Different methods for environment design:

- PAIRED (Protagonist Antagonist Induced Regret Environment Design):
    - Uses a generator to generate environments
    - The agent is called the protagonist and there is another agent (assumed optimal) called the antagonist
    - Generator loss function is $-regret = R_{protagonist} - R_{antagonist}(optimal)$
    - The result is that levels are generated just above the edge of agent's ability.
- PLR (Prioritized Level Replay):
    - Scoring levels based on their learning potential.
    - Learning potential is a **score** calculated from the agent's policy and TD error in a level (no assumption of a level's difficulty, but still generating levels on edge of the agent's limits).
    - Levels are sampled from a n-dim normal distribution with parameters determined by the score
- REPAIRED (Replay Enhanced PAIRED): combines the above methods. It uses both a regret-based minimax generator and a scoring method for replaying already seen levels.
- ACCEL (Evolving Curricula with Regret-Based Environment Design): Very similar to REPAIRED, but uses a genetic algorithm to edit previously seen levels.
- VACL (Variational Automatic Curriculum Learning):
    - Aims to expand task from easy to hard beside increasing the number of agents
    - Task expansion happens as an approximation of Stein Variational Inference (Couldn't understand exactly) which leads to training tasks from easy to hard
    - Least training time compared to others.
- Some other methods pretrain agents alone, and then put them in the real multi-agent env. Like CM3: COOPERATIVE MULTI-GOAL MULTI-STAGE MULTI-AGENT REINFORCEMENT LEARNING. Or just increase the number of agents gradually like Cooperative Multi-agent Control Using Deep Reinforcement Learning and EVOLUTIONARY POPULATION CURRICULUM FOR SCALING MULTI-AGENT REINFORCEMENT LEARNING

I preferred PLR over the rest because of its simpler concept, faster training, and better results.

Different learning algorithms:

- QMIX
- MAPPO
- MADDPG

Different task suggestions:

The envs are to be designed using Unity ML Agents

task1: picking up several boxes and putting them on a target by crossing maze-like paths in an episode. 4 agents. boxes randomly need 1,2,4 persons to carry. (works with force, acceleration and friction)

Curriculum learning happens thought number of boxes and path difficulty

task2: agent should divide and cove several spots (and avoid some other) to be able to achieve a goal. sometimes they need to use available boxes nearby to do so.

Curriculum learning happens thought number of boxes and path difficulty

task3: a group of soldiers protecting a headquarter from continuous enemy attacks

Curriculum learning happens thought number of both armies and number of headquarters

Task 4: search and rescue tasks. putting obstacles aside and grabbing a goal to a marked area

Task5: multi-robot warehouse. Several robots should unload a large warehouse in the shortest time.

credit scoring: All tasks are sparse-reward, with no prior knowledge of the environment.

Project Phases:

1. Choosing a task, a design method and a learning algorithm.
2. Implementing the design method on a single agent env
3. Implementing the design method on an existing multi-agent env. (like Gym)
4. Implementing the design method on an our env.

Challenges:

1. PLR works with TD error, but in decentralized env there are TD errors for each agent.
2. I will start with a centralized critic and move to decentralized if centralized was successful

Other subjects that interested me:

- two competitive groups playing against each other like open ai's hide and seek
- More than one social learner in a multi-armed bandit (extension of your paper)