Preference-based Player Modelling for CIVILIZATION IV

Matthijs Rohs

July 4, 2007

Abstract

The Artificial Intelligence in modern computer games is inferior to the average human intelligence. In order to improve computer game AI, better imitation of human behaviour is required. One way to achieve better game AI is to have the computer build a player model for its opponents, so that it can base its decisions on good predictions of what the human player wants to achieve. This paper describes a way of building such a player model for the turn-based strategy game CIVILIZATION IV.

1 Introduction

Games have been played in most cultures and societies throughout history [3]. Computer games appeared shortly after the first computers were introduced. One key component of games is challenge, which brings up the concept of dealing with opponents. Depending on the game, play is against human and/or computer-controlled opponents. Computer-controlled opponents are simulations of human opponents; their behaviour is determined by so-called 'game AI'. Currently, the AI in most computer games is inferior to human intelligence. Improvement of AI can be achieved by better imitation of human behaviour. One way to achieve this, is by letting the AI build models of opponents [5]. Opponent models can be either *explicit* or *implicit* [9]. An explicit model implies that the opponent's attributes exist separately from the decision-making process in the game AI. Implicit means that the game AI is fine-tuned to specific (types of) opponents. The use of explicit opponent models is considered to be a necessary requirement for successful game-play for games like ROSHAMBO [7] and POKER [4]. For this reason we think explicit opponent modelling is preferred over implicit opponent modelling when dealing with adaptive game AI.

According to Donkers and Spronck [6] the preference-based model is a class of the explicit opponent model. In this model, the game AI makes its decisions based on the action which has the highest preference in a certain game state. These models seem suitable in order to improve the game AI of one of the most complex game genres: turn-based strategy (TBS). Therefore our research focusses on the construction of preference-based models for turn-based strategy games, leading to the following research question: *'To what extent can a preference-based player model be constructed for a TBS game?'*.

Section 2 provides some background information on TBS games and the TBS game used for this research: CIVILIZATION IV (CIV4) [1]. Section 3 gives the definition of a preference-based player model and describes both a general implementation and a specific implementation of the model for CIV4. Section 4 gives the results of our research. Section 5 contains some discussion about our results. Finally, Section 6 gives our conclusions and discusses possible future work.

2 Background

This section describes some background information about TBS games in general (Subsection 2.1) and CIV4 in particular (Subsection 2.2).

2.1 TBS games

TBS games are, unlike real-time strategy games, characterized by interchanging turns rather than continuous 'real-time' play. Players can have their turns simultaneously (e.g., CIV4) or alternately (e.g., CHESS). Once a player entered his[1] turn, he is allowed a period of time (limited or unlimited) before making a move. This concept causes a separation between the game flow and the thinking process; players have more time to plan their moves. However, it is not uncommon to measure the time taken by a player to make his move. This is done in order to improve the fairness of the game and/or to set an upper limit to the length of the game.

2.2 CIV4

CIV4 is a TBS game in which the player builds an empire from scratch. All standard full-length games begin in '4000 BC' (game time) with a settler that builds a single city. From there on, the player expands an empire while competing with rival nations, utilizing the geography, developing infrastructure, and encouraging

[1]For the sake of readability, we use 'he' and 'his' in this paper, where respectively 'he/she' and 'his/her' are meant.

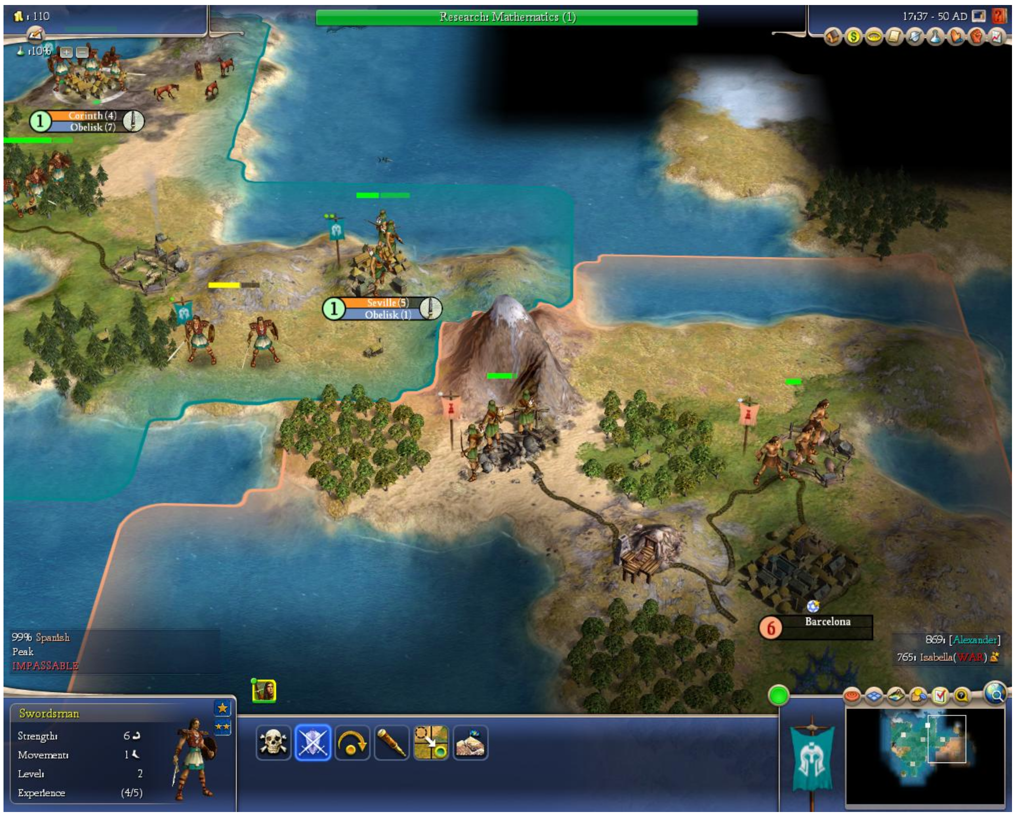

Figure 1: Alexander the Great during war with Isabella.

scientific and cultural progress. Figure 1 shows a screenshot of the game. In this game, Alexander the Great is playing against Isabella. Parts of their empires are visible, for example the cities Corinth and Seville belong to Alexander the Great, while Barcelona belongs to Isabella.

In contrast to most strategy games, conquest is not the only road to victory. There are six victory conditions [2]:

- time victory (achieve the highest score in '2050 AD'),
- conquest victory (eliminate all rivals),
- domination victory (control 75% of the world population and 75% of the world's land area),
- cultural victory (own three cities with 'legendary culture'),
- space race (construct all the components for a space ship), and
- diplomatic victory (win the election for world leader when the 'United Nations' is built).

Each civilization is represented by a leader. The general behaviour of a civilization is determined by the traits of its leader. There are twenty-six different leaders in the basic game, each of them having two out of eight different traits. The variation in traits causes each civilization to have different advantages during the game. For example, Alexander the Great has his traits set to 'philosophical' and 'aggressive'. Being philosophical helps the player's research for culture and science. Being aggressive brings advantages for training military units.

An aspect that separates Civ4 from other strategy games, is the fact that players can have peaceful relations with other participating players instead of having permanent war. For example, players can trade resources, have open borders and make defensive pacts with each other. Figure 2 shows a screenshot of a so-called 'diplomacy screen'.

The creators of Civ4 released a software development kit for their game, which contains the core game DLL source code, allowing programmers to completely rewrite or modify the game. Access to internal variables in Civ4 happens through the programming languages C++, Python and XML.

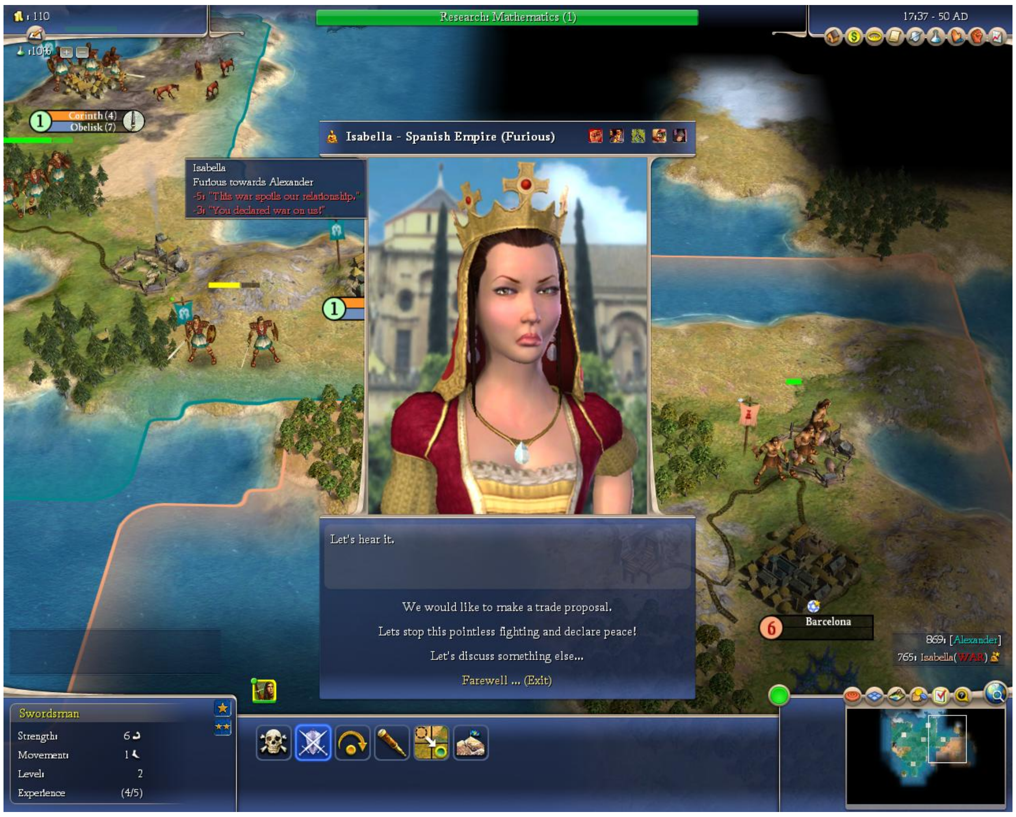
Research: Mathematics (1)
17:37 - 50 AD
Corinth (4)
Obelisk (7)
Isabella - Spanish Empire (Furious)
Isabella
Furious towards Alexander
-5: "This war spoils our relationship."
-3: "You declared war on us!"
Let's hear it.
We would like to make a trade proposal.
Lets stop this pointless fighting and declare peace!
Let's discuss something else...
Farewell ... (Exit)
Barcelona
869: [Alexander]
765: Isabella (WAR)
Swordsman
Strength: 6
Movement: 1
Level: 2
Experience (4/5)

Figure 2: Diplomacy screen of Isabella during war.

3 Preference-based player model

Our goal is to build a preference-based player model of a civilization's leader, whether this leader is human or AI. This section describes our method of building such a preference-based player model. In Subsection 3.1, our definition of this model is presented. Subsection 3.2 describes a general implementation of the model. Finally, Subsection 3.3 describes how to implement a preference-based player model for CIV4.

3.1 Definition

A player performs actions during a game. These actions cause internal variables for the current player to change. Keeping track of these changes may bring certain preferences to light. A preference-based player model attempts to track these preferences, validate them, and use them to determine actions. Donkers and Spronck [6] state that in order to build a preference-based player model, one must generally follow two steps. The first step is to determine the possible game states of interest. The second step is to keep track of the player's actions while being in a certain state.

The two game states of CIV4 we are interested in for this research are *peace* and *war*. Our general implementation of a preference-based player model is therefore based upon these two possible game states.

3.2 General implementation

A common form for a preference function is a linear combination of weights and game features:

$$V(s) = \sum_i w_i s_i,$$

where s is the game state, s_i are numerical measures of some features of the game state s, and w_i are the weights of the features in the function. The values of s_i and w_i can be either positive or negative.

After determining the game states and features of interest, preference values need to be assigned to the game states. First, we need to make sure that we are capable of storing player variables during gameplay, as these will represent the features of the game states. This process is different for each game, depending on whether the game contains a built-in mechanism for storage of player variables (e.g., a debugging tool) or not. Assuming there is a possibility to store certain internal variables, the next step is to determine the amount of test games that should run in order to determine reliable (average) values. This amount is also game dependant. Once the test games are finished and the data are acquired, $V(s)$ is calculated for each possible combination of weight factors w_i and stored variables s_i.

In order to properly compare the different test games, we normalize the preference values $V(s)$ to the continuous domain $[0, 100]$. These new values are defined as f. The formula for f is:

$$f_i = \frac{V_{s_i} + |V_{min}|}{|V_{min}| + V_{max}} \times 100,$$

where V_{min} and V_{max} are respectively the minimum and maximum of the original set $V(s)$.

In order to find the ideal weight factors w_i, we must calculate a factor we call *score* for each combination of the different weight factors. This *score* is a weighted average of all the f_i values of the same weights w_i. We want to have the lowest possible preference value for the first and the highest possible preference value for the second game state. To achieve this, we add '100 minus the f value' for the first game states and simply the f value itself for the second game states. The mathematical formula of $score_k$ is as follows:

$$score_k = \frac{(t_i \times 100) - \sum f_i + \sum f_j}{t_i + t_j},$$

where i represents the game state of peace and j the game state of war. t is the number of turns of a specific game state. The *score* with the highest value represents the best weights.

To find ideal preference limits for the game states, we need to look back again to the highest *score* and take the f values belonging to this score. By applying statistical formulas on the training data, we can calculate the limits. For example, a limit for peace could be 40 and a limit for war could be 60. This means that '$0 \leq$ preference for peace ≤ 40' and '$60 \leq$ preference for war ≤ 100'. Between 40 and 60 there is a grey area of uncertainty, no reliable predications can be made for f values in this interval. Figure 3 shows an illustration of this concept. Note that the t on the horizontal axis stands for time.

3.3 Implementation in CIV4

In order to build a player model which is capable of predicting when a player has a preference for either peace or war, we test our model with built-in AI characters. The following three AI characters are chosen for our experiments:

- *Alexander the Great* of Greece (philosophical and aggressive, tends easily towards warfare in order to conquer more land),
- *Gandhi* of India (spiritual and industrious, is generally peaceful), and

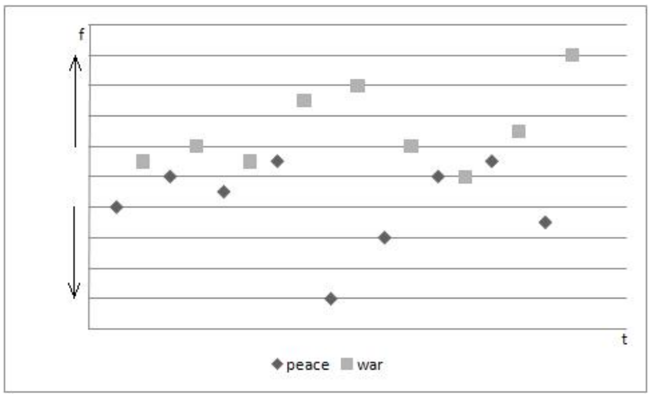

Figure 3: Concept of preference limits for peace, war and grey area in between.

- *Isabella* of Spain (spiritual and expansive, tends easily towards warfare when opponent has a different religion).

Each of these AI characters plays against one of the other two in a one-versus-one game, leading to six games in total. To assure consistent results, we use the same settings for each game. Since it is not possible in CIV4 to have two AI characters play against each other by default, a modification written in C++ and Python, called AIAUTOPLAY [8] is used to run the test games. In order to keep track of the mentioned two variables, the Python part of AIAUTOPLAY was customized, so that after each turn the variables get stored in a file.

A character's game state is determined by the following two variables:

- the difference in the number of *cities*, and
- the difference in the number of *units*.

These differences are based upon the two participating characters. Also, when creating a player model for character x who is playing against character y, the differences are defined as '$x - y$'. When we fill in our two variables of interest in the above mentioned general formula for $V(s)$, we get the following formula:

$$V(s) = w_c s_c + w_u s_u,$$

where s_c indicates the difference in cities, along with its weight factor w_c and s_u indicates the difference in units, along with its weight factor w_u.

In order to predict when a character has a preference for either peace or war, we have to find out in what game state the character who made peace or declared war was before he declared it. Game states of peace are assigned to both the participating characters, since it takes two to maintain peace. In the case of war, only the character who declared war gets the game states of war assigned.

After assigning each s_c, s_u and game state to the belonging character, we have to choose a domain for w_c and w_u. It appears that the relation s_c:s_u is about 1:10. In other words, the difference between the number of cities is about ten times smaller than the difference in units. For this reason we choose the discrete domain of w_c to be $[-100, 100]$, with an incremental step of 10 and the discrete domain of w_u to be $[-10, 10]$, with an incremental step of 1. Each of these values is combined, leading to a total of 441 possible combinations.

Now we have all the information we need to calculate the $V(s)$ values. We create one table for each of the three participating characters, containing the w values on the horizontal and the s values on the vertical axis. Once we applied above mentioned $V(s)$ formula for each character, we determine V_{min} and V_{max}. After that we replace each $V(s)$ value with its f value using above mentioned f_i formula.

The next step is to calculate the *score* values. For this we need to look to the columns of each table, to isolate each combination of w_c and w_u. A *score* value is the weighted sum of the f_i values with the same weight factors. After calculating these *score* values, $score_{max}$ represents the column with the best weight factors w_c and w_u for the current character.

To find the ideal preference limits for peace and war, we take the f values of the column with the best weight factors. The ideal preference limit for peace is the average of two lower limits; every f value between 0 and this limit is considered to indicate a preference for peace. These lower limits are calculated by subtracting two times the standard error of the mean from the average f values for peace and for war. The ideal preference limit for war is calculated by taking the average of two upper limits; every f value between this limit and 100 is considered to indicate a preference for war. The upper limits are calculated by adding two times the standard error of the mean from the average f values for peace and war. In between the limits there is a grey area of uncertainty; f values in this interval can not be classified with a reasonable reliability.

Finally, before we test the performance of our player models, we introduce a fourth AI character: *Saladin* from Arabia (philosophical and spiritual, shows generally unpredictable behaviour). Each of the previous mentioned three AI characters plays a game against Saladin. Again, we create three tables with $V(s)$ values and eventually the normalized f values. For each of the three AI characters, we take the f values of the column with the ideal weight factors which we learned from the training games. After numerically sorting these values, we calculate the percentage of correct and incorrect predictions compared to the preference limits we calculated.

4 Results

This section gives an overview of the results when applying our method described in the previous section.

	w_c	w_u	$score_{max}$
Alexander	100	10	54.4
Gandhi	-100	-10	52.3
Isabella	-100	10	52.7

Table 1: Ideal w_c and w_u values based upon $score_{max}$.

	avg	std	sem
peace	48.0	11.2	0.3
war	68.2	16.2	1.0

Table 2: Statistical values for Alexander.

	avg	std	sem
peace	47.4	13.5	0.4
war	28.7	2.2	0.5

Table 3: Statistical values for Gandhi.

Table 1 shows the ideal weight factors with their accompanying $score_{max}$ values. Tables 2, 3 and 4 show the average, standard deviation and the standard error of the mean for respectively Alexander the Great, Gandhi and Isabella. These values are based upon the f values of the best weight factors. It appears that for Gandhi and Isabella the average f's for peace are higher than those for war. For this reason, the models for this two characters are not good (more about this in Section 5). The model for Alexander the Great could be right, so we only test him against Saladin. The results against Saladin are shown in table 5. Note that incorrect means both a wrong prediction. Based upon the statistical data, we calculated that Alexander the Great has a preference for peace when $f \leq 56.8$ and a preference for war when $f \geq 59.3$. In between these limits is the grey area. It appears that the model did a correct prediction for Alexander the Great in 67.4% of the cases, while in 2.8% of the cases there is no prediction (grey area). From this test we can conclude that the model for Alexander the Great is a good representation of his behaviour (over $\frac{2}{3}$ of the cases is predicted correctly).

5 Discussion

Before we start the discussion of our results, note that a player model for the CIV4 AI characters can not be constructed with a 100% accuracy, because all decisions of

	avg	std	sem
peace	47.2	10.9	0.3
war	32.0	4.8	1.8

Table 4: Statistical values for Isabella.

	peace	war
correct	53.4%	14.0%
incorrect	29.8%	0.0%

Table 5: Performance for Alexander the Great.

the characters are influenced by random numbers[2]. Also, we are not sure if the internal variables used for decision-making when dealing with the game states peace and war are the same. It could be possible that the decision-making process for peace uses other variables than the process for war.

Each of the three characters we use to train our model provides lots of data as it goes for the game state of peace. There are two reasons for this. The first reason is that peace is the most occurring game state in all the training games. The second reason is that the game state of peace gets appointed to both participating players, while the game state of war only gets appointed to the character declaring war.

For Alexander the Great we have enough data for both peace and war to draw reasonable conclusions, although improvements are possible with more variables. Also, with more training games we could set more accurate preference limits, because all the incorrect predictions occurred during peace. A change in the limit for peace could already improve the model.

Gandhi barely ever declares war and Isabella only seems to declare war if the competing character has a different religion. The latter is rarely the case on the small maps where we perform our training games. Also, we do not take the variable religion into account. As a consequence, we have too little information about war behaviour for Gandhi and Isabella. For this reason, we currently can not build a player model for these two characters. There are two possibilities to resolve this. The first one is to take a greater range for w_c and w_u. The results could differ, but it is not most likely. The second and best possibility is to take other variables than s_c and s_u. These two variables seem almost nothing to predict as it goes for the prediction of war for Gandhi and Isabella, especially for Isabella the introduction of a religion variable could improve a lot.

[2]This statement is based upon personal communication with Soren Johnson, lead designer and AI programmer of CIV4.

6 Conclusions and future work

Subsection 6.1 concludes and Subsection 6.2 discusses future work.

6.1 Conclusions

The player models implemented in this research are successful in predicting a preference for peace or war for those AI characters driven by a desire to conquer another civilization when they have a supremacy on the battlefield. For AI characters more focussed on building and developing, results are less clear. To improve the results for such AI characters, other variables like culture, religion or research could be taken into account.

Since CIV4 is currently one of the most complex TBS games available, we can conclude that our preference-based player model is capable of making reasonable predictions of AI characters' behaviour as it goes for diplomacy in TBS games.

6.2 Future work

A first recommendation is to create and implement an algorithm which is able to perform real-time predictions of peace and war. This can be realized by training a model of a certain player with data from previously played games and live calculations during the current game.

In future research we will also build a player model with more and/or other variables than the two that were used in the current research. At least the comparison of the performances of different player models containing different variables will shed some light on which variables represent a certain game state the best.

A last recommendation is the creation of a self adapting weight algorithm. Currently the weights are precalculated and static. Having the weights automatically adjusted to better values during gameplay improves the performance of the player model's predictions for specific game states.

References

[1] 2K Games (2005a). Civilization IV. `http://www.2kgames.com/civ4/`.

[2] 2K Games (2005b). Civilization IV Manual.

[3] A&E Television Networks (2006). The History of Toys and Games. `http://www.history.com/exhibits/toys/`.

[4] Billings, Darse, Davidson, Aaron, Schaeffer, Jonathan, and Szafron, Duane (2002). The challenge of poker. *Artificial Intelligence*, Vol. 134, No. 1-2, pp. 201–240.

[5] Carmel, David and Markovitch, Shaul (1993). Learning models of the opponent's strategy in

game playing. *Proceedings of the AAAI Fall Symposium on Games: Planning and Learning* (eds. Susan Epstein and Robert Levinson), pp. 140–147, AAAI Press, Menlo Park, CA.

[6] Donkers, Jeroen and Spronck, Pieter (2006). Preference-based player modeling. *AI Game Programming Wisdom 3* (ed. Steve Rabin), pp. 647–660, Charles River Media, Boston, MA.

[7] Egnor, Dan (2000). Iocaine powder. *ICGA Journal*, Vol. 23, No. 1, pp. 33–35.

[8] jdog5000 (2006). AIAutoPlay modification component for Civilization IV. `http://forums.civfanatics.com/showthread.php?t=174812`.

[9] Spronck, Pieter (2005). A model for reliable adaptive game intelligence. *IJCAI 2005 Workshop on Reasoning, Representation, and Learning in Computer Games* (eds. David Aha, Héctor Munoz Avila, and Michael van Lent), pp. 95–100, U.S. Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington, DC.