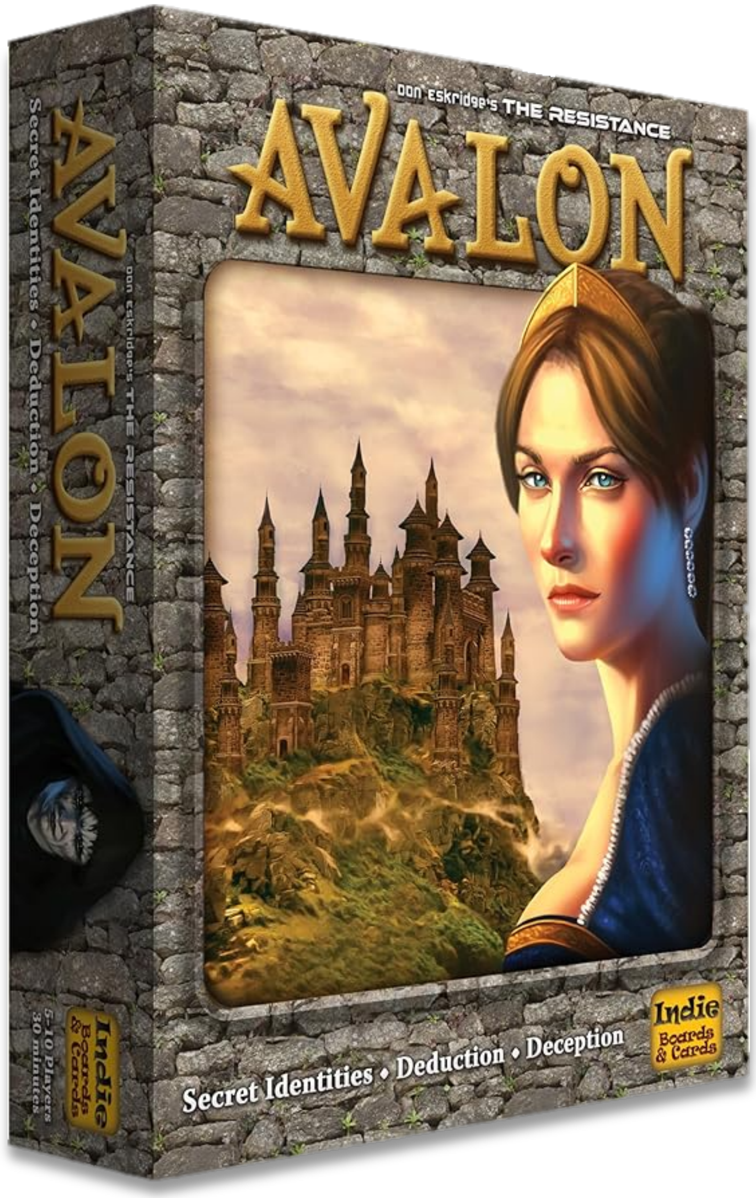
Strategist

Self-improvement of LLM Decision Making via Bi-Level Tree Search

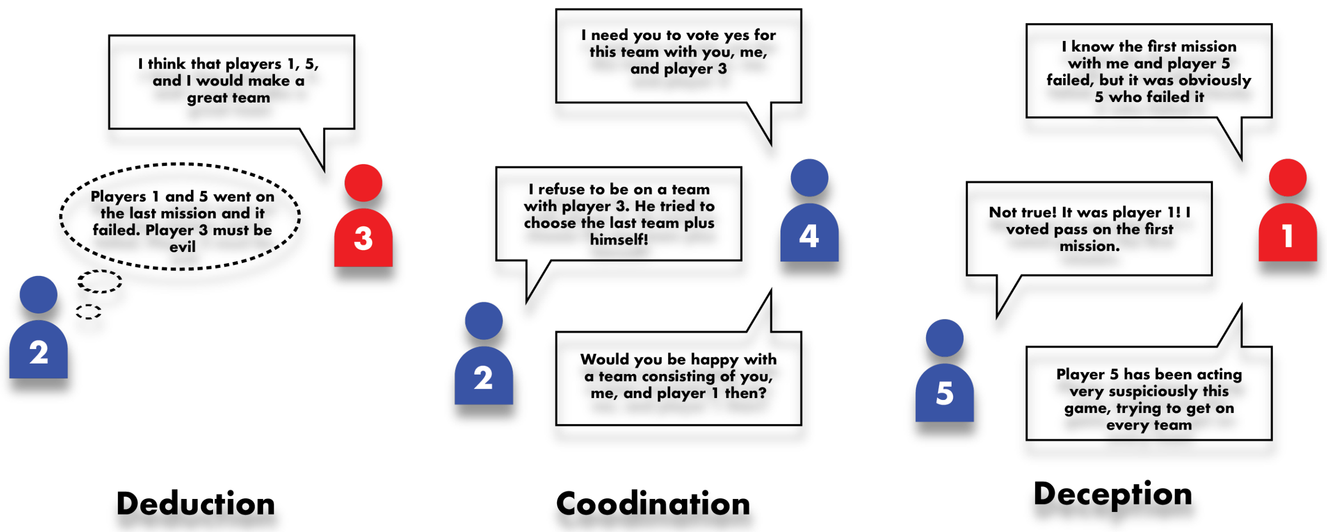
Resistance: Avalon

- Social deduction game
- Heavy language and conversation component



Challenging benchmark

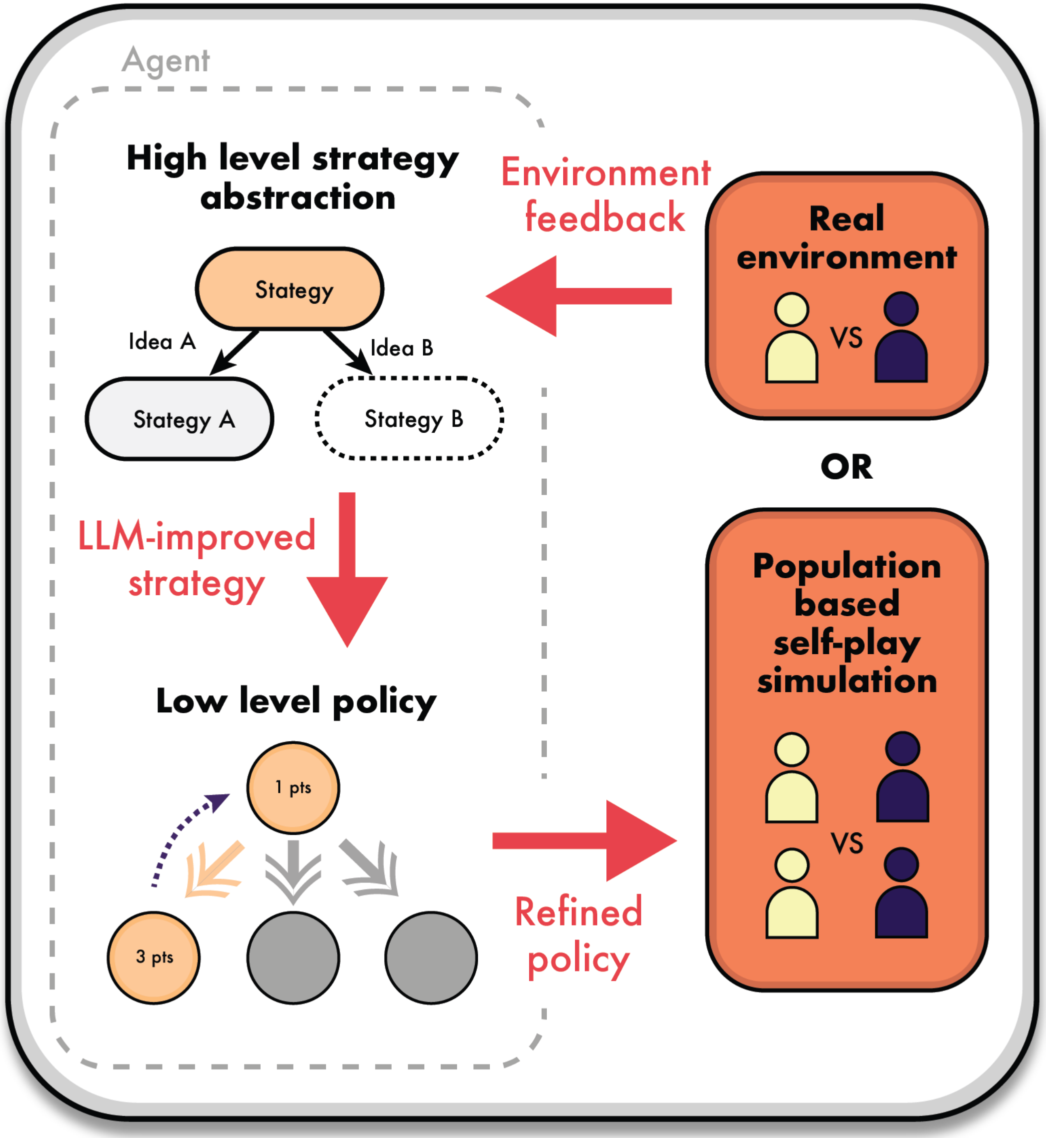
- Multi-turn
- Multi-agent
- Textual discussion
- Partial observability



Key challenge

How can we get an LLM-agent to self-improve itself for complicated tasks?

- Leverage abstraction strengths of LLM
- Extract high level "intuition" of LLM



Example value heuristic

The agent learns **how** to evaluate game states!

Very intuitive for both LLMs and humans

Avalon Value Heuristics Function (Before)

```python
def evaluate_state(state):
    num_successful_quests = sum(historical_quest_results)
    num_failed_quests = len(historical_quest_results) - num_successful_quests
    num_remaining_quests = len(num_participants_per_quest) - len(historical_quest_results)

    num_evil = len(players) - num_good
    num_evil_in_quest_team = len([player for player in quest_team if not is_good[player]])

    success_probability = 0.5
    if phase == 0:
        if num_successful_quests >= 3:
            success_probability = 0.9
        elif num_failed_quests >= 3:
            success_probability = 0.1
    elif phase == 1:
        success_probability = 0.8 if num_evil_in_quest_team == 0 else 0.2
    elif phase == 2:
        success_probability = 0.9 if num_successful_quests > num_failed_quests else 0.1
    elif phase == 3:
        if 'Merlin' in roles and 'Assassin' in roles:
            merlin_index = roles.index('Merlin')
            assassin_index = roles.index('Assassin')
            if assassin_index in quest_team:
                success_probability = 0.1
            else:
                success_probability = 0.9

    expected_winrates_per_player = dict()
    for player in players:
        if is_good[player]:
            expected_winrates_per_player[player] = success_probability
        else:
            expected_winrates_per_player[player] = 1 - success_probability

    intermediate_values = {
        'num_successful_quests': num_successful_quests,
        'num_failed_quests': num_failed_quests,
        'num_remaining_quests': num_remaining_quests,
        'num_evil_in_quest_team': num_evil_in_quest_team
    }

    return expected_winrates_per_player, intermediate_values
```

Example COT strategy

The agent learns **how** to craft deceptive statements!

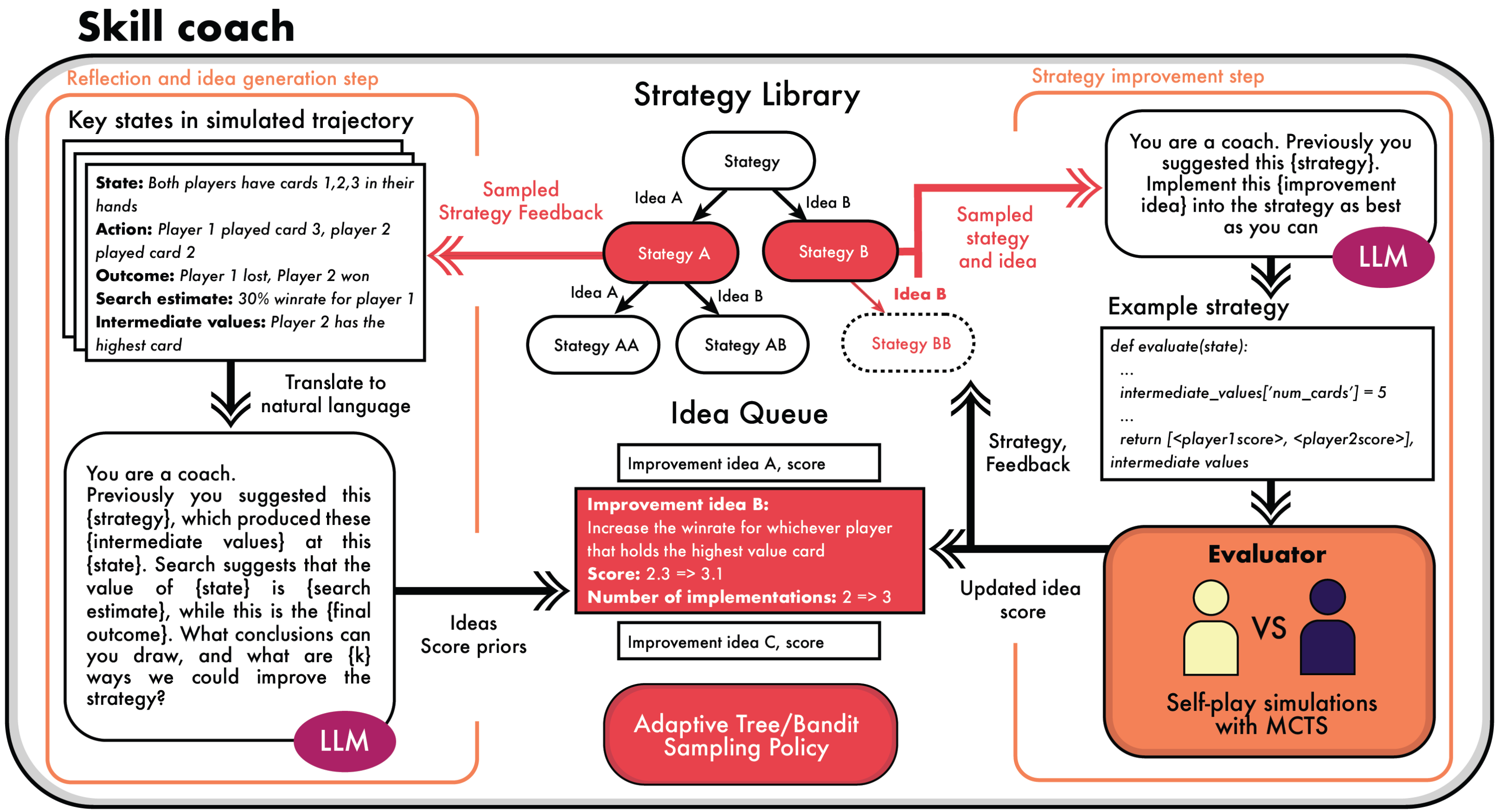
The agent learns a step-by-step COT process all by itself

Example Generated Dialogue Strategy Guide for Assassin

1. Q1: Which player seems to have a deeper understanding of the game flow than normal Servants should possess?
2. Q2: Develop a non-confrontational statement to subtly challenge this player. This should cause them to either prove their innocence or reveal more clues about their identity.
3. Q3: Who has been the most influential in the team selection and voting process?
4. Q4: Devise a statement to express agreement with this player's viewpoint subtly. This should make you less suspicious while enabling you to manipulate the discussion.
5. Q5: Which player seems the most supportive of your views and actions in the game?
6. Q6: Craft a statement subtly emphasizing your alignment with this supportive player's thoughts. This should increase your chances of being included in quest teams and reduce suspicion around you.

Self-improvement loop

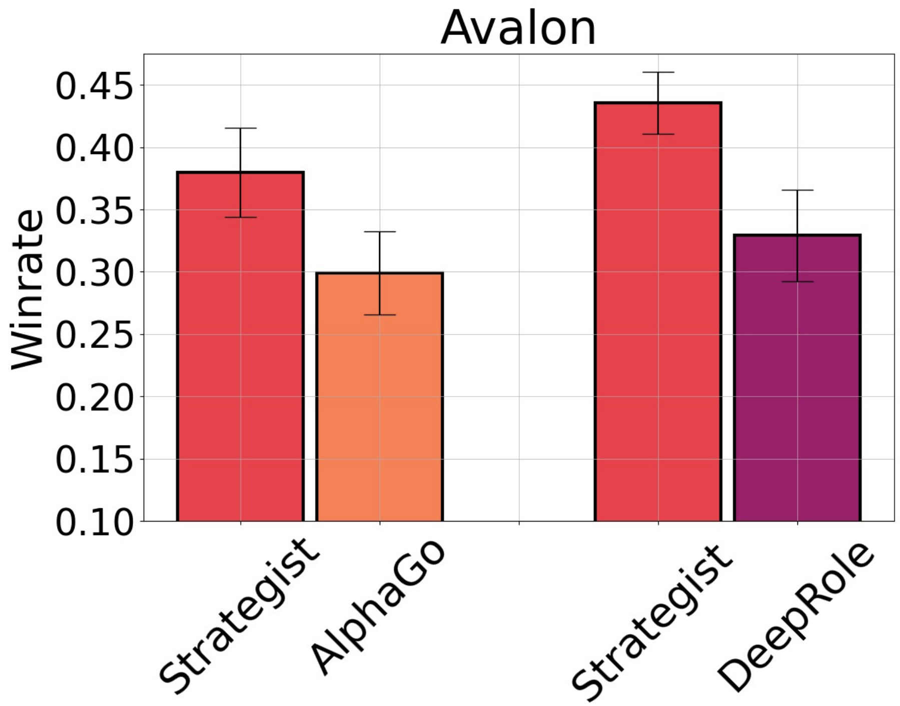
Use evolutionary search strategy



Strategist agent vs other AI agents

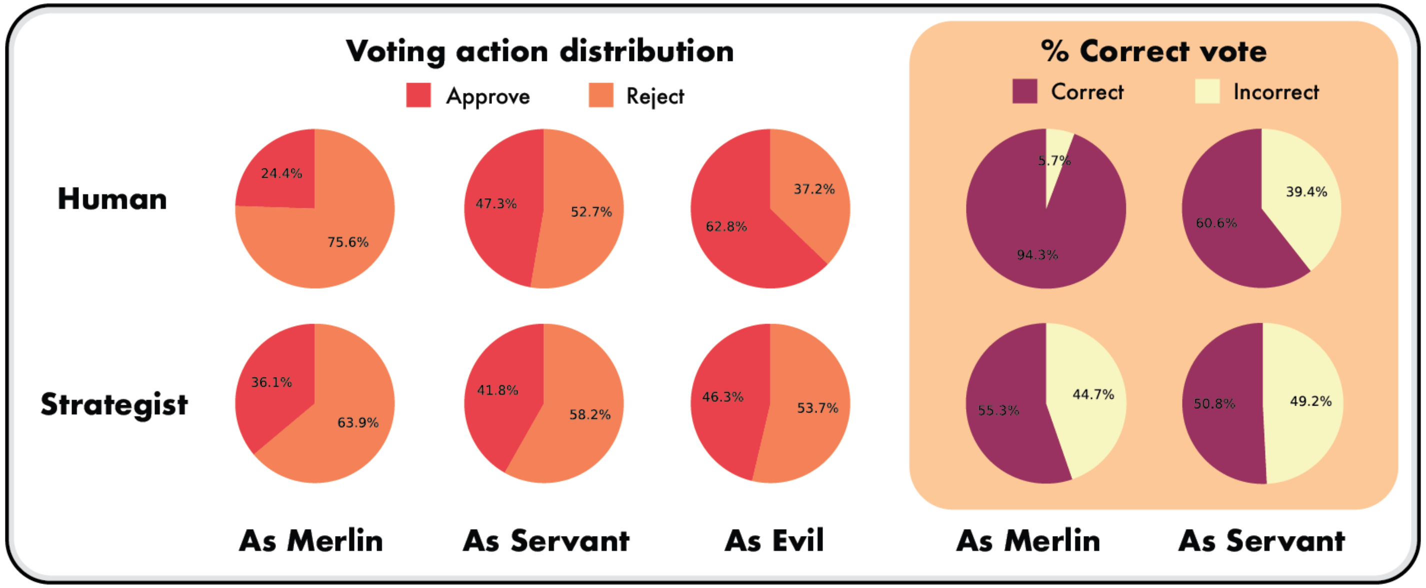
Table 5: Results of STRATEGIST playing against LLM-based baselines, i.e., ReAct and ReCon.

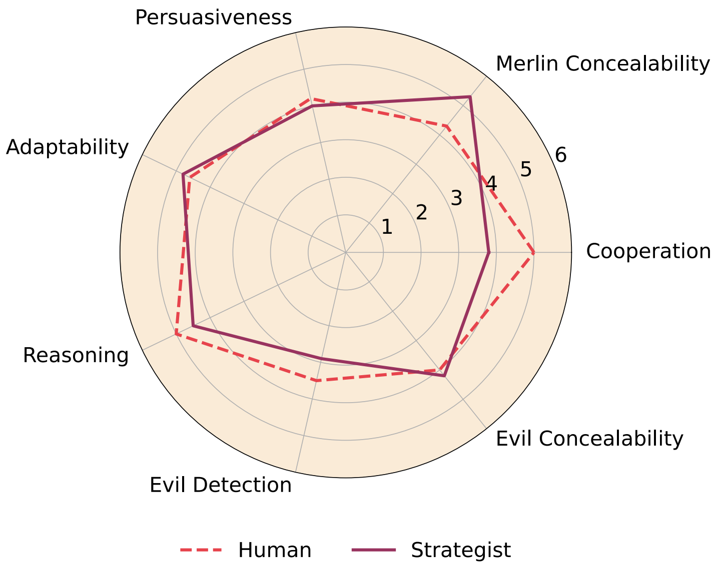
Metric	VS ReAct		VS ReCon	
	ReAct	STRATEGIST	ReCon	STRATEGIST
Winrate	47.5 ± 2.5	$\mathbf{52.5} \pm 2.5$	38.9 ± 5.5	$\mathbf{61.1} \pm 5.5$
#Tokens per round	56 ± 14.3	164.3 ± 27.7	245.7 ± 21.2	248.2 ± 24.1



Strategist compares favorably against both search only and LLM only methods

Strategist vs Humans





Strategist is much better at concealability than human players!