Shaping multi-agent systems with gradient reinforcement learning

Buffet, O., A. Dutech, and F. Charpillet (2007). Journal of Autonomous Agents and Multiagent Systems.

Presenter: devRandom

Crux of the paper

- How to link global cooperative task with a partially observable environment using local agents.
- Proposal of an original Reinforcement Learning (RL) methodology for the design of multi-agent systems. Agents face a sequence of progressively more complex tasks.
- Design simple reactive agents in a decentralized way to pose as independent learners
- Enable each agent to learn locally how to optimize global performance

Why Reinforcement Learning?

- Problem of automating the design of multi-agent systems (MAS) is tricky.
  - Reactive and cooperative agents rely on interaction amongst themselves to gather information
- Reinforcement Learning (RL) is a common tool to decision making under uncertainty
- Also used for MAS design under uncertain environment

The decentralized incremental learning algorithm

- The learning is decentralised because the group’s behaviour evolves through the independent learning processes of each agent.
- By incremental, it means that agents are progressively pitted against harder and harder tasks so as to progressively learn a more complex behaviour.

Types of agents

The agents are kept simple so as to avoid complexities and concentrate on the learning aspect

- Reactive : Agents act on current observations only.
- Situated with local perception : Limits the number of possibilities an agent can experience at one time, limits combinatorial explosion events.
- Possibly heterogenous : Every agent can acquire different behaviors.
- Cooperative : Agents share same goal, and that goal needs cooperation.

Markov Decision Process

- Classic MDP’s are defined with <S, A, T, r>
  - S is a finite set of states
  - A is finite set of actions
  - T is the transition function with mapping T : S X A $\longrightarrow$ [0,1], a probability.
  - r is mapping of S X A to a reward.

Partially observable MDP

- The framework deals with partial observations
- Agents do not have access to a complete state and are accessing partially observable MDPs (POMDP)
- POMDP adds a set Ω of possible observations and observation function O linking states to observations.

Complexity in framework

- Partial observability
- Non stationary transitions
- Multi-agent credit assignment

Shaping: Incremental RL

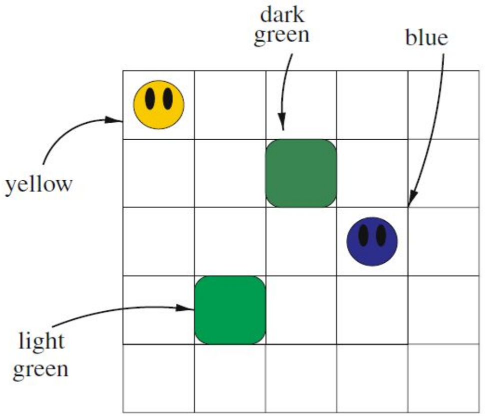
- Agent uses a similar but less complex problem to the goal.
- Progressively put in more and more complex problems eventually reaching back to the goal problem.
- For this system:
  - Rewards may be redefined.
  - Physics of a system may be altered (the MDP’s transition function).

Shaping for MAS

- Growing task complexity:
  - First task has very few actions that the agent can take that do not result in positive reinforcement, speeds up finding the ‘correct’ action.
  - Each ensuing task increases the number of actions and freedom an agent has.
- Growing MAS:
  - The number of agents in a task grows as well, done by taking current agents and cloning them.
  - Add more objects to environment, increasing complexity of learning for agents.
    - Agent’s internal architecture must be adapted for additional agents and objects.

Problem Description

- Agents (Yellow or Blue) must bring cubes together for reward, no reward is given at any other time.
- Agents can only move in the cardinal directions (N,S,E,W).
- Each agent knows basic information about the system.
  - Direction of each cube relative to themselves.
  - Direction of other agent relative to themselves
  - If they are close to a cube



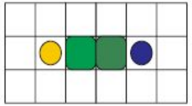
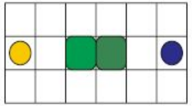
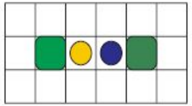
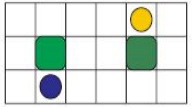
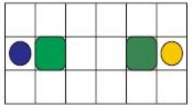
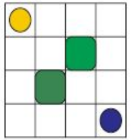
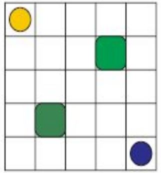
agent	dir(cl)	dir(cd)	dir(oa)	near(cl)	near(cd)
yellow	S	E	SE	no	no
blue	W	NW	NW	no	yes

cl: cube light
cd : cube dark

oa: other agent

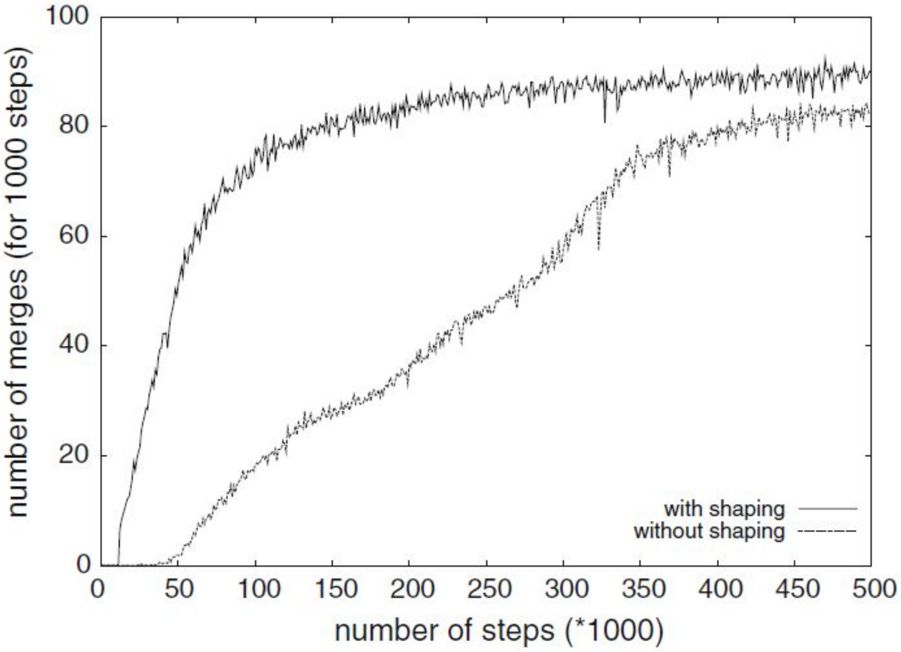
Experiment: 2 agents 2 cubes

- Sequence is run through once.
- Given n moves to complete (or explore), run for N trials.
- Agents are kept from one configuration to the next.
- Agents can only move once per step in time.

Starting configuration	n (moves)	N (trials)
	6	150
	6	100
	10	150
	20	150
	20	150
	100	15
	100	15

Results: 2 agents, 2 cubes

- Agents with shaping compared to those without shaping. (without any prior experience but using same algorithm)
- Compared using number of merges of the block per 1000 steps.
- Recording starts after 12000 time steps of above sequence.

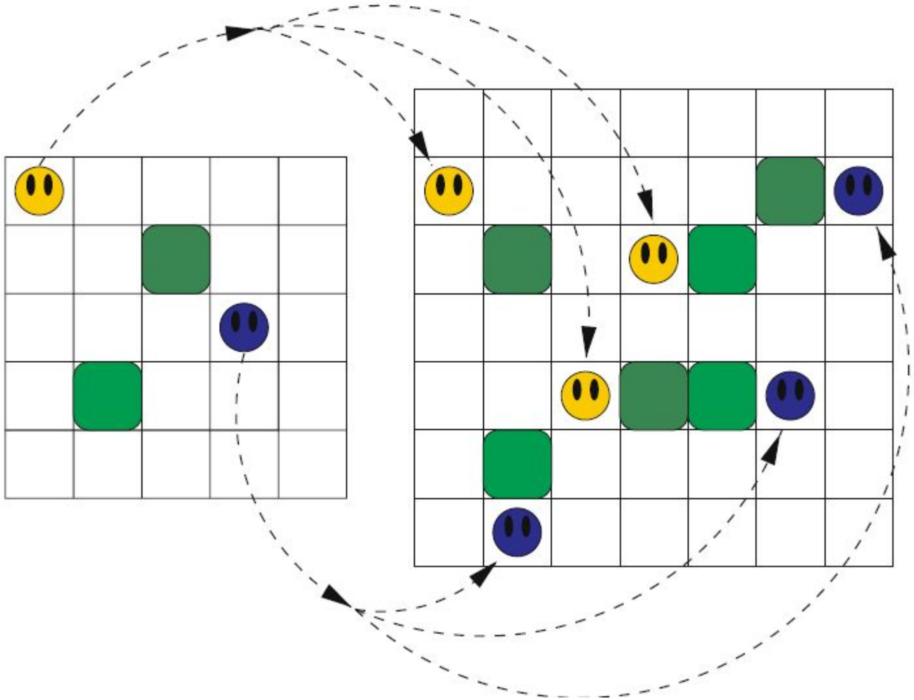


Results: 2 agents, 2 cubes

- Results show that convergence to 90 merges per 1000 steps is normal for both methods.
- But the agents with shaping converge much faster towards this.
- Shows that even though the training does not help during the early period in the experiment, it does help the agent learn more quickly once a merge is accomplished.

Experiment: n agents, m cubes

- 10 x 10 environment.
- 2, 4, 6, 8, 10 cubes and 2, 4, 8, 16, 20 agents.
- Agents spawn in random locations.
- Compared agents from 2 agents, 2 cubes experiment to random agents.
- tested 1000 steps per series, 100 series per situation.



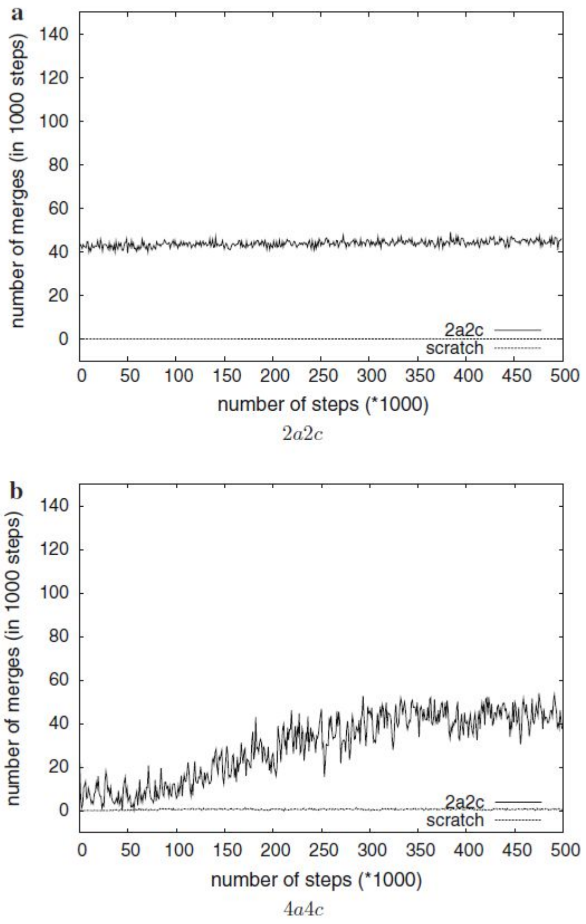
Results: n agents, m cubes

- Most merges when 2a2c.
- A higher number of cubes sometimes led to oscillatory motions for agents.
- Usually a larger number of agents helped if there were enough cubes. If there were not agents may block one another.

Cubes	Agents				
	2	4	8	16	20
a. Reuse of 2a2c agents					
2	40.4	30.0	20.0	12.7	11.0
4	7.6	17.1	17.5	13.9	12.9
6	3.4	11.2	14.7	15.7	16.5
8	1.9	8.6	13.5	15.9	18.0
10	1.6	6.7	11.0	17.7	20.6
b. Random agents					
2	0	0	0.1	0.3	0.4
4	0	0.1	0.2	1.2	1.8
6	0	0	0.5	1.9	3.6
8	0.1	0.2	1.0	4.1	6.0
10	0.1	0.3	1.1	6.1	7.3

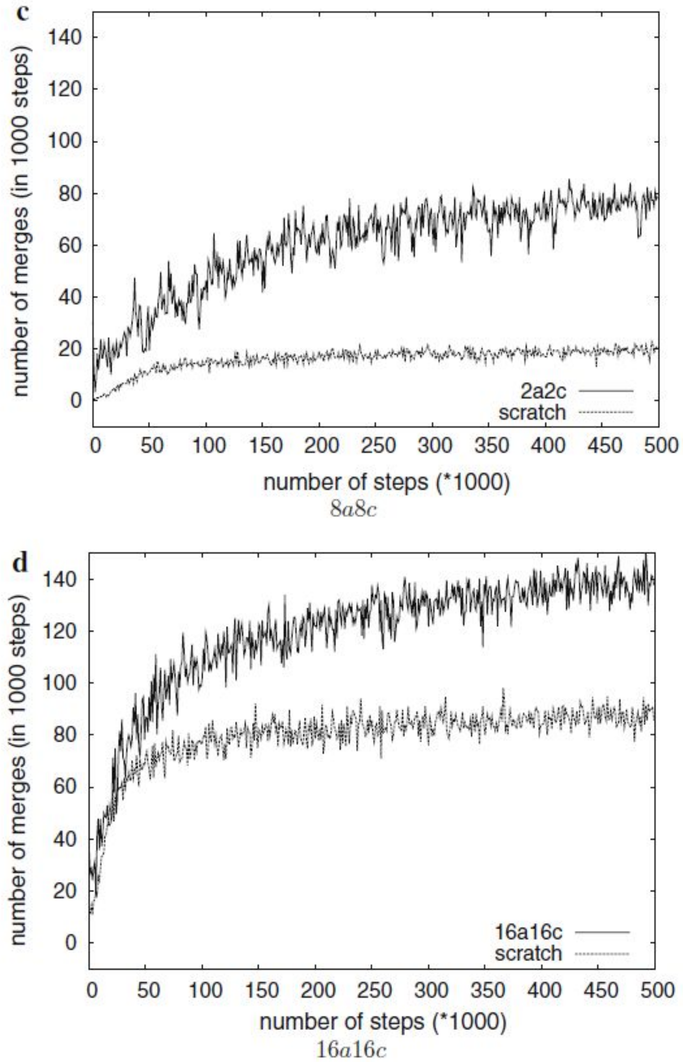
Comparing agents

- Agents with knowledge from 2a2c compared to agents without any previous knowledge.
- Number of merges for 1000 steps.
- in (a) and (b) default agents without any previous knowledge did not learn anything.
  - This is due to a large environment with small amount of cubes. This rarely led agents to pushing any cubes together accidentally.



Comparing Agents

- in (c) and (d) default agents were more likely to push block together accidentally and learn something.
- In general, agents with experience in similar situations performed better than default agents.
- Default agents in a environment with many objects and agents find it difficult to learn what they should do.



Discussion

- Communication
  - Explicit coordination could have benefited situations such as 2 agents with more cubes needed some kind of coordination in order to push a pair of cubes together.
  - However, would lead to increase cost and time spent communicating.
- Automated Shaping
  - There is a few problems in the method in this paper, one being a problem needs to be broken down in the similar more simple problems.
  - Another problem is in tasks which require specialized agents as this method utilizes cloning.

Conclusion

- The main issue that the paper discusses in how to automate multi-agent system
  - consisting of reactive and cooperating agents
  - Each acting independently
- Reinforcement Learning algorithm is proposed which helps with the above motto
- Results of conducted experiments show the efficiency of incremental learning, leads to better rates than agents learning from scratch.

Thank-you!