Learning Cooperation In Repeated Games

Michael Rovatsos and Jürgen Lind

German Research Center for Artificial Intelligence (DFKI)
Im Stadtwald, 66123 Saarbrücken, Germany
e-mail: {rovatsos,lind}@dfki.de
Tel: ++49-681-302-2464
Fax: ++49-681-302-2235

May 17, 1999

Abstract. In the field of multi-agent systems, the study of coordination, cooperation and collaboration assumes a prominent position. Most of the research concerned with these issues concentrates on explicit negotiation between agents, on the investigation of settings in which global *system goals* have to be balanced with agents' *individual goals* or on the exploitation of real-world knowledge to determine efficient coordination strategies. We present a social learning and reasoning component as part of a layered learning agent architecture for iterated multi-player games, which is capable of implementing cooperative behaviour in societies of purely "selfish" agents. This can be accomplished by learning about other agents' preferences, and by finding out how valuable other agents' actions are for the agent's own success. We claim that this is possible without any *a priori* knowledge of the underlying payoff matrices and *without* explicit communication between agents, and first experiments yield promising results.

1 Introduction

In [6], Genesereth et al. investigated the possibilities of effective interaction strategies among *non-benevolent* agents in games without communication. This gave interesting insights to issues such as cooperation and antagonism ([1]), and today the study of effective inter-agent coordination is one of the central areas of research in DAI. According to Jennings ([8]) coordination, i.e. "the process by which an agent reasons about its local actions and the (anticipated) actions of others to try and ensure the community acts in a coherent manner" is perhaps the key problem in DAI. This is due to the fact that, in multi-agent systems, mutual interdependence of action outcomes and global resource limitations require that the agent considers the effects of its actions on its peers and the effects of those peers' actions on its own performance when deciding what to do.
Game theory ([5], [15], [4]) offers a mathematical framework for the analysis of interaction and rationality on an abstract level by mapping the various joint action alternatives to

numerical utilities (payoffs) for each agent, and has hence been widely adopted in DAI by approaches that try to develop suitable techniques for effective agent behaviour in multi-agent environments.

However, most of these techniques either assume that agents have some knowledge of their peers' payoff matrices, or require explicit negotiation to achieve social compromise. In the present paper, we investigate the possibilities to achieve cooperative behaviour among purely "selfish" agents *without* any a priori knowledge of the payoff function. By "cooperative" we mean any coordinated behaviour that ensures a higher payoff to at least one agent while none of the other agents sacrifices payoffs compared to uncoordinated behaviour alternatives. This view is quite different from the "Principle of Social Rationality" that Campos and Jennings have proposed([9]) and which states that "if a member of a responsible society can perform an action whose joint benefit is greater than its joint loss, then it may select that action", because we lift the assumption of the society being responsible (i.e. concerned with balancing individual and system goals) and focus more on heterogeneous societies of anonymous agents with little communication and no explicit system goals. Such environments become more and more interesting when considering applications to Internet agents and electronic commerce, in which agents encounter (possibly non-benevolent) agents they have no knowledge about and with whom they don't share goals. In such settings, *learning* the structure of the underlying games is vital to adapt successfully to changes in the environment, and exact mathematical solution methods fail due to the lack of information.

We have extended the InteRRaP ([12]) paradigm by a three-layer learning component, which enables agents to gather knowledge about (i) their own payoff function (ii) the expected utility of action alternatives and (iii) the payoff structures of peers. All this is achieved in an environment in which only the agent's own received payoff and all players' action choices are communicated to the agent after each round. The acquired knowledge is then used to help optimize the agent's behaviour in the game. In this paper we concentrate on the component concerned with (iii), the Social Behaviour Engine (SBE), whose functioning we describe in detail.

Section 2 gives a brief overview of the game-theoretic model and of the agent architecture in which the SBE is embedded. In Section 3.1 we introduce Gain Models, Probabilistic Ordering Models and Recursive Belief Models, the basic building blocks of the SOCCER algorithm that the SBE employs to perform social reasoning and which is presented in Section 3.2. An evaluation of our model's performance in a resource load balancing application is outlined in Section 4 and Section 5 concludes with a discussion of open questions that will be looked at in future research.

2 System Overview

2.1 Experimental Environment

The experimental environment in which agents interact in our setting is given by an iterated n-player game in normal form. In such a game, each agent receives some numerical payoff after each round which corresponds to the utility it assigns to the outcome of the joint action just performed by all agents including itself. We adopt the following definition for this game (cf. also [4], [15]):

Definition 1 *An n-player* **game in normal form** $\Gamma = (P, A, f)$ *is defined by:*

- *the* **set of players** $P = \{p_1, p_2, \dots p_n\}$
- *the* **set of actions** $A = \{0, 1, \dots, m-1\}$ *available to each player in each round (for the sake of simplicity we shall assume that this set is common to all agents)*
- *the* **payoff function** $f : A^n \rightarrow \mathbf{R}^n$ *that maps each joint action* $(a_1, a_2, \dots a_n)$ *to a payoff vector* $(p_1, p_2, \dots, p_n)$ *such that* p_i *is the payoff for player* i.

We assume that before each round is played, each agent chooses some action from A to perform and communicates this decision to a Simulation Engine (representing the real world). After the collection of all agents' action decisions, the Simulation Engine computes the payoff vector and communicates to each agent only that agent's *own* payoff and the action choices of all other agents, so that agents have no knowledge whatsoever of the payoffs their peers receive. Furthermore, agents have no knowledge of the duration of the game, so to them it is infinite, and only the received payoff per round matters as a utility to be maximized (this implies that no discounting of anticipated future payoffs is possible). It is the sole and ultimate goal of each agent to maximize its own received payoff over time.

2.2 Agent Architecture

Before describing our approach to designing a social-reasoning component, we need to place it in the more general context of a learning architecture for rational behaviour in multi-player games - the **LAY**ered **L**earning **A**gent architecture. This architecture consists of three layers (the Utility Engine (UE), the Strategy Engine (SE) and the Social Behaviour Engine (SBE)) and is an extension of the InteRRaP ([12]) architecture. The UE, SE and SBE correspond to the Behaviour-Based Layer, the Local Planning Layer and the Social Planning Layer in InteRRaP respectively.

The UE's task is to build an approximation $\pi : A^n \rightarrow R$ of p_i's payoff function (since the actual payoff function f_i is unknown to the agent), the so-called *utility model*[1]. The SE uses this information to learn a model of the expected utility of actions, i.e. a numerical measure for the utility of actions to be executed in the next round by the agent. This *strategy model* is represented by a real-valued function $m : A \rightarrow [0; 1]$[2].

The SBE, finally, reasons about the *social cooperation potential* of the game. To this end, it maintains (recursive) models of peers' payoff functions for all opponents in its neighbourhood[3], updates them with information about past moves and payoffs (using the information provided by the UE), and decides on whether there are *socially feasible* action alternatives which ensure a high payoff while making a cooperative stance of the neighbours probable.

[1] In the current implementation, the utility estimate π is modelled by a layered feed-forward neural network using standard back-propagation (cf. for example [11]).

[2] We are currently implementing an SE based on the combination of genetic algorithms and instance-based learning to learn opponent action prediction.

[3] A neighbourhood N_i of player p_i is an arbitrary, non-empty subset of p_i's peers ($N_i \subseteq P - \{p_i\}$).

3 Social Reasoning in LAYLA

Adopting the viewpoint of some particular agent p_i in the society, the decision situation when trying to assess the potential for cooperation (and what it would do to achieve such cooperation) is the following: Agent p_i is basically aware of the fact that any other agent's actions affect its own payoffs. Therefore, it must try to make other agents act to its own favor, and, in turn, it should be willing to compromise and sacrifice some of its own possible benefits to initiate or keep up such cooperation if that seems promising (as e.g. in the Prisoner's Dilemma). Informally, such correlation between the two agents' actions is governed by the equation

$$\begin{aligned}(1)\ p_j\text{'s propensity to help } p_i \times (2) \text{ value of } p_j\text{'s help for } p_i \\ = (3)\ p_i\text{'s willingness to compromise towards } p_j \ (\star)\end{aligned}$$

where (1) reflects the assessment of p_j's behaviour in the course of the game and with respect to how favorable it is for p_i, (2) is a measure of how high p_i values p_j's possible help (i.e. the "power" of p_j towards p_i, the degree to which p_j is able to improve p_i's standing). These two values together determine the degree to which p_i is willing to sacrifice some of its own payoff in order to achieve compromise with p_j to the benefit of both. Note that by "sacrificing some of its own payoff" we mean choosing some alternative that is "safer" than the previous action in the sense that it ensures a possibly sub-maximal, yet more reliable payoff[4]. This notion does *not* bear any implications of "common" or "global" goals, however; we still view an agent as an individual utility maximizer.
But how can p_j obtain a model of (1) and (2)? A peer's propensity to help p_i will certainly depend on whether actions that p_i thinks of as "helpful" are desirable for the peer p_j; it will also depend on whether p_i would help p_j in return, and whether it has the *power* to help p_j at all, given that p_j's standing also depends an the remaining agents $P - \{p_i, p_j\}$. This "power" is nothing but what is expressed by (2), seen from the opposite point of view (i.e. the value of p_i's help for p_j).
It becomes clear that this mutual modeling of preferences and "values of peers" is inherently recursive in a society of rational agents reasoning about their opponents, so we have to decide on a method of coping with this recursive element. The position we assume is that we see rational agents as entities which know about the rationality of their opponents and which also know that they are being modelled as rational entities by their opponents (and know nothing more than that). This may seem a confusing way to put it, but more simply described, the nesting of models goes down to "level 3" meaning that in how p_i behaves towards some neighbour $p_j \in N_i$, p_i will consider:

what p_i knows about p_j,
what p_i thinks p_j knows about p_i and
what p_i thinks p_j knows about what p_i knows about p_j.

This view bears strong resemblance to Gmytrasiewicz' ([7]) Recursive Modeling Method, but its limited maximum nesting depth makes it much easier to handle than recursive models with infinite nesting.[5] The above considerations on two agents p_i and p_j already

[4] As an example, the option to "cooperate" in the Prisoner's Dilemma produces a lower payoff than "exploiting" the opponent, but attempts to "exploit" will not go unnoticed in the long run, thus yielding the poor results of the "defect/defect" joint action most of the time.

[5] Another (admittedly vague) reason for choosing this level of nesting is that we don't expect humans to go into "deeper" recursive reasoning, and humans can still infer "what one likes from what one does".

hint at our strategy of decomposing the analysis of neighbourhoods of arbitrary size into many binary modeling steps, so for the moment we shall only focus on a single agent p_i and the model it tries to build up of some peer p_j in the course of the game (in Section 3.2.3, we shall return to k-sized neighbourhoods by combining the results of several binary interdependency analyses).
In the following sections we introduce formal frameworks for modeling (a) the *value*[6] and (b) the *preference structure* of an arbitrary peer in a multi-player environment: Gain Models and Probabilistic Ordering Models.

3.1 Modeling Binary Interdependencies in n-Player Games

3.1.1 Gain Models

We are looking for an approximation of the value of p_j's actions for p_i, given the payoffs actually received by p_i after each round. It is part of the nature of the game that those payoffs depend on the actions of agents other than p_j, therefore we will first introduce those values for a constant behaviour of all agents except p_i and p_j and then "approximate those other agents away".
Loosely speaking, if p_i intends to play $l \in A$, the value v_l of p_j's action k_1 is larger than that of k_2, if the distance of the respective payoffs to the minimal payoff for p_i under p_j's actions is larger for k_1 than it is for k_2. Thus, if $\pi : A^n \to \mathbf{R}$ is p_i's payoff function estimate,

$$v_l(k_1) > v_l(k_2) \iff \pi(l, k_1, a_{-\{i,j\}}) - \min_{m \in A} \pi(l, m, a_{-\{i,j\}}) > \pi(l, k_2, a_{-\{i,j\}}) - \min_{m \in A} \pi(l, m, a_{-\{i,j\}})$$

if we assume some constant behaviour $a_{-\{i,j\}}$ for the remaining agents[7].
If we were to compute the exact values for v_l for the whole spectrum of $a_{-\{i,j\}}$, the right-hand side of the above definition would have to be replaced by

$$\sum_{a_{-\{i,j\}} \in A^{n-2}} \Pr(a_{-\{i,j\}}) \cdot v_l(k_1) > \sum_{a_{-\{i,j\}} \in A^{n-2}} \Pr(a_{-\{i,j\}}) \cdot v_l(k_2)$$

where $\Pr(a_{-\{i,j\}})$ is the joint probability of the remaining agents' actions. Such values are, of course, highly infeasible to compute, since even the update of the probabilities requires exponential space and an exponential game duration to yield reliable values. We therefore simplify this model to the easy-to-update approximation of *gain*-values defined as follows for any $l, k \in A$:

Definition 2 *Gains*

$$mingain_l(k) = \min_{a_{-\{i,j\}} \in A^{n-2}} \left(\pi(l, k, a_{-\{i,j\}}) - \min_{m \in A} \pi(l, m, a_{-\{i,j\}}) \right) \quad (1)$$

$$maxgain_l(k) = \max_{a_{-\{i,j\}} \in A^{n-2}} \left(\pi(l, k, a_{-\{i,j\}}) - \min_{m \in A} \pi(l, m, a_{-\{i,j\}}) \right) \quad (2)$$

[6] "Value" of a peer and "value of a peer's help" shall be used as synonyms henceforth.

[7] $a_{-\{i,j\}}$ is short for $(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n)$ i.e. the joint action taken by all agents but p_i and p_j (likewise, $a_{-\{j\}}$ is short for $(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$ i.e. the joint action taken by all agents but p_i).

$$risk_l = \max_{a_{-\{i\}} \in A^{n-1}} \pi(l, a_{-\{i\}}) - \min_{a_{-\{i\}} \in A^{n-1}} \pi(l, a_{-\{i\}}) \tag{3}$$

$$gain_l(k) = \frac{\lambda \cdot maxgain_l(k) + (1-\lambda) \cdot mingain_l(k)}{risk_l}, \quad \lambda \in [0;1] \tag{4}$$

Equations (1) and (2) provide measures for the maximum/minimum "gain" in payoffs (measured as the distance to the minimum payoff with respect to p_j's actions) yet experienced (under all joint actions performed by the remaining agents). Equation (3) computes the overall risk for p_i in taking action l, i.e. the difference between the absolute minimum and maximum payoff under l. Finally, equation (4) combines the best and worst possible gains weighted by some "optimism parameter" λ and computes the gain of p_j playing k as that weighted gain combination in relation to the risk of action l.
As an example, if k always yields the absolutely maximal payoff for p_i when p_i plays l (regardless of what others do), then $gain_l(k) = 1$, implying that whatever others do, p_j can always help p_i make the best out of l, so this depicts a situation where p_j is very powerful for p_i.
Now that we have a means of measuring the value of some peer's actions for the agent, we need to model the first part of equation ($\star$): the assessment of p_j's propensity to help p_i.

3.1.2 Probabilistic Ordering Models

A Probabilistic Ordering Model (POM) is a probabilistic approximation of the orderings that govern p_j's preferences concerning the combinations of its own actions and the actions of p_i, i.e. it is a model one agent has of its peer's Gain Model (GM). While Gmytrasiewicz uses a probability distribution over various distinct opponent payoff matrices to model the peer's decision-making process, we encapsulate the uncertainty here in *rank probabilities*. This means that we are only interested in the probability that the value of some (p_i,p_j) action combination (l, k) in p_j's GM takes on a certain ranking position among all (a_i, a_j) rather than in the probability with which it has a concrete numerical value. We believe that such a probability distribution over combined actions and ranks yields a more compact representation of all possible matrix entry orderings than a (probability-weighted) enumeration of all payoff matrices that correspond to those orderings. We first define POMs formally before going into the details of their use.

Definition 3 *A* **Probabilistic Ordering Model** *is a structure $s = \langle A, M, R, r, p\rangle$ where*

- $A = \{0, 1, \ldots, m-1\}$ *is the* **set of actions***,*
- $M : A \times A \to \boldsymbol{R}$ *is a real-valued matrix,*
- $R = \{1, 2, \ldots r\}$ *is the* **set of ranks** *,*
- $r : A^2 \to R$ *is a* **ranking function***, such that*
  1. $\forall a, b, a', b' \in A . r(a', b') < r(a, b) \iff M(a', b') > M(a, b)$*,*
  2. $\forall a, b \in A . r(a, b) = k \Rightarrow \forall k' \neq k . r(a, b) \neq k'$ *and*

r	1	2	3	4
$(0,0)$	0.25	0.25	0.25	0.25
$(0,1)$	0.40	0.29	0.19	0.12
$(1,0)$	0.12	0.19	0.29	0.40
$(1,1)$	0.25	0.25	0.25	0.25

(a) After two add(s,(0,1),(1,0),0.3) operations; the probabilities of the higher ranks for (0,1) have increased to a greater degree than those of low ranks, with opposite effects on the rank probabilities of (1,0). Thus, the algorithm assigns larger probability shifts to those ranks who can cause δ most easily.

r	1	2	3	4
$(0,0)$	0.96	0.02	0.01	0.01
$(0,1)$	0.02	0.92	0.04	0.02
$(1,0)$	0.02	0.04	0.92	0.02
$(1,1)$	0.01	0.01	0.02	0.96

(b) After 20 iterations of adding 0.3 to all binary relations in in the chain $(0,0) > (0,1) > (1,0) > (1,1)$; the POM captures the transitive nature of "$>$" by clearly distributing all ranks among the matrix entries. This is a very interesting property, because we get the transitivity conclusions "for free".

r	1	2	3	4
$(0,0)$	0.44	0.16	0.14	0.26
$(0,1)$	0.03	0.92	0.04	0.01
$(1,0)$	0.01	0.04	0.92	0.03
$(1,1)$	0.33	0.12	0.11	0.44

(c) This is the result of adding 0.3 twice to $(1,1) > (0,0)$ in the POM shown in (b); the model uncovers the violation of the antisymmetricity of "$>$" and reduces/increases the probabilities of extreme ranks for (1,1) and (0,0) much quicker than it changed them in (a). (1,0) and (0,1) remain unchanged.

Figure 1: The POMs shown in (a)-(c) show the results of several (series of) operations on a simple POM with $|A^2| = |R| = 4$; the rows show the probability of each entry of the matrix $A \times A$ assuming rank r (represented by the columns). Initially, all POM-entries are 0.25.

3. $\forall a, b \in A. \exists k. r(a, b) = k$[8]*.*

- $p : A^2 \times R \to [0; 1]$ *is the* **rank probability function***, that assigns to each pair* (a, b) *a non-negative probability with which it assumes position* k *in the ordering of all matrix entries.*

If we take M to be p_j's actual GM matrix and if we choose R large enough to represent any ordering in GM ($r \geq m^2$), then POMs allow for the representation of any probability distribution over the orderings in that GM. Hence, p_i can use evidence obtained by action observations to infer changes in the probability of certain GM orderings by updating the individual rank probabilities of the respective binary action tuples (locally). More specifically, the observations will be used by p_i to increase one of $P((a', b') > (a, b))$ or $P((a', b') < (a, b))$ by some small constant $\delta \in [0; 1]$. Using elementary transformation rules for probabilities and the conditions for r in the POM definition, for any $a, b \in A$ these two probabilities can be computed by using the equation $P((a', b') > (a, b)) = P(r(a', b') < r(a, b))$.
By "reversing" the procedure with which we compute $P(r(a', b') < r(a, b))$ (as the disjunction of individual rank conditions $((r(a', b') = 1 \wedge r(a, b) = 2) \vee (r(a', b') = 1 \wedge r(a, b) = 3 \ldots))$ we can define a POM-update function $\mathsf{add}(s, (a', b'), (a, b), \delta)$ that computes updates for each elementary rank probability $p(r(a, b) = k)$ in such a way that $P((a', b') > (a, b))$ increases by δ in the resulting POM.
Instead of going into the technical details of this construction, we refer the reader to Figure 1 which shows some examples of POMs and operations performed on them to illustrate their behaviour.

[8] These conditions require that each matrix entry (a, b) (i) holds exactly one rank in the ordering and (ii) that this rank is lower than that of some other entry, if the value of the first entry is higher (i.e. the maximal entry in M has rank 1).

3.1.3 Recursive Belief Modeling

We now have shown how an agent p_i can update a model of its own gains with respect to some peer p_j's actions and how the models it has of p_j's GM will be constructed . As mentioned before, we intend to conduct the modeling down to the third level of nesting. Therefore we equip agent p_i's SBE with the following four components for each agent $p_j \in N_i$:

Model	*Function*	*Type*
$s^0(l,k)$	p_i's model of its own gains	GM
$s^1(k,l)$	p_i's model of p_j's gains	POM
$s^2(l,k)$	p_i's model of p_j's model of p_i's gains	POM
$s^3(k,l)$	p_i's model of p_j's model of p_i's model of p_j's gains	POM

The GM s^0 of p_i will be constructed as described in 3.1.1, i.e. $\forall l,k \in A.\forall k \in A.s^0(l,k) = \text{gain}_l(k)$. The update algorithm for it, UPDATEGM will not be described in detail, since it is clear that it can be conducted by simply keeping track of the maximal/minimal payoffs and the corresponding joint actions during the game.
The remaining models will be POMs with an initially equal distribution over all ranks for every matrix entry (we call such a POM an *empty* POM). These POMs will be updated after each round of the game using two different algorithms UPDATEPOM_UNPROFILED and UPDATEPOM_PROFILED, depending on whether the behaviour of the agent whose gains are being modelled is profiled or not, i.e. whether it exhibits some preference for specific actions or not. We now present these two algorithms.

3.1.4 The UPDATEPOM algorithms

UPDATEPOM_UNPROFILED

This algorithm is employed whenever a SBE component other than s^0 has to be updated and the opponent of the agent whose gains are being modelled exhibits an "unprofiled" behaviour, i.e. it shows no preference for any action or, to put it another way, does not show any *intentionality* in its actions. This might be the case if (a) the peer is still busy exploring the outcomes of its actions or (b) if the peer *is* a sub-intentional part of the environment. A possible condition to decide whether some peer behaviour $\text{Pr}^j : A \to [0;1]$[9] is profiled or not is given by $\sum_{k=0}^{m-1}(\text{Pr}^j(k) - \frac{1}{m})^2 > t$ where $t \in [0;1]$ is a *variance threshold* (which is exceeded if the variance to the supposed mean $\frac{1}{m}$ becomes too high).
We explain UPDATEPOM_UNPROFILED informally for a situation in which p_j exhibits an unprofiled behaviour and p_i acts intentionally.
Given this situation, how should p_i's GM be modelled by p_j? Since p_j reveals no preferences through its actions (and is aware of that), it knows that p_i's s^1 it an empty POM. Therefore p_j assumes p_i to act irrespective of p_j's own actions, i.e. p_i will choose that action which provides the highest gain in s^0 regardless of what p_j plays. If p_i just played l, the expected shift for p_i was

$$\sum_{k \in A} \text{Pr}^j(k) \cdot s^0(l,k)$$

[9] Pr is the posterior probability of the peer to play $a \in A$ for every a, i.e. the frequency with which it has chosen that action in the past.

because for p_i, p_j's action probabilities are only distributed according to Pr^j. Assuming that p_i is a rational player, if p_i played l_0 in the previous round, it must have tried to improve its expected payoff when choosing l, which means that

$$\sum_{k\in A} \mathrm{Pr}^j(k)\cdot s^0(l,k) \quad > \quad \sum_{k\in A} \mathrm{Pr}^j(k)\cdot s^0(l_0,k)$$

must hold. Given that this inequality might have been caused by any of the $s^0(l,k) > s^0(l_0,k)$ with probability proportional to the respective $\mathrm{Pr}^j(k)$, we add some small quantity $\mathrm{Pr}^j(k)\cdot\delta$ to the probabilities of all such relations. Therefore, we can perform the operations

$$\forall k\in A.\quad \mathsf{add}(s^2,(l,k),(l_0,k),\mathrm{Pr}^j(k)\cdot\delta)$$

to update s^2 accordingly.[10]

UPDATEPOM_PROFILED

The algorithm just presented covers all cases in which at least one of the two agents exhibits an unprofiled behaviour. If both act like deliberative, rational players, things get much more complicated, because action choice depends on the opponent's expected behaviour, the opponent's behaviour depends on the first agent's behaviour, and so on. Since we have decided to consider this nesting only up to the third step, we can assume a rational choice model for players in which p_i computes the (predicted) probability $E^j(k)$ of p_j taking action k in a two-move sequence $(l_0,k_0)\to(l,k)$ according to

$$E^j(k)=\sum_{l'\in A} s^1((k,l')>(k_0,l_0))\sum_{k'\in A} s^2((l',k')>s^2(l_0,k_0))\sum_{l'\in A} s^3((k,l')>(k_0,l_0))\mathrm{Pr}^i(a_l)$$

. This formula is a direct result of move-to-move rationality of the players (i.e. not choosing an action if its expected gain is lower than that of the action taken in the previous round) combined with the effect of POM s^{k+1} on s^k in predicting the opponent's behaviour and combined with the assumption that "beyond" s^3, agents can only use the posterior action frequencies of their opponent to predict their future behaviour. We will use it as a basis for computing $E^j(k)$ in our top-level algorithm (cf. Section 3.2), and at the same time it is the starting point for UPDATEPOM_PROFILED. Briefly summarized, all UPDATEPOM_PROFILED does is to update the probability of one "$>$"-relation in s^3 according to the action observation, and to propagate the resulting probability changes from s^3 to s^2 and from s^2 to s^1 according to the above equation (we refer the interested reader to [14] for technical details).

In the next section we present a social decision-making algorithm (called the **SOC**ial **C**oordination **E**nabling **R**einforcement algorithm) which suggests a method of implementing the right-hand side of equation ($\star$) formulated in Section 3 by employing the formalisms introduced in this section.

3.2 SOCCER - An Algorithm for Social Reasoning

3.2.1 Social Decision-Making and Compromise

The basic idea behind the notion of "compromise" is to sacrifice some of the maximally possible utility in order to make opponents act in a way that ensures a payoff which may

[10]The same operations would be performed on s^1 and s^3 if p_i was acting in an unprofiled way, and p_j exhibited a profiled behaviour.

be lower than that maximally possible utility but better than the payoff distributed to the agents if they all act greedily.

Intuitively, an agent is willing to compromise or cooperate if it would give away some proportion of the gain it expects for itself for the sake of achieving that compromise. In terms of our formal framework this translates into the principle

> "instead of choosing that action which promises the highest utility for p_i it will also consider other, sub-optimal actions if the expected behaviour of p_j given those actions promised a gain at least as big as the amount of utility sacrificed".

In terms of our recursive gain model system, this can be restated as follows: if p_i and p_j played (l_0, k_0) in the previous round, and if p_i has computed some predicted behaviour E^j for p_j then p_i's expected gain in the next round is

$$\sum_{k \in A} E^j(k) \cdot s^0((l,k) > (l_0, k_0))$$

it p_i intends to play l. Then, if we let $\gamma \in \mathbf{R}$ be a *compromise factor* that expresses how big a percentage of its expected maximal gain p_i is willing to sacrifice, and m is p_i's strategy model obtained from the SE, we can construct the decision function SocDecideNext for p_i as follows:

1. Let $L := \{l \in A \quad | \quad m(l) \geq \max_{l' \in A} m(l') - \gamma \cdot \sum_{k \in A} E^j(k) \cdot s^0((l,k) > (l_0,k_0))\}$.
2. If $L = \emptyset$ then decrease γ until $L \neq \emptyset$.
3. If $\gamma = 0$ and still $L = \emptyset$ choose $l_{next} = \arg\max_{l \in A} m(l)$ to be played in the next round.
4. Else choose $l_{next} = \arg\max_{l \in L} \sum_{k \in A} s^1((k,l) > (k_0, l_0))$ to be played in the next round.

In 1. we construct the set L of all actions whose payoff is greater than the difference between the maximally possible payoff and the γ-weighted expected gain induced by p_j's behaviour (we call L the set of *socially feasible actions* towards p_j). If this set is empty, we decrease γ until L becomes non-empty, which means that we look for actions which enact less compromise, if "we can't be as nice as we would like to be". If γ becomes zero and L is still empty, no compromise is possible, and we choose the "greedy" action in 3.; else, we choose that action in L which maximizes the sum of p_j's (alleged) gains under all of its actions (after all, we want to consider what p_j wants, as well).

One might be tempted to infer, at this point, from the design of this function, that we have circumvented the problem of antagonism by endowing agents with a *cooperative bias*, which inevitably will make them cooperate in practice, because they are designed to do so. However, this is not the case, since our agents still do not pursue any other goal than to maximize their individual utility while their propensity to cooperate springs solely from their ability to detect the potential for cooperation to the benefit of both themselves and their opponent.

```
SOCCER-2(p_i, p_j, s^0, s^1, s^2, s^3, (l, k), (l_0, k_0))
    UpdateActionProbs();
    UpdateGm(s^0);
    IF profiled(j) = false THEN
        IF exploration_needs_fulfilled(i) = false THEN
            Explore();
        ELSE
            UpdatePom_unprofiled(s^2);
            NoCoordDecideNext();
        END;
    ELSE
        IF exploration_needs_fulfilled(i) = false THEN
            UpdatePom_unprofiled(s^1);
            UpdatePom_unprofiled(s^3);
            Explore();
        ELSE
            UpdatePom_profiled(s^1, s^2, s^3);
            SocDecideNext();
        END;
    END;
END
```

Figure 2: The SOCCER-2 algorithm.

3.2.2 The Basic 2-Player Algorithm SOCCER-2

The top-level social-reasoning algorithm is rather trivial and relies on the notions of "exploration needs" and "profiled behaviour". "Exploration needs" reflect the necessity for any agent to try out actions at the beginning of the game until it has a satisfactory picture of its own payoff function. We assume that as long as those needs are not satisfied, the agent doesn't go into any strategic/social reasoning at all, thus behaving seemingly sub-intentionally for its peers. We shall use *exploration_needs_fulfilled* as a testable predicate for this (the testing condition might be, e.g. "has the error in payoff prediction fallen below some threshold?") This notion is crucial to the construction of a social reasoning algorithm, because it provides a clear condition of when to switch the social decision-making component "on" and "off".
The basic SOCCER-2 algorithm for an interaction situation between p_i and p_j is shown in Figure 4 for p_i. It makes use of three functions not mentioned yet: UpdateActionProbs is a simple action-frequency updating function that keeps track of how often p_i and p_j have performed particular actions in the past; Explore() does nothing but to decide what to play next on the basis of the exploration needs of the UE, i.e. it leaves the action decision to the lowest layer. NoCoordDecideNext(), finally, works exactly like Explore() except that it hands the decision control over to the SE when there is no need to coordinate actions with those of others.

3.2.3 Generalisation to Arbitrarily-Sized Neighbourhoods: SOCCER-N

The extension of any of the above SOCCER-versions is quite straightforward: for each agent p_j in p_i's neighbourhood N_i, p_i constructs the set of socially feasible actions L_j (cf. SocDecideNext in Section 3.2.1) separately and chooses l_{next} from $L = \bigcup L_j$ according to

$$l_{next} = \arg\max_{l \in L} \sum_{p_j \in N_i} \sum_{k \in A} s_j^1((k,l) > (k_0, l_0))$$

i.e. we choose that action in L that seems most desirable for the largest possible number of agents in the neighbourhood.

The separate computation of the L_j implies that we allow for various degrees of cooperation with each neighbour, a fact which makes the implicit deal-making highly flexible and adaptive to peers with different "power" and "helpfulness".

4 Evaluation

To evaluate the performance of the constructed SBE, we test it on a resource-load balancing problem (cf. also [10], [3]), in which n agents have access to $\mathcal{R}$ resources $R_1, R_2, \dots R_n$ whose total "value" decreases with the number of accessors. More specifically, the game is defined by $A = \{0, 1, \dots |\mathcal{R}|^2\}$ (for the sake of simplicity, $|\mathcal{R}| = 2^k$ for some $k \in \mathbf{N}$), and by the payoff function f_i for every agent p_i computed as

$$f_i(a_1, \dots, a_i, \dots, a_n) = \sum_{R_k \in \mathcal{R}} \left(\frac{v_k}{\left(\sum_{p_j \in P} \beta(a_j)[k]\right) \cdot T_k} - c_k \right)$$

where $\beta : A \to \{0;1\}^{\mathcal{R}}$ is the binary encoding of actions ($\beta(a_j)[k]$ is the kth component of this encoding), v_k is the value of resource R_k, c_k are the access costs for R_k and T_k is a access time factor increasing linearly with the number of accessing agents (i.e. those agents, whose (binary encoded) action is 1 in the kth component).

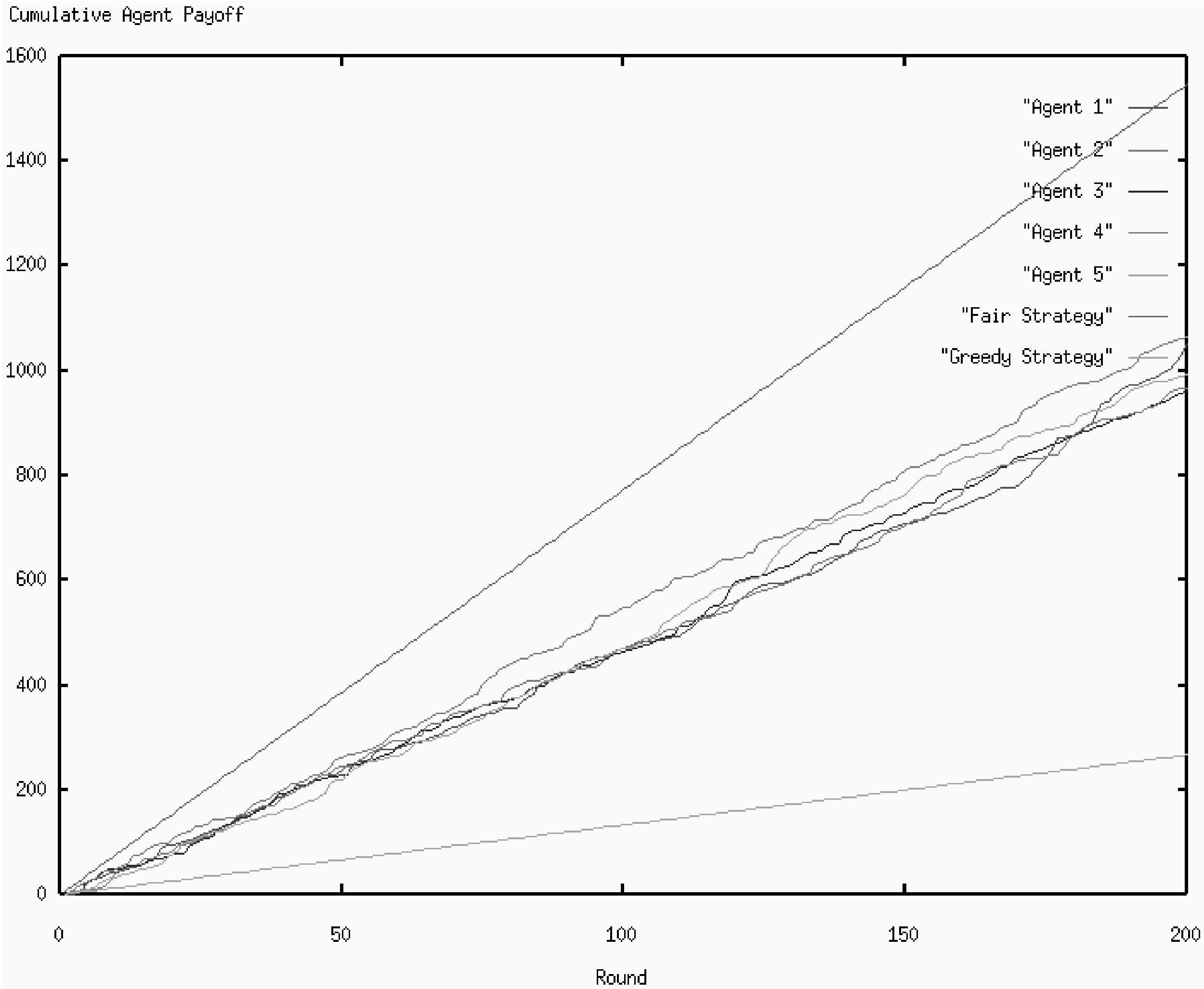
The idea behind this function is that each agent gets the sum of resource values (diminished by the costs of accessing them) for all resources it accesses, but that the value of each resource has to be divided among the agents.

We choose this payoff function for two reasons: firstly, it reflects a class of real-world problems, in which resources, when distributed among several accessors fail to produce their maximal value because of arising coordination costs (e.g. CPU scheduling - the sum of the CPU-time percentages allocated to processes is always below 100% of the total CPU time).

Secondly, it has two interesting features that make it attractive to be used in the context of coordinated behaviour - 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 ([4] provides a good introduction to these game-theoretic concepts).

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.

Figure 3 shows a plot of a sample SOCCER simulation. The simulation was conducted



Cumulative agent payoffs in a SOCCER-based simulation (5 agents, 4 resources): The straight lines show the cumulative payoffs in each round under the fair (=Pareto-optimal) and greedy strategies. The curves in between are the agents' cumulative payoffs. q

on a group of five agents with four resources at their disposal, and each agent had two neighbours. It can clearly be seen that the gradient of agents' cumulative payoffs clearly exceeded that of the greedy cumulative payoff line and approaches that of the "fair", i.e. Pareto-optimal behaviour. Admittedly, this surprisingly good performance is partly due to randomization in action choice, but since the agents' performance is clearly above the mean between "exploiting" and "being exploited", we can safely state that beneficial coordination has evolved. Given that agents started the game with no knowledge of payoff matrices whatsoever, this is a rather impressive result.

It should be mentioned that the agents in this simulation were programmed to be "naive" and to compromise fully with agents that they think of as cooperative. Notions of deception and distrust were not incorporated in agents' reasoning mechanisms, but they surely offer interesting extensions to our architecture.

5 Conclusions, Future Work

We have presented a social learning and reasoning component as part of a layered learning architecture that learns cooperative behaviour in communication-less settings, if the underlying games allow a coordinated behaviour beneficial to all agents. The emergent cooperation is achieved only very gradually and is far from being optimal, but given that agents start the game with virtually no information and that they can only learn from observed peer behaviour, it still offers interesting insights to the dynamics of rational behaviour at a very basic level.
The following questions will have to be looked at in the near future: How susceptible is such reasoning to deceptive agents? Are there ways to deceive peers at all? How does explicit communication affect the learning results? Are the strategies employed by LAYLA agents robust to non-cooperative "deviants"? What successful strategies concerning the use of the compromise factor γ can be devised? To what degree does the accuracy of the UE and the SE affect the results of the SBE?
These and other questions offer many dimensions to evaluate the LAYLA architecture and to extend our learning model to a generic, abstract agent architecture for the study and promotion of emergent cooperation.

References

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

[2] A. Bandura. *Social Learning Theory.* Prentice-Hall, 1977.

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

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

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

[6] M. R. Genesereth, M. L. Ginsberg, 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.

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

[8] N. R. Jennings. Coordination Techniques for Distributed Artificial Intelligence. *Foundations of Distibuted Artificial Intelligence* (eds. O'Hare, G. and Jennings, N. R.): 187-210, Wiley, 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] K. Kuwabara, T. Ishida. Equilibratory Approach to Distributed Resource Allocation: Toward Coordinated Balancing. *Proceedings of the 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Artificial Social Systems* (eds. Castefranchi, C. and Werner, E.): 133-146, *Lecture Notes in Artificial Intelligence*, vol. 830, Springer, 1994.

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

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

[13] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible inference*. Morgan Kaufmann, 1988.

[14] M. Rovatsos. *Modeling social interaction dynamics for multi-agent societies: A layered learning approach to adaptive behaviour in games with an application to resource load balancing*. Diploma Thesis, Universität des Saarlandes, *to appear*.

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