Model checking of game-theoretic learning of cooperation by robots

Hongyang Qu, Michalis Smyrnakis and Sandor M. Veres

Department of Automatic Control and Systems Engineering, University of Sheffield
This research was supported by EPSRC Grant EP/J011894/2
A distributed decision making example in daily life
An example in robotics

- Two UAVs fly towards each other at the same altitude
- Each UAV has two actions:
  - High altitude and low altitude
- UAV 1: action A (high) or B (low)
- UAV2: action C (low) and D (high)

<table>
<thead>
<tr>
<th></th>
<th>Action C</th>
<th>Action D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action A</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Action B</td>
<td>✗</td>
<td>✔️</td>
</tr>
</tbody>
</table>

This scenario can be cast as a game
Game theory and learning algorithms

- Game theory is the study of mathematical models of conflict and cooperation between intelligent rational decision-makers.
- Small communication costs
  - Suitable for robots in a complex environment.
Motivation

• Coordination is needed to efficiently execute tasks for robot teams

• Game-theoretic learning algorithms are used as coordination mechanism of robots working together

• Most of them converge to an optimal solution for certain classes of games when the game is played for a large number of rounds
Motivation II

Cooperation needs to be safe and verified, hence we need to answer:

• How algorithms behave:
  • Where the agents have a *limited number of coordination steps*. In real life / robotics applications a large number of rounds is not practical.
  • Where their *initial choices of robots* are random.
  • Where the class of the game does not *strictly* belong to the ones for which the algorithm converges to an optimal solution.

• How can users compare the performance of robot teams?
  • An algorithm may converge *fast* in an optimal solution but with *huge computational cost*
  • Other algorithms have *similar performance with significantly lower cost*
Outline

• Game-theoretic definitions and learning algorithms
• Model checking framework for performance evaluation
• Experimental results
• Conclusions and future work
Outline

- Game-theoretic definitions and Learning algorithms
- Model checking framework for performance evaluation
- Experimental results
- Conclusions and future work
Elements of a game

• A set of Players $i = 1,2, ..., I$
• A set of actions for each player $s^i \in S^i$
• A set of joint actions $s = (s^1, s^2, ..., s^I) \in S = S^1 \times S^2 \times ... \times S^I$
• A reward function for each player, $r^i: s \to R$
• A set of strategies (probability to select an action) for each player $\sigma^i \in \Sigma^i$
• A set of joint strategies $\sigma = (\sigma^1, \sigma^2, ..., \sigma^I) \in \Sigma = \Sigma^1 \times \Sigma^2 \times ... \times \Sigma^I$
Nash equilibrium and decision rules

• A Nash equilibrium is a joint strategy in which if it is played, no player would deviate from it unilaterally.

\[ r^i(\sigma^1, \sigma^2, ..., \sigma^i, ..., \sigma^I) \geq r^i(\sigma^1, \sigma^2, ..., \sigma'^i, ..., \sigma^I) \quad \forall \sigma'^i \in \Sigma^i, \forall i \in I \]

• Decision rules
  • Best response (players choose the action which maximises their expected reward)
    \[ BR^i = \arg\max_{s^i} r^i(\sigma^1, \sigma^2, ..., s^i, ..., \sigma^I) \]
  • Smooth best response (players choose an action using a probability distribution)
    \[ \overline{Br}^i(s^i) = \frac{r^i(\sigma^1, \sigma^2, ..., s^i, ..., \sigma^I)}{\sum_{s^j \in \Sigma^i} r^i(\sigma^1, \sigma^2, ..., s^j, ..., \sigma^I)} \]
Examples of games

• Simple coordination game

<table>
<thead>
<tr>
<th>Row Player</th>
<th>Column Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_2</td>
<td>C_1: α, α</td>
</tr>
<tr>
<td></td>
<td>C_2: 0, 0</td>
</tr>
<tr>
<td>R_1</td>
<td>C_1: 0, 0</td>
</tr>
<tr>
<td></td>
<td>C_2: α, α</td>
</tr>
</tbody>
</table>

• Stag and hunt game

<table>
<thead>
<tr>
<th>Row Player</th>
<th>Column Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_2</td>
<td>C_1: 6, 6</td>
</tr>
<tr>
<td></td>
<td>C_2: 0, 5</td>
</tr>
<tr>
<td>R_1</td>
<td>C_1: 0, 0</td>
</tr>
<tr>
<td></td>
<td>C_2: 4, 4</td>
</tr>
</tbody>
</table>

• Bayesian Games (Environment’s state (s_1 or s_2) influences the rewards of players)

• Stackelberg game (Environment is in s_1).

<table>
<thead>
<tr>
<th>Row Player</th>
<th>Column Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_2</td>
<td>C_1: 2, 1</td>
</tr>
<tr>
<td></td>
<td>C_2: 4, 0</td>
</tr>
<tr>
<td>R_1</td>
<td>C_1: 1, 0</td>
</tr>
<tr>
<td></td>
<td>C_2: 3, 2</td>
</tr>
</tbody>
</table>

• Stackelberg game (Environment is in s_2).

<table>
<thead>
<tr>
<th>Row Player</th>
<th>Column Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_2</td>
<td>C_1: 1, 1</td>
</tr>
<tr>
<td></td>
<td>C_2: 2, 0</td>
</tr>
<tr>
<td>R_1</td>
<td>C_1: 0, 1</td>
</tr>
<tr>
<td></td>
<td>C_2: 3, 2</td>
</tr>
</tbody>
</table>
Examples of algorithms

Fictitious play

**Initialisation**
- Initialise weights to estimate opponents strategies
- Choose an action

**Estimation**
- Update the weights by averaging over all the previously observed actions
- Estimate the other players’ strategies using the updated weights

**Decision**
- Each player uses the estimates of other players’ strategies and chooses the action that maximises his expected rewards

Repeat till termination

Geometric fictitious play

**Initialisation**
- Initialise weights to estimate opponents strategies
- Choose an action

**Estimation**
- Update the weights. Give higher impact to the most recently observed action, by discounting the historic observations by a constant factor.
- Estimate the other players’ strategies using the updated weights

**Decision**
- Each player uses the estimates of other players’ strategies and chooses the action that maximises his expected rewards

Repeat till termination
Outline

• Game-theoretic definitions and Learning algorithms
• Model checking framework for performance evaluation
• Experimental results
• Conclusions and future work
Model checking (formal verification)

- It is a **systematic** way to check all behaviour of a system with respect to certain specification.

System \[ \xrightarrow{\text{Abstraction}} \] Mathematical model \[ \xrightarrow{\text{Verification algorithm}} \] Result

- Abstraction
- Logic formula
- Verification algorithm

\[
\mathbf{L}(\text{call} \lor \text{open}) \implies (\neg \text{at floor} \lor \neg \text{open}) \lor \\
(\text{open} \lor (\text{at floor} \land \neg \text{open}) \lor \\
(\text{open} \lor ((\neg \text{at floor} \land \neg \text{open}) \lor \\
(\text{open} \lor ((\neg \text{at floor} \land \neg \text{open}) \lor \\
(\text{open} \lor (\neg \text{at floor} \land \text{open})))))))
\]
Probabilistic Model Checking

• Probabilistic model checking is used to verify/analyse systems that have probabilistic behaviour
  • First step: build a mathematical model (usually MDP/DTMC/CTMC) for the system in question
  • Second step: run a probabilistic model checking algorithm to compute probabilities against a probabilistic specification
Discrete-Time Markov Chains (DTMCs)

• A DTMC is a probabilistic state-transition system with transitions labelled
  • A state is a possible configuration of the system
  • Transitions between states represent evolution of the system
  • From a state, the system can move to other states with certain probabilities

• A DTMC is memoryless, which means the probability distribution in a state does not depend on the history of evolution
Verification framework for analysing performance of learning algorithms
The answers our framework can provide

• Reachability properties
  • The probability of reaching a set of states from the initial state
  • Example: Pareto-efficient Nash equilibria can be reached in 10 iterations with probability 90%.

• Steady state properties
  • The probability of staying in a state in the long run
  • Example: What is the probability of occurrence of a specific state in the long run?

• Reward properties
  • Properties about instantaneous/cumulative rewards attached to states and/or transitions
  • Example: What is the average time needed to reach Pareto-efficient Nash equilibria?
An example in robotics

- Two UAVs fly towards each other at the same altitude

- Each UAV has two actions:
  - High altitude and low altitude

- UAV 1: action A (high) or B (low)
- UAV 2: action C (low) and D (high)

<table>
<thead>
<tr>
<th></th>
<th>Action C</th>
<th>Action D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action A</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Action B</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>
Reward matrix

<table>
<thead>
<tr>
<th></th>
<th>Action C</th>
<th>Action D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action A</td>
<td>1, 1</td>
<td>0, 0</td>
</tr>
<tr>
<td>Action B</td>
<td>0, 0</td>
<td>1, 1</td>
</tr>
</tbody>
</table>

Initial estimation of opponents’ strategy
Player 1: [0.9 0.1]
Player 2: [0.1 0.9]
Mathematical model

Reward matrix

<table>
<thead>
<tr>
<th></th>
<th>Action C</th>
<th>Action D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action A</td>
<td>1, 1</td>
<td>0, 0</td>
</tr>
<tr>
<td>Action B</td>
<td>0, 0</td>
<td>1, 1</td>
</tr>
</tbody>
</table>

Initial estimation of opponents’ strategy
Player 1: [0.9  0.1]
Player 2: [0.1  0.9]
Mathematical model

Reward matrix

<table>
<thead>
<tr>
<th></th>
<th>Action C</th>
<th>Action D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action A</td>
<td>1, 1</td>
<td>0, 0</td>
</tr>
<tr>
<td>Action B</td>
<td>0, 0</td>
<td>1, 1</td>
</tr>
</tbody>
</table>

Initial estimation of opponents’ strategy

Player 1: [0.9  0.1]
Player 2: [0.1  0.9]
Mathematical models for other algorithms

GFP & AFFFP

RM & MGRM

SAP
Outline

• Game-theoretic definitions and Learning algorithms
• Model checking framework for performance evaluation
• Experimental results
• Conclusions and future work
Algorithmic performance evaluations for the simple coordination game

<table>
<thead>
<tr>
<th></th>
<th>FP</th>
<th>GFP</th>
<th>AFFFP</th>
<th>DSA</th>
<th>RM</th>
<th>MGRM</th>
<th>SAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of States</td>
<td>9</td>
<td>64</td>
<td>31</td>
<td>5</td>
<td>9</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>Number of iterations</td>
<td>2</td>
<td>57</td>
<td>24</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Convergence probability</td>
<td>0.7204</td>
<td>0.8313</td>
<td>0.6666</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Algorithmic performance in a complex coordination game

- set $n = 5$, $\zeta = 1 + \frac{1}{n^{1-\sigma}}$, $\beta = 1 - \frac{1}{n^{2-1-\sigma}}$
- $u(i, j) = 1$, $\forall i \in [n + 1, 4n]$, $j = i$
- $u(i, j) = 1$, $\forall i \in [2, n]$, $j = i - 1$
- $u(i, j) = \zeta$, $\forall i \in [n + 1, 4n]$, $j = i - 1$
- $u(i, j) = \zeta$, $i = 2n + 1$, $j = 4n$
- $u(i, j) = \beta$, $\forall j \leq 2n$, $i > j$
- $u(i, j) = \beta$, $\forall i - j \leq n$, $i > j$
- $u(i, j) = \beta$, $\forall i \in [2n + 1, j - n]$, $j \in [3n + 1, 4n]$
- $u(i, j) = 0$, Otherwise

<table>
<thead>
<tr>
<th>States</th>
<th>FP</th>
<th>GFP</th>
<th>AFFFP</th>
<th>DSA</th>
<th>RM</th>
<th>MGRM</th>
<th>SAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2373</td>
<td>2678</td>
<td>2271</td>
<td>362</td>
<td>1121</td>
<td>1121</td>
<td>1113</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>32</td>
<td>32</td>
<td>1</td>
<td>22</td>
<td>22</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>0.9969</td>
<td>0.999</td>
<td>0.9115</td>
<td>0.0</td>
<td>0.04</td>
<td>0.04</td>
<td>0.1905</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>States</th>
<th>FP</th>
<th>GFP</th>
<th>AFFFP</th>
<th>DSA</th>
<th>RM</th>
<th>MGRM</th>
<th>SAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2373</td>
<td>2678</td>
<td>2271</td>
<td>362</td>
<td>1121</td>
<td>1121</td>
<td>1113</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>32</td>
<td>32</td>
<td>1</td>
<td>22</td>
<td>22</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>0.9969</td>
<td>0.999</td>
<td>0.9115</td>
<td>0.0</td>
<td>0.04</td>
<td>0.04</td>
<td>0.1905</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions and future work

• The short term behaviour of game-theoretic learning algorithms can be studied under our framework

• The proposed framework can serve as a comparison tool between learning algorithms in robot cooperation

• Future work
  • Test more realistic games/simulation scenarios
  • Study more distributed optimisation algorithms (may non-game-theoretic)
  • Investigate cases where players/agents use different learning algorithms