AI-powered decision-making beta software release

Deliverable D3.2

Distribution level: Public

Version: 1.0

Date of delivery: 31/03/2026

AI4REALNET has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101119527, and from the Swiss State Secretariat for Education, Research and Innovation.

Documentation of contributions for WP3

This slide deck explains the code developed within WP3 of the AI4REALNET project.

AI4REALNET focuses on AI-based solutions for critical infrastructure systems, including electricity, railway, and air traffic management, where AI supports and augments human decision-making.

The project develops advanced decision-making methods using supervised and reinforcement learning, ensuring trustworthiness, safety, resilience, and security. It integrates human-in-the-loop approaches and autonomous AI systems operating under human supervision and embedded safety rules.

The framework is validated through six industry-driven use cases across three network infrastructures, addressing challenges such as decarbonization, digitalization, and operational resilience

Overview

- **T 3.1 Human-in-the-Loop Decision Making**
  - RL Agent Uncertainty Quantification
  - Conformal Prediction Framework
  - Epistemic Uncertainty and Predicting RL Agent Failure Probability
  - Failure Probability Framework
  - Agent As A Service (A3S) & Trace RL
- **T3.2 – Multi-Objective Decision-Making**
- **T3.3 – Interactive AI to Augment Decision-Making**
  - Inverse Reinforcement Learning with Risk-sensitive behavior
  - Shaping AI Behavior to Operational 'Best Practices'
  - Interactive Preference Learning in ATM Sectorization
  - Human Learning Support
- **T3.4 – Integrated Autonomous AI-driven Decision Systems**

Where to find more information in the Documentation

D3.2 – Software Deliverable	Chapters in the D3.1 Documentation
T3.1 Human-in-the-Loop Decision Making	**Ch.1–2:** Human-centered AI & Joint Decision-Making Challenges
Epistemic Uncertainty and Predicting RL Agent Failure Probability	**Ch.4:** Conformal Risk Assessment & Prediction Intervals
Agent-as-a-Service (A3S) & TraceRL	**Ch.3:** Agent-as-a-Service & TraceRL GUI
T3.2 Multi-Objective Decision-Making	**Ch.5:** Multi-Objective Decision-Making with AI
T3.3 Interactive AI / Preference Learning	**Ch.6:** Interactive AI to Augment Decision-Making
T3.4 Integrated Autonomous AI-driven Decision System	**Ch.7:** Autonomous AI Driven-Decision Systems

Link to the Github-Code-Repository:

https://github.com/AI4REALNET

T3.1 – Human-in-the-Loop Decision Making

Leader: EnliteAI

Contributors: INESC TEC, UvA, SBB, FLAT

Overview T3.1

Human-in-the-Loop Decision Making

- RL Agent Uncertainty Quantification
- Conformal Prediction Framework
- Epistemic Uncertainty and Predicting RL Agent
- Failure Probability Framework
- Agent As A Service (A3S) & Trace RL

RL Agent Uncertainty Quantification

Contributors: INESC TEC

Context

Motivation

AI-based assistants may produce inefficient recommendations for resolving power grid congestion. It is necessary to capture future grid outcomes such as line loadings, and its associated uncertainty.

Proposed methodology

Development of a framework:

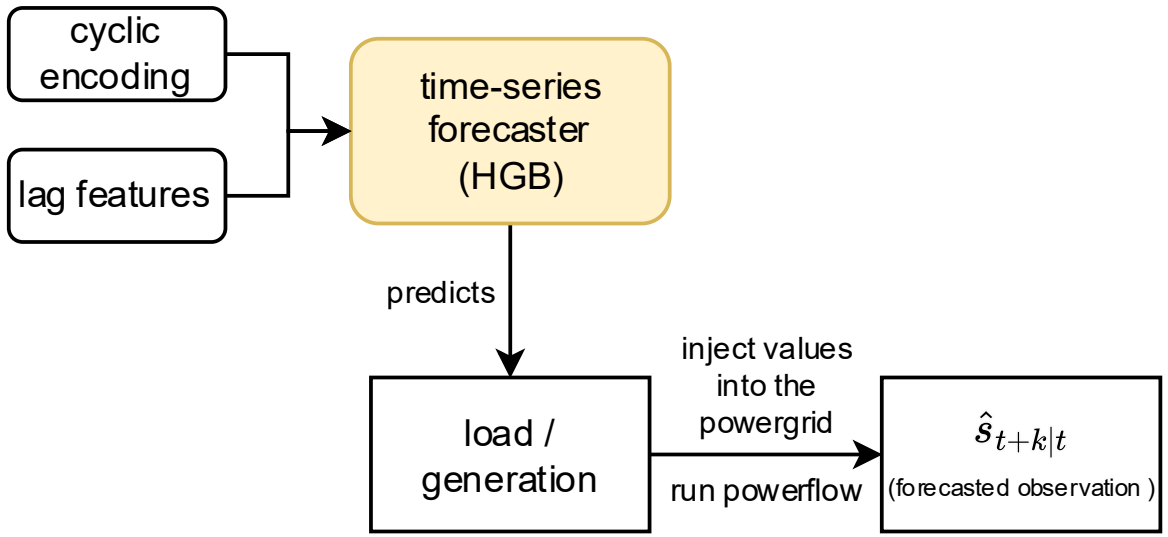
- Obtains line loading forecast values by simulating an RL agent action after injecting the grid with future load/generation values.
- Provides intervals of uncertainty around point forecasts of the forecasted line loadings using Conformal Prediction.

GitHub

Conformal Prediction Framework

- Obtaining the forecasted state
- Obtaining the forecasted line loadings
- Framework pipeline
- Model types
- Repository structure overview
- Starting a new experiment
- Output structure

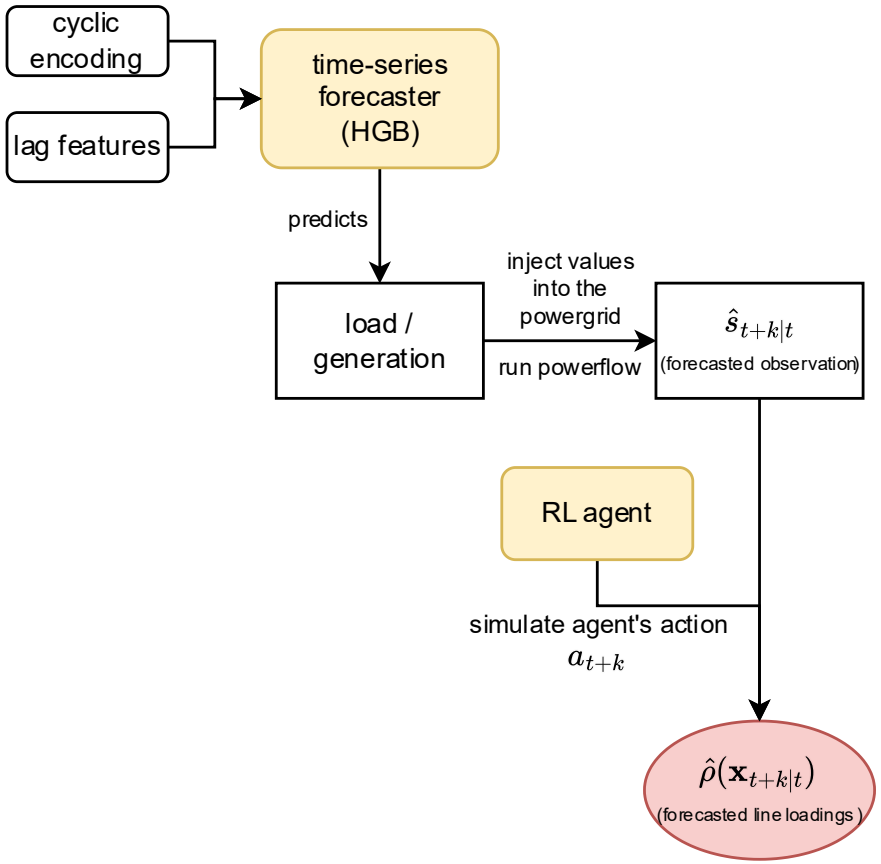
Obtaining the forecasted states



- A time-series forecaster trained as a multi-output regressor is used to predict active and reactive power injections using historical observations and temporal variables (lagged features) for all loads and generators over a 1-hour time horizon (timesteps $t+1, \ldots, t+12$).

- These forecasted values are injected into the grid, and a powerflow is ran to obtain the forecasted state of the grid.

Obtaining the forecasted line loadings



The Conformal Prediction framework aims to provide intervals of uncertainty around the line loadings forecasts.

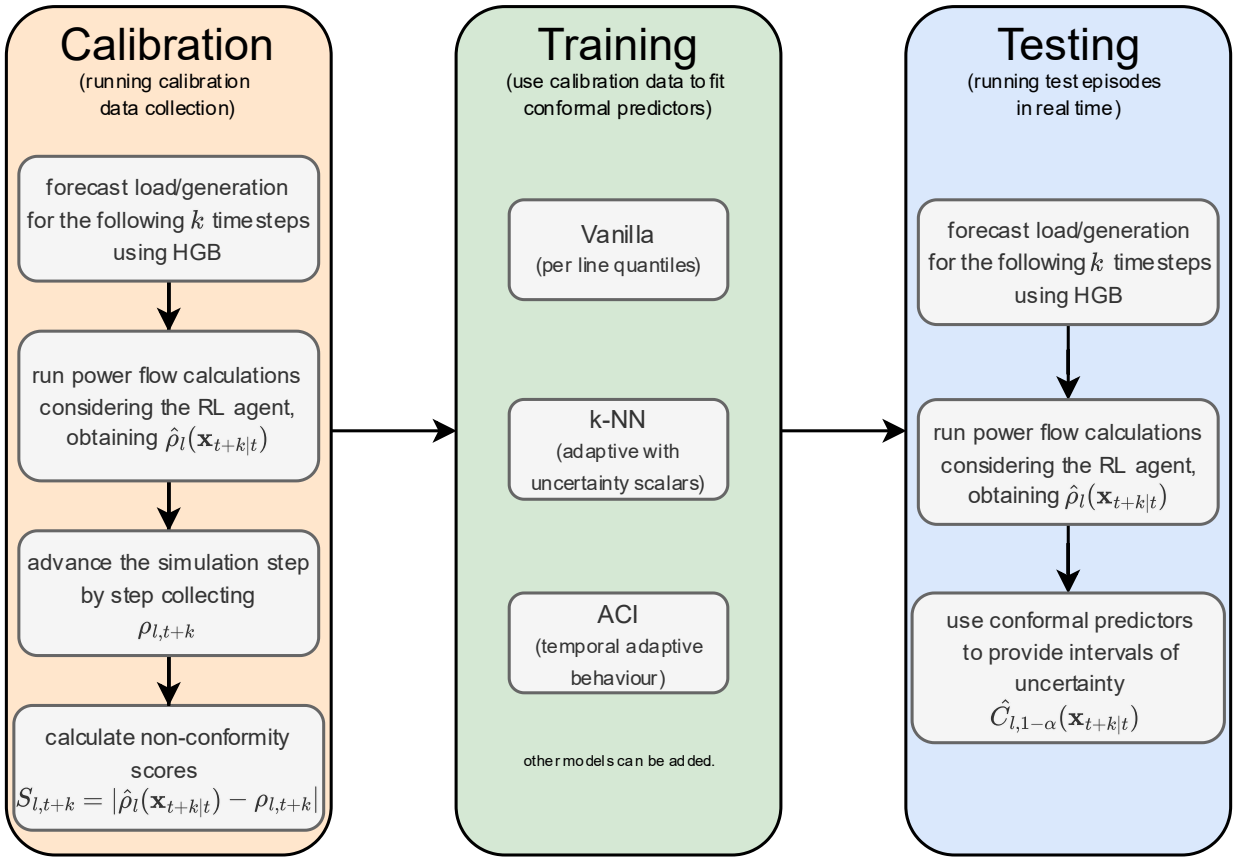
To obtain the forecasted value of the line loadings, the action a_{t+k} that would be taken by the RL agent in the forecasted state $\hat{s}_{t+k|t}$ is simulated resulting in the forecasted state $\mathbf{x}_{t+k|t}$.

From this forecasted state $\mathbf{x}_{t+k|t}$ we obtain the forecasted values of the line loadings $\hat{\rho}(\mathbf{x}_{t+k|t})$

The objective is that given a user-definable parameter $\alpha \in [0,1]$, Conformal Prediction provides an interval of uncertainty $\mathcal{C}_{1-\alpha}$ around the forecasted $\hat{\rho}(\mathbf{x}_{t+k|t})$ such that:

$$P\left(\rho_{t+k} \in \mathcal{C}_{1-\alpha}(\mathbf{x}_{t+k|t})\right) \geq 1-\alpha$$

Conformal Prediction framework



The framework has a pipeline that follows three consecutive phases

Calibration:

- Collecting the calibration dataset

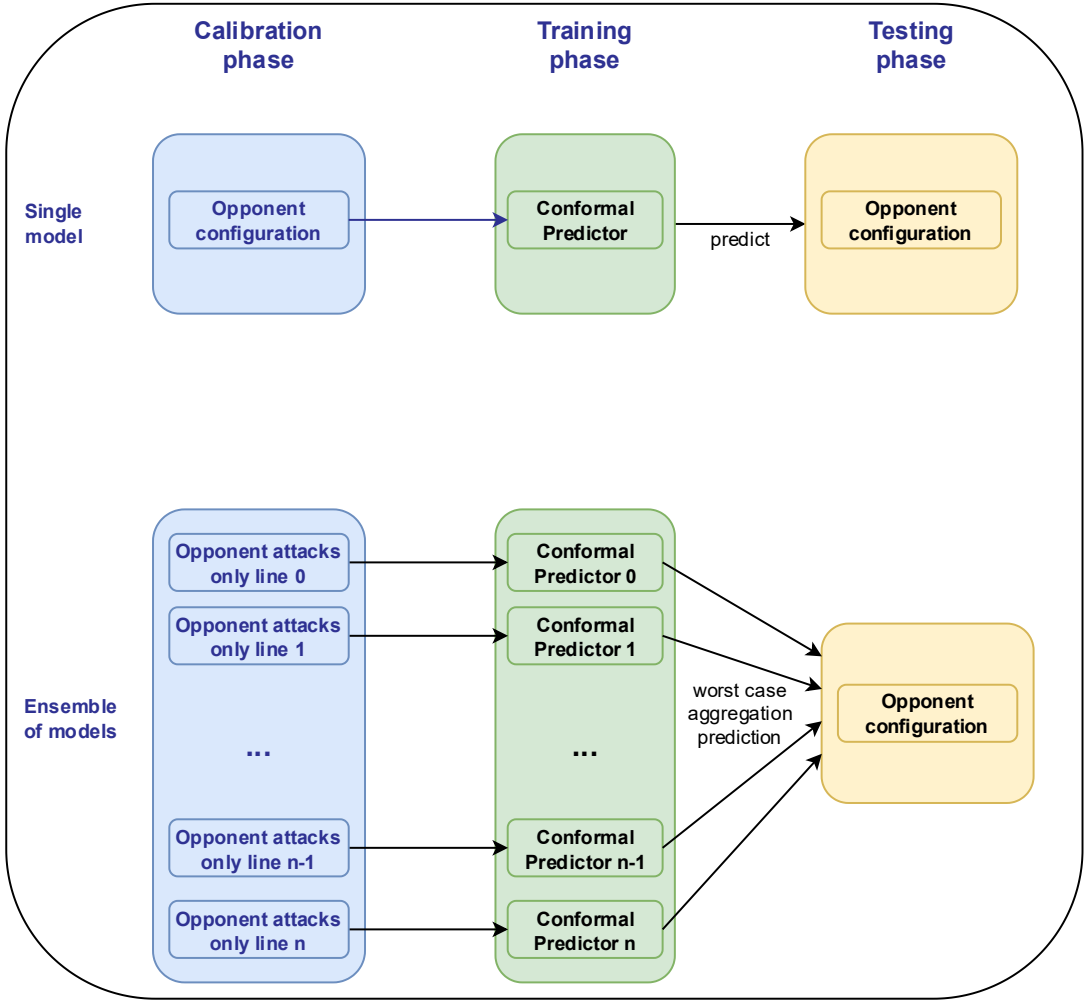
Training:

- Using the calibration dataset to “fit” the Conformal Predictors

Testing:

- Running real-time episodes, using Conformal Predictors to obtain intervals of uncertainty around the forecasted line loadings

Model types



Single model

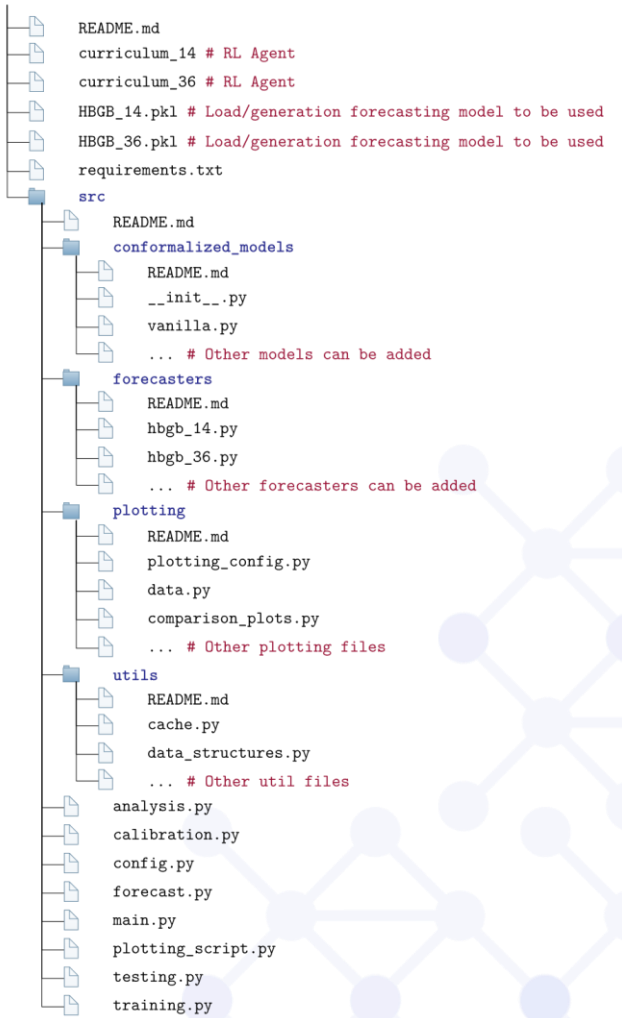
- In the calibration phase, an opponent configuration is set, with $\mathcal{A} \subseteq \mathcal{L}$ where $\mathcal{A}$ represents the set of attacked lines (possibly $\mathcal{A} = \emptyset$), and $\mathcal{L}$ represents the set of all power lines.
- In the training phase, a single predictor is fit.
- In the testing phase, the predictor generates intervals of uncertainty around the line loadings point forecasts. The lines can be subjected to opponent attacks ($|\mathcal{A}| \geq 0$)

Ensemble of models

- In the calibration phase, data is collected for each scenario, with $|\mathcal{A}| = 1$.
- In the training phase, a predictor is fit in each of the scenarios.
- In the testing phase, the opponent can be configured to suffer attacks in $\mathcal{B} \subseteq \mathcal{L}$, with $|\mathcal{B}| \geq 0$. All the Conformal Predictors generate a prediction, and the worst-case scenario is used as a conservative estimate.

Code Structure (Overview)

- The repository contains instructions, having **README.md** files in every directory
- In the **root/** directory there needs to exist both the RL agent (e.g., *curriculum_14/)* and the forecaster (e.g., *HBGB_14.pkl*).
- The framework is contained inside the **src/** folder
  - **config.py** is the configuration file and all settings must be set there
  - **main.py** is the entry point for running the pipeline once config.py is set
  - **plotting_script.py** can be used as a standalone script
  - **calibration.py**, **training.py** and **testing.py** represent the three main phases
  - The **conformalized_models/** directory contains the implementation of the conformal predictors
  - An interface between the forecaster and the framework must be implemented inside the **forecasters/** directory
  - The **plotting/** directory contains the utilities for generating the plots
  - The **utils/** directory contains utilities used by the pipeline



Starting a new experiment

- To start a new experience set the parameters in **config.py**:
  - For example:
    - ALPHA = 0.1 # Significance level. Can be a list or a single value. If it is a list, it will run an experiment for each alpha
    - CALIB_EPISODES = 30 # Number of episodes from which we collect the calibration dataset
    - TEST_EPISODES = 30 # Number of episodes to run, to observe the results
    - OUTPUT_DIR = "RESULTS/MY_EXP" # Directory where to save the results (.csv and plots)
    - STEPS_TO_RUN = 8064 # Maximum number of steps to run each episode for
    - AUTO_GEN_PLOTS = True # Whether to automatically generate plots. If set to false, it only generates .csv's that can be later used by the plotting_script.py
    - MODELS = { # Which Conformal Predictors to use

      "vanilla": (True, "vanilla"),

      "knn_norm": (True, "knn_norm"),

      "aci": (True, "aci"),

      … # other models

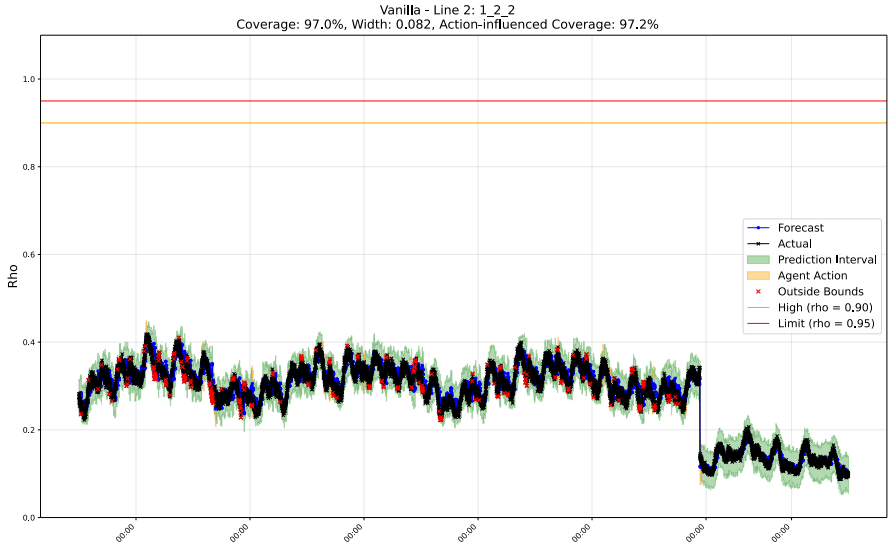
      }
    - ENV_NAME = "l2rpn_case14_sandbox" # Grid2Op Environment
    - other parameters.. # For example, opponent can be configured

- Navigate to the **src/** directory and run:

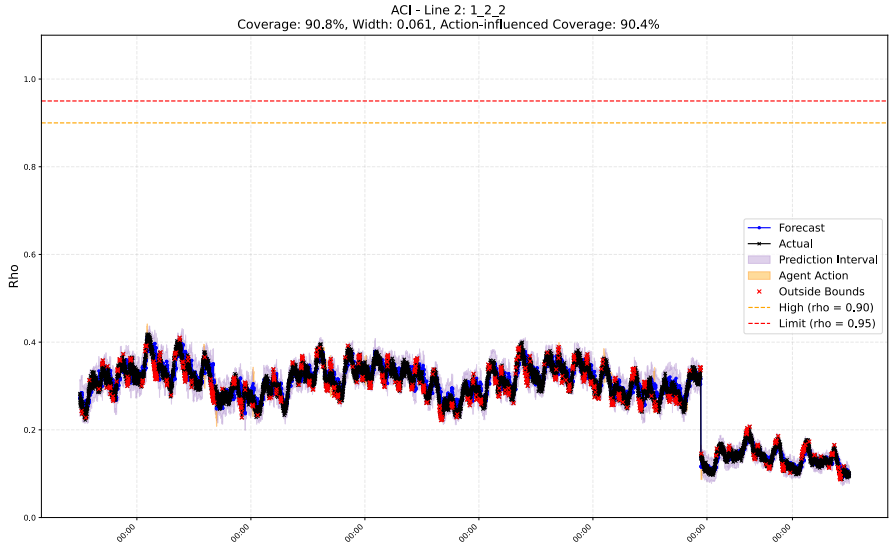
  $ ***python main.py x***

Results

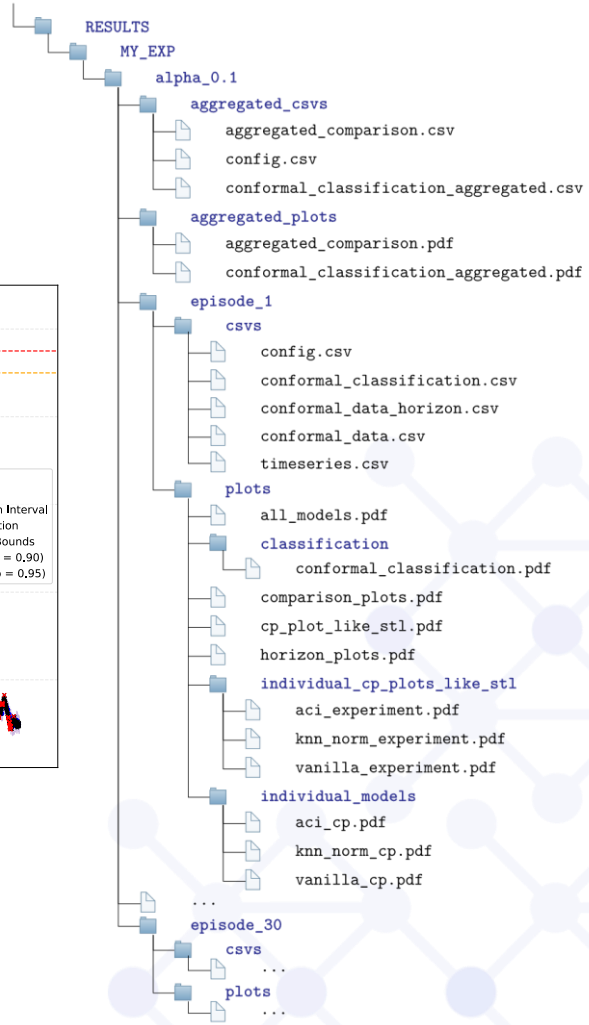
After running the experiment, we will have the output structure on the right.



Example: Split CP on line 2 - Chronic 931

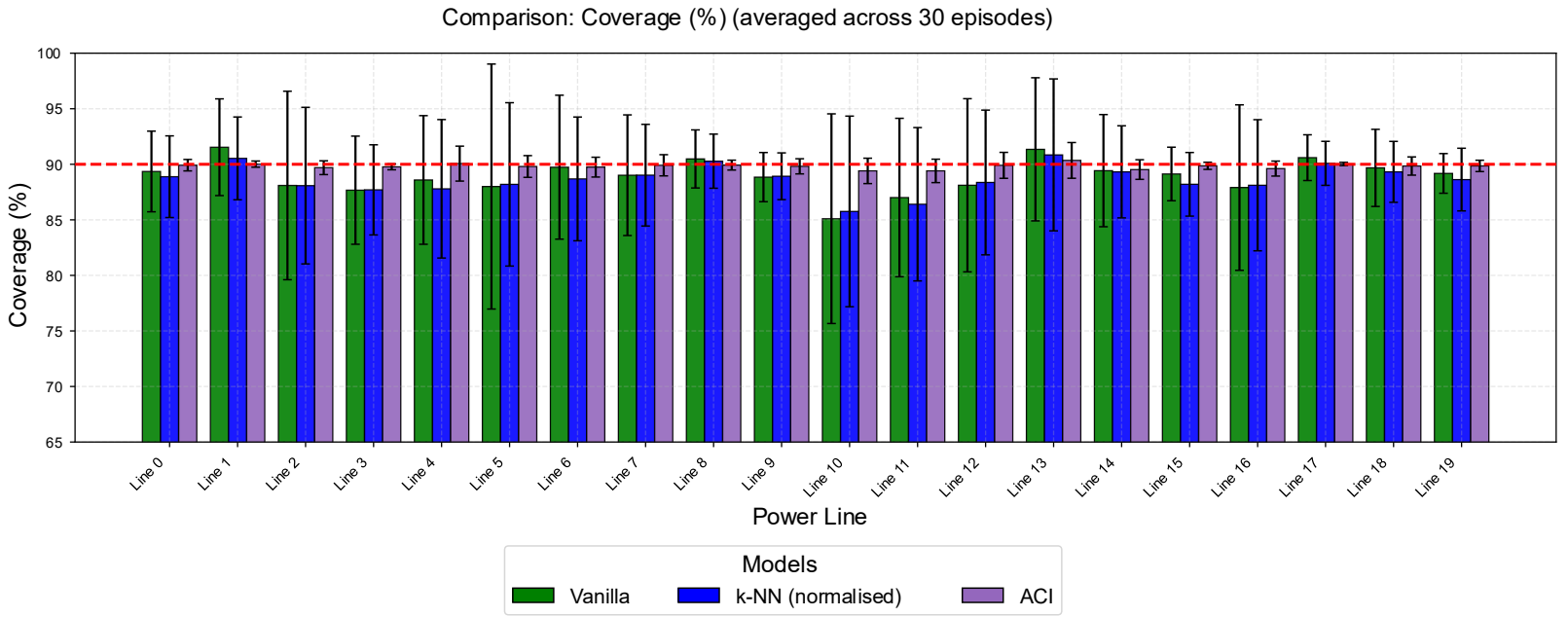


Example: ACI on line 2 - Chronic 931



Results

After running the experiment, we will have the output structure on the right.



Example: Coverage averaged from chronic 931 to chronic 960

```
RESULTS
MY_EXP
alpha_0.1
aggregated_csvs
aggregated_comparison.csv
config.csv
conformal_classification_aggregated.csv
aggregated_plots
aggregated_comparison.pdf
conformal_classification_aggregated.pdf
episode_1
csvs
config.csv
conformal_classification.csv
conformal_data_horizon.csv
conformal_data.csv
timeseries.csv
plots
all_models.pdf
classification
conformal_classification.pdf
comparison_plots.pdf
cp_plot_like_stl.pdf
horizon_plots.pdf
individual_cp_plots_like_stl
aci_experiment.pdf
knn_norm_experiment.pdf
vanilla_experiment.pdf
individual_models
aci_cp.pdf
knn_norm_cp.pdf
vanilla_cp.pdf
...
episode_30
csvs
...
plots
...
```

Authors	Institution
Hugo Cardante	INESC TEC
Ricardo Bessa	INESC TEC
Margarida Costa	INESC TEC
Carla Gonçalves	INESC TEC
Pedro Ferreira	INESC TEC

Epistemic Uncertainty and Predicting RL Agent Failure Probability

Contributors: INESC TEC

Context

Motivation

RL agents might operate in high-risk states without realizing it. To ensure safety, we must **estimate epistemic and aleatoric uncertainty** metrics and integrate this information into a classifier that predicts if the AI agent will provide recommendations that solve congestion problems, avoid cascading failures before they occur

Proposed methodology

Failure prediction framework leveraging **Uncertainty Quantification (UQ)** to create a self-awareness feature for AI-based assistants

GitHub

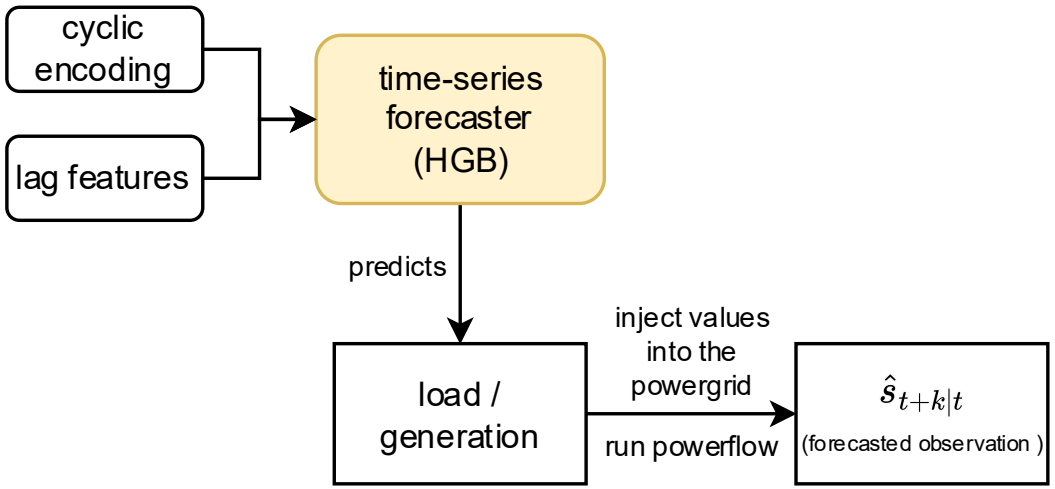
Proposed Methodology

1. **Uncertainty decomposition**
   - **Aleatoric (data):** Captures stochastic variability in load and generation (via *Residual Forecasting*)
   - **Epistemic (model):** Detects unknown or *Out-of-Distribution* grid states (via *Evidential Neural Network*)
2. **Risk classification**
   - Integrates uncertainty metrics with physical grid states (power flows, topology)
   - Trains a classifier to predict minutes ahead if (the probability) the AI assistant recommendations can avoid **cascading failure probability** prior to action execution
3. **Final objective**
   - Provide and predict a confidence level of the AI assistant recommendation to prevent risky operations in critical environments

Failure Probability Framework

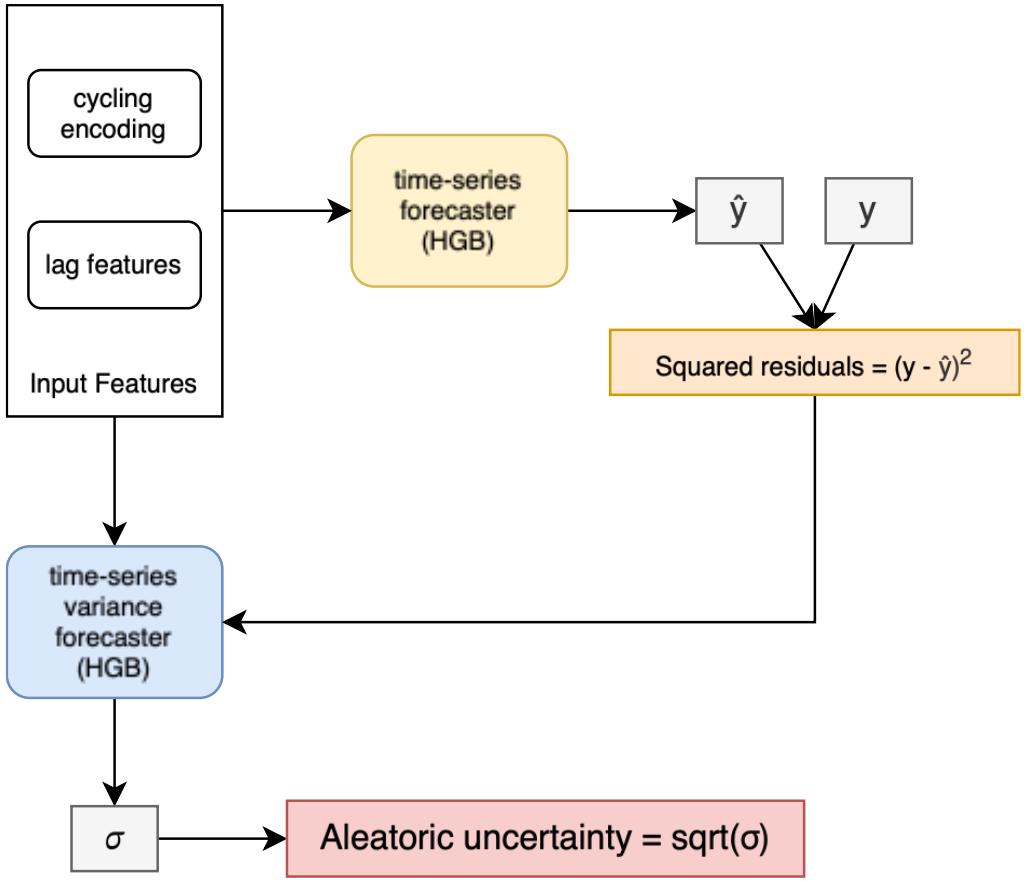
- Obtaining the forecasting states
- Obtaining the aleatoric uncertainty of forecasting states
- Obtaining the epistemic uncertainty of the agent for each observation
- Framework pipeline
- Repository structure overview
- Starting a new experiment

Obtaining the forecasting states



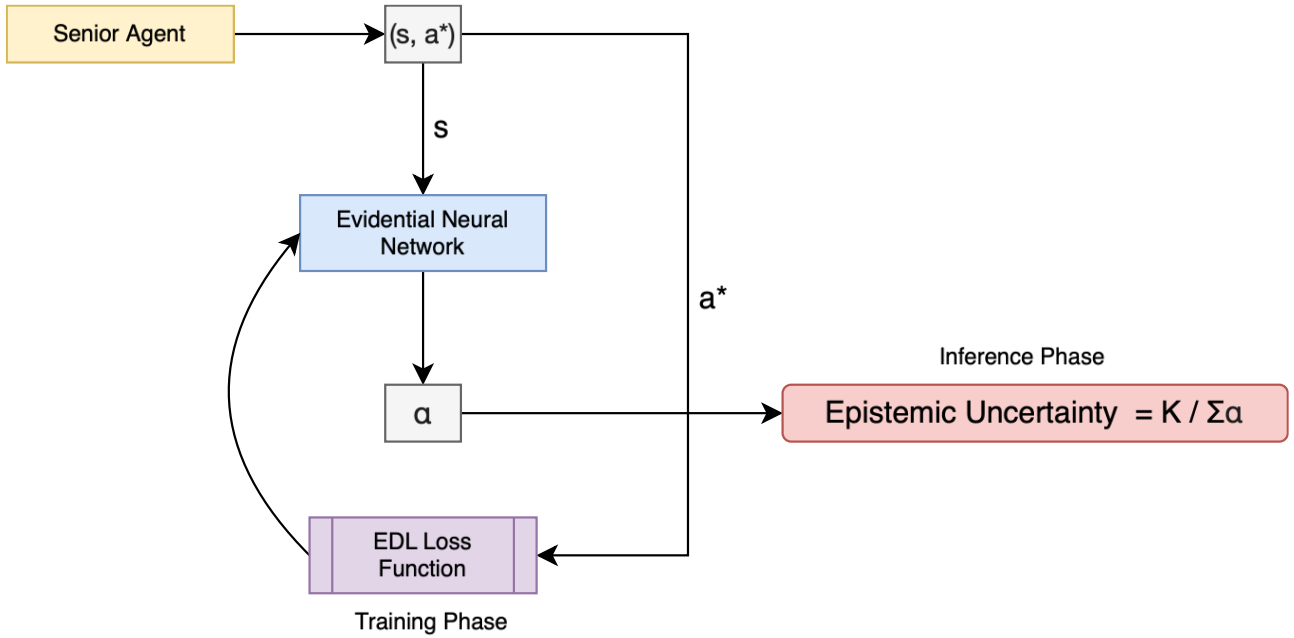
- A time-series forecaster trained as a multi-output regressor is used to predict active and reactive power injections using historical observations and temporal variables (lagged features) for all loads and generators over a 1-hour time horizon (timesteps t+1, …, t+12).
- These forecasted values are injected into the grid, and a power flow is run to obtain the forecasted state of the grid.

Obtaining the aleatoric uncertainty



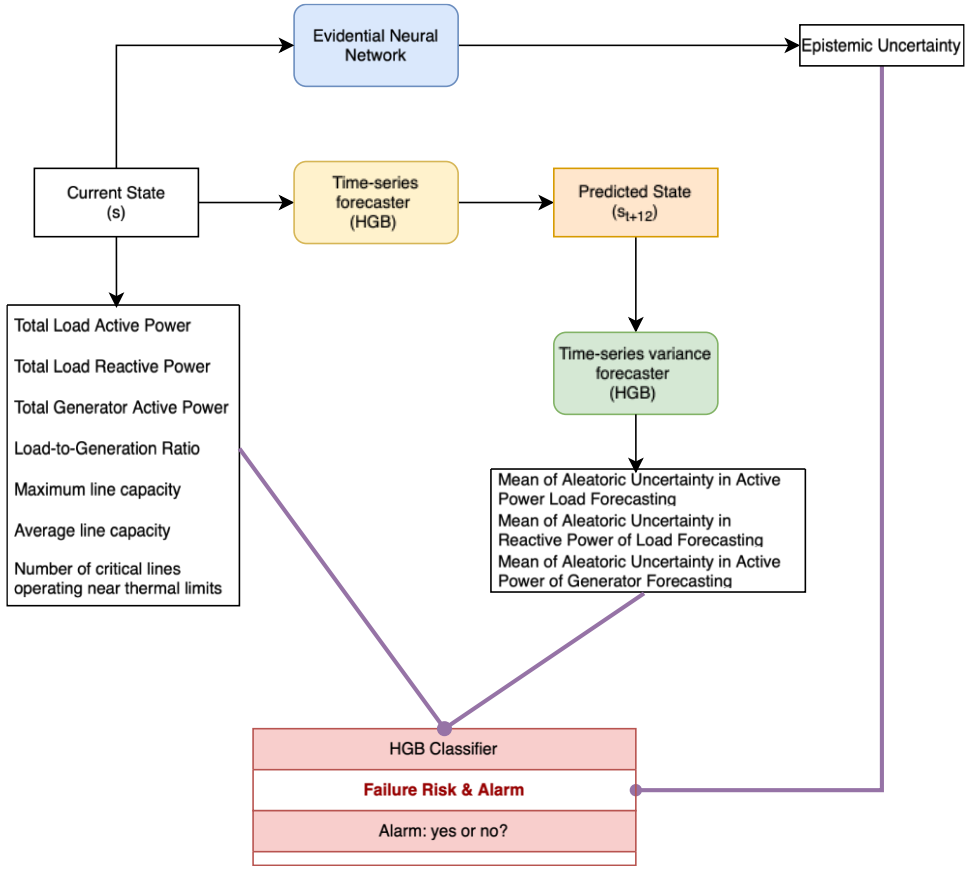
- Squared residuals are computed between the ground truth and the predictions generated by the previously described mean model. These residuals subsequently serve as the target variable for training a secondary regression model, which predicts the data variance to quantify the **aleatoric uncertainty**.

Obtaining the epistemic uncertainty of agent for each observation



- Epistemic uncertainty is quantified via **Knowledge Distillation**, where an **Evidential Neural Network (ENN)** is optimized to replicate the expert policy of the Senior Agent
- Rather than standard softmax probabilities, the network parametrizes a **Dirichlet distribution (α)** over the action space. Consequently, model ignorance is derived analytically as the inverse of total evidence ($u=K/\sum\alpha$), acting as a direct proxy for Out-of-Distribution (OOD) states

Framework pipeline



- *Objective*: Predict whether a **future line disconnection** will lead to a system failure **one hour ahead**, using the **current grid state** and **forecast uncertainty**
- **Current Grid State (s_t)**
  - Global load-generation balance
  - Thermal stress indicators
  - Epistemic Uncertainty (ENN) -> confidence in the current state estimation
- **Future Scenario Forecast (s_{t+12})**
  - Load and generation projections
  - Aleatoric uncertainty -> intrinsic variability of forecasts
- **Contingency Analysis**
  - *What-if* disconnection of a specific powerline
  - Failure evaluation **1 hour after the contingency**
- **Failure Classification**
  - Inputs: current grid state + predictive uncertainties + disconnected line
  - Binary output:
    - **0** – stable operation
    - **1** – predicted failure -> **alarm triggered**

Repository structure overview

- In the **grid_security_project/** root directory:
  - **run_pipeline.py:** The main entry point. Orchestrates the full pipeline execution (Training -> Data Collection -> Inference.
  - **README.md:** Documentation and setup instructions.
  - **agent/:** **P**re-trained RL agents required for the simulation. (eg.: *agent/network36*)
  - **data/:**
    - Training data to forecaster models (eg.: *X_train_36.npy, y_train_36.npy*).
    - ENN training and validation datasets (eg:. *senior_expert_train_36.npz* e *senior_expert_val_36.npz*).
    - Training data for risk classifier (*uncertainty_disconnection_analysis.csv*).
  - **forecasts/:** Trained load and generation forecaster models(e.g.: *HGBG_36pkl*).
  - **models/:** Stores the serialized models generated by the pipeline:
    - **Aleatoric Model** (eg.: *models/HBGB_36_aleatoric.pkl*)
    - **Epistemic Model** (eg.: *models/enn_36.pth*)
    - **Risk Classifier** (eg.: *models/final_classifier36.pkl*)
  - **src/:** Core implementation of the framework:
    - **config.py:** Configuration file and all settings must be set there.
    - **train_forecast.py, training_enn.py, collect_data.py, train_classifier.py:** pipeline core modules.
    - **enn_models.py** (Evidential Neural Newtork Architecture)
    - **utilis.py** (Feature Enginnering): utilites.

```
./ grid_security_project
    agents # Pre-trained Grid2Op agent
        network36
    data # Generated datasets
        uncertainty_disconnection_analysis.csv
        X_train_36.npy
        y_train_36.npy
    forecasts # Forecaster model
        HBGB_36.pkl
    models # Serialized pipeline outputs
        enn_36.pth # Epistemic (ENN)
        enn_data # ENN datasets
            senior_expert_train_l2rpn_icaps_2021_small.npz
            senior_expert_val_l2rpn_icaps_2021_small.npz
        HBGB_36_aleatoric.pkl # Aleatoric model
        final_classifier36.pkl # Risk classifier
    src # Core modules
        collect_data.py
        config.py # Global settings
        enn_models.py # ENN architectures
        train_classifier.py
        train_forecast.py
        training_enn.py
        utils.py # Feature engineering
    run_pipeline.py
    requirements.txt
```

Starting a new experiment

- To start a new experience set the parameters in config.py:

Parameters	Value	Result
ACTIVE_ENV	l2rpn_case14_sandox / l2rpn_icaps_2021	The script loads Config14/ Config36 and the specific input dimension (467/1363)
TRAIN_MODE	True/False	Starts/Skips the training loop and attempts to load the .pth and .pkl files from your models/ directory.
TEST_SINGLE_EPISODE	True/False	The run_pipeline.py will launch one full/all episode(s) (scenario) to evaluate how weel the agent manages the grid.

- Navigate to the src/ directory and run:

$ python run_pipeline.py

Authors	Institution
Margarida Costa	INESC TEC
Ricardo Bessa	INESC TEC
Carla Gonçalves	INESC TEC
Pedro Ferreira	INESC TEC

Agent As A Service (A3S) & Trace RL

EnliteAI

- Context
- Methodology
- Original contributions
- Overview of the repository
- Interfaces & Interaction Pattern

Context

Motivation

WP2 produced high-performing agents, but operational deployment in critical infrastructures requires human oversight, auditability, and safe intervention during rollout—especially when context shifts or confidence drops.

Definition

Agent-As-A-Service (A3S) is a runtime service layer that wraps an existing agent + simulation environment and exposes standardized endpoints to restore state, query action spaces, and simulate “what-if” futures, enabling operators to inspect, challenge, and override decisions with minimal cognitive load.

Use cases

Human-in-the-loop decision support for operations; offline replay & audit of decisions; operator training via simulation-backed counterfactual exploration.

Methodology (Human-in-the-Loop Integration)

Design goal: “wrap, don’t retrain.”
A3S deliberately targets the **rollout phase**: taking a trained agent and turning it into a supervised, steerable decision-support service

Core loop (Human in the Loop):

- **Agent proposes** an action for the current state
- A3S exposes **action space + state** through service endpoints
- Operator can **inspect the trajectory** (replay / KPIs / renderings)
- Operator can **override actions** at selected decision points
- A3S **simulates forward** from the restored state to verify consequences
- UI updates with the **alternative future** (branch) for comparison

Implementation pattern:

- Backend service hosts the agent + environment, exposed via lightweight IPC (Redis-based)
- Configuration via YAML/Hydra to swap environments/agents without code changes
- Frontend connects to backend for interactive what-if rollouts and action overrides

Original contribution

1) A3S: Standardized "agent wrapper" for supervised deployment

Transforms WP2 agents into a service-oriented component usable by operator tools, simulators, and dashboards—without touching training code.

2) Simulation-backed interpretability (not static XAI)

Instead of explaining a single action post-hoc, A3S supports interactive validation:
"*What happens if we take action B instead of A?*" — answered by live rollout simulation.

3) Autonomy-level tuning at runtime

Supports a continuum from:

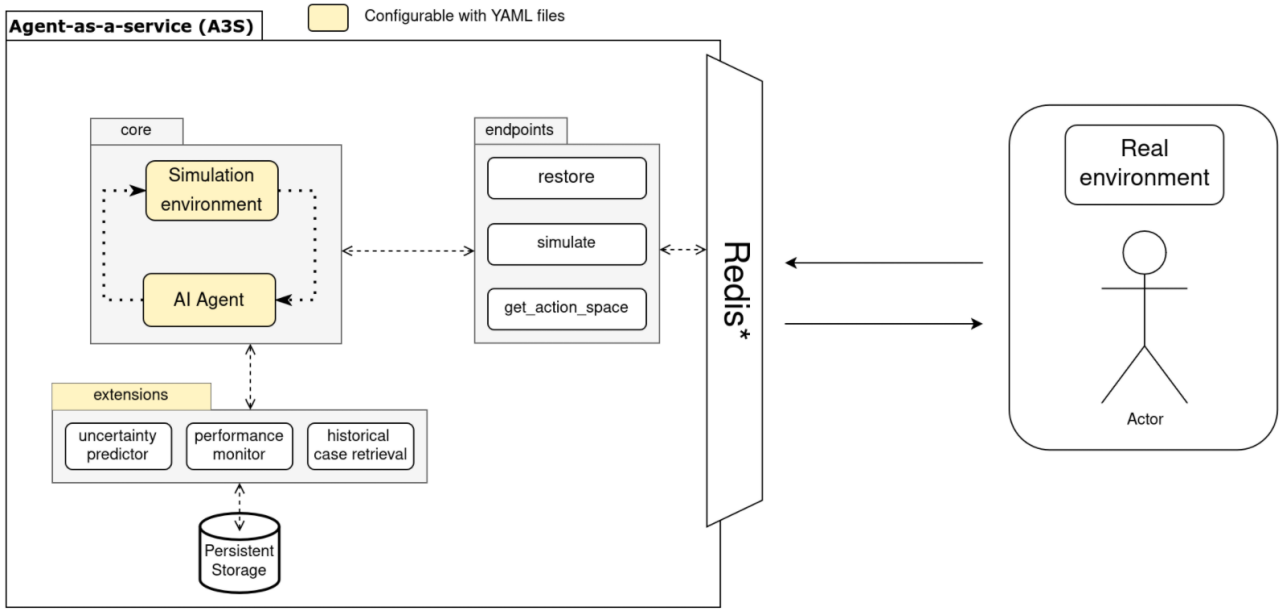
- autonomous recommendations → supervised mode → operator override → simulation-only validation.

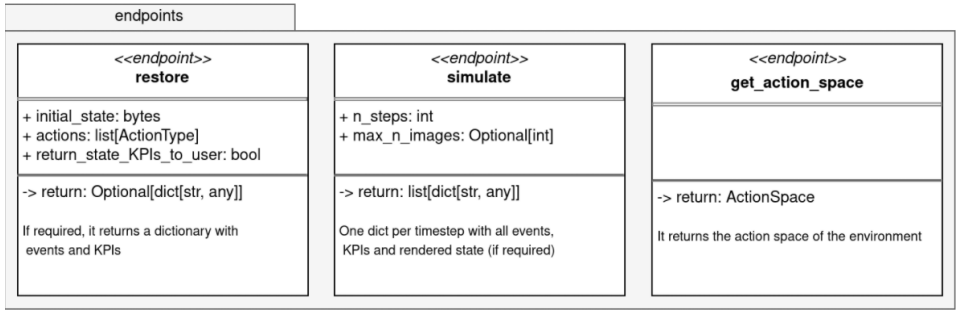
4) TraceRL integration for human-centered inspection

TraceRL operationalizes A3S through:

- Trajectory trees (branching futures)
- Visual inspection of decision blocks
- Action override & re-computation from the UI

Original contribution





Overview of repository structure

Two main deliverables inside one repo:

- **A3S backend** (Task 3.1): restore/simulate/action-space service for live rollout supervision
- **TraceRL** (Task 2.3): trajectory recording + Dash visualization + branching what-if analysis

```
agent-as-a-service-trace-rl/
├─ agent-as-a-service/          # A3S backend (agent + env as a service)
│  └─ agent_as_a_service/
│     ├─ conf/                  # Hydra YAML configs (env/agent/container/wrappers)
│     └─ src/                   # service runner / container logic
├─ trace-rl/                    # TraceRL (trajectory + visualization + HITL UI)
│  ├─ scripts/                  # trajectory collector (e.g., Gym demo)
│  └─ trace_rl/
│     └─ app/                   # Dash application (UI), connects to backend
├─ assets/                      # images / examples used in docs & UI
├─ environment.yml              # conda env (Dash, Gymnasium, Flatland, Torch, etc.)
├─ pyproject.toml               # packaging (trace_rl + agent_as_a_service)
└─ readme.md                    # setup + end-to-end workflow
```

A3S Interfaces & Interaction Pattern

Minimal endpoint set enabling HITL:

- **Restore** → jump to any prior decision point (replay / audit)
- **Get action space** → make operator choices explicit & valid
- **Simulate** → verify consequences before committing (counterfactual safety check)

Human cognitive load principle:

- show *only* the information needed to decide (current state, feasible actions, predicted outcomes),
- enable fast branching ("try option A vs B") instead of deep model introspection.

TraceRL UI connection:

- connect to backend → select decision block → override → simulate → compare branches.

Link to the repository

Link to repository: **https://github.com/AI4REALNET/agent-as-a-service-trace-rl**

Authors	Institution
Alberto Castagna	EnliteAI
Anton Fuxjaeger	EnliteAI

T3.2 – Multi-Objective Decision-Making

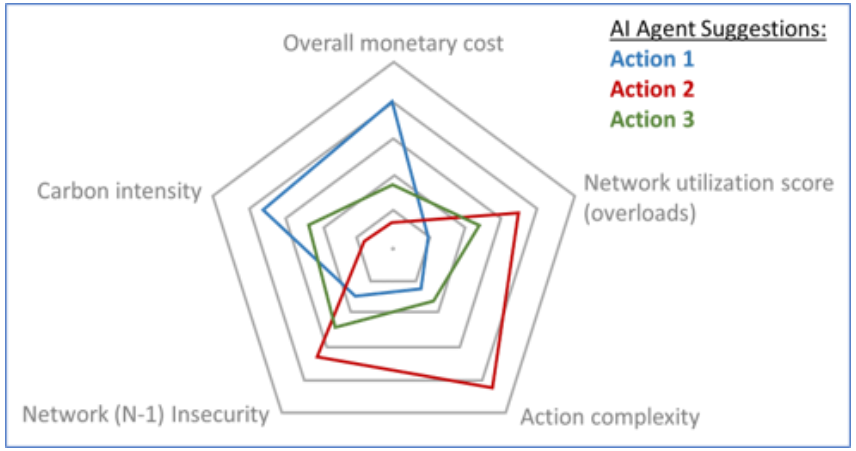
Leader: UKassel

Contributors: IRTSX, FHG, FLAT

Context

Motivation

Competing objectives in complex network operations require a dynamic assessment of a situation by human operators. As a result, the relative importance (weight) of objectives is not known a priori, and might change in real time.



Strategy

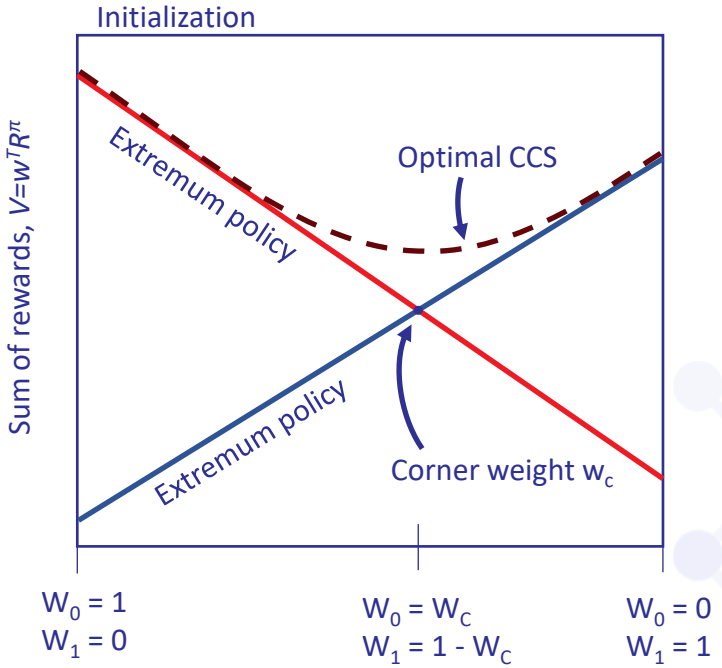
The strategy to adapt to multi-objective environments is to create an **ensemble of policies** that collectively define a **pareto front**, allowing the human operator to choose an action out of a list of optimal strategies based on his assessment of the relative importance of objectives.

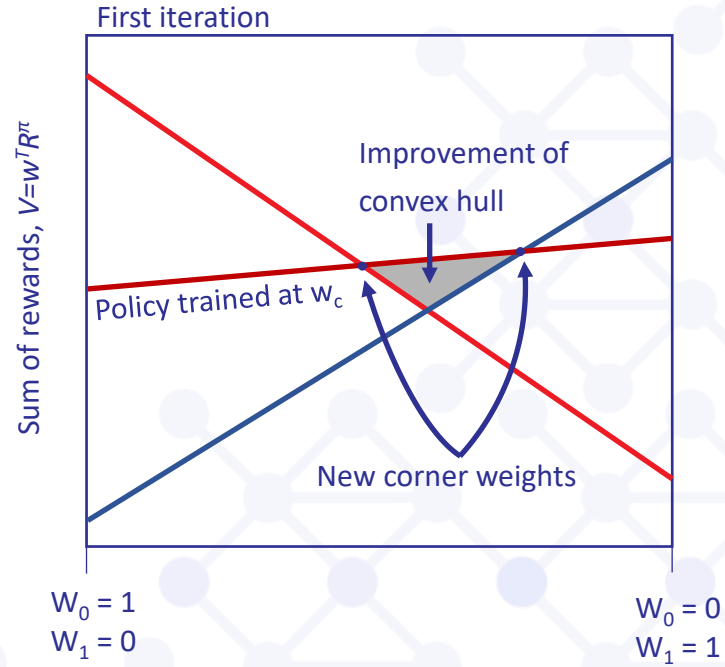
Use cases

Successfully implemented in the electrical grid domain; plans to implement in the Air Traffic domain (BlueSky)

Methodology (1/2)

- Main goal: develop an **ensemble of policies** to accommodate any linear combination of weighted objectives
- Optimistic Linear Support Algorithm:
  - Train on a **scalarized objective** (linear weights assigned to each objective)
  - Start with extreme scenarios:
    - $w_0 = 1, w_1 = 0$
    - $w_0 = 0, w_1 = 1$
  - Iteratively identify corner weights, find optimal policy at those points to efficiently expand **convex coverage set**
- Set of policies and weights defines the **pareto front**





Methodology (2/2)

- Determination of Pareto Front:
  Deep Optimistic Linear Support (DOL) algorithm
  - General algorithm, domain agnostic
  - Code implementation on GitHub
- Environment:
  Integrates with the MORL-Baselines toolkit
- Agents and training:
  - Agent-agnostic, example implementations shown with Multi-objective PPO algorithm

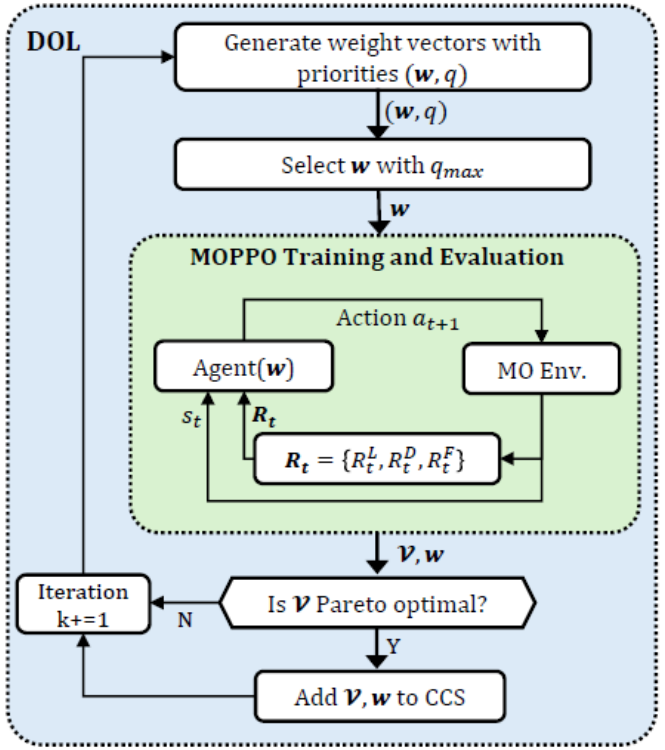


Fig. 1: Schematic of the proposed MORL approach with deep optimistic linear support.

Code Structure, Status and Plans

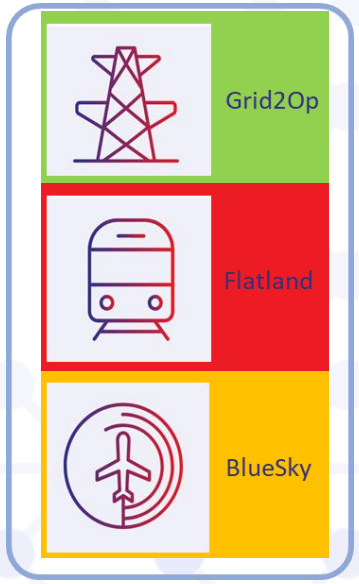
```
├── ai4realnet_morl
│   ├── agent
│   │   ├── MO_BaselineAgents.py
│   │   └── MO_PPO.py
│   ├── envs
│   │   ├── CustomGymEnv.py
│   │   ├── EnvSetup.py
│   │   └── GridRewards.py
│   ├── utils
│   │   ├── Dataloader.py
│   │   ├── MO_PPO_train_utils.py
│   │   └── gym_eval.py
│   └── wrapper
│   │   ├── ols.py
│   │   └── ols_DOL.py        main executable
├── scripts
│   └── plot_morl_figures.py
├── README.md
├── requirements.txt
└── setup.py
```

Current Version

- Domain-agnostic package: ai4realnet-morl
- Example implementation for Grid2Op: grid2op-ai4realnet-morl

Status and Plans

- Implementation carried out for Grid2Op Environment
- Work in progress to implement in BlueSky environment



Authors	Institution
Mohamed Hassouna	UKassel
Eduardo Vilches	UKassel

T3.3 – Interactive AI to Augment Decision-Making

Leader: TUD

Contributors: POLIMI, UvA, FHNW, SBB, FLATLAND

Overview 3.3

Interactive AI to Augment Decision-Making

- Inverse Reinforcement Learning with Risk-sensitive behavior
- Shaping AI Behavior to Operational 'Best Practices'
- Interactive Preference Learning in ATM Sectorization
- Human Learning Support

Inverse Reinforcement Learning with Risk-sensitive behavior

POLIMI

- Context
- Methodology
- Original contributions
- Overview of the repository
- Experiments

Context

Motivation

Humans commonly engage in risk-sensitive behaviors in the presence of stochasticity, but no algorithm in the literature allows to learn their utility function, whose knowledge would allow predictive and descriptive applications on their behaviors

Definition

The utility function of an agent represents its risk attitude, and formally represents its valuation for money or goods at stake

Use cases

Knowing the utility of an agent allows predicting its behavior for instance in games but also in financial applications

Methodology

- **Main idea** = Assume the human expert takes optimal risk-sensitive decisions based on a utility function in a known Markov Decision Process.

Collect **large number of samples** from the environment and the expert → Use **maximum likelihood estimators** for the quantities (e.g., occupancy measure) at stake → Perform **projected subgradient descent** to obtain a good utility function

Original contribution

- **Algorithm #1 |** CATY-UL
  - **Short description:** Classify input utilities as compatible or not
  - **Contribution:** First provably efficient algorithm of this kind
  - **Implemented features:** Tabular MDPs
- **Algorithm #2 |** TRACTOR-UL
  - **Short description:** Extract a meaningful utility from expert demonstrations
  - **Contribution:** Projected subgradient descent-based algorithm
  - **Implemented features:** Tabular MDPs

Overview of repository structure

The code is composed of two main files:

- 📂 **algorithm.py** containing implementations of CATY-UL and TRACTOR-UL
- 📂 **environment.py** for simulations in tabular MDPs

A description of how to use the code is provided in the README.

Experiments

Two main experiments:

- **Experiment 1:** Test on true human demonstrations provided by 15 colleagues in the lab the amount of non-Markovianity exhibited.
- **Experiment 2:** Run TRACTOR-UL on synthetic data on various MDPs to assess its computational and sample complexities in practice.

Shaping AI Behavior to Operational 'Best Practices'

TUD

- Context
- Methodology
- Original contributions
- Overview of the repository

RL learning & behavior shaping

Motivation

Accelerate learning and demonstrated RL behavior to operationally acceptable performance using an augmented RL architecture.

Developed Functions

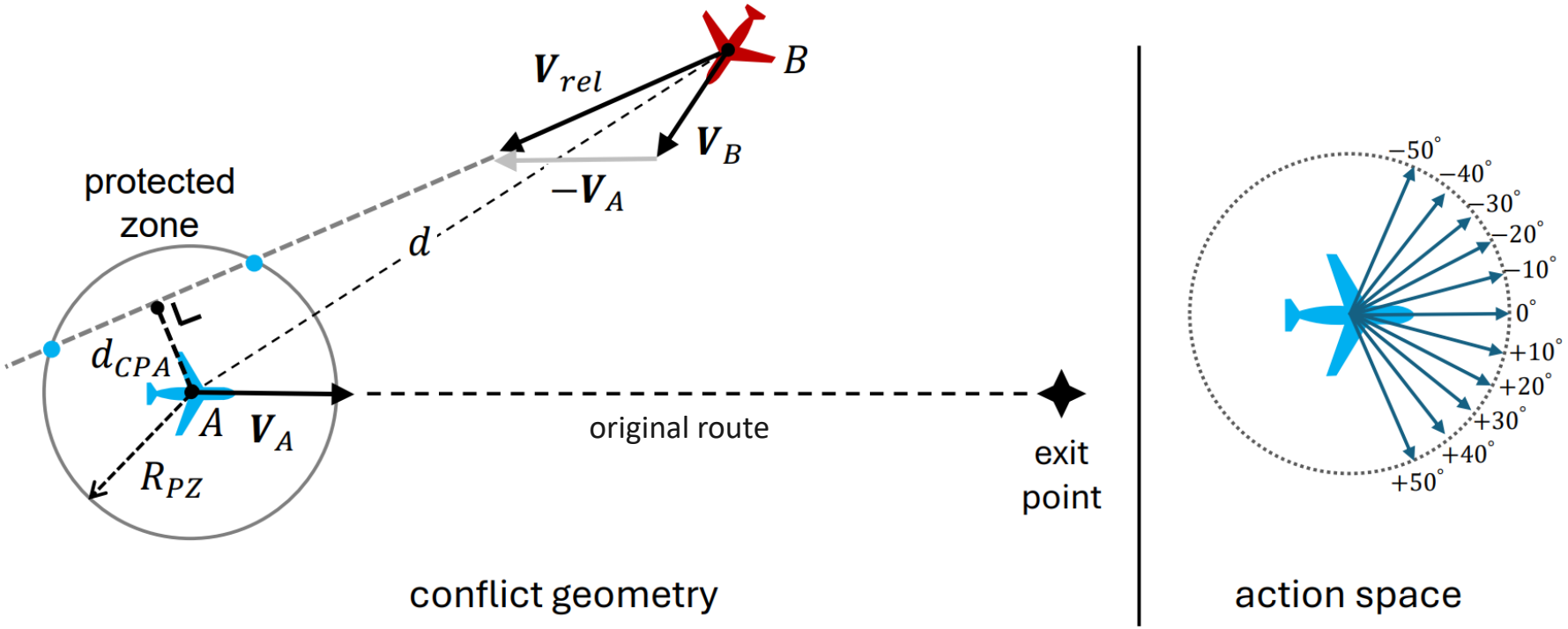
Action Shielding: during training and/or operation, shield all unsafe actions from the RL agent.
Human Feedback: during training, humans shape the RL policy using pairwise or single episode ratings.
Expert Demonstrations: during training, a heuristic algorithm modeled after human expertise shapes the RL policy by imitating the "expert" while allowing exploration.

Use cases

Applicable to Air Traffic Management (ATM) flow management use case (BlueSky)

Context: Aircraft conflict resolution in 2D

The goal is to avoid violating the minimum allowed separation of 5 NM between the two aircraft, while minimizing the route and heading deviation from the original trajectory, and minimize the number of (heading and/or speed) maneuvers taken. The resolution maneuver should be aligned with operational 'best practices' to foster high acceptance among air traffic controllers.



Methodology: Q-learning with function approximation

Action-value function

$$Q(s,a) = \mathbf{w_a}^{\top} \phi(s)$$

Feature vector

$$\phi(s) = \left[d,\ \dot{d},\ \sin\theta_{\text{rel}}, \cos\theta_{\text{rel}},\ d_{\text{CPA}},\ t_{\text{CPA}},\ s_{\perp},\ s_{\parallel},\ \sin\theta_{\text{orig}}, \cos\theta_{\text{orig}},\ 1 \right]^{\top}$$

Q-learning update

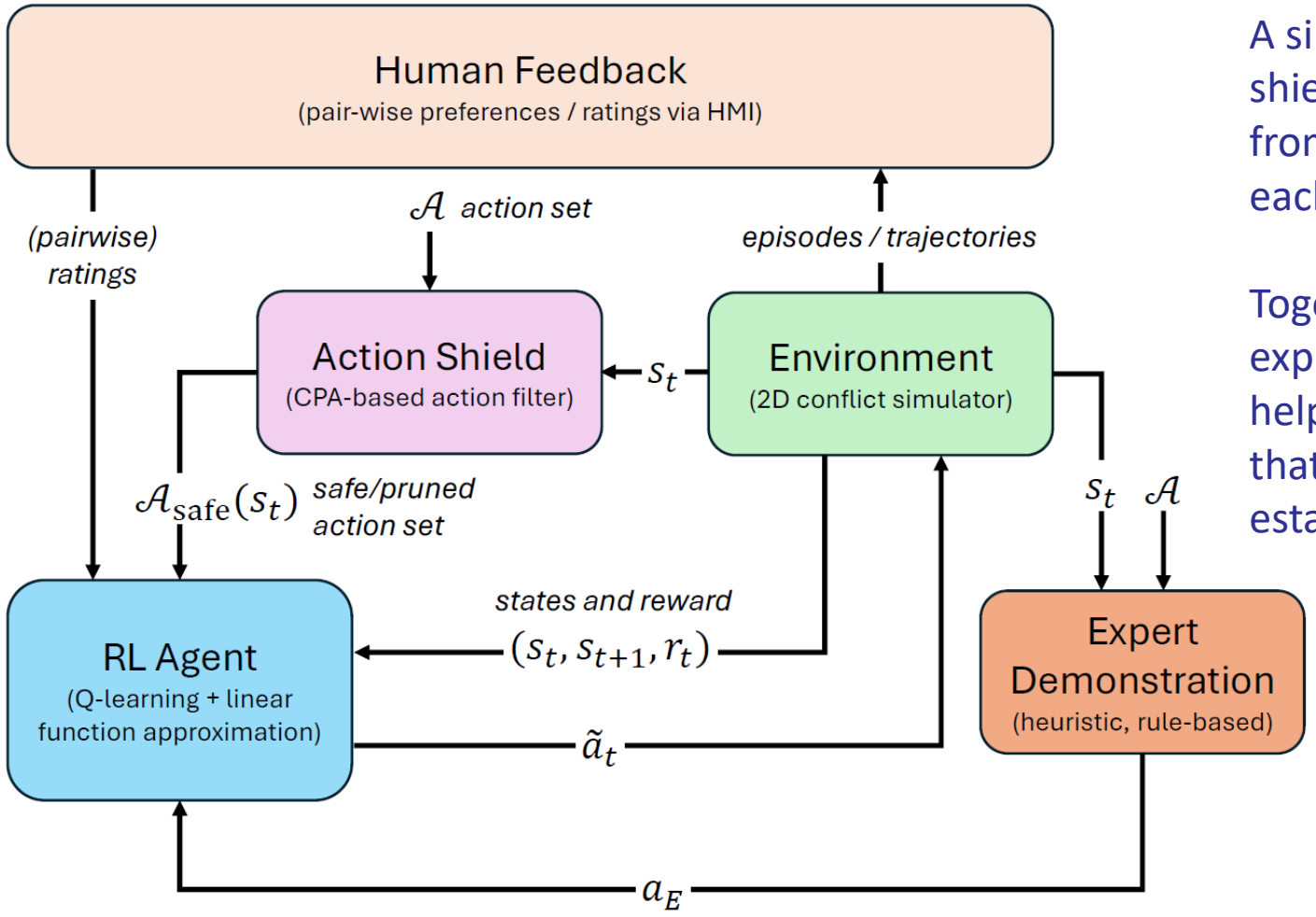
$$\delta = r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

$$\mathbf{w_a} \leftarrow \mathbf{w_a} + \alpha(\delta\, \boldsymbol{\phi}(s) - \lambda \mathbf{w_a})$$

Learning rate

L2 regularization factor

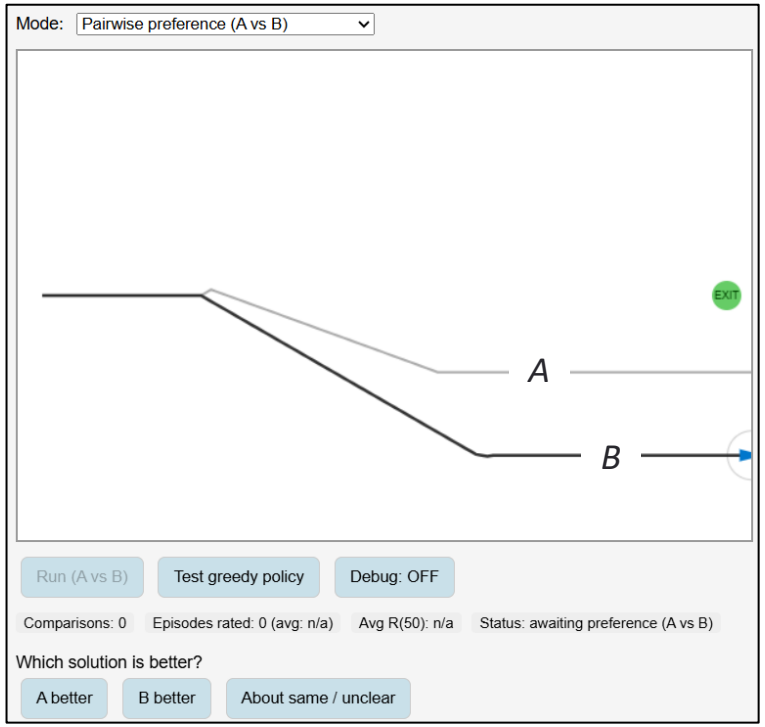
Original contributions: RL behavior shaping architecture

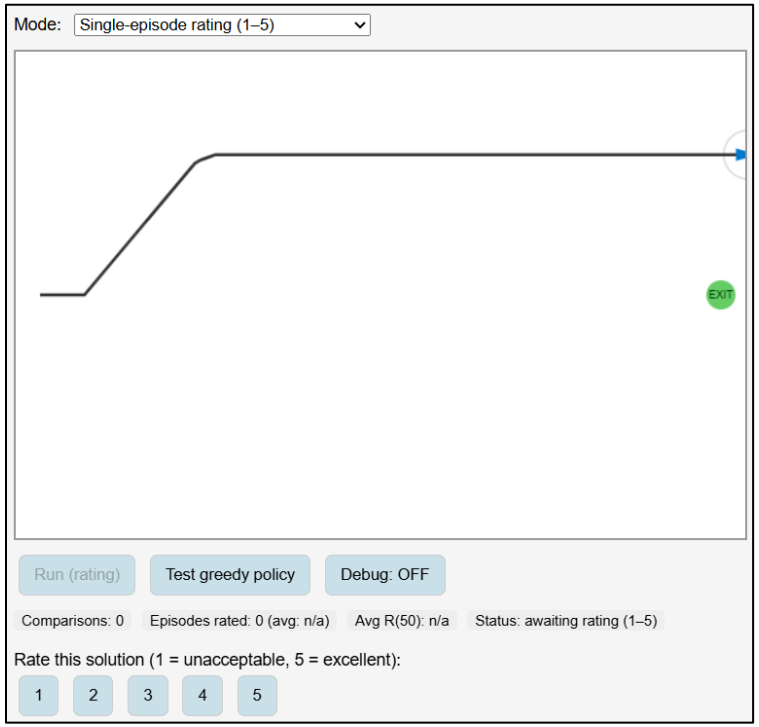


A single RL architecture with pre-selection (pre-shielding) action filter, human feedback and learning from demonstrations, enabling the evaluation of each technique both individually and in combination.

Together, these mechanisms reduce the effective exploration burden, improve sample efficiency, and help the learned policy converge toward solutions that are both performant and consistent with established operational practices.

Original contributions: Human Feedback





If A is preferred over B:

$$\mathbf{w}_a \leftarrow \mathbf{w}_a + \alpha_h\left(\nabla_w Q(A) - \nabla_w Q(B)\right)$$

If B is preferred over A:

$$\mathbf{w}_a \leftarrow \mathbf{w}_a + \alpha_h\left(\nabla_w Q(B) - \nabla_w Q(A)\right)$$

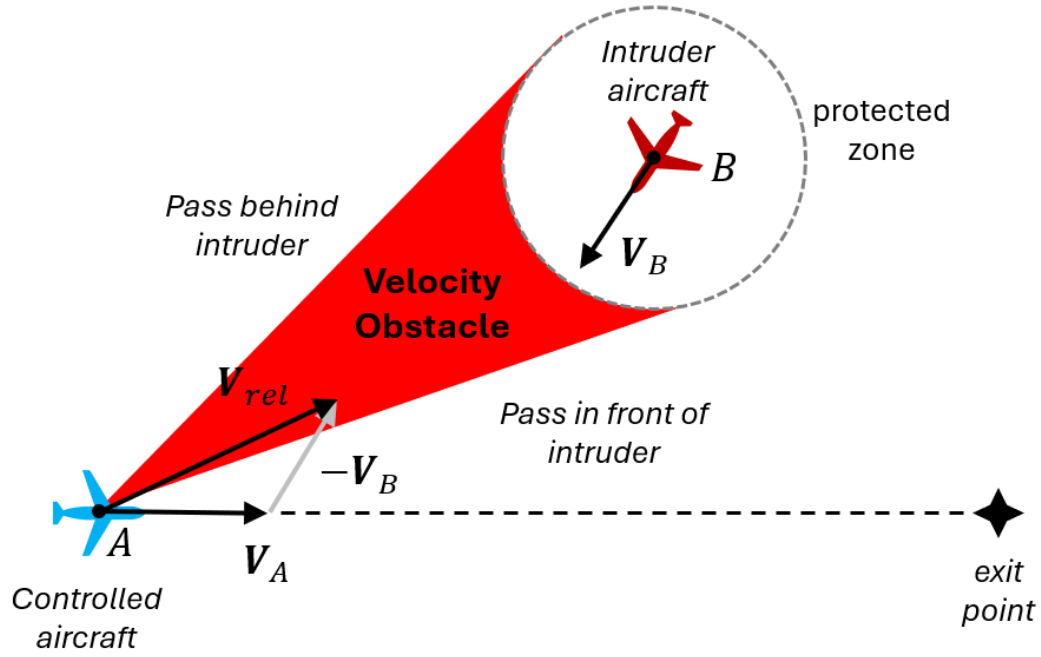
Human rating $r \in \{1,2,3,4,5\}$ mapped to scalar $R \in [-1,1]$:

$$\mathbf{w}_a \leftarrow \mathbf{w}_a + \alpha_h\left(R - Q(s)\right)\phi(s)$$

(α_h is feedback gain)

Original contributions: “expert” guidance

A rule-based Velocity Obstacle (VO) formulation is used to generate control actions that place the relative velocity of the controlled aircraft outside the collision cone (i.e., the velocity obstacle) induced by the intruder's protected zone, thereby providing guidance signals for the learning agent. As a rule, candidate maneuvers are further constrained to ensure a fixed *pass-behind* geometry, which is commonly aimed for by *expert* human controllers.



Q-function weight update:

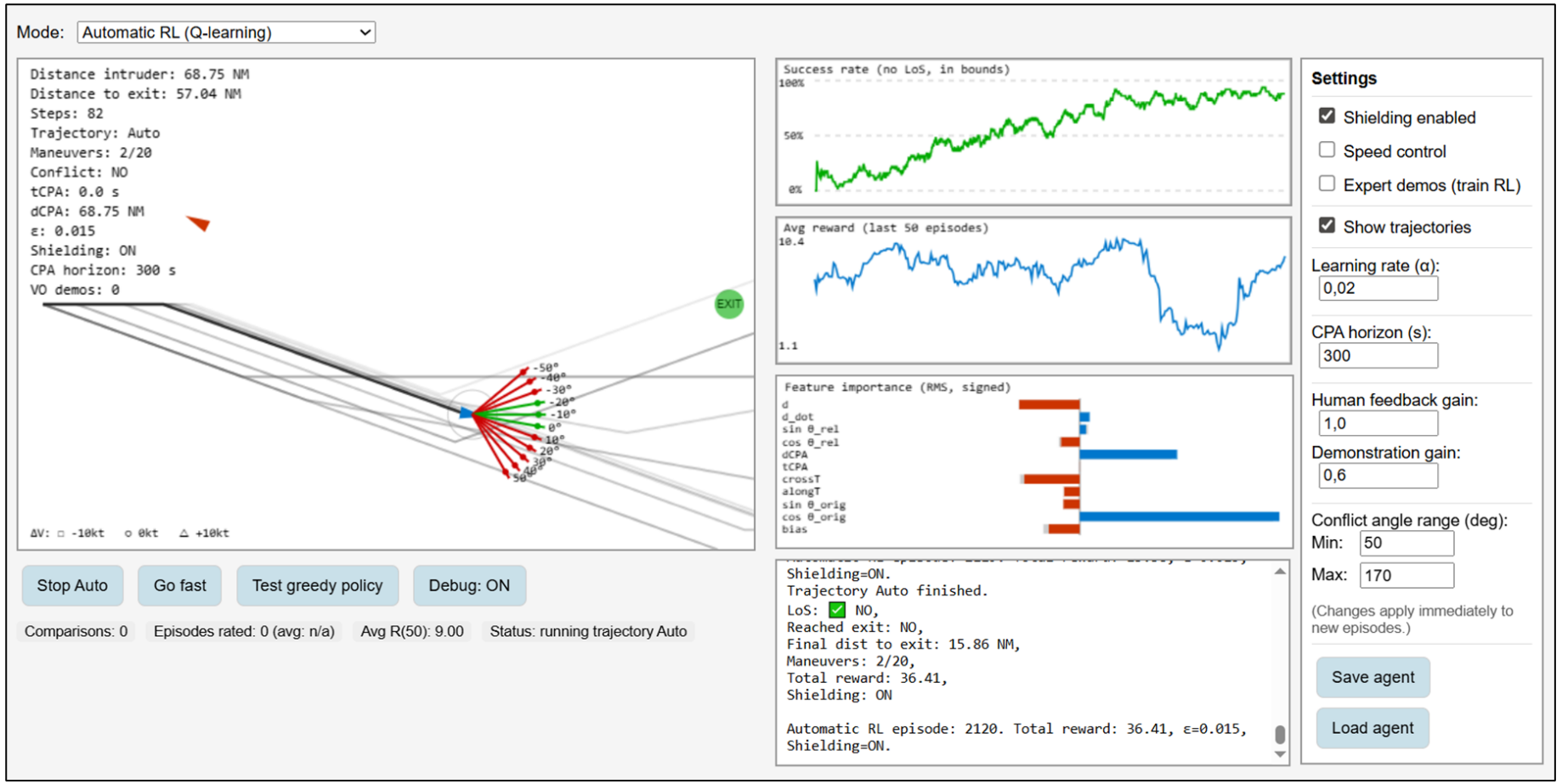
$$\mathbf{w}_{\mathbf{a}_t^E} \leftarrow \mathbf{w}_{\mathbf{a}_t^E} + \alpha_E \left(Q_E - Q\left(s_t, a_t^E\right)\right) \phi(s_t)$$

with:

$Q_E = \max_a Q(s,a) + m$ target to exceed the value of the best competing action by at least a margin gap m

$Q\left(s_t, a_t^E\right)$ current estimate of how good the “expert’s” action a_t^E is in state s

Original contributions: Interactive demonstrator



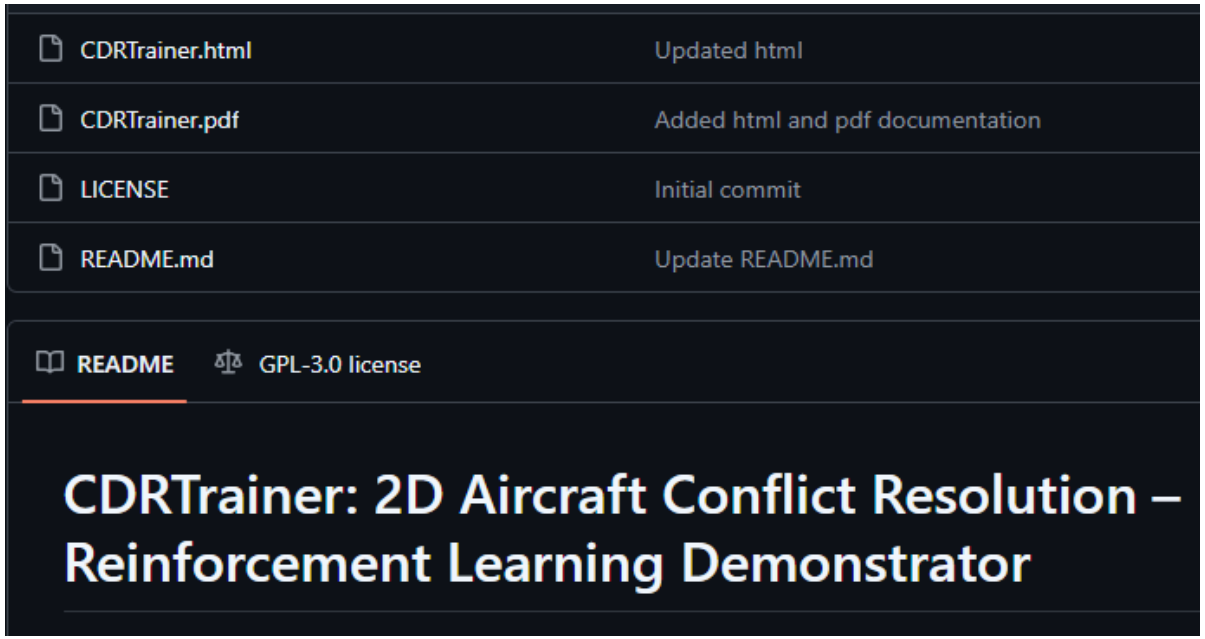
- Real-time monitoring of RL training process (trajectories, rewards, success)
- Live feature relevance plot during training for explainability
- Traceability log and debug view for inspecting RL decisions
- Switch between modes (human feedback and automatic RL)
- Modify settings to inspect impact on RL behavior and training process
- Expert demonstrations via well-known ATC best practice (“pass behind”)

Standalone JavaScript/HTML application (runs in the browser, no external libraries needed)

Overview of the repository

JavaScript/ HTML source

Documentation



edu.nl/phngn

Authors	Institution
Clark Borst	TUD
Giulia Leto	TUD

Interactive Preference Learning in ATM Sectorization

TUD

- Context
- Methodology
- Original contributions
- Overview of the repository

Real-time interactive learning

Motivation

Due to scarcity of labeled human data in many domains, develop an interactive solution assistant supporting operators in real time while learning preferences from domain expert annotations.

Developed Functions

Human-in-the-loop optimization framework via interactive evolutionary computation – using multi-objective adaptations of a Covariance Matrix Adaptation Evolutionary Strategies (CMS-ES) algorithm – allowing human operators to revise, accept, reject, nudge and steer optimizations in preferred directions, while a Kernel Density Estimation-based meta learner learns from human annotations.

Use cases

Applicable to Air Traffic Management (ATM) sectorization use case

Context: ATM use case on sectorization

Dynamic Airspace Sectorization (DAS): "a continuous restructuring of airspace sectors that ensures efficient allocation of scarce resources (e.g., Air Traffic Controllers) considering operational, economic and ecological constraints in both nominal and variable air traffic conditions." (Gerdes *et al.*, 2018)

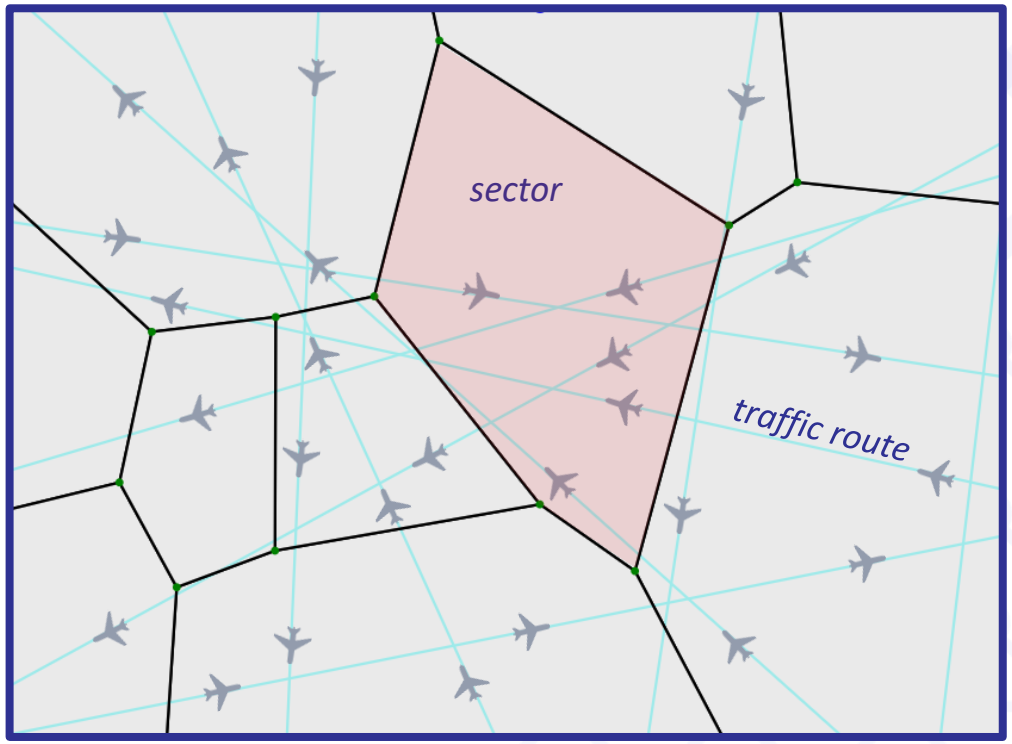
Human Role: flow and capacity manager

Objectives:

- Keep predicted controller workload within acceptable bounds (min/max)
- Balance predicted workload across sectors (low standard deviation)
- Minimize the number of handover points (coordination workload)

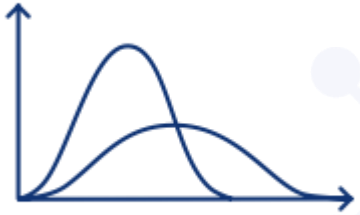
Operational constraints:

- Number of airspace sectors equal the number of available controllers
- Ensure convex airspace sectors
- Sectors should not be too small
- Avoid route crossings too close to sector boundaries
- Ensure sufficient route lengths within sectors
- Avoid route segments to coincide with sector vertices
- Avoid shallow crossing angles between routes and sector boundaries



Methodology: environment modelling

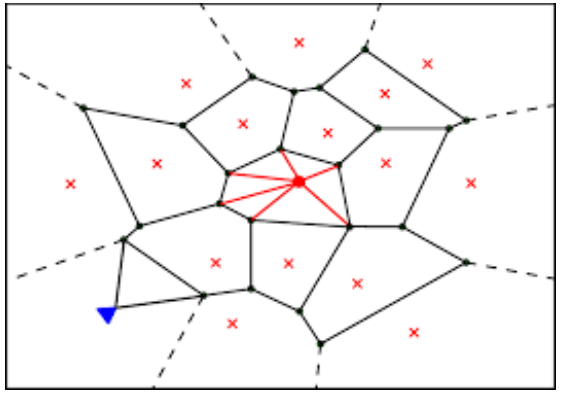
- **Trajectory generation:**
  - Interpolation of flight plan data (geometric calculations)
  - Historical flight data (distributions on FIR airspace entry/exit times)
  - Incorporate trajectory uncertainty (uncertainty distributions on position, arrival times and flight speed)



- **Workload prediction**: (weighted) sum of complexity (CX) factors
  - CX_1: Number of route crossings
  - CX_2: Number of coordination points (= route-airspace crossings)
  - CX_3: Number of predicted flights per sector, per time slice
  - CX_4: Number of predicted conflicts per sector, per time slice
  - …

$$WL = \sum w_i \cdot CX_i \qquad (w_i \text{ from literature})$$

- **Convex sector geometry**:
  - Voronoi partitioning
  - Fortune's algorithm (fast!)
  - Number of centers = number of available controllers



Methodology: interactive optimization

- **Multi-objective optimization**: minimize cost function $F(\boldsymbol{x}, t)$ using Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES) with random scalarization between two competing objectives:

$$F(\mathbf{x}, t) = J(\mathbf{x}) - \alpha(t)\, R(\mathbf{x}, t)$$

$$w_{\text{std}} f_{\text{std}} + w_{\text{handoffs}} f_{\text{handoffs}} + \lambda P(\mathbf{x})$$

$R(\boldsymbol{x}, t)$: learned preference reward

task constraints

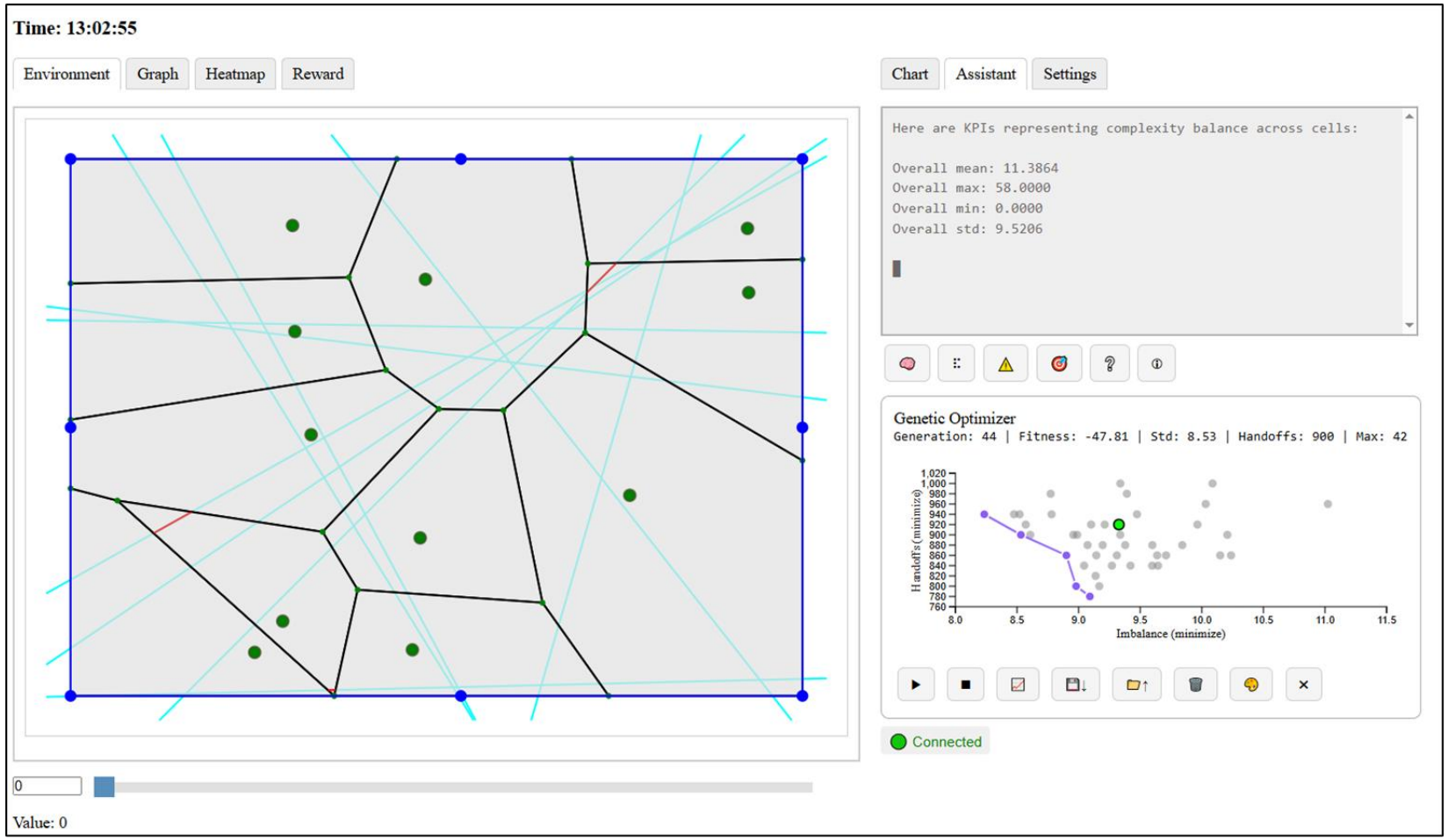
- **Human guidance**:
  - **Injection**: replace or bias the CMA-ES mean with user-provided centers;
  - **Preference** shaping: reward or penalize regions of the search space using **kernel model**; $R_{\text{kernel}}(\mathbf{x}) = \frac{1}{Z} \sum_i y_i \exp\left[-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right]$
  - **Restarting**: soft or hard reinitialization around a preferred layout.

- **Supervised meta-learning of kernel parameters**:
  - Dynamically adjust (using gradient) how strongly and how broadly human feedback shapes the optimization (reward) landscape;
  - Learn from historical feedback: more coherent → influence sharpens → faster convergence near regions aligned with human preference.

- **Continual learning**:
  - Preserve learned preferences between sessions for continuity and cumulative learning
  - Learn to generalize between sessions (and airspaces) as long as human feedback remains consistent

Original contributions: HMI

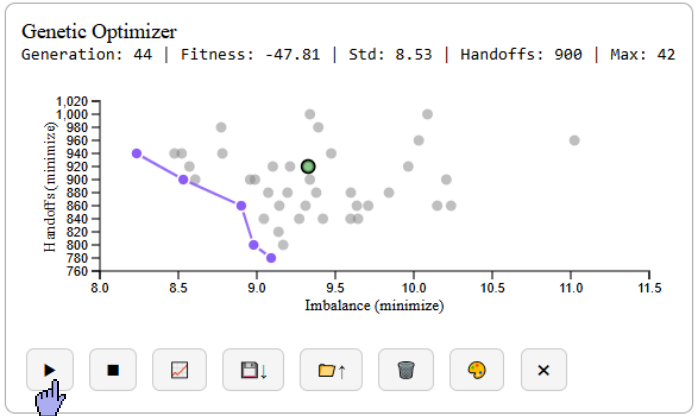
HMI for interactive optimization & preference learning



- Interactive Evolutionary Computing via CMA-ES
- Human steering of a multi-objective optimization problem
- Interactive Pareto Frontier to record human-preferred weights
- Understanding impact of hyperparameters & settings
- Observe and intervene optimization process step by step
- Record preferred solutions as labelled samples for meta-learning
- Standalone, fully integrated environment, enabling controlled human-in-the-loop experiments

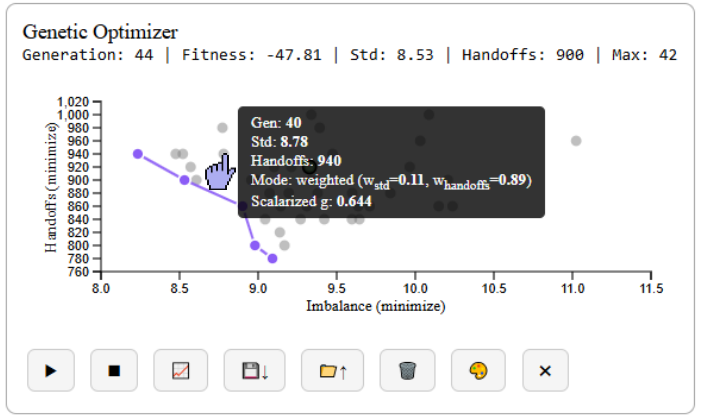
Original contributions: Interactive learning

1 Press ‘play’ to start optimization



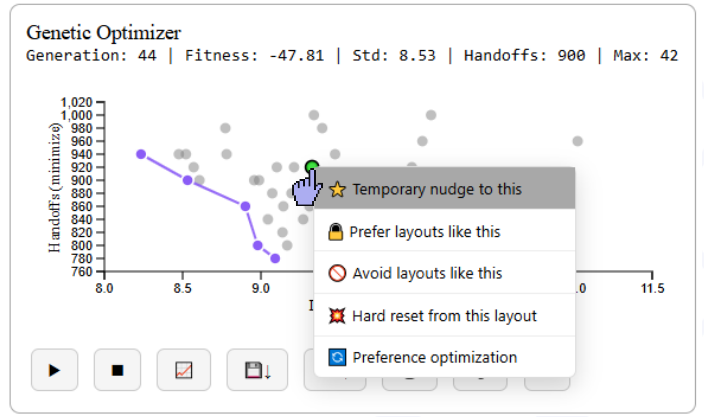
Observe CMA-ES generate solutions and dynamically forms a Pareto front. During CMA-ES optimization, the changing sectorization topology can be observed.

2 Inspect sectorization solutions



The process can be stopped at any time or automatically stops when solutions do not change over at least 10 generations. Hovering the mouse cursor over solutions reveals the objectives trade-off and weights. Clicking a point in the Pareto graph will display the corresponding sectorization configuration in the Environment. Users can **nudge** or **refine** solutions manually by dragging Voronoi centers.

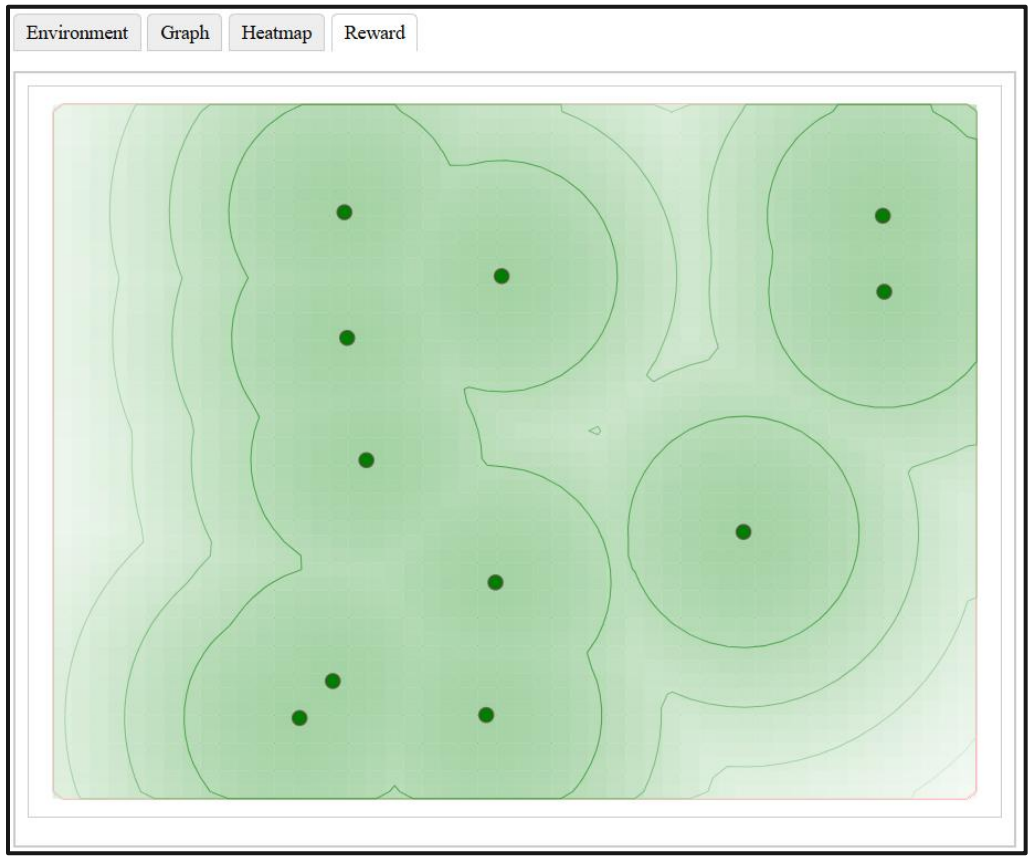
3 Annotate solutions



By right clicking Pareto points, solutions that are ***preferred*** or should ***be avoided*** can be annotated. The annotated solutions will be added as labelled samples to a preference learner.

Restarting CMA-ES will generate new solutions around the preferred sectorization configurations

Original contributions: Preference reward landscape



Preferences are directly mapped to preferred Voronoi center locations, enabling the construction of a reward landscape that highlights the regions toward which CMA-ES solutions are drawn.

Preferred solutions can be saved to and loaded from a JSON file, enabling air traffic controllers to access their personal profiles and/or retrieve favorable solutions from a shared database.

Clear preferences

Save preferred solutions

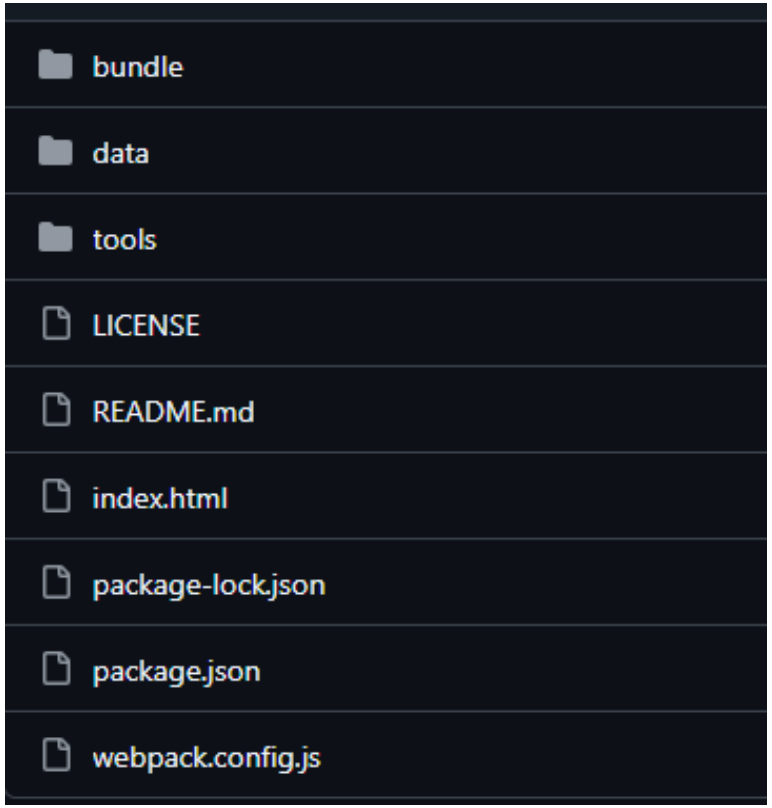
Load preferred solutions


```
{
  "profiles": {
    "controller_A": {
      "samples": [
        {
          "centers": [ ···
          ],
          "label": 1,
          "weights": {
            "std": 0.19608487033927302,
            "handoffs": 0.803915129660727
          },
          "std": 9.327041084131341,
          "handoffs": 920,
          "max": 53,
          "fitness": -56.752041084131335,
          "cost": 0.20056885634717167,
          "timeWindow": {
            "start": null,
            "end": null
          },
          "time": 103006.19999999925
        }
      ]
    }
  },
  "activeProfile": "controller_A",
  "comparisons": [],
  "config": {
    "USE_KERNEL_MODEL": true,
    "USE_RAMP_INFLUENCE": true,
    "USE_PAIRWISE": true,
    "USE_CMA_BLEND": true,
    "kernelSigma": 150,
    "prefWeight": 0.2,
    "prefWeightMax": 1,
    "prefRampRate": 0.02,
    "prefConsistency": 0
  }
}
```

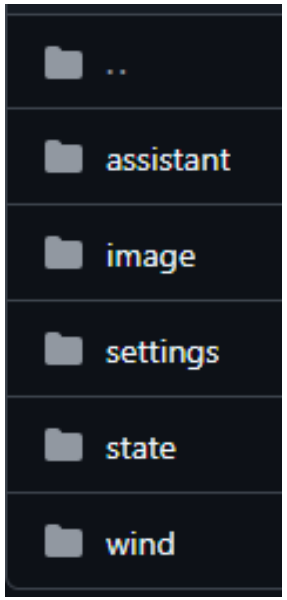
Overview of the repository

JavaScript source

Main HTML file to open in browser



Data files & settings:



edu.nl/k7jvb

Authors	Institution
Clark Borst	TUD
Giulia Leto	TUD

Human Learning Support

FHNW CLS & APS

Functions for Human Learning in Co-Learning Systems

Motivation

The developed “Human Learning Functions” actively supports the human through all phases of the cognitive learning cycle.

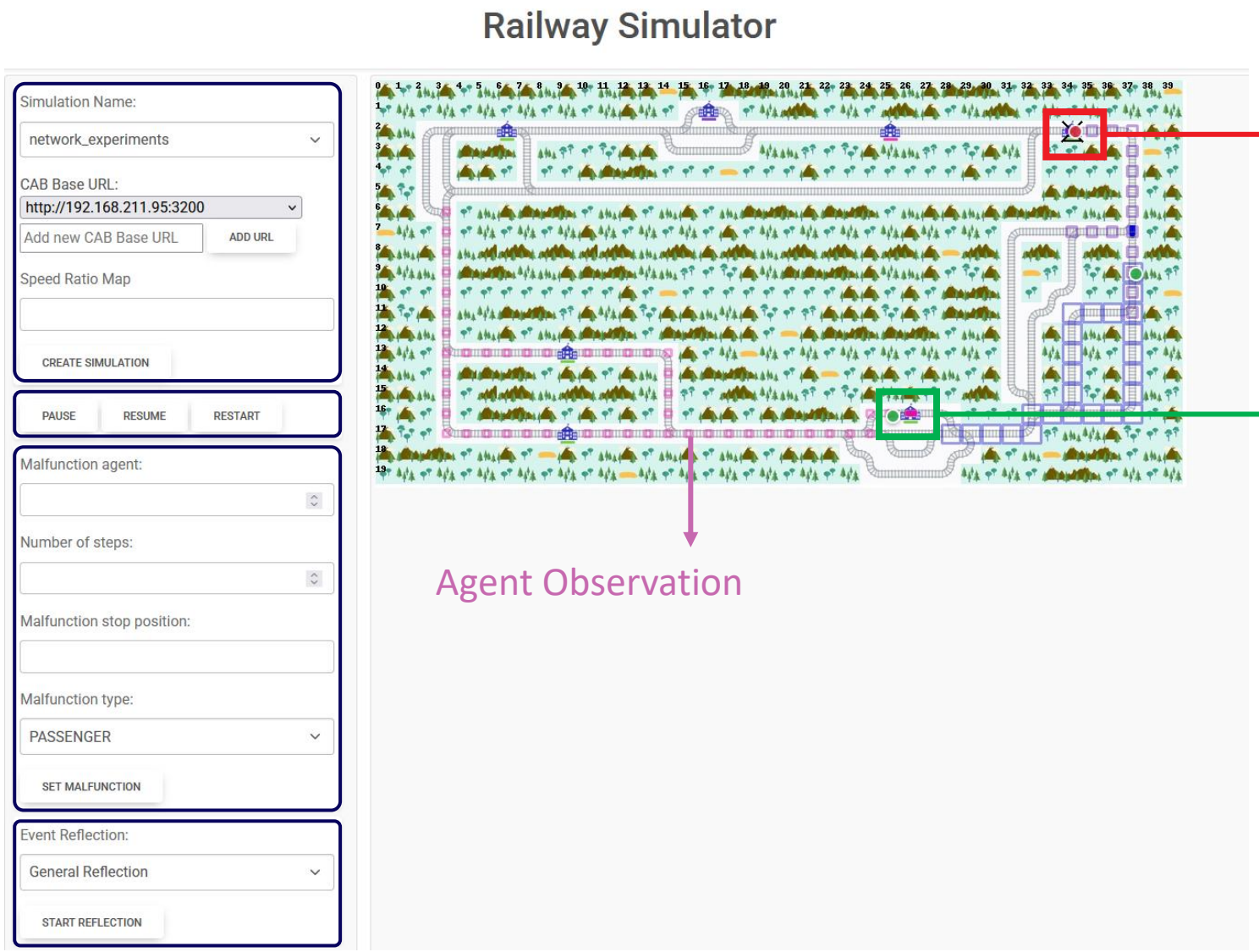
Developed Functions

Evaluative: during operations, solutions generated by AI agents and solutions suggested by the human are evaluated.

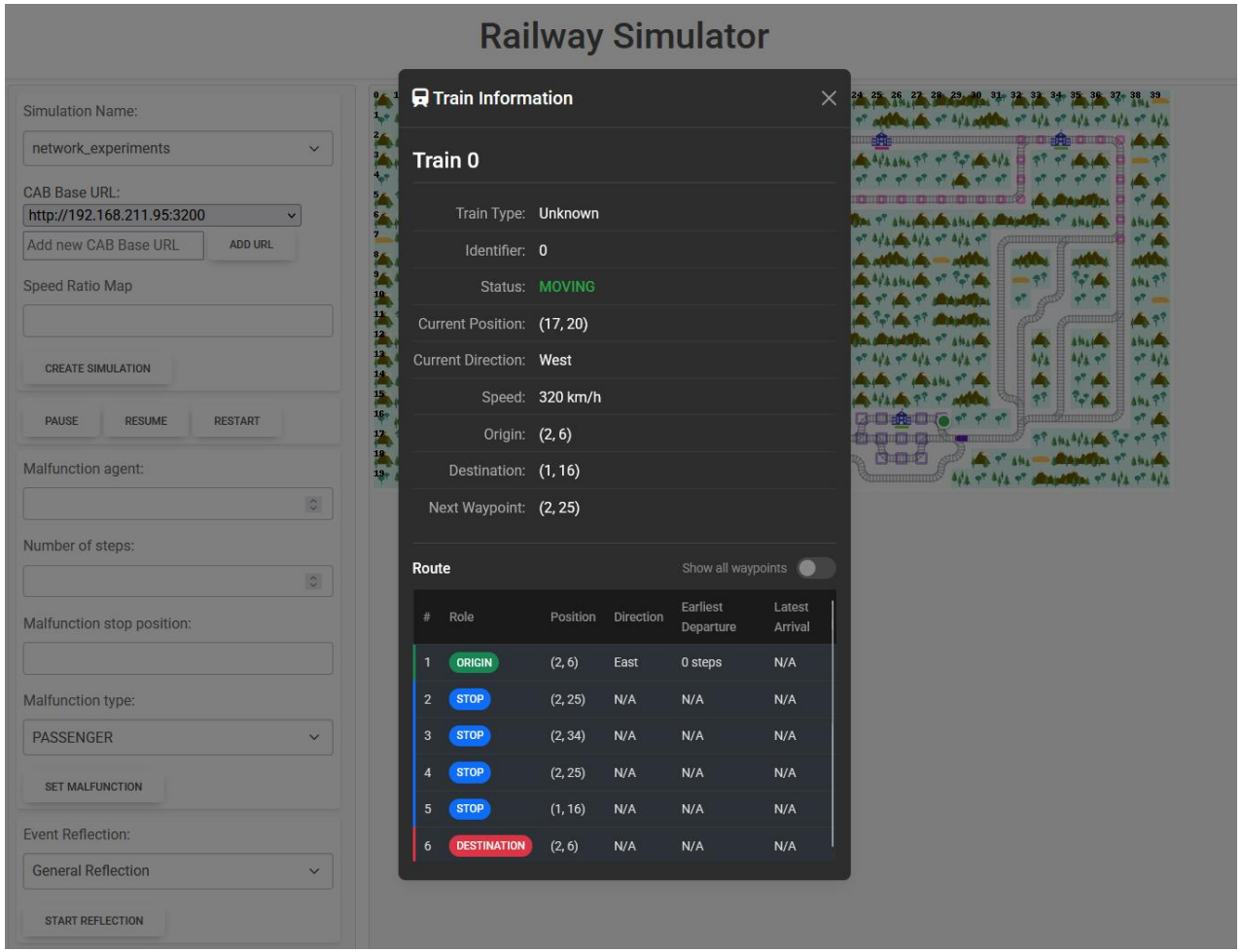
Reflection: after operations, active reflection is supported and logged, allowing for anonymized knowledge sharing.

Simulation: after operations, the human can revisit experienced situations or simulate fictive situations.

HMI Main Screen



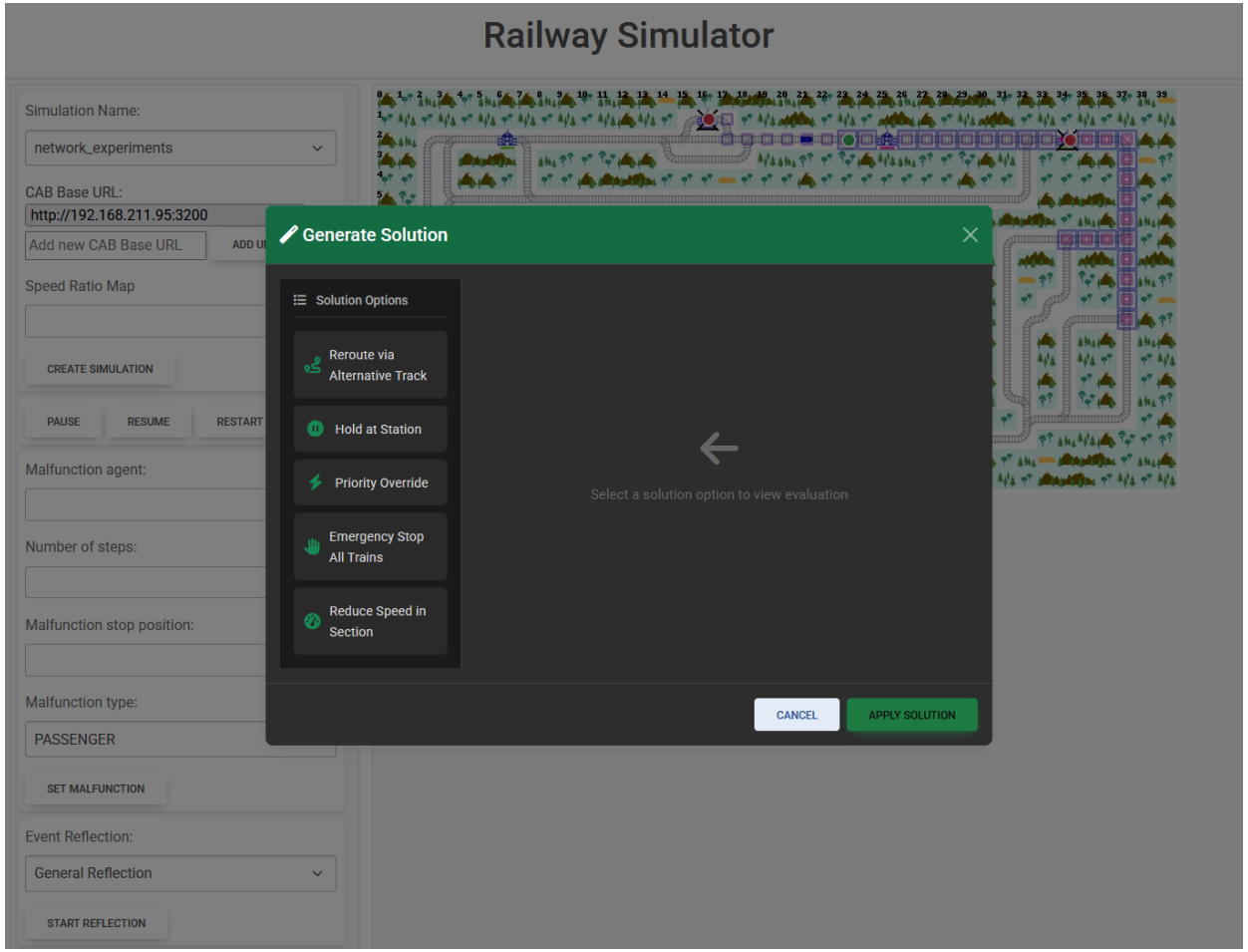
Train Information



Overview of Train Information:

- Train Type
- Identifier
- Status
- Current Position (grid coordinates)
- Current direction (cardinal)
- Speed (km/h)
- Origin, destination and next waypoint
- Planned route

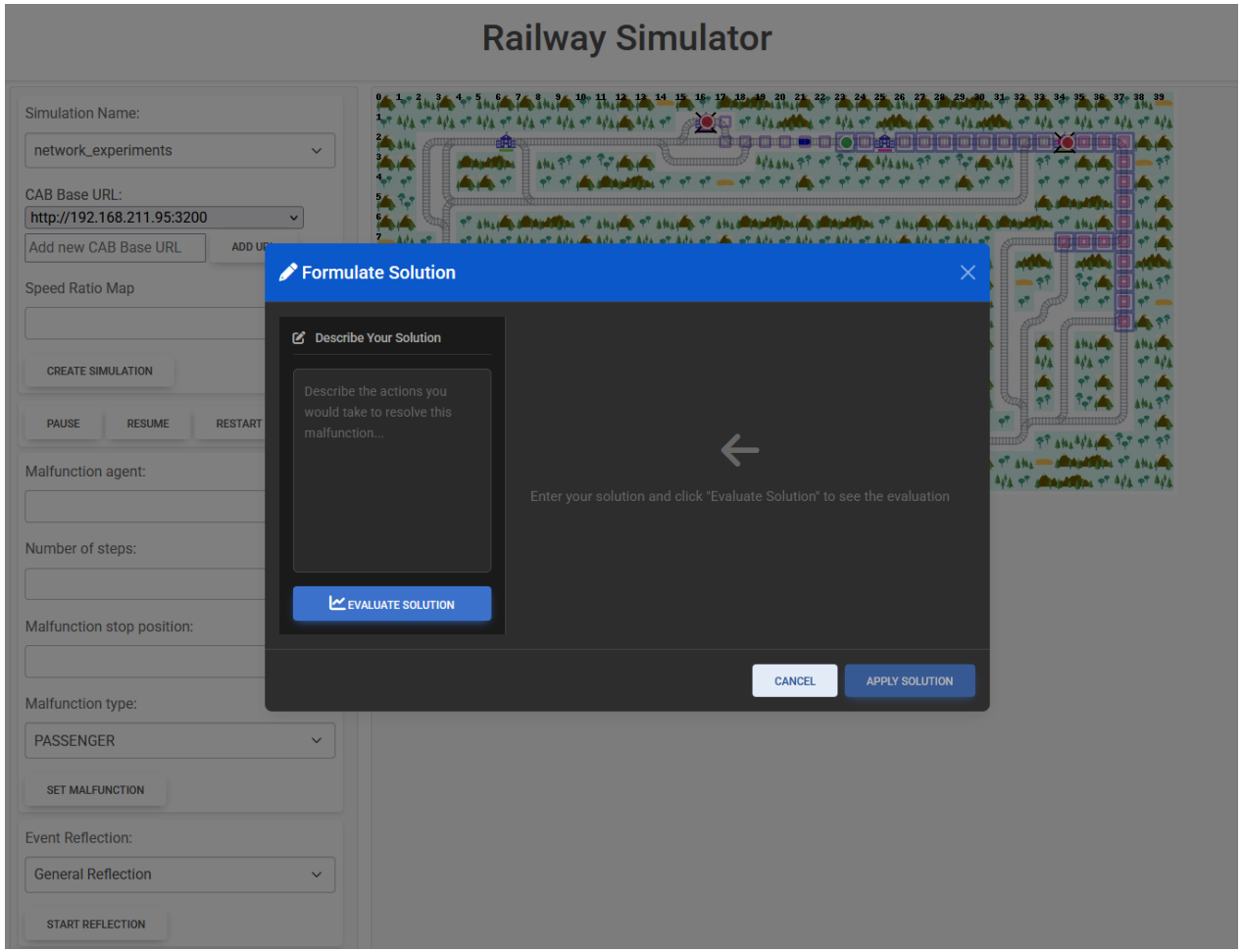
Solution Generation and Evaluation



Evaluation Criteria:

- → General risk assessment
- → Number of actions
- → Impacted trains
- → Current and expected delays for each impacted train after solution application.

Solution Formulation and Evaluation



Solution Input: Human agent can input a solution in natural language which is tokenized and used for the same evaluation as the generated solutions.

Repo Overview

HMI Extension branched from decouple-simulator-interactiveAI

- docs/postman_collection
- flatland_simulator
- resources
- .gitignore
- Dockerfile
- README.md
- docker-compose.yml
- requirements.txt
- run.py

→ **Main Source Code**

- experiment_configurations → **WoZ experiment set-up**
- networks → **environments and translation scripts**
- simulations → **simulations in standard form**
- src
- static → **front-end interface**
- templates
- views
- app.py

Repository-Github-Link

Authors	Institution
Julia Usher	FHNW
Manuel Renold	FHNW
Toni Wäfler	FHNW
Samira Hamouche	FHNW
Nerissa Dettling	FHNW
Roman Ließner	DB
Manuel Meier	FLATLAND

T3.4 – Integrated Autonomous AI-driven Decision Systems

Leader: FHNW (CLS)

Contributors: FHNW (APS), SBB, DB, FLAT

Summary

1. PPO & IMPALA Implementations for Flatland
2. Graph Translation of flatland environment
3. Communication via Negotiation Proxy for conflict management

Contribution: PPO & IMPALA for Flatland

Motivation

Dedicated Flatland implementations of state-of-the-art algorithms which allow for modular and customizable training and parallelization.

Developed Functions

1. **Non-parallelized PPO**
2. **Synchronous PPO**
3. **Asynchronous PPO (IMPALA)**

Contribution: PPO & IMPALA for Flatland

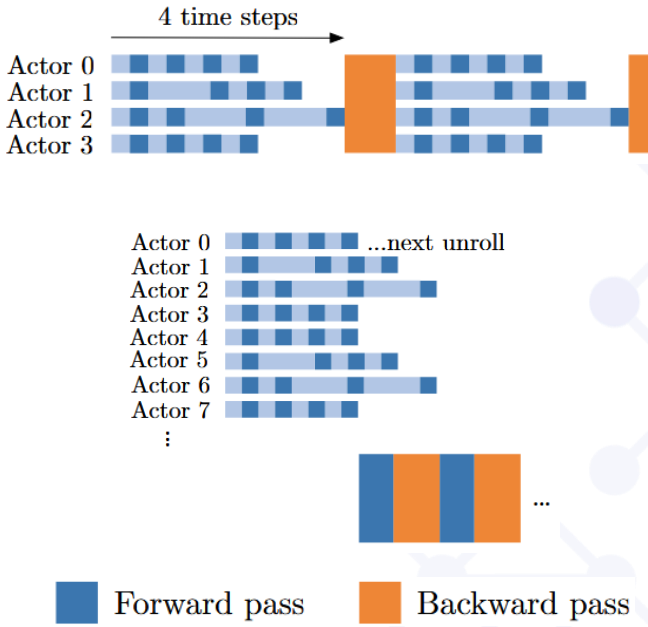
To accelerate training times, two parallelized training pipelines for RL Agents in Flatland were developed, based on Proximal Policy Optimization.

1. Synchronous PPO

→ Networks are updated once all workers have finished generating rollouts.

2. Asynchronous PPO (IMPALA)

→ Networks are updated continuously once sufficient rollouts have been generated.



Espeholt et al. (2018)

Contribution: FlatlandMultiDiGraph

Motivation

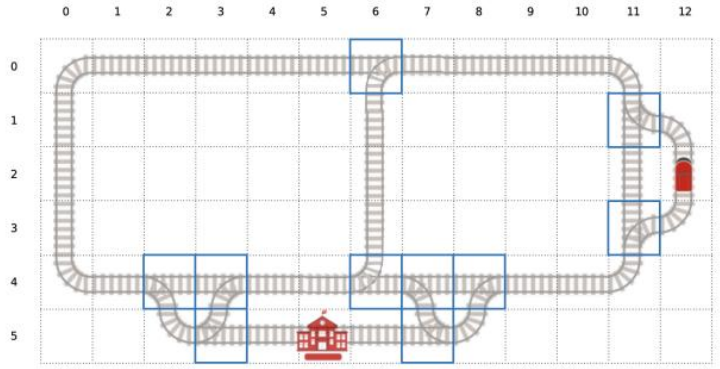
Translation of Flatland environment to a graph network, allowing generalizability of the solutions and quick conflict recognition

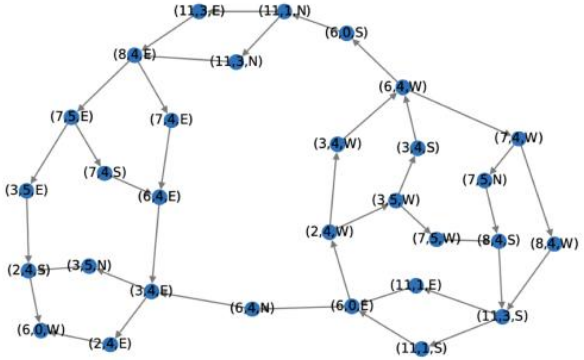
Developed Functions

Multi-directional graph translation of flatland environment based on existing solution by SBB

Graph Representation

Extension of existing work by SBB: FlatlandGraphBuilder





Existing System: nodes are categorized by their location and the cardinal direction from which they are entered

- Up to 4 nodes per switch
- Geographically “close“ locations are far apart and not linked in the graph

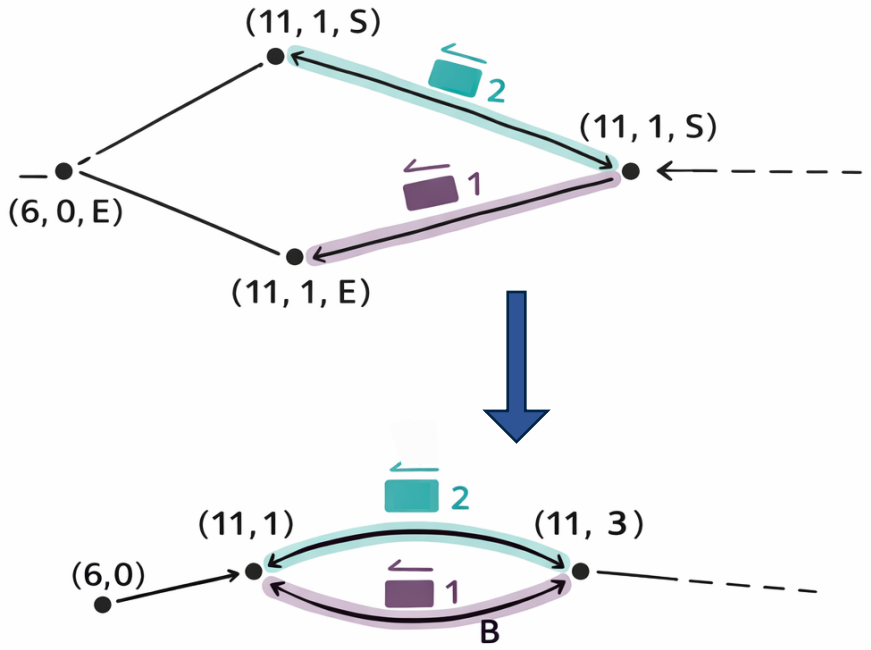
Issue: messages between geographically close agents must propagate through geographically “irrelevant” nodes

Extension: MultiDiGraph

New Solution: Multi-Di Graph considers only 1 node per switch, with multiple edges (tracks / paths) between nodes.

- Greatly simplifies the graph
- Geographically relevant clusters remain together

Use: the MultiDiGraph allows for faster and more efficient conflict recognition, which initiates **conflict negotiation** via the negotiation proxy.



Contribution: Negotiation Proxy

Motivation

Translation of Flatland environment to a graph network, allowing generalizability of the solutions and quick conflict recognition

Developed Functions

Multi-directional graph translation of flatland environment based on existing solution by SBB

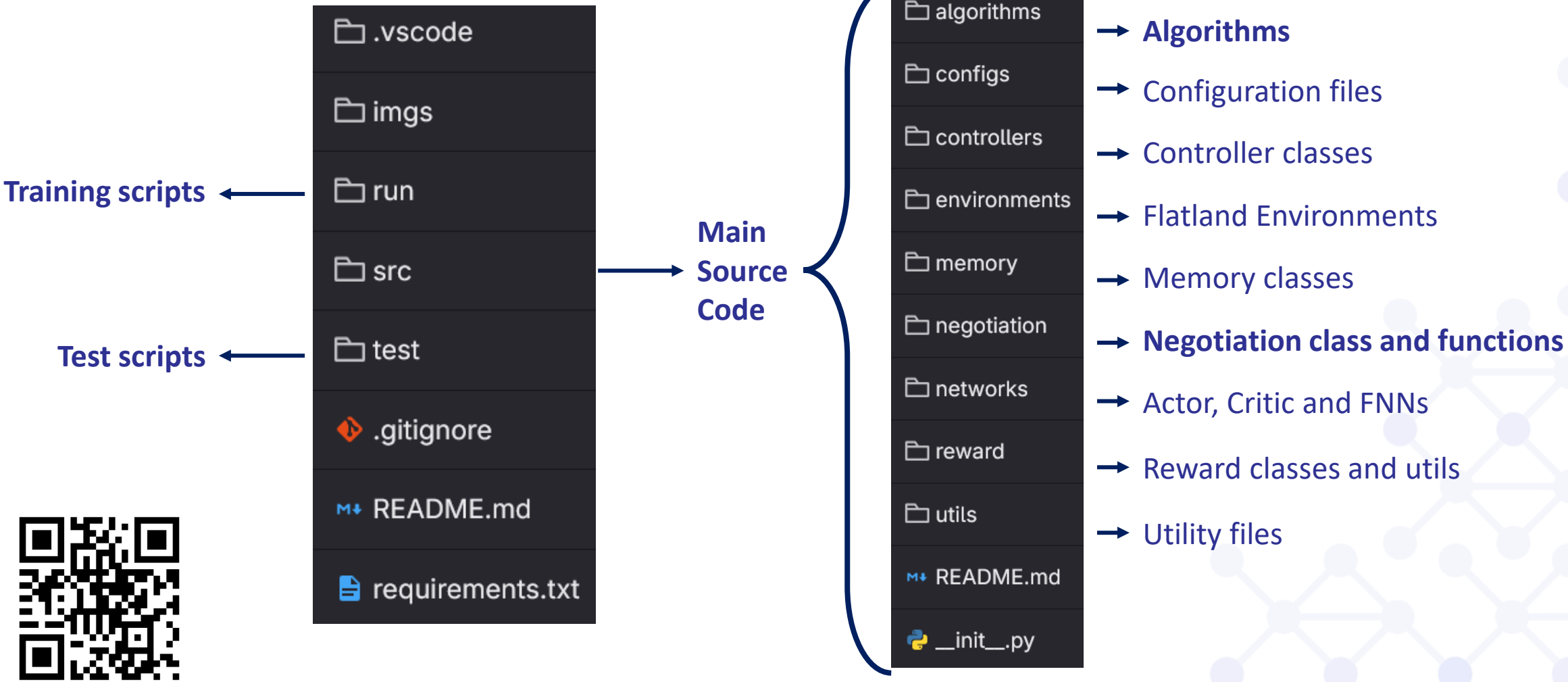
Contribution: Conflict Negotiation

- When messages propagate to the same node → **Negotiation Proxy** initialized to solve the conflict.

→ Using auction-based conflict resolution using various system impacts (i.e., delay cost, safety margin, congestion risk, etc.), a transparent and interpretable trace of conflict solving is developed.

→ Implementation of the negotiation system is work in progress.

Repository Overview



Repository-Github-Link

Authors	Institution
Julia Usher	FHNW
Manuel Renold	FHNW
Toni Wäfler	FHNW
Samira Hamouche	FHNW
Nerissa Dettling	FHNW
Roman Ließner	DB
Manuel Meier	FLATLAND

AI4REALNET

AI4REALNET has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101119527 and from the Swiss State Secretariat for Education, Research and Innovation

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.