Hierarchical Common-Sense Interaction Learning

Michael Rovatsos
Knowbotic Systems GmbH & Co KG

Jürgen Lind
German Research Center for Artificial Intelligence (DFKI)

Introduction

- The need for *coordination* among agents is inherent to the nature of multi-agent systems.

- Difficulty of predicting society-level phenomena on the grounds of local interactions suggests *learning* coordination strategies.

- Game-theoretic models widely used to model interaction situations at an abstract level (e.g. mechanism design, game-learning).

- However, little research focuses on agents learning something about the interaction *itself*.

- Idea: enable agents to develop a commonsense ("naive"?) understanding of the ongoing interaction.

- Our approach:
  - decompose the “coordination learning problem” in an intuitive way into several learning goals,
  - devise a hierarchical learning architecture to solve sub-problems and
  - integrate results.

- Identification of three *essential determinants* of interaction:
  1. **interdependence modalities**
  2. **opponent behaviour**
  3. **cooperation potential**

- Objective: construction of a *layered learning architecture* that integrates learning components for these (as an extension of the *InteRRaP* architecture).

- We concentrate on *learning* coordination in societies of purely *selfish* agents:
  - for abstract interaction situations (repeated n-player games),
  - without explicit communication and
  - without any prior knowledge of payoff functions, opponent strategies and opponent decision-making processes.

- Overview:
  1. Interaction Scenario
  2. Intuitive model of the "coordination problem"
  3. The LAYLA agent architecture
  4. Experimental Results
  5. Conclusions

Interaction Scenario

- n-person games in normal form with black-box payoff function (private knowledge of the *Simulation Engine*).

- Simulation procedure in round t:

  1. agents (players) $N = \{1, \ldots n\}$ communicate their action choices (taken from a joint strategy space $S = \times_{i \in N} S_i$) to the Simulation Engine (SE),

  2. SE computes all the resulting payoff $u_i(s_1, \ldots s_n)$ for every agent i,

  3. each agent is notified of the performed joint action $(s_1, \ldots s_n)$ and of the *private* payoff u_i it receives,

  4. round $t + 1$ is started.

- Repeated for a finite number of rounds which is *unknown* to the agents; no knowledge of the payoffs opponents receive.

Intuitive Model of the Coordination Problem

- Starting point: agents as individual utility-maximisers **but** problem of "egoist traps", esp. in the case of non-pareto-optimal Nash equilibria.

- *Socially coherent* behaviour can be defined as $OPT \subseteq S$ where

$$opt \in OPT \iff \text{$u(opt)$ is in the \textit{kernel} of the game}$$

- *What do agents need to know in order to converge to such behaviour?*

- Decomposition of learning problem into sub-problems corresponding to *essential determinants of interaction*

Interdependence Modalities:

- Denote *"what the interaction consists of"* i.e. in which way actors' actions affect each other.

- In repeated n-player games equivalent to learning the utility function.

$\Rightarrow$ Learning task: construct an explicit representation $\pi : S \rightarrow \mathbf{R}$ of agent i's private payoff function u_i

Opponent Behaviour Prediction:

- Important to predict others' future actions to plan strategically.

- Enables reasoning about what the interaction *will* be like (rather than what it *could* be like).

- Learning task: learn a function that can be used to predict any future opponent action sequence on the basis of past joint actions.

Cooperation Potential:

- Difference to opponent behaviour prediction: cooperation potential learning helps to *alter* opponent behaviour rather than only anticipate it.

- Learn to predict own action sequences that will "massage" the opponents into their most cooperative stance.

$\Rightarrow$ Clearly all three learning goals hardly achievable in the presented form, but valuable for defining the overall problem.

The LAYLA Agent Architecture

- Reasoning layers in the InteRRaP agent architecture correspond to the identified learning tasks.

- Idea of the **LAY**ered **L**earning **A**gent architecture: extend each InteRRaP layer by a learning component to attack (simplified versions of) the learning problems.

- Devise concrete learning algorithms for the layers for the specific problem of learning repeated games.

$\Rightarrow$ prototypical **Utility Engine**, **Strategy Engine** and **Social Behaviour Engine**

Utility Engine (L^{IM}): learn an approximation π of the actual payoff function u_i.

- Straightforward supervised learning problem: given joint-action/payoff pairs, approximate the payoff function.

- Employ standard multi-layer feed-forward neural networks that are trained with samples of the form
$$\left\langle \beta(s^{(t)}), \frac{u_i^{(t)}}{\max_{t' \leq t} u_i^{(t')}} \right\rangle$$

- Learning success satisfactory, but disadvantage: neural network design choices handcrafted.

Strategy Engine (L^{OBP}): learn an action-value function $m : S_i \rightarrow [0;1]$ to approximate the expected utility of actions.

- Uses a combination of genetic algorithms and nearest-neighbour learning.

- Trained by using pairs of consecutive opponent action pairs parametrised by the reasoning agent's own action

$$s_{-i}^{(t-1)} \xrightarrow{s_i^{(t-1)}} s_{-i}^{(t)}$$

- Fitness values of individuals depend on their validity with respect to past experience.

- Standard one-point crossover and mutation, wildcard bits; $|S_i|$ populations, one for each action of agent i

- Nearest-neighbour heuristic used to predict next opponent action depending on the previous action.

- Reduction of L^{OBP} to a one-step lookahead.

- By making use of the utility function approximator π a function
$$m(s_i) = \frac{\pi(\tilde{s}_{-i}, s_i)}{\sum_{s_i \in S_i} \pi(\tilde{s}_{-i}, s_i)}$$
can be calculated in each step (given the previous opponent action $\tilde{s}_{-i}$).

$\Rightarrow$ Ideally, m is maximal iff s_i is the (greedy) *best response* to the predicted next opponent action.

Social Behaviour Engine (L^{CP}):

- Learn peer preference structures, the "value" of peers for the agent.

- Use the learned to concepts to approximate the opponent's reasoning mechanism.

- Developed special algorithm for L^{CP} based on *gain models*.
  Idea: approximate two-player payoff dependencies within n-player interactions by

  1. combining worst-case and best-case payoffs for action combinations (s_i, s_j) and

  2. considering the overall *risk* of action s_i.

- *Probabilistic Ordering Models* are used for the approximation of the peer's gains.

- Recursive reasoning down to "level 3".

- Line of social reasoning:

  1. Assess the value of "help" that is provided to i by peer j by particular strategies of j and vice versa.

  2. Use 1. to compute the probability with which j will play any s_j if i plays any s_i.

  3. Use 2. to determine the *expected gain* $g_i(s_i)$ of every action s_i.

  4. Construct the set of *socially feasible actions*

     $L_j = \{s_i | m(s_i) + \gamma \cdot g_i(s_i) > max_{s_i'} m(s_i')\}$
     (*compromise factor* $\gamma \in [0; 1]$).

  5. Repeat 1.-4. for every peer j in a neighbourhood $N_i \subseteq N - \{i\}$.

  6. Construct the union of all socially feasible action sets $L = \bigcup L_j$.

     If empty, play according to m_i.

     Else choose that $s_i \in L$ that (allegedly) maximises opponents' expected gains.

- No built-in cooperativeness, but ability to detect cooperation potentials and notion of reciprocity.

- Integration of learning layers:

  - *downward commitment*:

    whenever compromise is possible, greedy choices are overruled;

    Utility Engine exploration action choices can be overruled by the Strategy Engine;

  - *upward activation*:

    learning of super-layer does not start until sub-layer makes sufficient progress;

    the Utility Engine and Strategy Engine perform supervised learning, so current errors can be measured
    $\Rightarrow$ use of thresholds

Experimental Results

- Application scenario:

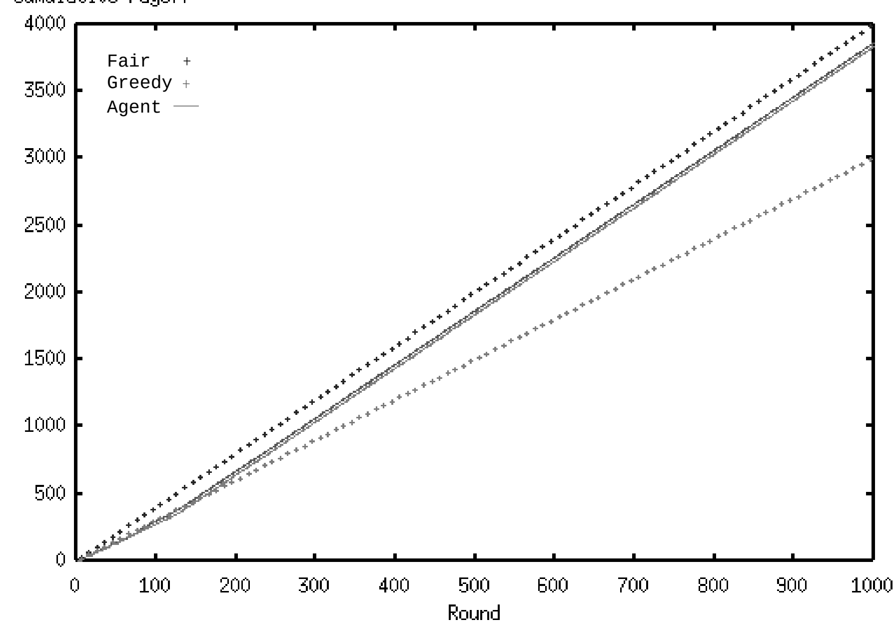
  resource-load balancing

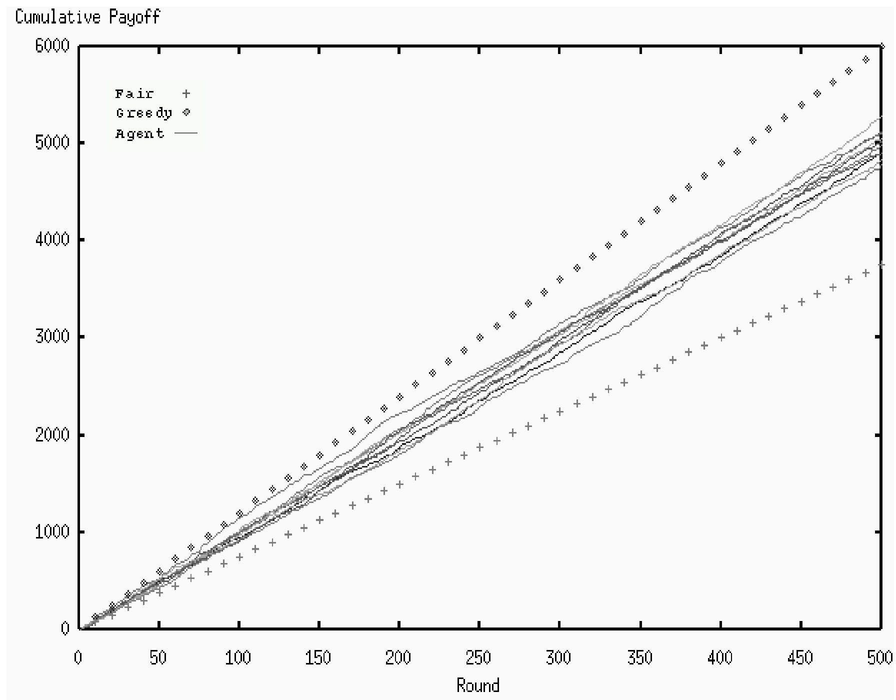
- Special properties:

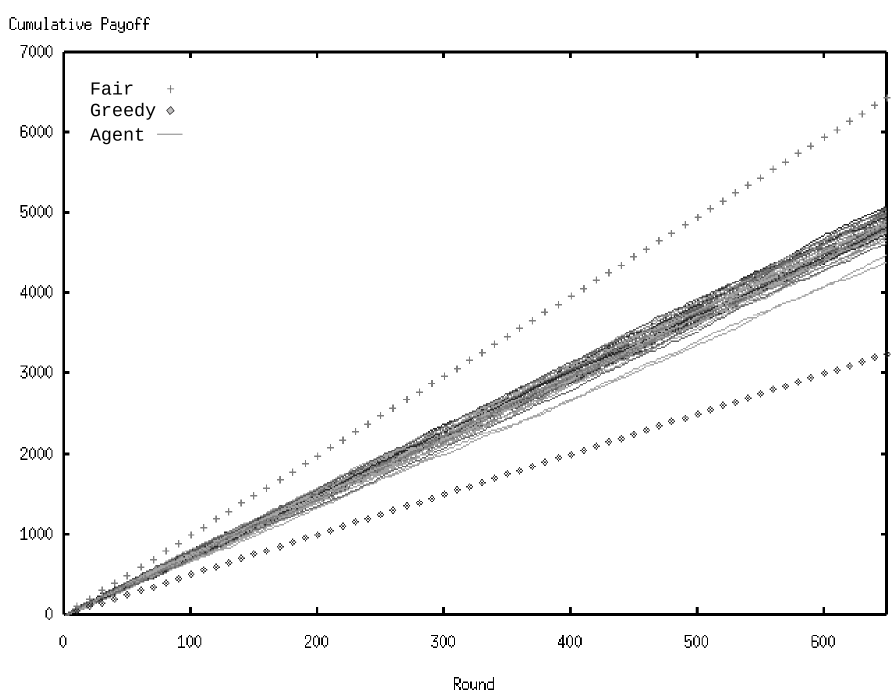
  1. A single strict Nash equilibrium that is not collectively rational ("greedy" action combination).

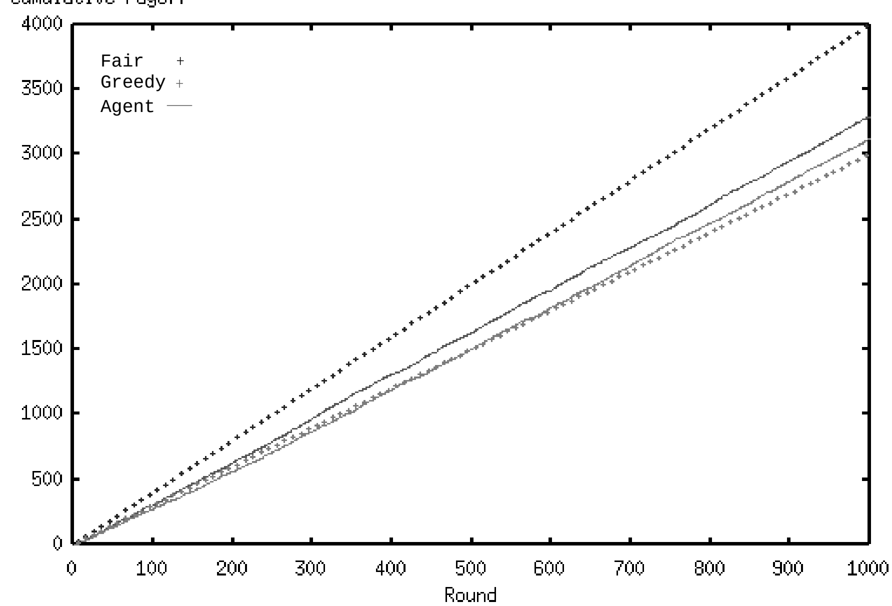
  2. Several "fair" resource allocation strategies, that provide higher payoffs to *all* agents than the equilibrium

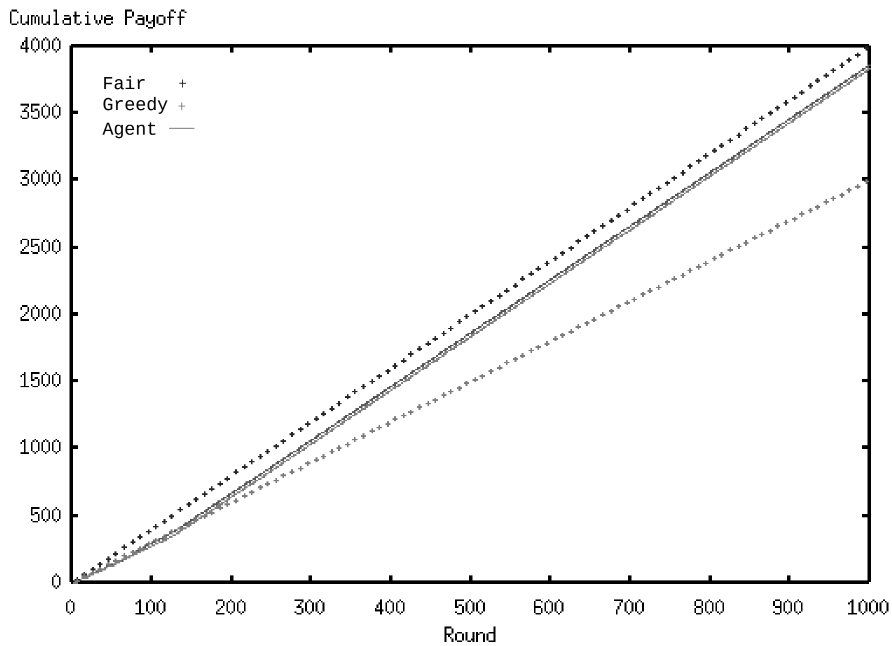
- Tests in two-player two-resource, ten-player five-resource and fifty-player five-resource settings.

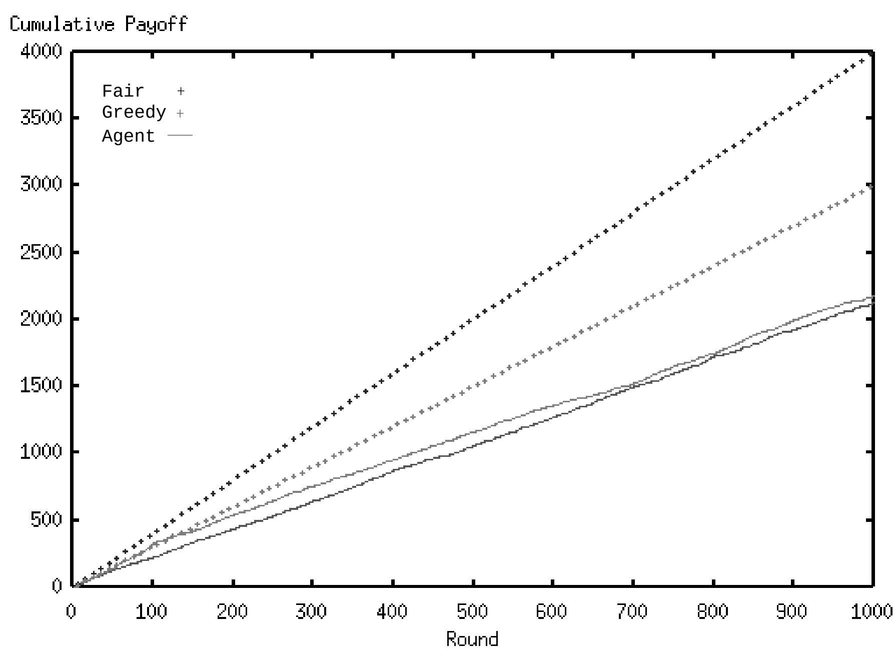

Fair
Greedy
Agent
Round
0
500
1000
1500
2000
2500
3000
3500
4000
100
200
300
400
500
600
700
800
900
1000


Cumulative Payoff
Fair
Greedy
Agent
Round
0
1000
2000
3000
4000
5000
6000
50
100
150
200
250
300
350
400
450
500


Cumulative Payoff
Fair
Greedy
Agent
Round
0
1000
2000
3000
4000
5000
6000
7000
100
200
300
400
500
600


Cumulative Payoff
Fair
Greedy
Agent
Round


Cumulative Payoff
Fair
Greedy
Agent
Round


Cumulative Payoff
Fair
Greedy
Agent
Round

Conclusions

- Selfish agents *can* learn to behave cooperatively
  - in games in which it is tempting to defect,
  - without being able to communicate,
  - with very little prior knowledge and
  - (although things get harder) even in games with *vast* strategy spaces.

- Layered learning offers the possibility to decompose hard problems into simpler ones but
  - the decomposition itself was not "learned",
  - the architecture is only relevant in the context of repeated games and
  - learning algorithms were tuned to match the needs of the application scenario.

- Drawbacks:
  - dependence of success on appropriate choice of compromise factor γ,
  - lack of meta-reasoning capabilities,
  - lack of dependable multi-agent learning and layered learning theory to compare our results with and
  - high complexity.
- Open issues:
  - Extensive form games,
  - non-game-theoretic interaction models
  - use of communication and
  - meta-learning.