Formation

Collective Intelligence - Second Assignment

Contributors: Márk Róbert Baricz, Richard Sipos

Key elements from previous semester

- PPO Clip as the main algorithm
- CTDE paradigm
- Hydra config setup, logging to WandB
- Visualizer script to export GIFs (improvement needed)

Main features introduced in our version

- SDF interface with support for 3 shapes (circle, polygon, star)
- Observation and reward redesign
- Assignment strategies (hungarian, greedy)
- Multi-shape scenes
- Dynamic reconfiguration mid-episode
- Metrics (boundary error, uniformity, collision rate)
- Improved visualizer script
- Extended and fixed unit tests
- Hydra support for sweeps
- Added multirun support
- Ablation & Analysis feature
- Created comparison charts for ablation run
- Dockerization fixes
- Descriptive README file

Shape Model and Scenarios

SDF interface

- although there is support for 3 shapes (circle, polygon, star), there are only two shape families (Circle, Polygon), because Polygon class is generalized in a way that it can compute the signed distance for non-convex polygons also => basically defining a polygon shape is done by defining the coordinates of the vertices, and those vertices can also form non-convex shapes too
- signed_distance() method of the shapes returns the signed distance, negative inside, zero on boundary

Multi-shape scenes

- there is support in the config files to define multiple shapes in one scene
- shapes can also overlap each other
- it should be specified how many agents should target the shapes (e.g. 4 agents to shape A, 6 agents to shape B)

Dynamic Reconfiguration

- in the config file the shapes can also be changed by defining the reconfiguration_step and the reconfiguration_shape
- the new shape can be defined in a similar way, just nested under a specific key
- agents automatically move towards the new shape when it has changed
- dynamic reconfiguration also supports multi-shape scenes (e.g. a variation of a multi-shape scene changes to another variation of it/or with totally new shapes)

Assignment strategies

Assignment strategies

- there is support for two different assignment strategies:
    - hungarian: each agent gets a unique target position, so they will evenly occupy the perimeter of the targeted shape; the computation of the target positions happens in an optimized way
    - greedy: each agent targets the closest target position to his current/starting position -> multiple agents can target the same position

Observation and reward redesign

Observation and reward redesign

- the signed distance and the target positions are also taken into consideration in the case of observations and the computation of the reward
- there is a reward given based on the signed distance (exp(-5*ϕ(x_i)**2), but also based on the distance to the target position (exp(-2*dist_to_target(x_i)**2)
- future improvement ideas:
    - penalty for high velocity when close to target => reduce “jittering”
    - tweaking with the alpha values of the rewards above => possible achievement of faster convergence
- failed attempts:
    - there has been an issue of getting infinity values, causing the calculations to “deteriorate” the done progress, the final model ending to contain NaN weights => solution: safety check both in calculations and while training (skipping epochs where we get wrong weight updates)

visualize.py

visualize.py

- the visualizer script also needed an update for a better experience of development and testing
- on the exported GIF now there is also shown the target shape(s) (with dashed lines) and the target positions (with crosses)
- some export settings have also been changed, so now the agents move “faster” and their movement is more smooth, caused by increasing the fps
- there was an error in the initial version (from previous semester) that the processed location of the agents wasn’t the real location, so the exported GIFs never displayed the true movement of the agents

Metrics

Metrics

- we introduced three metrics to help evaluate the trained model:
  - **Boundary Error:** Measures how close agents are to the target shape boundary using the signed distance (SDF)
  - **Uniformity:** Measures how evenly agents are distributed along the target shape
  - **Collision Rate:** Measures how often agents collide with each other
- the metrics are printed to the CLI and logged to W&B after each training run
- the metrics for further analysis of the models, for example ablation studies
- a known limitation is that episodes are capped at 500 steps for performance reasons

Multirun

Multirun

- we introduced a multirun feature that allows training multiple models in a single execution
- we added two Hydra sweep configuration files:
  - sweep_lr: sweeps the learning rate
  - sweep_assign: sweeps the assignment strategy
- to improve storage and space management, we implemented a dedicated multirun directory with a clear folder structure for organizing outputs (date + time)
- finally, we introduced seed parameters to ensure experiment reproducibility

Ablation & Analysis

Ablation & Analysis process

- to run the ablation analysis, you first need to configure and execute a multirun (you can use sweep_assign and/or sweep_lr)
- the analysis script then aggregates results across seeds and prints the mean and standard deviation for each configuration among the models trained in the same multirun
- you must provide a sweep_id to the ablation analyzer so it can filter and evaluate only the models produced within that specific multirun
- finally, the analyzer generates a CSV file containing the key properties and metrics of the analyzed models

Example of generated CSV file row

Run	Learning Rate	Seed	Boundary Error (mean)	Boundary Error (max)	Uniformity (coeff)	Collision Rate
run-20251220_232632-894dzw7w	0.01	1	3.520	6.937	0.803	0.352

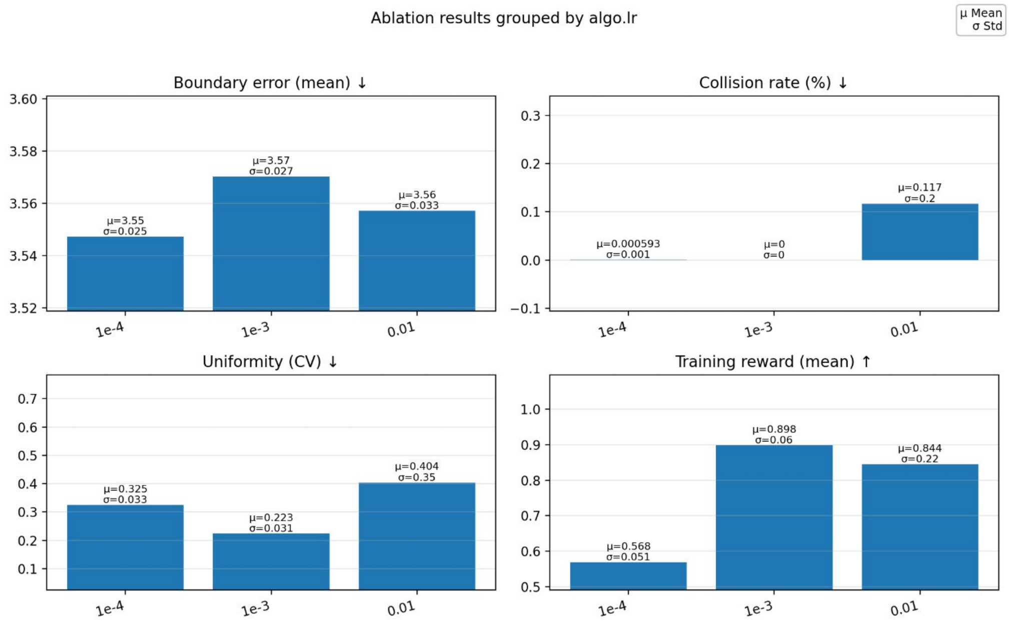
Chart creation

Chart Generation for ablation

- we store the aggregated ablation results
- the /scripts/plot_runs_csv.py script generates charts from the CSV file produced by analyze_ablations.py
- we aggregate metrics across different seed runs using the provided where filter, and then compare the aggregated configurations visually in the resulting charts
- a possible future improvement is to introduce other chart types and make this code reusable for other features as well

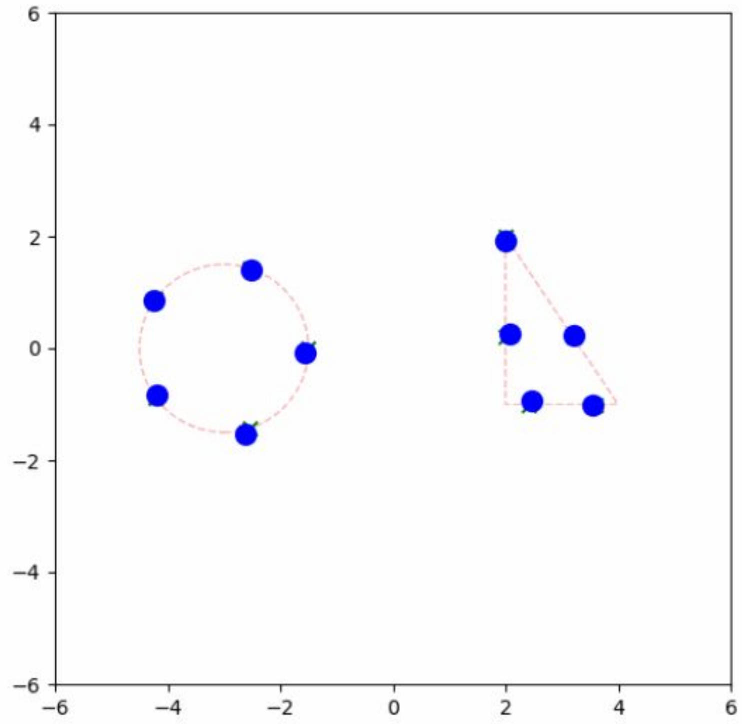
Ablation run chart comparison

- our chart compares the three core metrics and the training reward
- each column value is computed from the mean and standard deviation of runs that share the same configuration but use different seeds
- all charts are generated from this previously aggregated data.



Generated GIF

- generated by running the default_exp.yaml experiment
- the GIF is included in the repository as formation.gif file



Thank you!

Contributors:

- Márk Róbert Baricz
- Richard Sipos

More information about the project can be found in the README file.

https://github.com/elte-collective-intelligence/student-formation/tree/2025/26/1