Learning Classifier Systems in non-static collaborative multi-agent scenarios

Outline

- Basic Problem Scenario (Grid, Agents)
- Agents
- Basic Classifier System Operation
- Example Classifiers
- Classifier generation
- Movement of Goal Agent
- Reward (single-/multistep, sharing between agents)
- Communication and Organization
- Tests, Goal of thesis

Scenario

- 100x100 Grid (Torus)
- Obstacles
- Variable number of agents
- Limited sight range
- Moving goal agent

Agents

- Limited local communcation
- Limited sight range (4 directions (conic), binary sensors (agent in sight, goal agent in sight)
- 6 actions (4 directions, random direction, no movement)
- Optimization: Only one bit for the goal agent, position of other agents are relative (absolute directions do not matter)
- => 5 bits, one action byte (0..5)
- Maybe introduce later: Distance, state of the agent

Basic Classifier System Operation

- Get input, choose matching action, get reward.
- Store acquired knowledge: How well did some action perform given a specific input? ⇨ Rule base
- Classifier: Structure to store input, action and fitness.
- Simple approach: Binary code, avg. reward as fitness.
- Use fitness to decide between rules in case more than one classifier matches (because of wildcards)
- Learning: Generate new classifiers (GA; recombination, mutation), evaluate classifiers → update fitness.

Generalisation: Wildcards

- Efficiency: Keep rule base small.
- Avoid redundant information.
- Example: Multiplexer
  100010:1 and 101111:1: Different input, same output.
- Use a third symbol # as "don't care" / wildcard:
  10##1#:1 covers 8 different inputs and gives always correct output.
- Classifiers can be subsumed, size of population can be reduced.
- Problem: Make sure no information is lost (relevant for more difficult problems).

Example matchings

- 1.1010: Goal agent in sight, an agent is in the same direction and a second agent is in the opposite direction
- 0.1111: There are agents in all four different directions
- 0.0111: There is only one agent (classifier matches same situations as classifiers 0.1011, 0.1101 and 0.1110)
- 0.00*0: Up to one agent is in sight

Example classifiers

- 0.1101 => 2: Move into the direction where there is no agent in sight (classifier is equal to 0.1110 => 3, 0.0111 => 0 and 0.1011 => 1)
- 1.1000 => 5: There is an agent in the same direction as the goal agent, do not move towards the goal agent (5: No action)
- 1.1101 => 2: Move into the direction where there is no agent in sight (classifier applies to four situations, where the goal agent is north, east, south or west)
- #.#### => 6: move randomly on the field

Use what you know or learn something new?

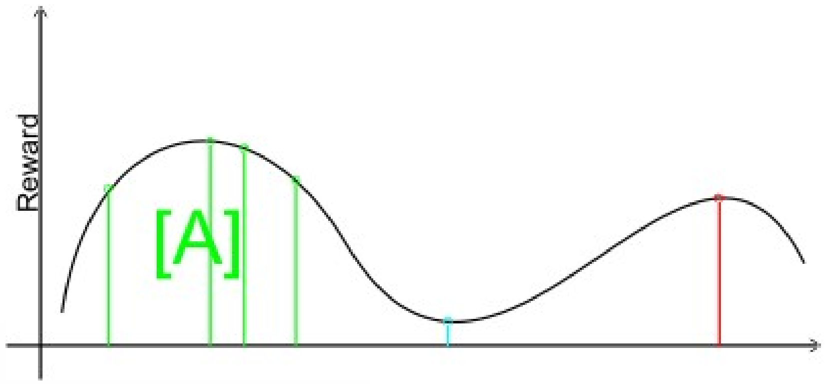
- Exploration vs. Exploitation: Learning implies wrong or at least non-optimal actions.
- **Exploitation** of acquired knowledge: Always use classifier with highest fitness value.
  - Deterministic behaviour,
  - evaluation of (few) classifiers becomes more reliable,
  - no improvements of knowledge.
- **Exploration** of new possibilities: Generate new classifiers, choose classifier randomly.
  - Enhance knowledge about problem,
  - performance worse than optimum,
  - dangerous / illegal actions?

Algorithmic description

- Select all classifiers in the population with condition matching input → **Match set** [M].
- Choose an action present in [M]: For every distinct action present, compute average fitness value of classifiers with that action.
  - Exploitation: Choose action with highest fitness value.
  - Exploration: Choose action randomly.
- All classifiers in [M] with chosen action → **Action Set** [A].
- Compute **reward**: External reward + highest discounted average fitness value in [M].
- Update **fitness** values of classifiers in previous action set $[A]_{-1}$ with this reward.

Classifier generation

- Two sources for new classifier:
  - **Genetic Algorithm**: Chooses two members of Action Set (probability proportional to fitness value) → two new classifiers (crossover). Offspring subject to mutation.
    Niche-GA: Recombination of similar classifiers; mutation to cover entire problem space.
  - **Covering**: No matching classifier available → generate classifier with matching condition and random action.



Goal agent

- No movement => Problem is equal to maze problem
- Predictable movement => Problem is similar to maze problem
- More interesting: Unpredictable moving goal agent, neither single-step nor multi-step solution applies
- Addition: Goal agent moves faster than individual agents, allows strategy stay & observe instead of constantly following the goal agent (goal agent can show up anywhere, covering an area not covered by other agents is better)

Reward (single-step)

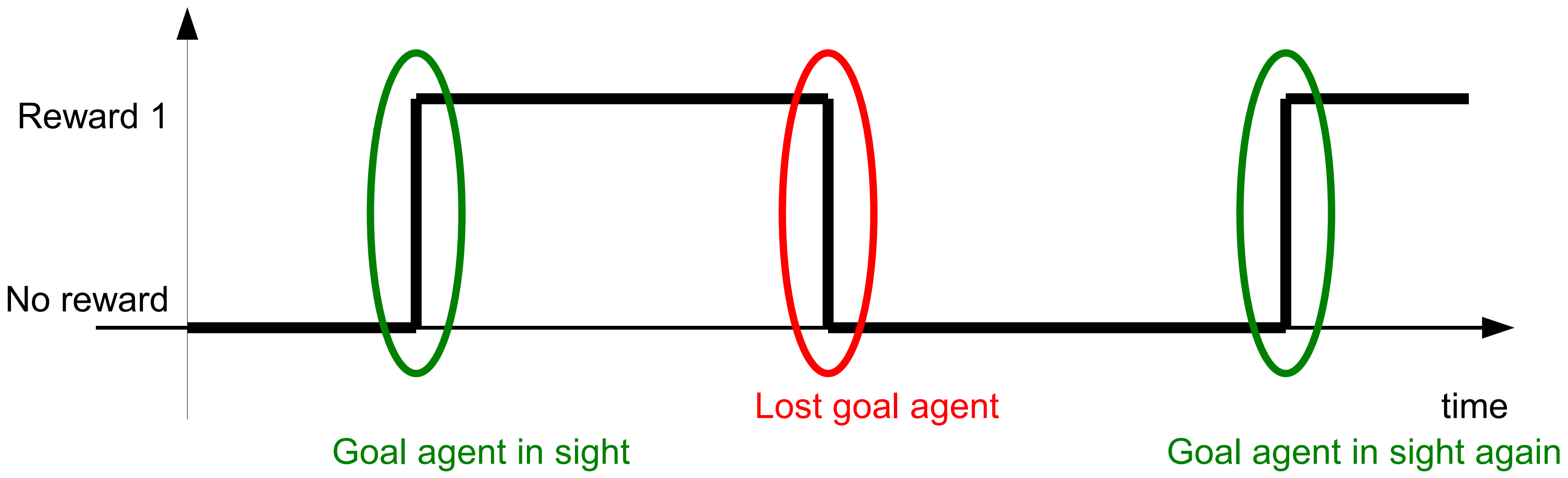
- Singlestep: We have to know after each action how good that action was
- Impossible to know except when using local heuristics (e.g. distance to goal agent and other agents, covered area etc.)
- When using local heuristics all we get with LCS is an adaption to the heuristic, optimal rule set can directly be computated out of the heuristic

Reward (multi-step)

- Whenever we reach the goal cell we assume that all actions contributed to the solution (favoring shorter solutions)
- Not applicable in this scenario because goal cell is moving, environment is non-static and actually there is no goal but a constant surveillance problem

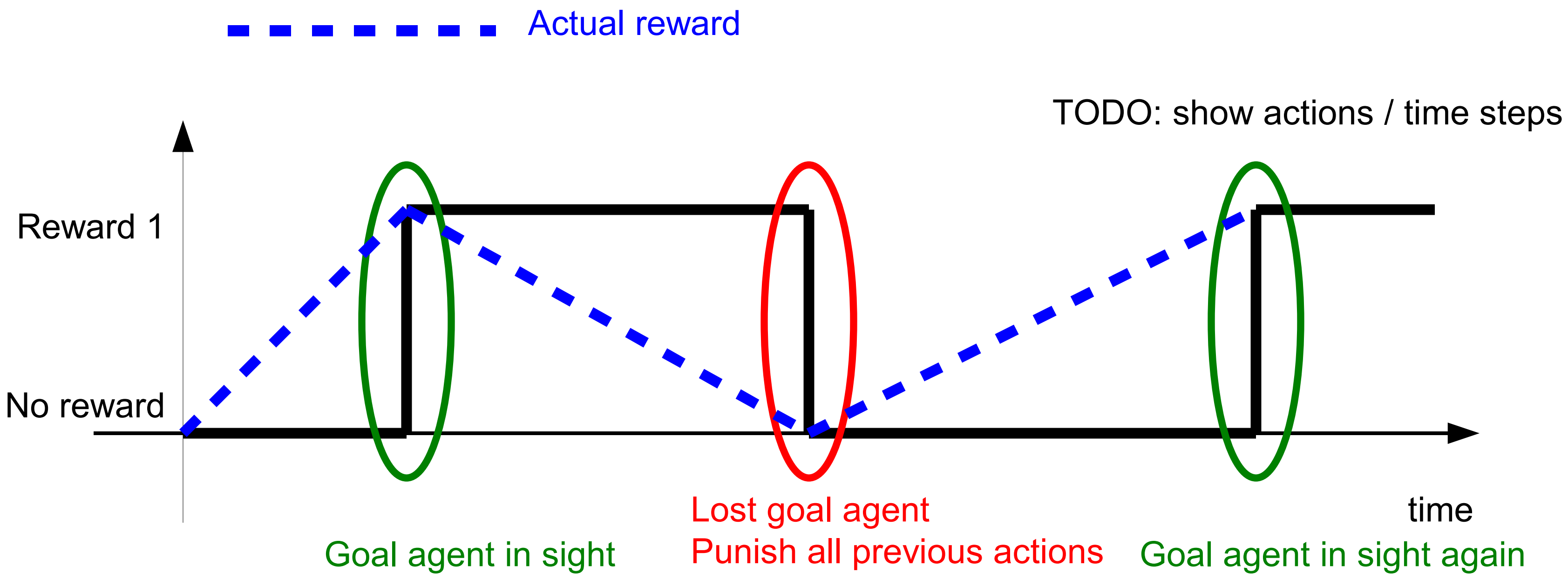
Reward in this scenario

- Usual distribution of the reward in one run is characterized by long episodes of 0 (goal agent not in range) and 1 (goal agent in range)
- Relevant events are not that we have the goal agent in range but if we gain or loose sight to the goal agent



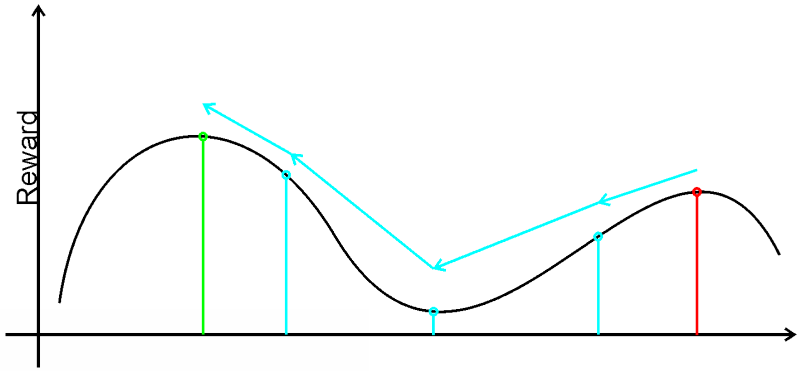
Share of reward between classifiers

- It takes less effort to keep an goal agent in sight than to find it in the first place
- Waiting in one place (where the goal agent currently isn't) should be punished


Actual reward
TODO: show actions / time steps
Reward 1
No reward
Lost goal agent
Punish all previous actions
time
Goal agent in sight
Goal agent in sight again

Average reward: Enough for optimal performance?

- Classifiers with low fitness are removed from population.
- Low-reward actions might be necessary to reach global optimum, but are removed due to low fitness.
- Very general classifiers might get higher average reward than more specific classifiers with actually better action.
- Extension: Fitness for reproduction and deletion based on **accuracy of prediction** → XCS (Wilson).



Share of reward between agents

- Goal agent movement basically is random, having e.g. 4 agents cover an non-overlapping area will cause a random agent's action classifier set get rewarded while the other 3 agent's action classifier set get penalized
- If agents are related and/or have similar classifier sets all agents' action classifier sets should be rewarded
- => Local populations form a group, if one agent sees the goal agent all agents get the reward
- Focus of research: Resulting homogenity in the population
- Balance between egoism and altruism (TODO)

Communication and Organization

- No central organisatory unit, no global communcation
- Limited local communication
- Succesful rules would spread through the population
- Counter-measure: rules are only exchanged when making contact with other agents in order to create local populations binding their fitness together
- Still focus of research

Goal

- Test of effectivity compared to common approaches (AI, randomized algorithms, single step LCS)
- Test of cooperation, with and without rule exchange
- Test of global rules (OCS)
- Test of reaction of the system to changes in the environment (obstacles, different goal agent movement pattern, introducing new agents / removing agents)

Thank you for your attention