Hierarchical Common-Sense Interaction Learning

Michael Rovatsos
Knowbotic Systems
Daimlerstraße 32, 60314 Frankfurt, Germany
rovatsos@knowbotic-systems.com

Jürgen Lind
Multiagent Systems Group, DFKI
Im Stadtwald, 66123 Saarbrücken, Germany
lind@dfki.de

Abstract

We describe a hierarchical learning approach for effective coordination in repeated games based on a common-sense decomposition of the "coordination problem". In contrast to most other research on mechanism design and game-learning, we concentrate on breaking down the top-level problem into simpler learning tasks concerned with learning (i) utility functions, (ii) best-response strategies and (iii) cooperation potentials. We also report on empirical results with the layered learning architecture LAYLA *that is constructed using these sub-components in a resource-load balancing scenario. The positive results show that the approach deserves further investigation, although a number of (possibly problem-inherent) difficulties illustrate the limitations of learning approaches in real-world applications.*

1. Introduction

The study of coordination assumes a prominent position within most of current DAI research, because the ways in which autonomous agents situated in co-inhabited worlds manage their interactions determine the global system behaviour that will evolve within that society. In the light of the dynamic nature of distributed open systems and considering the difficulties in predicting (from a designer point of view) the society-level phenomena on the grounds of microscopic inter-agent interactions, it seems particularly important to develop architectures and algorithms that will allow agents to *learn* how to coordinate rather than use fixed, built-in coordination mechanisms.

While game-theoretic models have been widely used to study interaction phenomena, they have mostly aimed at solving coordination problems only indirectly, because no reasoning is conducted about the interaction *itself* by the agents. Instead, mathematical analysis of decision situations (as in the area of *mechanism design* ([4] provides an overview)) or data-driven adaptation of learning algorithms (especially in work on *game learning*, cf., e.g., [5]) is employed to yield the desired behaviour.

In our work, we followed a different approach. We investigated the possibilities of learning how to interact effectively in repeated *n*-player games on the grounds of developing a common-sense (one could even say "naive") understanding of the interaction. To this end, we suggested an intuitive decomposition of the "coordination learning problem", and a hierarchical learning architecture was devised that would facilitate the learning of essential sub-tasks necessary to solve this problem by integrating different learning components.

To ensure that the observed long-term evolution of cooperative, better-than-equilibrium behaviour is solely due to the interaction learning and reasoning that takes place, we apply very rigorous assumptions to the experimental environment: we consider only purely self-interested agents (individual utility-maximisers) to rule out the possibility of "built-in" cooperativeness, require that agents have no prior knowledge of the game and of opponents' strategies whatsoever and exclude the possibility for an agent of observing what payoffs other agents receive. Additionally, we prohibit the use of communication (see also [7]) as a means of exchanging information, so that agents must make decisions only on the basis of local payoff information and action observation.

Experiments with the layered learning architecture (that is based on the hybrid InteRRaP [11] architecture) prove that cooperative behaviour can evolve as "social rationality" [9] within societies of purely selfish individuals if they are provided with means to reason about the *essential determinants* of interactions in a common-sense fashion.

The remaining sections are organised as follows: in Section 2, an outline of the experimental setting is provided. Section 3 introduces the principles of decomposing the "coordination problem" in an intuitive way, thus providing the main directions for the design of the layered learning agent architecture LAYLA, that is described in more detail in Section 4. This is followed by a description of empirical results in Section 5 that were obtained from experiments in

a resource-sharing scenario, and Section 6 rounds up with conclusions and directions for future work on the subject.

2. Interaction scenario

2.1. Repeated normal form games

Repeated *n*-player games in strategic form [1] can be seen as a "normal form" of coordination problems: they define repeated interactions between several agents in which these agents choose particular actions in each round and, as a consequence, receive some numerical payoff. Thereby, the obtained payoffs are assumed to represent the utility that agents assign to the outcome of the joint action taken simultaneously by all agents.
Formally, such a game can be defined as follows:

Definition 1 *An* n**-player game in normal form** $\Gamma = \langle N, S, u \rangle$ *is defined by:*

- *the* **set of players** $N = \{1, \ldots, n\}$,
- *a finite, non-empty* **set of strategies** $S_i = \{s_{i1}, \ldots, s_{im_i}\}$ *for each player* i. *The Cartesian product of all players' strategy sets* $S = \times_{i \in N} S_i$ *is called the* **strategy space** *of the game, and a conjoint strategy selection of all players* $s = (s_1, \ldots, s_n) \in S$ *is called a* **joint strategy**.
- *a* **payoff function** $u : S \rightarrow \mathbf{R}^n$ *which assigns a payoff vector* $u(s) = (u_1, \ldots, u_n)$ *to each joint strategy* s, *such that* u_i *is the payoff that player* i *will receive.*

Games that are of particular interest are games that reflect "social dilemmata", such as the Prisoners' Dilemma [2]. Such games are typically characterised by the fact that the sets of their Nash equilibria and pareto-optimal solutions (cf. [6] for definitions of these terms) are disjoint. In such games, choosing the best reply to any opponent strategy may be sub-optimal to all players, so that even self-interested agents behave irrationally if they greedily seek to maximise their worst-case payoff.

In our resource-load balancing application scenario, we have developed a payoff function that has a single, strict Nash equilibrium (by which all agents greedily access all resources simultaneously, which then become helplessly overloaded) and several pareto-optimal resource allocation strategies that are beneficial to all players (the case in which agents reach fruitful compromise about how to distribute resources efficiently).

2.2. Game simulations

In the experimental system we use to evaluate our architecture, agents choose some particular strategy (action) in each round simultaneously and communicate it to the Simulation Engine, the sole system component that has knowledge of the payoff function. Once all agents' choices have been collected, their execution is implemented by the Simulation Engine, and the individual payoffs are distributed to the agents as well as information about all agents' choices. Thereby, it is the ultimate and only goal of each agent to maximise the utility it receives over time.

Noise may come in (in the form of Gaussian perturbations) at any point in this procedure, i.e. in action execution, payoff perception and action perception (of other agents' actions). Thus, every possibility of effectory and sensory failure can be simulated, so that a realistic interaction environment is realised.

To start off with, agents have no knowledge whatsoever about the effects of their choices and they have no information about the future behaviour of their peers. Also, they never perceive what payoffs other agents obtain in the course of the game. The motivation behind this model is to restrict the scope of available information to "seeing what others do and noticing how happy I am with it" so that a minimal set of percepts can be determined that suffices for the agent to build an understanding of the concrete interaction environment and to learn to behave optimally in this environment.

3. Intuitive model of the coordination problem

Informally, an agent is coordinating its local actions perfectly with those of its co-actors, if it can maximise the utility it obtains from interactions. However, individual utility maximisation can lead to "egoist traps", if agents fail to recognise potentials for mutually beneficial cooperation, i.e. the existence of pareto-optimal strategy combinations that are more profitable than best-response equilibria for at least one player, while none of the remaining players is at a disadvantage.

In the context of repeated one-stage games, we could thus characterise a set of joint strategies $OPT \subseteq S$ as *socially coherent*, if and only if the payoff vector $u(opt)$ of any $opt \in OPT$ is in the kernel[1] (cf. [4] for definitions) of the game Γ, remembering that the kernel requires payoff distributions to be both *individually* rational (in that the optimal coalition guarantees to each participant at least as much payoff as she could gain by her own strength) and *socially* rational (in that it provides the coalition as a whole with at least as much payoff as any other possible coalition). While this construction might seem questionable as a concept of "socially rational behaviour" in the general case, it provides us with a simple working definition for it in the

[1] For the games we consider, we will assume that this kernel is non-empty.

realm of repeated games among selfish agents. In particular, it defines a set of optimal behaviours against which we can measure the actual performance of our learning architecture.

Given the top-level task of the society to converge to strategies in OPT, one may proceed and ask how this can be achieved among agents devoid of communication and cooperative attitude that are not controlled by some central authority. Our approach to solve this problem was as simple as to ask: What does an agent need to *know* to reach this goal, and how should it *use* this knowledge?

It was found that a very intuitive decomposition of the problem, as it would seem natural for humans, suffices to solve it. It is based on the idea that any interaction situation is characterised by the so-called *essential determinants*: the modalities of interdependence, opponent behaviour and cooperation potential.

Modalities of interdependence describe what the interaction consists of, i.e. what actions agents have at their disposal and how the outcomes of those actions depend on each other. In other words, knowledge about interdependence modalities is knowledge about "what would happen to agents $a_1, a_2, \ldots a_n$ if they concurrently performed actions $s_1, s_2, \ldots s_n$ in situation S". In game-theoretic models, these interdependencies are captured by the payoff function (which does not depend on any situation context). Therefore learning them implies learning the payoff function, i.e. a mapping from the joint strategy space to the utilities the learning agent receives.

Formally, the interdependence modalities learning task L^{IM} in repeated games is completed, if an explicit representation $\pi : S \rightarrow \mathbf{R}$ of agent i's private payoff function is learned that is identical to the true (unknown) payoff function of the game:

$$\forall s \in S. \quad \pi(s) = u_i(s) \qquad .$$

Opponent behaviour is a concept that is somewhat complementary to that of interdependence modalities, since it determines what the enacted interaction *will* be like given the strategies that opponents pursue. Quite naturally, it is important for an agent not only to know how its opponents might affect its own standing in theory, but to be able to infer in one way or another what choices they will make in the future (knowing that the agent's own success depends on the future actions of opponents). In the case of repeated games, such opponent behaviour might for example be characterised by some meta-strategy such as the *maximin* principle, or it might simply be some fixed pattern of behaviour that is implemented by the opponent "mechanically".

Formally, an opponent behaviour prediction learner ideally would solve the task L^{OBP} by learning a function

$$f : S^{t_1} \times S_i^{t_2} \rightarrow S_{-i}^{t_2}$$

(where S^t denotes any t-long sequence of strategies), i.e. f is a function that predicts for any sequence S^{t_1} of joint strategies already played and any future sequence $S_i^{t_2}$ of strategies played by agent i (the learner) what strategies ($S_{-i}^{t_2}$) the opponents of i will play in the next t_2 rounds.

Hence, the perfect OBP learner would be able to predict what others will do for any future sequence of own strategy choices and any past set of action observations.

The *cooperation potential* of an interaction situation, finally, is the most complex of the three proposed concepts. Basically, it combines the knowledge of the interdependencies and opponents' anticipated behaviour to reason about whether and how the interaction situation can be exploited to the benefit of all participants, i.e. whether there are ways to achieve fruitful cooperation. Knowledge about such potentials can be used to "massage" the opponent into its most cooperative stance by finding ways to *alter* opponent behaviour in the long term. So while opponent behaviour is considered to be fixed when predicted, a cooperation potential learner would strive to determine the reasoning principles of an opponent and how they might be used to its own benefit[2].

It is a quite complex undertaking to define the corresponding learning task L^{CP} formally. We have found the definition of a function

$$h(S^{t_1}) = (s_i^{(t_1+1)}, \ldots, s_i^{(t_1+t_2)}) \iff$$
$$\forall j > t_1 + t_2 . \exists opt \in OPT . s_{-i}^{(j)} = opt_{-i}$$

to be a suitable target function for the cooperation potential learner. S^{t_1} denotes a past sequence of t_1 joint actions as before, and $s_i^{(t_1+1)}, \ldots, s_i^{(t_1+t_2)}$ is a sequence of subsequent actions of agent i until round t_2. For any past sequence of joint strategies, the function h will return a sequence of own future action decisions that will make the opponents $N - \{i\}$ play their part in some optimal strategy *opt* forever after round t_2 (one may additionally require that t_2 be minimal).

Thus, h will suggest a series of future actions for the learner i that will make others behave optimally, so that a perfect coalition will be established (and never be left again) forever after.

It is quite clear that the three target functions defined for the learning sub-problems are probably unsolvable in their strict formulation. However, they provide us with rigid mathematical definitions of problems so that we can devise approximate solution algorithms for them. The particular machine learning algorithms we use in our prototype con-

[2]The distinction between opponent behaviour and cooperation potential is best illustrated by considering sub-intentional opponents, e.g. environmental variables that exhibit a fixed behaviour and are devoid of any reasoning capabilities – in their case, cooperation potential learning is useless, because there is no possibility of influencing their "choices".

stitute the central part of the system and will be described in more detail in the following paragraphs.

4. The LAYLA agent architecture

The Layered Learning Agent architecture is a rather simple instance of the more generic view of layered learning laid out in [13] (which, in turn, borrows a lot from the methodology put forward in [16]) especially designed for game-learning players. It is and extension of the hybrid InteRRaP architecture in the sense that the three InteRRaP layers are extended by learning components, one for each of the learning tasks described above.

4.1. Extending InteRRaP to a layered learning architecture

The original InteRRaP [11] architecture is based on the idea that each agent consists of three layers, the lowest of which implements low-level situation-action patterns of behaviour and also manages the agent's sensing and action (*Behaviour-Based Layer*), while the intermediate layer encapsulates the agent's long-term planning capabilities and individual goals (*Local Planning Layer*). The topmost layer (*Social Planning Layer*) is concerned with negotiation, communication and coordination, i.e. any social activities of the agent.

The idea in LAYLA is to add a *learning component* to each of the three layers in compliance with their abstraction levels. For the particular task of game-learning, this can be done in a straightforward manner: the L^{IM} learner is added to the Behaviour-Based layer, because it is concerned with gathering knowledge about the nature and effects of atomic "black-box" actions and is thus at the lowest level of interaction learning. The L^{OBP} learning component, on the other hand, is closely related to "local planning", because predicting opponent behaviour is used to develop optimal strategies (which are equivalent to plans in the context of repeated games). L^{CP}, finally, reflects a social learning process – meta-strategies that are supposed to yield desirable behaviours on the opponents' side are nothing else but social plans that aim at achieving coordinated decision-making in the society.

In the prototypical implementation of game-learning LAYLA agents that we have provided, the respective learning modules are called the Utility Engine, the Strategy Engine and the Social Behaviour Engine, each named after the primary target of their learning activity. In the following paragraphs, we explain what learning algorithms were chosen for the learning layers and how they were integrated to a coherent agent architecture.

4.2. The Utility Engine

The Utility Engine is the simplest of the three learning layers. It uses multi-layer feed-forward neural networks that are consecutively fed "joint action/private payoff"-pairs as learning samples and approximate the agent's own payoff function u_i in the course of the game. We use standard back-propagation as an update algorithm for these networks (cf., e.g., [10]) and provide them with I/O pairs

$$\left\langle \beta(s^{(t)}), \frac{u_i^{(t)}}{\max_{t' \leq t} u_i^{(t')}} \right\rangle$$

in round t which consist of a binary representation $\beta(s^{(t)})$ of the current joint action $s^{(t)}$ and the fraction of the newly obtained payoff $u_i^{(t)}$ divided by the maximally experienced payoff as yet. The conversion of actions into binary tuples is simply a result of fine-tuning decisions in our particular application scenario (cf. Section 5).

Squashing the data for the output layer (which consists of a single unit) to the interval [0;1] is necessary to conform with the output range of common sigmoid units in neural networks. (More specific design choices based on the of the resource-load balancing scenario can be found in [13].)

Quite clearly, the data needed to update these neural nets is readily available from the percepts that are issued by the Simulation Engine, so each new round that is played provides the Utility Engine learning algorithm with a new sample. The training strategy consists of training the network on all samples yet received once after each round.

4.3. The Strategy Engine

The Strategy Engine uses a combination of genetic algorithms (GAs) and nearest-neighbour learning (cf. [10]) that is similar to that of classifier systems (see for example [14]).

In order to predict future opponent actions, current opponent actions $s_{-i}^{(t)}$ are seen as a consequence of (i) the previous joint action $s_{-i}^{(t-1)}$ and (ii) the previous action of the learning agent i ($s^{(t-1)}$) itself. Such a relationship between two consecutive joint actions can be written as a rule

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

so that, if one sample set D is constructed for every s_i (a total of $|S_i|$ sample sets), observed pairs of subsequent opponent actions can be stored as symmetric bit-strings

$$\beta(s_{-i}^{(t-1)}) \cdot \beta(s_{-i}^{(t)})$$

in round t as an instance in the sample set $D(s_i^{(t-1)})$).
The idea is now to construct $|S_i|$ corresponding GA populations $G(s_i)$ of some pre-defined size, and to allow them to reproduce according to the fitness function

$$fitness(h) = \left(\frac{1}{|D(s_i)|} \sum_{d \in D(s_i)} mv(h,d) \right)^2$$

where $mv(h,d)$ is a "match value" from the interval [0;1] that is proportional to the number of matching bits between any hypothesis $h \in G(s_i)$ and any known sample $d \in D(s_i)$ (again, we refer the interested reader to [13] for details).
In running the GAs, we use standard one-point crossover and standard fixed-probability mutation. Additionally, we allow for wildcards # in GA bit-strings with "don't care"-semantics, so that beyond the simplistic "if $s_{-i}^{(t-1)}$ then $s_{-i}^{(t)}$"-rules generalized statements can be generated about opponent behaviour.

The nearest-neighbour method comes in at the point at which we want to use the trained populations to predict the next opponent action. This is necessary because we cannot expect the precondition part of any individual in the GA population to match the actual previous joint action completely. We therefore compute the distance between individuals (bit-strings) h and the actual previous joint action $s_{-i}^{(t-1)}$ as

$$distance(h, s_{-i}) = \sqrt{\sum_{l=1}^{|pre(h)|} \chi(h_l, \beta(a_{-i})[l])^2}$$

where

$$\chi(h_l, \beta_l) = \begin{cases} 0 & \text{if } h_l = \beta_l \vee h_l = \# \\ 1 & \text{else} \end{cases}$$

is the distance between two bit values, which is zero if they are both equal or if the value of h in bit l is '#'; $|pre(h)|$ is simply the number of bits in the "first half" of bit-string h.
Using this function, we can determine the n nearest neighbours of $s_{-i}^{(t-1)}$ in every population $G(s_i)$ and take a majority vote amongst their postconditions to determine the most probable value of every bit in the next opponent action.
Since this is always done for every s_i, we can compare the opponent actions that are predicted under any choice of s_i with respect to the payoffs that would be achieved if the predictions actually occurred by using the π-approximator of the Utility Engine. This allows for the computation of an individual action-value function

$$m : S_i \to [0;1]$$

defined by

$$m(s_i) = \frac{\pi(\tilde{s}_{-i}, s_i)}{\sum_{s_i \in S_i} \pi(\tilde{s}_{-i}, s_i)}$$

where $\tilde{s}_{-i}$ is the *predicted* next opponent action, provided that i plays s_i (the denominator is simply a normalising constant).

If the prediction of next opponent actions is accurate enough, a greedy player i may simply choose to play that s_i for which m is maximal, because this particular s_i is nothing but the best-response strategy. However, we have argued before that such individual best-reply behaviour might not be socially rational, and this calls for the construction of a third component on top of the Strategy Engine.

4.4. The Social Behaviour Engine

Social reasoning in LAYLA is guided by the following ideas (given that agent i is conducting the reasoning):

1. Assess the value of "help" that is provided to i by peer j by particular strategies that agent may choose.

2. Likewise, determine how valuable certain actions of i might be for j.

3. On the grounds of 1. and 2., compute that probability with which j will play s_j if i plays s_i, for every $s_i \in S_i$ and $s_j \in S_j$.

4. Determine the *expected gain* $g_i(s_i)$ of every action s_i taking into account the probabilities of all actions s_j under the assumption that s_i will be played by using the results of 3.

5. If there exist actions s_i for which

   $$m(s_i) + \gamma \cdot g_i(s_i) > max_{s_i' \in S_i} m(s_i')$$

   holds, include them in the set of socially feasible actions L_j ($\gamma \in [0;1]$ is the so-called *compromise factor*).

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

7. Construct the union of all socially feasible action sets $L = \bigcup_{j \in N_i} L_j$. If it is empty, choose $\arg\max_{s_i \in S_i} m(s_i)$ (the greedy best-response action) to be played in the next round. Else, choose $s_i^* = \arg\max_{s_i \in L} \sum_{j \in N_i} g_j(s_i)$ to be played in the next round, i.e. that action that maximises the sum of *peers'* expected gains g_j.

Although the technicalities of the underlying formalisms that are used to perform these steps are far too complex to be described here in detail (cf. [13] or [12] for a full account), we shall attempt to make some points more precise. Steps 1. and 2. are essentially conducted by using *gain models*, i.e. approximations of two-player payoff dependencies in n-player games. These gains are computed by combining best-case and worst-case payoffs obtained under two-player action combinations (s_i, s_j). The amount of "help" that is granted by a certain action s_j to i if it is playing s_i is then computed by comparing the outcome of (s_i, s_j) for i to that of the worst alternative that j might have chosen instead of s_j. Also, the overall risk that i runs when playing s_i is taken into account.

To determine the conditional probabilities in step 3., recursive gain models are combined (down to "level 3" in the terminology of [17]), because it is clear that reasoning about how one might help the other (if the other helped him in turn, if it had helped the other in the first place, etc.) is inherently recursive.

While own gain models can be gradually obtained through payoff and action observation, peer gain models have to be guessed. To this end, we have devised a formalism called Probabilistic Ordering Model (POM) together with appropriate update functions that enables agents to reason about the *orderings* that govern opponent gain models, rather than try to determine the precise quantities in the (s_j, s_i) matrices (again, see [12] for details).

Step 4. is pretty straightforward: in the same fashion as expected payoffs are usually computed, expected gains can be derived from the gain models and the conditional probabilities obtained in 3.

Step 5. constitutes the central "social reasoning" step in the procedure. It allows strategies to be considered for future decisions if their individual action-value m plus their expected gain (weighted by a compromise factor) is higher than the individual action-value of the "greedy" action alternative. Thus, compromiseful strategies might be selected if the gain expected from the compromise outweighs their sub-optimality with respect to individual action values.

Finally, steps 6. and 7. perform the same reasoning for each peer in some neighbourhood and combine the results in one big set of socially feasible actions. Action selection is ultimately made according to the principle "choose that socially feasible action that is best for most of the neighbours" if L is non-empty. If L is empty, agent i chooses the best-reply strategy that was suggested by the Strategy Engine.

It should be remarked that agents which employ this kind of reasoning do not apply any built-in cooperativeness. They remain self-interested individual utility maximisers, but they additionally have the ability to detect potentials for cooperation. Only if the benefits expected from such cooperation appear to outweigh the possible risks in initiating the cooperation will the agents be willing to deviate from "socially ignorant" best-response behaviour.

Before proceeding to the empirical evaluation of the architecture, we briefly discuss some issues that arise in the process of integrating the three layers.

4.5. Integration

Interaction between the Strategy Engine and the Social Behaviour Engine has already been implicitly discussed in the previous paragraph – whenever compromise is possible, Strategy Engine choices will be overruled by Social Behaviour Engine choices. This line of *downward commitment* is continued toward the Utility Engine in terms of exploration: as soon as payoff prediction accuracy has reached a satisfactory level (as π approaches u_i) exploration choices made by the Utility Engine will be overruled by Strategy Engine choices. On the other hand, *upward activation* is realised in that higher-level learning does not start before lower-level learners have made sufficient progress.

5. Evaluation

To evaluate the performance of the architecture, we test it on a resource-load balancing problem (cf. also [15] and in particular [3]), in which n agents have access to $\mathcal{R}$ resources $R_1, R_2, \ldots R_n$ whose total "value" decreases with the number of accessors. In each round, every agent may choose to access some arbitrary subset of $\mathcal{R}$ and gets some non-negative utility for each accessed resource which depends on the global "load" of that resource.

The payoff function we choose (cf. [13, 12]) exhibits the features that were put forward for "social dilemmata" in the introductory section: a single, strict Nash Equilibrium (the situation in which all agents greedily access all resources in every round) and a (family of) pareto-optimal solution(s) (the situation in which every agent gets a maximal equal share of the resources) which yield a *higher* payoff than the equilibrium for all agents. This brings up the question of how we can make agents "move away" from the alluring equilibrium to reach the much more desirable, pareto-optimal solution.

We have conducted extensive empirical experiments with the LAYLA architecture to analyse its performance. Due to space limitations, we can only report here on the general overall performance that was observed for various problem (game) sizes (an extensive account of other tests can be found in [13]) .

In the actual experiments, it was observed that agents converge to *optimal* behaviour in about 80% of the simulations in the two-player case, while they did substantially

better than equilibrium in larger games. Given that this behaviour emerges in a setting of purely selfish agents that have no prior knowledge of the game this is quite surprising as it proves that cooperation can be learned in principle among socially rational agents. Figure 2 shows sample plots for a two-player two-resource game and ten-player/fifty-player five-resource games. The cumulative agent payoffs are shown as solid curves ("Agent"), while cumulative payoffs of the optimal ("Fair") and equilibrium strategies ("Greedy") that can be determined through a mathematical analysis of the underlying payoff function are shown as dotted lines. A fact that cannot be seen in these plots is that agent behaviour actually continues to improve in the ten/fifty-player case, if only by quantities of 0.2% and 0.05% every hundred rounds, respectively. This illustrates that the learning process actually continues, and given that these are games with strategy spaces of (roughly) size $10^{15}(\approx 2^{5^{10}})$ and $10^{75}(\approx 2^{5^{50}})$ this is an impressive result.

However, these positive results are subject to a very careful choice of γ. Not surprisingly, it turned out that agents who are overtly cooperative or overtly egoistic will fail to achieve optimal cooperation and the fact that we had to derive the optimal compromise factors for the resource-load balancing game mathematically (rather than allow it to be learned by the agents as well) is probably the most severe drawback of the architecture. Here we see a clear need for the development of a meta-reasoning component that will enable agents to adapt their "attitude" to the current situation.

6. Conclusions, future work

This paper provided an overview of our approach on interaction learning that is based on a hierarchical common-sense decomposition of the "coordination problem". For the specific class of coordination problems that are reflected by repeated n-player games, a layered learning architecture was devised that complies with this decomposition. Particular learning algorithms for the individual layers were presented, followed by some remarks on their integration in the LAYLA architecture. Subsequently, experimental results in a resource-load balancing application scenario were reported that proved the adequacy of our approach.

What are the conclusions we can draw from our efforts? Firstly, and most importantly, that the common-sense interaction learning policy implemented by our decomposition principle appears very promising, because it enables purely selfish agents to benefit from (otherwise possibly undiscovered) cooperation potentials. At the same time, it illustrates that in very large, hard games we should look for *satisficing* rather than optimal behaviour – their huge strategy spaces make it impossible to converge to optimal patterns of behaviour within a reasonable numbers of interactions. Yet it is obviously possible to do much better than equilibrium within very little time, and "continuous (although sometimes only marginal) improvement" throughout the interaction can be ensured. This is quite reassuring considering the complex character of the underlying interactions.

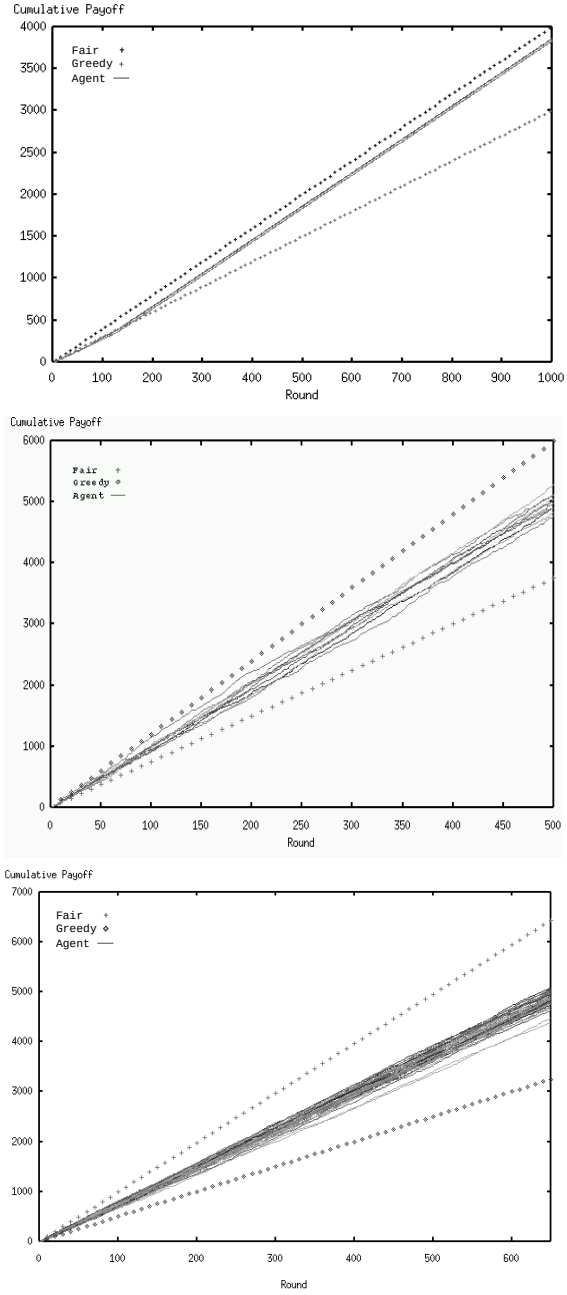

Figure 1. Performance plots for a two-player simulation (top), a ten-player game (middle) and a fifty-player society (bottom).

We have already discussed the need of a meta-layer to make the interplay between the learning layers in LAYLA more flexible and also to make the on-line adaptation of parameters possible (especially that of γ). Other benefits that could be accrued from adding such a component to the system would include the on-line management of reasoning resources, adaptation to changes in the game (which is not yet possible) and a more flexible control of exploration policies, social attitudes and non-determinism in action selection. These are certainly issues that should be looked at in the future.

Another possible direction for future work would be to evaluate system performance for different (classes of) games, to extend the approach to settings in which communication is available, or even to allow for sequential, stateful interactions.

In our opinion, research towards further concepts of common-sense reasoning about interaction and interaction learning is needed. It might include (but is not limited to) an analysis and adaptation of concepts from the social sciences (such as social metaphors) and symbolic models of interaction knowledge and inference that can be used to explicitly reason about coordination in con-inhabited environments.

We believe that what is still missing in DAI research is a *social level characterisation* [9] of intelligent systems that describes global system behaviour on the grounds of local agent interactions – our work can be seen as a contribution to these.

References

[1] R. J. Aumann, M. B. Maschler. *Repeated Games with Incomplete Information*. MIT Press, 1995.

[2] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[3] C. Bichhieri, M. E. Pollack, and C. Rovelli. The potential for Evolution of Cooperation among Web Agents. *Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems* (ed. Sen, S.): 6-11, 1996.

[4] K. Fischer, C. Ruß, and G. Vierke. *Decision Theory and Coordination in Multiagent Systems*. Research Report RR-98-02, German Research Center for Artificial Intelligence, DFKI GmbH, 1998.

[5] Y. Freund, D. Ron, M. Kearns, R. Rubinfeld, Y. Mansour, and R. E. Schapire. Efficient Algorithms for Learning to Play Repeated Games Against Computationally Bounded Adversaries. *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, 1995.

[6] D. Fudenberg, J. Tirole. *Game Theory*. MIT Press, 1991.

[7] M. R. Genesereth, M. L. Ginsberg, and J. S. Rosenschein. Cooperation without Communication. *Proceedings of the 5th National Conference on Artificial Intelligence* (eds. Kehler, T. and Rosenschein, S.), vol. 1: 51-57, Morgan Kaufmann, 1986.

[8] P. J. Gmytrasiewicz. An Approach to User Modeling in Decision Support Systems. *Proceedings of the Fifth International Conference on User Modeling*, User Modeling, 1996.

[9] N. R. Jennings, J. Campos. Towards a Social Level Characterization of Socially Responsible Agents. *IEEE Proceedings on Software Engineering*: 11-25, 1997.

[10] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[11] J. P. Müller. *The Design of Intelligent Agents: A Layered Approach, Lecture Notes in Artificial Intelligence*, vol. 1177, 1996.

[12] M. Rovatsos, J. Lind. Learning Cooperation in Repeated Games. *Proceedings of the Workshop on Agents learning about, from and with other Agents (IJCAI-99), Stockholm, Sweden* (ed. J. M. Vidal), 1999.

[13] M. Rovatsos. *LAYLA – An InteRRaP extension for Layered Learning in Repeated Games*. Diploma Thesis, Universität des Saarlandes, 1999.

[14] S. Sen, M. Sekaran. Multiagent Coordination with Learning Classifier Systems. In *Lecture Notes in Computer Science*, vol. 1042, Springer, Berlin.

[15] A. Schaerf, Y. Shoham and M. Tennenholtz. Adaptive load balancing: A study in multi-agent learning. *Journal of Artificial Intelligence Research*, (2):475-500, 1995.

[16] P. Stone. *Layered Learning in Multi-Agent Systems*. Ph.D. Dissertation, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. Available as technical report CMU-CS-98-187.

[17] J. M. Vidal, E. H. Durfee. Learning nested models in an information economy. *Journal of Experimental and Theoretical Artificial Intelligence: Special Issue in DAI Systems*, 1998.

[18] J. Weibull. *Evolutionary Game Theory*. MIT Press, 1995.