2 Distributed secure state estimation problem statement

As introduced in Section 1.3, the decentralized nature of CPSs does not always allow having a central server with access to all measurements and data. In a distributed processing setting, each agent aims to solve a global optimization problem, having a partial knowledge of it, i.e. its own measurements and the information exchanged with the neighboring nodes. The DSSE problem can be solved by treating it as a *consensus* problem: the objective of each sensor is to reach consensus on the state of the system [16].
The exchange of information among sensors occurs through a communication network, which is modeled by means of graph theory. A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of vertices (or nodes) $\mathcal{V} = \{1,2,\dots,q\}$ with cardinality q (i.e. the number of sensors) and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, with the property that $(i,i) \in \mathcal{E}$ for all $i \in \mathcal{V}$. The set of neighbors $\mathcal{N}_i$ of a vertex $i \in \mathcal{V}$ is the set of vertices connected to i by an edge, such that $\mathcal{N}(i) = \{j \in \mathcal{V} | (j,i) \in \mathcal{E}\}$. If for each $(j,i) \in \mathcal{E}$, also $(i,j) \in \mathcal{E}$, the graph $\mathcal{G}$ is *undirected*, i.e. all edges are bidirectional. A graph is said to be *strongly connected* if for every pair of nodes (i,j) there exists a path of directed edges that goes from i to j [13], in such a way that there are not disconnected clusters of nodes. Strongly connected graphs and one-hop neighbors communications are considered here.
The topology (or connectivity) of a weighted graph $\mathcal{G}$ is represented by a stochastic matrix $Q \in \mathbb{R}^{q,q}$. A matrix is said to be *stochastic* if $Q_{i,j} \geq 0$ and $\sum_{j=1}^{q} Q_{i,j} = 1$: a stochastic matrix is a non-negative square matrix whose rows sum to unity. The main properties of a stochastic matrix are that all the eigenvalues λ_i are in the unit circle, i.e. $|\lambda_i| \leq 1$, and $Q\mathbf{1} = \mathbf{1}$, i.e. $\mathbf{1}$ is an eigenvector of matrix Q associated with the Perron-Frobenius eigenvalue $|\lambda_{PF}| = 1$, also known as leading eigenvalue. Ordering the eigenvalues by magnitude $1 = |\lambda_{PF}| > |\lambda_2| \geq \cdots \geq |\lambda_q|$, the *essential spectral radius* of a stochastic matrix Q is defined as $esr(Q) = |\lambda_2|$, i.e. the second largest eigenvalue in magnitude.
The matrix Q is said to be *adapted* to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if $Q_{i,j} = 0$ for all $(j,i) \notin \mathcal{E}$. Consequently, it results that $Q_{i,j} > 0$ for all $(j,i) \in \mathcal{E}$ and the value $Q_{i,j}$ represents the reliability of the information coming from node j to node i. Each row Q_i with $i \in \{1,\dots,q\}$ contains the weights of the links incoming to node i. If $Q_{i,j} = Q_{j,i}$ for any (i,j), matrix Q is said to be *doubly-stochastic*, i.e. stochastic and symmetric, and the corresponding graph $\mathcal{G}$ is *undirected*.
As a simple example, the problem could be to estimate the state $\tilde{x} \in \mathbb{R}^n$ given the linear measurements $y = C\tilde{x} + \eta + a \in \mathbb{R}^q$, such that each node stores its own scalar measurement $y_i \in \mathbb{R}$ and C_i, i.e. the i^{th} row of matrix C. In this case, each agent computes a local estimate $x^{(i)} \in \mathbb{R}^n$ of $\tilde{x}$, which is the only information



$$Q = \begin{pmatrix} 0.1 & 0.2 & 0.7 \\ 1 & 0 & 0 \\ 0 & 0.4 & 0.6 \end{pmatrix}, \ \mathcal{N} = \{1,2,3\}, \ \mathcal{E} = \{(1,1), (2,1), (3,1), (1,2), (2,3), (3,3)\}$$

Figure 2.1: Example of a simple topology.

that it can share with the neighboring nodes. Denoting the local estimate of node $i \in \{1, \dots, q\}$ at time k as $x^{(i)}(k) \in \mathbb{R}^n$, the goal is to design a distributed algorithm where each agent updates the local estimate $x^{(i)}(k)$ so that it converges asymptotically to a global estimate $x^* \in \mathbb{R}^n$ of $\tilde{x}$, by means of local computation and neighboring communication only [13], as graphically shown in Figure 2.2. Each

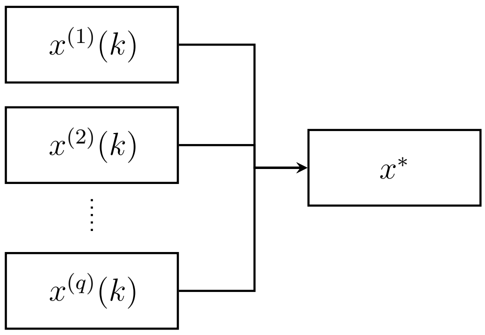


Figure 2.2: Consensus of local state estimates.

local estimate $x^{(i)}(k)$ at time k of the i^{th} sensor can be considered as the i^{th} row of a matrix $X(k)$:

$$X(k) = \begin{pmatrix} x_1^{(1)}(k) & x_2^{(1)}(k) & \dots & x_n^{(1)}(k) \\ x_1^{(2)}(k) & x_2^{(2)}(k) & \dots & x_n^{(2)}(k) \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(q)}(k) & x_2^{(q)}(k) & \dots & x_n^{(q)}(k) \end{pmatrix} \in \mathbb{R}^{q,n}$$

To compute the estimate at iteration $k+1$, each agent i forms a convex combination of its current estimate $x^{(i)}(k)$ with the estimates received from the other agents:

$$X(k+1) = QX(k) \tag{2.9}$$

This *consensus algorithm* can be interpreted as a distributed algorithm:

$$x^{(i)}(k+1) = \sum_{j=1}^{q} Q_{i,j}x^{(j)}(k) = Q_{i,i}x^{(i)}(k) + \sum_{j\in\mathcal{N}(i)\setminus\{i\}} Q_{i,j}x^{(j)}(k) \tag{2.10}$$

From [23], the following results are derived.

Definition 2. Consider equation 2.9. The stochastic matrix Q solves the consensus problem if there exist $\alpha_i \in \mathbb{R}$, $\forall i \in \{1,\dots,n\}$ such that

$$\lim_{k\to\infty} X(k) = \lim_{k\to\infty} Q^k X(0) = \mathbf{1}[\alpha_1\ \alpha_2\ \dots\ \alpha_n] \in \mathbb{R}^{q,n}$$

where $\mathbf{1} \in \mathbb{R}^q$ is a vector of ones.
Q solves the average consensus problem if

$$\lim_{k\to\infty} X(k) = \mathbf{1}[\alpha_1\ \alpha_2\ \dots\ \alpha_n] \text{ with } \alpha_i = \frac{1}{q}\sum_{j=1}^{q} x_i^{(j)}(0)$$

Theorem 1. Let Q be a stochastic matrix, where $\lambda_1 = \lambda_{PF} = 1$ and $\lambda_1 > |\lambda_2| \geq \dots \geq |\lambda_q|$. Then, Q achieves consensus, i.e. solves the consensus problem. If Q is a doubly-stochastic matrix, then Q achieves average consensus.

Theorem 2. The convergence of the consensus algorithm is exponential and the convergence rate is determined by the essential spectral radius $esr(Q) = |\lambda_2|$.

We now focus on solving distributed optimization problems of the form:

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{with } f(x) = \sum_{i=1}^{q} f_i(x) \tag{2.11}$$

where $f_i : \mathbb{R}^n \mapsto \mathbb{R}$, $i = \{1,\dots,q\}$ are convex functions. The goal of the multi-agent system is to solve problem 2.11 in a collaborative way [14]. Each f_i is assumed to be known only by the corresponding agent i. The agents aim to solve a global optimization problem, finding a common solution $x^* = \text{argmin}_{x\in\mathbb{R}^n} \sum_{i=1}^{q} f_i(x)$, but each node has only a partial knowledge of the problem, i.e. the respective function f_i [13]. Each agent i has its own local estimate $x^{(i)} \in \mathbb{R}^n$ that can be shared based on the topology of the connection graph, represented by the matrix Q. Since here the problem of interest is the DSSE, the Lasso problem 2.6 (with $\tau = 1$) is formulated as follows:

$$\min_{x\in\mathbb{R}^n,\, a\in\mathbb{R}^q} \quad \sum_{i=1}^{q} \left[\frac{1}{2}\left\| y_i - \begin{pmatrix} C & I \end{pmatrix}_i \begin{pmatrix} x \\ a \end{pmatrix} \right\|_2^2 + \lambda \|a\|_1\right] \tag{2.12}$$

where $y_i \in \mathbb{R}$ is the scalar measurement of sensor i and $(C\ I)_i$ is the i^{th} row of matrix $(C\ I)$. A distributed version of the IST algorithm is implemented.

2.2.1 Distributed Iterative Shrinkage-Thresholding algorithm

Given matrix $C \in \mathbb{R}^{q,n}$, a k-sparse vector $\tilde{x} \in \mathbb{R}^n$ (with $k \ll n$), and noisy measurements $y = C\tilde{x} + \eta$, where $\eta \in \mathbb{R}^q$ is the measurement noise, the "weighted" Lasso problem 2.7 is rewritten as a distributed convex optimization problem:

$$\min_{x \in \mathbb{R}^n} \quad \sum_{i=1}^{q} \left[\frac{1}{2} \|y_i - C_i x\|_2^2 + \Lambda^T |x| \right] \tag{2.13}$$

Each node i seeks to recover the sparse vector $\tilde{x}$ from its own scalar measurement y_i and to enforce agreement, i.e. consensus, with the estimates computed by other sensors in the network [15]. The DIST algorithm is implemented for this purpose. Compared to the IST, proving the convergence of the DIST algorithm is more complex. A possible proof is proposed in [24], under some specific assumptions. Each iteration consists of a gradient and consensus step, and a shrinkage-thresholding step. In the consensus step, the agent i computes a weighted mean of its neighbors' states, where the weights are given by the i^{th} row of matrix Q, i.e. algorithm 2.10 is executed. The DIST algorithm's ease of implementation is counterbalanced by its slow execution speed, similarly to the IST.

Algorithm 4 DIST

```
Input: $C$, $Q$, $y$, $\gamma > 0$, $\Lambda$
Output: $\hat{x}^{(i)}$ = estimate of $\tilde{x}$ by sensor $i$, $\forall i \in \{1, \dots, q\}$
Initialization: $x^{(i)}(1) \in \mathbb{R}^n$, e.g. $x^{(i)}(1) = 0$, $\forall i \in \{1, \dots, q\}$
1: for $t = 1, \dots, T_{max}$ do
2:     for $i = 1, \dots, q$ do
3:         $x^{(i)}(t+1) = \mathbb{S}_{\gamma\Lambda}\left[\sum_{j=1}^{q} Q_{i,j} x^{(j)}(t) + \gamma C_i^T (y_i - C_i x^{(i)}(t))\right]$
4:     end for
5: end for
6: $\hat{x}^{(i)} = x^{(i)}(T_{max})$
```

where $\gamma \in \mathbb{R}_+$ is the learning rate, defined such that $\gamma < \|C\|_2^{-2}$ to guarantee the convergence to the minimum of the Lasso functional. The value of T_{max} is determined by the chosen stopping criterion.

2.2.2 Distributed Sparse Observer

As the SSE problem in Section 2.1.2, the DSSE problem can also be stated and solved in an equivalent dynamic perspective. The aim of each node is to estimate the current state $x(k)$ and the constant support of the attack vectors, as well as to reach consensus with the other agents' estimates, using the last measurement $y_i(k)$ and the current state of the neighboring nodes, weighted by matrix Q components. This problem can be solved by developing an online version of the DIST algorithm, here referred to as *Distributed Sparse Observer*. This algorithm constitutes an original contribution, not found in the existing literature and developed in this thesis work. It is inspired by the widely recognized centralized version, here referred to as Sparse Observer, adapting it to a decentralized setting. This algorithm does not employ a temporal window of measurements, i.e. $\tau = 1$.

Algorithm 5 Distributed Sparse Observer

```
Input: $A$, $C$, $Q$, $x_0$, $\gamma > 0$, $\Lambda$
Output: $\hat{z}^{(i)}(k) = \begin{pmatrix} \hat{x}^{(i)}(k) \\ \hat{a}^{(i)}(k) \end{pmatrix}$ = estimate of $\begin{pmatrix} x(k) \\ a(k) \end{pmatrix}$ by sensor $i$, $\forall i \in \{1, \dots, q\}$
Initialization: $x(1) = x_0 \in \mathbb{R}^n$, $\hat{z}^{(i)}(1) = 0 \in \mathbb{R}^{n+q}$ $\forall i \in \{1, \dots, q\}$, $G = (C \ I_q)$
1: for $k = 1, \dots, T$ do
2:     $x(k+1) = Ax(k)$ ← System Dynamics
3:     $y(k) = Cx(k) + \eta(k) + a(k)$
4:     for $i = 1, \dots, q$ do
5:         $\hat{z}^{(i)}(k+\frac{1}{2}) = \mathbb{S}_{\gamma\Lambda}\left[\sum_{j=1}^{q} Q_{i,j}\hat{z}^{(j)}(k) + \gamma G_i^T(y_i(k) - G_i\hat{z}^{(i)}(k))\right]$ ← DIST step
6:         $\hat{a}^{(i)}(k+1) = \hat{a}^{(i)}(k+\frac{1}{2})$
7:         $\hat{x}^{(i)}(k+1) = A\hat{x}^{(i)}(k+\frac{1}{2})$ ← State update
8:     end for
9: end for
```

The same considerations done for Algorithm 2 are valid for Algorithm 5: it executes one DIST step at each time instant k and it exploits the knowledge of the system dynamics A. At each time instant k, the Lasso problem can be formulated as:

$$\min_{x \in \mathbb{R}^n,\, a \in \mathbb{R}^q} \quad \sum_{i=1}^{q} \left[\frac{1}{2} \left\| y_i(k) - \begin{pmatrix} C & I \end{pmatrix}_i \begin{pmatrix} x \\ a \end{pmatrix} \right\|_2^2 + \Lambda^T \left| \begin{pmatrix} x \\ a \end{pmatrix} \right| \right] \tag{2.14}$$

Chapter 3

Secure dense state estimation under sparse sensor attacks

3.1 Problem definition

In this chapter, a random, synthetic CPS modeled as a discrete-time linear time-invariant dynamical system is considered:

$$\begin{aligned} x(k+1) &= Ax(k) \\ y(k) &= Cx(k) + \eta(k) + a(k) \end{aligned} \tag{3.1}$$

where $x(k) \in \mathbb{R}^n$, with $n = 10$, is the system state at time $k \in \mathbb{N}$, $y(k) \in \mathbb{R}^q$, with $q = 20$, are the observed measurements, $\eta(k) \in \mathbb{R}^q$ is the random measurement noise vector, such that $\eta(k) \sim \mathcal{N}(0, \sigma^2)$, with $\sigma = 10^{-2}$, and $a(k) \in \mathbb{R}^q$ is the attack vector, defined such that $\|a(k)\|_0 \leq s$, with $s = 2$, and with random uniformly distributed support set constant over time. Moreover, the random components of the attack vector are uniformly distributed in the range $a_i(k) \in [-2, -1] \cup [1, 2]$. This type of attack is known as *unaware*, since it is assumed that the malicious attacker does not know the value of the sensor measurement $y_i(k)$. Initially, time-invariant attacks are considered, such that $a_i(k) = a_i$, $\forall k \in \mathbb{N}$. In Section 3.4, time-varying attacks with a constant random support set are examined. The matrices $A \in \mathbb{R}^{n,n}$, $C \in \mathbb{R}^{q,n}$ and the initial state x_0 are generated independently, according to a standard normal distribution. The system matrix A is then normalized to guarantee stability. The term *dense* state estimation derives from the non-sparse nature of the state vector. Our goal is to compare the performance of the proposed Sparse Observer (Algorithm 2) with that of the state-of-the-art ETPL observer (Algorithm 3) for solving the online SSE problem, and to evaluate the distributed version of the observer, named Distributed Sparse Observer (Algorithm 5).

3.2 Numerical simulations

In this section, the results of the numerical simulations are reported. Each algorithm is executed for 100 runs, each $T = 1000$ time steps long. At each time step $k \in \{1, \dots, T\}$, an estimate $\hat{x}(k)$ of the real state vector $\tilde{x}(k)$, along with an estimate $\hat{a}(k)$ of the real attack vector $\tilde{a}(k)$, are computed. The time evolution of two metrics is evaluated, averaged over the multiple runs [22]:

- the state estimation error $\dfrac{\|\tilde{x}(k) - \hat{x}(k)\|_2}{\|\tilde{x}(k)\|_2}$;
- the attack support error $\sum_{j=1}^{q} |\mathbf{1}(\tilde{a}_j(k) \neq 0) - \mathbf{1}(\hat{a}_j(k) \neq 0)|$, where $\mathbf{1}(v) = 1$ if v is true and 0 otherwise.

In the distributed case, the i^{th} sensor provides an estimate of the state $\hat{x}^{(i)}(k)$ and of the attack vector $\hat{a}^{(i)}(k)$, therefore the metrics are evaluated separately for each sensor. The performances of the centralized and distributed sparse observers are strictly related to the choice of the value of the regularization parameters vector $\Lambda \in \mathbb{R}^{n+q\tau}$ components. Therefore, a tuning of these hyperparameters is performed, before conducting the simulations. Since the state is expected to be non-sparse, the corresponding first n components of Λ are null, i.e. no regularization is needed for the state vector. On the other hand, the components corresponding to the attack vectors must be appropriately chosen. Different λ values are tested and for each of them, 250 runs are executed, each $T = 1000$ time steps long. The optimal value is chosen as the one that provides the smallest mean state estimation error. The regularization term in problem 2.8 (with $\tau = 1$) becomes:

$$\Lambda^T \left| \begin{pmatrix} x \\ a \end{pmatrix} \right| = (\underbrace{0 \ \dots \ 0}_{\in \mathbb{R}^n} \ \underbrace{\lambda \ \dots \ \lambda}_{\in \mathbb{R}^q}) \left| \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ a_1 \\ \vdots \\ a_q \end{pmatrix} \right| = 0 \cdot |x_1| + \dots + 0 \cdot |x_n| + \lambda \cdot |a_1| + \dots + \lambda \cdot |a_q|$$

The same applies to the $\tau > 1$ case.
Furthermore, to improve the rate of attack detection, defined as the number of times that the support of the attack vector is correctly estimated, which means that the sensors under attack are identified, the nonzero components of the attack vector

estimate $\hat{a}(k)$ below a certain scalar threshold are replaced with zeros. Therefore, the value of this threshold must also be tuned.

$$\hat{a}_i(k) = \begin{cases} \hat{a}_i(k) & \text{if } |\hat{a}_i(k)| > \hat{a}_{thres} \\ 0 & \text{if } |\hat{a}_i(k)| \leq \hat{a}_{thres} \end{cases} \tag{3.2}$$

where $\hat{a}_{thres} \in \mathbb{R}$.

3.2.1 Sparse Observer

The performance of the Sparse Observer is compared with that of the ETPL observer for three different values of the time window length τ. The learning rate γ is defined as $\gamma = \|(\mathcal{O}\ I)\|_2^{-2} - 1 \cdot 10^{-8}$, to guarantee the convergence to the minimum of the Lasso functional. Table 3.1 shows the chosen values, after appropriate tuning, of the regularization parameter and of the attack estimate threshold for the Sparse Observer algorithm. Figure 3.1 shows how the regularization parameter λ is chosen for $\tau = 1$ (the same procedure is used for $\tau = 2$ and $\tau = 10$). As regards the ETPL algorithm, $\hat{a}_{thres}$ is set to 0.2 for any τ.

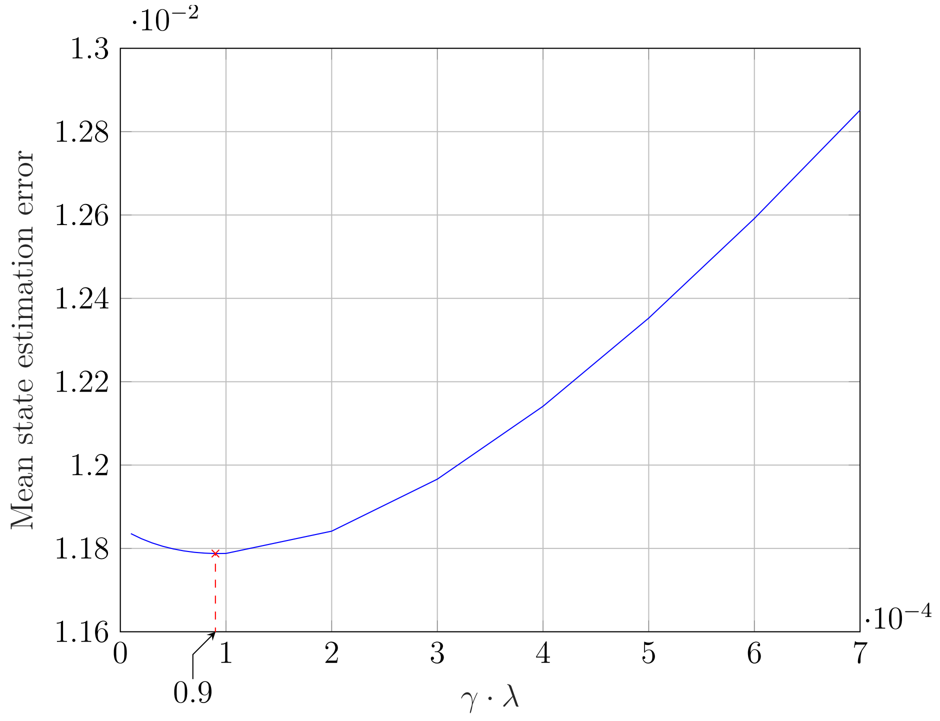


Figure 3.1: Sparse observer λ tuning for $\tau = 1$.

	Sparse Observer		
	$\tau = 1$	$\tau = 2$	$\tau = 10$
$\gamma \cdot \lambda$	$9 \cdot 10^{-5}$	$1 \cdot 10^{-4}$	$2 \cdot 10^{-5}$
$\hat{a}_{thres}$	0.2	0.2	0.02

Table 3.1: Sparse Observer λ and $\hat{a}_{thres}$ values for different time window lengths.

The results of the numerical simulations are shown in Figures 3.2, 3.3, 3.4.

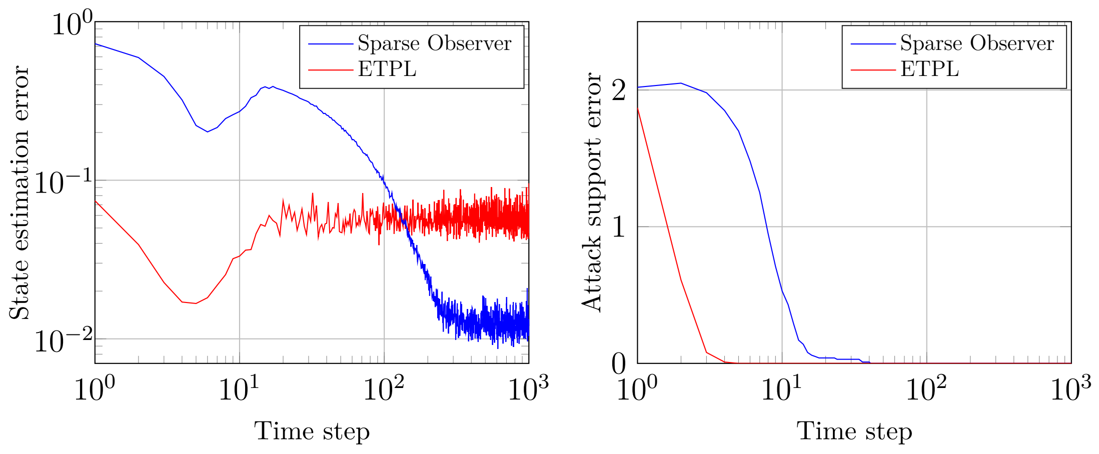


Figure 3.2: Sparse Observer vs ETPL with $n = 10$, $q = 20$, $s = 2$, $\tau = 1$.

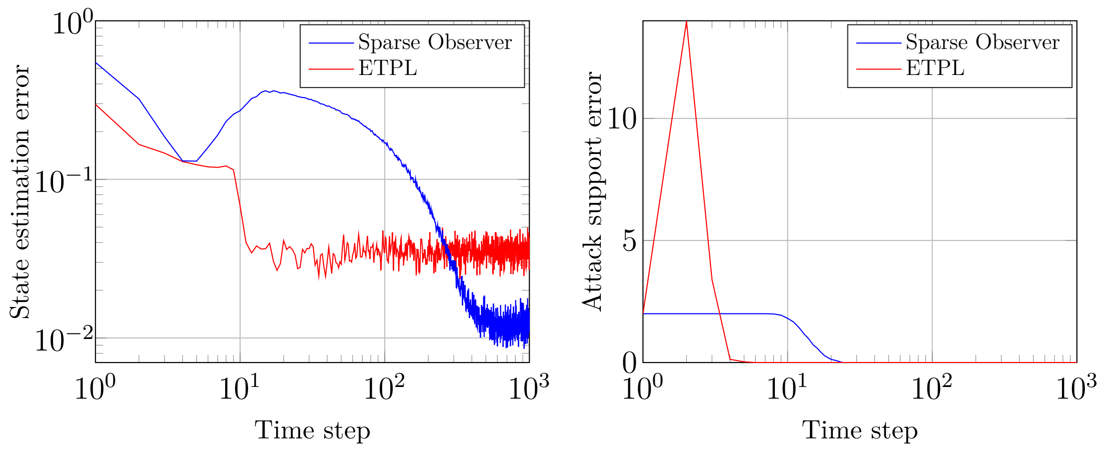


Figure 3.3: Sparse Observer vs ETPL with $n = 10$, $q = 20$, $s = 2$, $\tau = 2$.

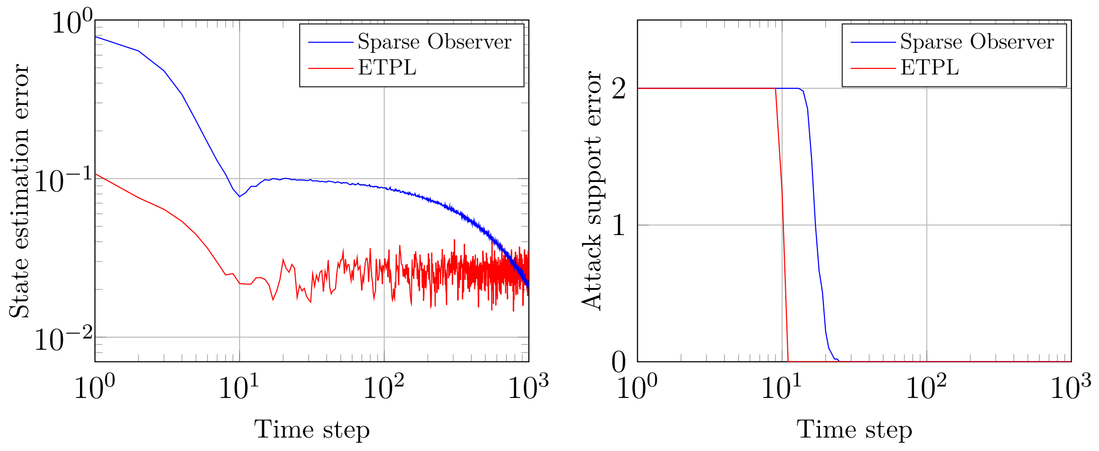


Figure 3.4: Sparse Observer vs ETPL with $n = 10$, $q = 20$, $s = 2$, $\tau = 10$.

Table 3.2 shows the mean value over the $T = 1000$ time instants of the state estimation error.

	Mean state estimation error		
	$\tau = 1$	$\tau = 2$	$\tau = 10$
Sparse Observer	0.0384	0.0505	0.0532
ETPL	0.0570	0.0363	0.0261

Table 3.2: Mean state estimation error over time for different time window lengths.

For $\tau = 1$, i.e. considering only the last q measurements $y(k)$ at each time instant k to produce an online estimate, the Sparse Observer outperforms ETPL in terms of state estimation error, whereas ETPL converges faster to the correct attack support, even though both exhibit satisfactory performance in attack detection. Increasing the time window extension to $\tau = 2$ and to $\tau = 10$, the mean state estimation error increases for the Sparse Observer, while decreases for ETPL. The Sparse Observer performance degradation in state estimation for greater τ values is due to the fact that employing more measurements also means incorporating more attacks. On the other hand, this behavior does not hold for ETPL, which, on the contrary, improves the convergence of the state estimate to the real dense state for growing values of τ, thanks to the intrinsic structure of this observer, which allows it to take advantage of using more measurements. As regards the attack support error, the same considerations made for the $\tau = 1$ case can be extended to $\tau = 2$ and $\tau = 10$: ETPL converges faster to the true attack vector, but nonetheless, the Sparse Observer provides good attack detection performance.

3.2.2 Distributed Sparse Observer

In this section, the same online SSE problem (with $\tau = 1$) is solved using a distributed approach, considered more suitable to the intrinsic decentralized architecture of CPSs. Since distributed methods require the exchange of information among sensors, a communication network is needed. The connection graph topology considered here is represented by a doubly-stochastic matrix $Q \in \mathbb{R}^{20,20}$, such that each node i has 9 incoming (and 9 outgoing) edges, including the self-loop, i.e. each row Q_i of matrix Q has 9 nonzero elements, $\sum_{j=1}^{q} Q_{i,j} = 1$ and $Q = Q^T$. This standard topology matrix is indicated as $Q^{[9]}$ to distinguish it from the other topologies examined in Section 3.3.
A looser definition of consensus is employed for the estimates provided by the $q = 20$ different nodes. At time instant k, each sensor $i \in \{1, \dots, q\}$ computes

$$\hat{z}^{(i)}(k) = \begin{pmatrix} \hat{x}^{(i)}(k) \\ \hat{a}^{(i)}(k) \end{pmatrix} = \text{estimate of} \begin{pmatrix} x(k) \\ a(k) \end{pmatrix} \tag{3.3}$$

The consensus time instant t_{cons} is defined as the time instant k from which the following two conditions are satisfied:

1. the state estimates $\hat{x}^{(i)}(k) \in \mathbb{R}^n$, $\forall i \in \{1, \dots, q\}$ are such that, for each component $j \in \{1, \dots, n\}$:

$$\max_i \hat{x}_j^{(i)}(k) - \min_i \hat{x}_j^{(i)}(k) \leq 5 \cdot 10^{-2} \tag{3.4}$$

2. the attack vector estimates $\hat{a}^{(i)}(k)$, $\forall i \in \{1, \dots, q\}$ have the same support set.

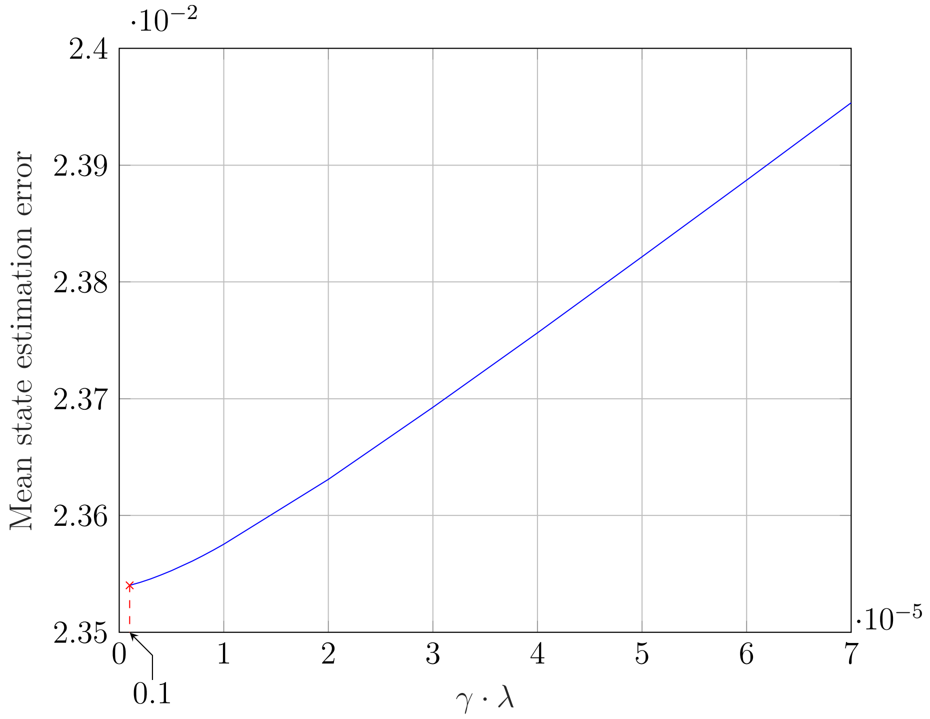
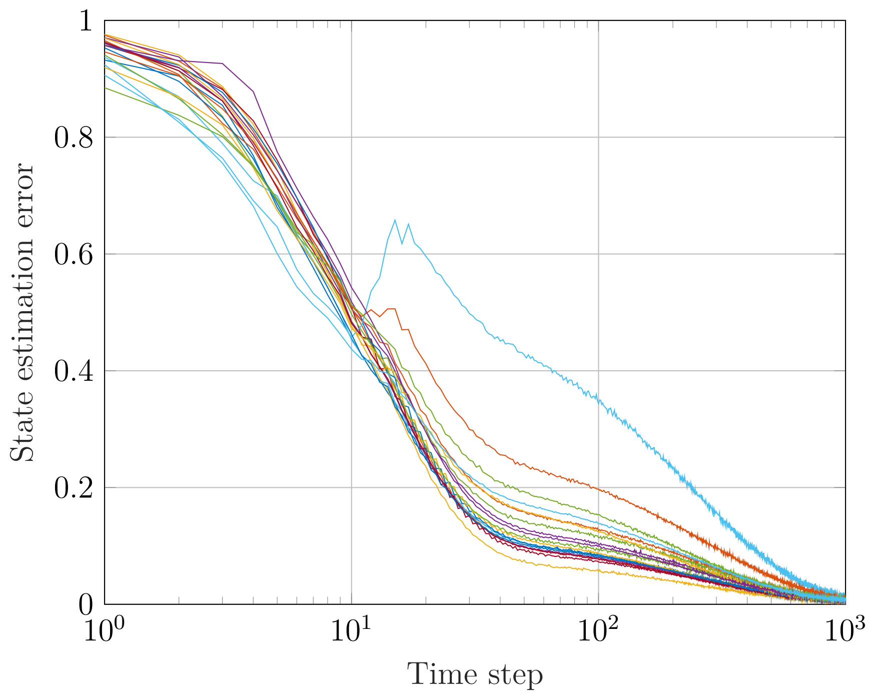


Figure 3.5: Distributed Sparse observer λ tuning.

The learning rate γ is defined as $\gamma = \min_i \|(C\ I)_i\|_2^{-2} - 1 \cdot 10^{-8}$, whereas hyperparameter tuning leads to $\lambda = \dfrac{1 \cdot 10^{-6}}{\gamma}$ (Figure 3.5) and $\hat{a}_{thres} = 0.1$.
In Figure 3.6, the results of the numerical simulations are illustrated; each line corresponds to a specific sensor.



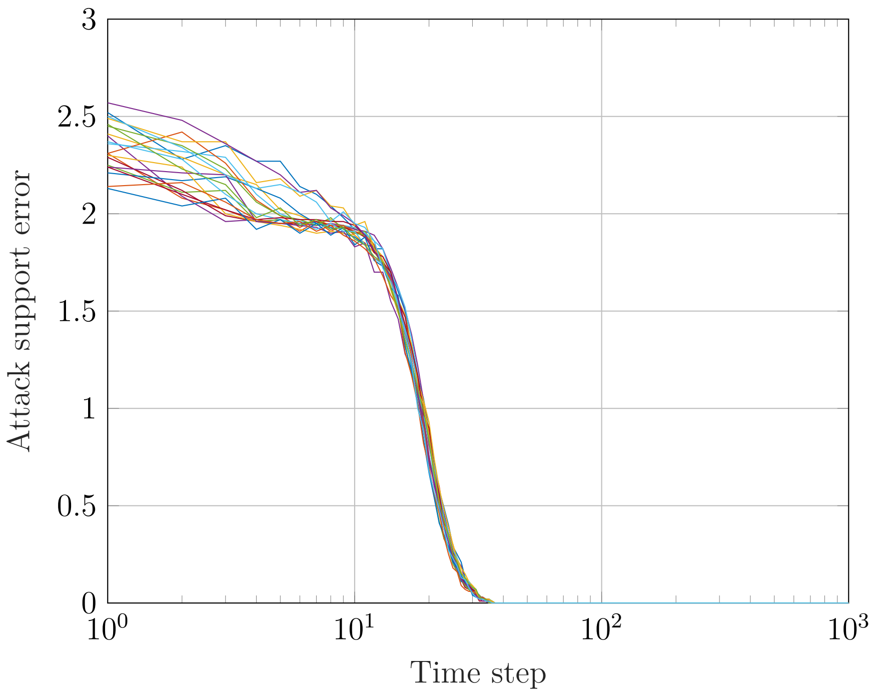


Figure 3.6: Distributed Sparse Observer with $n = 10$, $q = 20$, $s = 2$, standard topology $Q = Q^{[9]}$. Each line corresponds to a specific sensor.

3.3 Topology analysis

In this section, different topologies from the standard one are considered, with the aim of determining how the topology affects the performance of the distributed algorithm by varying the number of connections among sensors. The evaluated topologies are the following, ordered based on the increasing number of connections per node:

- $Q = Q^{[5]}$, doubly-stochastic matrix such that each node i has 5 incoming (and 5 outgoing) edges, including the self-loop. Each sensor is connected to 4 different nodes, which correspond to 20% of the total number of sensors.
- $Q = Q^{[9]}$, doubly-stochastic matrix such that each node i has 9 incoming (and 9 outgoing) edges, including the self-loop (standard topology defined and employed in Section 3.2.2). Each sensor is connected to 8 different nodes, which correspond to 40% of the total number of sensors.
- $Q = Q^{[13]}$, doubly-stochastic matrix such that each node i has 13 incoming (and 13 outgoing) edges, including the self-loop. Each sensor is connected to 12 different nodes, which correspond to 60% of the total number of sensors.

Since in the distributed case each sensor provides its own estimate, the state estimation error and the attack support error must be averaged twice to obtain a scalar mean value: first with respect to the T time instants and then with respect to the number q of sensors. Table 3.3 shows the corresponding results for each topology Q.

	Mean state estimation error	Mean attack support error
$Q = Q^{[5]}$	0.0558	0.0415
$Q = Q^{[9]}$	0.0532	0.0376
$Q = Q^{[13]}$	0.0521	0.0363

Table 3.3: Mean state estimation error and mean attack support error over time and over $q = 20$ sensors, for each topology.

It follows that, as expected, the mean errors decrease for an increasing number of connections among agents. In particular, the performance improves more using 9 instead of 5 connections than going from 9 to 13 connections. In fact, the mean state estimation error decreases by 4.7% from $Q^{[5]}$ to $Q^{[9]}$ and by 2.1% from $Q^{[9]}$ to $Q^{[13]}$. Analogously, the mean attack support error diminishes by 9.4% from $Q^{[5]}$ to

$Q^{[9]}$ and by 3.5% from $Q^{[9]}$ to $Q^{[13]}$. Figures 3.7 and 3.8 illustrate the results for $Q = Q^{[5]}$ and $Q = Q^{[13]}$, respectively.

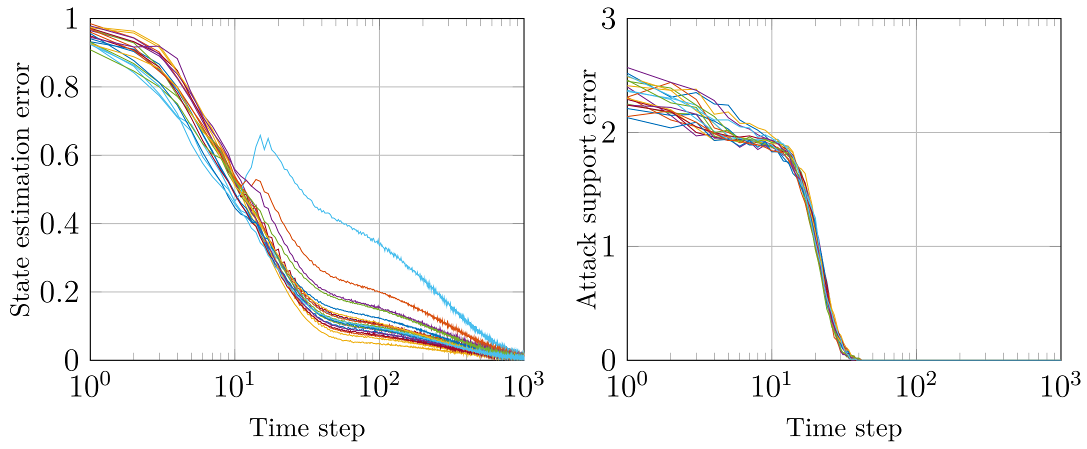


Figure 3.7: Distributed Sparse Observer with $n = 10$, $q = 20$, $s = 2$, $Q = Q^{[5]}$. Each line corresponds to a specific sensor.

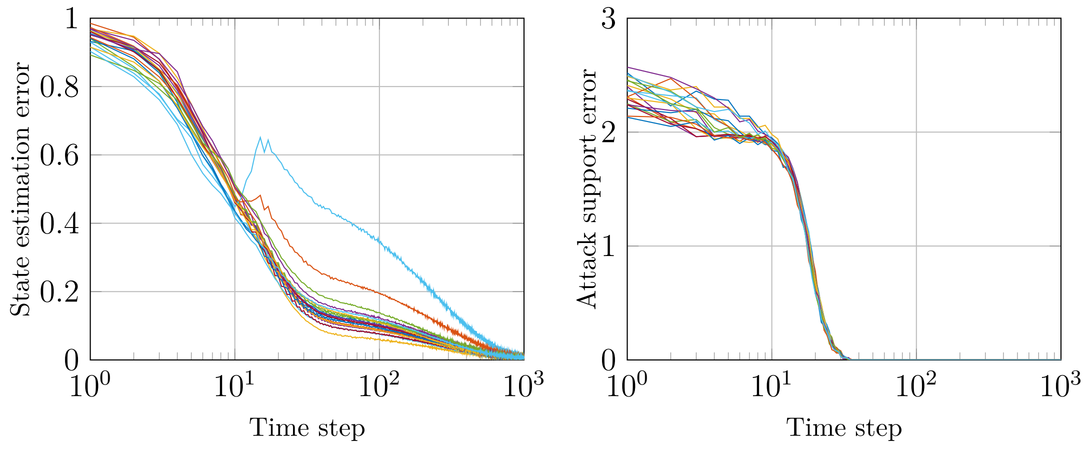


Figure 3.8: Distributed Sparse Observer with $n = 10$, $q = 20$, $s = 2$, $Q = Q^{[13]}$. Each line corresponds to a specific sensor.

To further investigate the performance differences arising from varying the number of connections among sensor nodes, additional metrics are employed:

- ϕ_{cons} represents the fraction of runs that achieve consensus within $T = 1000$ time instants with respect to the total number of runs.
- $t_{cons}^{\max}$ denotes the maximum consensus time instant t_{cons} among all the runs.

- t_{cons}^{mean} is the mean consensus time instant t_{cons} over all the runs.
- MRAD stands for Mean Rate of Attack Detection and denotes the mean value over the runs of the rate of attack detection, i.e. the number of times that the support of the attack vector is correctly estimated after consensus among nodes estimates is reached.

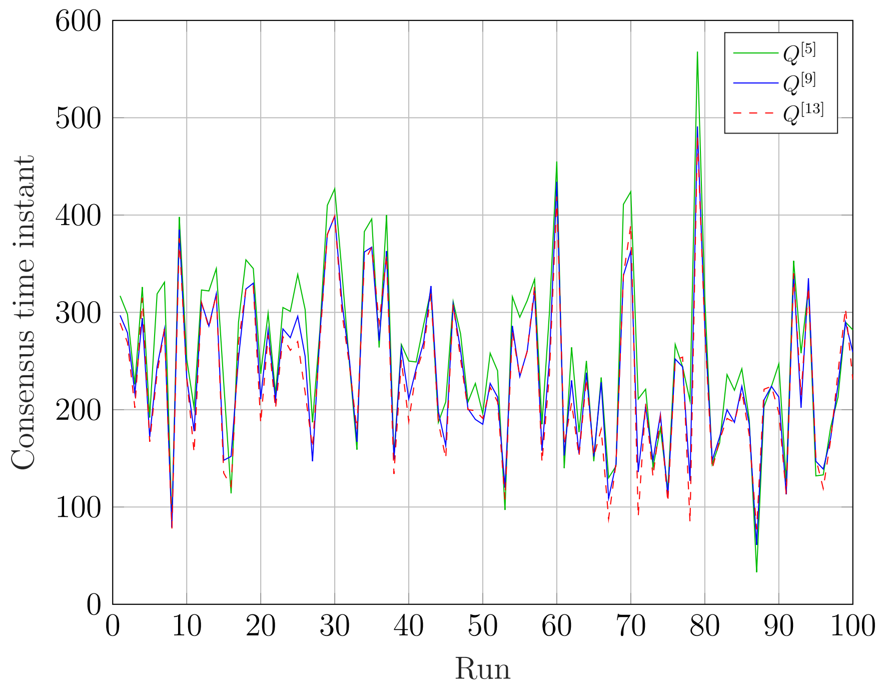
	ϕ_{cons}	t_{cons}^{max} (s)	t_{cons}^{mean} (s)	MRAD
$Q = Q^{[5]}$	1	568	256.84	744.16
$Q = Q^{[9]}$	1	491	237.08	763.92
$Q = Q^{[13]}$	1	482	231.32	769.68

Table 3.4: Distributed Sparse Observer performance comparison between different topologies.

Firstly, $\phi_{cons} = 1$ for each topology matrix Q examined, which means that consensus is achieved within T in all the 100 runs. Then:

- t_{cons}^{max} decreases by 13.6% from $Q^{[5]}$ to $Q^{[9]}$ and by 1.8% from $Q^{[9]}$ to $Q^{[13]}$;
- t_{cons}^{mean} decreases by 7.7% from $Q^{[5]}$ to $Q^{[9]}$ and by 2.4% from $Q^{[9]}$ to $Q^{[13]}$;
- MRAD increases by 2.7% from $Q^{[5]}$ to $Q^{[9]}$ and by 0.8% from $Q^{[9]}$ to $Q^{[13]}$;

The same considerations made for Table 3.3 are valid for Table 3.4. Therefore, we can conclude that the number of connections given by matrix $Q^{[9]}$, i.e. 8 neighboring nodes for each sensor plus the self-loop, is sufficient to have satisfactory results, both in terms of estimation and of consensus among agents. In Figure 3.9, the consensus time instant t_{cons} and the rate of attack detection are represented as a function of the specific execution to graphically visualize the performance differences among the three topologies. From these two graphs, it is possible to notice that the rate of attack detection is complementary to the consensus time instant with respect to T: once consensus among the estimates computed by the q agents is reached, the sensors under attack are correctly detected at each time instant $k \in \{t_{cons}, \dots, T\}$ for each of the 100 executions. The same conclusion can be derived from the corresponding mean values reported in Table 3.4.



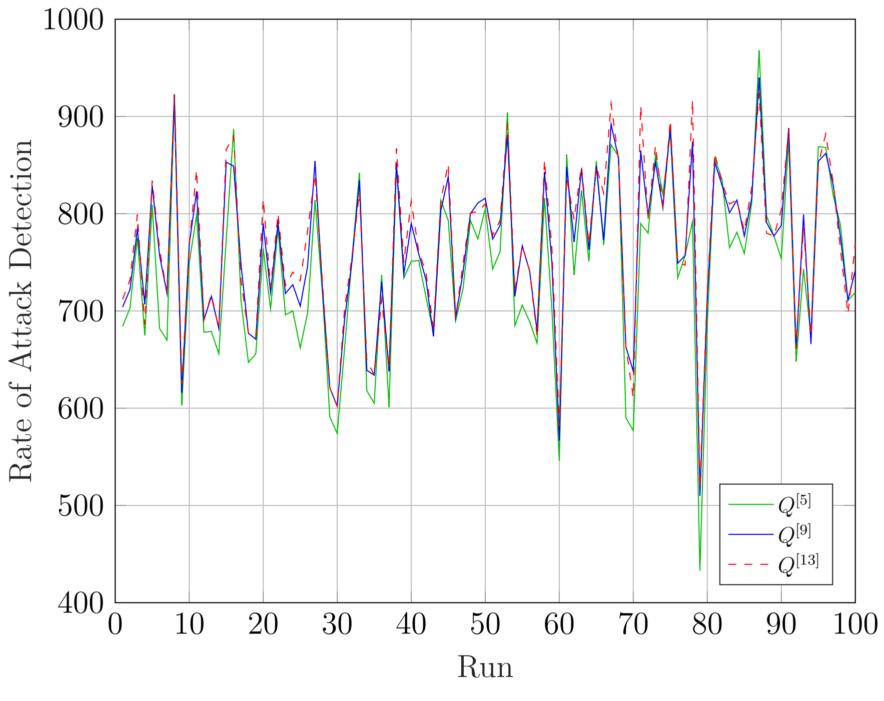


Figure 3.9: Consensus time instant t_{cons} and rate of attack detection for each run, topologies comparison.

3.4 Time-varying attacks

Until now, the malicious attacks in the CPS model 3.1 have been considered as time-invariant with constant support. In this section, time-varying adversarial attacks $a(k)$ with constant support are employed, in order to examine the developed algorithms performance differences with the previous case. In practice, the time-varying attack vector $a(k)$ is generated by taking the $s = 2$ nonzero components of the previously created attack vector $a_i \in [-2, -1] \cup [1, 2]$ and adding a different uniformly distributed random number in the range $[-0.3, 0.3]$ for each time instant $k \in \{1, \ldots, T\}$. A possible time evolution of the time-varying sensor attacks is represented in Figure 3.10.

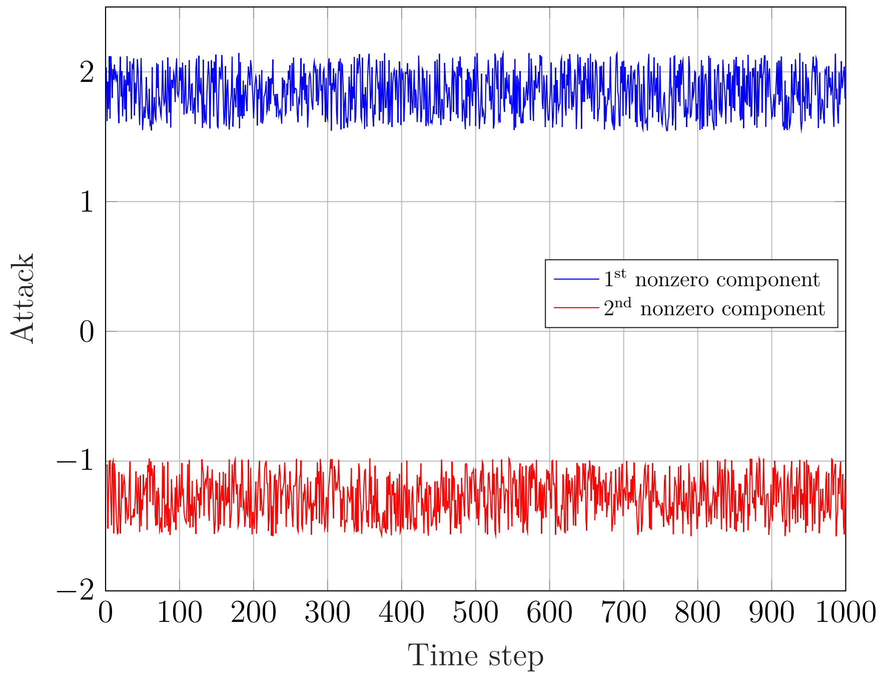


Figure 3.10: Example of the time evolution of the time-varying sensor attacks.

3.4.1 Sparse Observer

The Sparse Observer performance in the presence of time-varying attacks is compared with the results obtained in Section 3.2.1 for constant attacks for three different values of the time window length τ (Figures 3.11, 3.12, 3.13). Moreover, Table 3.5 illustrates the mean state estimation error of Sparse Observer and ETPL employing the time-varying attack model. The regularization parameters vector Λ and the attack estimate threshold $\hat{a}_{thres}$ maintain the same values illustrated in Table 3.1.

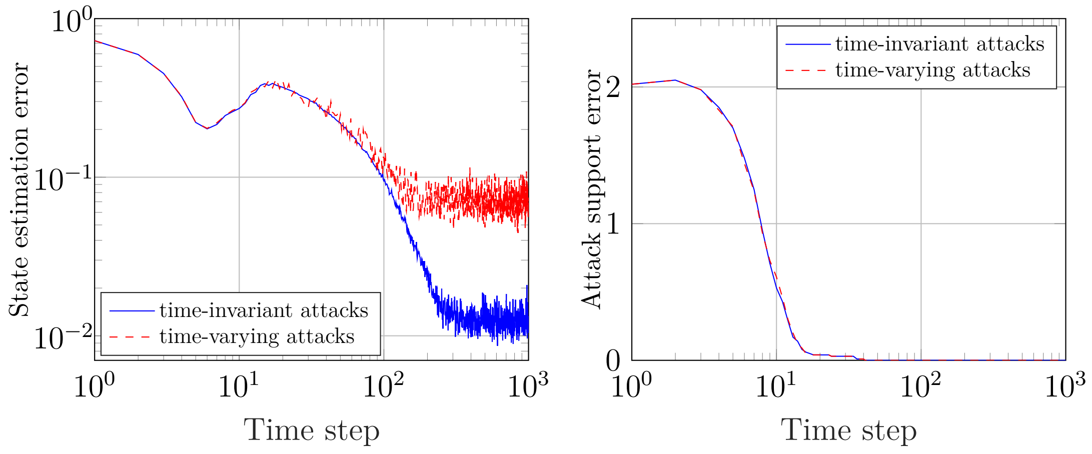


Figure 3.11: Sparse Observer, time-invariant vs time-varying attacks with $n = 10$, $q = 20$, $s = 2$, $\tau = 1$.

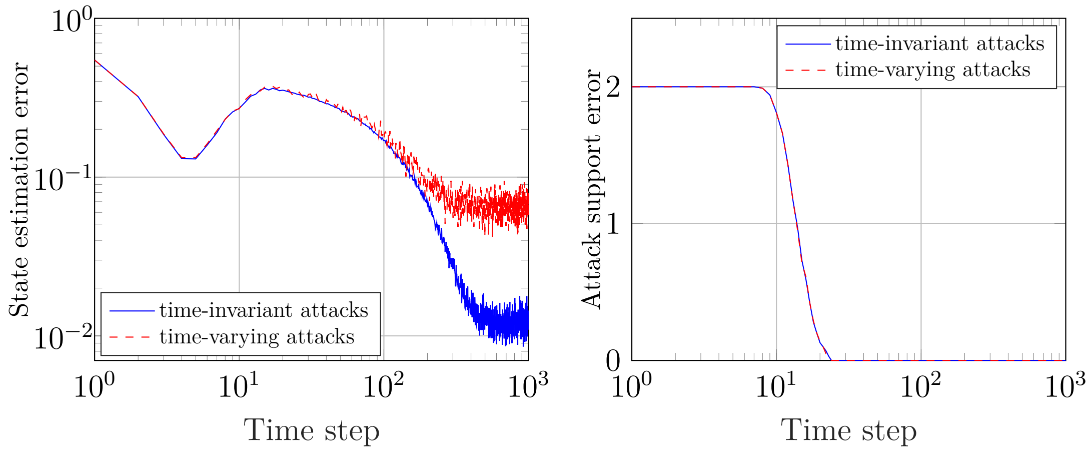


Figure 3.12: Sparse Observer, time-invariant vs time-varying attacks with $n = 10$, $q = 20$, $s = 2$, $\tau = 2$.

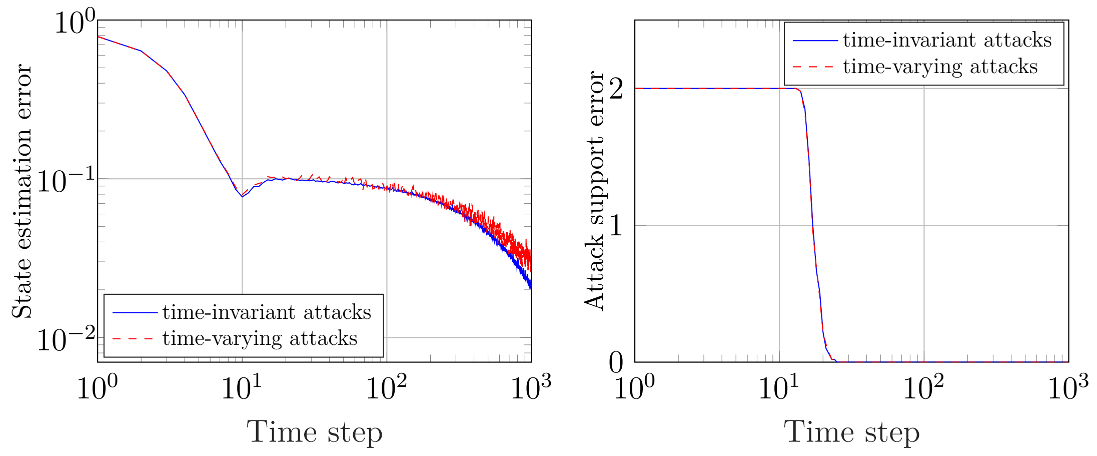


Figure 3.13: Sparse Observer, time-invariant vs time-varying attacks with $n = 10$, $q = 20$, $s = 2$, $\tau = 10$.

	Mean state estimation error		
	$\tau = 1$	$\tau = 2$	$\tau = 10$
Sparse Observer	0.0902	0.0932	0.0586
ETPL	0.0570	0.0362	0.0261

Table 3.5: Mean state estimation error over time for different time window lengths (time-varying attacks case).

In the time-varying attacks case, the ETPL observer outperforms the Sparse Observer in terms of state estimation error for any value of $\tau \in \{1, 2, 10\}$. In fact, while the results using ETPL are almost identical in the two attack model cases, the Sparse Observer is found to be non-robust to non-constant attacks for small values of τ. In particular, the mean state estimation error increases by 135% for $\tau = 1$ and by 84.6% for $\tau = 2$, with respect to the values in Table 3.2. On the other hand, for $\tau = 10$ the mean estimation error increases by just 10.2% with respect to the constant attacks case. This means that the Sparse Observer becomes more robust to the presence of time-varying attacks by extending the measurement time window length.
As regards the attack estimate results, the figures above demonstrate that the Sparse Observer attack support error presents almost the same trend for the two types of attacks. To further emphasize this aspect, Figures 3.14, 3.15, 3.16 display the rate of attack detection as a function of the run, for $\tau = 1$, $\tau = 2$ and $\tau = 10$, respectively, and Table 3.6 contains the corresponding mean values. The same

conclusion can be drawn, as the variation of the attacks over time does not affect the Sparse Observer ability to detect the attacked sensors.

	Mean rate of attack detection		
	$\tau = 1$	$\tau = 2$	$\tau = 10$
Time-invariant attacks	989.80	984.30	981.87
Time-varying attacks	989.75	984.31	981.90

Table 3.6: Mean rate of attack detection over the runs for different time window lengths, time-invariant vs time-varying attacks (Sparse Observer).

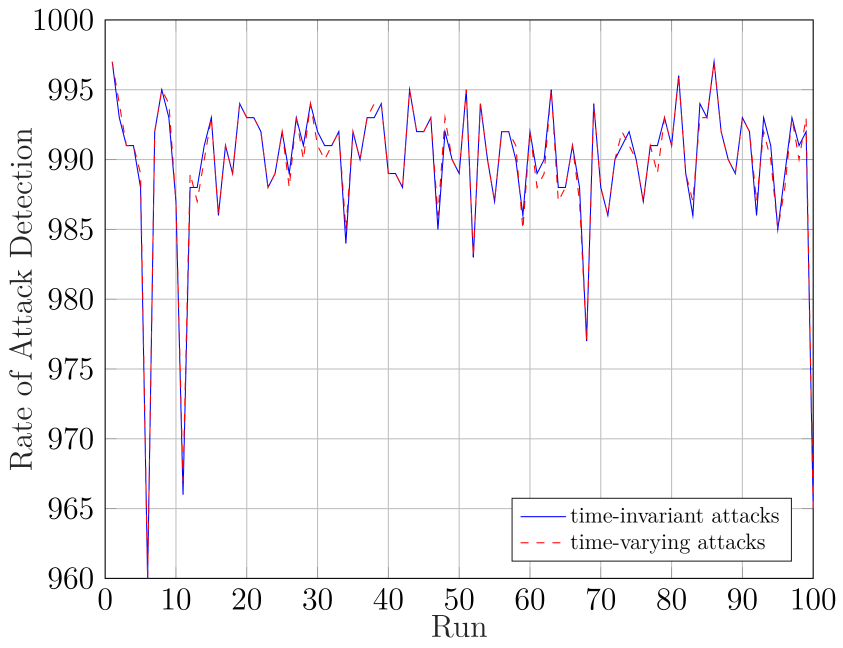


Figure 3.14: Rate of attack detection for each run with $\tau = 1$, time-invariant vs time-varying attacks (Sparse Observer).

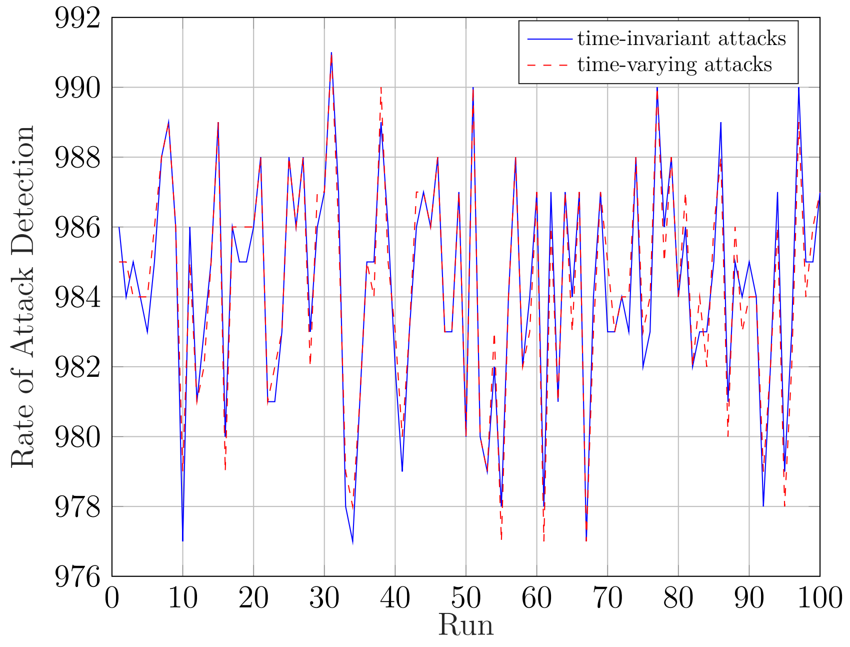


Figure 3.15: Rate of attack detection for each run with $\tau = 2$, time-invariant vs time-varying attacks (Sparse Observer).

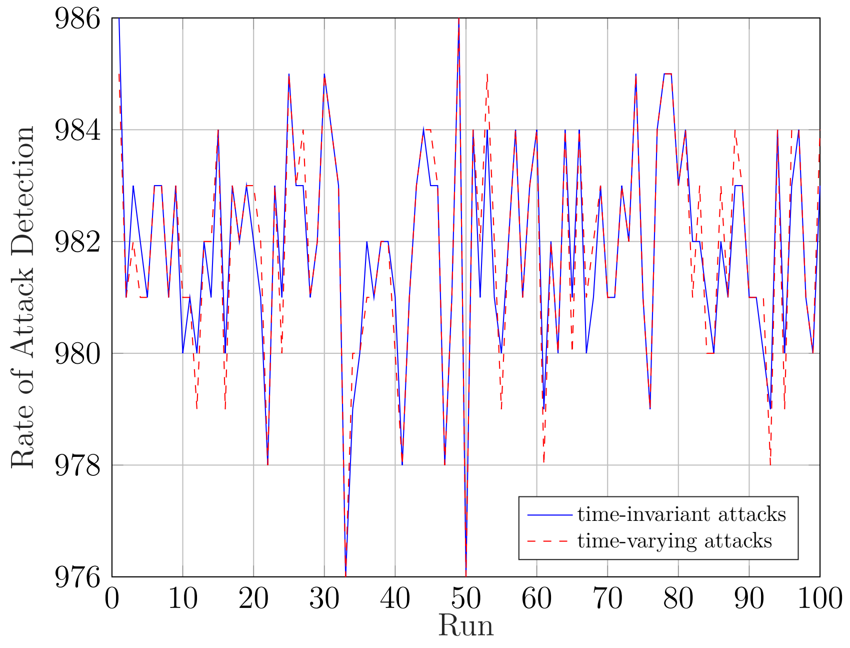


Figure 3.16: Rate of attack detection for each run with $\tau = 10$, time-invariant vs time-varying attacks (Sparse Observer).

3.4.2 Distributed Sparse Observer

The Distributed Sparse Observer algorithm is tested with time-varying sensor attacks for the three different connection topologies Q introduced in Section 3.3. The results of the numerical simulations are shown in Figures 3.17, 3.18, 3.19. Table 3.7 illustrates the respective mean values, whereas Table 3.8 contains the additional metrics defined to investigate the consensus performance. The regularization parameters vector Λ and the attack estimate threshold $\hat{a}_{thres}$ maintain the same values illustrated in Section 3.2.2. For any Q, the mean state estimation error

	Mean state estimation error	Mean attack support error
$Q = Q^{[5]}$	0.0656	0.0414
$Q = Q^{[9]}$	0.0632	0.0376
$Q = Q^{[13]}$	0.0629	0.0363

Table 3.7: Mean state estimation error and mean attack support error over time and over $q = 20$ sensors, for each topology (time-varying attacks case).

increases from the time-invariant to the time-varying attacks case: by 17.6% for $Q^{[5]}$, by 18.8% for $Q^{[9]}$, and by 20.7% for $Q^{[13]}$. As concerns the mean attack support error, the values are essentially unchanged. These considerations are graphically supported by Figures 3.17, 3.18, 3.19.

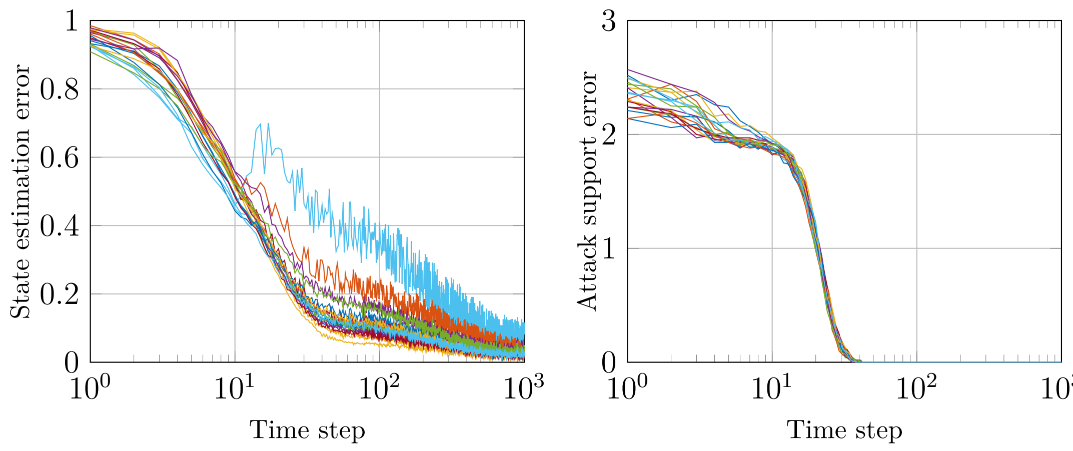


Figure 3.17: Distributed Sparse Observer, time-varying attacks with $n = 10$, $q = 20$, $s = 2$, $Q = Q^{[5]}$. Each line corresponds to a specific sensor.

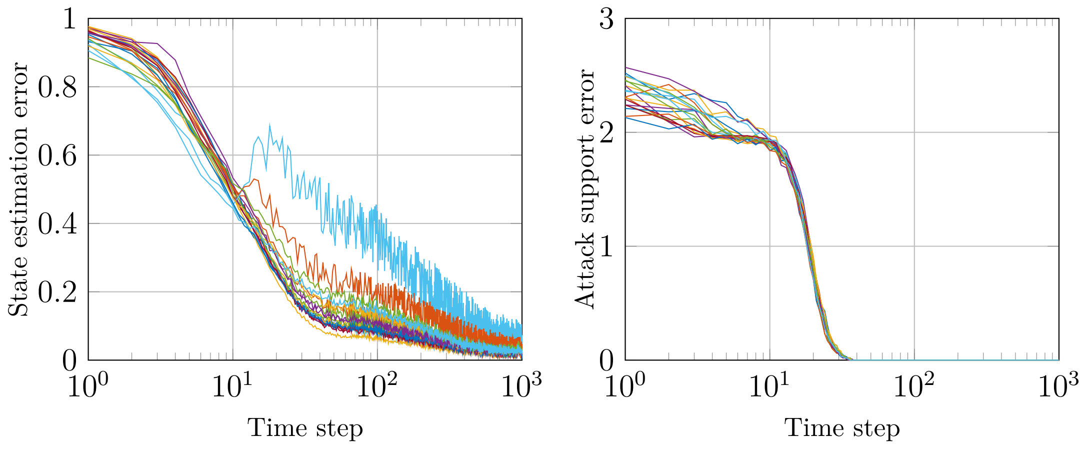


Figure 3.18: Distributed Sparse Observer, time-varying attacks with $n = 10$, $q = 20$, $s = 2$, $Q = Q^{[9]}$. Each line corresponds to a specific sensor.

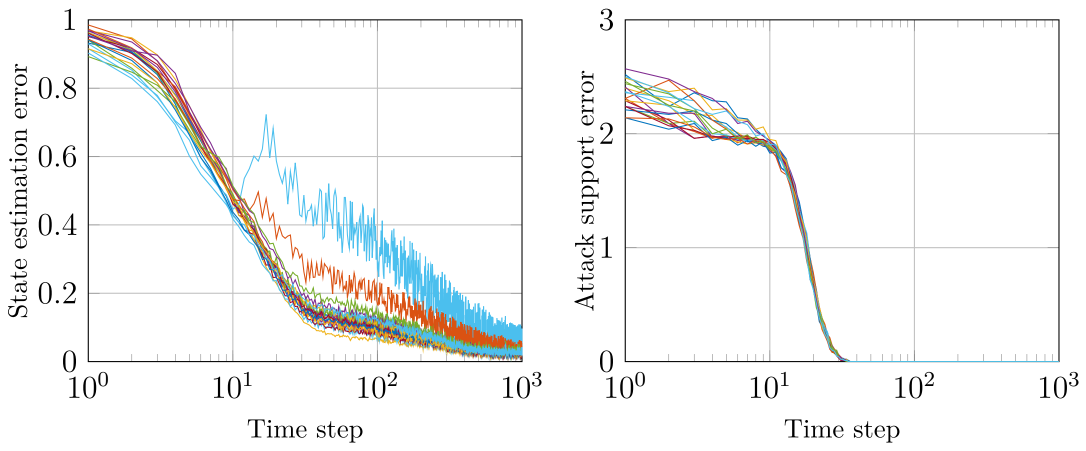


Figure 3.19: Distributed Sparse Observer, time-varying attacks with $n = 10$, $q = 20$, $s = 2$, $Q = Q^{[13]}$. Each line corresponds to a specific sensor.

From Table 3.8, $\phi_{cons} = 1$ for any topology matrix Q examined, which means that consensus is achieved within T in all the 100 runs, as in Table 3.4. As regards the other metrics, they all present worse values with respect to the constant attacks case:

- $t_{cons}^{\max}$ increases by 75.4% for $Q^{[5]}$, by 102.9% for $Q^{[9]}$ and by 98.1% for $Q^{[13]}$;
- t_{cons}^{mean} increases by 76% for $Q^{[5]}$, by 72.8% for $Q^{[9]}$ and by 69.5% for $Q^{[13]}$;
- MRAD decreases by 26.2% for $Q^{[5]}$, by 22.6% for $Q^{[9]}$ and by 20.9% for $Q^{[13]}$;

	ϕ_{cons}	$t_{cons}^{\max}$ (s)	t_{cons}^{mean} (s)	MRAD
$Q = Q^{[5]}$	1	996	451.95	549.05
$Q = Q^{[9]}$	1	996	409.55	591.45
$Q = Q^{[13]}$	1	955	392.01	608.99

Table 3.8: Distributed Sparse Observer performance comparison between different topologies (time-varying attacks case).

In Figure 3.20, the consensus time instant t_{cons} and the rate of attack detection are represented as a function of the specific execution to graphically visualize the performance differences among the two different types of attacks for the standard topology $Q = Q^{[9]}$ (the same trends are observed for the other topologies as well). Also in the non-constant attacks case, the rate of attack detection is complementary to the consensus time instant with respect to T: the sensors under attack are correctly detected at each time instant $k \in \{t_{cons}, \dots, T\}$ for each of the 100 executions. The same conclusion can be derived from the corresponding mean values reported in Table 3.8.
However, it is possible to notice that the mean rate of attack detection decline from the time-invariant to the time-varying attacks case, clearly evident in Table 3.9, is a direct consequence of the growth of t_{cons}^{mean}, since the rate of attack detection is computed once consensus is reached. If *partial* consensus time instant is considered, i.e. the time instant k from which only the second of the two consensus conditions defined in Section 3.2.2, the one relative to attacks, is verified, then the performances are similar for the two considered cases, as illustrated in Figure 3.21. This consideration is consistent with the behavior of the attack support error, which, as previously said, assumes almost the same values in the two cases.

	MRAD		
	$Q = Q^{[5]}$	$Q = Q^{[9]}$	$Q = Q^{[13]}$
Time-invariant attacks	744.16	763.92	769.68
Time-varying attacks	549.05	591.45	608.99

Table 3.9: Mean rate of attack detection for different topologies, time-invariant vs time-varying attacks (Distributed Sparse Observer).

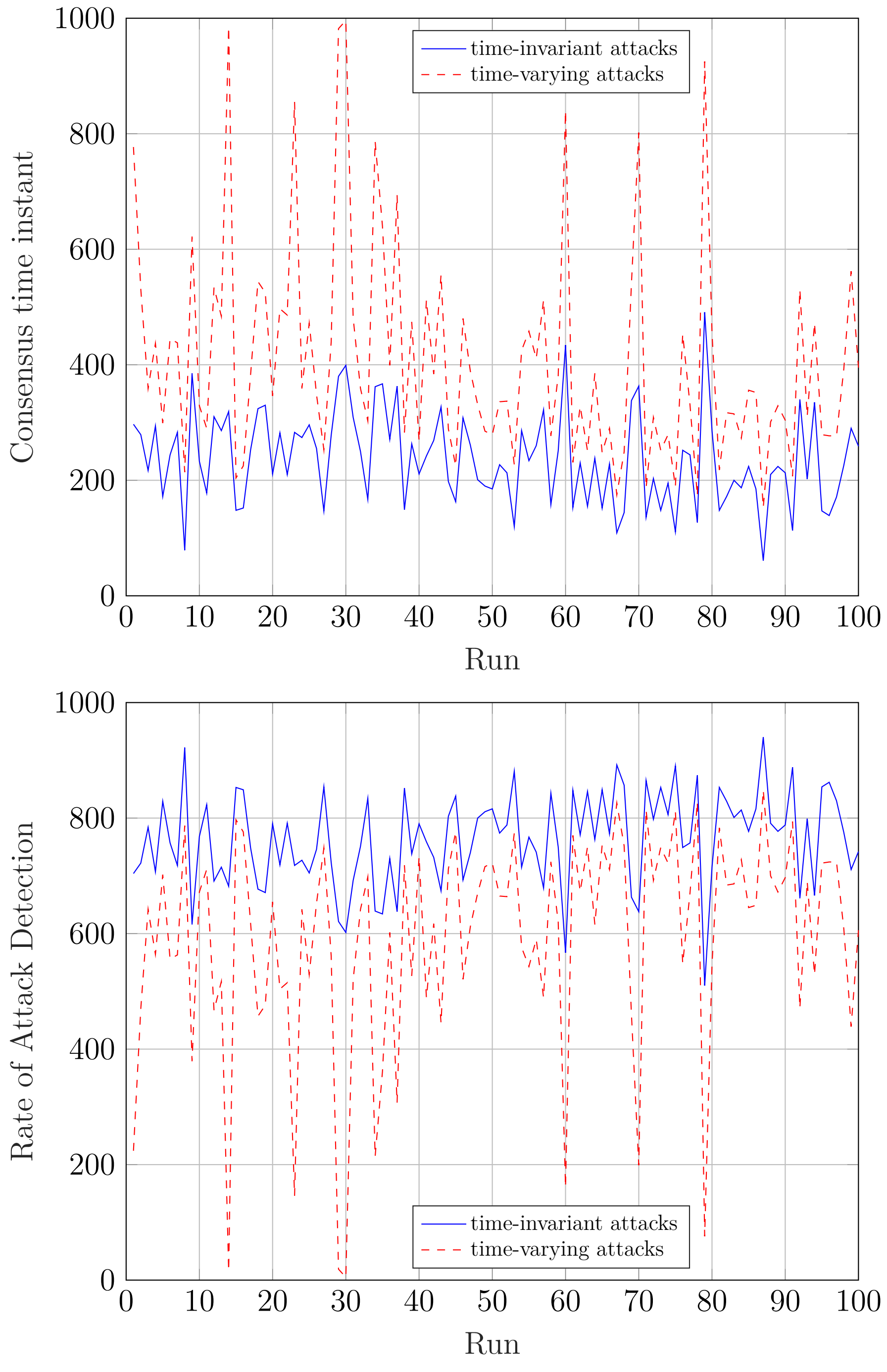


Figure 3.20: Consensus time instant t_{cons} and rate of attack detection for each run with $Q = Q^{[9]}$, time-invariant vs time-varying attacks.

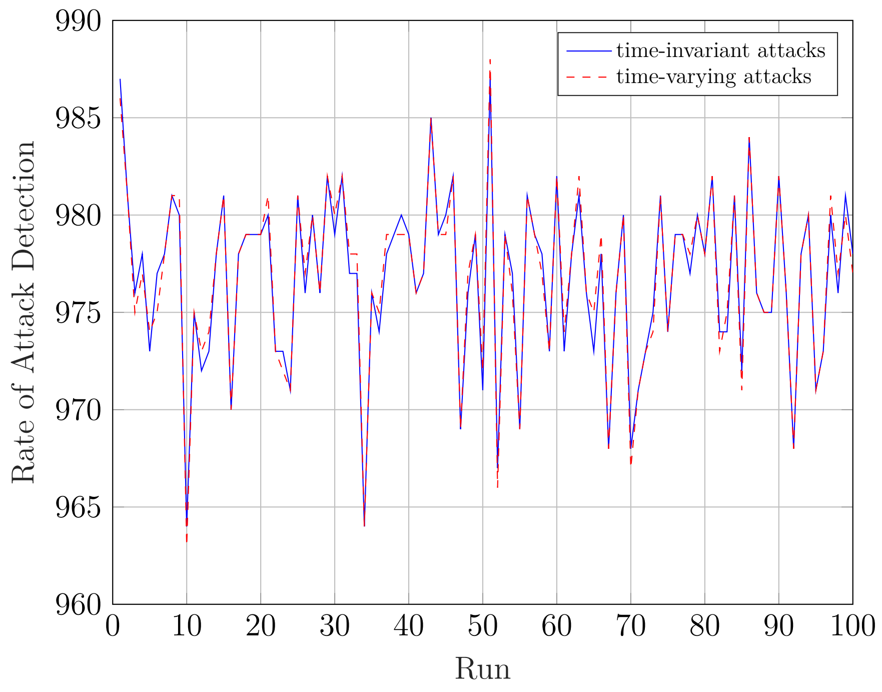


Figure 3.21: Rate of attack detection computed from partial consensus time instant for each run with $Q = Q^{[9]}$, time-invariant vs time-varying attacks.

3.5 Hyperparameters variations

In the previous sections, numerical simulations have been executed with system state dimension $n = 10$, $q = 20$ sensors, and $s = 2$ sensor attacks. In this section, new numerical simulations are performed using different values of n, q, and s, to test the algorithms in different configurations. Three experiments are conducted, and for each n, q, s setting, the algorithms are executed for 100 runs, each $T = 1000$ time steps long. The performance is evaluated considering the mean state estimation error and the mean attack support error for each setting.
In the first experiment, $n = 10$ and $q = 20$ are fixed, while the number of sensors under attack s varies in the range $[1, 6]$. In Figures 3.22, 3.23 are illustrated the corresponding results. As regards the Distributed Sparse Observer, Table 3.10 contains the values of the additional metrics defined to further investigate consensus performance. Figures 3.24, 3.25 show the Distributed Sparse Observer results for $s = 1$ and $s = 6$, respectively. The regularization parameters vector Λ and the attack estimate threshold $\hat{a}_{thres}$ maintain the same values illustrated in Section 3.2.

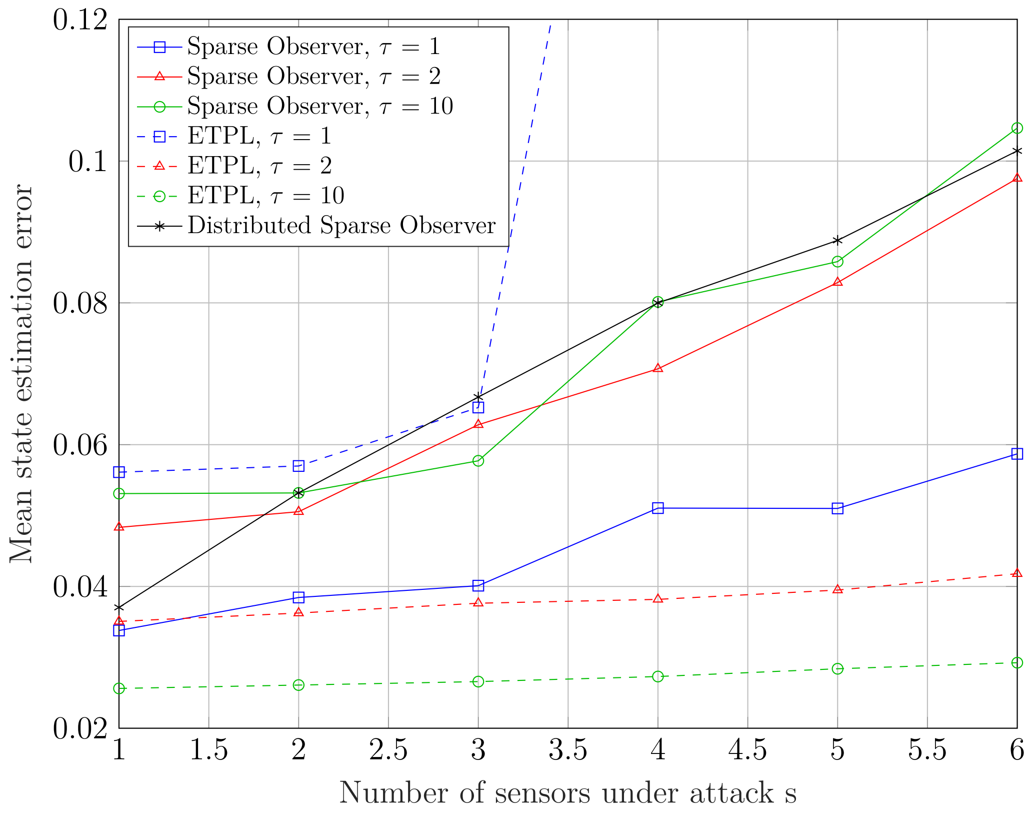


Figure 3.22: Mean state estimation error for different values of s.

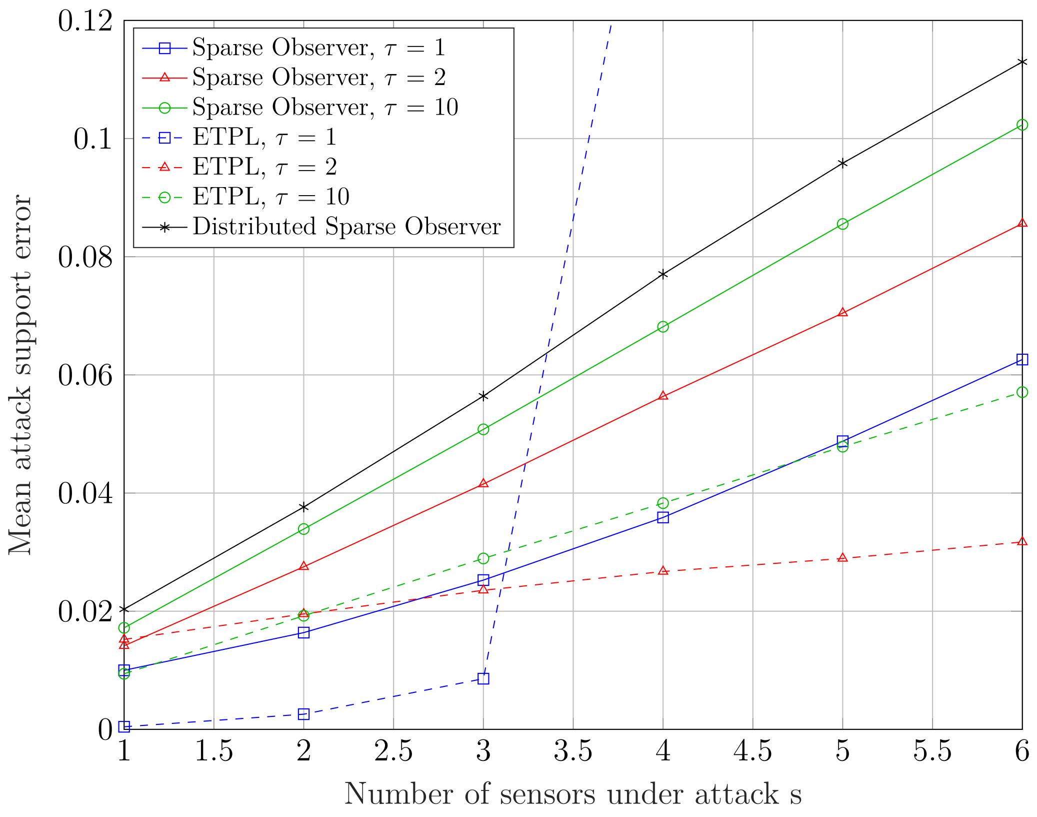


Figure 3.23: Mean attack support error for different values of s.

	ϕ_{cons}	$t_{cons}^{\max}$ (s)	t_{cons}^{mean} (s)	MRAD
$s = 1$	1	422	138.82	978.84
$s = 2$	1	491	237.08	976.76
$s = 3$	1	490	291.69	974.45
$s = 4$	1	479	329.81	972.96
$s = 5$	1	516	336.87	971.94
$s = 6$	1	495	367.62	971.55

Table 3.10: Distributed Sparse Observer performance comparison between different values of s.

As expected, both the mean state estimation error and the mean attack support error increase for greater values of s since more measurements are corrupted by adversarial attacks. For the Distributed Sparse Observer, it results that $\phi_{cons} = 1$

for each value of s examined, which means that consensus is achieved within T in all the 100 runs. The mean consensus time instant t_{cons}^{mean} gradually increases for greater s values, whereas the mean rate of attack detection, computed from the *partial* consensus time instant, is nearly the same in all the cases.

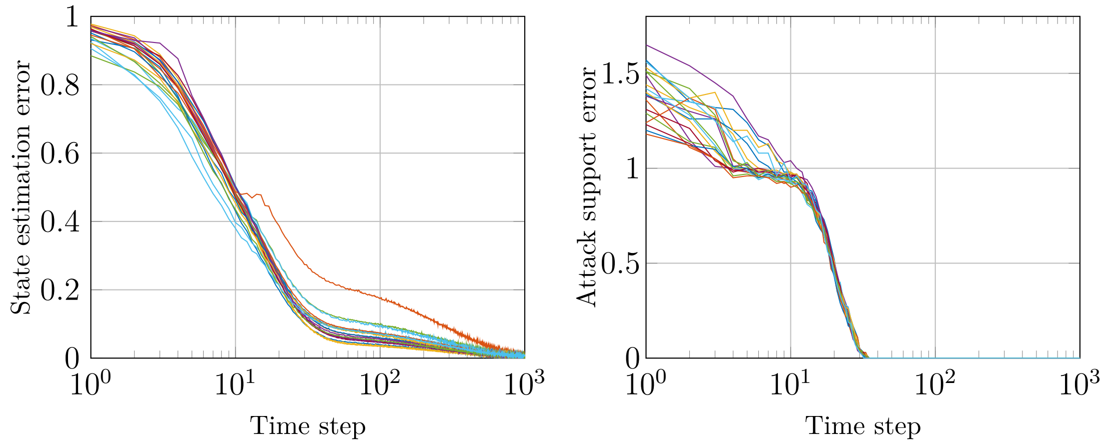


Figure 3.24: Distributed Sparse Observer with $n = 10$, $q = 20$, $s = 1$, $Q = Q^{[9]}$. Each line corresponds to a specific sensor.

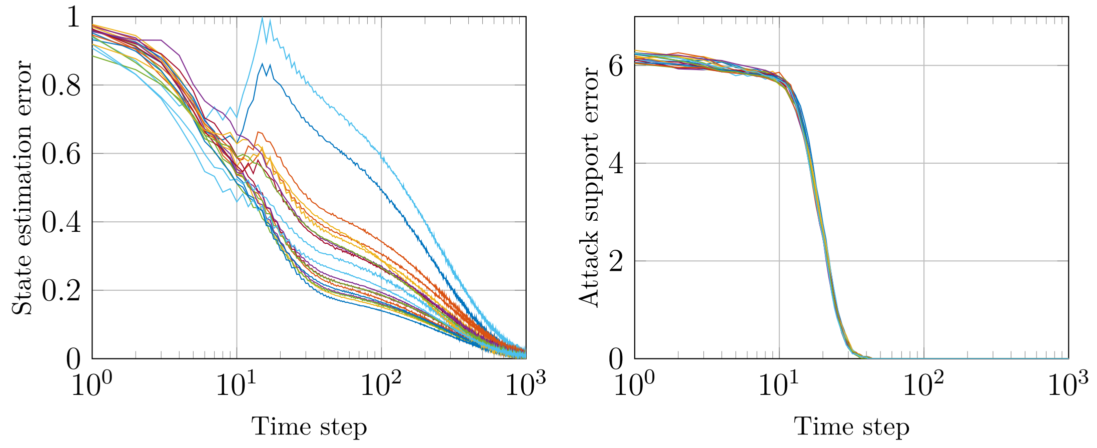


Figure 3.25: Distributed Sparse Observer with $n = 10$, $q = 20$, $s = 6$, $Q = Q^{[9]}$. Each line corresponds to a specific sensor.

In the second experiment, $n = 10$ and $s = 2$ are fixed, while the total number of sensors q varies in the range [15, 45], in steps of 5. In Figures 3.26, 3.27 are illustrated the corresponding results. As regards the Distributed Sparse Observer, Table 3.11 contains the values of the additional metrics defined to further investigate consensus performance. Figures 3.28, 3.29 show the Distributed Sparse Observer results for $q = 15$ and $q = 45$, respectively. For $q = 20$, the considered connection topology is the standard one $Q = Q^{[9]}$, defined in Section 3.2.2. For the other cases, equivalent connection matrices are employed, i.e. doubly-stochastic matrices such that each node is connected to a number of agents equal to 40% of the total number of nodes. The regularization parameters vector Λ maintain the same values illustrated in Section 3.2, while the attack estimate threshold $\hat{a}_{thres}$ is gradually reduced for increasing values of q.

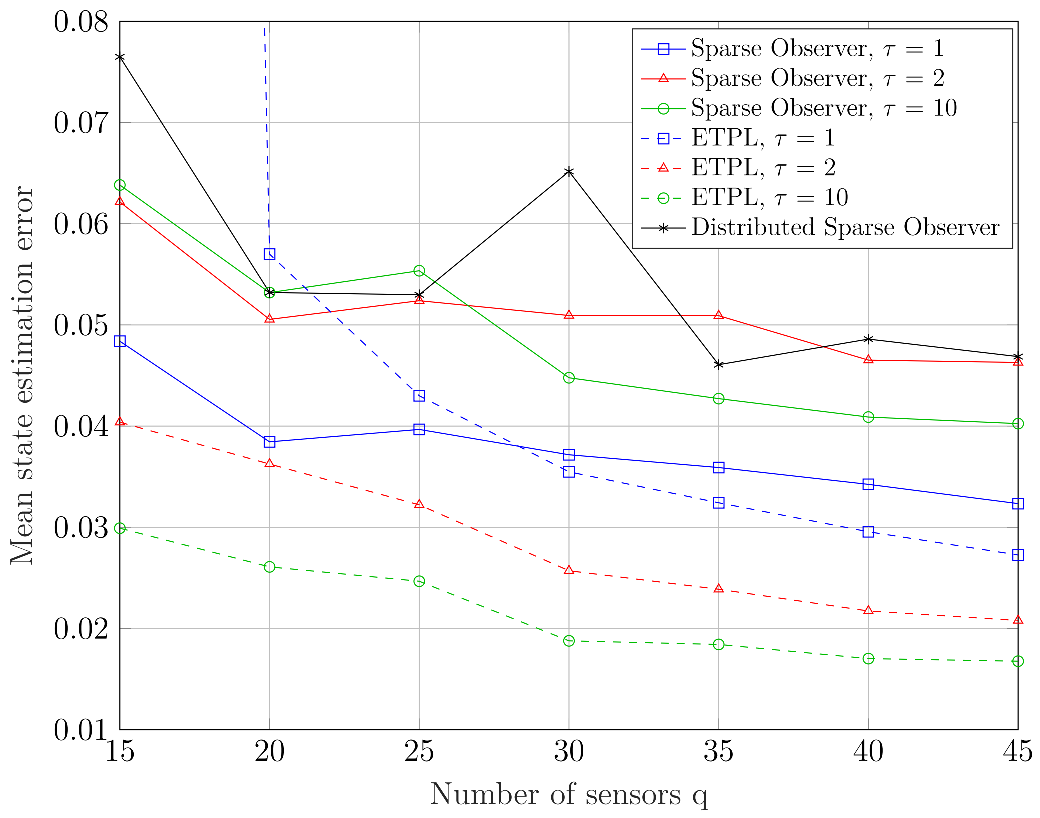


Figure 3.26: Mean state estimation error for different values of q.

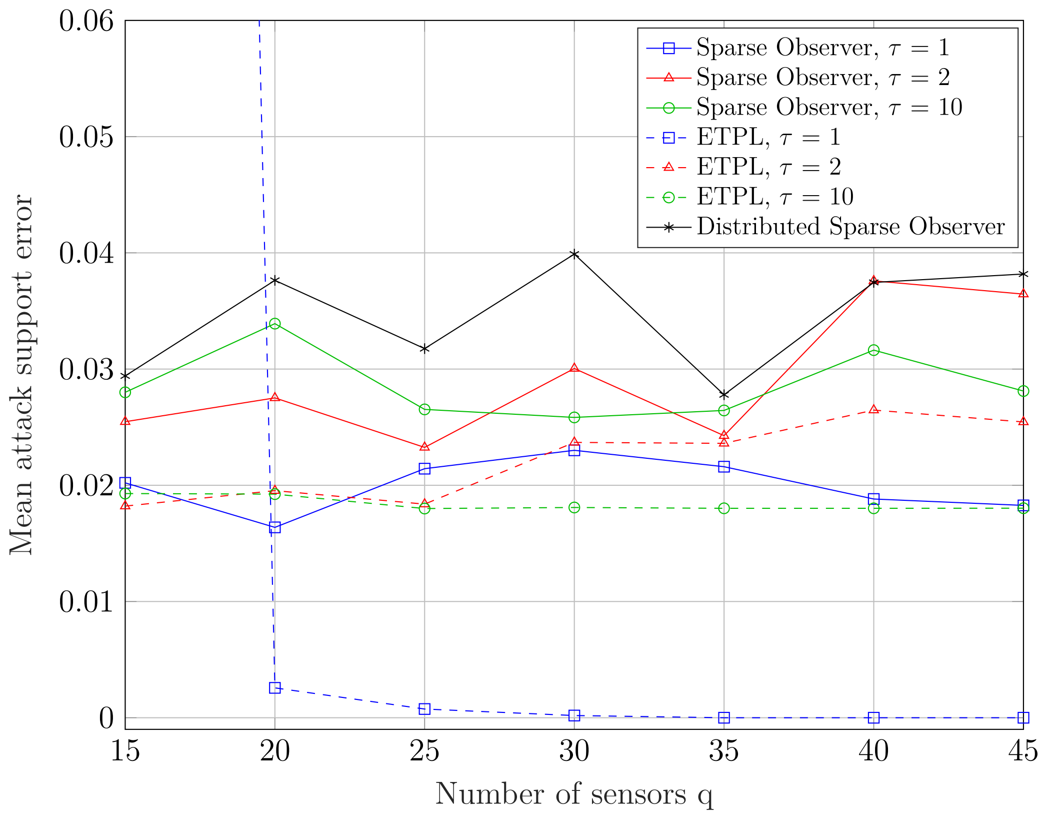


Figure 3.27: Mean attack support error for different values of q.

	ϕ_{cons}	$t_{cons}^{\max}$ (s)	t_{cons}^{mean} (s)	MRAD
$q = 15$	1	357	224.92	978.83
$q = 20$	1	491	237.08	976.76
$q = 25$	1	474	305.22	980.92
$q = 30$	1	776	407.84	970.95
$q = 35$	1	675	434.98	983.08
$q = 40$	0.99	996	526.78	978.64
$q = 45$	0.98	885	532.59	978.03

Table 3.11: Distributed Sparse Observer performance comparison between different values of q.

Overall, the mean state estimation error decreases for greater values of q since more measurements are available to the FC for the centralized algorithms. On the other hand, in the distributed case, each node has access to its own scalar measurements besides the estimates exchanged with neighboring nodes, which is the only term whose dimension depends on q. The mean attack support error does not exhibit a well-defined trend as q varies, for the various algorithms. As regards the Distributed Sparse Observer, from Table 3.11, it follows that the consensus performance in terms of ϕ_{cons}, $t_{cons}^{\max}$ and t_{cons}^{mean} tends to deteriorate for increasing values of q. This derives from the fact that consensus among estimates must be reached by a greater number of sensors. The mean rate of attack detection, computed from the *partial* consensus time instant, is nearly the same in all the cases.

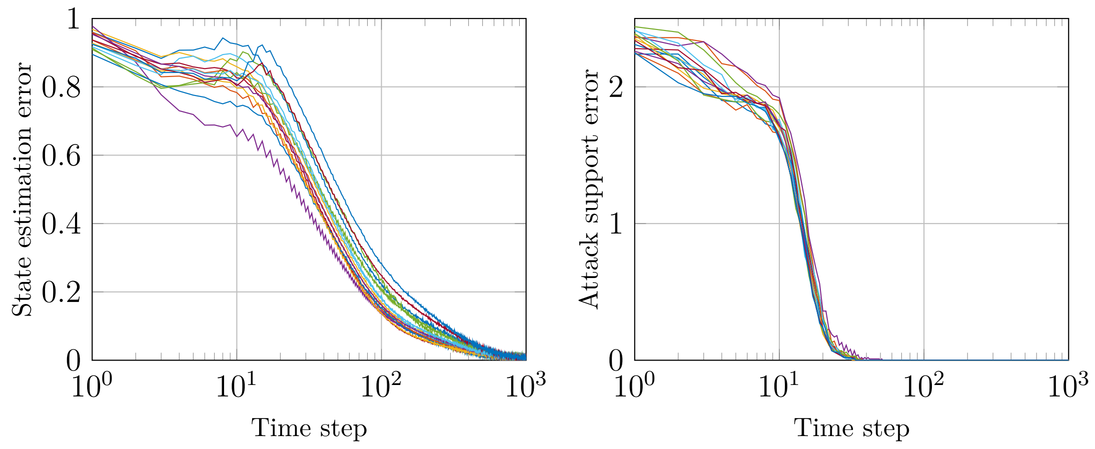


Figure 3.28: Distributed Sparse Observer with $n = 10$, $q = 15$, $s = 2$. Each line corresponds to a specific sensor.

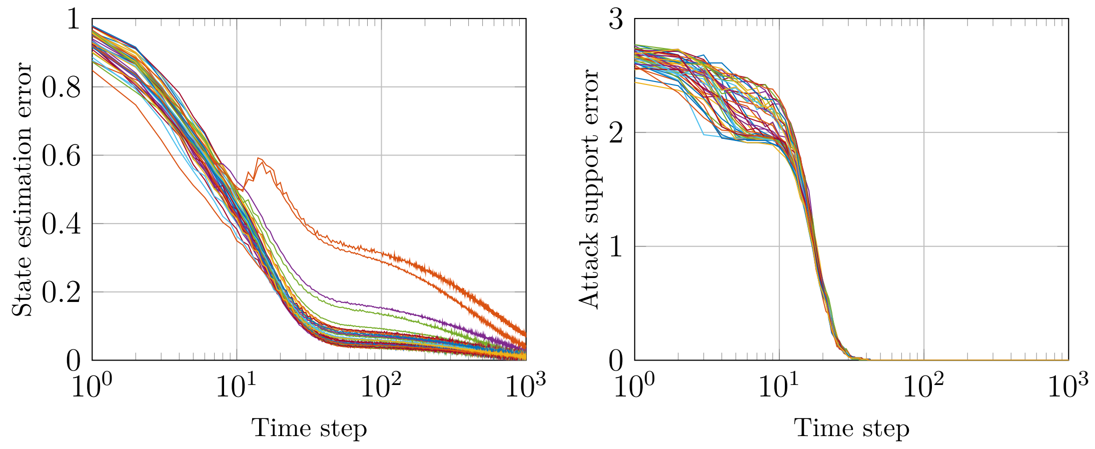


Figure 3.29: Distributed Sparse Observer with $n = 10$, $q = 45$, $s = 2$. Each line corresponds to a specific sensor.

In the last experiment, the default values of the hyperparameters are doubled, such that $n = 20$, $q = 40$, and $s = 4$. Table 3.13 shows the results in terms of mean state estimation error and mean attack support error for each algorithm in the standard and doubled configurations. The regularization parameter and the attack estimate threshold values are further tuned, and the chosen values are shown in Table 3.12. As regards the ETPL algorithm, $\hat{a}_{thres}$ is set to 0.4 for any τ.

	Sparse Observer			Distributed Sparse Observer
	$\tau = 1$	$\tau = 2$	$\tau = 10$	
$\gamma \cdot \lambda$	$7 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$9 \cdot 10^{-5}$	$1 \cdot 10^{-7}$
$\hat{a}_{thres}$	0.5	0.4	0.1	0.25

Table 3.12: λ and $\hat{a}_{thres}$ values for $n = 20$, $q = 40$, $s = 4$.

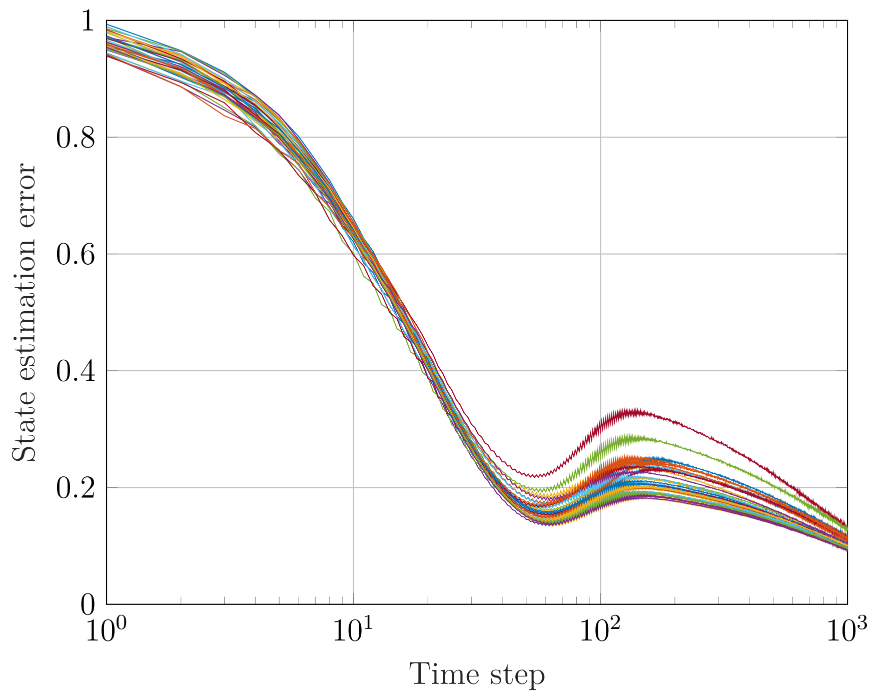
		Mean state estimation error		Mean attack support error	
		$n = 10$, $q = 20$, $s = 2$	$n = 20$, $q = 40$, $s = 4$	$n = 10$, $q = 20$, $s = 2$	$n = 20$, $q = 40$, $s = 4$
Sparse Observer	$\tau = 1$	0.0384	0.1033	0.0164	0.2133
	$\tau = 2$	0.0505	0.1293	0.0275	0.3877
	$\tau = 10$	0.0532	0.2374	0.0339	0.5957
ETPL	$\tau = 1$	0.0570	0.0309	0.0026	0.0022
	$\tau = 2$	0.0363	0.0213	0.0195	0.0493
	$\tau = 10$	0.0261	0.0154	0.0192	0.0364
Distributed Sparse Observer		0.0532	0.1671	0.0376	0.6247

Table 3.13: Mean state estimation error and mean attack support error comparison between standard and doubled settings.

From Table 3.13, it results that the mean state estimation error and the mean attack support error values increase considerably for both the centralized and distributed versions of the Sparse Observer. On the other hand, the mean state estimation error decreases for ETPL, which exploits the assumption of knowing a tight upper bound on the number of sensors under attack. Table 3.14 contains further results about the Distributed Sparse Observer. The consensus performance in the new setting is much worse compared to the standard configuration, as not only does consensus among estimates need to be reached by twice the number of sensors, but also the number of components of the state to be estimated double. In Figure 3.30 are illustrated the Distributed Sparse Observer results for $n = 20$, $q = 40$, $s = 4$.

	ϕ_{cons}	t_{cons}^{max} (s)	t_{cons}^{mean} (s)	MRAD
$n = 10$, $q = 20$, $s = 2$	1	491	237.08	976.76
$n = 20$, $q = 40$, $s = 4$	0.79	999	849.30	790.25

Table 3.14: Distributed Sparse Observer performance comparison between standard and doubled settings.



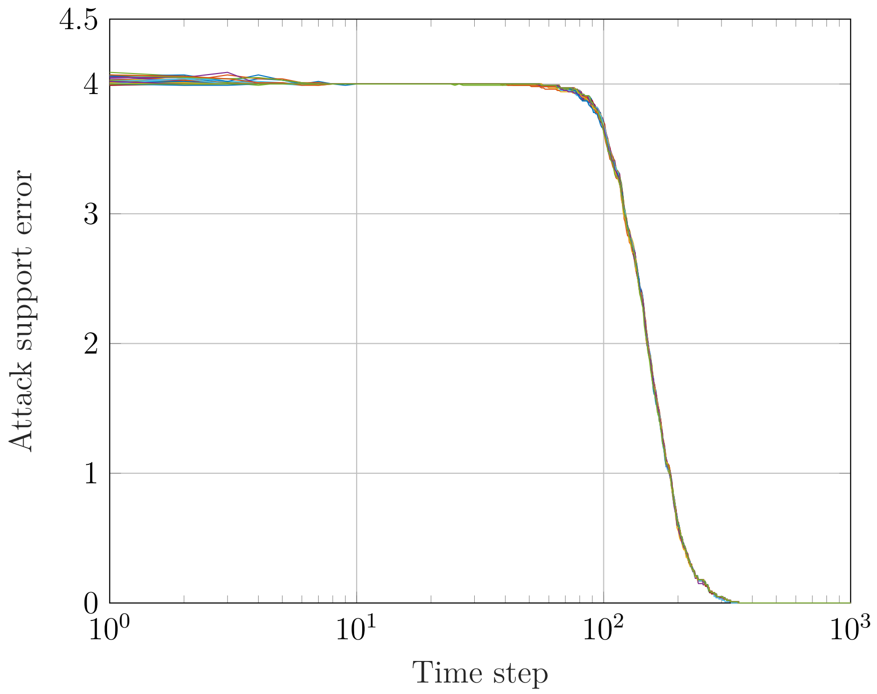


Figure 3.30: Distributed Sparse Observer with $n = 20$, $q = 40$, $s = 4$. Each line corresponds to a specific sensor.

Chapter 4

Real-world application: localization problem

4.1 Introduction to localization

Localization consists in estimating the position of a target. In recent years, much attention has been devoted to the development of indoor localization services, taking advantage of the evolution of Wireless Sensor Networks (WSNs) [25]. A WSN is a network of microprocessors equipped with sensors that can be exploited for localization tasks in indoor environments or in outdoor adverse conditions where global positioning system (GPS) is not suitable due to high hardware costs or environmental conditions. The potential applications of WSNs are numerous: object tracking, traffic monitoring, measuring radiation levels, detecting seismic activities, location detection of products in warehouses or of medical personnel and equipment in hospitals [26], [27].
Indoor localization can be accomplished by measuring the *Received Signal Strength* (RSS), i.e. the power transmitted by the target. Two different approaches are possible: *distance-prediction* based or *fingerprinting* based. Distance-prediction based systems estimate the position of the target by calculating its distances from at least three reference points using a known radio propagation model [28]. The main disadvantage of this method is that it suffers from multi-path and refraction effects due to the presence of obstacles, possibly moving [25]. Fingerprinting methods are employed to overcome these inaccuracy issues. They consist in creating a signature map in order to represent the physical space by capturing the variations of the dynamic nature of indoor radio propagation [28].
Here we focus on RSS-fingerprinting methods for indoor localization. These techniques rely on the strength attenuation of radio signals during propagation, which consequently can be used to determine the distance of the target broadcasting

the signal [29]. The RSS at distance d is commonly modeled as [30]:

$$P_r(d) = P_t - \overline{PL}(d_0) - 10n\log_{10}\left(\frac{d}{d_0}\right) - \eta_\sigma \tag{4.1}$$

where $P_r(d)$ is the receiving power at distance d in dBm (decibel-milliwatts), P_t is the transmitting power and $\overline{PL}(d_0)$ is the average of the path loss value measurements at a reference distance d_0 (usually, $d_0 = 1\,\mathrm{m}$). The constant n, known as attenuation coefficient, depends on the transmission medium, e.g. indoor/outdoor environment, and ranges typically from 2 to 4. η_σ is a zero-mean Gaussian distributed random noise with standard deviation σ.
The WSN, composed of q sensors, is randomly deployed in the room where localization of the target has to be performed. A rectangular room is considered, whose area is divided into p cells. In this setting, localization of the target means detecting the cell occupied by the device. The number of cells p represents a trade-off between localization accuracy and complexity. The higher is p, the more accurate is the localization, since the cells have a smaller size, at the cost of greater storage and time requirements and increased problem complexity.
RSS-fingerprinting methods generally consist of two phases: the *training* (or off-line) phase and the *runtime* (or on-line) phase. Given a target and a WSN, during the training phase a signature map or *dictionary* is created: the target is placed in turn in each cell, e.g. in the center of the cell, and broadcasts a signal. Each of the q sensors measures and stores the corresponding RSS values. Therefore, each sensor i builds its own dictionary, represented by the vector $D_i = (D_{i,1}, \dots, D_{i,p})^T \in \mathbb{R}^p$. Globally, the WSN builds a dictionary $D \in \mathbb{R}^{q,p}$, such that the component $D_{i,j}$ is the RSS-measurement acquired by the i^{th} sensor when the device is in cell j.

$$D = \begin{pmatrix} D_{1,1} & D_{1,2} & \dots & D_{1,p} \\ D_{2,1} & D_{2,2} & \dots & D_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ D_{q,1} & D_{q,2} & \dots & D_{q,p} \end{pmatrix} \in \mathbb{R}^{q,p} \tag{4.2}$$

Due to the presence of noise, both in the signal transmission and caused by stable or moving obstacles in the localization area, each sensor i can take multiple RSS-measurements for the same cell, since redundancy helps to enhance the probability of accurate localization [25]. Here, for the sake of simplicity, a single acquisition is considered for each cell.
During the runtime phase, the actual localization is performed by exploiting the dictionary built during the training step. The moving target broadcasts a signal, and each sensor measures the RSS value, such that the i^{th} sensor acquires only one measurement $y_i \in \mathbb{R}$. The online measurements are compared with the fingerprints obtained during the offline phase to perform localization.

The training phase, which is typical of fingerprinting methods, has the advantage of increasing the accuracy of the localization performed during the online phase. On the other hand, dictionary construction is time consuming and, moreover, each sensor has to store and exchange a large amount of data. To address these issues, the localization task can be reformulated as a sparse approximation problem [30] and, since RSS is additive, more than one target can be localized. Given $j \geq 1$ targets moving in a room, the positions of the targets are represented by a vector of length p such that only j entries are non-zero, the ones corresponding to the occupied cells. Since the devices to track are usually few compared to the number of cells, the position vector is j-sparse. Compressive sensing theory is helpful to solve this type of problem, since it involves an underdetermined linear system under sparsity constraints [25], [28], [30].

4.1.1 Distributed localization

Once the training phase is completed and dictionary D is given, two possible settings can be employed for the runtime phase:

- Centralized setting: the sensors convey their data to a centralized fusion center, which runs the localization algorithm. The FC stores the global dictionary D and the runtime measurements $(y_1, ..., y_q)^T \in \mathbb{R}^q$ acquired by the sensors.
- Distributed setting: each sensor stores its runtime measurement $y_i \in \mathbb{R}$ and its own dictionary $D_i \in \mathbb{R}^p$ (i^{th} row of D) and does not share them. Moreover, each agent i has its own local estimate $\hat{x}^{(i)} \in \mathbb{R}^p$ and shares it with the neighboring nodes it is connected to. The connection topology is represented by a stochastic matrix Q.

The advantages of the distributed approach over the centralized one have been discussed in Section 1.3. In the next sections, we focus on solving a localization problem using the distributed version of the Sparse Observer, implemented in Section 2.2.2.

4.2 Problem definition

A localization problem where j targets move in a square room, discretized in $p = 100$ square cells, is examined. It is assumed that the square room has a 10 m side, and therefore each cell has a side of 1 m and an area of 1 m^2. The WSN is composed of $q = 25$ sensors randomly deployed in the room. The dictionary $D \in \mathbb{R}^{q,p}$ is considered given, i.e. the training phase has already been completed, and we focus on the runtime phase. The CPS can be modeled as a discrete-time linear time-invariant dynamical system:

$$\begin{aligned} x(k+1) &= Ax(k) \\ y(k) &= Dx(k) + \eta(k) + a(k) \end{aligned} \tag{4.3}$$

where $x(k) \in \{0,\, 1\}^p$ represents the current positions of the targets, such that if $x_i(k) = 1$, one of the targets is placed in the cell i at time $k \in \mathbb{N}$, $y(k) \in \mathbb{R}^q$ are the acquired RSS-measurements, $\eta(k) \in \mathbb{R}^q$ is the random measurement noise vector, such that $\eta(k) \sim \mathcal{N}(0, \sigma^2)$, with $\sigma = 10^{-2}$, and $a(k) \in \mathbb{R}^q$ is the attack vector, defined such that $\|a(k)\|_0 \leq s$, with $s = 2$, and with random uniformly distributed support set constant over time. Moreover, the attack vector components are defined as $a_i(k) = a_i = 30$, $\forall k \in \mathbb{N}$, therefore unaware time-invariant attacks are considered. In Section 4.5, time-varying attacks with a constant random support set are examined. The initial state $x_0 \in \{0,\, 1\}^p$ is generated with uniformly distributed random support.
From this setting, it follows that the localization problem can be cast into a Lasso problem, exploiting the sparsity of the state vector $x(k)$, since only $j \ll p$ components of the state are nonzero. In the absence of malicious sensor attacks, at each time instant k the Lasso problem can be formulated as:

$$\min_{x \in \mathbb{R}^p} \frac{1}{2}\|y(k) - Dx\|_2^2 + \Lambda^T|x| = \min_{x \in \mathbb{R}^p} \sum_{i=1}^{q} \left[\frac{1}{2}\|y_i(k) - D_i x\|_2^2 + \Lambda^T|x|\right] \tag{4.4}$$

where $\Lambda \in \mathbb{R}^p$ is the regularization parameters vector. Analogously, in the presence of s-sparse sensor attacks:

$$\min_{x \in \mathbb{R}^p,\, a \in \mathbb{R}^q} \quad \sum_{i=1}^{q} \left[\frac{1}{2}\left\| y_i(k) - \begin{pmatrix} D & I \end{pmatrix}_i \begin{pmatrix} x \\ a \end{pmatrix} \right\|_2^2 + \Lambda^T \left| \begin{pmatrix} x \\ a \end{pmatrix} \right| \right] \tag{4.5}$$

where $(D\ I)_i$ is the i^{th} row of matrix $(D\ I)$, $I \in \mathbb{R}^{q,q}$ is the identity matrix and $\Lambda \in \mathbb{R}^{p+q}$.
Our goal is to employ the distributed version of the observer, named Distributed Sparse Observer (Algorithm 5), to solve this localization problem and evaluate its performance, in the presence of adversarial attacks on $s = 2$ sensors.

4.3 Numerical simulations

In this section, the results of the numerical simulations are reported. The Distributed Sparse Observer algorithm is executed for 100 runs, each $T = 1000$ time steps long. At each time step $k \in \{1, \ldots, T\}$, an estimate $\hat{x}^{(i)}(k)$ of the real state vector $\tilde{x}(k)$, along with an estimate $\hat{a}^{(i)}(k)$ of the real attack vector $\tilde{a}(k)$, are computed by each sensor $i \in \{1, \ldots, q\}$. Since in this case the state is not dense and the relevant information is the state vector support, which determines in which cells the targets are present, the time evolution of the following metrics is evaluated, averaged over the multiple runs:

- the state support error $\sum_{j=1}^{p} |\mathbf{1}(\tilde{x}_j(k) \neq 0) - \mathbf{1}(\hat{x}_j(k) \neq 0)|$, where $\mathbf{1}(v) = 1$ if v is true and 0 otherwise;
- the mean Euclidean distance between the real and estimated positions of the targets in the square room over the j targets;
- the attack support error $\sum_{j=1}^{q} |\mathbf{1}(\tilde{a}_j(k) \neq 0) - \mathbf{1}(\hat{a}_j(k) \neq 0)|$.

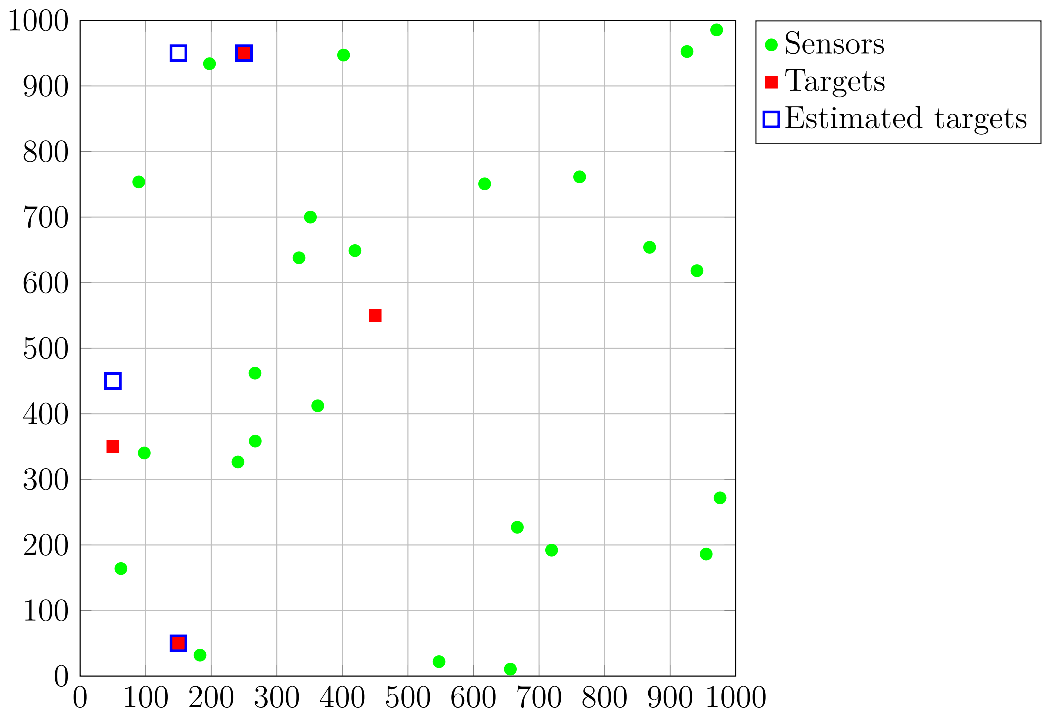
The state support error metric is useful since it indicates if the cell currently occupied by the target is correctly detected. On the other hand, if the cell is not properly estimated, it provides no information about the distance between the estimated target position and the correct one. For this reason, the Euclidean distance metric is introduced. In particular, it is supposed that each target occupies the center of its respective cell, which corresponds to a pair of coordinates in space. Defining the origin of the reference system allows the computation of the Euclidean distance of each estimated target position from the closest real target that is not already correctly estimated in the specific time instant k. Figure 4.1 provides an example of how the Euclidean distance is computed with $j = 4$ targets. The regularization parameters vector $\Lambda \in \mathbb{R}^{p+q}$ is defined as:

$$\Lambda^T \left| \begin{pmatrix} x \\ a \end{pmatrix} \right| = (\underbrace{\lambda_x \ \ldots \ \lambda_x}_{\in \mathbb{R}^p} \ \underbrace{\lambda_a \ \ldots \ \lambda_a}_{\in \mathbb{R}^q}) \left| \begin{pmatrix} x_1 \\ \vdots \\ x_p \\ a_1 \\ \vdots \\ a_q \end{pmatrix} \right| =$$

$$= \lambda_x \cdot |x_1| + \ldots + \lambda_x \cdot |x_p| + \lambda_a \cdot |a_1| + \ldots + \lambda_a \cdot |a_q|$$

Before executing the simulations, a tuning of the components of vector Λ is performed. Initially, sensor attacks are not taken into account, i.e. the attack-free

localization problem is solved. Different λ_x values are tested and for each of them, 250 runs are executed, each $T = 1000$ time steps long. The optimal value is chosen as the one that provides the smallest mean consensus time instant. Then, $s = 2$ sensor attacks are introduced: different λ_a values are tested, keeping λ_x fixed at the value obtained from the previous tuning, and for each of them, 250 runs are executed, each $T = 1000$ time steps long. The optimal value is chosen as the one that provides the highest mean rate of attack detection or, equivalently, the smallest mean consensus time instant for the attack estimates.
Moreover, as done is Section 3.2, a threshold $\hat{a}_{thres}$ below which the components of the estimated attack vector are considered null is defined. Lastly, since the state estimate is expected to be j-sparse, only the j largest components of $\hat{x}^{(i)}(k)$, $\forall i \in \{1, \ldots, q\}$ and $\forall k \in \mathbb{N}$, are considered, setting to zero all the others.



Euclidean distances $= \begin{pmatrix} 0, & 100, & 500, & 0 \end{pmatrix}$, ordering the cells starting from (0,0) and proceeding row by row.

Figure 4.1: Example of Euclidean distance computation with $j = 4$ targets and $p = 100$ cells.

4.3.1 Localization with sensor attacks

The performance of the Distributed Sparse Observer is assessed for solving the localization problem with $s = 2$ sensor attacks, in the presence of either $j = 2$ or $j = 4$ moving targets. The connection graph standard topology considered here is represented by a stochastic matrix $Q \in \mathbb{R}^{25,25}$, such that node i has 11 incoming edges, including the self-loop, i.e. each row Q_i of matrix Q has 11 nonzero elements and $\sum_{j=1}^{q} Q_{i,j} = 1$. This standard topology is indicated as $Q^{[11]}$ to distinguish it from the other topologies examined in Section 4.4. Differently from Section 3.3, here we do not consider doubly-stochastic topology matrices in order to address a more general case.
A looser definition of consensus is employed for the estimates provided by the $q = 25$ different nodes. At time instant k, each sensor $i \in \{1, \dots, q\}$ computes

$$\hat{z}^{(i)}(k) = \begin{pmatrix} \hat{x}^{(i)}(k) \\ \hat{a}^{(i)}(k) \end{pmatrix} = \text{estimate of} \begin{pmatrix} x(k) \\ a(k) \end{pmatrix} \tag{4.6}$$

The consensus time instant t_{cons} is defined as the time instant k from which the following two conditions are satisfied:

1. the state vector estimates $\hat{x}^{(i)}(k)$, $\forall i \in \{1, \dots, q\}$ have the same support set;
2. the attack vector estimates $\hat{a}^{(i)}(k)$, $\forall i \in \{1, \dots, q\}$ have the same support set.

The learning rate γ is defined as $\gamma = \min_i \|(D\ I)_i\|_2^2 - 1 \cdot 10^{-8}$, whereas hyperparameter tuning leads to the values in Table 4.1. Figures 4.2, 4.3 show how the regularization parameters λ_x and λ_a are chosen for $j = 2$ (the same procedure is used for $j = 4$).

	Distributed Sparse Observer	
	$j = 2$	$j = 4$
$\gamma \cdot \lambda_x$	$1 \cdot 10^{-6}$	$5 \cdot 10^{-7}$
$\gamma \cdot \lambda_a$	$8 \cdot 10^{-4}$	$8 \cdot 10^{-4}$
$\hat{a}_{thres}$	0.1	0.1

Table 4.1: Distributed Sparse Observer λ_x, λ_a and $\hat{a}_{thres}$ values for each number of targets.

The results of the numerical simulations are shown in Figures 4.4, 4.5, where each line corresponds to a specific sensor. To properly compare the performance of the algorithm, the same attack $a(k)$ and noise $\eta(k)$ vectors are used in the two cases. The initial condition x_0, originally generated for $j = 2$ targets, is extended for the $j = 4$ case to have a total of 4 nonzero components.

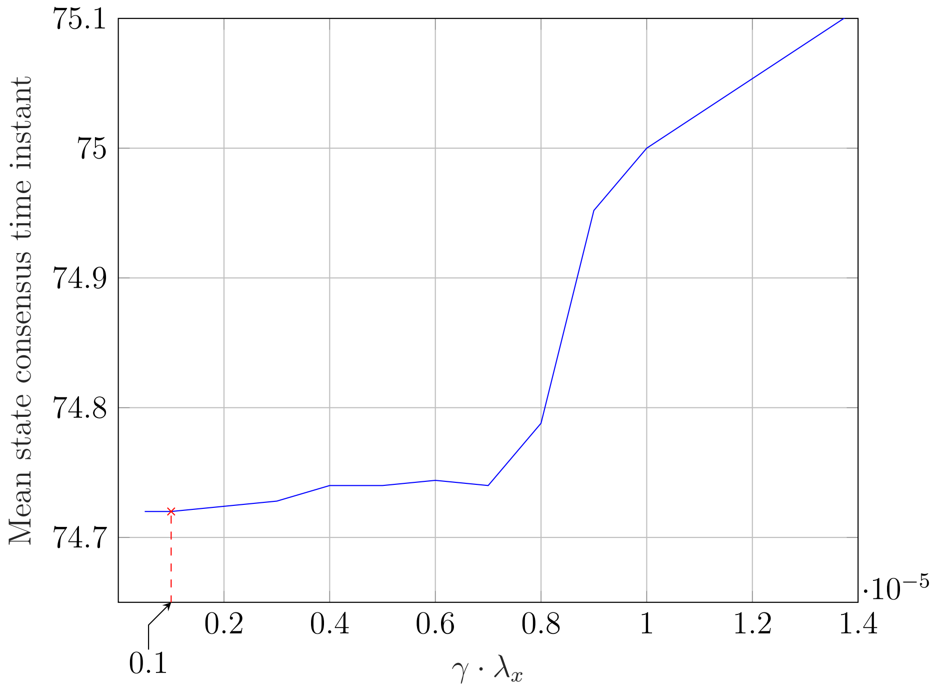


Figure 4.2: Distributed Sparse observer λ_x tuning for $j = 2$.

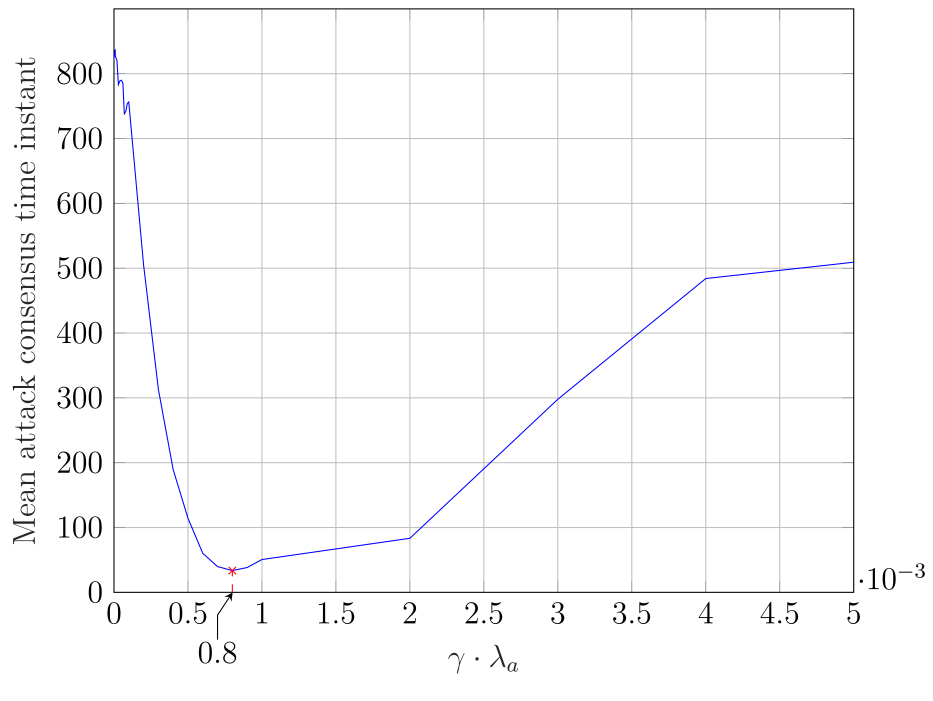


Figure 4.3: Distributed Sparse observer λ_a tuning for $j = 2$.

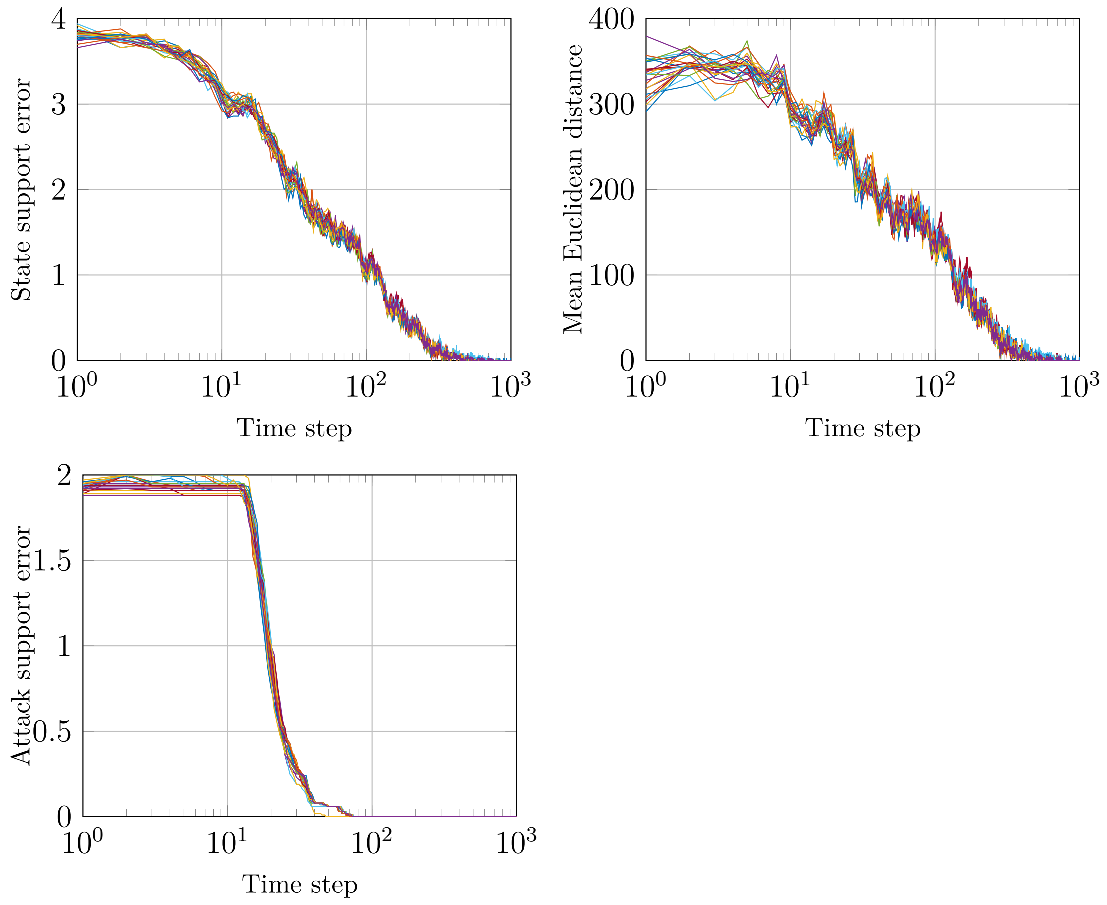


Figure 4.4: Distributed Sparse Observer with $p = 100$, $q = 25$, $s = 2$, $j = 2$, standard topology $Q = Q^{[11]}$. Each line corresponds to a specific sensor.

Table 4.2 contains the values of the state support error, the mean Euclidean distance and the attack support error averaged over the T time instants and the q sensors. As expected, the value of the mean state support error is lower for the $j = 2$ targets

	Mean state support error	Mean Euclidean distance	Mean attack support error
$j = 2$	0.3016	35.4444	0.0425
$j = 4$	0.7145	37.7622	0.0425

Table 4.2: State support error, mean Euclidean distance and attack support error averaged over time and over $q = 25$ sensors, for each number of targets.

case with respect to the $j = 4$ case. In particular, by doubling the number of

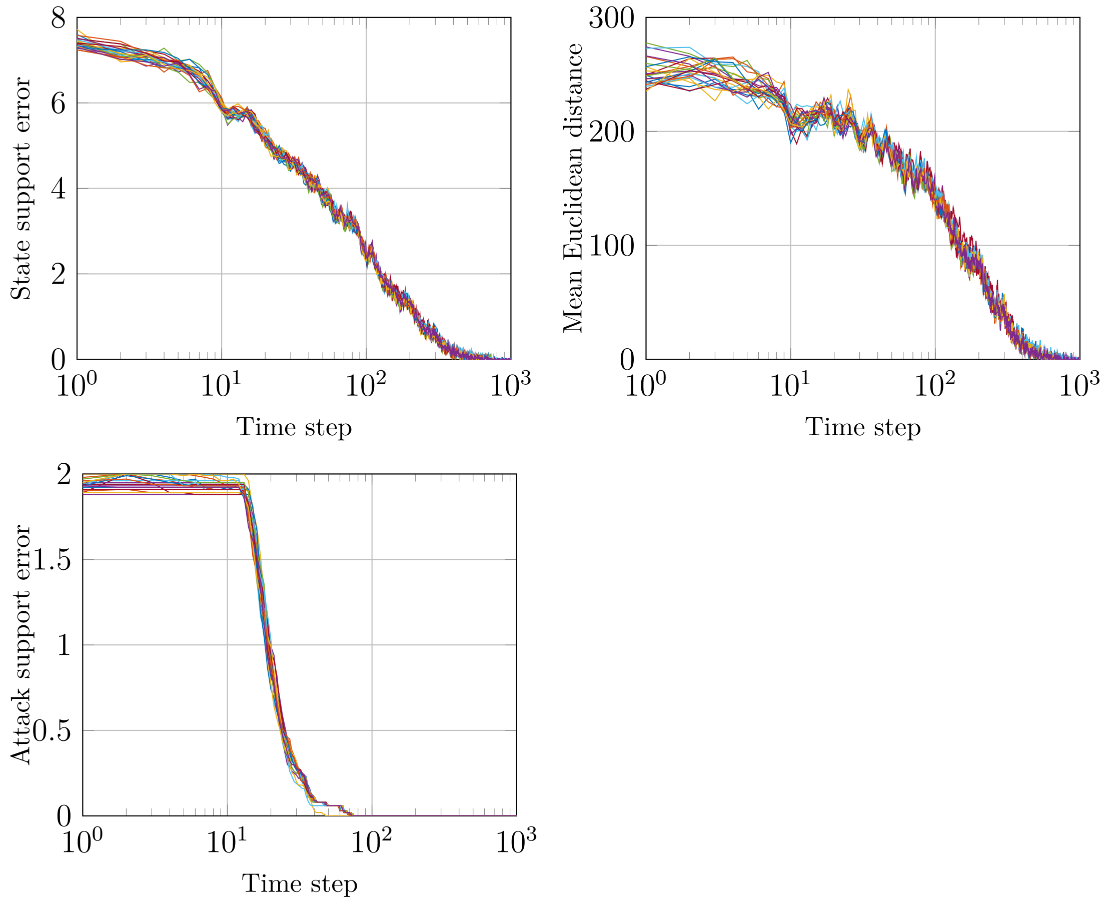


Figure 4.5: Distributed Sparse Observer with $p = 100$, $q = 25$, $s = 2$, $j = 4$, standard topology $Q = Q^{[11]}$. Each line corresponds to a specific sensor.

targets, the mean state support error increases by 137%. On the other hand, the mean Euclidean distance increases only by 6.54%. From these results, it follows that the error in detecting the correct cell occupied by the targets increases with the number of targets, while the distance in space between the estimated positions and the real ones remains relatively constant. As regards the attack support error, the mean value is the same in both cases, leading to the conclusion that the number of targets to track does not influence this metric. Table 4.3 shows the values of the additional metrics defined in Section 3.3, plus the Mean Rate of Position Detection (MRPD), which denotes the mean value over the runs of the rate of position detection, i.e. the number of times that the support of the state vector is correctly estimated after consensus among nodes estimates is reached. From Figure 4.6 it is clear that the rate of position detection is complementary to the consensus time instant with respect to T. Therefore, once consensus among the nodes' estimates is achieved, the positions of the targets are correctly detected at each time instant $k \in \{t_{cons}, \dots, T\}$ for each of the 100 executions. The same consideration applies to the rate of attack detection, which means that the corresponding plot coincides with the rate of position detection one and, for this reason, it is not reported here. As a consequence, the MRPD and MRAD values, computed after consensus is reached, coincide. To provide more informative data, in Table 4.3 the MRAD values are computed from the *partial* consensus time instant, i.e. the time instant k from which only the second consensus condition, the one relative to attacks, is verified.

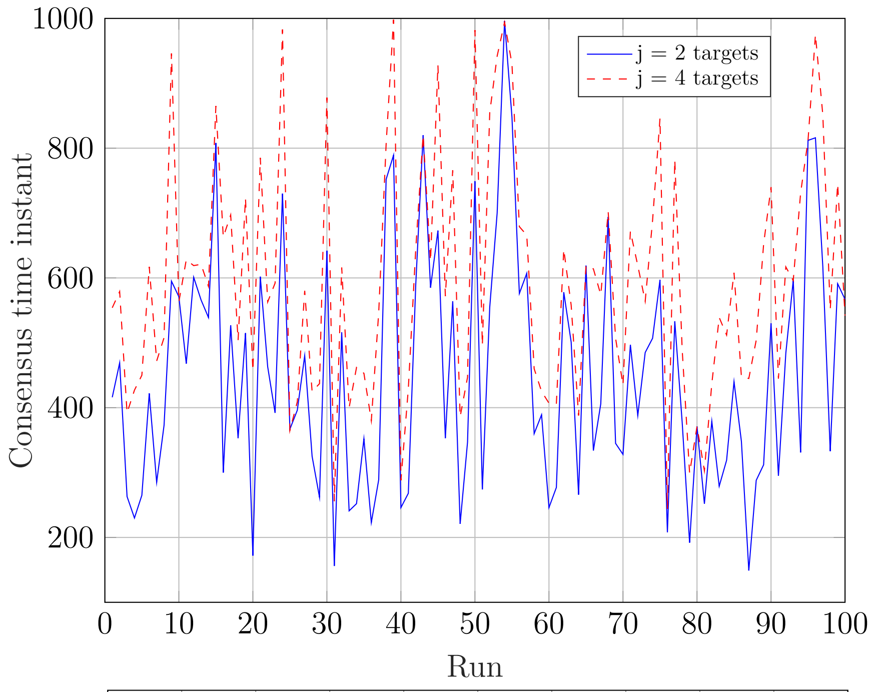
	ϕ_{cons}	$t_{cons}^{\max}$ (s)	t_{cons}^{mean} (s)	MRPD	MRAD
$j = 2$	1	996	451.74	549.26	972.07
$j = 4$	1	998	596.24	404.76	972.00

Table 4.3: Distributed Sparse Observer performance comparison between $j = 2$ and $j = 4$ targets.

Firstly, $\phi_{cons} = 1$ for each number of targets j examined, which means that consensus is achieved within T in all the 100 runs, and the maximum consensus time instant $t_{cons}^{\max}$ is nearly the same in both cases. Then:

- t_{cons}^{mean} increases by 32% from $j = 2$ to $j = 4$ targets;
- MRPD decreases by 26% from $j = 2$ to $j = 4$ targets.

Lastly, MRAD computed from the partial consensus time instant is approximately the same for the two cases. It is possible to conclude that the number of targets to localize influences the performance of the Distributed Sparse Observer algorithm in terms of consensus time instant and, consequently, position detection. In particular, the sensors require more time to reach a consensus among the state estimates for a greater number of targets. On the other hand, the mean rate of attack detection does not depend on the number of targets, consistently with the attack support error performance. This conclusion is graphically supported by Figure 4.7, which shows the MRAD computed from the partial consensus time instant as a function of the execution.



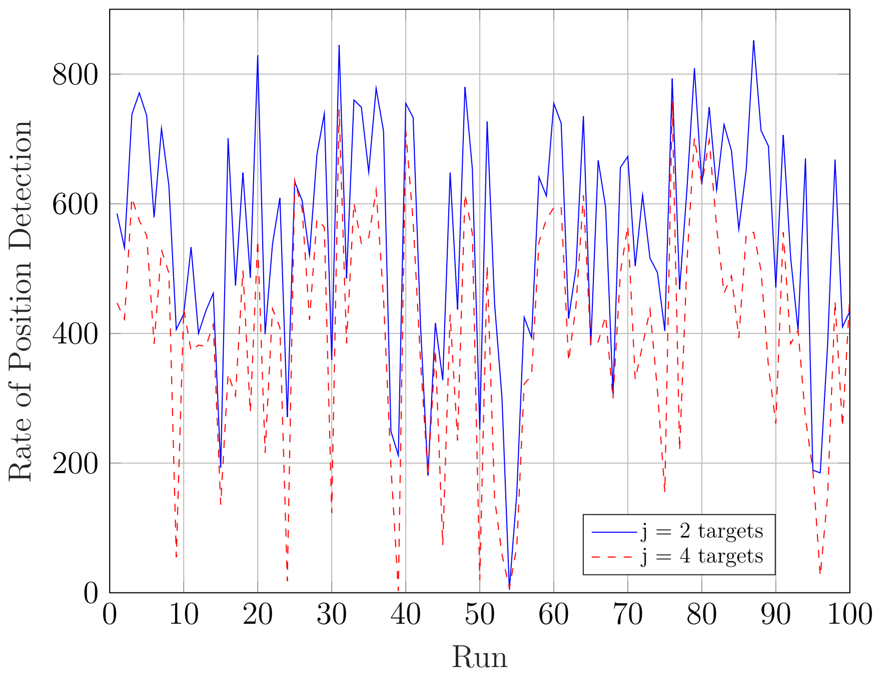


Figure 4.6: Consensus time instant t_{cons} and rate of position detection for each run with $Q = Q^{[11]}$, $j = 2$ vs $j = 4$ targets.

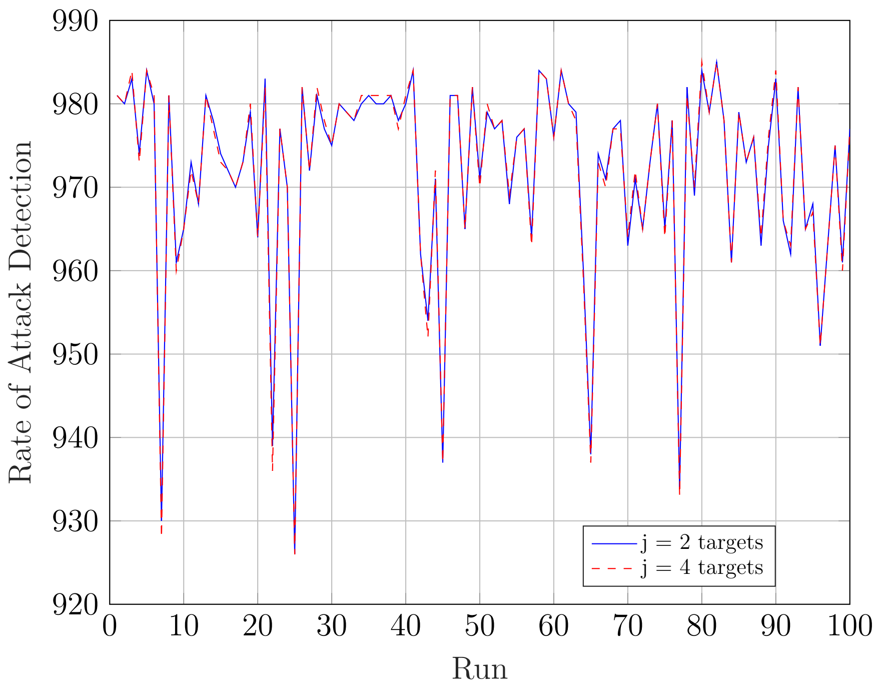


Figure 4.7: Rate of attack detection computed from partial consensus time instant for each run with $Q = Q^{[11]}$, $j = 2$ vs $j = 4$ targets.

4.4 Topology analysis

In this section, different topologies from the standard one are considered, with the aim of determining how the topology affects the performance of the Distributed Sparse Observer algorithm for solving the localization problem by varying the number of connections among sensors. The case with $j = 4$ targets is considered here (the same trends are observed for the $j = 2$ case as well). The evaluated topologies are the following, ordered based on the increasing number of connections per node:

- $Q = Q^{[6]}$, stochastic matrix such that each node i has 6 incoming edges, including the self-loop. Each sensor receives information from 5 different nodes, which corresponds to 20% of the total number of sensors.
- $Q = Q^{[11]}$, stochastic matrix such that each node i has 11 incoming edges, including the self-loop (standard topology defined and employed in Section 4.3.1). Each sensor receives information from 10 different nodes, which corresponds to 40% of the total number of sensors.
- $Q = Q^{[16]}$, stochastic matrix such that each node i has 16 incoming edges, including the self-loop. Each sensor receives information from 15 different nodes, which corresponds to 60% of the total number of sensors.

Table 4.4 contains the values of the state support error, the mean Euclidean distance and the attack support error averaged over the T time instants and the q sensors, for each topology Q.

	Mean state support error	Mean Euclidean distance	Mean attack support error
$Q = Q^{[6]}$	0.7054	36.9628	0.0541
$Q = Q^{[11]}$	0.7145	37.7622	0.0425
$Q = Q^{[16]}$	0.7017	36.9876	0.0413

Table 4.4: State support error, mean Euclidean distance and attack support error averaged over time and over $q = 25$ sensors for each topology, with $j = 4$ targets.

Figures 4.8 and 4.9 illustrate the results for $Q = Q^{[6]}$ and $Q = Q^{[16]}$, respectively. Table 4.5 contains the additional metrics defined to investigate the consensus performance. As in Table 4.3, MRAD is computed from the partial consensus time instant onward.

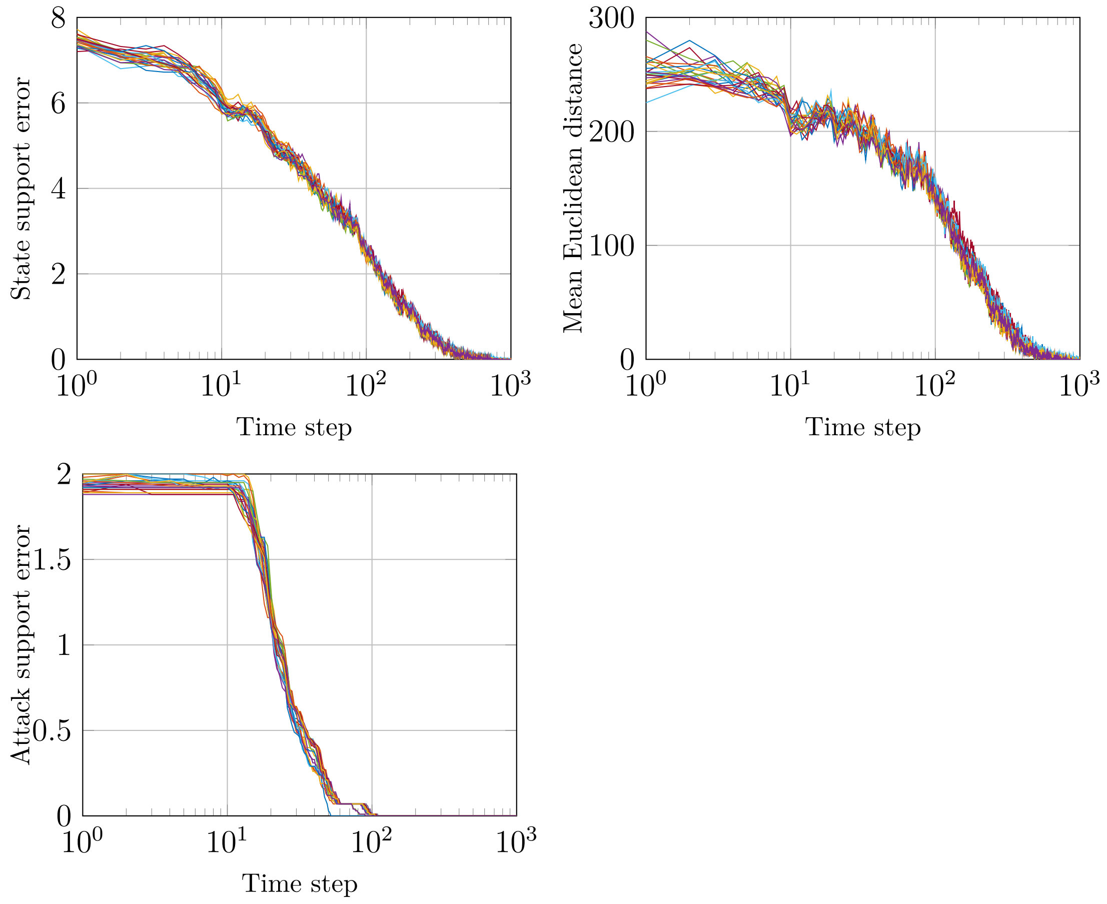


Figure 4.8: Distributed Sparse Observer with $p = 100$, $q = 25$, $s = 2$, $j = 4$, $Q = Q^{[6]}$. Each line corresponds to a specific sensor.

From Table 4.4, it is possible to notice that the mean attack support error decreases by 21.4% from $Q^{[6]}$ to $Q^{[11]}$ and only by 2.8% from $Q^{[11]}$ to $Q^{[16]}$. On the other hand, the mean state support error and the mean Euclidean distance seem to exhibit an unexpected trend. In particular, the mean state support error increases by 1.29% from $Q^{[6]}$ to $Q^{[11]}$ and, as regards the Euclidean distance, topology $Q^{[6]}$ provides a lower value than that obtained with $Q^{[16]}$. Actually, these results provide only a partial view, and it is necessary to further analyze the performance of the different topologies with the metrics in Table 4.5. With the topology matrix $Q^{[6]}$, it results that $\phi_{cons} = 0.98$, which means that consensus is not achieved within $T = 1000$ time instants in 2 out of the 100 executions. Specifically, consensus is not reached since the first of the two consensus conditions defined in Section 4.3.1, the one involving the state estimates, is not satisfied. For the other two topologies, $\phi_{cons} = 1$.

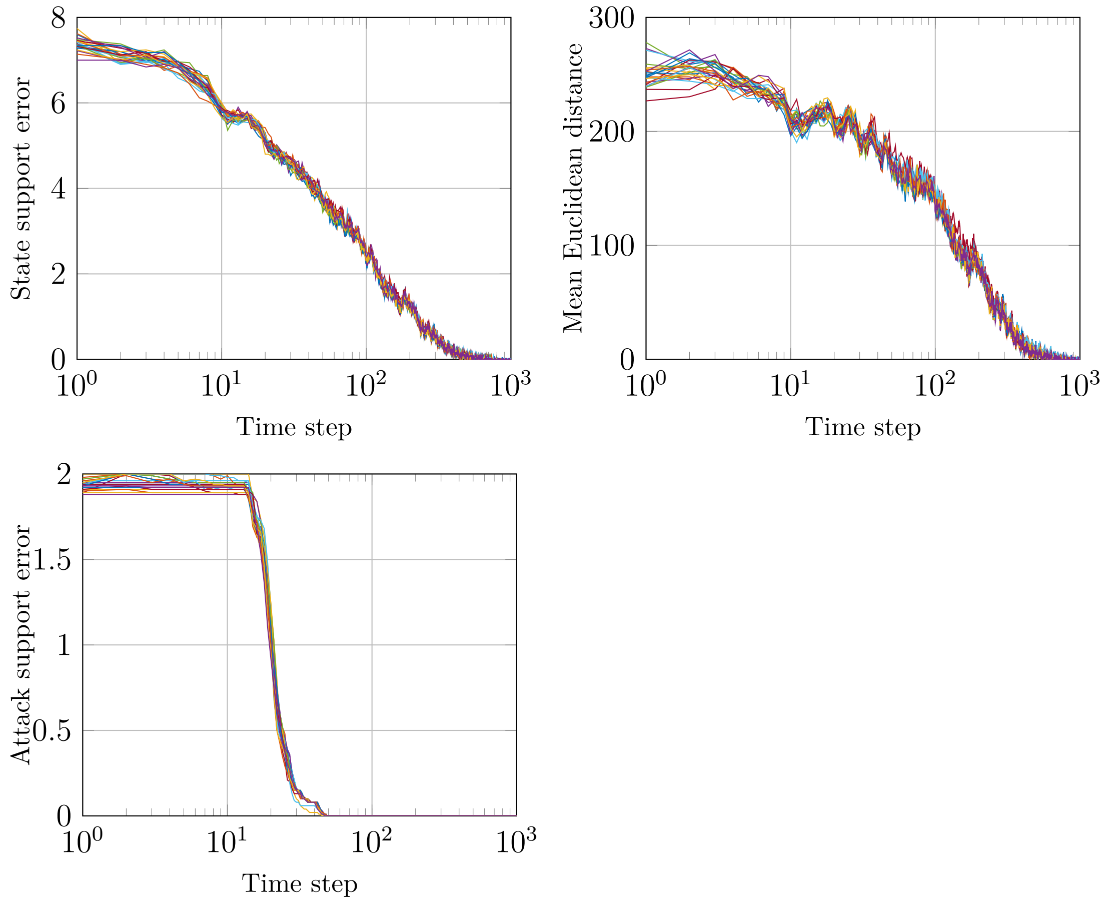


Figure 4.9: Distributed Sparse Observer with $p = 100$, $q = 25$, $s = 2$, $j = 4$, $Q = Q^{[16]}$. Each line corresponds to a specific sensor.

	ϕ_{cons}	$t_{cons}^{\max}$	t_{cons}^{mean}	MRPD	MRAD
		(s)	(s)		
$Q = Q^{[6]}$	0.98	997	625.13	375.86	961.73
$Q = Q^{[11]}$	1	998	596.24	404.76	972.00
$Q = Q^{[16]}$	1	998	581.79	419.21	974.22

Table 4.5: Distributed Sparse Observer performance comparison between different topologies, with $j = 4$ targets.

Moreover, the maximum consensus time instant $t_{cons}^{\max}$ is practically the same for the three different topology matrices. As regards the other metrics:

- t_{cons}^{mean} decreases by 4.62% from $Q^{[6]}$ to $Q^{[11]}$ and by 2.42% from $Q^{[11]}$ to $Q^{[16]}$;
- MRPD increases by 7.69% from $Q^{[6]}$ to $Q^{[11]}$ and by 3.57% from $Q^{[11]}$ to $Q^{[16]}$;
- MRAD increases by 1.07% from $Q^{[6]}$ to $Q^{[11]}$ and by 0.23% from $Q^{[11]}$ to $Q^{[16]}$;

Figure 4.10 represents the rate of attack detection computed from the partial consensus time instant. Figure 4.11 shows the consensus time instant and the rate of position detection with respect to the specific run. The "x" marks indicate that consensus has not been reached in that particular run. Also for topologies $Q^{[6]}$ and $Q^{[16]}$, the rate of position detection is complementary to the consensus time instant with respect to T.
Overall, topology $Q^{[6]}$ exhibits poor performance, whereas the addition of 5 connections from $Q^{[11]}$ to $Q^{[16]}$ does not determine a significant improvement in the values of the considered metrics. Therefore, we can conclude that the number of connections given by matrix $Q^{[11]}$, i.e. 10 incoming edges for each sensor plus the self-loop, is sufficient to have satisfactory results, both in terms of estimation and of consensus among agents.

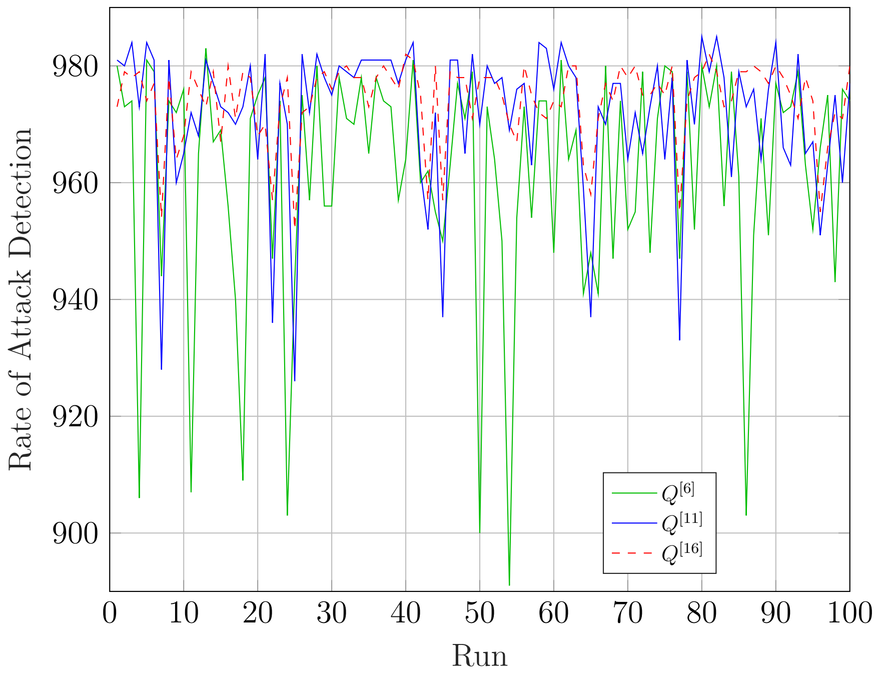
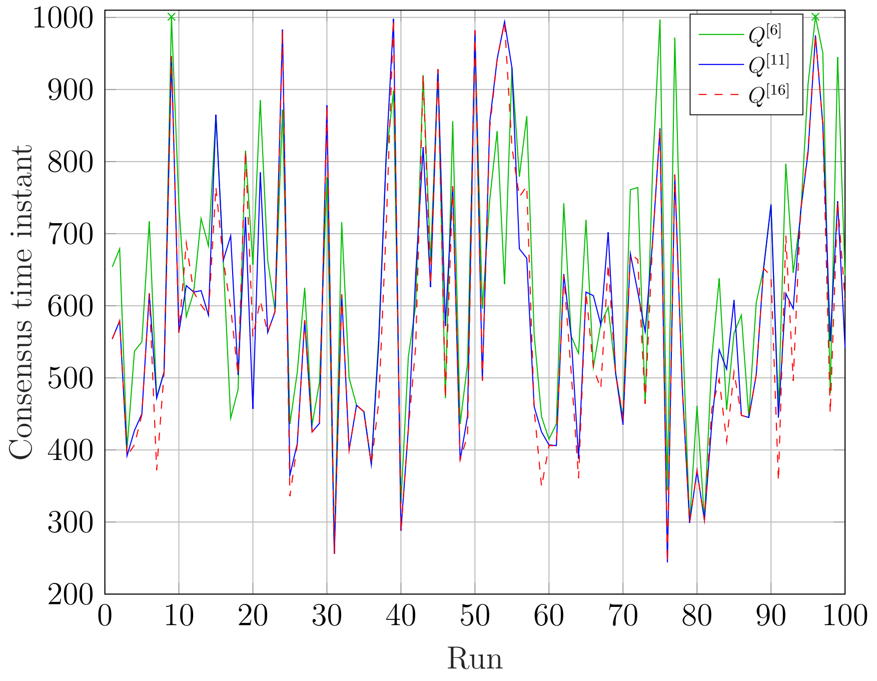


Figure 4.10: Rate of attack detection computed from partial consensus time instant for each run, topologies comparison.



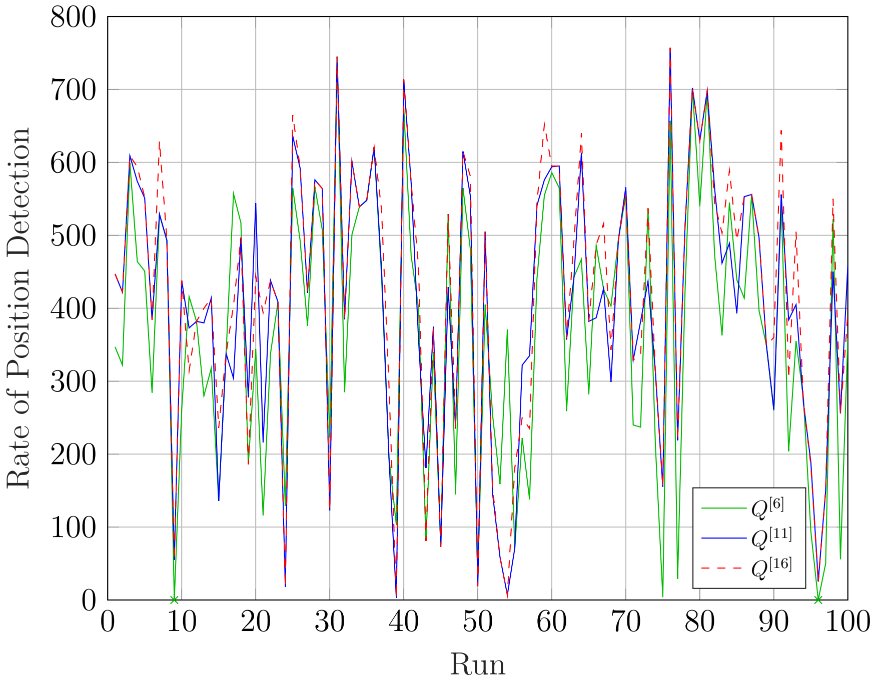


Figure 4.11: Consensus time instant t_{cons} and rate of position detection for each run, topologies comparison.

4.5 Time-varying attacks

Until now, the malicious attacks in the CPS model 4.3 have been considered as time-invariant with constant support. In this section, time-varying adversarial attacks $a(k)$ with constant support are employed, in order to examine the Distributed Sparse Observer algorithm performance differences with the previous case. In practice, the time-varying attack vector $a(k)$ is generated by taking the support of the previously created attack vector and defining each of the $s = 2$ nonzero components $a_i(k)$ as a uniformly distributed random number in the range [25, 35], for each time instant $k \in \{1, \ldots, T\}$. A possible time evolution of one of the $s = 2$ nonzero components of the time-varying sensor attacks is represented in Figure 4.12 (for clarity, only the first 200 time steps are displayed). The other nonzero component follows an equivalent trend.

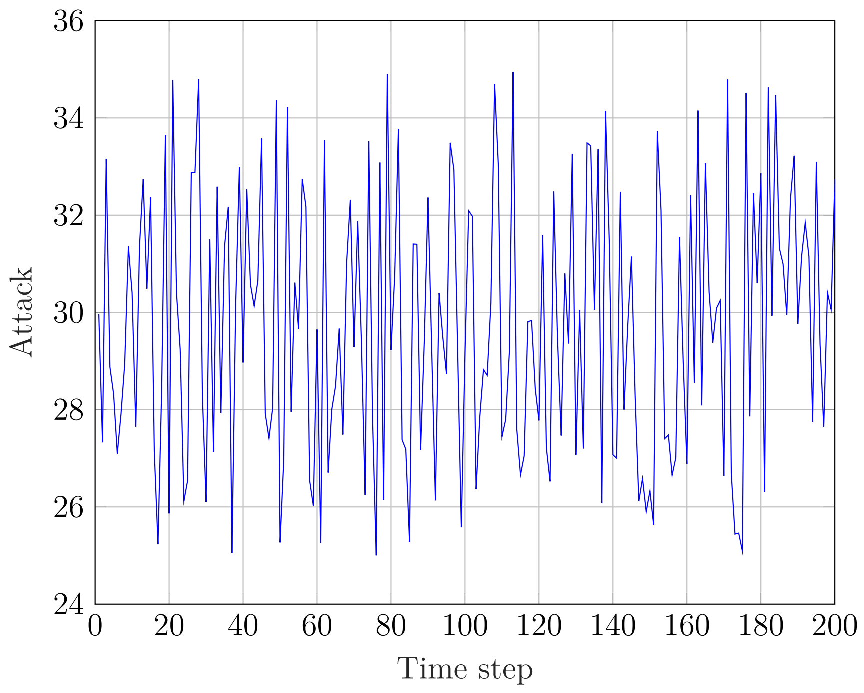


Figure 4.12: Example of the time evolution of one nonzero component of the time-varying sensor attacks (first 200 time steps).

4.5.1 Localization with sensor attacks

The Distributed Sparse Observer algorithm is tested for solving the localization problem with $j = 4$ targets with time-varying sensor attacks for the three different connection topologies Q introduced in Section 4.4. The results of the numerical simulations are shown in Figures 4.13, 4.14, 4.15. Table 4.6 illustrates the respective mean values, whereas Table 4.7 contains the additional metrics defined to investigate the consensus performance. The regularization parameters vector Λ and the attack estimate threshold $\hat{a}_{thres}$ maintain the same values illustrated in Section 4.3.1.

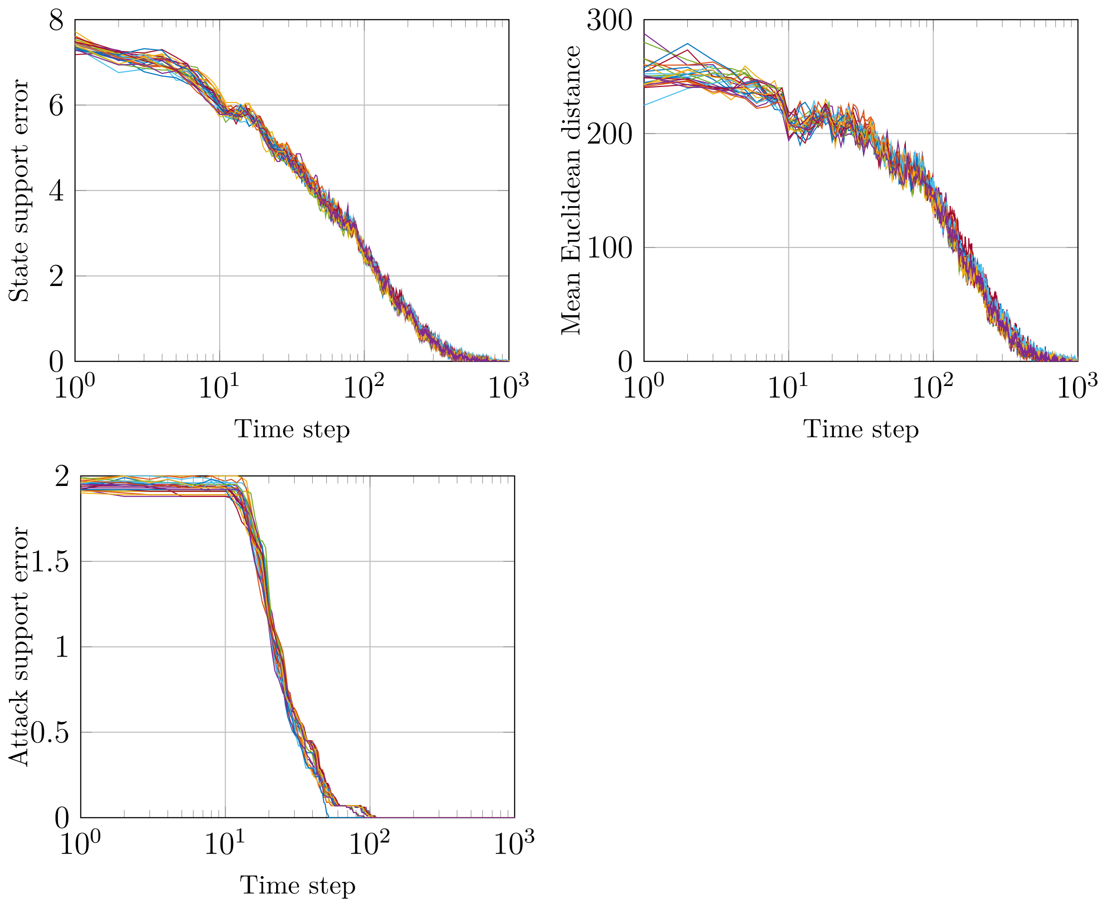


Figure 4.13: Distributed Sparse Observer, time-varying attacks with $p = 100$, $q = 25$, $s = 2$, $j = 4$, $Q = Q^{[6]}$. Each line corresponds to a specific sensor.

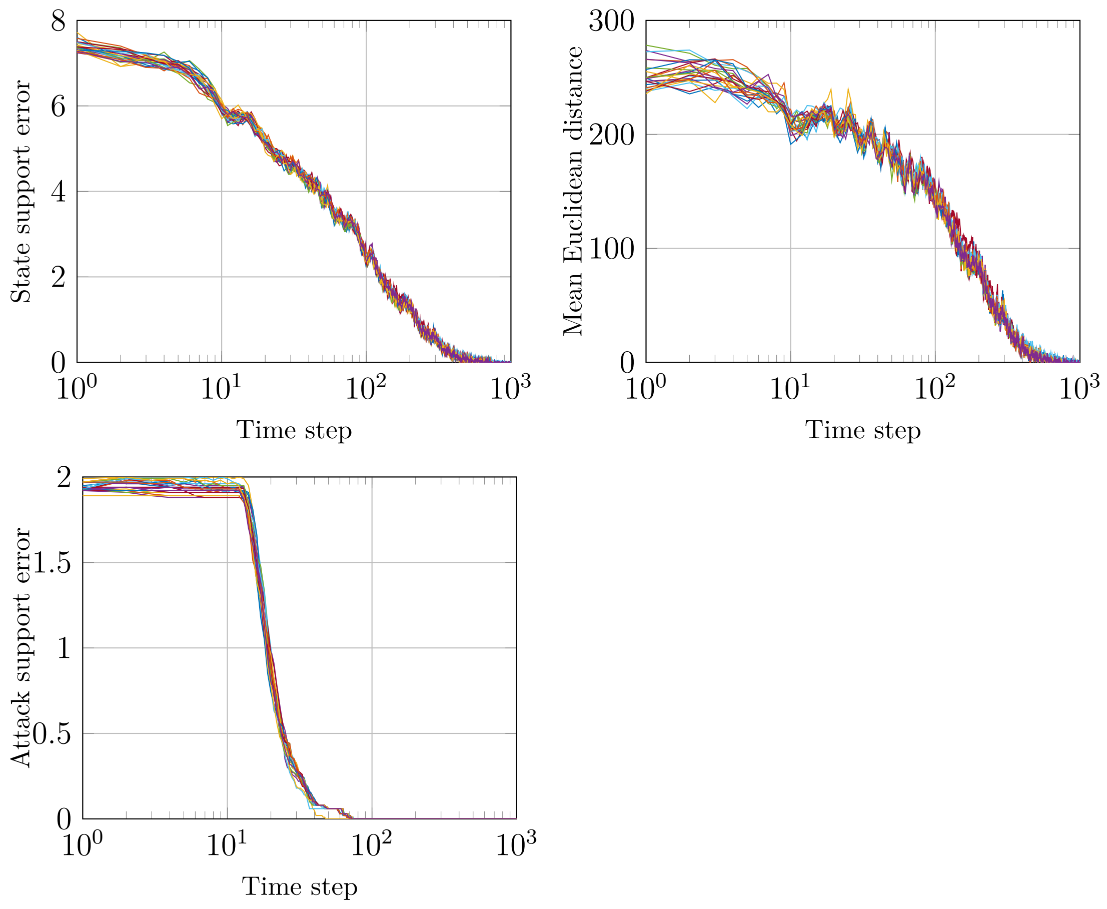


Figure 4.14: Distributed Sparse Observer, time-varying attacks with $p = 100$, $q = 25$, $s = 2$, $j = 4$, $Q = Q^{[11]}$. Each line corresponds to a specific sensor.

	Mean state support error	Mean Euclidean distance	Mean attack support error
$Q = Q^{[6]}$	0.7131	37.3849	0.0541
$Q = Q^{[11]}$	0.7195	38.0911	0.0424
$Q = Q^{[16]}$	0.7092	37.3670	0.0414

Table 4.6: State support error, mean Euclidean distance and attack support error averaged over time and over $q = 25$ sensors for each topology, with $j = 4$ targets (time-varying attacks case).

For any Q, the mean state estimation error and the mean Euclidean distance increase slightly, approximately by 1%, from the time-invariant to the time-varying

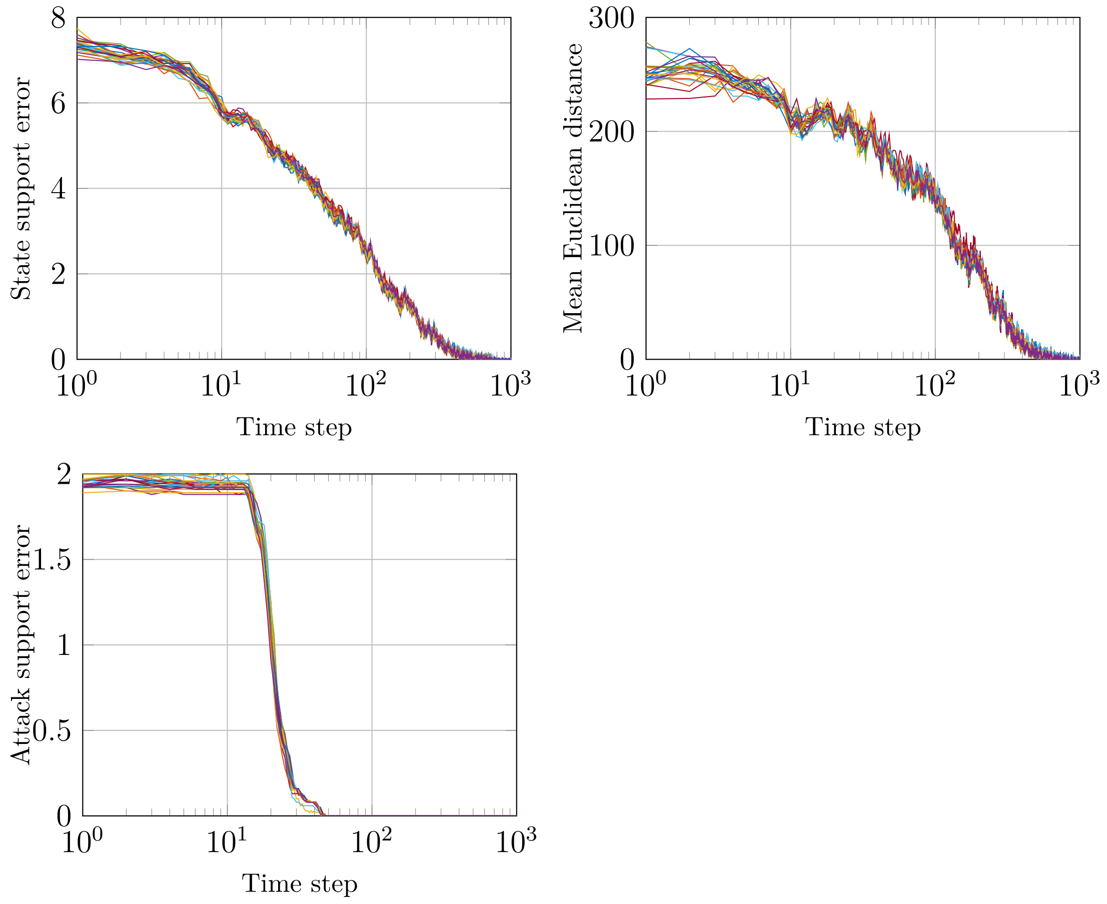


Figure 4.15: Distributed Sparse Observer, time-varying attacks with $p = 100$, $q = 25$, $s = 2$, $j = 4$, $Q = Q^{[16]}$. Each line corresponds to a specific sensor.

attacks case. As regards the mean attack support error, the values are essentially unchanged. In this case too, a more in-depth analysis is needed.

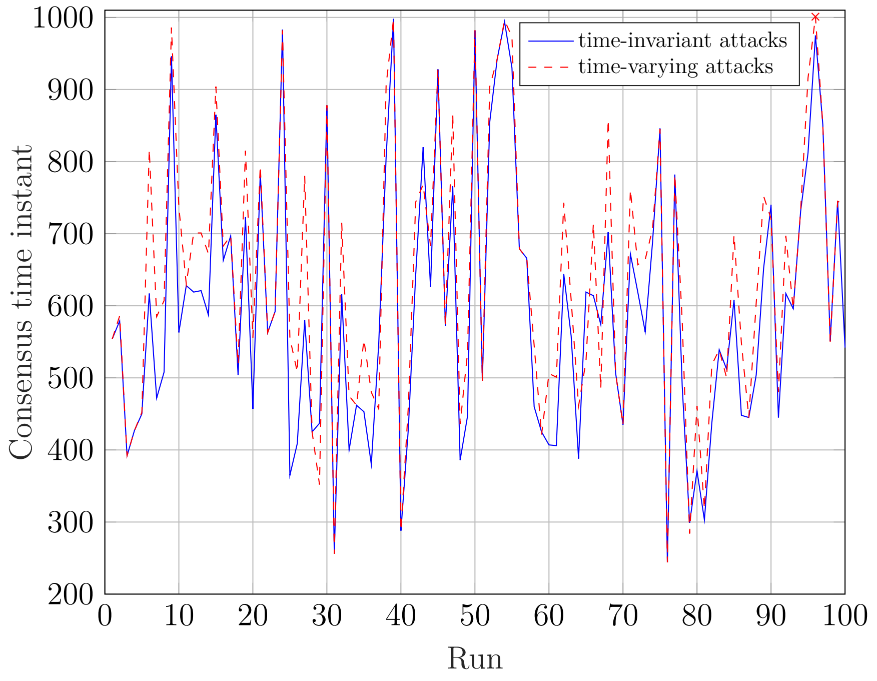
	ϕ_{cons}	$t^{\max}_{cons}$ (s)	t^{mean}_{cons} (s)	MRPD	MRAD
$Q = Q^{[6]}$	1	997	680.89	320.11	961.83
$Q = Q^{[11]}$	0.99	998	635.45	365.55	972.03
$Q = Q^{[16]}$	0.98	998	617.43	383.57	974.10

Table 4.7: Distributed Sparse Observer performance comparison between different topologies, with $j = 4$ targets (time-varying attacks case).

From Table 4.7, $t_{cons}^{\max}$ exhibits the same values as in the constant attacks case for any Q. As regards the other metrics:

- ϕ_{cons} increases from 0.98 to 1 for $Q^{[6]}$, while decreases from 1 to 0.99 for $Q^{[11]}$ and from 1 to 0.98 for $Q^{[16]}$;
- t_{cons}^{mean} increases by 8.92% for $Q^{[6]}$, by 6.58% for $Q^{[11]}$ and by 6.13% for $Q^{[16]}$;
- MRPD decreases by 14.83% for $Q^{[6]}$, by 9.69% for $Q^{[11]}$ and by 8.5% for $Q^{[16]}$;

The mean rate of attack detection is essentially unchanged with respect to the time-invariant attacks case, as previously noted for the mean attack support error. Overall, increasing the number of connections results in less degradation of performance with respect to the constant attacks case, in terms of consensus time instant and position detection. On the other hand, it is possible that consensus is not reached in some of the executions, 1% for $Q^{[11]}$ and 2% for $Q^{[16]}$. A possible solution could be to impose more restrictive assumptions on the topology matrices, e.g. the use of doubly-stochastic matrices, such that the number of outgoing edges is equal to that of incoming edges, for each node.
In Figure 4.16, the consensus time instant t_{cons} and the rate of position detection are represented as a function of the specific execution to graphically visualize the performance differences among the two types of attacks for the standard topology $Q = Q^{[11]}$ (the same trends are observed for the other topologies as well). The "x" marks indicate that consensus has not been reached in that particular run. Also in this case, the rate of position detection is complementary to the consensus time instant with respect to T. Figure 4.17 shows the rate of attack detection computed from the partial consensus time instant.



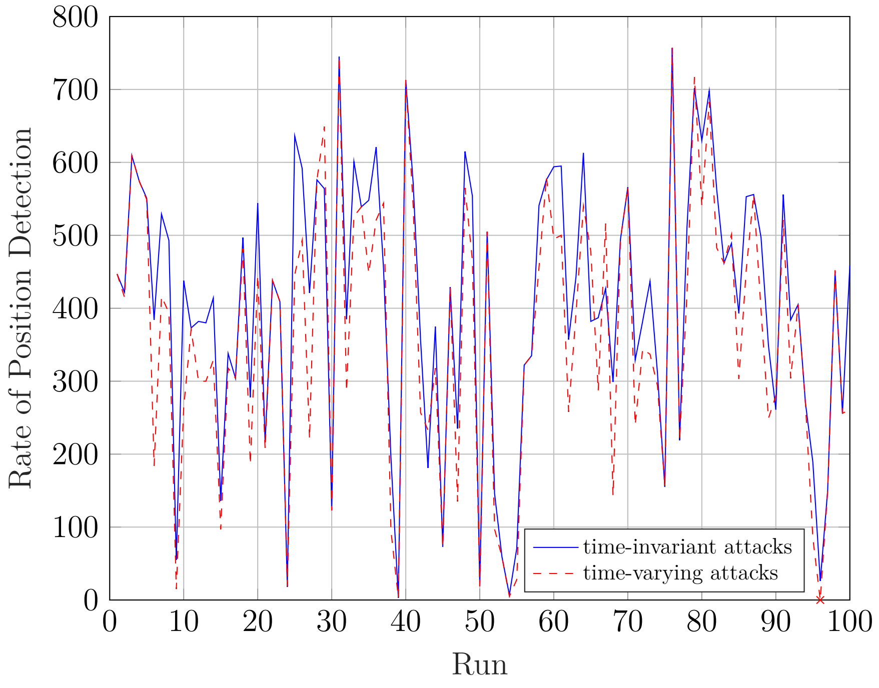


Figure 4.16: Consensus time instant t_{cons} and rate of position detection for each run with $Q = Q^{[11]}$, time-invariant vs time-varying attacks..

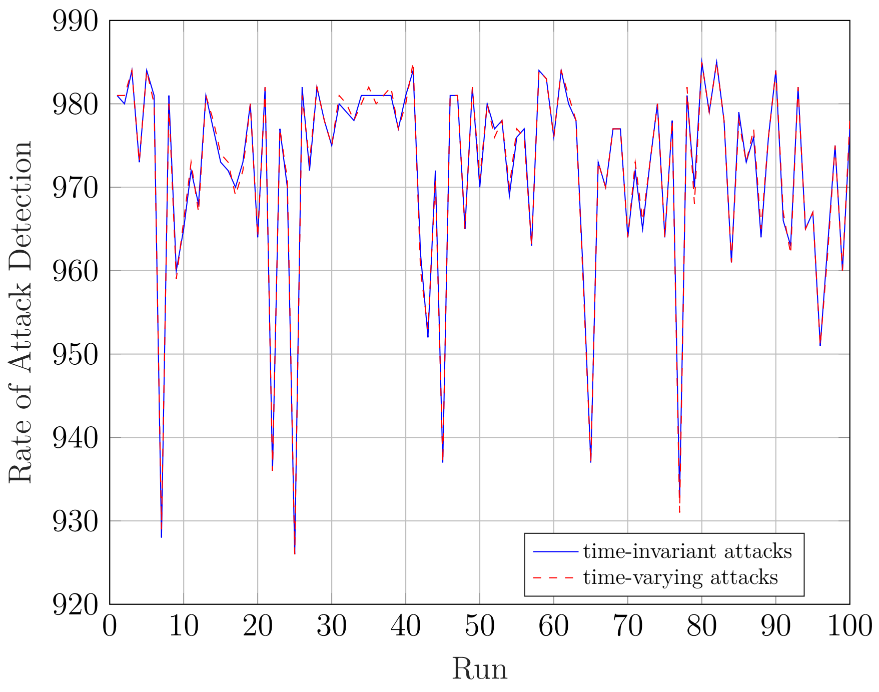


Figure 4.17: Rate of attack detection computed from partial consensus time instant for each run with $Q = Q^{[11]}$, time-invariant vs time-varying attacks..

Chapter 5

Conclusions

In this work, we have first defined the secure state estimation problem from a mathematical perspective, and we have analyzed it on the basis of the existing literature. Subsequently, various algorithms have been compared to solve an illustrative example, with the aim of estimating the non-sparse state of a simulated, random CPS. A Luenberger-like observer, named Sparse Observer, has been implemented using the IST algorithm, whose convergence has been deeply studied and proven in literature. Sparse Observer performance has been compared with the results obtained using the ETPL observer, which adopts event-triggered techniques to enhance the computational performance and is considered a standard algorithm.
The main contribution of our work is the design of a distributed version of the observer, in order to overcome the security and reliability issues arising from the presence of a central computing device. The idea is to decentralize not only measurement acquisition but also computation execution to fully leverage the distributed structure of CPSs. The objective of each agent is to find an estimate of the system state based on local information while simultaneously reaching consensus with other agents. The same Distributed Sparse Observer has then been employed for solving a mobile target localization problem in an indoor environment. It results that the distributed approach exhibits satisfactory performance, comparable to that of its centralized counterpart and of the state-of-the-art ETPL observer, considering that, unlike the latter, it is not assumed to know a tight upper bound on the number of sensors under attack. Moreover, each node has only a partial knowledge of the problem and a common solution is obtained through the exchange of information among connected agents.
Years of research have led to highly efficient algorithms, which require the presence of a centralized FC that has access to global information and measurements received by the sensors. On the other hand, a standard regarding the distributed approach has not yet been established in the literature and convergence conditions for the distributed algorithms are more challenging to formulate and demonstrate.

This thesis aims to provide a possible development in this context, providing a distributed algorithm based on consensus between agents, which allows compliance with privacy and resilience requirements in the event of failures and does not require long-range information exchanges. Future work could be to employ the Distributed Sparse Observer algorithm in on-field tests to evaluate its performance on real-world problems, e.g. a real experimental localization problem could be addressed.

Bibliography

[1] Edward Ashford Lee and Sanjit Arunkumar Seshia. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach.* 2nd ed. MIT press, 2017 (cit. on p. 1).

[2] Rajeev Alur. *Principles of Cyber-Physical Systems.* MIT Press, 2015 (cit. on p. 1).

[3] Kyoung-Dae Kim and P. R. Kumar. «Cyber–Physical Systems: A Perspective at the Centennial». In: *Proceedings of the IEEE* 100.Special Centennial Issue (2012), pp. 1287–1308. DOI: 10.1109/JPROC.2012.2189792 (cit. on p. 1).

[4] Syed Hassan Ahmed, Gwanghyeon Kim, and Dongkyun Kim. «Cyber Physical System: Architecture, applications and research challenges». In: *2013 IFIP Wireless Days (WD).* 2013, pp. 1–5. DOI: 10.1109/WD.2013.6686528 (cit. on p. 1).

[5] Edward A. Lee. «Cyber Physical Systems: Design Challenges». In: *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC).* 2008, pp. 363–369. DOI: 10.1109/ISORC.2008.25 (cit. on p. 2).

[6] An-Yang Lu and Guang-Hong Yang. «A Polynomial-Time Algorithm for the Secure State Estimation Problem Under Sparse Sensor Attacks via State Decomposition Technique». In: *IEEE Transactions on Automatic Control* 68.12 (2023), pp. 7451–7465. DOI: 10.1109/TAC.2023.3278839 (cit. on pp. 2, 4).

[7] Hamza Fawzi, Paulo Tabuada, and Suhas Diggavi. «Secure Estimation and Control for Cyber-Physical Systems Under Adversarial Attacks». In: *IEEE Transactions on Automatic Control* 59.6 (2014), pp. 1454–1467. DOI: 10.1109/TAC.2014.2303233 (cit. on pp. 2, 4, 7, 8).

[8] Stefano Zanero. «When Cyber Got Real: Challenges in Securing Cyber-Physical Systems». In: *2018 IEEE SENSORS.* 2018, pp. 1–4. DOI: 10.1109/ICSENS.2018.8589798 (cit. on p. 2).

[9] Seyed Mehran Dibaji, Mohammad Pirani, David Bezalel Flamholz, Anuradha M. Annaswamy, Karl Henrik Johansson, and Aranya Chakrabortty. «A systems and control perspective of CPS security». In: *Annual Reviews in Control* 47 (2019), pp. 394–411. ISSN: 1367-5788. DOI: `https://doi.org/10.1016/j.arcontrol.2019.04.011` (cit. on p. 2).

[10] Yanwen Mao, Aritra Mitra, Shreyas Sundaram, and Paulo Tabuada. «On the computational complexity of the secure state-reconstruction problem». In: *Automatica* 136 (2022), p. 110083. ISSN: 0005-1098. DOI: `https://doi.org/10.1016/j.automatica.2021.110083` (cit. on p. 3).

[11] Yasser Shoukry and Paulo Tabuada. «Event-Triggered State Observers for Sparse Sensor Noise/Attacks». In: *IEEE Transactions on Automatic Control* 61.8 (2016), pp. 2079–2091. DOI: `10.1109/TAC.2015.2492159` (cit. on pp. 4, 7, 12, 13).

[12] Yasser Shoukry, Pierluigi Nuzzo, Alberto Puggelli, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, and Paulo Tabuada. «Secure State Estimation for Cyber-Physical Systems Under Sensor Attacks: A Satisfiability Modulo Theory Approach». In: *IEEE Transactions on Automatic Control* 62.10 (2017), pp. 4917–4932. DOI: `10.1109/TAC.2017.2676679` (cit. on p. 4).

[13] Giuseppe Notarstefano, Ivano Notarnicola, and Andrea Camisa. *Distributed Optimization for Smart Cyber-Physical Networks.* 2019 (cit. on pp. 4, 14–16).

[14] Angelia Nedic and Ji Liu. «Distributed Optimization for Control». In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (May 2018), pp. 77–103. DOI: `10.1146/annurev-control-060117-105131` (cit. on pp. 4, 16).

[15] Chiara Ravazzi, Sophie Marie Fosson, and Enrico Magli. «Distributed soft thresholding for sparse signal recovery». In: *2013 IEEE Global Communications Conference (GLOBECOM).* 2013, pp. 3429–3434. DOI: `10.1109/GLOCOM.2013.6831603` (cit. on pp. 4, 17).

[16] Yanwen Mao and Paulo Tabuada. «Decentralized Secure State-Tracking in Multiagent Systems». In: *IEEE Transactions on Automatic Control* 68.7 (2023), pp. 4053–4064. DOI: `10.1109/TAC.2022.3200951` (cit. on pp. 5, 14).

[17] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing.* Birkhäuser Basel, 2013. DOI: `10.1007/978-0-8176-4948-7` (cit. on p. 9).

[18] Emmanuel J. Candes and Terence Tao. «Decoding by linear programming». In: *IEEE Transactions on Information Theory* 51.12 (2005), pp. 4203–4215. DOI: `10.1109/TIT.2005.858979` (cit. on p. 9).

[19] Robert Tibshirani. «Regression Shrinkage and Selection via the Lasso». In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. URL: `http://www.jstor.org/stable/2346178` (cit. on p. 9).

[20] Ingrid Daubechies, Michel Defrise, and Christine Mol. «An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint». In: *Communications on Pure and Applied Mathematics* 57 (Nov. 2004). DOI: `10.1002/cpa.20042` (cit. on p. 10).

[21] Amir Beck and Marc Teboulle. «A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems». In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202. DOI: `10.1137/080716542` (cit. on p. 10).

[22] Vito Cerone, Sophie Marie Fosson, Diego Regruto, and Francesco Ripa. «Lasso-based state estimation for cyber-physical systems under sensor attacks». Submitted to IFAC SYSID Conference. 2024 (cit. on pp. 11, 20).

[23] Federica Garin and Luca Schenato. «A Survey on Distributed Estimation and Control Applications Using Linear Consensus Algorithms». In: *Networked Control Systems*. Ed. by Alberto Bemporad, Maurice Heemels, and Mikael Johansson. London: Springer London, 2010, pp. 75–107. ISBN: 978-0-85729-033-5. DOI: `10.1007/978-0-85729-033-5_3` (cit. on p. 16).

[24] Chiara Ravazzi, Sophie Marie Fosson, and Enrico Magli. «Distributed iterative thresholding for l0/l1-regularized linear inverse problems». In: *IEEE Transactions on Information Theory* 61.4 (2015), pp. 2081–2100. DOI: `10.1109/TIT.2015.2403263` (cit. on p. 17).

[25] Alessandro Bay, Diego Carrera, Sophie Fosson, Pasqualina Fragneto, Marco Grella, Chiara Ravazzi, and Enrico Magli. «Block-sparsity-based localization in wireless sensor networks». In: *EURASIP Journal on Wireless Communications and Networking* 2015 (June 2015), p. 182. DOI: `10.1186/s13638-015-0410-6` (cit. on pp. 50–52).

[26] Jing Wang, Ratan Ghosh, and Sajal Das. «A survey on sensor localization». In: *Journal of Control Theory and Applications* 8 (Feb. 2010), pp. 2–11. DOI: `10.1007/s11768-010-9187-7` (cit. on p. 50).

[27] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. «Survey of Wireless Indoor Positioning Techniques and Systems». In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.6 (2007), pp. 1067–1080. DOI: `10.1109/TSMCC.2007.905750` (cit. on p. 50).

[28] Sofia Nikitaki and Panagiotis Tsakalides. «Decentralized indoor wireless localization using compressed sensing of signal-strength fingerprints». In: *Proceedings of the 7th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks.* PM2HW2N '12. Paphos, Cyprus: Association for Computing Machinery, 2012, pp. 37–44. ISBN: 9781450316262. DOI: `10.1145/2387191.2387198` (cit. on pp. 50, 52).

[29] Yunhao Liu and Zheng Yang. *Location, Localization, and Localizability: Location-awareness Technology for Wireless Networks.* Jan. 2011, pp. 1–154. ISBN: 978-1-4419-7370-2. DOI: `10.1007/978-1-4419-7371-9` (cit. on p. 51).

[30] Sofia Nikitaki and Panagiotis Tsakalides. «Localization in wireless networks via spatial sparsity». In: *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers.* 2010, pp. 236–239. DOI: `10.1109/ACSSC.2010.5757507` (cit. on pp. 51, 52).