IAC-22,A6,IP,65,x69321

GA-based optimal tasking for SST sensor networks in the SENSIT tool

Giovanni Purpura, Andrea De Vittori[a]*, Riccardo Cipollone[a], Luca Facchini [a], Pierluigi Di Lizia[a], Mauro Massari[a], Alessandra Di Cecco[b], Luca Salotti[b]

[a] *Politecnico di Milano, Via Giuseppe La Masa, 34, 20156 Milano, Italy*, andrea.devittori@polimi.it
[b] *Italian Space Agency (ASI), via del Politecnico snc, 00133 Rome, Italy*
* Corresponding Author

Abstract

The ability to simulate the behavior of different sensor configurations is critical for the development of a sensor network that provides data for Space Surveillance and Tracking (SST) services. Any software suite devoted to this shall be able to assess the performance of existing networks in terms of effectiveness and robustness, as well as to estimate the effects of structural changes, such as the addition or the upgrade of sensors. This paper is devoted to describing how SΞNSIT tackles the above problem. SΞNSIT (Space Surveillance Sensor Network SImulation Tool) is a software suite designed to perform an analysis of the observational and cataloging capabilities of a sensor network. The software can model optical, radar and laser ranging sensors and simulate different operational scenarios. The user shall define the sensors composing the network and a population of space objects. Typical sensor properties that can be set include type, mode (survey/tracking), location, accuracy, pointing constraints, detectability limits and operating hours. Inputs are processed to predict transits that can be observed by each sensor. This allows to assess the network capabilities in terms of catalog coverage: the sensors are compared against each other to identify overlapping in the sets of observable objects and estimate the level of complementarity or redundancy. Afterwards, the tool can simulate the operations of the network. First, an observation schedule is compiled, using an optimization algorithm based on tunable criteria. This removes overlaps caused by objects passing at the same time. Then, the software simulates and processes the measurements gathered during passes, carrying out orbit determination, aiming at assessing the network capability in terms of catalog build-up and maintenance. The results are proposed in tables and graphs with different levels of detail, starting from a general performance overview up to the list of the passes. The user can also browse the object catalog of the network and analyze its evolution. Moreover, the tool allows to export intermediate data, such as the observable passes, the optimized schedule, and the pointing requirements. The modularity of the software grants easy modification of the properties of the network, to carry out a sensitivity analysis to different parameters. This is expected to ease the setup process of sensor networks for SST, as well as the identification of the most promising upgrades to be recommended. The paper presents in detail the software architecture and its functionalities, and shows the results provided in typical use cases.

Keywords: Sensor networks; Space object cataloguing; Space Surveillance and Tracking; Space Situational Awareness; Genetic algorithm

Acronyms/Abbreviations

Comma-Separated Values (CSV); European Space Agency (ESA); Field Of Regard (FOR); Field Of View (FOV); Graphical User Interface (GUI); Initial Orbit Determination (IOD); JavaScript Object Notation (JSON); Local Orbital Frame (QSW); Non-Linear Least Squares (NLS); Orbit Determination (OD); Radar Cross Section (RCS); Refined Orbit Determination (ROD); Space Surveillance and Tracking (SST); Two-Line Element set (TLE)

1. Introduction

Space-Based Assets are the gateway to providing services worldwide and are key to the progress of global economies. Indeed, satellite constellations serve for communication, navigation, and observation purposes. That said, thousands of satellites are being launched globally, increasing the population of objects around the Earth. Only 13% of these are actively controlled [1]; the rest could potentially damage orbiting assets jeopardizing communication networks with the Earth.

Developing an effective air traffic control system for space would benefit ground and orbiting infrastructures. To lessen this threat, sensor networks for surveying and tracking such objects are of utmost importance, as well as informing the stakeholders. Cataloguing such objects is strictly related to the quality and cooperation of space surveillance

systems. Knowing through dedicated software simulations how to optimally operate existing sensor networks and where to locate new ones can have a substantial impact on catalogue build-up and maintenance.

At the European level, BAS3E (Banc d'Analyse et de Simulation d'un Systeme de Surveillance de l'Espace - Simulation and Analysis Bench for Space Surveillance System) and S3TOC (Spanish Space Surveillance and Tracking Operations Center) are two examples of available sensor network simulation tools.

BAS3E, developed by CNES [2], is a complete SST simulation framework conceived to evolve the current SST network, both software, and hardware-wise. It features:

- Detection, tracking and generation of observations of space objects
- Object identification and tracking correlation
- Orbit determination
- Maintenance of a space debris catalogue
- Centralized / de-centralized tasking and scheduling

S3TOC locates in the Torrejón de Ardoz Military Air Base [3] in Spain, and it consists of the following elements:

- Data Processing and Cataloguing
- Service Processing
- Sensor Planning and Tasking
- Fragmentation messages
- Service Provision

The Italian SΞNSIT is a software tool akin to its European counterparts. It models sensor networks and assesses their performance in coverage and catalogue maintenance terms. Thanks to its intuitive interface, the user can perform simulations and sensitivity analyses by tweaking the network configuration.

2. SΞNSIT

Fig. 1 SΞNSIT logo.

The Space Surveillance Sensor Network Simulation Tool (SΞNSIT) is a software tool conceived by Politecnico di Milano in cooperation with the Italian Space Agency and with contributions from the SpaceDyS company. A preliminary version of SΞNSIT has already been presented in [4]. This work shows SΞNSIT building blocks with dedicated simulations in case of optimized schedules.

SΞNSIT is designed in Python 3 and C++; it is compatible with Windows, macOS, and Linux. From an astronomical computation standpoint, it relies on the NASA/NAIF SPICE library [5].

The sensor characteristics and configuration files are saved in YAML format, while SQLite databases store intermediate steps to alleviate the overall computational burden.

SΞNSIT operates in two modes: the first is through a Graphical User Interface (GUI) based on the Qt library and the other by command line.

Given a collection of space objects, a sensor network, and a simulation time frame, SΞNSIT performs the following tasks:

- observable transits computation of space objects over the selected ground stations,
- optimal GA-based observation schedules according to user-defined criteria
- measurements emulation
- orbit determination with synthetic measurements provided the sensor accuracies
- catalogue build-up and maintenance according to the orbit determination performance

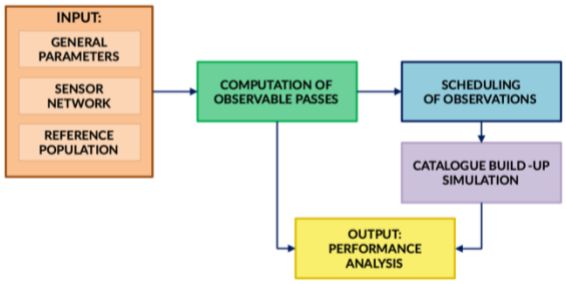

Fig. 2 SΞNSIT architecture.

SΞNSIT is made of five distinct modules, as depicted in *Fig. 2 SΞNSIT architecture*.

- **Data initialization**: It collects and organizes the provided inputs. Additionally, it updates the SPICE kernels (files comprising planetary ephemerides and leap second information).
- **Pass computation**: objects passages evaluation belonging to the reference population. It accounts for imposed observability constraints.
- **Scheduling of observations**: By selecting a subset of previously computed passages, the algorithm produces optimal observation schedules via a genetic algorithm.
- **Catalogue build-up**: starting from a schedule, it simulates measurements needed for network catalogue build-up and maintenance.

- **Performance analysis**: data analysis block with tables and charts reporting an overview of the sensor network performance.

2.1 Data initialization

The data initialization block manages the software setup, loading the reference population of objects and SPICE kernels

2.1.1 Configuration

The GUI configuration tab (*Fig. 3*) is intended to set these properties:

- Generic simulation parameters tuning in accordance with the available computational power,
- accuracy thresholds to establish if an object is still catalogued in the light of its propagated position covariance,
- Simulation time window,
- Space-Track login credentials for automatic download of TLEs (optional).

The GUI stores the parameters in editable YAML files.

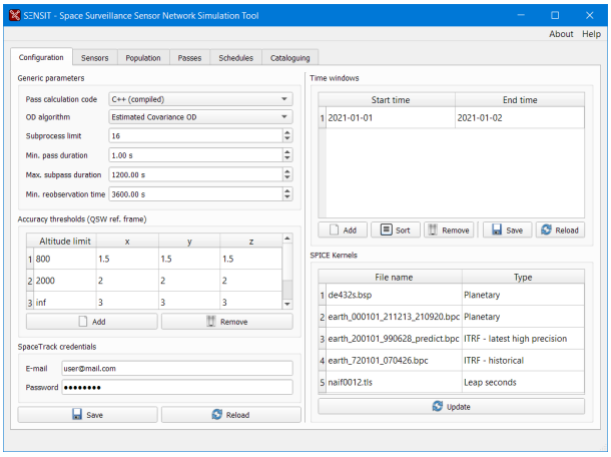

Fig. 3 SΞNSIT GUI: configuration tab.

2.1.2 Sensors

The sensor network configures either through the GUI (*Fig. 4*) or employing a YAML file formatted in line with the software manual.

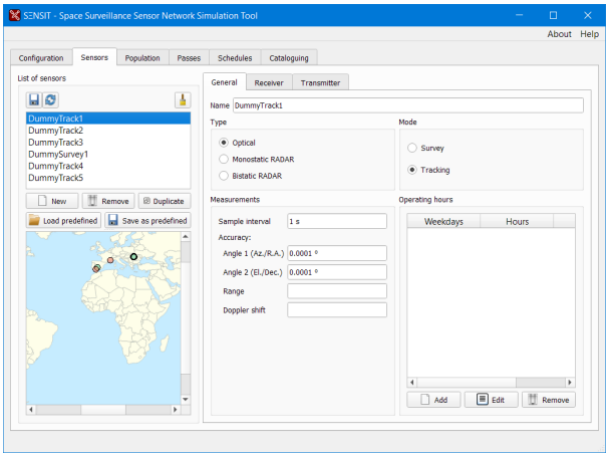

Fig. 4 SΞNSIT GUI: sensors tab – general.

The parameters for each sensor are:

- Name
- Type (optical, radar mono/bistatic)
- Mode (tracking, survey)
- Working hours (optional)
- Measurement sample interval
- Measurement accuracies

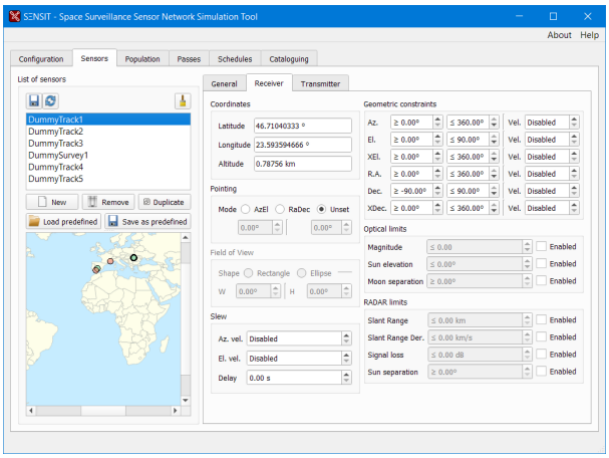

Fig. 5 SΞNSIT GUI: sensors tab – receiver.

For the receiver (*Fig. 5*), the inputs are:

- Geographical coordinates
- Pointing and Field of View (for survey sensors)
- Slew speed (for tracking sensors)
- Geometrical constraints (e.g., Field of Regard)
- Optical and radar signal limits

The fields for a bistatic radar mirror the receiver ones leaving out optical and radar signal limits.

2.1.3 Reference population

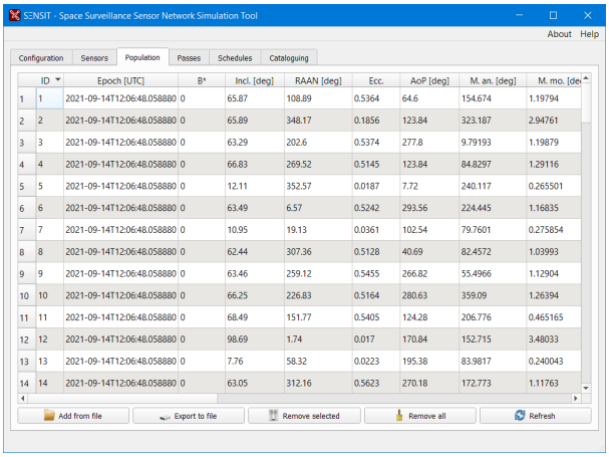

Fig. 6 SΞNSIT GUI: population tab.

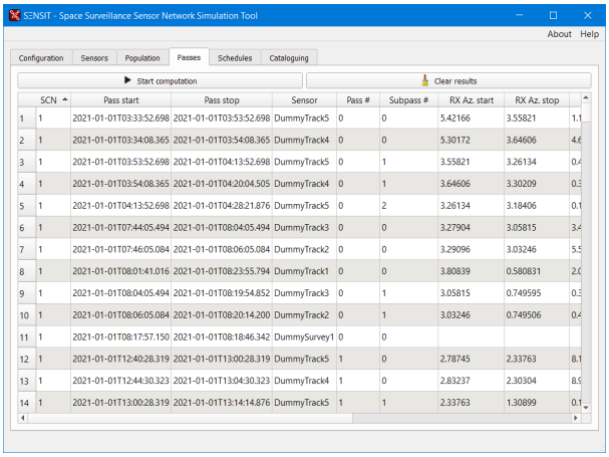

Fig. 7 SΞNSIT GUI: passes tab.

The reference population of space objects loads via GUI (*Fig. 6*) or with a command line script, accepting the following formats:

- Two-Line Elements in a text file,
- Cartesian states in CSV format,
- List of Satellite Catalog Numbers (NORAD IDs), for which TLEs automatically download from Space-Track.org,
- ESA MASTER population file (*.pop),
- SGP4 elements (in CSV or JSON format).

Regardless of the file format, they all convert to SGP4 elements and are saved in an SQLite database.

An initial state covariance can optionally be set in various reference frames [6].

Moreover, loading the RCS and the object's intrinsic brightness enables the radar signal loss and the optical magnitude computation for the link budget equation.

2.2 Pass computation

This section deals with the visible passages computation of the reference population as seen in (*Fig. 7*). Relying on a bisection algorithm, it leverages several optimizations:

- Caching stations and objects states.
- The core of the algorithm is written in C++
- Parallel computing

The algorithm, summarized in *Fig. 8*, splits the propagation time window into segments.

It checks the observability condition, producing a preliminary estimation of the passages using a bisection-like procedure to speed up the computation. Successively, the bisection finetunes the start and stop epochs of the passes.

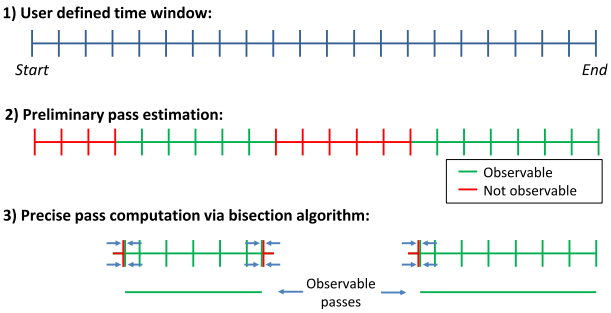

Fig. 8 SΞNSIT algorithm for passes computation.

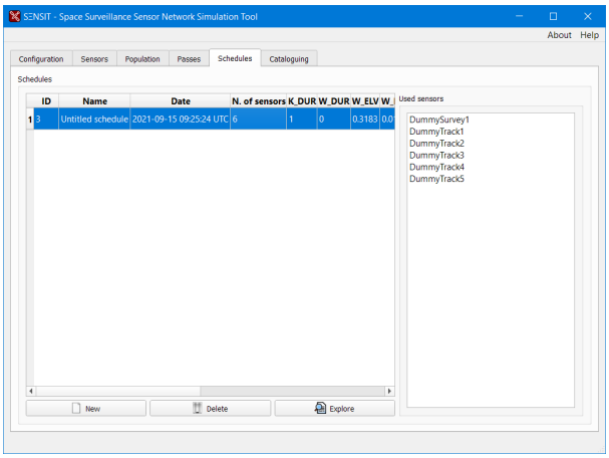

Fig. 9 ΞNSIT GUI: schedules tab.

2.3 Scheduling of observations

Multiple objects may simultaneously transit over the ground stations. Thus, optimal scheduling for tracking sensors is crucial for efficient and effective orbit population monitoring, as outlined in *Fig. 9*.

2.3.1 Scheduling rules

SΞNSIT features scheduling rules presented hereafter:

- For survey sensors, all transits are observable.
- Tracking sensors chase an object if:
  - the sensor is not already busy following another object,
  - the sensor can perform the slew maneuver in time,
  - a certain time has passed since the last observation of the same object.

When two passes violate scheduling rules, they are in conflict. Provided an ordered list of *n* observable passes, the relative overlaps describe an *(n, n)* Boolean matrix **M** (see *Fig. 10*). If the ith passage conflicts with the jth one, the matrix will have *true* in both cell *(i, j)* and *(j, i)*. Conversely, for not overlaying passes, the corresponding positions are *false*. The conflict matrix has the following properties:

- **M** is symmetric by definition,
- All diagonal elements are all *false*, given a pass never conflicts with itself,
- It is generally sparse (i.e., most of its values are *false*) and stored in memory-efficient representations.

Even the schedule is a Boolean vector associated to the list of passages. If the ith passage is detectable, the ith item of the vector will be *true*, otherwise *false*.

Fig. 11 shows an example of overlapping passages. Assuming that a sensor can pinpoint only one object at a time, pass #1 is superimposed with pass #0 and pass #2, while pass #3 has no conflicts.

M =	0	1	2	3	s =
0	F	T	F	F	T
1	T	F	T	F	F
2	F	T	F	F	T
3	F	F	F	F	T

Fig. 10 Boolean representation of a conflict matrix (M) and a schedule (s).

This representation computes the number of conflicts *c* present in a schedule with *Eq. 1*. Boolean values *true* and *false* implicitly convert to 1 and 0:

$$c = \frac{\mathbf{s}^{\mathbf{T}}\mathbf{M}\,\mathbf{s}}{2} \qquad (1)$$

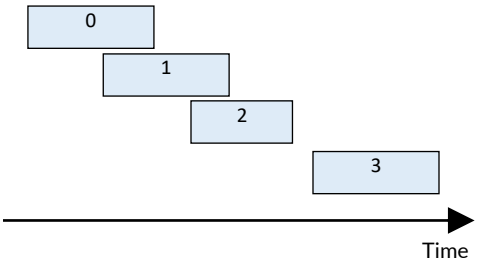

Fig. 11 Example of a timeline of passes with overlaps.

2.3.2 Fitness value

The number of all possible schedules is 2^n, and the ones meeting the condition $c = 0$ are feasible. To programmatically choose the optimal ones, a *fitness value* f is assigned to all of them.

The overall *fitness* is the summation of the scores linked to the scheduled passes (*Eq. 2*).

$$f_i = {v_d}^{k_d} \sum_n w_n v_n \text{, with n} \in (d, e, \rho, L, m) \qquad (2)$$

$$F = \boldsymbol{s} \cdot \boldsymbol{f} \qquad (3)$$

The exponent k_d and the weights w_n are adjustable under the parameter importance for v_n, which represents:

- v_d: Pass duration
- v_e: Max. elevation as seen by the receiver
- v_ρ: Reciprocal of the min. distance from the receiving station
- v_L: Reciprocal of the min. radar signal loss
- v_m: Reciprocal of the min. optical magnitude

2.3.3 Genetic algorithm

The schedule solves as an optimization problem (maximize $\boldsymbol{s} \cdot \boldsymbol{f}$) subject to a constraint ($\mathbf{s}^{\mathbf{T}}\mathbf{M}\,\mathbf{s} = 0$).

The Boolean representation is tailor-made for binary genetic algorithms realized with the DEAP library [7]. Specifically, it starts by creating *popsize* schedules, instantiated as follows:

1. Start from an empty schedule.
2. Add all the passages featuring no conflicts (e.g., on survey sensors).
3. Add one passage at a time, ensuring no overlaps.
4. Stop when conflicts appear in the schedule.

Afterwards, for some iterations (*generations*), the following steps are executed:

5. **Selection**: shortlist the best schedule among three randomly chosen ones for *popsize* times (to keep the size of the population constant).

6. **Cross-over**: randomly selects two schedules and swaps a segment of them. It leads to the addition and removal of passes; other conflicting passes are deleted.
7. **Mutation**: randomly inserts or removes passages.
8. **Refill**: randomly adds non-conflicting passages to a schedule as far as possible (similarly to steps 3 and 4).

The user can tune various optimization parameters through the GUI (*Fig. 12*), such as:

- The weights associated to f.
- The population size.
- The number of generations.
- The probabilities cross-over, mutation and refill.
- The initial seed for the random number generator.

The GA scheduler tab shows real-time statistics.

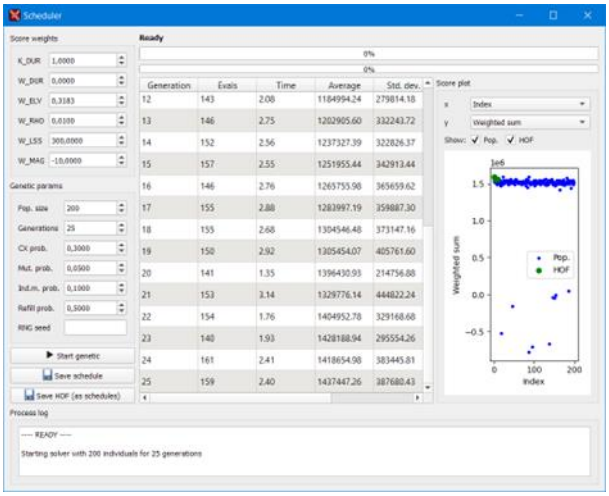

Fig. 12 SΞNSIT GUI: genetic scheduler tab.

A list of the best ten schedules, the *hall of fame*, is kept in memory either as a list of passes (*Fig. 13*) or as a timeline (*Fig. 14*). The latter exhibits the chosen bits of passes (in red) and the not-chosen ones (in white). The overall timeline for each sensor is colored blue at the top of the graph.

The obtained schedule exports also in CSV format for further analyses.

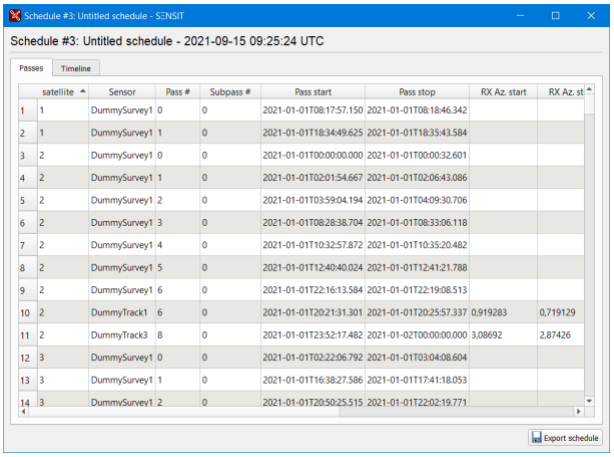

Fig. 13 SΞNSIT GUI: list of scheduled passes.

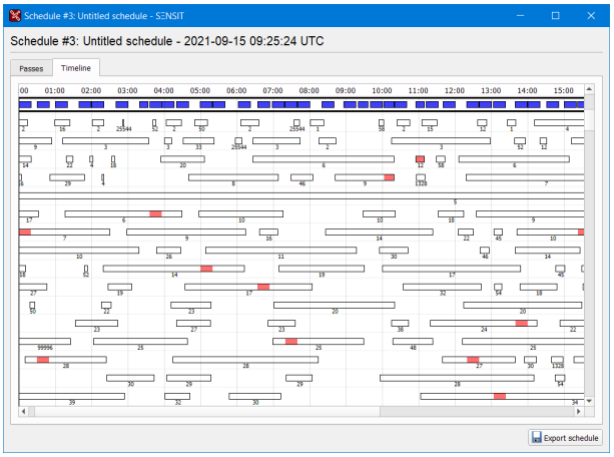

Fig. 14 SΞNSIT GUI: timeline showing scheduled portions of passes (in red).

2.4. Catalogue build-up

The third module harnesses one of the schedules for network catalogue build-up and maintenance (*Fig. 15*). The user has the option to include just a subset of sensors to conveniently execute different simulations and compare the results.

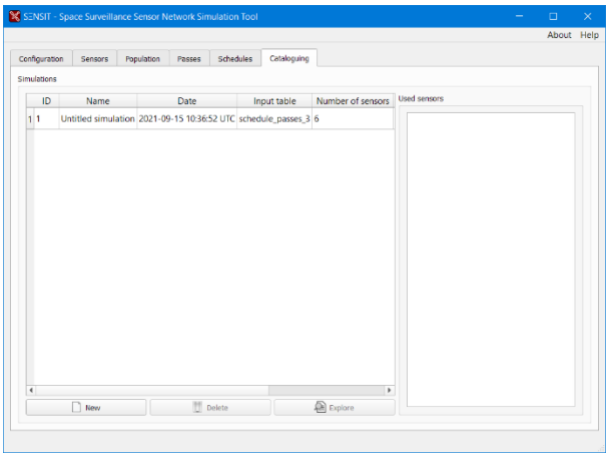

Fig. 15 SΞNSIT GUI: cataloguing tab.

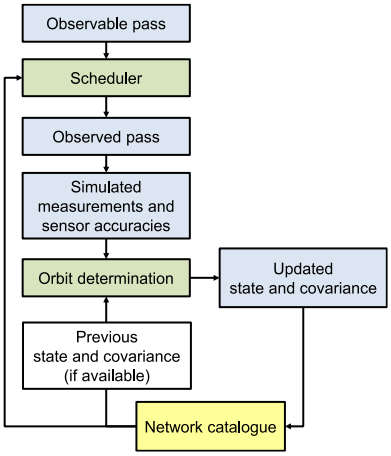

Fig. 16 Scheme of the catalogue build-up simulation.

This module performs a cataloguing simulation by processing the computed passages with conflict-free schedules or with all passes.. The user can only consider a subset of all the available sensors to run the simulation. In this way, it is possible to perform different simulations and compare them. *Fig. 16* shows the steps of the catalogue build-up process. Once all possible passages over the sensor have been computed, a basic scheduler ensures that tracking sensors can only observe objects in the catalogues with a sufficiently accurate position knowledge. If the trace of the 3x3 position covariance sub-matrix, expressed in QSW frame, is lower than a user-defined threshold, then the object still belongs to the sensor network.

New or ”lost” objects are catalogued with an Initial Orbit Determination (IOD) performed by a survey sensor.

The covariance propagates using a Keplerian state transition matrix. Whenever a catalogued object has a valid covariance, the basic scheduler decides if the object is visible or not. In such case, a Refined Orbit Determination (ROD) updates the covariance of the object, as illustrated in *Fig. 17.*

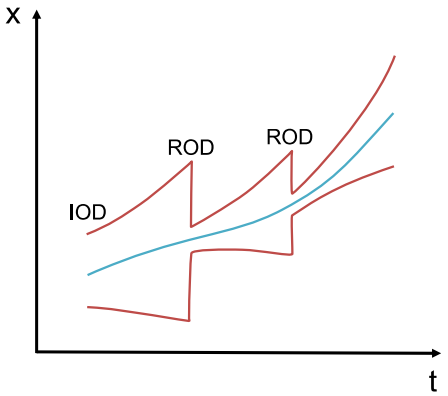

Fig. 17 Covariance evolution during catalogue maintenance: IOD and ROD process.

Propagating and updating the covariances is done by The Orbit Determination sub-module with two different pipelines. The first method, Estimated covariance OD, outputs the final covariance by knowing the Jacobian measurements matrix with respect to the object initial state (using the Keplerian assumption) and known initial covariance matrix. Hence, being the formulation analytical, this method may lack accuracy, but it is quicker. The second one is a Non-Linear Least Squares OD. It performs the orbit determination iteratively with synthetically generated data. Every time an IOD/ROD is executed, the updated covariance matrices are stored in a local database.

3 Results

The results of the previous steps are saved into a local SQLite database. This allows a code optimization without computing the same quantity multiple times. Eventually, GUI displays all relevant results to the user. The main visualization tables available are explained below.

3.1. Passes

A window portrays the list of all computed passages showing for each of them, the NORAD ID, the sensor name, initial and final epochs, orbit determination type and updated covariance. On top, a redundancy matrix (*Fig. 18*) indicates the ratio between the observable passages by any couple of sensors.

	DUMMY1	DUMMY2	DUMMY3	DUMMY4
DUMMY1	100%	97.8%	100%	62.4%
DUMMY2	100%	100%	100%	62.6%
DUMMY3	100%	97.8%	100%	62.4%
DUMMY4	100%	98.3%	100%	100%

Fig. 18 Redundancy matrix.

3.1.1. Population

The observable population is illustrated by means of a scatter plot where the axes represent two arbitrary orbital elements of an object. The color indicates the number of passes (*Fig. 19*). It is interactive: besides changing the quantities on the axis, it is also possible to select either all sensors or a specific one: the color scale changes accordingly.

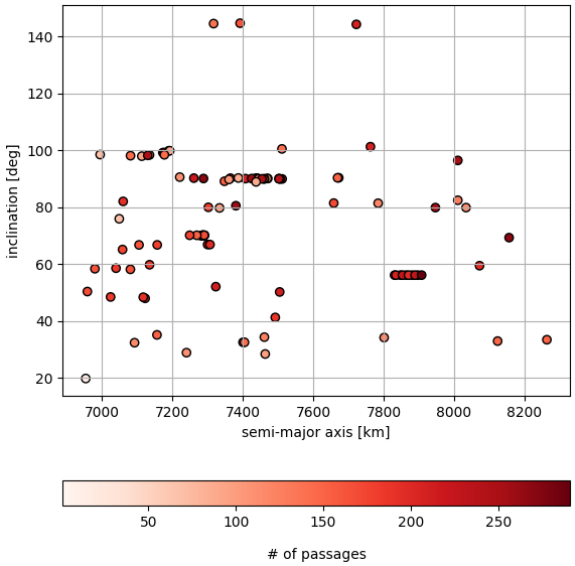

Fig. 19 Catalogue population plot.

3.1.2. Coverage and covariance evolution

The catalogue evolution over time is in terms of percentage concerning the reference population, both for the total population and for a specific subset population defined by the user.

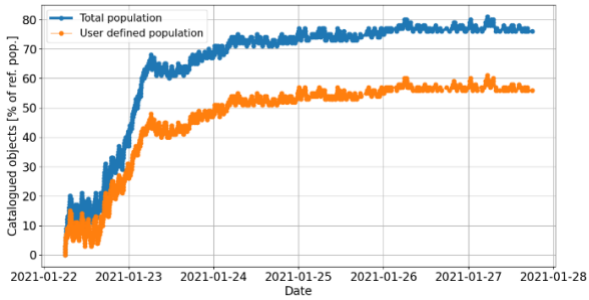

Fig. 20 Network coverage plot.

In addition, another window is for checking the evolution of the covariance in time of any object (see *Fig. 21*). This allows us to understand when a specific target is in the catalogue of a sensor network. The plot also displays, with different colors, when IOD/ROD occurs according to the computed schedule.

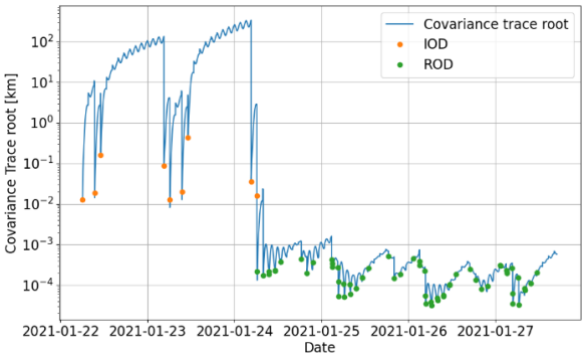

Fig. 21 Covariance evolution for a specific target.

3.1.3. Maximum re-observation time

The re-observation time is the time elapsing between two consecutive observable passes. The lower it is, the more the network updates the covariance estimates in the catalogue. This information is reported in terms of cumulative distribution. For example, *Fig. 22* shows that 87% of the population can be observed at least twice a day.

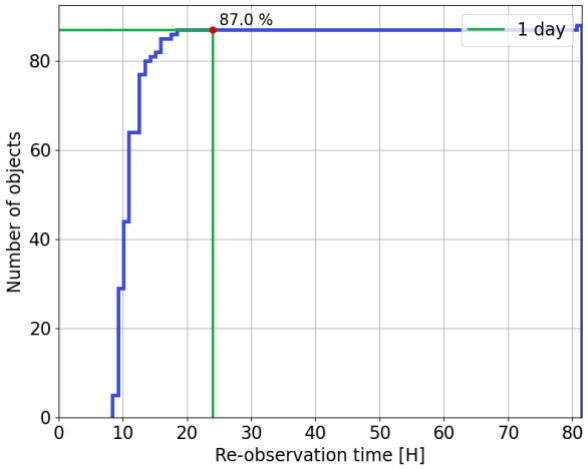

Fig. 22 Cumulative Distribution of the maximum re-observation time.

3.2. *Computational time*

This section reports the computation time needed to perform different simulations with varying input parameters. The analysis has been conducted on a PC featuring 16 GB of RAM, a 3700x AMD processor with 8 physical and 16 logical cores.

The following scenarios are considered:

- Population: 10, 100, 1000.
- Number of involved in the network: 1, 5, 10
- Simulation time frame: 3 days

The results for the computation of the observable passes are in *Tab. 1* and *Fig. 23*. It can be clearly seen that the main responsible is the number of objects, while the amount of sensors has little impact.

Tab. 1 Time required (s) by pass computation.

	1 sensor	**5 sensors**	**10 sensors**
10 obj.	3	4	4
100 obj.	21	23	25
1000 obj.	204	223	244

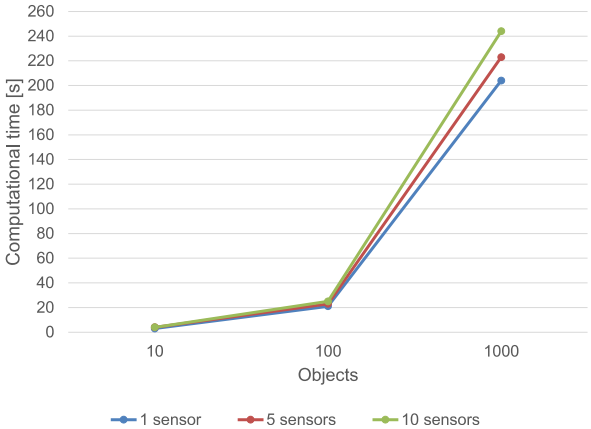

Fig. 23 Computational time required for passages computation.

Tab. 2 and *Tab. 3* (same results are also graphically illustrated in *Fig. 24* and *Fig. 25*) instead report the time needed to perform catalog build-up and maintenance using, respectively, the Estimated Covariance OD and Least Square OD method. In this case, b the number of objects and sensors influence the time needed. As expected, the first method ensures limited computational burden, especially for a large population.

Tab. 2 Time required (s) by catalogue build-up using estimated OD..

	1 sensor	**5 sensors**	**10 sensors**
10 obj.	1	1	1
100 obj.	3	3	3
1000 obj.	19	30	83

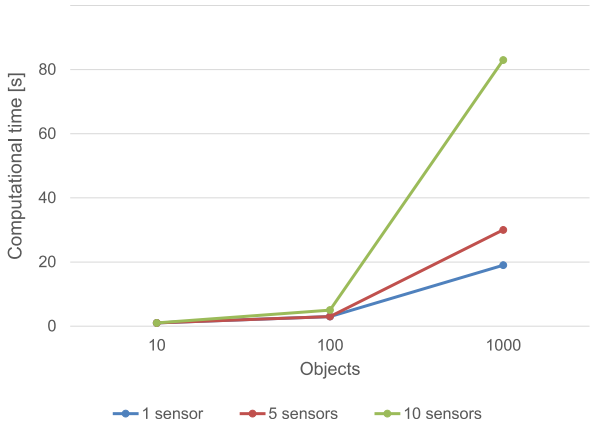

Fig. 24 Time required by catalogue build-up using Estimated Covariance OD.

Tab. 3 Time required (s) by catalogue build-up using NLS OD.

	1 sensor	**5 sensors**	**10 sensors**
10 obj.	1	1	1
100 obj.	4	6	12
1000 obj.	40	63	191

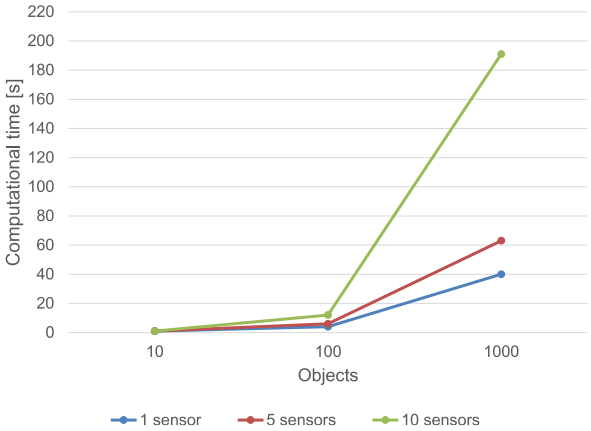

Fig. 25 Time required for catalogue build-up using NLS OD method.

4. Conclusions

SΞNSIT is a software tool with a friendly GUI intended to compute the coverage and catalogue build-up and maintenance of space objects by a ground-based sensor network. An optimal scheduler generates conflict-free schedules using a genetic algorithm. In this framework, the envisioned improvements regard the optimization function and the fine-tuning of the scheduler parameters. Concerning the modeling aspect of the sensors, the main missing feature is introducing accurate or customizable radar beam patterns for passes computation. Additionally, it would be advisable to consider tracking errors quantify their effect (this could be important for telescopes and laser sensors), adverse weather conditions for signal loss, maintenance downtime or missed detections. Regarding the catalogue build-up and maintenance simulation, an add-on could be providing only range or doppler information in some radar sensors. A further improvement is to consider the effect of the expected Signal-to-Noise ratio on the measurement covariance instead of using fixed values for each sensor.

Acknowledgements

The results of this project have been supported by the agreement of the Italian Space Agency and the National Institute for Astrophysics on Space Debris (Detriti Spaziali – Supporto alle attività IADC e SST 2019-2021, n. 2020-6-HH.0).

References

[1] "Protection of space assets," [Online]. Available: https://vision.esa.int/protection-of-space-assets/

[2] V. Morand, C. Yanez and J. C. Dolado Perez, "BAS3E: A framework to Conceive, Design, and Validate Present and Future SST Architectures," in *First International Orbital Debris Conference*, 2019.

[3] N. O. Gómez, I. A. Gómez and S. A. Vildarraz, "Architectural description of the Spanish Space Surveillance and Tracking System," 2017.

[4] Purpura, Giovanni & De Vittori, Andrea & Cipollone, Riccardo & Di Lizia, Pierluigi & Massari, Mauro & Colombo, Camilla & di Cecco, Alessandra & Salotti, Luca. (2021). SENSIT: a software suite for observation scheduling and performance assessment of SST sensor networks. *72nd International Astronautical Congress (IAC 2021)*.

[5] C. Acton, "Ancillary Data Services of NASA's Navigation and Ancillary Information Facility," *Planetary and Space Science*, vol. 44, no. 1, pp. 65-70, 1996.

[6] CCSDS Secretariat, National Aeronautics and Space Administration, *Navigation Data - Definitions and Conventions*, Washington, DC, 2019.

[7] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau and C. Gagné, "DEAP: Evolutionary Algorithms Made Easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171-2175, jul 2012.