DSMUS 1.0

Sergio Redondo, Héctor Sainz, Arucas Chacón.

Abstract – **This paper shows how we have developed a video game for Nintendo DS in order to play "Mus", one of the most popular Spanish card games, thereby we considered designing a video game would be an interesting way of applying our acquired knowledge about artificial intelligence.**

Key Words – **Artificial Inteligent, Case-Based Reasoning, Mus.**

I. Introduction

Our inteligent system will be built through **Case-based reasoning**,i.e, firstly, our video game will only know how to play without experience. Every game will be stored and used as previous experience for next games.So, the video game, little by little, will be able to use its previous experience, which has been acquired by playing, in order to play better [2].

Although our first option was Bayesian Networks, Nintendo DS features did not allow to import an external library to interact with Nintendo DS devices.As we could not use a bayesian network henceforth CBR was our best option.Now it only needs a simple text file where the cases are saved [1].

PALIB (PA9.h), which is a C++ library, has been used to handle NDS's features[4].There are more libraries,such as Libnds, but Palib is the best for beginners [3].

According to the relevance of CBR , our project could be an example of how useful and easy it is.

As there are many parameters to be taken into account while a player is playing "Mus" we decided *Nearest neighbour tehcnique* would be the best option to implement our strategy video-game [2].

A typical algorithm for calculating nearest neighbour is:

$$\frac{\sum_{i=1}^{n} w_i \times sim\left(f_i^I, f_i^R\right)}{\sum_{i=1}^{n} w_i} \quad (1)$$

Where *w* is the importance weighting of a feature , *sim* is the similarity function, and *fI* and *fR* are the values for feature i in the input and retrieved cases respectively.

II. Cases

We use cases in order to decide which is the best move to win a round ("Grande", "Chica","Pares" and "Juego").

Our algorithm produces a result by employing the formula :

$$x + \sum_{i=1}^{i=4} parameter_i * weight_i \quad (2)$$

X: card index according to the cards value and round.

The parameters or features and its weights ,that will be taken into account to calculate the case, are detailed below (order by priority):

- My hand➔ player's current hand.
- IamFirstOverMyOpponent➔ I have priority over my opponent.
- Near30Opponent➔ My opponent is closer to 30 than me.
- Near30Me ➔ I am closer to 30 than my opponent.

The algorithm result will be a case index, which will be used to find the best suitable case.

Weights:

- 1 -> value= position in 332 posibilities for "Grande" and "Chica"

There are 332 different combinations of cards.

- 0.9 -> BeingMano (positive value(yes)=1*0.9, negative value(no)=0)
- 0.8 -> Near30Opponent (value=Opponent's points*0.8)
- 0.7 -> Near30Me (value=My points*0.6)

Cases are stored in simple text files. In **DSMUS 1.0** there is one file per round ("Grande" ,"Chica","Pares" and "Juego") and per player, that means sixteen files. In future versions a player will be able to play against Nintendo DS.

III. Architecture Files

In this chapter it is described how a case file is structured.

File Name: "casesXY.dat " :

X: player (0,1,2,3)
Y: round (0="Grande", 1="Chica", 2="Pares", 3="Juego")

For instance:
cases00.dat, cases01.dat, cases02.dat, cases03.dat, cases10.dat, ... cases33.dat.

At the beginning of the game, every file is dumped into a list (that will be contained in a matrix of lists) in order to speed up access.
At the end of the game the case matrix will be dump into its corresponding file.
As it was said before, a case is a set of fields or paramaters: index ,won ,lost and used times.

```
struct caso{
  float index;
  int used;
  int won;
  int lost;
  struct case *next;
}* CaseMatrix[4][4];
```

An example of a case file content could be:

```
#332.4$1$1$0$##330.4$1$0$1$##220.6$1$
0$1$##214.6$1$0$1$##0$1$0$1$#
```

- '#' is used as a case divider and '$' is used as a divider between fields.

For instance:

Caso1: #332.4$1$1$0$#
Index: 332.4
Times used: 1
Times won: 1
Times lost: 0
Caso2: #330.4$1$0$1$#
Index: 330.4
Times used: 1
Times won: 0
Times lost: 1
Caso 3:#220.6$1$0$1$#
Index: 220.6
Times used: 1
Times won: 0
Times lost: 1

Where the index is the first field, the second one means how many times this case has been used, the next one specifies how many times this case has won, and the last one shows how many times this case has lost.

IV. How to chose the case – Similarity function (queHaces)

The cases are indexed by a float number, which is the result of an algorithm based on the nearest neighbour.

Firstly, once the index case is calculated by the formula explained before, the system will search in the corresponding file for the correct index. If the index is in the case file, it will be used. If it has not been found, another new one will be created.(check the chapter "**how it is created").**

Secondly, in order to design a game as real as possible, it has been used a likelihood of doubt in order to bet or not according to a random value.
In this way, players will know the posibilities of winning and losing with the case but they will be free to decide their next move (simulated by a random value).

For instance:

Cards: 'r''r''s''p' (king,king,knave and card whose number is two or one)
Round: "Grande"
Player: Number 3

1. The case index is calculated by the formula.
- Card index: cards value in this round:

'r''r''r''r' is the best card combination for "Grande" round, whose value is 332.
As there are 332 cards combinations for "Grande" round.
'p''p''p''p' is the worst card combination for "Grande" round, whose value is 1.

Example 'r''r''s''p' index: 280

- IamManoOverMyOpponent whose value could be 1 or 0.

IamManoOverMyOpponent=1

- Near30Opponent and Near30Me whose values could be 1 or 0.

Near30Opponent = 0
Near30Me = 1
Case index = 280+1*0.9+0*0.8+1*0.7 = 281,6

2. The case index is searched in the corresponding file considering player and round.

File: cases30.dat

#332.4$1$1$0$##330.4$1$0$1$##281.6$6$4$0$##214.6$1$0$1$##0$1$0$1$#

Wanted index : 281,6➔.It is in the case file.
Times used:6
Times won:4
Times lost:0
Times skipped: 2= (6-(4+0))

3. Calculating percentage of victory.
percentage=(won*100)/used;

*percentage=(4*100)/6 = 66,66*

Now, player knows his percentage of victory, but wining still depends on his behaviour and a random value.

4. Calculating random value.

Random value = 80

If the random value is lower than the percentage of wining, player will bet. otherwise he will pass that round.

He will pass this round because 80 > 66,66

When a case is reused, the used, won or lost times fields will be increased by one.

V. How is it created a new case?

If the case is not in the file of cases, a new case is created in order to decide what a player is going to do in this round by averaging between the border indexes
It is worth highlighting that this operation is only useful in order to decide what a player is going to do in this round.However this case is not stored in a case matrix.
The strored one will be:

Case Index = whatever it is
Times used:1
Times won:0 or 1 (it depends on the winner)
Times lost:1 or 0 (it depends on the winner)

According to decision making, an example of case could be:

For instance:

The index 290 is looked for but it is not in the file but there are the indexes:

Index Number of times :used won lost
300 12,3 10 0
280 4 1 2
So,it is calculated the average between 300 and 280's parameters.
New case:
290 8,15 5,5 1

This way of managing doubt has also been taken into account to decide if a player wants his cards or not**. (quieroMus function)**

How to decide if a player wants his cards?

It is calculated for every index (importance of the cards) of every round.
The hightest value is 277 acording to the best cards (four kings), so it is calculated a rule of three between the index and 277.

Once the player knows his cards's **"importance",** he will make the decision, which is simulated by a random value, about drawing or not other cards.

VI. How to decide which cards is a player going to play with? – ("DESCARTE")

In **DSMUS 1.0** players only keep king cards, i.e, starting on the basis that players do not want their cards ("querer Mus"), they count how many kings they have.

According to the number of kings the best combination of cards that a player could have if he keep one,two or three cards it's calculated.

For instance:
Cards : 'r''r''4''p'.

1. Calculating best possible combinations in order to have better cards:
- one card left:
best possible combinations: RRR4, RR44, RR74; RRRP, RRPP
-two cards left:
best possible combinations: RRRR

2. Taking every combination into account, it calculates the best one.

According to the group of the best combination belongs ,i.e. one card left or two cards left, that will be the best decision.

Best option: RRRR

So, player will decide to leave two cards.

Best option: RRRR (the best cards)

VII. Interaction of players

Firstly, there are two points of view according to players' role: game partner and opponent.

PARTNERS

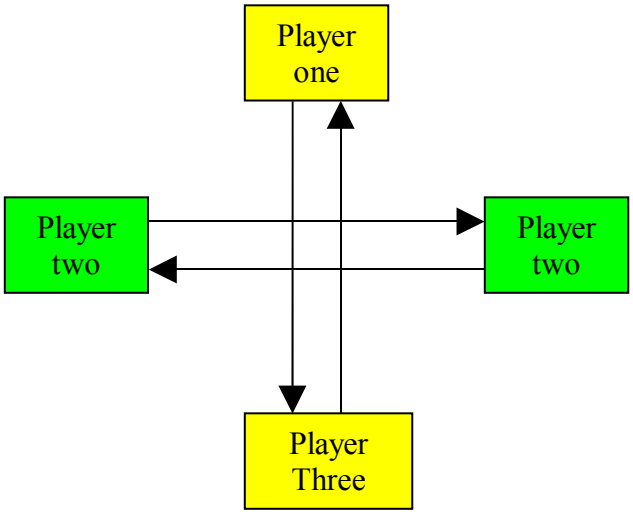

Fig 1.Partners Diagram

OPONENTS

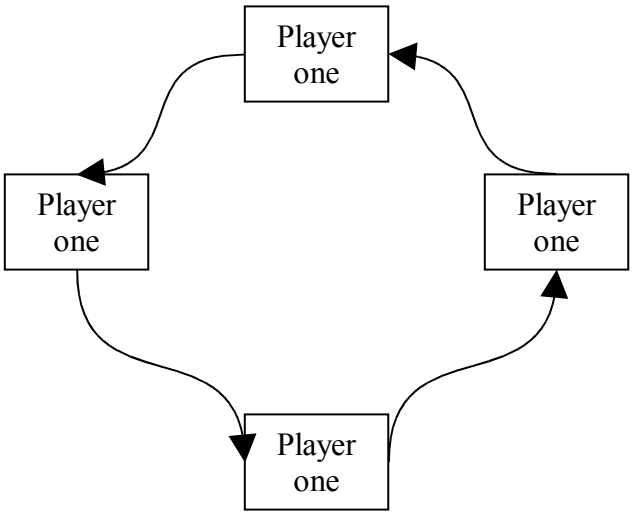

Fig 2. Opponent Diagram

This is a recursive function (jugar) which is going to be explained by an example.

ACTION: ALL OF THEM PASS

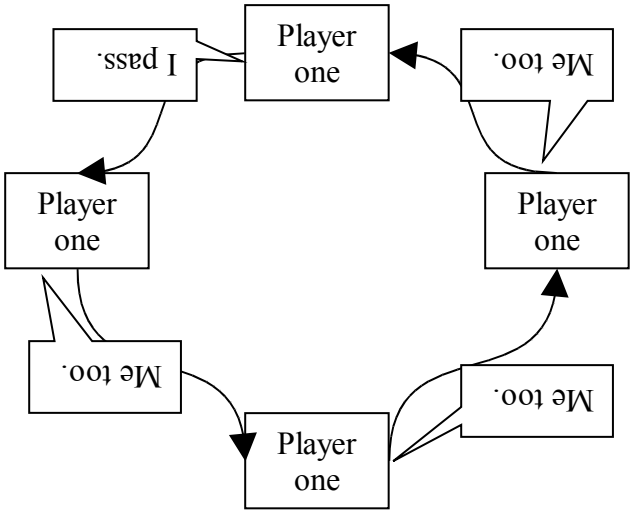

Fig 3. Game function Diagram I

So, if a player makes the decision of passing, the control will be given to his opponent.

ACTION: Player Three bets ("envida")

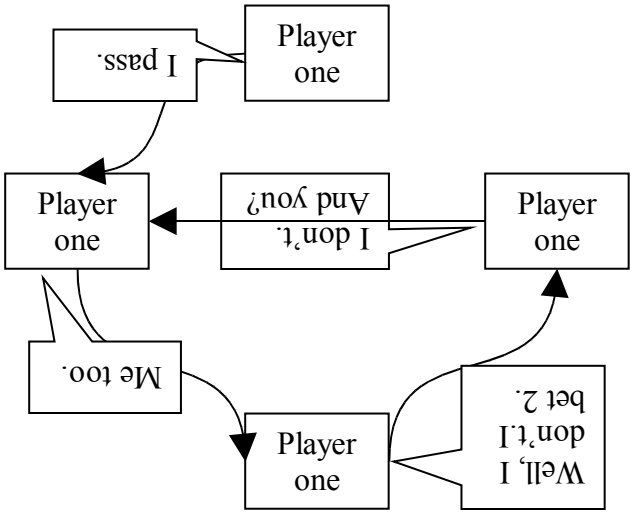

Fig 4. Game function Diagram II

So, if a player has to answer back a bet, assuming he does not want to bet, he will have to ask to his partner.

ACTION: Player Two bets ("envida")

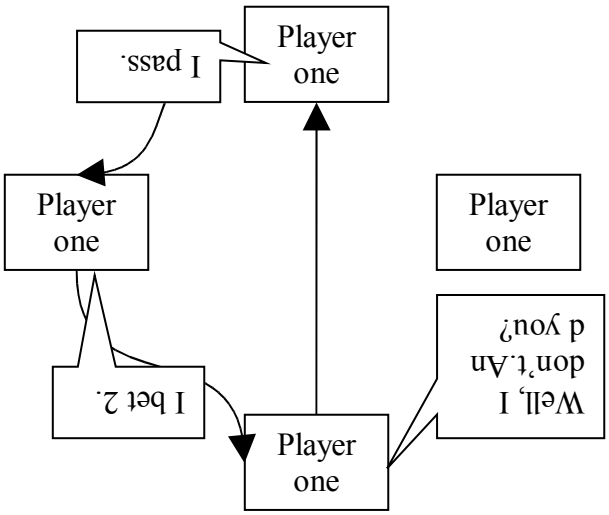

Fig 5. Game function Diagram III

This function finishes if :
- all of the playes pass this round
- one of the player agree with a bet which an opponent made.
- "ORDAGO" ➔ a bet which is higher than 30 points

VIII. TOOLS

As it was said before, it has been used Palib library in order to handle NDS features, but there are more choices, such as, DSLua, which is a SDK , whose programs have to be written in Lua programming language , an important disadvantage[8].

A.Card game "Mús"

Mus is a popular Spanish card game originating from the Navarre and Basque regions in Spain. From there it spread all over the country, where it is now the most played card game, spawning countless Mus clubs or *peñas* and becoming a staple game among college students. It is highly regarded, being considered by many as one of the finest card games.

It is played (normally in two pairs) with the Spanish deck which is a deck of 40 cards (without eights or nines). The game has four rounds:

- **Grande** (*Biggest*): playing for the highest combination of cards.
- **Pequeña** or **Chica** (*Smallest*): playing for the lowest combination of cards.
- **Pares** (*Pairs*): playing for the best matching card combination.
- **Juego** (*Game*): playing for cards total values of 31 or more. Sometimes replaced by a **Punto** (*Point*) special round.

It has a distinctive feature in that signals (*señas*) between players are an accepted as part of the game.[7]

B.CBR

Case-based reasoning is a problem solving paradigm that in many respects is fundamentally different from other major AI approaches. Instead of relying solely on general knowledge of a problem domain, or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize the *specific* knowledge of previously experienced, concrete problem situations (cases). A new problem is solved by finding a similar past case, and reusing it in the new problem situation. A second important difference is that CBR also is an approach to incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it immediately available for future problems.[2]

IX. REFERENCES

[1] Wikipedia – Bayesian network
http://es.wikipedia.org/wiki/Red_Bayesiana
[2] Cased-Based Reasoning - Papers
http://www.ai-cbr.org/classroom/cbr-review.html
http://www.iiia.csic.es/People/enric/AICom.html
http://www.cs.indiana.edu/~leake/papers/p-96-01_dir.html/paper.html
[3] Web libnds.
http://devkitpro.sourceforge.net/devkitProWiki/libnds/
[4] Web Palib
http://www.palib.com/
[5] Web Devkitpro.
http://www.devkitpro.org/
[6] Palib API.
http://palib.info/Doc/PAlibDoc%20Eng/modules.html
[7] Wikipedia-Mus (card game)
http://en.wikipedia.org/wiki/Mus_%28card_game%29
[8] DSLua information
http://www.bio-gaming.com/jeremy/dslua/documentation/xhtml/index.html