Multi-Agent Systems
Projects

Tommaso Padoan

University of Trieste

Further Instructions

Each project should be reported in a report ∼1 page long (without references) to be uploaded on the Team of the course, strictly before the exam call.

Although not required, along the report you can also submit other relevant material, e.g. code. In the case of code, uploading on github and sharing the link in the report is appreciated.

Projects Proposals

1 Labelled Transition Systems for Agents' Behaviour

Setting: Prison Puzzle

Three prisoners kept in separate cells get a chance to be released. From time to time, one of them will be carried to a special room (in unknown order, possibly multiple times consecutively, but with a fair schedule to avoid infinite wait) and then back to the cell. The room is completely empty except for a switch that can turn the light on and off (the light is not visible from outside). Initially the light in the room is off.

At any time, if any of them says that all the prisoners have already entered the room at least once and this is true, then all prisoners will be released (but if it is false, then they will not be released nor have ulterior chances).

Before the challenge starts, the prisoners are allowed to discuss together some "protocol" to follow, after that they cannot directly interact any more. One of the prisoner can also bring with him (shackled to him) a counter allowing for counting up to a certain number, he must choose and fix such a number before starting the challenge.

Objectives

- Define the LTSs describing how each of the three prisoners should behave to guarantee their release, in compliance with the rules given above, independently of the order in which they will be eventually carried to the room. Moreover, explain what counter (number) will be used and why. Depending on your solution, prisoners may have different LTSs, or some may share the same behaviour. Whenever a prisoner is sure that all of them have already entered the room at least ones, he can perform action *done* to terminate the challenge.

  Non-deterministic choices are always assumed to be dictated by synchronizations with the environment. Recall that all interaction with components of the environment (the room, the scheduler, and the chosen counter) must be synchronizations. For example, a prisoner can perform action *on*, to turn on the light in the room, only when the room itself can perform the *on* action, synchronizing with it, that is only when the light is currently off. (Hint: one can check the state of the light in the room by trying to turn it on or off, respectively, since only one of the two actions can actually synchronize depending on the current state.)

  The LTSs representing the room with the light, a simple fair scheduler for the prisoners' turns, and the counter to count up to n are given below.

Room

$$\begin{tikzcd}
r_0 \arrow[d, bend left, "\text{enter}"] \\
r_2 \arrow[u, bend left, "\text{exit}"] \arrow[d, bend left, "\text{on}"] \\
r_3 \arrow[u, bend left, "\text{off}"] \arrow[d, bend left, "\text{exit}"] \\
r_1 \arrow[u, bend left, "\text{enter}"]
\end{tikzcd}$$

Scheduler

$$\begin{tikzcd}
 & t_0 \arrow[dr, "\text{cell:1}"] & \\
t_2 \arrow[ur, "\text{cell:3}"] & & t_1 \arrow[ll, "\text{cell:2}"]
\end{tikzcd}$$

Counter to n

$$\begin{tikzcd}
c_0 \arrow[loop right, "\text{cont}"] \arrow[d, "\text{add}"] \\
c_1 \arrow[loop right, "\text{cont}"] \arrow[d, dashed] \\
c_{n-1} \arrow[loop right, "\text{cont}"] \arrow[d, "\text{add}"] \\
c_n \arrow[loop right, "\text{end}"]
\end{tikzcd}$$

The initial states are r_0, t_0, and c_0, respectively. Actions *on* and *off* of the room mean to turn the light inside on and off respectively, thus the light is off in states r_0, r_2 and it is on in states r_1, r_3. Action *cell:i*, for $i \in \{1, 2, 3\}$, of the scheduler corresponds to starting the turn of prisoner i, like the jailer opening the cell (each prisoner i can perform the corresponding *cell:i* action himself to synchronize with the scheduler to know when is his turn to act). Action *end* of the counter corresponds to reading that it has reached the chosen number, while *cont* would be used to check that the counter has not yet reached the end.

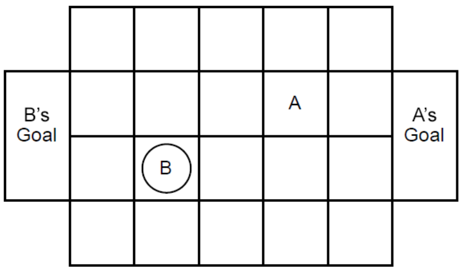
- Express the property *"eventually the prisoners give the correct solution"* as a temporal (recursive) property in HML, and explain why it should hold on the initial state of the whole system obtained via parallel composition of all the agents' LTSs (do not draw such a system which would be huge).
- Discuss how the LTSs of the prisoners (and the choice of counter) could be modified to let them solve the problem even if the initial state of the light would be unknown, that is, if the initial state of the room could be both r_0 or r_1 and the prisoners do not know which.

2 Multi-Agent Reinforcement Learning on Stochastic Game

Setting: Simplified Football Game

A simplified version of football introduced by [Lit94]. The game is played on a grid 4×5 as depicted below. Two players, A and B, occupy distinct squares of the grid and can choose one of 5 actions on each turn: N, S, E, W, or *stand*, corresponding to wanting to move one square towards the specified direction (North, South, East, West) or standing still in the current position. Once both players have selected their actions, the two moves are executed in random order (uniformly). Initially players are positioned as shown in the figure. The circle in the figures represents the ball possession. Initially possession of the ball goes to one or the other player at random (uniformly). When a player with the ball steps into its opponent's goal area (left for A, right for B), that player wins the game and receives $+1$ reward (while its opponent receives -1). Then, the game is reset to the initial configuration, and ball possession is reassigned randomly to one of the players.

When a player executes an action that would take it to the square occupied by the other player, possession of the ball goes to the latter player and the move does not take place. A good defensive maneuver, then, is to stand where the player with the ball wants to go. For example, if the two players are one in front of the other and they both choose to go towards the other, none of them will actually move and ball possession will shift to (or stay with) the one randomly extracted to move first, since it will stop the other's movement during the second action, taking (or keeping) the ball.



Objectives

- Implement the belief-based joint action learning algorithm (explained in class), with behavioural strategies. As belief function you can use the one mentioned in class (frequency of the opponent's actions in each state) and/or experiment your own.

- Use the implemented algorithm to learn behavioural strategies for the two-player zero-sum stochastic game representing the simplified version of football by [Lit94]. As in the paper, the learning must be performed against a random opponent, and against another learner of identical design, for 10^6 steps and parameters:

$$\begin{aligned} \epsilon &= 0.2 \\ \alpha_0 &= 1.0 \\ \alpha_{t+1} &= \alpha_t \cdot 10^{\log_{10}(0.01)/10^6} \cong \alpha_t \cdot 0.9999954 \\ \gamma &= 0.9 \end{aligned}$$

- Test, for 10^5 steps, the obtained strategies against a random opponent and against each other, counting the total number of games actually finished and the % of games won. To emulate the discount factor, at every step the game can terminate immediately with probability $1 - \gamma = 0.1$, resulting in a draw (0 reward to both players) and resetting the game. Then discuss your findings, and try to compare them with those in [Lit94], reported below, even though the latter were also tested against Q-learned best response strategies.

	minimax-Q (random learning)		minimax-Q (identical learning)		independent-Q (random learning)		independent-Q (identical learning)	
	% won	games	% won	games	% won	games	% won	games
vs. random	99.3	6500	99.3	7200	99.4	11300	99.5	8600
vs. Q-best	35.0	4300	37.5	4400	0.0	5500	0.0	1200

References

[Lit94] M. L. Littman (1994) *Markov games as a framework for multi-agent reinforcement learning.* Proceedings of ICML'94.

3 Automated Mechanism Design for Bartering

Setting: One-on-one Bartering

Two agents have an initial set of 3 goods each. They also have a valuation for every subset of the whole set of 6 goods they have together. It is possible that both agents can get better off by trading some goods. Suppose, however, that the agents cannot make any form of payment. All they can do is trade goods. This is known as bartering. Additionally, suppose that one agent (agent 1) can dictate the rules of the bartering process. Agent 1 can credibly say to agent 2: "we will barter by my rules, or not at all". This makes agent 1 the mechanism designer and agent 2 the only player. The set of outcomes O is the set of all allocations of the goods (the partitioning of the 6 goods, of which there are $2^6 = |O|$). Agent 2 is to report (possibly lying) his preferences over the goods (a value in $\{1, 2, 3, 4, 5\}$ for each item). Based on the report an outcome is chosen. Then, the valuation $u : \Theta \times O \to \mathbb{Z}$ that agent 2 has of each allocation is the sum of the corresponding values, minus the valuation of what he owned to begin with. So, the mechanism is direct, where the preferences are both the type and the possible actions of agent 2. The outcome function, which is selected by agent 1 in advance, must be truthful so that agent 2 has no incentive to lie. Also, the outcome function must ensure that agent 2 does not incur a loss as a result of participating in the mechanism (known as individual rationality constraint), otherwise he would not trade at all. Under these constraints, agent 1 wants to maximise her own valuation $g : O \to \mathbb{Z}$ (defined in the same manner as agent 2, but based on her own preferences, assumed to be fixed) of her resulting allocation of goods under the mechanism.

Objectives

- Implement the branch and bound depth-first search algorithm for automated mechanism design introduced by [CS04] and reported below.

```
SEARCH1($X$, $Y$, $w$, $d$):
  if $d$ > $|O|$:
    $CB$ = $X$
    $L$ = $w$
  else:
    if $v(X \cup \{o_d\}, Y)$ > $L$:
      SEARCH1($X \cup \{o_d\}$, $Y$, $v(X \cup \{o_d\}, Y)$, $d+1$)
    if $\forall \theta \in \Theta. \exists o \in O \setminus (Y \cup \{o_d\}). u(\theta, o) \geq 0$ and $v(X, Y \cup o_d)$ > L:
      SEARCH1($X$, $Y \cup o_d$, $v(X, Y \cup o_d)$, $d+1$)

BnB-DFS():
  $CB$ = None
  $L$ = $-\infty$
  SEARCH1($\emptyset$, $\emptyset$, 0, 1)
  return $CB$
```

  As in the paper, the definition of function v on disjoint subsets of outcomes $X, Y \subseteq O$ is

$$v(X,Y) = \sum_{\theta \in \Theta} p(\theta) \cdot \max\{g(o) \mid o \in O \setminus Y \wedge \forall x \in X. u(\theta, o) \geq u(\theta, x)\}$$

  where p is the probability distribution over types Θ, so $p(\theta) = 1/|\Theta|$ if uniform. Also, notice that outcomes are indexed by d, so that o_d is the d-th outcome, for $1 \leq d \leq |O|$.

- Apply the algorithm to the above setting to find a truthful individually rational deterministic mechanism without payments. To do this, fix beforehand $50 = |\Theta|$ different possible types Θ for agent 2 uniformly distributed, by randomly drawing a preference value for each good for each type. Similarly, uniformly randomly draw and fix the preferences for agent 1 beforehand (just once). If the computation takes too long, first try with agents having 2 goods each, 4 in total, and $|\Theta| = 15$.

  Then, verify that the obtained mechanism is truthful and individually rational. Note that, given output CB of the algorithm, the corresponding mechanism is

  $$M_{CB}(\theta) = \arg\max_{o \in CB} \{g(o) \ \mid \ \forall x \in CB.\, u(\theta, o) \geq u(\theta, x)\}$$

  which chooses an optimal outcome for agent 1 among those in CB providing the highest utility to agent 2.

References

[CS04] V. Conitzer and T. Sandholm (2004) *An algorithm for automatically designing deterministic mechanisms without payments.* Proceedings of AAMAS'04.