Multi-Agent Systems
Project Presentation

Professors: Andrea Omicini, Roberta Calegari
Student: Andrea Perna
University of Bologna

Project Motivations

Project Goals

Multi-Agent System

Design a modular Multi-Agent System (MAS) for Search and Rescue missions (SAR) in Python;

Engineer a clear separation between agents and environment;

Create a distributed network of robotic agents modelled with a strongly connected graph;

Distributed Optimization

Implement distributed aggregative optimization algorithm to provide distributed coordination;

Agents aim to fulfill both local and global targets, by reaching consensus on global information;

Ensure efficient navigation with obstacle avoidance;

Beliefs-Desires-Intentions

Leverage the BDI framework to govern robots' behaviors;

Design mentalistic agents, inspired by agency's strong definition;

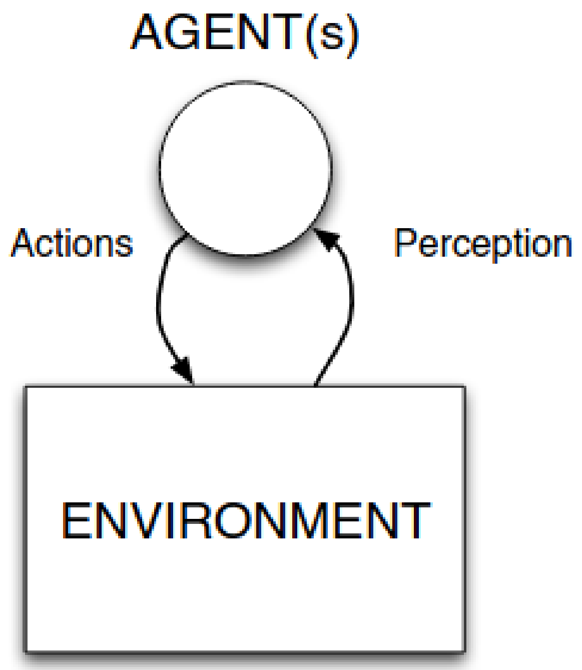
Enable real-time decision-making, to cope with dynamic and unpredictable environments;

System Architecture

The system's modularity separates environment and agents in the MAS, using agent-oriented programming to give robots the autonomy to achieve their goals.

- **Environment Class**
  - **House:** Simulated layout with defined rooms, obstacles, and openings.
  - **Survivors**: Individuals needing rescue, placed in the environment.
  - **Ambulance**: Vehicle transporting survivors to safety;

- **Agent Class**
  - Agents behave using the **Belief-Desire-Intention** (BDI) framework.
  - Use **distributed aggregative optimization** algorithm to make optimal decisions aiming to achieve both local and global tasks;
  - Navigate the environment while performing **obstacle avoidance**;



Agent(s) Model: Russel and Norvig, 2022

Consensus Theory

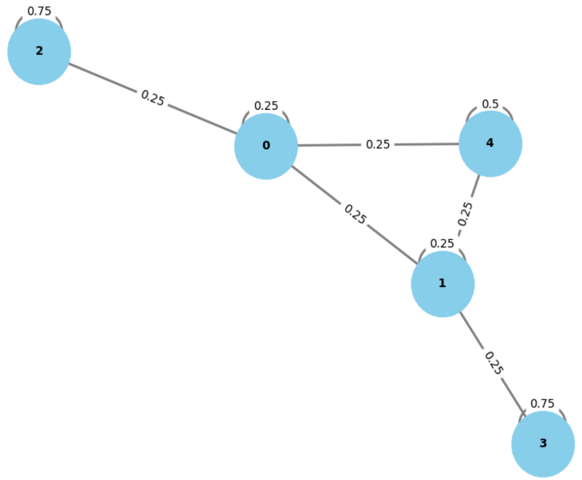
Consensus Theory provides mathematical tools to reach **agreement** among distributed agents on a quantity.

- **Mathematical Model**: agents i in graph G iteratively update their states via local communication with out-neighbours $j \in \mathcal{N}_i^{out}$ (agents receiving data from i):

$$z_i^{k+1} = \sum_{j \in \mathcal{N}_i^{out}} a_{ij} z_j^k$$

- **Adjacency Matrix**: weights $a_{ij} \geq 0$ represent communication strength between i and j in directed graph G. For a fixed topology, it forms an *LTI* system $z^{k+1} = Az^k$, where $A \in \mathbb{R}^{NxN}$ is the square adjacency matrix of the communication network;

- **Average Consensus**: agents reach *consensus* on the average of the initial estimates if A is doubly stochastic (i.e., $A\mathbf{1} = \mathbf{1}$ and $\mathbf{1}^T A = \mathbf{1}^T$) and G is strongly connected and aperiodic. This ensures *simple stability* of the *LTI* system;



Graph Connectivity, 5 Agents Network

Optimization Theory

In MAS, optimization is the key to determine the most efficient agent trajectories from available alternatives:

- **Gradient Descent**: iteratively updates decision variables z^k $\forall k \in \{1, \ldots, max\_iters\}$ in the steepest descent direction $d^k = -\nabla \ell(z^k)$ with step-size $\alpha^k > 0$, minimizing the cost $\ell \in \mathbb{R}^d \longrightarrow \mathbb{R}$ to solve $\min_{z \in \mathbb{R}^d} \ell(z)$:

$$z^{k+1} = z^k - \alpha^k \nabla \ell(z^k)$$

- **Convexity**: any local minimum of $\ell(z)$ is also a global minimum, i.e., z_i^* is an optimal global solution if:

$$\ell(\theta z_1 + (1-\theta) z_2) \leq \theta \ell(z_1) + (1-\theta)\ell(z_2)\ \forall z_1, z_2 \in \mathbb{R}^d, \theta \in [0,1];$$

- **Convergence**: convergence of z_i^k to z_i^* $\forall i$ is guaranteed when ℓ is convex and α^k is sufficiently small to ensure a stable decrease in the cost, i.e., $\ell(z^{k+1}) < \ell(z^k)$ $\forall k$ to solutions z^* such that $\nabla \ell(z^*) = 0$ and $\nabla^2 \ell(z^*) \geq 0$;

Gradient Tracking Algorithm

- **Distributed Optimization**: Through local updates, agents aim to track a global estimate of the gradient while optimizing local decision estimates ${z_i}^k$ of ${z_i}^*$ via minimization of aggregated cost function $\ell(z) = \sum_{i=1}^{N} \ell_i(zi)$;

- **Dynamic Average Consensus**: each agent maintains a local estimate ${s_i}^k$ (tracker) of a time-varying signal $\overline{r^k}$, (e.g., gradient and formation's barycenter), through an averaging system with local innovation to track changes;

- **Gradient Tracking Algorithm**: combines consensus and optimization to track total gradient $\frac{1}{N}\sum_{i=1}^{N} \nabla\ell_i\left({z_i}^k\right)$:

$$ {z_i}^{k+1} = \sum_{j \in {\mathcal{N}_i}^{out}} a_{ij}{z_j}^k - \alpha^k {s_i}^k, \qquad {z_i}^0 \epsilon \mathbb{R}^d \ \forall i \in \{1, \dots, N\} $$

$$ {s_i}^{k+1} = \sum_{j \in {\mathcal{N}_i}^{out}} a_{ij}{s_j}^k + \left(\nabla\ell_i\left({z_i}^{k+1}\right) - \nabla\ell_i\left({z_i}^k\right)\right), \qquad {s_i}^0 = \nabla\ell({z_i}^0) \ \forall i \in \{1, \dots, N\}, $$

- **Convergence**: the algorithm converges if A is doubly stochastic, graph G is connected and each cost function l_i is strongly convex with a Lipschitz gradient, ensuring $\lim_{k\to\infty} \left\|{z_i}^k - {z_i}^*\right\| = 0$ for all agents i;

Distributed Aggregative Optimization

Aggregative Optimization involves N **robots** optimizing positions $z_i \in \mathbb{R}^2, i \in 1, \dots, N$ by minimizing local cost functions $\ell_{i(z_i, \sigma(z))}$, which rely on global time-varying information estimated through Gradient Tracking algorithm:

- Agents $i \in [1, N]$ independently optimize **local decision variables** $z_i{}^k$, i.e., N different solutions $z_i{}^*$ are computed;

$$\min_{z_1,\dots,z_N} \sum_{i=1}^{N} \ell_i\left(z_i, \sigma(z)\right), \quad \text{with} \quad \sigma(z) = \frac{1}{N}\sum_{i=1}^{N} \phi_i\left(z_i\right)$$

- The problem involves **two global terms**: the cumulative gradient $\sum_{j=1}^{N} \nabla_2 \ell_j(zjk, \sigma(z^k))$ and the barycenter position $\sigma(z)$;

$$z_i^{k+1} = z_i^k - \alpha\left(\nabla_1 \ell_i\left(z_i^k, s_i^k\right) + \nabla\phi_i\left(z_i^k\right) v_i^k\right)$$

$$s_i^{k+1} = \sum_{j \in \mathcal{N}_i} a_{ij} s_j^k + \phi_i\left(z_i^{k+1}\right) - \phi_i\left(z_i^k\right)$$

$$v_i^{k+1} = \sum_{j \in \mathcal{N}_i} a_{ij} v_j^k + \nabla_2 \ell_i\left(z_i^{k+1}, s_i^{k+1}\right) - \nabla_2 \ell_i\left(z_i^k, s_i^k\right)$$

- Each agent i is aware only of $\emptyset_i$ and ℓ_i, ad maintains **estimate** $z_i{}^k$ of the optimal solution $z_i{}^*$ and **trackers** $s_i{}^k$ and $v_i{}^k$ of global time-varying signals $\sigma(z^k)$ and $\sum_{j=1}^{N} \nabla_2 \ell_j\left(z_j{}^k, \sigma(z^k)\right)$ to track;

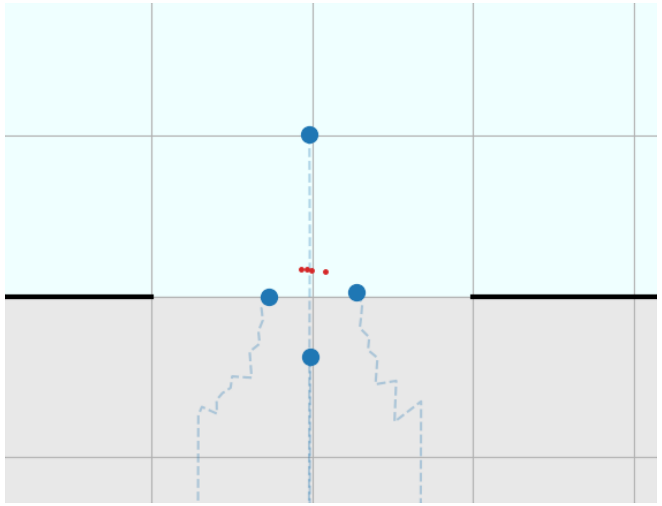
$$z_i^0 \in \mathbb{R}^{n_i}, \; s_i^0 = \phi_i\left(z_i^0\right), \text{ and } v_i^0 = \nabla_2 \ell_i\left(z_i^0, s_i^0\right)$$

Obstacle Avoidance

- Integrated into agents' costs to navigate the environment while safely **avoiding collisions** and maintaining **tight formation**;
- The system assigns **potential functions** to obstacle points, generating **repulsive forces** with gain K as agents approach;
- They represent **artefacts** for the robotic agents in the MAS;
- The repulsive force, derived from the potential's gradient, directs agents along the **steepest descent** path of the function U_{rep};
- **Local minima** may arise in complex cluttered environments, potentially trapping agents in sub-optimal positions;

$$U_{\text{rep},i}(q) = \begin{cases} \frac{1}{2} K_{r_i} \left(\frac{1}{d_i(q)} - \frac{1}{q^*} \right)^2 & \text{if } d_i(q) < q^* \\ 0 & \text{otherwise} \end{cases}$$

$$\nabla U_{\text{rep},i}(q) = \begin{cases} K_{r_i} \left(\frac{1}{q^*} - \frac{1}{d_i(q)} \right) \frac{\nabla d_i(q)}{d_i(q)^2} & \text{if } d_i(q) \le q^* \\ 0 & \text{if } d_i(q) > q^* \end{cases}$$



Example of Room Walls Avoidance

Cost Function

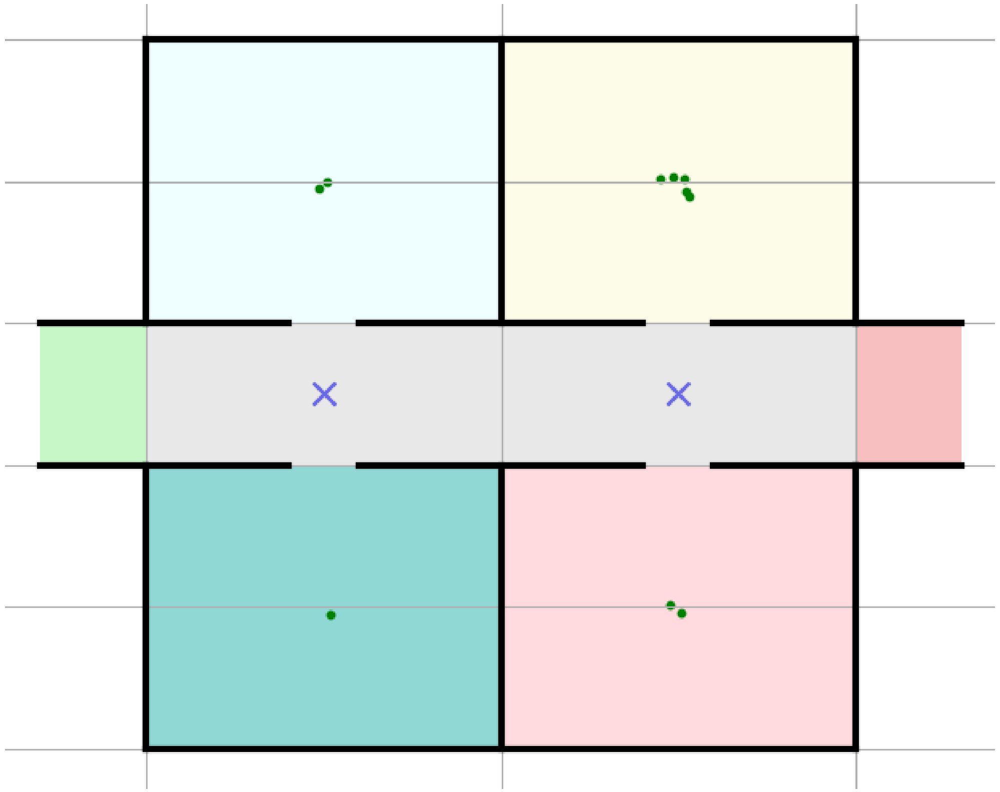
The algorithm's effectiveness hinges on the cost function convex structure, dictating **agents' behaviors**:

$$\ell_i\big(z_i, \sigma(z)\big) = \gamma_{rlt}\|z_i - ri\|^2 + \gamma_{bar}\|\sigma(z) - b\|^2 + \gamma_{agg}\|z_i - \sigma(z)\|^2 + obst\_pot$$

- **Local Target Attraction:** $\gamma_{rlt}\|z_i - ri\|^2$, based on the distance of each robot i to its local target $r_i \in \mathbb{R}^2$;
- **Barycenter Goal:** $\gamma_{bar}\|\sigma(z) - b\|^2$, based on the distance between the barycenter $\sigma(z)$ and global target $b \in \mathbb{R}^2$;
- **Barycenter Attraction:** $\gamma_{agg}\|z_i - \sigma(z)\|^2$, based on the distance between the robot i and the barycenter $\sigma(z)$;
- **Potential Function:** $obst\_pot$, based on the distance between the robot i and each obstacle point;

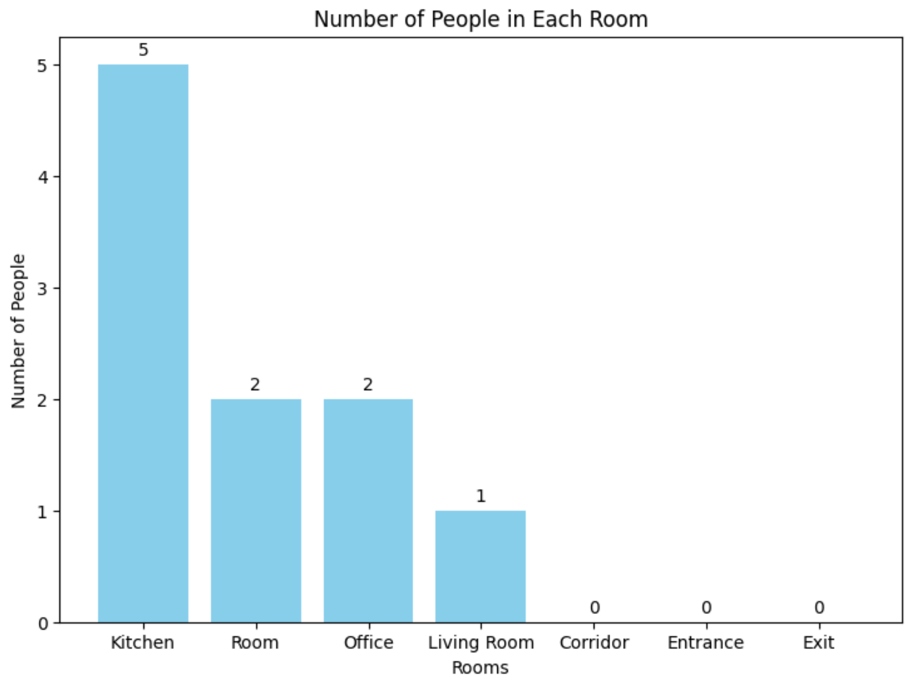
Environment

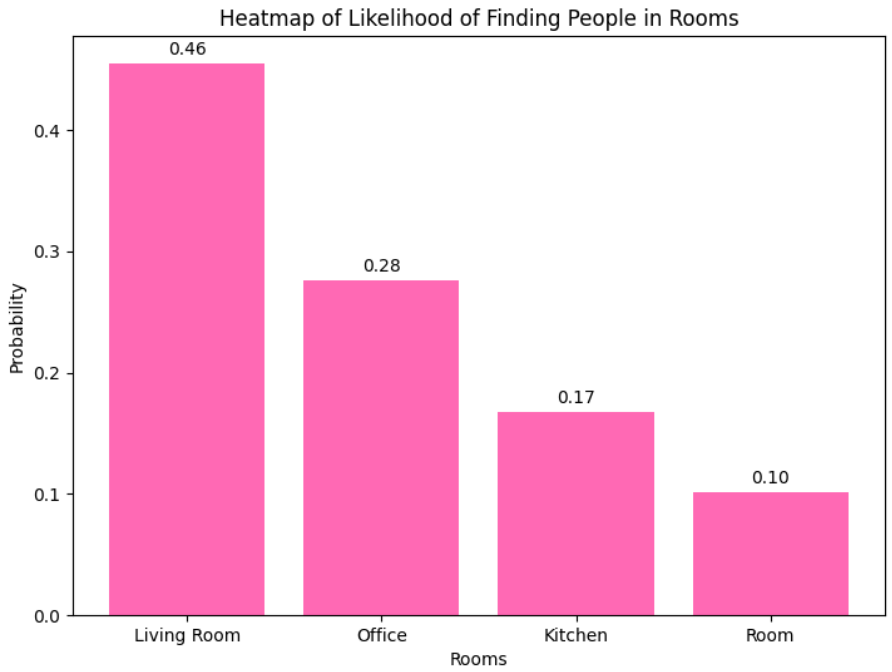
- **Ambulance and Survivors**: FSMs (*artefacts*) used to simulate people in rooms and the ambulance;
- **Obstacles and Openings**: walls, obstacle points and openings define the navigable space for the agents;
- **Midpoints**: used for optimized navigation in plans, guiding agents efficiently between rooms;
- **Pragmatical Actions**: situated agents may change the state of the environment with pragmatical actions;
- **Visualization**: the *plot_house()* function provides a color-coded graphical representation of the environment;



Rooms Initialization

At the beginning of the simulation, survivors are randomly placed in rooms according to a **uniform distribution**. The exact number of people for each room is determined by the probabilities, and it is not known by the agents.





Survivors

Finite State Machine

Survivor are managed by an FSM with states: *Steady*, *Escorted*, *Saved*, *Transported* and *Healed*;

Randomly initialized in rooms according to house’s heatmap;

Saved survivors use P-controllers to reach the *Idle* ambulance;

Groups Creation

Agents gather *Steady* survivors using the *create_escort_group()* function;

max_survivors_escort caps the number of people escorted at once;

If survivors are left due to group limits, agents prioritize returning to their room over exploring new areas;

Rescue

Survivors transition to *Escorted* when grouped and moved to *Exit* with agents, where they are *Saved*;

Saved survivors are delivered to the hospital upon ambulance arrival, entering *Transported* state;

The mission continues until all survivors are *Healed* in hospital;

Ambulance

Finite State Machine

Operate in three states: *Idle*, *Departing* and *Returning*;

Ambulance movements are synchronized with agents and survivors' activities;

Arrival and Departure

Departing once a group of escorted survivors boards the ambulance;

After a while, another ambulance is dispatched to the exit in *Returning* state to collect the next group;

Hospital

Transport survivors to the hospital, marking them *Healed* upon arrival;

If all survivors have been healed, then the ambulance must stay *Idle*, to prevent unnecessary dispatches;

Beliefs, Desires, Intentions

Beliefs

Represent agent's knowledge about the MAS environment;

Initialized from a YAML file;

Include room locations, obstacle points and survivor statuses;

Dynamically updated through *perceive_environment()* and *brf()*;

Desires

Define the agent's objectives based on the current context, such as escorting survivors, visiting rooms or reach exit;

Function *generate_options()* generates feasible desires on updated beliefs;

Multiple, possibly conflicting, desires can co-exist inside agents' minds;

Intentions

Represent the plans and actions chosen to fulfill agents' desires;

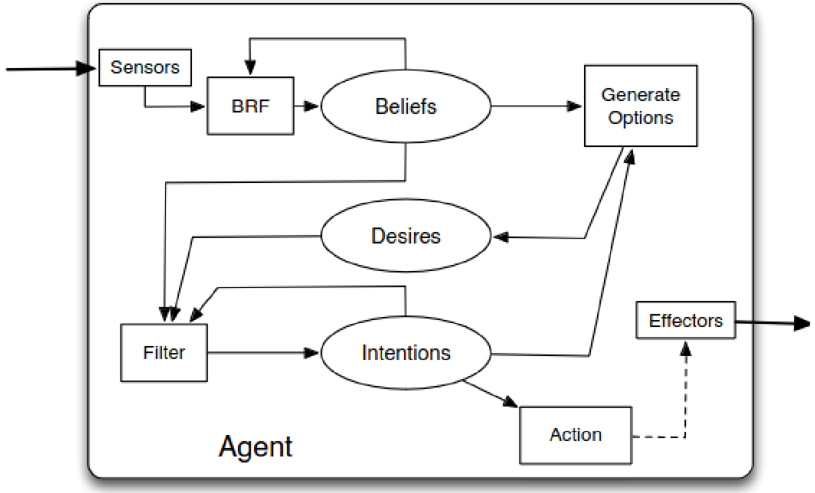
Function *filter_options()* selects and prioritizes one single desire;

Implemented by *create_plan()* through waypoints, obstacles and escort groups' generation;

BDI Control Loop

The control loop of BDI agents represents their “**minds**”, continuously updating beliefs, generating desires, filtering options into intentions and executing plans to achieve their goals. Such a model follows agency’s strong definition.

```
while True:
    perceptions = perceive_environment()
    beliefs = belief_revision_function(perceptions, beliefs)
    desires = generate_options(beliefs)
    intentions = filter_options(beliefs, desires)
    plan, obstacles = create_plan(beliefs, intentions)
    execute_plan(plan, obstacles, beliefs, desires, intentions)
    if check_termination(): break
```

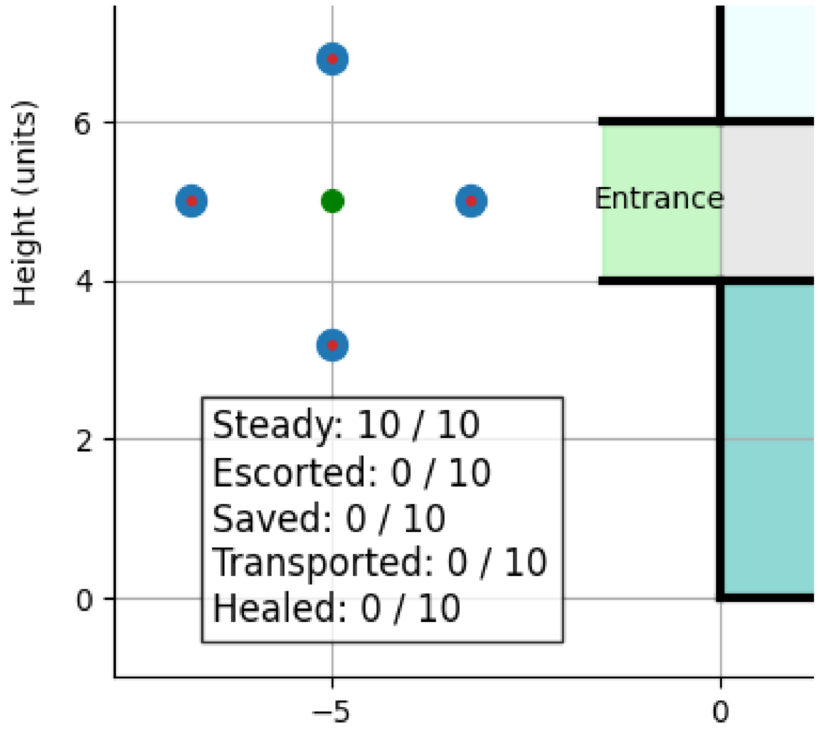


Basic Architecture of a BDI Agent, Wooldridge 2009

Trajectories Initialization

The *trajectory_initialization()* method sets up the trajectories for the MAS' components across iterations, space and dimensions.

- **Trajectory Initialization**: prepares trajectories for formation targets (R, b), agent positions (Z), barycenter and gradient trackers (S, V);
- **Circular Formation**: utilize *generate_circular_positions()* to set the robots' initial formation radially around entrance point with N agents;
- **Targets Definition**: similarly to circular formation, set local target coordinates R radially scattered around global coordinates b;
- **Cost and Gradient Monitoring**: initialize arrays to track the cost function F and gradients $(grad\_z, grad\_s)$, essential for movements;



Robots Trajectory Initialization, 4 Agents

Techniques and Procedures

Entering the House

Agents initially move to the local targets placed in *Entrance* room;

Visit rooms with highest likelihood of finding survivors, continuously updating beliefs and desires;

A plan is created to fulfill the intention, using rooms' waypoints, obstacles and formation's gains;

Escorting Survivors

Agents form a feasible escort group once they detect survivors in rooms;

Safely navigate to the exit adjusting their movements to avoid obstacles while maintaining a tight formation;

Perform communication and pragmatical actions with neighbors and the environment in the MAS;

Exiting the House

Once all known survivors have left the rooms, set intention to *Exit*;

Agents may enter *Standby* mode at the exit if all survivors are *Saved*, but not yet fully *Healed*;

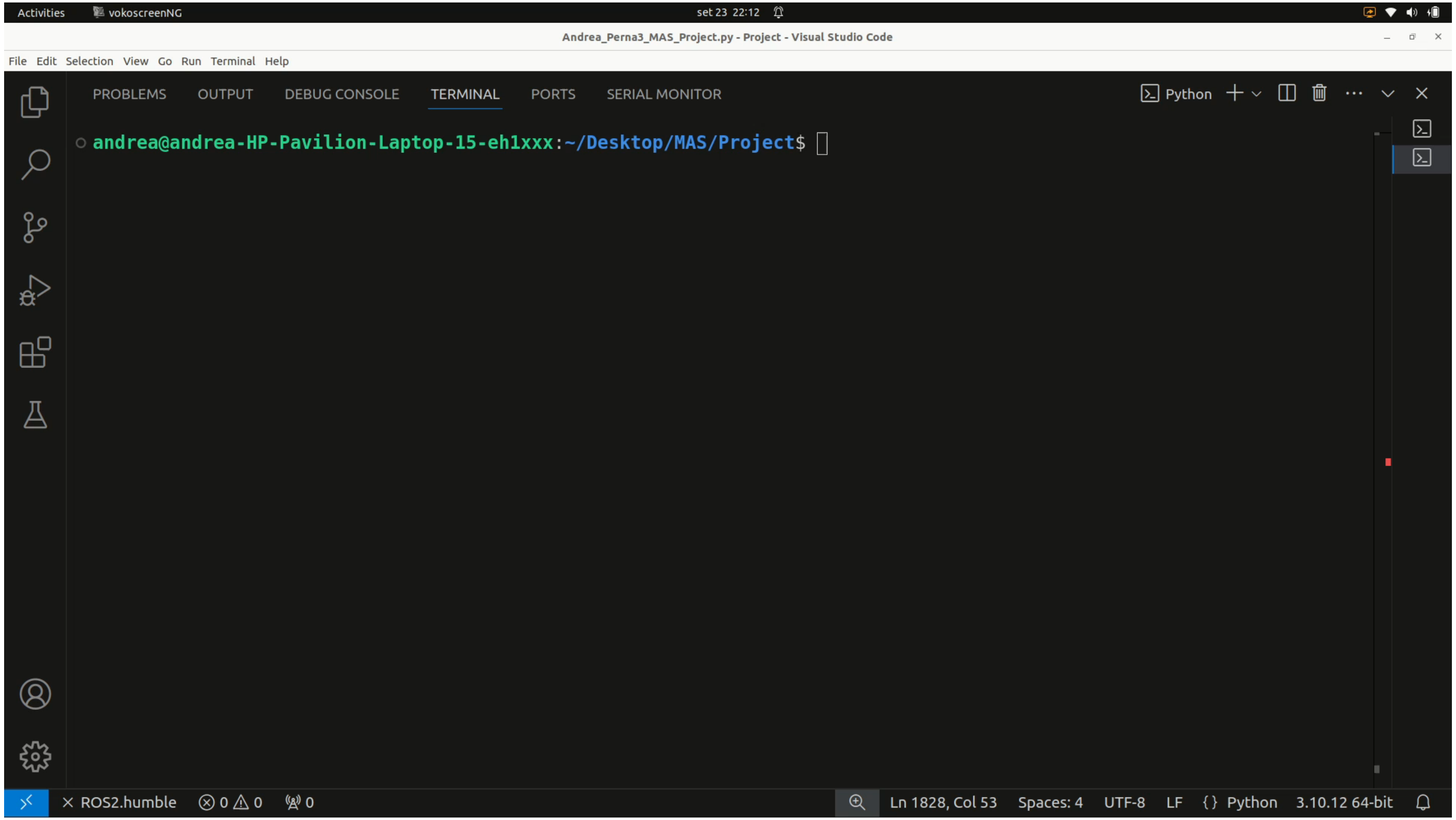
Agents in the MAS are driven by the *goal* of ensuring all survivors are safely *Escorted* and *Healed*;

Visualization

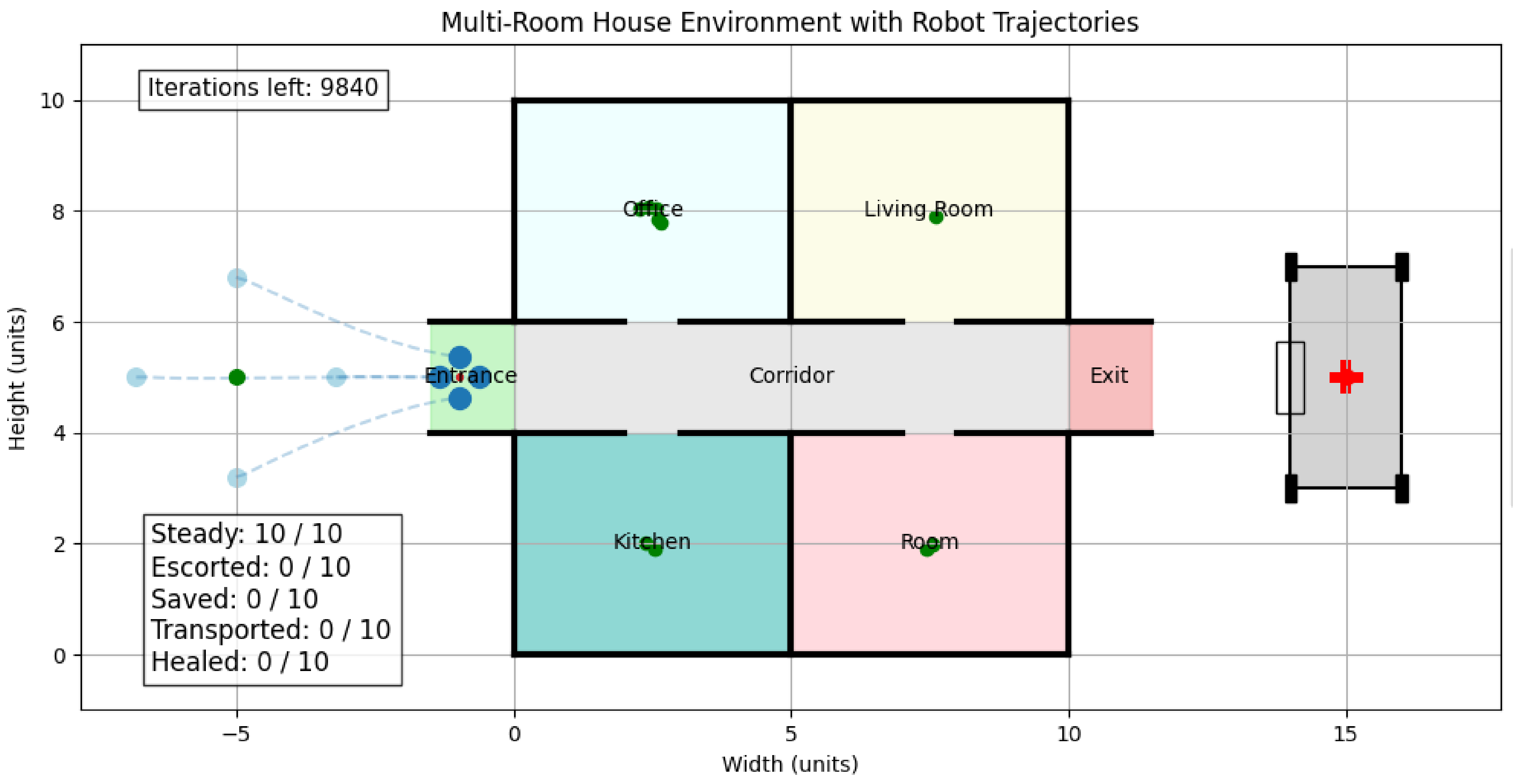
The *SAR_animation()* function animates the trajectories of robots, survivors and the ambulance across iterations.

- **Plot Limits:** based on *view_type* parameter in YAML file, they can be either static or dynamic;
- **House Layout**: house is shown in background with walls, corridors, obstacles in a color-coded fashion;
- **Robots Trajectories**: robots' initial positions and trajectories are shown to illustrate movement strategies;
- **Survivors Trajectories**: people's positions are dynamically updated, showing progress from detection to rescue;
- **Ambulance Trajectory**: the ambulance, with details like blinking lights, visually reflects its current state;
- **Dynamic Annotations**: real-time updates on key metrics, like iterations left and survivor status;

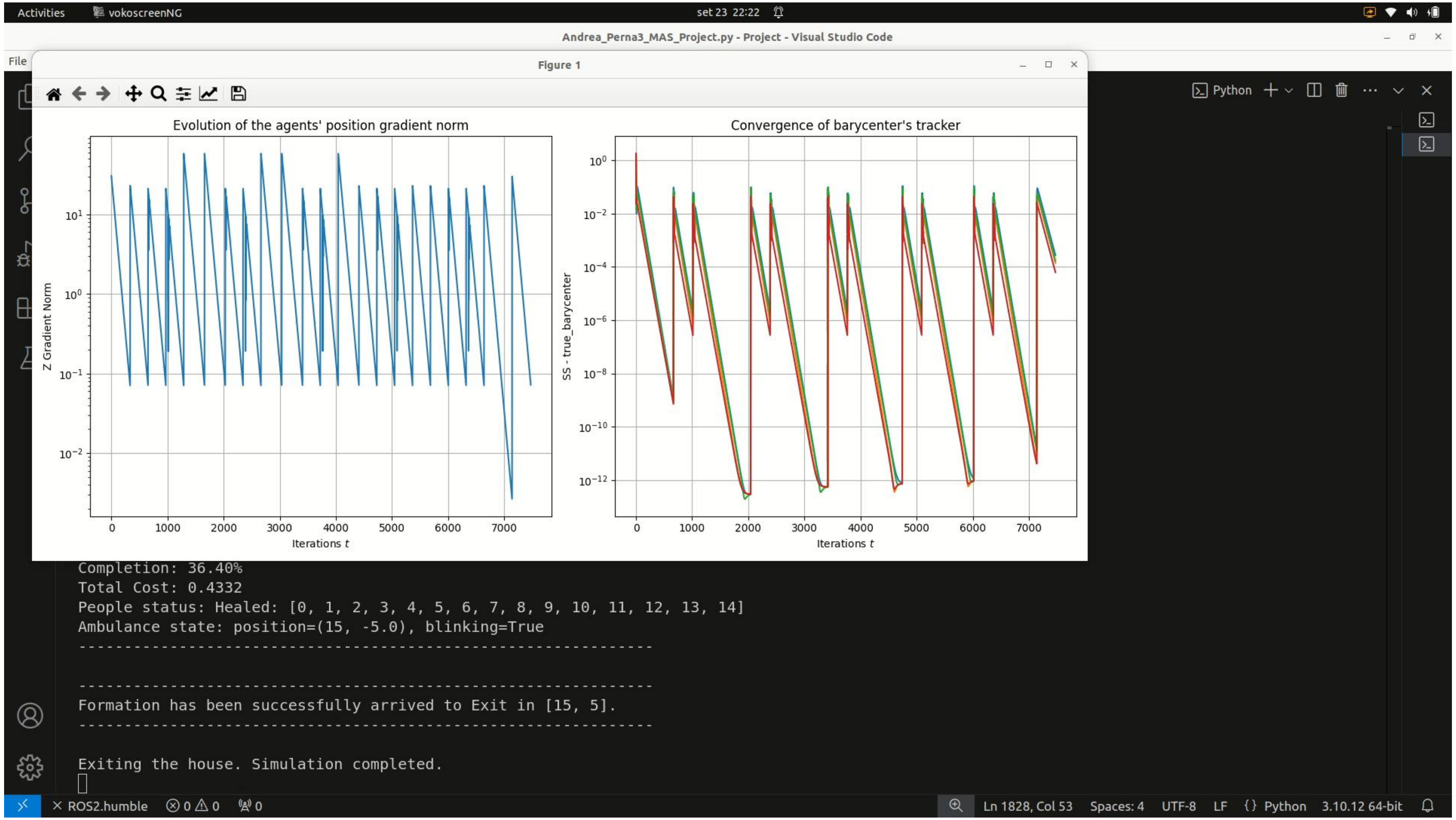
SAR Animation – Computations



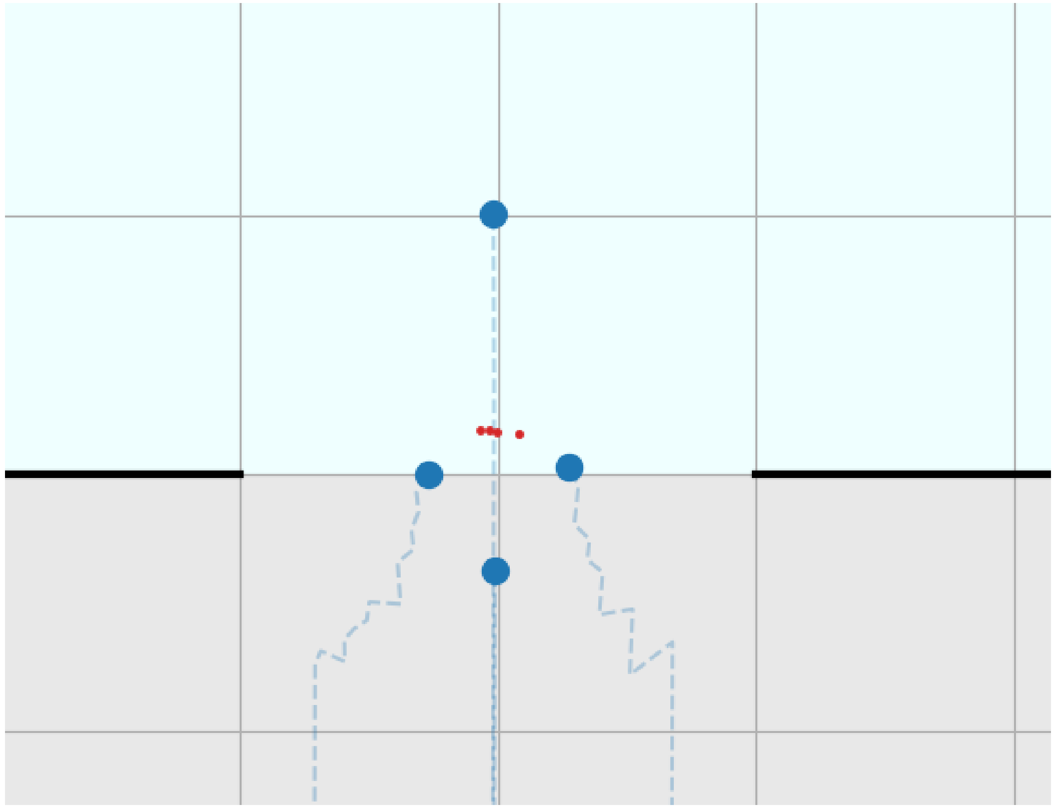
SAR Animation – Static View

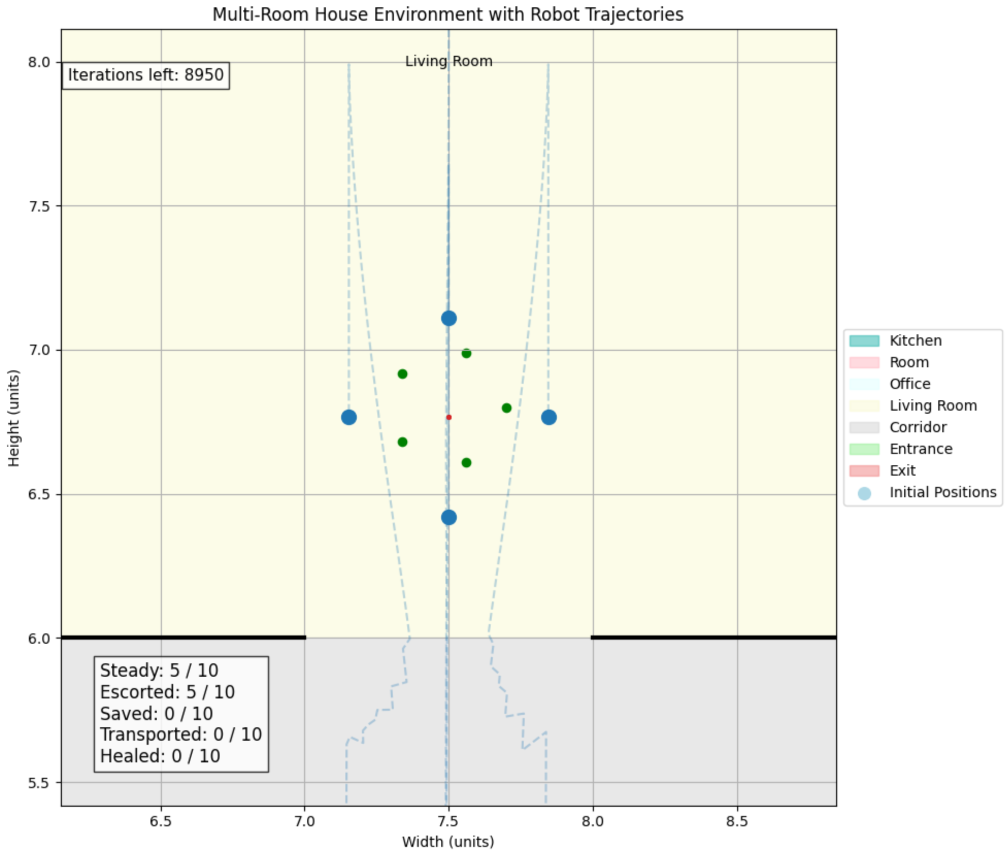


SAR Animation – Static View

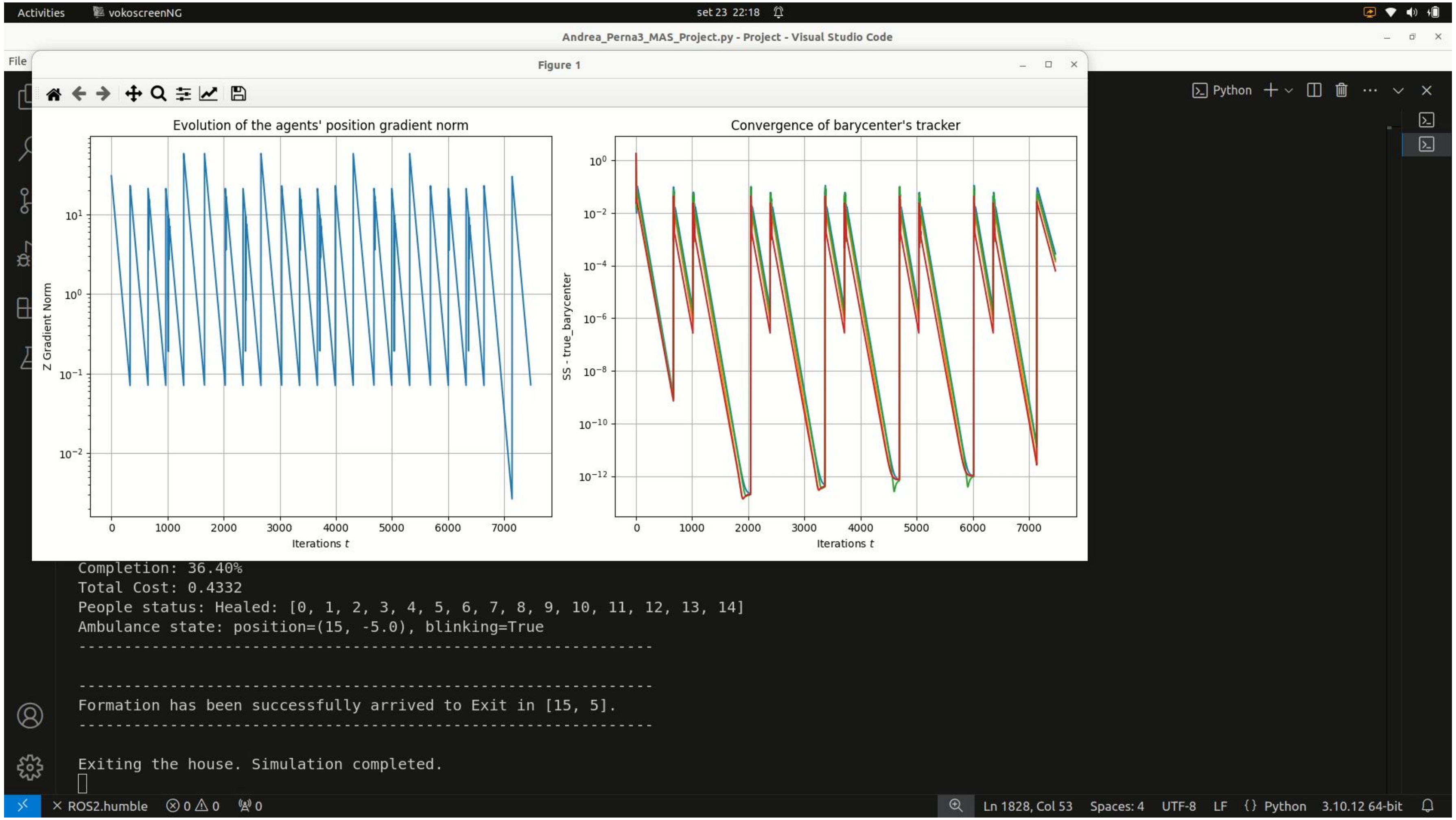


SAR Animation – Dynamic View

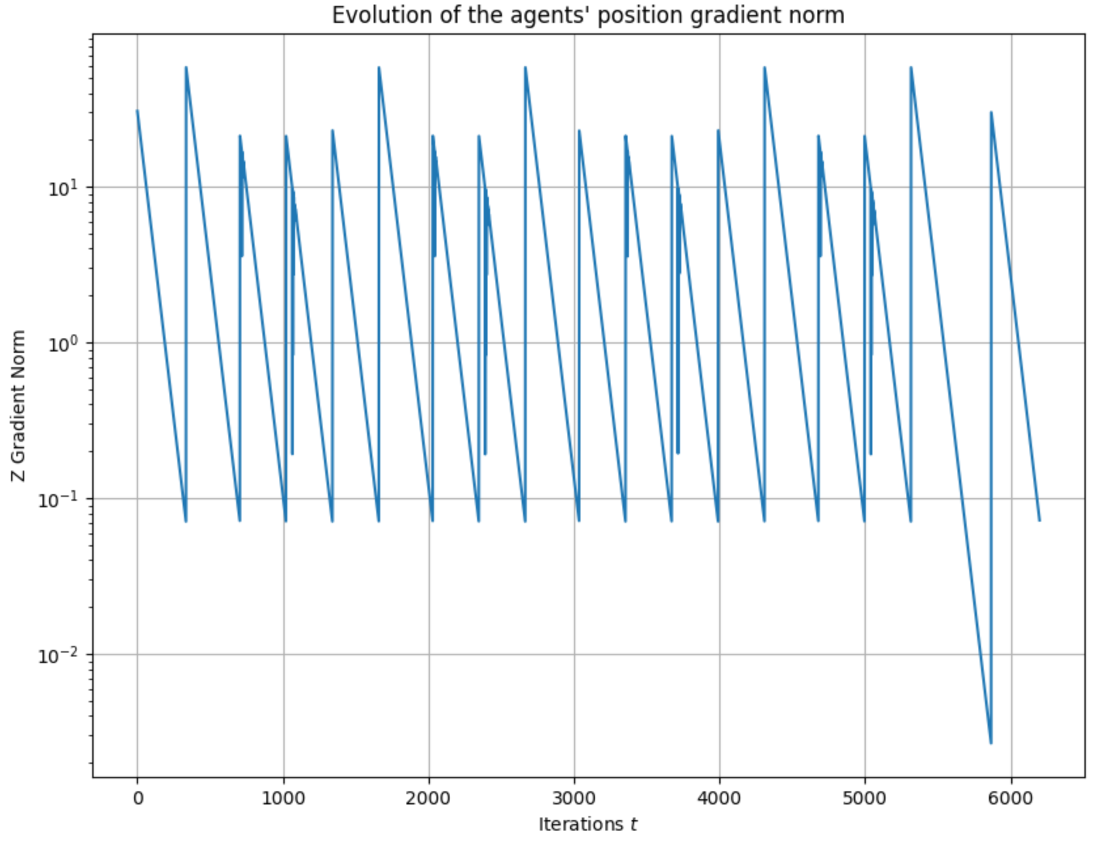


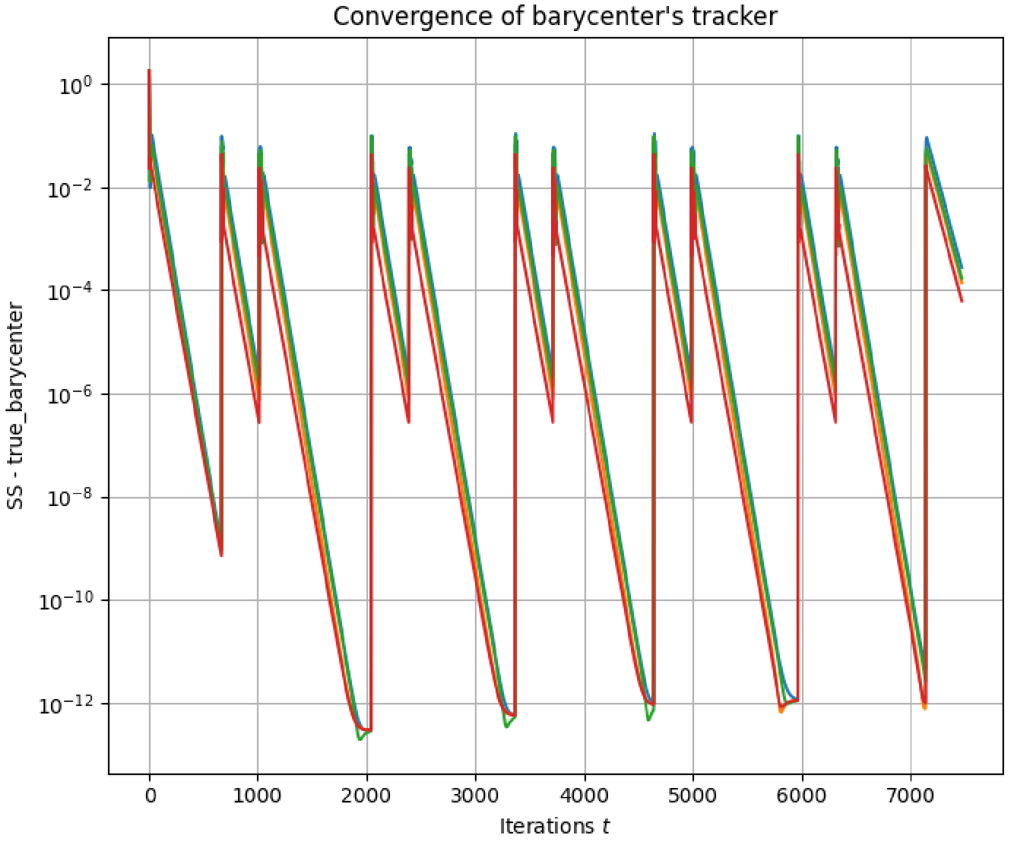


SAR Animation – Dynamic View



Costs and Gradients

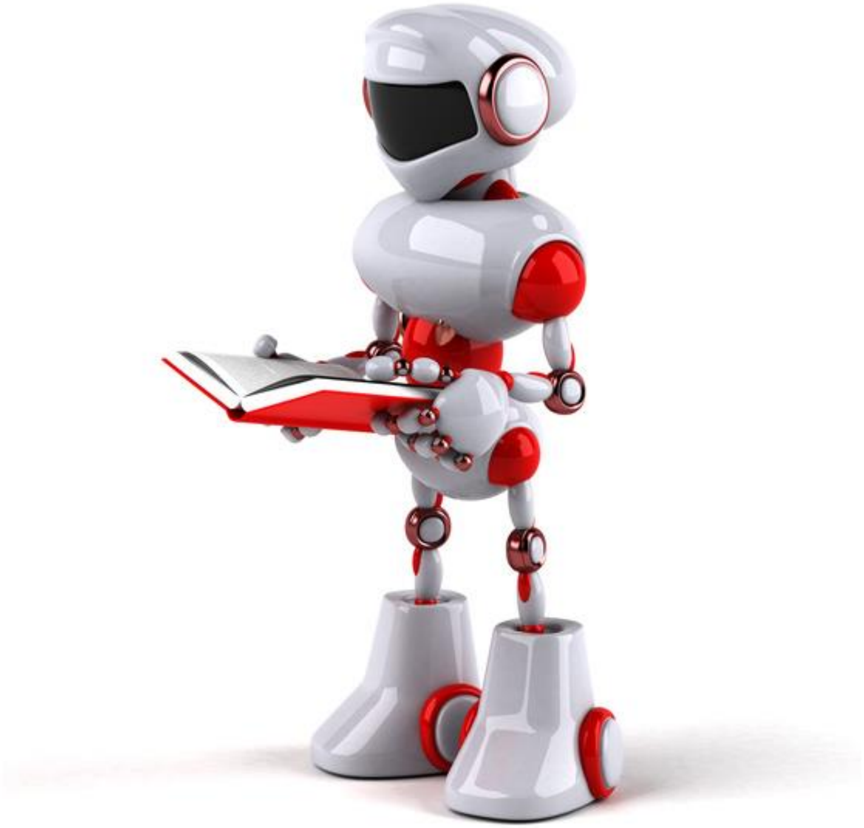




Summary

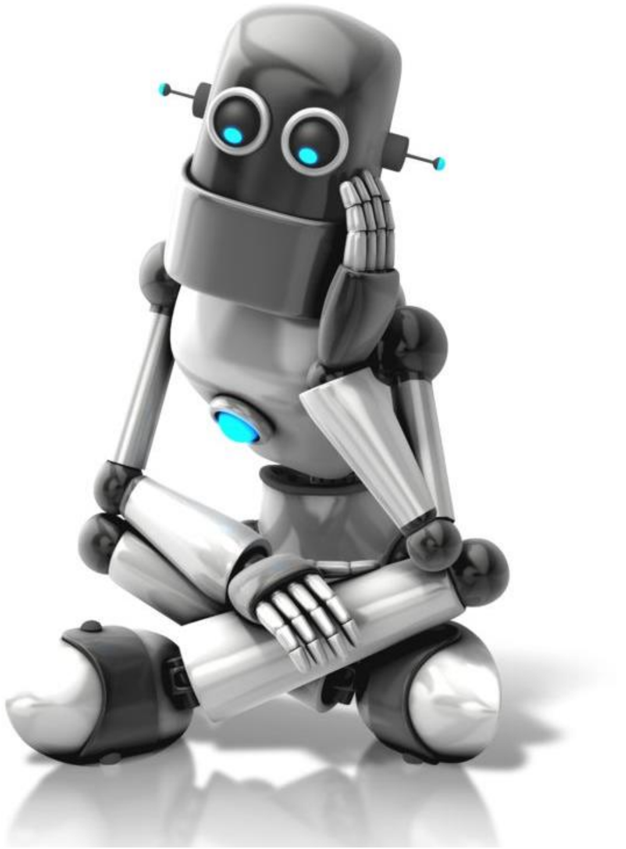
To summarize, the **main characteristics** of the developed Multi-Agent System project are:

- Modularity and Extensibility;
- Distributed Aggregative Optimization;
- BDI Mentalistic Agents;
- Obstacle Avoidance via Potential Functions;
- Real-time Consensus Coordination;
- Visualization of SAR Simulations;



Future Enhancements

- Implementation in **ROS2** distributed framework;
- Simultaneous Localization and Mapping (**SLAM**)
- Multi-Agent Reinforcement Learning (**MARL**);
- Tuple-based Coordination (**TuCSoN**);
- **Battery** management and **energy** constraints;
- Decentralized **task allocation** to handle resources;
- Keep track of **health statuses** of survivors;
- **Computer Vision** features to enhance robots' perception.



Thanks for your attention!

Keep The Gradient

- Bruno Siciliano -