Modeling and Control of Cyberphysical systems

AY. 2023/24

Federico Morro - s324114
Francesco Antonio Novia - s329754
Giovanni Tortia - s328867

Project report

Part 1

Prof. S.M. Fosson

Task 1 - Implementation of ISTA

The target of this task was to use the ISTA (Iterative Soft Thresholding Algorithm) to recover the state of an h-sparse state vector x (of dimension p), subject to a certain noise (a gaussian noise in this specific implementation).

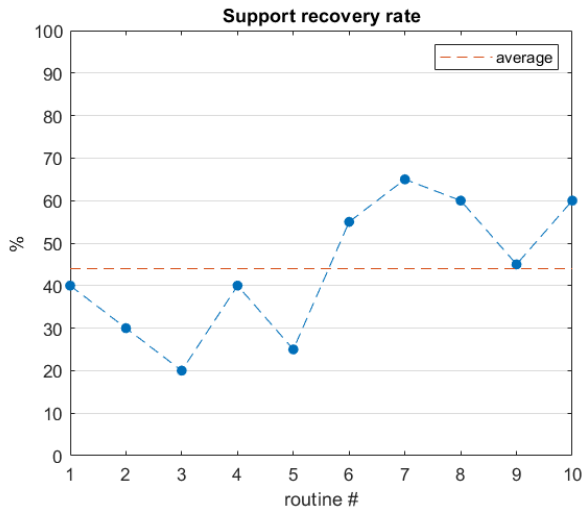
$$\min_{x \in \mathbb{R}^p} ||Cx - y||_2^2 + \lambda ||x||_1$$

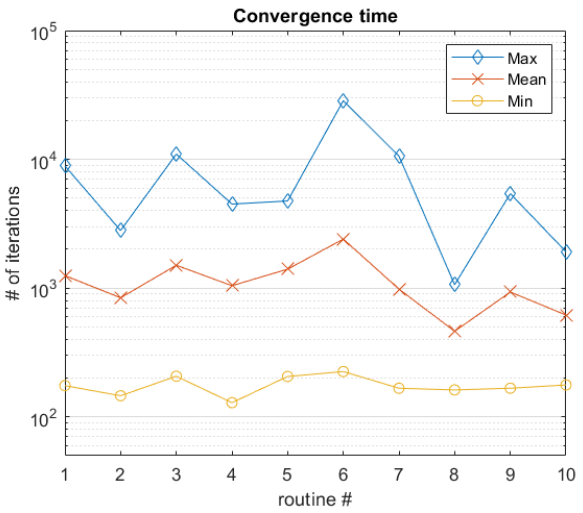
Assuming $h \ll p$, we could run several simulations and evaluate some global indices of accuracy:

- the support recovery rate by simply comparing the number of non-zero elements in the estimated vector with the value of h, at the end of the algorithm iterations;
- the convergence time, in term of number of iteration needed to achieve a given stability of the solution;
- the variation of the results when changing the hyperparameters value.

In order to obtain a complete overview of the results, we first performed 10 routines, each one made of 20 simulations, keeping the hyperparameters values constant (suggested setting described in task specifications).

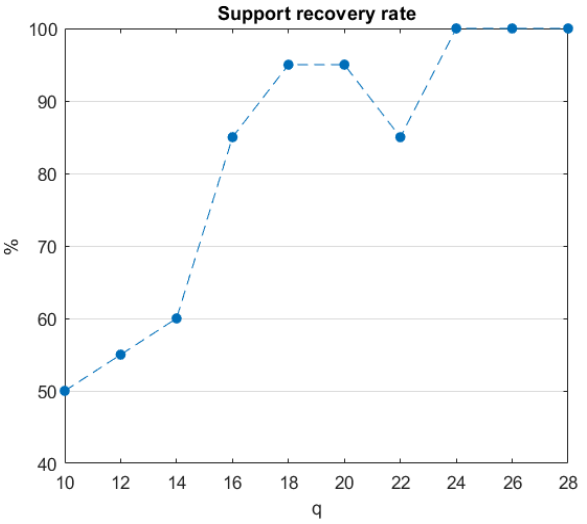
The obtained results are displayed in the following figures:

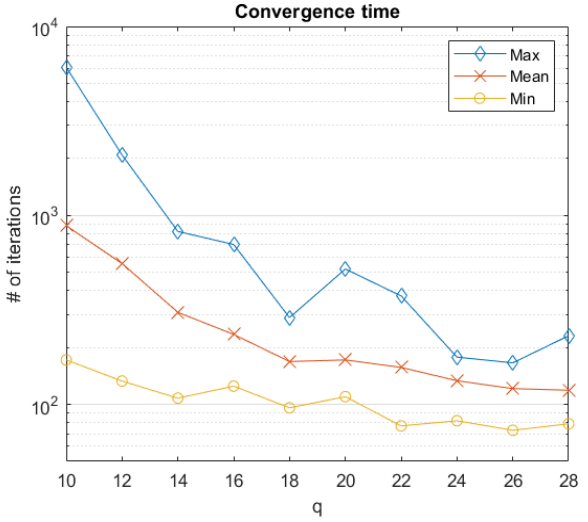




We can observe that with this setting, the recovery rate is about 40-50% on average, with peaks of over 60%, while the mean convergence time is of the order of the thousands of iterations (~10^3).

Then, we analyzed the previous statistics while increasing the value of q (corresponding to the number of sensors), always in groups of 20 simulations per routine:

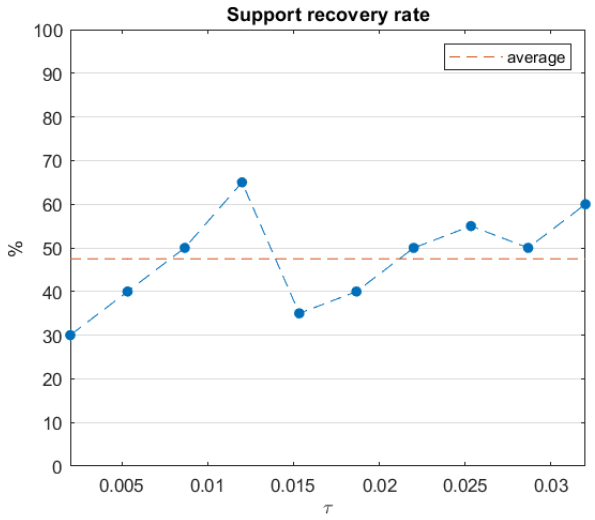


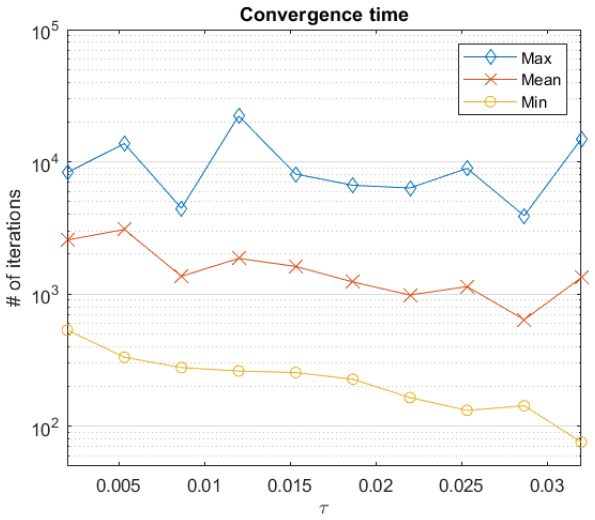


It is quite clear that the algorithm performance gets significantly better when q becomes larger: it is then possible to achieve a perfect estimate in terms of support recovery (100% for the last three routines) and also a definitely lower convergence time (in the order of the hundreds of iterations). This is clearly due to the fact that incrementing the number of sensors, and thus of measurements performed, makes the estimation of the same number of variables easier even in presence of noise.

Another analogue simulation was performed while varying the value of τ while keeping the product $\tau\lambda$ constant. We were forced to choose a quite narrow range for τ (in particular $[0.002 - 0.032]$) so as to avoid generating numerical errors during calculations,

which would compromise the convergence of the algorithm. The figures below show the summary of results:

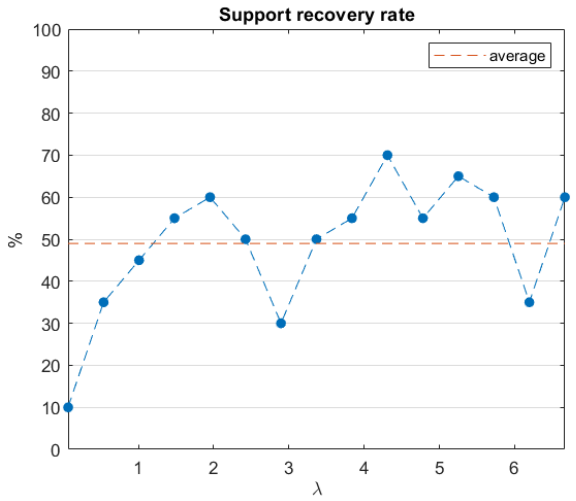


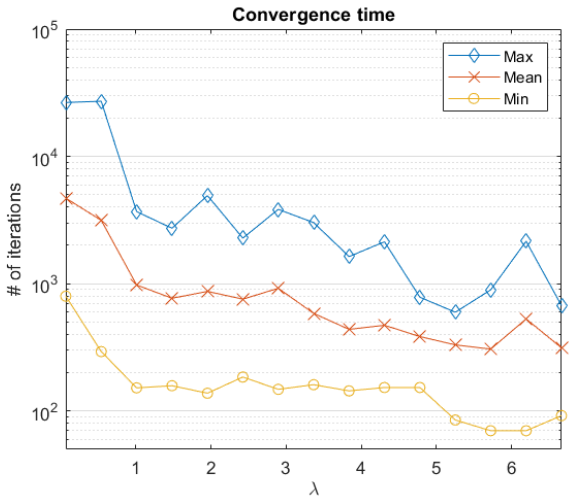


We can observe that, save for a slight improvement in the minimum and (even less) in the mean convergence time, the behavior of the algorithm is only slightly affected by this specific parameter variation.

Actually the two parameters τ and λ play an essential role in ISTA: from theory we know that their product τλ establishes the width of the gradient descent and shrinkage-thresholding steps. This may explain the previous results, obtained with constant τλ product, and so we could expect a smaller convergence time when, for instance, increasing λ.

So our last simulation was properly intended to analyze the dependency on λ, but this time keeping τ constant (in this implementation $\tau = 0.015$).





While on one hand the global accuracy, in terms of support recovery rate, slightly increases, on the other hand the convergence time is subject to a more marked improvement, as predicted.

Task 2 - Secure estimation under sparse sensor attacks

After having analyzed the general performance of the ISTA algorithm, a further step was exploiting it to estimate a **non**-sparse state vector x using measurements provided by q sensors which are under **sparse** attacks.

$$\min_{x \in \mathbb{R}^p} \frac{1}{2} ||Cx - y||_2^2 + \sum_{i=1}^{p} \lambda_i |x_i|$$

This time, in order to evaluate the accuracy of the algorithm, two different statistics were taken into account:

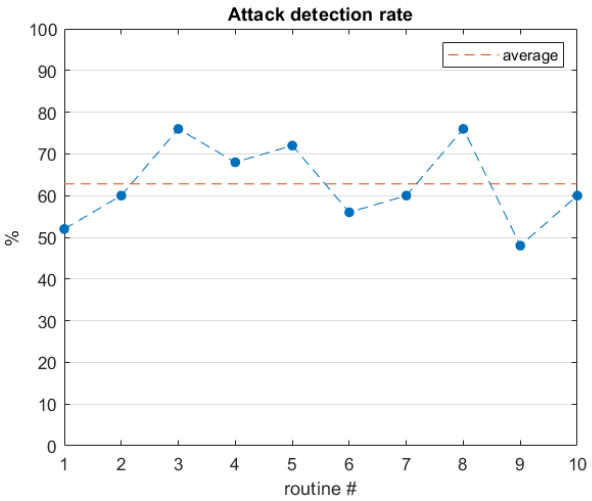
- ➢ detection of attacked sensors, as the number and index of non-zero components of the estimated attacks array a, with respect to the real ones;
- ➢ estimation accuracy, computed as the square distance between the real state vector and the one estimated starting from corrupted sensor measurements.

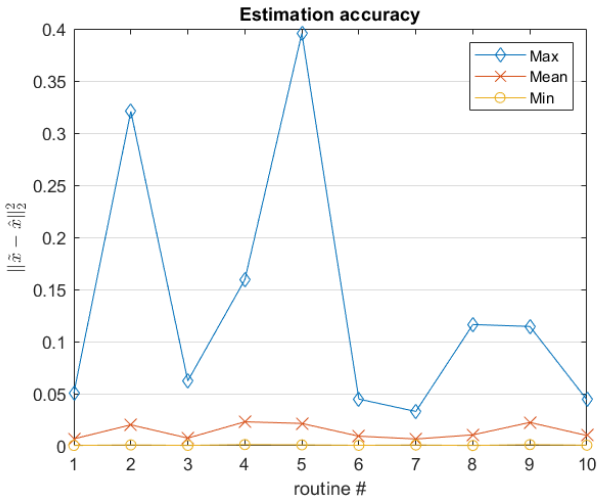
Moreover, we considered two different configurations corresponding to distinct attack typologies:

- unaware attacks: random generated (uniform distribution);
- aware attacks: generated as a function of actual sensor measurements (in particular $a_i = y_i/2$).

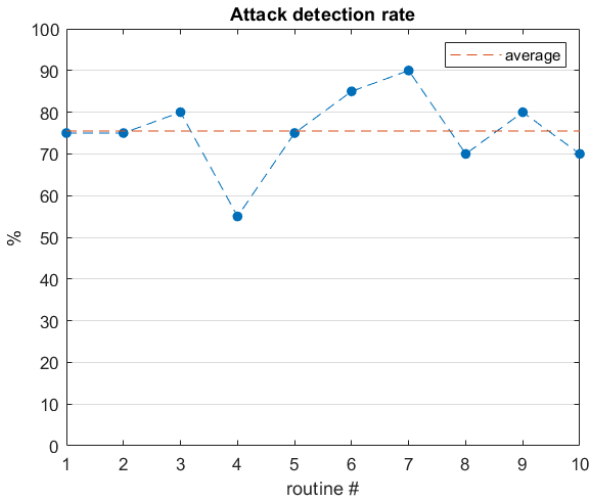
Similarly to the previous task, we run 10 different routines of 20 simulations each, for both attack configurations described above. The figures below collect the relevant statistics:

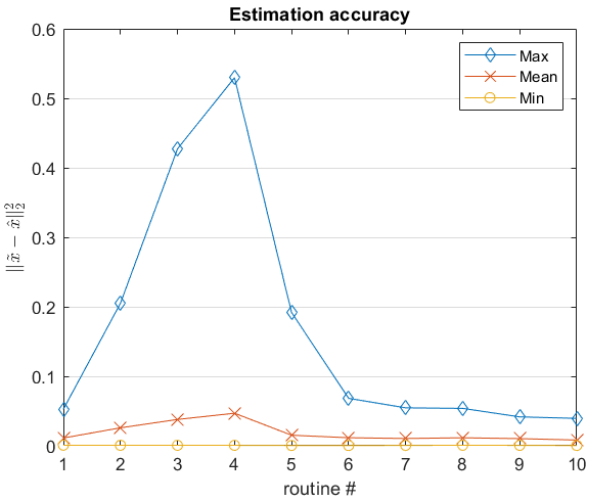
1. Aware attacks





2. Unaware attacks





In case of aware attacks the algorithm achieves decent results for both detection rate ($\sim$65% on average) and estimation accuracy (average distance always less than 0.03); in the case of unaware attacks, however, the measurements corruption becomes more stationary and independent, which leads to an even better performance, with a peak of 90% in attack detection while maintaining estimation accuracy values similar to the previous ones.

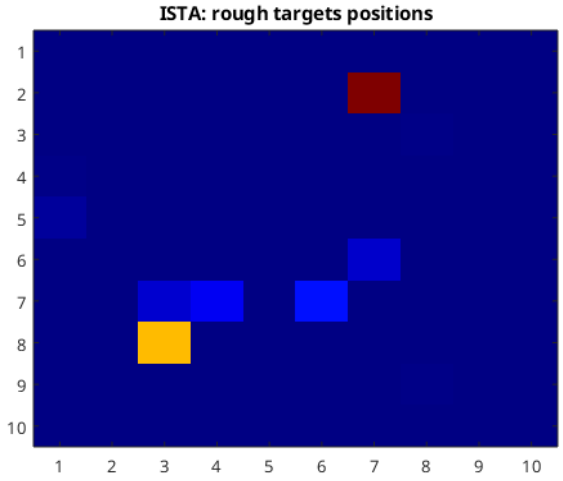
Task 3 - Target localization under sparse sensor attacks

This task required localizing 3 targets using RSS fingerprinting with a number of sensors under adversarial attack. This can be achieved by solving the minimization problem:

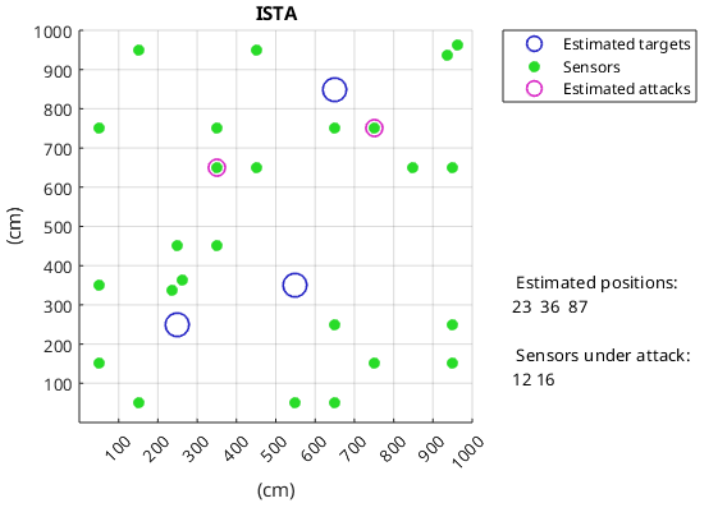
$$\min_{x\in\mathbb{R}^p, a\in\mathbb{R}^q} \left|\left| \begin{pmatrix} D & I \end{pmatrix} \begin{pmatrix} x \\ a \end{pmatrix} - y \right|\right|_2^2 + \lambda_1 ||x||_1 + \lambda_2 ||a||_1$$

Which can be done through the ISTA algorithm. This results in a block vector composed by the rough estimate of the positions and by the estimate of which sensors are being attacked. Since both of these vectors are estimates, only values higher than a threshold should be considered: for the attack vector we arbitrarily set this to 10^{-3}, whereas for the position vector we chose the 3 highest values. This approach results in the positions being correctly estimated as [23, 36, 87], while the attacked sensors are estimated to be [12, 16].

From the rough estimate obtained with the ISTA algorithm (displayed below with a color map), it is clear that without knowing the actual number, 2 targets are clearly estimated, whereas some cells have pretty close estimated values, so one might assume there to be 4 or 5 targets in total.



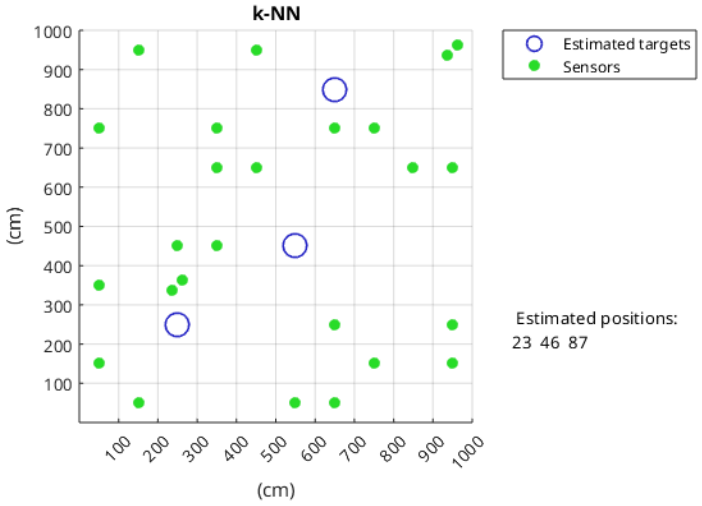
Exploiting the knowledge about the number of targets in post-processing, the result obtained is the following:



Another possible approach is to use the k-Nearest Neighbors algorithm, which consists in trying all the possible permutations of the positions of the targets, computing what would be the corresponding sensor readings and find the configuration closer to the actual measurement, which minimizes (for 3 targets):

$$\left(\hat{j}_1, \hat{j}_2, \hat{j}_3\right) = \arg \min_{j_1, j_2, j_3 = 1, \ldots, p} ||D_{j_1} + D_{j_2} + D_{j_3} - y||_2^2$$

Compared to the previous approach this is considerably faster: ISTA takes 292680 iterations to obtain a result, while the k-NN algorithm only takes 161700. There is however a tradeoff in accuracy: in this case the result was [23, 46, 87], as unlike ISTA, the k-NN algorithm is not able to account for the attacks, meaning that the estimate of the third target is slightly off. The k-NN considers as a target one of the cells that, using ISTA, had a value close to that of the third target.

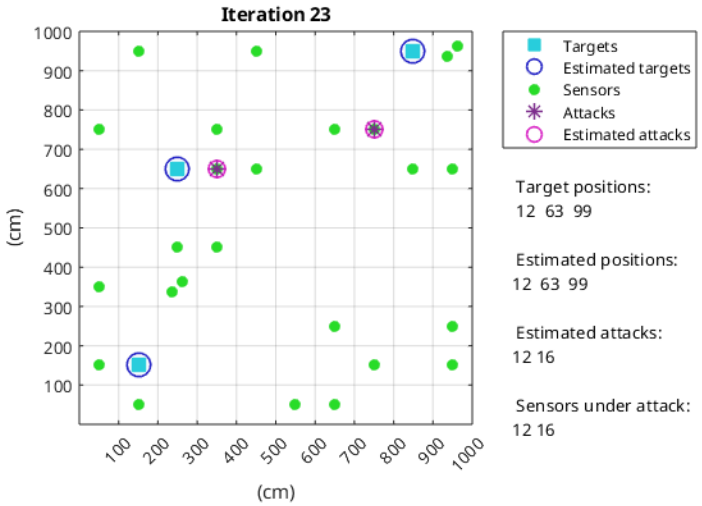


Task 4 - Sparse observer

For this task, 3 moving targets had to be localized by means of a sparse observer based on an on-line optimization algorithm: at each iteration, a better estimate of the targets' positions is computed, using both the readings from the sensors and the system's dynamics.

$$x(k+1) = Ax(k)$$

When under time-invariant attacks, the observer converges within 23 iterations, giving an exact estimate of both the positions and the attacks after filtering out values under a certain threshold, as in the previous task.

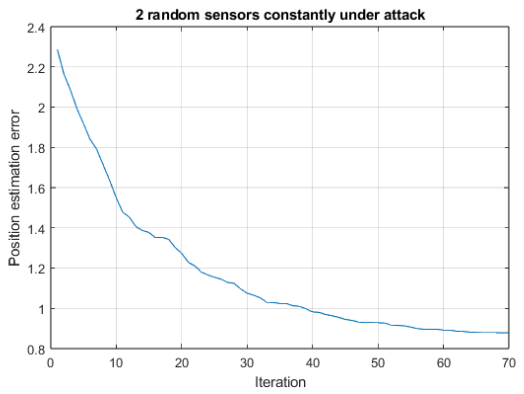


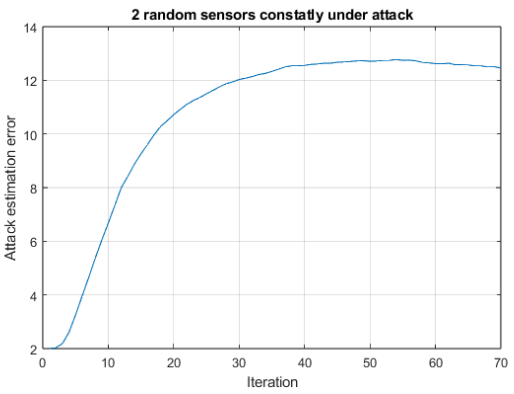
To test the accuracy of the estimates for different time-varying attack methodologies, the following test was set up: $K = 70$ iterations of the algorithm are run, starting from random initial positions, using different attack patterns, for $M = 1000$ different simulations. At each iteration k the mean squared error of the position estimates is measured and averaged over all simulations; since these two vectors are logical (i.e. their only values are zeros and ones), this quantity is equal to twice the amount of wrong estimates, by halving it then we can obtain the average number of incorrectly estimated positions at each iteration.

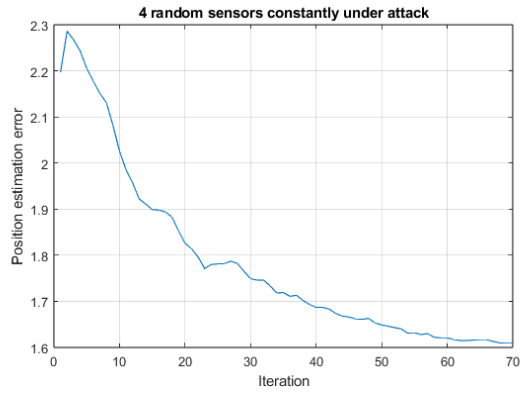
The same approach was also used for attack estimation, in this case however each misplaced attack only increases the error by 1, so it doesn't need to be halved:

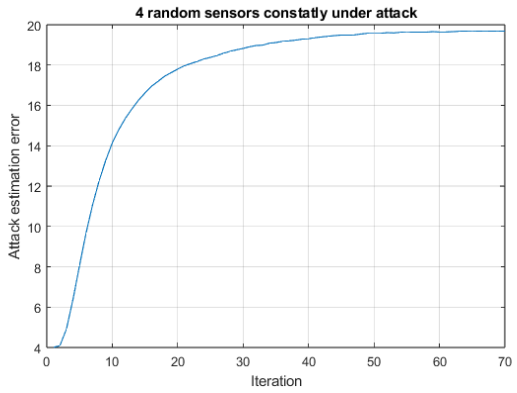
$$\forall k = 1, \dots, K \quad \bar{e}_x(k) = \frac{1}{M} \sum_{sim=1}^{M} \frac{1}{2} ||\hat{x}_{sim}(k) - x(k)||_1 \quad \bar{e}_a(k) = \frac{1}{M} \sum_{sim=1}^{M} ||\hat{a}_{sim}(k) - a(k)||_1$$

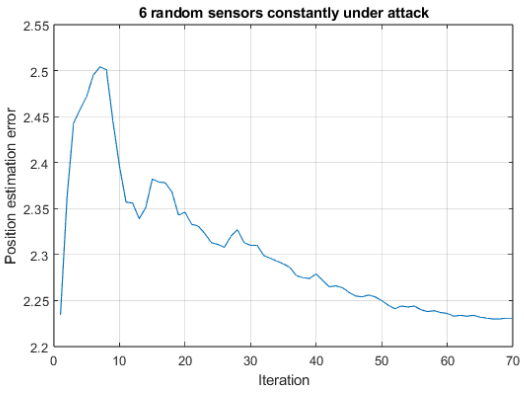
By plotting the results, it can be observed that in all cases the error converges, the final value however gets increasingly bigger with the number of attacked sensors, with 6 being the number of attacks that results in more than 2 targets being misplaced on average. It is interesting to notice how the attack detection is more accurate when the attacked sensors are changing.

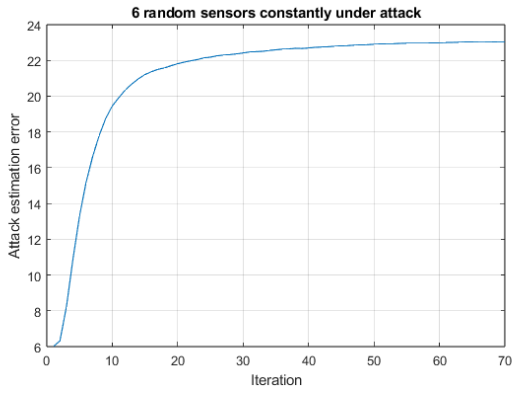

2 random sensors constantly under attack
Position estimation error
Iteration

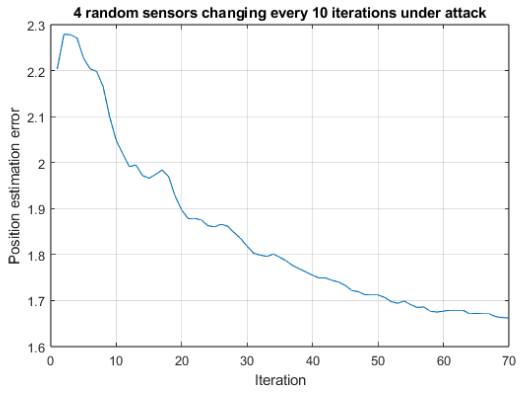

2 random sensors constatly under attack
Attack estimation error
Iteration

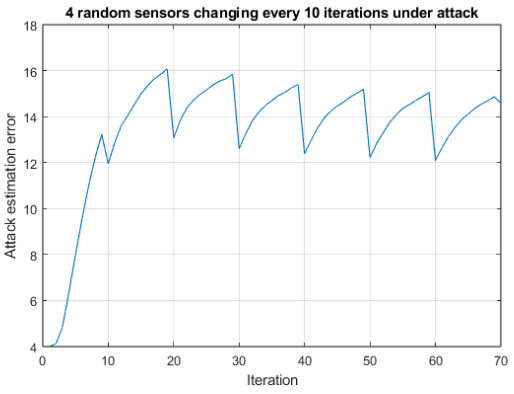

4 random sensors constantly under attack
Position estimation error
Iteration


4 random sensors constatly under attack
Attack estimation error
Iteration


6 random sensors constantly under attack
Position estimation error
Iteration


6 random sensors constantly under attack
Attack estimation error
Iteration

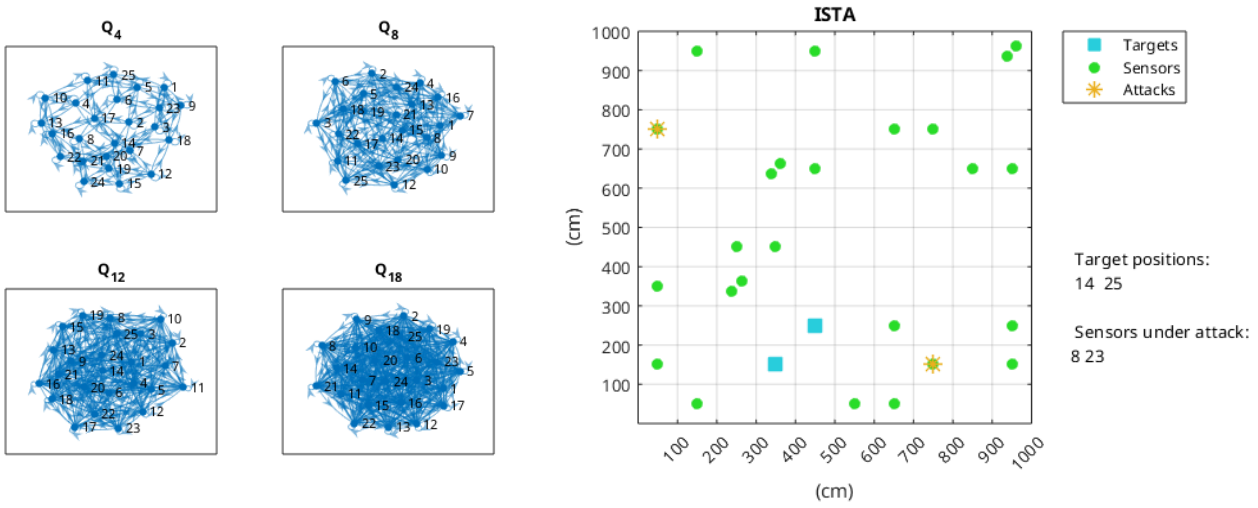

4 random sensors changing every 10 iterations under attack
Position estimation error
Iteration


4 random sensors changing every 10 iterations under attack
Attack estimation error
Iteration

Task 5: Distributed target localization under sparse sensor attacks

The goal of this task is the same as task 3, i.e. the localization of two targets, this time solved in an in-network distributed way, exploiting the DISTA. The task proposed 4 different settings, each described by a stochastic matrix Q which represents the network topology.

We first solved the problem using ISTA to have a solution to which we can compare the estimates obtained by different sensors via DISTA.



After that, we carried out the preliminary analysis on the configurations by performing an analysis of the eigenvalues of the matrices Q. In particular, we are interested in the essential spectral radius, i.e. the eigenvalue with the second highest magnitude (the biggest one will always be 1 due to the properties of the stochastic matrix), since if it has magnitude strictly smaller than 1, we have a sufficient condition for ensuring convergence.

Moreover, it is possible to obtain some information about the convergence speed, since it's inversely proportional to the magnitude of the largest eigenvalues. It is clear from the results that graphs with a higher number of connections between nodes, produce a lower ESR and therefore a faster convergence, which is reasonable since more information is being shared between nodes.

In the table below, we reported the results, comparing each configuration by analyzing:

- the time needed to reach consensus of all sensors on the estimates, separately for x and a, i.e. when all sensors have the same estimates
- the time needed to converge to the exact estimate for all sensors, separately for x and a, i.e. when all sensors have the same estimate, which is the correct one
- the time needed to reach the algorithm termination condition

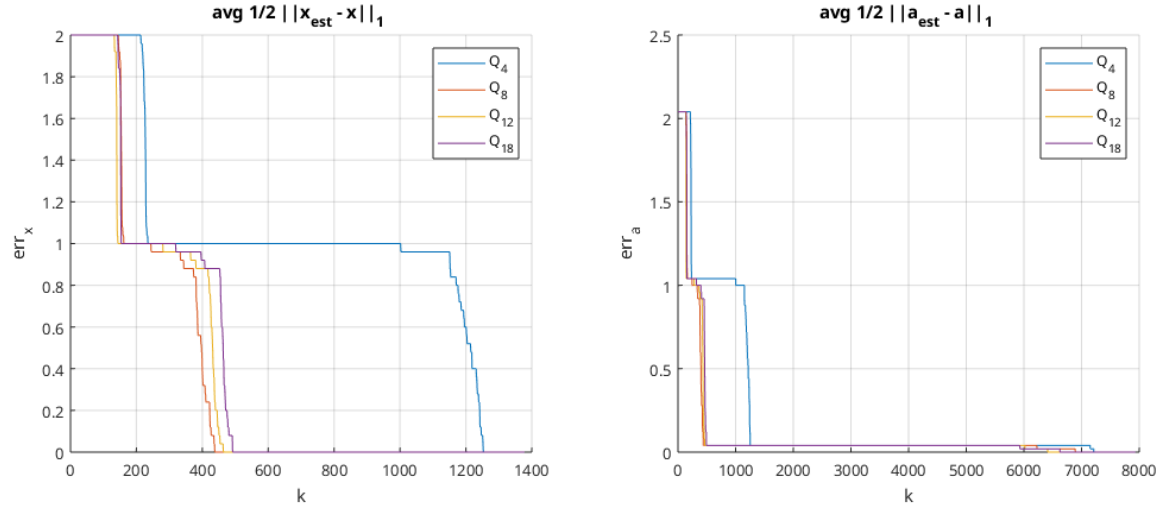
	ESR	$k_{x,\,consensus}$	$k_{a,\,consensus}$	$k_{x,\,conver.}$	$k_{a,\,conver.}$	$k_{term.}$
Q_4	0.7785	22	7'162	1'253	7'212	10'279
Q_8	0.4964	19	6'227	438	6'889	10'499
Q_{12}	0.3701	16	6'020	464	6'413	10'402
Q_{18}	0.2196	12	5'933	492	6'625	10'638

It's clear that the most critical part is the attacks' estimation. The time needed to reach consensus is far higher, although, after reaching it, the convergence to exact values is faster. It's worth noting that the consensus on the targets' positions is reached almost immediately, while the convergence to exact values takes longer, as expected. Moreover, in both cases, it is possible to notice the inverse proportionality that holds between the configuration matrix ESR and the time when consensus is reached; while the convergence to exact values does not follow the same behavior, since in some cases a faster consensus may negatively affect the converging speed to exact estimates.

In the following graph, we report the average over all sensors s (q sensors) of the difference in $l1$-norm between the estimated vectors and the actual vectors, over time k.

$$\forall k = 1, \ldots, K, \quad \bar{e}_{x,k} = \frac{1}{q}\sum_{s=1}^{q}\frac{1}{2}||\hat{x}_s - x||_1, \quad \bar{e}_{a,k} = \frac{1}{q}\sum_{s=1}^{q}\frac{1}{2}||\hat{a}_s - a||_1$$

Due to the fact that the estimated and the correct vector are boolean vectors (i.e. 1 if the target/attack is in the cell/sensor, 0 otherwise), the $l1$-norm is double the number of wrong predictions.



The graphs highlight the convergence trends:

- regarding the estimate of x, the nodes are remarkably fast in getting the correct estimate, except for the Q_4 configuration, in which the convergence takes a while due to 1 target being estimated incorrectly
- regarding the a estimate, instead, the worst case is again given by the Q_4 configuration, but all configurations struggle to get all nodes to achieve the correct estimate, in fact the average error does not go to 0 for thousands of iterations.