The emergence of cooperative playing routines: optimality and learning *

R.A. Marques Pereira

mp@cs.unitn.it

Paolo Patelli

paolo@black.gelso.unitn.it

Laboratorio di Economia Sperimentale e Computazionale, Università di Trento

Via Inama 5-7, TN 38100 Trento, Italy Tel (39-461) 882147, 882246

January 1996 (Revised Version)

Abstract

We investigate the emergence of (optimal and suboptimal) behavioural routines in the context of a cooperative game. In particular we construct a search model of the gradient descent type for the optimization of 'static' and 'dynamic' playing routines. That optimality study sets the basis for the analysis of the dynamics and modelling of routine learning. In the last part of the paper we propose a learning heuristics for the development of routinized behaviour on the basis of a simple network model of the subject player.

Keywords: optimal cooperative routines, discrete optimization and search, routine learning, network models, bounded rationality, game theory

*The two authors have elaborated together every part of this research. However, as far as legal requirements are concerned, R.A. Marques Pereira takes responsibility for sections 1,2,5 and P. Patelli takes responsibility for sections 3,4.

1 Introduction

The questions of bounded rationality, behavioural routines and procedural learning [3] [4] [5] [6] [7] [8] [9] [18] play a crucial role in modern theories of evolutionary economics. In this research project we analyse the emergence of behavioural routines in the context of a simple experimental setting, that of a cooperative card game [6]. Our ultimate goal is to characterize the type of routines that emerge and understand the nature of the learning process.

The game involves two players - colourkeeper and numberkeeper - and the six cards 2,3,4 ♡ and 2,3,4 ♣. The board on which the game is played is as shown in figure 1.

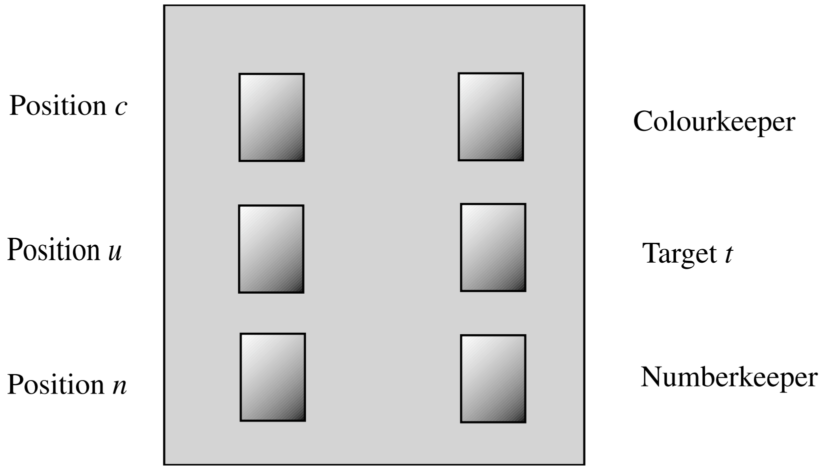

Figure 1: The board

The cards in positions u and t (target) are face-up, the others are face-down. As a result, each player sees its own card and the two cards in positions u, t. Neither player sees the other player's card.

The states of the game with 2 ♡ in the target position are called terminal states. Once the cards are dealt the two players are supposed to cooperate in order to transform the given initial state into a terminal state. Each player, in turn, modifies the state of the game by applying one of the following transformation operators,

$$\mathcal{T}, \mathcal{U}, \mathcal{C}, \mathcal{N} \text{ or } \mathcal{P}(\text{pass}).$$

The four transformation operators $\mathcal{T}$, $\mathcal{U}$, $\mathcal{C}$, $\mathcal{N}$ exchange the card in the player's hand with the card in position t, u, c, n respectively. The use of the operator $\mathcal{T}$ is constrained: the colourkeeper can play $\mathcal{T}$only if his card and the one in position t have the same colour; a similar rule applies to the numberkeeper, with reference to numbers instead of colours.

In the laboratory study of Cohen & Bacdayan [6] and Egidi [8] [9] the two players are encouraged not only to complete the game but also to play in an efficient manner. An incentive system is used which rewards the two players (in equal measure) in proportion to the number of hands successfully completed within a given amount of time. Moreover, a fixed cost per move is subtracted from the final payoff in order to discourage unnecessary moves. For a detailed discussion of the experimental setting see the original references indicated above.

In their original experiment Cohen & Bacdayan recorded the performances of 32 pairs of subjects playing two separate game sessions of 40 minutes each. Naturally, the sequence of hands played during each session was the same for all pairs. In our analysis of the resulting data the main goal is to study whether or not subjects do develop behavioural routines for cooperative playing and, if they do, identify which routines emerge and how.

The report is organized as follows: in the next section we introduce a special state representation based on the modular structure of the game. The modular representation suggests a set of simple static routines which codify good cooperative playing in a large number of cases. As a result we consider those routines as appropriate templates for our analysis of the development of cooperative routines. In the third and fourth sections we define the static and dynamic playing paradigms plus a search algorithm (based on a discrete gradient descent scheme) for the extraction of the optimal routine set. Finally, in the last section, we propose a learning model [10] [12] [13] [15] based on the adaptive performance of a neural network architecture [1] [2] [11] [14] [17].

2 The modular representation

The card game conceived by Cohen & Bacdayan has a natural modular structure. Any global solution to a particular hand decomposes into a sequence of local solutions associated with target transitions. This crucial property is best illustrated by means of the structural graph in figure 2 (see [8] [9] [16]),

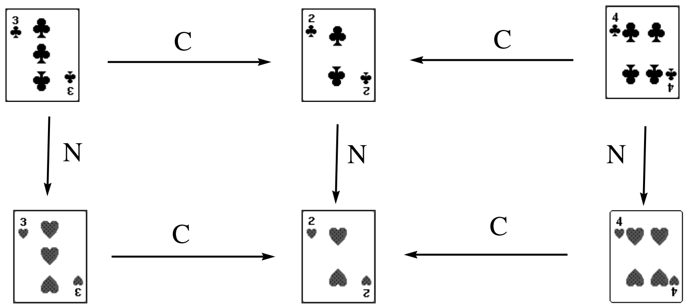

Figure 2: The structural graph

where the six nodes indicate the card in target and the arrows indicate positive target transitions (i.e. those that come closer to game completion). The negative target transitions are those obtained by inverting the arrows and the indifferent target transitions are the ones between 3 and 4 $\heartsuit$, or 3 and 4$\clubsuit$. Next to each target transition shown in the graph we indicate which of the two players can produce it (recall that the constraints regarding the use of the operator $\mathcal{T}$ are different for colourkeeper and numberkeeper).

We say that a game configuration is of level I if the card in target is either 3$\heartsuit$, 4$\heartsuit$ or 2$\clubsuit$. Instead, a game configuration is of level II if the card in target is either 3$\clubsuit$ or 4$\clubsuit$. With reference to the structural graph above, the geometrical meaning of the definition of level should be transparent.

In a game of level I there is only one interesting card to look for, the 2$\heartsuit$. In case 2$\clubsuit$ is in target, for instance, the numberkeeper has only to find the 2$\heartsuit$ to complete the game, while the colourkeeper should do nothing except reveal the 2$\heartsuit$ if he has it in hand (by playing $\mathcal{U}$). The same happens if either 3$\heartsuit$ or 4$\heartsuit$ are in target, with colourkeeper and numberkeeper interchanged.

The cooperative structure of a level II game is more interesting. The reason is that either player can produce the first target transition, that which transforms the level II configuration into one of level I. Clearly, the actual target transition depends on which player produces it. Assume, for instance, that the card in target is 4$\clubsuit$. In that case (see the structural graph above) the colourkeeper can produce a positive target transition with the 2$\clubsuit$ while the numberkeeper can do as much with the 4$\heartsuit$. Once one of the two players has produced the first target transition it is up to the other player to complete the game with the 2♡.

There are thus three key cards [8] [9] [16] in a game of level II. From the point of view of player X (colourkeeper or numberkeeper) the key cards are

$\Uparrow$ **flag** - the card with which player X can produce a +tt

$\Downarrow$ **dual flag** - the card with which player Y can produce a +tt

$\Updownarrow$ **double flag** - the 2♡, i.e. the card with which to complete the game after player Y has produced the first +tt

where +tt stands for 'positive target transition'. As an example suppose 3♣ is in target. From the numberkeeper's point of view the key cards are

$$\Uparrow = 3\heartsuit \qquad \Downarrow = 2\clubsuit \qquad \Updownarrow = 2\heartsuit$$

whereas from the point of view of the colourkeeper,

$$\Uparrow = 2\clubsuit \qquad \Downarrow = 3\heartsuit \qquad \Updownarrow = 2\heartsuit.$$

The modular representation (in terms of flags) of level II states has several advantages: the most important of these is that it captures the essential aspects of the game dynamics, according to the structural graph mentioned before. In doing so it provides a universal description of all level II games - no matter whether in target is 3♣ or 4♣, or whether the player considered is the colourkeeper or the numberkeeper - and thus opens the way to a universal characterization of the behavioural routines developed by subject players.

Moreover the universality of the modular representation leads to a finer and more reliable statistics of the experimental data and is also generalizable to complex games with more than two levels.

We now turn to the problem of routinized behaviour and, in particular, to the question of which rule templates are appropriate for its description. In this respect our strategy is to begin with the simplest possibility, i.e. that the subject players develop a pattern of cooperative playing depending solely on the visible state of the game. In other words, we assume a static routine paradigm in which the learning process leads to input-output rules where the input is constructed from the three visible cards - h (hand), u and t - and the output is the associated transformation operator - $\mathcal{T}$, $\mathcal{U}$, $\mathcal{S}$, $\mathcal{P}$. The operator $\mathcal{S}$(search) stands for a random choice between $\mathcal{C}$ and $\mathcal{N}$.

In this article we consider only games of level II for they are the most interesting from the point of view of cooperation. In those cases the modular representation suggests the set of reasonable behavioural routines illustrated in table 1. The first rule reads *if flag is in position h then play $\mathcal{T}$*; the second

Priority code	Condition	Action	
I	$h = \Uparrow$	$\mathcal{T}$	
I	$u = \Uparrow$	$\mathcal{U}$	
II	$h = \Updownarrow$	$\mathcal{P}$	
II	$u = \Updownarrow$	$\mathcal{U}$	
III	$h = \Downarrow$	$\mathcal{U}$	
III	$u = \Downarrow$	$\mathcal{S}$	search for $\Updownarrow$
$\bullet$	otherwise	$\mathcal{S}$	search for $\Uparrow$ or $\Updownarrow$

Table 1: Condition-Action routines

rule *if flag is in position u then play $\mathcal{U}$*, etc. The last rule means *if none of the preceding rules applies then play $\mathcal{S}$*.

The total number of static rules is 7. The first six rules are organized in three different groups (pairs) associated with priority codes from I to III. In each pair the first rule concerns the card in position h while the second rule regards the card in position u. Whereas the two rules in a pair are clearly mutually exclusive, the first and second rules from different groups are not necessarily so. As an example, it could happen that rules 1 and 4 are both applicable. In those cases dominates the rule with higher priority (lower priority code).

The applicability of each rule depends on the visible state of the game, as seen by either one of the two players. In other words, it depends on the cards in positions h, u and t. Within the universal modular representation, however, the information regarding the card in t is used only to set the card-value of the various flags. In this respect it plays the role of a pre-processing mechanism. Once the card-values of the flags have been assigned the visible state of the game is fully specified by the two positions h and u.

Our set of static rules performs optimally in a large number of cases and only moderately sub-optimally in the few remaining ones. Moreover it is simple and reflects common sense intuition within the modular approach. For these reasons we think that the static rule paradigm provides an appropriate framework for the study of cooperative routines in our card game. In what follows we introduce a search model designed to extract the optimal input-output routines of the static type.

3 Static routines

In the previous section we explained the modular representation and proposed to investigate the dynamics of procedural game playing within the static routine paradigm. In this section we introduce a search model in the space of static routines that looks for the optimal routine set of the static type.

The motivation is twofold: on one hand we wish to classify static routines according to their performance quality, as well as to establish whether our set of static routines is indeed the optimal one; on the other hand the search algorithm provides an opportunity to render explicit the limitations of the static routine paradigm, either for lack of efficiency or for poor cooperation.

A static routine table S is essentially an artificial player which responds in a predefined manner to each possible static configuration of the game. By static configuration we mean the present visible state of the game, i.e. the two cards in positions h and u .

The static routine tables are organized in 7 different rows, each of which concerns one of the possible static configurations as seen by the subject player. The game configurations are expressed in the modular representation: each of the two positions h and u can assume one of the following three values: flag, doubleflag or null (i.e. else).

The number of visible flag configurations is 7: 2 with two flags, 4 with one flag and 1 with no flags. The structure of a table S of static routines is thus as in table 2.

The input-output data flow of each of the 7 rows in the static rule table S is illustrated by the diagram in figure 3.

A static routine table assigns a definite response to each of the possible static configurations. The possible responses, or moves, are $\mathcal{T}$(target, only when h is flag), $\mathcal{U}$(up), $\mathcal{S}$(search) and $\mathcal{P}$(pass). The number of different static routine tables (static artificial players) is thus $4 \cdot 4 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 = 3888$.

The modular representation is convenient due to its universality, i.e. it

Rule index	Condition			Action
1	$h = \Uparrow$	$u = -$	$\longrightarrow$	?
2	$h = \Uparrow$	$u = \Updownarrow$	$\longrightarrow$	?
3	$h = -$	$u = \Uparrow$	$\longrightarrow$	?
4	$h = -$	$u = -$	$\longrightarrow$	?
5	$h = \Updownarrow$	$u = -$	$\longrightarrow$	?
6	$h = \Updownarrow$	$u = \Uparrow$	$\longrightarrow$	?
7	$h = -$	$u = \Updownarrow$	$\longrightarrow$	?

Table 2: Static Routine Table

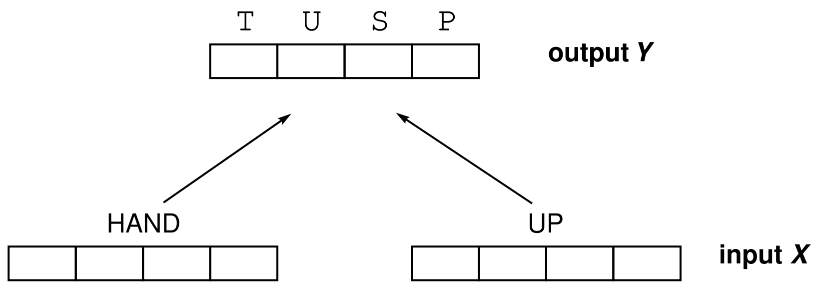

Figure 3: Input-output rule structure

is applicable to both players. The card-values of the flags, instead, depend on the actual card in target. They must be reset each time, for colourkeeper and numberkeeper in turn.

When written in the modular representation the number of possible distinct hands of level II reduces to 60: given the card in target, whose role is to define the card-values of the various flags, there remain 5 positions among which to distribute 3 flags, and thus $5 \cdot 4 \cdot 3 = 60$.

The search model is based on a cost function $F = F(S)$ which assigns a numerical cost to each static routine table S. The cost $F(S)$ associated to a static routine table is given by the convex combination of an efficiency cost $effic(S)$ and a cooperative cost $coop(S)$,

$$F(S) \ = \ (1-\alpha) \cdot effic(S) \ + \ \alpha \cdot coop(S)$$

where the weighting coefficient α is exogenous (i.e. fixed by the operator). Clearly, the optimal static rule table S is the one which minimizes the cost function F.

The efficiency cost $effic(S)$ simply counts the number of moves necessary to complete the full set of 60 hands (of level II). If for a given hand the artificial player reaches the threshold of 10 moves the game is interrupted and that particular hand contributes 10 to the efficiency cost.

The cooperative cost $coop(S)$ counts the total number of strategy changes ocurred during playing. The strategy changes within one hand are detected by means of 8 strategy markers, 4 for strategy + (*'get flag'*) and 4 for strategy − (*'get double flag'*). The two sets of strategy markers are described in tables 3 and 4.

Each player has a three state $(+1, 0, -1)$ strategy indicator which is updated every time the player's move coincides with a strategy marker. At the begining of each hand both markers are set to null. Each 'reset' of the strategy indicator (after the first setting) is counted as a strategy change.

This completes the description of how to compute the cost $F(S)$ of a given static routine table S. We now explain the structure of the search algorithm that looks for the optimal static routine table (the one with minimal cost):

1. Choose arbitrarily an initial static routine table S_0 and compute $F(S_0)$. The algorithm is based on a type of gradient descent procedure: in each iteration each individual degree of freedom changes in the locally optimal direction.

$strategy_+$ (GET $\Uparrow$)	
$h = \Uparrow$	followed by move $\mathcal{T}$
$u = \Uparrow$	followed by move $\mathcal{U}$
$h = \Updownarrow$	not followed by move $\mathcal{P}$
$u = \Updownarrow$	not followed by move $\mathcal{U}$

Table 3: Strategy markers for GET $\Uparrow$

$strategy_-$ (GET $\Updownarrow$)	
$h = \Updownarrow$	followed by move $\mathcal{P}$
$u = \Updownarrow$	followed by move $\mathcal{U}$
$h = \Uparrow$	not followed by move $\mathcal{T}$
$u = \Uparrow$	not followed by move $\mathcal{U}$

Table 4: Strategy markers for GET $\Updownarrow$

2. Examine each of the 7 rows of the static routine table individually, keeping the remaining 6 rows fixed, and determine the (locally) optimal move for that row by computing the cost associated with each of the three/four possible responses.

3. When the (locally) optimal move for a given row has been determined update that row's response and move on to the following row.

4. Repeat the procedure until the algorithm converges.

In the case $\alpha = 0$ we have found a global optimum plus a local optimum which disappears when $\alpha = 0.5$. The optimal and suboptimal static routine tables are presented in tables 5 and 6.

h	u	**action**
$\Uparrow$	-	$\mathcal{T}$
$\Uparrow$	$\Updownarrow$	$\mathcal{T}$
-	$\Uparrow$	$\mathcal{U}$
-	-	$\mathcal{S}$
$\Updownarrow$	-	$\mathcal{P}$
$\Updownarrow$	$\Uparrow$	$\mathcal{P}$
-	$\Updownarrow$	$\mathcal{U}$

Table 5: Optimal static routines

4 Dynamic routines

We now examine the dynamic playing paradigm, in which the artificial player responds not only to the static configuration of the game but also to the previous move by the other player. The dynamic model, therefore, corresponds to an artificial player with minimal memory: it remembers only the previous move.

In a dynamic routine table D the previous move is encoded in the $(+1, 0, -1)$ representation for the other player's strategy indicator. Consistently

h	u	**action**
$\Uparrow$	-	$\mathcal{T}$
$\Uparrow$	$\Updownarrow$	$\mathcal{T}$
-	$\Uparrow$	$\mathcal{S}$
-	-	$\mathcal{S}$
$\Updownarrow$	-	$\mathcal{P}$
$\Updownarrow$	$\Uparrow$	$\mathcal{P}$
-	$\Updownarrow$	$\mathcal{U}$

Table 6: Suboptimal static routines

with the minimal memory principle, however, the strategy indicators are updated each time according to the actual move the other player has made. If that move does not coincide with any of the 8 markers then the strategy indicator is set to 0. The three cases 'active', 'static' and 'passive' are associated with the three strategy indicator values +1, 0 and -1.

The dynamic routine tables are therefore organized in 3 sets of 7 rows, one for each strategy indicator value $(+1, 0, -1)$. The search algorithm operates essentially as before, visiting in turn the $3.7 = 21$ rows of the dynamic table. In this case there are no local optima. The (globally) optimal dynamic routine table is presented in tables 7 (static), 8 (active), 9 (passive).

Static			
h	u		action
$\Uparrow$	-	$\longrightarrow$	$\mathcal{T}$
$\Uparrow$	$\Updownarrow$	$\longrightarrow$	$\mathcal{T}$
-	$\Uparrow$	$\longrightarrow$	$\mathcal{U}$
-	-	$\longrightarrow$	$\mathcal{S}$
$\Updownarrow$	-	$\longrightarrow$	$\mathcal{P}$
$\Updownarrow$	$\Uparrow$	$\longrightarrow$	$\mathcal{U}$
-	$\Updownarrow$	$\longrightarrow$	$\mathcal{U}$

Table 7: Dynamic table: no markers set

Active			
h	u		action
$\Uparrow$	-	$\longrightarrow$	$\mathcal{S}$
$\Uparrow$	$\Updownarrow$	$\longrightarrow$	$\mathcal{U}$
-	$\Uparrow$	$\longrightarrow$	$\mathcal{S}$
-	-	$\longrightarrow$	$\mathcal{S}$
$\Updownarrow$	-	$\longrightarrow$	$\mathcal{P}$
$\Updownarrow$	$\Uparrow$	$\longrightarrow$	$\mathcal{P}$
-	$\Updownarrow$	$\longrightarrow$	$\mathcal{U}$

Table 8: Dynamic table: active marker set

Passive			
h	u		action
$\Uparrow$	-	$\longrightarrow$	$\mathcal{T}$
$\Uparrow$	$\Updownarrow$	$\longrightarrow$	-
-	$\Uparrow$	$\longrightarrow$	$\mathcal{U}$
-	-	$\longrightarrow$	$\mathcal{S}$
$\Updownarrow$	-	$\longrightarrow$	-
$\Updownarrow$	$\Uparrow$	$\longrightarrow$	-
-	$\Updownarrow$	$\longrightarrow$	-

Table 9: Dynamic table: passive marker set

5 A learning model

After having established a normative standard of optimallity with our static and dynamic routine tables we address the crucial issue of learning in the context of our cooperative game. In this section we propose a learning mechanism - architecture and heuristics - with which to model the emergence of cooperative routines among the subject players.

The idea is as follows: each subject player is modelled by an adaptive network as in figure 4. The network architecture models the decisional structure of the subject player and is therefore the same in all subjects. The network parameters, on the other hand, change from one subject to another, they are the distinctive individual labels of the various subject players.

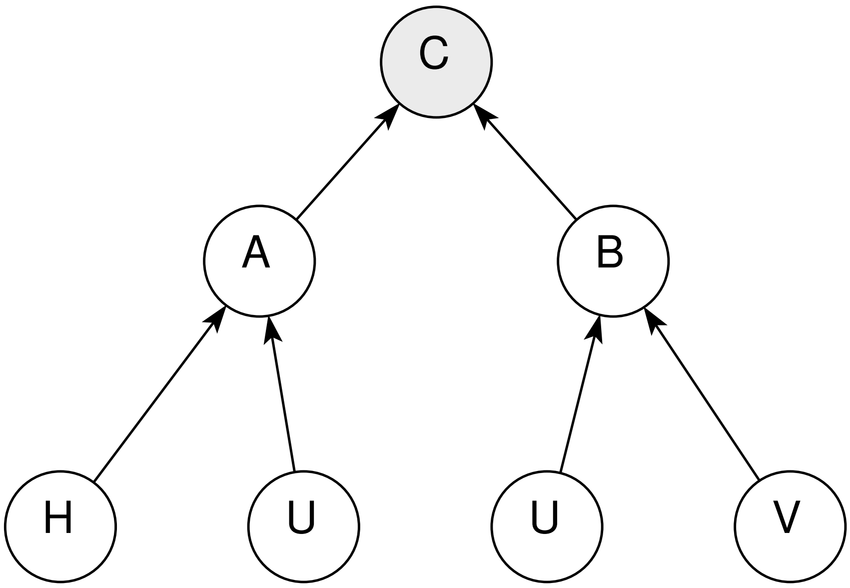

Figure 4: The network model of the subject player

The sub-network dominated by node A is called the 'self' part of the network. Its role is to construct a strategy preference based only on the static inputs H (card in h) and U (card in u). The two input nodes H,U take values within the interval $[-1, +1]$ according to the following 'self' flag representation,

$\Uparrow$=+1 $\star$=0 $\Updownarrow$=-1 ($\star$ means 'else') .

On the basis of the inputs H,U the strategy node A constructs a strategy preference with the usual local network law '$y = \sigma(\sum \omega_i x_i)$' as in figure 5. The sigma function σ is illustrated in the final section at the end of the report. The numerical semantics of node A is consistent with the 'self' flag representation characteristic of its sub-network,

A= +1	means strategy GET $\Uparrow$
A= 0	means strategy UNCLEAR
A= -1	means strategy GET $\Updownarrow$

At this point it is clear that the 'self' sub-network models the static part of the subject player. In the optimal case therefore the parameters a_1 and a_2 are such that the 'self' sub-network emulates the optimal static routine table (table 10).

H	**U**	**A**	h	u	**action**
+1	0	+1	$\Uparrow$	$\star$	$\mathcal{T}$
+1	-1	+1	$\Uparrow$	$\Updownarrow$	$\mathcal{T}$
0	+1	+1	$\star$	$\Uparrow$	$\mathcal{U}$
0	0	0	$\star$	$\star$	$\mathcal{S}$
-1	0	-1	$\Updownarrow$	$\star$	$\mathcal{P}$
-1	+1	-1	$\Updownarrow$	$\Uparrow$	$\mathcal{P}$
0	-1	-1	$\star$	$\Updownarrow$	$\mathcal{U}$

Table 10: Sub-network A (optimal)

In the optimal case the appropriate values for the parameters are $a_1 = 2$ & $a_2 = 1$ as can be seen from the linear separation diagram in figure 6.

The sub-network dominated by node B is called the 'dual' part of the network. Its role is to construct the strategy preference of the other player

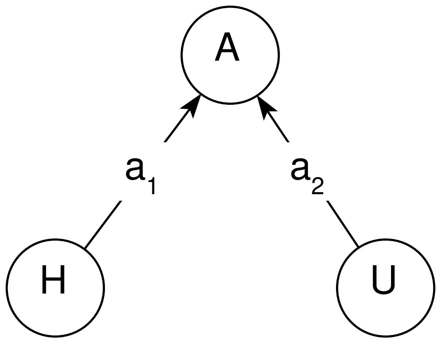

Figure 5: The ‘self’ sub-network $A = \sigma(a_1 \cdot H + a_2 \cdot U)$

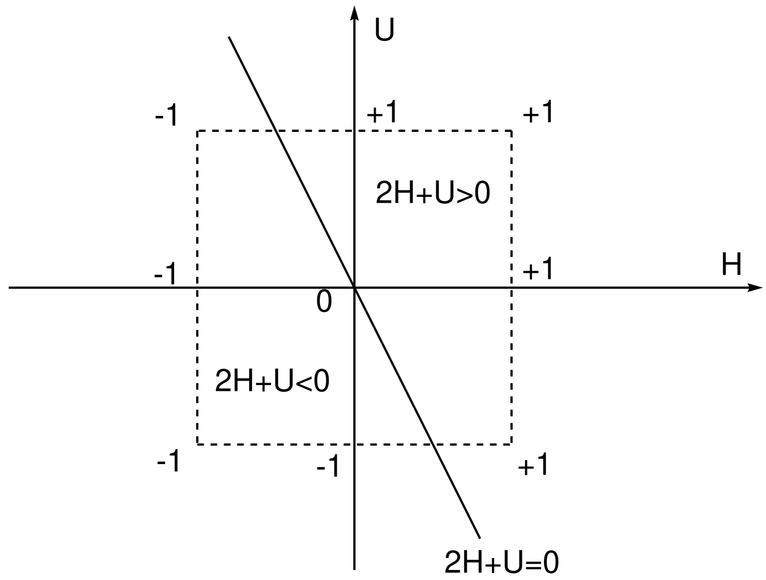

Figure 6: Sub-network A (linear separation diagram)

based on the inpus V and U, where V stands for the card in position u before the previous move. As before, the two inputs take values within the $[-1, +1]$ interval but this time according to the 'dual' flag representation,

$$\Downarrow\text{=+1} \qquad \star\text{=0} \qquad \Updownarrow\text{=-1} \qquad (\star \text{ means 'else'}) \ .$$

On the basis of the inputs V, U the strategy node B constructs the other player's strategy preference (figure 7). The numerical semantics of node B is consistent with the 'dual' flag representation characteristic of its sub-network,

B= +1	means strategy GET $\Downarrow$
B= 0	means strategy UNCLEAR
B= -1	means strategy GET $\Updownarrow$

The 'dual' sub-network models the dynamic part of the subject player. In the optimal case therefore the parameters b_1 and b_2 are such that the 'dual' sub-network emulates the following optimal dynamic routine table (table 11).

V	**U**	**B**	v	u	**strategy**
+1	0	+1	$\Downarrow$	$\star$	+
+1	-1	+1	$\Downarrow$	$\Updownarrow$	+
0	+1	-1	$\star$	$\Downarrow$	-
0	0	0	$\star$	$\star$	?
-1	0	-1	$\Updownarrow$	$\star$	-
-1	+1	-1	$\Updownarrow$	$\Downarrow$	-
0	-1	+1	$\star$	$\Updownarrow$	+

Table 11: Sub-network B (optimal)

In the optimal case the appropriate values for the parameters are $b_1 = 1$ & $b_2 = -2$ as can be seen from the linear separation diagram in figure 8.

The leading node $C = -AB$ of the learning network plays a coordinating role with respect to the strategy preferences expressed by nodes A and B, as

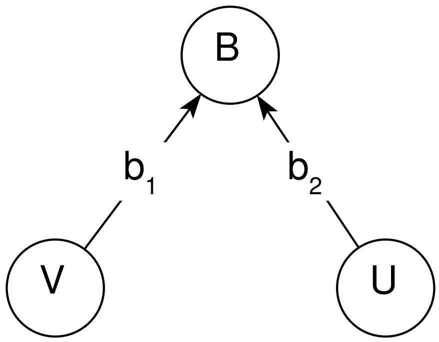

Figure 7: The 'dual' sub-network $B = \sigma(b_1 \cdot V + b_2 \cdot U)$

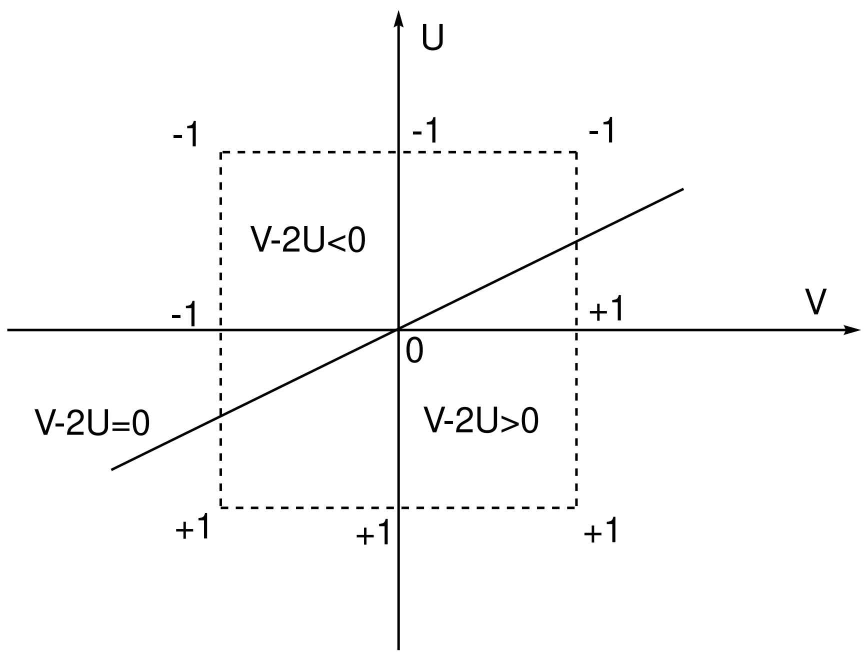

Figure 8: Sub-network B (linear separation diagram)

illustrated in figure 9. The optimally coordinated preferences are those with $C = 1$, the worse cases instead are those with $C = -1$.

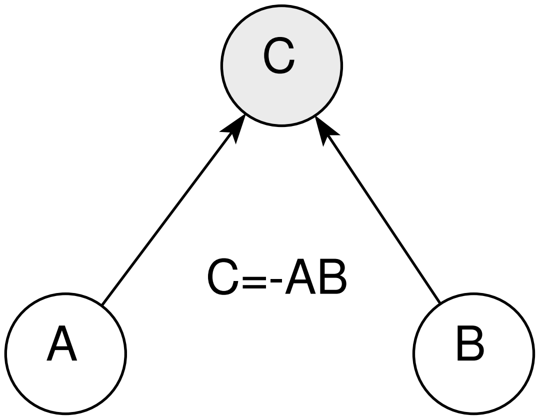

Figure 9: The leading node

The learning heuristics of our network model is based on two principles: the *consistency principle* states that the four network parameters 'learn' maximal consistency and the *stability principle* states that the parameters 'learn' to stabilize (from one move to the next) the strategy preferences expressed by nodes A and B.

Our learning heuristics corresponds to an optimization algorithm based on a cost function F which is a convex combination of a consistency term and a stability term,

$$F = F(a_1, a_2, b_1, b_2) = (1 - \mu)(1 - C)^2 + \mu[(A - A^\dagger)^2 + (B - B^\dagger)^2]/2$$

where $A^\dagger$ and $B^\dagger$ stand for the previous A and B values. The resulting learning process is an iterative gradient descent mechanism with respect to the four parameters of the cost function F,

$$a_1' = a_1 - \lambda \frac{\partial F}{\partial a_1} \qquad a_2' = a_2 - \lambda \frac{\partial F}{\partial a_2}$$

$$b_1' = b_1 - \lambda \frac{\partial F}{\partial b_1} \qquad b_2' = b_2 - \lambda \frac{\partial F}{\partial b_2}$$

where λ is the so-called learning rate parameter.

6 Concluding remarks

The learning heuristics above acts on the network parameters on the basis of a learning table of selected examples of *good playing*, each of which corresponds to an optimal transition from one game configuration (H,U,U,V) to the next. The game configurations (input vectors for the network model) are extracted from the following combinatorial table regarding all possible distributions of the key cards $\Uparrow$, $\Downarrow$, $\Updownarrow$,

H	**U**	**U**	**V**				
$\star$	$\star$	$\star$	$\star$	$\Uparrow$	$\Downarrow$	$\Updownarrow$	(4)
$\star$	$\Uparrow$	$\Uparrow$	$\star$	$\Uparrow$	$\Downarrow$	$\Updownarrow$	(4)
$\star$	$\Downarrow$	$\Downarrow$	$\star$	$\Uparrow$	$\Downarrow$	$\Updownarrow$	(4)
$\star$	$\Updownarrow$	$\Updownarrow$	$\star$	$\Uparrow$	$\Downarrow$	$\Updownarrow$	(4)
$\Uparrow$	$\star$	$\star$	$\star$		$\Downarrow$	$\Updownarrow$	(3)
$\Uparrow$	$\Downarrow$	$\Downarrow$	$\star$		$\Downarrow$	$\Updownarrow$	(3)
$\Uparrow$	$\Updownarrow$	$\Updownarrow$	$\star$		$\Downarrow$	$\Updownarrow$	(3)
$\Downarrow$	$\star$	$\star$	$\star$	$\Uparrow$		$\Updownarrow$	(3)
$\Downarrow$	$\Uparrow$	$\Uparrow$	$\star$	$\Uparrow$		$\Updownarrow$	(3)
$\Downarrow$	$\Updownarrow$	$\Updownarrow$	$\star$	$\Uparrow$		$\Updownarrow$	(3)
$\Updownarrow$	$\star$	$\star$	$\star$	$\Uparrow$	$\Downarrow$		(3)
$\Updownarrow$	$\Uparrow$	$\Uparrow$	$\star$	$\Uparrow$	$\Downarrow$		(3)
$\Updownarrow$	$\Downarrow$	$\Downarrow$	$\star$	$\Uparrow$	$\Downarrow$		(3)

Table 12: Full combinatorial table for $\Uparrow$, $\Downarrow$, $\Updownarrow$ configurations (43)

Clearly, not all rows in table 12 are relevant for the construction of the learning table, since the 'self' part of the network disregards the key card $\Downarrow$ and the 'dual' part of the network, in turn, disregards the key card $\Uparrow$. The description and results of the experimental testing (computer simulation) of the learning scheme will be presented elsewhere.

The sigma function σ mentioned in the previous section is given by

$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad , \quad \sigma(x) \in (-1, +1)$$

$$\dot{\sigma}(x) = (1 - \sigma^2(x)) \quad , \quad \dot{\sigma}(0) = 1$$

Acknowledgements

The authors are grateful to Prof. Massimo Egidi (Direttore del Laboratorio di Economia Sperimentale e Computazionale CEEL) for many helpful discussions and useful comments.

References

[1] J. A. Anderson, A. Pellionisz, and E. Rosenfeld, editors. *Neurocomputing: Directions for Research*. MIT Press, Cambridge MA, 1990.

[2] J. A. Anderson and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research*. MIT Press, Cambridge MA, 1988.

[3] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York NY, 1984.

[4] K. Binmore. Modelling rational players (parts 1 and 2). *Economics and Philosophy*, pages 9–55, 179–214, 1990.

[5] M. D. Cohen. Individual learning and organizational routines: emerging connections. *Organization Science*, 2(1):135–139, 1991.

[6] M. D. Cohen and P. Bacdayan. Organizational routines are stored as procedural memory: evidence from a laboratory study. *Organization Science*, December:554–568, 1994.

[7] G. Dosi and M. Egidi. Substantive and procedural uncertainty. an exploration of economic behaviours in complex and changing environments. *Journal of Evolutionary Economics*, 1:145–168, 1991.

[8] M. Egidi. Routines, hierarchies of problems and procedural behaviour: some evidence from experiments. In K. Arrow et al., editors, *The Rational Foundations of Economic Behaviour*. MacMillan, London, 1996 in print.

[9] M. Egidi and A. Narduzzo. The emergence of path-dependent behaviour in cooperative contexts. Technical Report 4, Laboratorio di Economia Sperimentale e Computazionale, University of Trento, Italy, 1996.

[10] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, Reading MA, 1989.

[11] R. Hecht-Nielsen. *Neurocomputing.* Addison Wesley, Reading MA, 1990.

[12] J. H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, Cambridge MA, 2nd edition, 1992.

[13] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. *Induction - Processes of Inference, Learning and Discovery.* MIT Press, Cambridge MA, 1988.

[14] S. A. Kauffman. Adaptation on rugged fitness landscapes. In D. L. Stein, editor, *Lectures in the Sciences of Complexity*, pages 527–618. Addison Wesley, Redwood City CA, 1989.

[15] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, Oxford, 1993.

[16] R. A. Marques Pereira. A cooperative game study: an adaptive network for the representation and learning of symbolic rules. Technical Report 17-DISA-93, Dipartimento di Informatica e Studi Aziendali, University of Trento, Italy, 1993.

[17] D. E. Rumelhart, J. L. McClelland, and the PDP research group. *Parallel Distributed Processing (2 volumes).* MIT Press, Cambridge MA, 1986.

[18] H. A. Simon. From substantive to procedural rationality. In S. J. Latsis, editor, *Method and Appraisal in Economics*, pages 129–148. Cambridge University Press, Cambridge, 1976.