Characterizing and Modeling Agents in Digital Games

Marlos C. Machado and Gisele L. Pappa and Luiz Chaimowicz
Department of Computer Science
Federal University of Minas Gerais
Belo Horizonte, Brazil

Abstract

A promising approach in digital games is the possibility of customizing the game according to different demands. Artificial Intelligence algorithms play an important role in this direction, allowing the implementation of different behaviors for game agents. To accomplish this, it is necessary to model these agents in such way their behavior can be easily tunned to address different game features. In this paper, we discuss a generic representation to model virtual agents in digital games. Agents are modeled using a linear combination of different variables, which are used to represent specific game features. We perform experiments with FPS and Strategy games (COUNTER STRIKE and CIVILIZATION IV, respectively) and results show the effectiveness of this approach in characterizing and modeling agents. We were able to infer agents models by observing matches and also to generate different behaviors varying agent's models.

Keywords:: Agents Modeling and Characterization, Computer Games, Player Modeling, Adaptive Game AI

Author's Contact:

{marlos, glpappa, chaimo}@dcc.ufmg.br

1 Introduction

In most digital games, agents' behavior is modeled using simple Artificial Intelligence (AI) techniques such as Finite State Machines and Decision Trees. The implemented behaviors are generally static and do not allow changes and adaptation [Bakkes et al. 2009]. But in recent years, an increase in the processing power of computers and game consoles, combined with a greater interest in game AI, from both the industry and academia, have propelled the study and development of more sophisticated agent behaviors in games. As a consequence, we have seen the development of more immersive and challenging games.

As discussed by Manovich [2001], Taylor [2002] and Bakkes et al. [2009], immersion can be considered a general measure of entertainment. It is generally related to how absorbing and engaging a game is. In general, predictable behaviors break down the immersion and, consequently, make games less fun. For example, a predictable behavior may allow the player to discover a specific opponent weakness and repeatedly explore it during the game. Charles and Black [2004] affirm that "Often this means that the player finds it easier to succeed in the game but their enjoyment of the game is lessened because the challenge that they face is reduced and they are not encouraged to explore the full features of the game".

Thus, adaptability is becoming a key feature in modern video games. By adaptability we mean the ability of adapting agents' behavior to specific game conditions. As an example, in a soccer game, the computer controlled agents may evolve their tactics to avoid always suffering goals in the same way. This makes the game more interesting and challenging. Note that this adaptability does not prevent the game designer to define a consistent game, with a well defined structure. The game continues with its predefined structure, but does not present an obvious weakness.

There are several characteristics that must be considered when designing adaptable behaviors in games, such as *when*, *how* and *why* to do it. Three possibilities of *when* to change behavior are, for example, dynamically in a game level, between levels, or between games. It can be done autonomously, or by some kind of player interference, such as selecting a difficulty level. Furthermore, the reasons for the change must also be stated since it may occur due to specific game features, or to general player performance. In fact, several works in the literature consider that modeling player behavior is an essential action towards an adaptive game AI [Charles and Black 2004], [Charles et al. 2005], [Spronck 2005], [Laviers et al. 2009].

In order to implement adaptable behaviors in games, it is necessary to use a representation capable of modeling agents according to different features of the game environment. This paper evaluates a methodology that is able to generically model agents in digital games. This methodology uses a representation that is a weighted sum of variables and can be applicable to most agents. Additionally, as discussed in the following sections, it satisfies some important industry requirements such as *customization* and *explicitness* [Isla 2005]. We base our framework on the work of Houlette [2003], performing experiments with different game genres in order to evaluate the representation. We were able to show the effectiveness of this approach in characterizing and modeling agents. Applying this framework on the strategy game CIVILIZATION IV[1] we are able to infer model parameters by observing different matches. In the opposite way, we use the framework to generate different agents behavior on COUNTER STRIKE[2], a typical First Person Shooter (*FPS*) game.

This paper is organized as follows: the next Section discusses some of the related work in the area. In Section 3, we present and detail the proposed model. Section 4 discusses the representativeness of this model and its applicability in commercial games. In Section 5 we validate this model analyzing it in two different games. Finally, Section 6 concludes this paper and discusses some future directions.

2 Related Work

Most works related to agents' models for games that allow adaptation are tightly related to player modeling, which aims at the generation of models based on the player characteristics and behaviors. Two taxonomies that try to review and organize the area have been proposed almost at the same time [Machado et al. 2011a] and [Smith et al. 2011]. In these papers several works are discussed and categorized. Another work with this goal is [Bakkes et al. 2012], where the authors present a survey of the field.

One of the first works that proposed a more complex characterization for virtual agents was [Houlette 2003], where the author suggested a model that would be able to represent players and virtual agents preferences in more complex games such as *First Person Shooters*. Houlette described this representation as "a collection of numeric attributes, or *traits*, that describe the playing style of an individual player" [Houlette 2003].

Several works have already proposed player and agent models for different platforms and goals but the majority of them is tailored to very specific applications. As far as we know, one of the few works that discusses a general methodology for obtaining models through different environments is [Charles and Black 2004]. In spite of that, it does not validate its assumptions in a real game. A sequence of this work is presented in [Charles et al. 2005], where the authors also discuss a high-level framework for adaptive game AI. They briefly present an approach for player modeling with factorial models but do not investigate it further.

[1] Firaxis Games, 2K Games, Civilization(2005): http://www.2kgames.com/civ4/

[2] Valve Corporation, Counter Strike(1999): http://www.valvesoftware.com/games/

Some of the most recent works, now tailored to specific games, are [Machado et al. 2012] and [Spronck and den Teuling 2010], which identified different agents preferences in the game CIVILIZATION IV with kernel machine techniques; and [Drachen et al. 2009] where the authors used neural networks to discover different player types in the game TOMB RAIDER: UNDERWORLD. One of the problems of these and other approaches [Laviers et al. 2009], [Tan et al. 2011] is that they are related to complex classifiers that are difficult to understand, generate and use in generic applications.

Other approaches for agents' modeling are presented in [Machado et al. 2011b] and [Doirado and Martinho 2010]. In the first work the authors discuss agents characterization in the game CIVILIZATION IV using a methodology based on linear regressions to model agents preferences. Finally, Doirado and Martinho [2010] present an interesting work that modeled agents intentions, proposing a framework called DOGMATE.

As mentioned, the agent representation presented in this paper is based on the model proposed in [Houlette 2003]. This representation was introduced almost as a theoretical model since no real implementation or evaluation of its feasibility was presented. Thus, our main contribution is to present an experimental evaluation of the method, applying it in different commercial games, both for modeling and characterizing agents. We also discuss its applicability considering the industry requirements listed in [Isla 2005].

3 Agent Representation

The representation presented in this paper to model game agents is based on two main components: a set of variables representing specific features of the game and a set of weights that multiply these variables. The set of variables is determined by the game designer and the AI programmer, and it is based on the *domain knowledge* of each game, an inevitable requirement for adaptive AI as Spronck [2005] and Bakkes et al. [2009] previously stated. The weights represent the importance that a specific agent gives to the feature represented by that variable. They can be manually set by the designer / programmer or learned from experience. Completely different behaviors can be obtained varying the weights of each agent, which allows agents to adapt to different game conditions.

More formally, the model is represented as a weighted sum of n different variables:

$$Pm = \sum_{i=1}^{n} w_i \times c_i,$$

where w_i is a weight for characteristic c_i of the model Pm.

This representation is generic and can be applied to various games such as *Chess*, *Poker*, *FPS* and *Strategy* games. It is very simple, yet powerful enough to represent and model different agent behaviors. The main characteristic of this representation is that it is defined as a vector of weights for different game features, that can be agent's preferences or knowledge, for example. This modeling is generic but the model's features/variables are particular to each game and must be defined by an expert. The main advantage of this approach is that, once techniques are created to infer weights, they can be applied to different games that use this approach.

As a didactic example we may use *Poker* to illustrate our discussion. It is a very hard game for computers to play and, so far, they are not able to defeat the best human players. A promising approach is player modeling and there are many works in this area such as [Billings et al. 1998], [Davidson et al. 2000], [Southey et al. 2005] and [Bard and Bowling 2007].

To apply the proposed representation to model *Poker* players, it is necessary to represent different agent behaviors. A very simplistic model, created just for this example, can be extracted analyzing relevant game features such as agents temerity (T_e), probabilistic capacity (P_c) and bluff tendency (B_t). We could define each of these variables as:

- T_e representing the agents tendency to take risks;
- P_c representing agent's awareness about the probability of winning given a card distribution in a match;
- B_t describing how much does an specific agent bluffs during a game.

Once we created this simple set of variables we generate the following representation, capable of modeling completely different agents:

$$Pm = w_1 T_e + w_2 P_c + w_3 B_t.$$

To exemplify, defining weights between 0 and 1 and a higher value representing a stronger preference, we could easily generate an expert conservative player, represented as $Pm_1 = 0.1T_e + 0.9P_c + 0.2B_t$ or an aggressive player, $Pm_2 = 0.8T_e + 0.6P_c + 0.5B_t$.

Thus, this representation implies in two different tasks to generate an agent model:

1. To define the model variables: with the *domain knowledge* of the specific genre or game, define what are the relevant features to be modeled and in what level of abstraction it will be done.
2. To define the variables weights: this is what distinguishes behaviors, *i.e.*, different behaviors are set in this second step. It can be done to model non-player characters as well as to human players.

The definition of variables is an easier task since we just need to define what we want to model, but the weights setup is a very hard task for several reasons. Maybe the biggest one is the fact that there is no rule of thumb for doing it. Generating these weights (or other type of modeling) from repeated play, observing players and other agents and tracking game results, often involves the use of machine learning techniques and the most used are those inspired in nature as neural networks or genetic algorithms. A work that can be seen as performing this task setting weights only equal to 0 or 1 in the game CIVILIZATION IV is [Machado et al. 2012], despite not directing using this discussed model. The approach used is based on Support Vector Machines (SVMs), a machine learning technique.

Finally, once we have defined a representation for digital agents we need to certify its applicability. In the next section we will discuss this topic and how to evaluate the model representativeness.

4 Applicability

4.1 Representativeness

A model, to be useful, must be capable of satisfactorily represent different agents with different characteristics. This topic includes the capacity of the methodology to represent agents allowing *any* behavior derived from the model, what could be seen as a coverage requirement. As we already discussed, once appropriate features are selected, this is completely feasible.

Furthermore, to evaluate this model representativeness, it must allow two tasks to be performed. These tasks are:

1. The generation of different behaviors by varying the model. In our approach, it would be different weights generating different behaviors; and
2. A model that generates a specific behavior must be "inferable", i.e., one must be able to infer a model that generates an observed behavior. This task, in our modeling, consists in inferring variables weights from observation.

The requirements listed above create a cycle: once we observe a specific behavior we must be able to infer the model which generated it and we must be capable of generating different behaviors from the model.

We present the results for both tests in Section 5, corroborating our assumption about this representation usefulness, since we were able to perform both tasks. Our experiments were performed in two commercial games, CIVILIZATION IV and COUNTER STRIKE. This flow is presented in Figure 1.

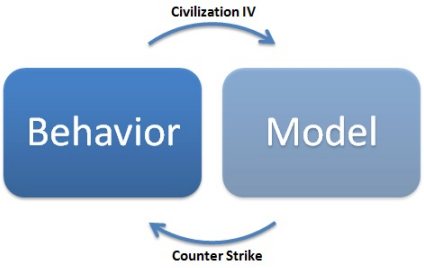

Figure 1: *Summary of the proposed evaluation.*

4.2 Applicability in Commercial Games

An important topic to discuss before presenting our results is related to the applicability of this modeling in commercial games. In general, the game industry is somewhat reluctant about the game AI solutions proposed by academy. As discussed by Fairclough et al. [2001], Laird and Lent [2001], Nareyek [2004] and Bakkes et al. [2009], there are several reasons for this: the concern about unpredictable behavior, the necessity of heavy modification and specialization for each game, and the difficulty in understanding the reasons for some observed behaviors and in modifying any configuration.

We argue that due to its simplicity and expressiveness, the representation presented in this paper avoids most of these problems. We use as a base for this discussion the work of Isla [2005], which presents the AI architecture of a very well-succeeded game, HALO 2[3], and discusses several design principles that should be targeted when developing the AI of a complex game.

First of all, Isla defines some basic AI requirements: *coherence, transparency, run-time, mental-bandwidth, usability, variety* and *variability*. Coherence is related to actions' selection at appropriate times, *i.e.*, how we select actions once we have defined an agent model. Thus, coherence is much more related to the mapping between agent models and actions than to the model itself. On the other hand, transparency is one of the major points of our representation, since we are able to understand agents and even try to predict its behaviors only observing the linear combination and its weights. This is also related to mental-bandwidth, since we need to "reason about what's going on in the system" [Isla 2005].

The representation is also important and useful to level designers since it allows usability (the characters are configurable since we only need to change their weights) and variety (the AI works differently for each different weight combination). Variability, as many of the discussed requirements, is related to the use of the model. It is important to observe that, once this model is defined, it is necessary that game programmers use the weights paradigm to develop their games. It is also important to note that this representation does not limit or hinder any desirable feature as neural network or kernel machine models generally do. Moreover, this approach, due to its simplicity, does not impact game performance.

Once we discussed the basic AI requirements, we are able to show that this representation goes further and also meets the design principles proposed by [Isla 2005]. The first one is *"Everything is customizable"*, *i.e.*, the model should be general enough to allow modifications in behaviors. This is exactly the main advantage of this representation since it can be seen as an organized set of parameters, a clear definition of customization.

The second design principle, *"Value explicitness above all other things"* is obviously met by the explicit use of weights. Also, the use of weights simplifies the generation and test of specific behaviors and makes easier the process of generating relatively similar behaviors with some small variations in each weight. This is exactly Isla's fourth principle: *"Take something that works and then vary from it"*. (The third principle, *"Hackability is key"*, is not applicable directly to the representation but to other programming levels).

It is interesting to note that, despite the independent development of [Houlette 2003] and [Isla 2005], it seems they were developed together because of the similarity in most of the discussed requirements. A final sentence of [Isla 2005] evidences its usefulness: "... we are not interested in a scripting system in which the designer specifies EVERYTHING the AI does and where it goes - that would be too complex. We do need, however, the AI to be able to handle high-level direction: direct them to behave generally aggressively, or generally cowardly.".

5 Experiments

We performed two types of experiments using two different commercial games in order to show the representativeness of the discussed model. They follow the approach discussed in Subsection 4.1 to evaluate the model's representativeness.

In the first set of experiments, we tried to infer the model of different agents of the game CIVILIZATION IV, using an approach derived from [Machado et al. 2011b]. Basically, from behavior observation, we try to infer some of the weights that could model the agents and compare them with the predefined agent model. This shows how different behaviors can be explained and expressed by different models.

The second set of experiments goes in the opposite direction. Using the game COUNTER STRIKE, we model different agents varying the model weights and observe the resultant behaviors in the game. This shows how small changes in the model can generate different behaviors in the game.

Figure 1 summarizes this approach: using CIVILIZATION IV we try to infer the model from the observed behaviors, while using COUNTER STRIKE we show how different models result in different behaviors.

We have selected these two games to perform the experiments due to the interface they offer to the programmer. Different game developers have different approaches to data extraction and game modification and it is important to evaluate them before planning the experiments. We were not able to use one unique game platform due to this limitation. A discussion about several game platforms and its possible use in game research is available in [Machado et al. 2011a].

CIVILIZATION IV presents a simple interface for data extraction and has agent models, easily adapted to the one discussed here. These two characteristics allowed us to extract gameplay data, analyze it, and infer an agent model, comparable to the one already available in the game. Unfortunately, it does not easily allow AI modification, what the game COUNTER STRIKE allows. This is why we selected it in the second set of experiments. COUNTER STRIKE does not provide an easy interface for data extraction, not allowing us to use it in the first set of experiments.

5.1 Civilization IV: from behaviors to models

Our first experiment aimed at showing that it is possible to infer weights for the discussed representation, i.e., we show that it is possible to infer models observing behaviors. We have used the game CIVILIZATION IV to perform our tests, since it allows data to be easily extracted and explicitly presents agents characteristics in *XML* files, what allowed us to convert it to our representation.

CIVILIZATION IV is a Turn-Based Strategy Game (TBS) where each player controls a civilization and evolves it until its annihilation or victory (Figure 2). The main goal of a player is to overcome all the others. This can be achieved in six different ways, called victory conditions, ranging from peaceful conditions, as diplomacy, to military conditions. This is one of the reasons this game is so interesting to study: different agents are modeled intending to generate behaviors that seek different victory types, assuring a "preference" in the game.

[3]Bungie Studios, Microsoft Games Studios, Halo 2 (2004): http://www.microsoft.com/games/halo2/

Figure 2: *Snapshot of* CIVILIZATION IV.

As previously stated, CIVILIZATION IV provides *XML* files with several game characteristics, such as agents preferences, buildings and units info. This interface allows us to observe and modify the game characteristics [Spronck and den Teuling 2010]. Each agent presents several attributes in the XML defining them. This is quite useful since we are able to correctly generate relevant variables for the model. Six agents preferences are defined in the game, they are: *Culture, Gold, Growth, Military, Religion* and *Science*. Each of these preferences serve as multipliers to agents decisions and cost of specific actions.

Since we do not intend to present the best possible model for the game, we decided to use the preferences described and valued in the *XML* to generate an agent representation:

$$Pm = w_1 C_u + w_2 G_o + w_3 G_r + w_4 M_i + w_5 R_e + w_6 S_c,$$

where C_u is preference for culture, G_o for gold, G_r for growth, M_i for military, R_e for religion and S_c for science.

Our goal is to infer different models from different behaviors, to verify that this representation is coherent with observed behaviors. To do it we selected two different agents where one of them has a high preference for a specific feature while the other does not. Our goal is to analyze game indicators that would, theoretically, be affected by this preference in order to compare behaviors to infer weights for the representation above. We expect higher weights for the agent with a higher preference.

For example, agent *Alexander* has a strong *Military* preference while agent *Hatshepsut* prefers *Culture*. These agents, who were randomly selected, may be represented by us using the information available on the *XML* files. Extracting the respective weights from the configuration files we obtain the following models (Pm_A for *Alexander* and Pm_H for *Hatshepsut*):

$$Pm_A = 0C_u + 0G_o + 2G_r + 5M_i + 0R_e + 0S_c$$

$$Pm_H = 5C_u + 0G_o + 0G_r + 0M_i + 2R_e + 0S_c$$

As we can see, the set of variables different from zero of each agent is a disjoint set and it is expected to result in distinct behaviors during the game, assuming that the above representation is useful.

One of our previous works [Machado et al. 2011b] characterizes behaviors of several agents with linear regressions and shows differences between different agents. We will make a similar evaluation here, instantiating our modeling to the representation presented in Section 3, which was not used in [Machado et al. 2011b].

To generate matches between two agents without human interference and log them, we used an script called *AIAutoPlay*, developed in [Spronck and den Teuling 2010]. The used dataset is the same presented in [Spronck and den Teuling 2010] and we use it to look for different behavior patterns, respecting the agents selection described above. This dataset contains 40 matches of each agent, and each match is composed of, at most, 460 turns.

We processed this data computing its mean for the 40 games played by the analyzed agent, generating 460 points, each one responsible for one turn. The plots for *Alexander* and *Hatshepsut* overall culture score along all matches is presented in Figure 3. An analogous plot, but for *Culture Rate*, is presented in Figure 4. *Culture* is defined as "overall culture score" and *Culture Rate* as the "amount of culture gained per turn" [Spronck and den Teuling 2010].

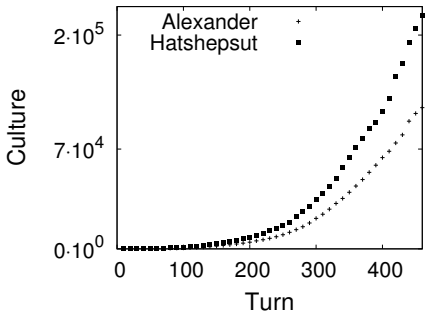

Figure 3: *Comparison of Culture between different agents.*

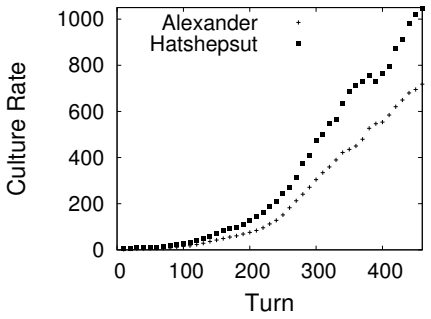

Figure 4: *Comparison of Culture Rate between different agents.*

As expected, based on the agent's models, *Hatshepsut*'s curves represent a superior bound for *Alexander*'s curves. This is coherent with *Hatshepsut*'s higher preference for *Culture*.

To verify an statistical difference, we performed linear regressions to these data looking for statistically different coefficients in the straight-line equations. Since the data represented in the above figures were not linear, we applied transformations to make it linear. For Figure 3 we extracted the 5^{th}-root while for Figure 4 the 4^{th}-root. These results were presented by us in a previous work [Machado et al. 2011b] where we characterized several agent's behaviors. Our main contribution here is to show the representative power of the discussed model, showing that it is possible to infer weights from observation; while in [Machado et al. 2011b] we just characterized agents' behavior without even mentioning a specific representation.

After the data processing, we obtained the following line equations for *Alexander* (y_A) and *Hatshepsut* (y_H), for *Culture*. Between parenthesis we present R^2, i.e., the coefficient of determination[4], a general metric of how good the regression is in predicting other values.

$$y_A = 0.0183x + 1.7772 \ (99.86\%)$$
$$y_H = 0.0194x + 2.1366 \ (99.85\%)$$

Using the same pattern, the respective line equations for the *Culture Rate* indicator were:

$$y_A = 0.0096x + 1.0939 \ (99.93\%)$$

[4] "The fraction of the variation that is explained determines the goodness of the regression and is called the coefficient of determination, R^2" [Jain 1991]

$$y_H = 0.0101x + 1.3567\ (99.11\%)$$

Finally, we were able to show, using *paired t-tests*, that both coefficients are statistically different with a confidence of 99%.

After this characterization we can clearly see that *Alexander* indicators are bounded below those from *Hatshepsut*. Then we can infer that $C_{uA} < C_{uH}$. Simplifying the assignment, assuming that we have only two different values, 0 and 5, we can say that $Pm_A = 0C_u + ...$ and $Pm_H = 5C_u + ...$, which corresponds to the agent original model.

We performed the same operations for other two randomly selected preferences using an indicator that is related to a variable that is the same in both agents. Our objective here was to show that it is possible to generate models in the discussed representation from data collected during play.

The two other preferences modeled are *Growth* and *Gold*, represented by variables G_r and G_o respectively. For the *Growth* variable we characterized three different indicators, *Cities*, *Lands* and *Plots*. The first one is defined as the "Number of cities", the second as "Amount of land tiles" and the third as "Amount of land and water tiles". The Gold variable was characterized with *Gold* and *Gold Rate*, respectively "Amount of gold" and "Amount of gold gained per turn".

We used the same two agents, *Alexander* and *Hatshepsut*, to obtain G_r weights. We divided the growth data in two different periods: an expansionist, when there still exist unoccupied lands that are easily dominated, and a maintenance, when there is almost an stabilization of the indicators since all the world has already been "colonized" by some agent. This division was made due to the game characteristics and it allowed us to clearly see two different behaviors for the analyzed agents using the indicator *Cities*. From the obtained data, with a confidence of 99%, we can say that the linear equations are different for each agent on the expansionist period. In fact, *Alexander* has a slight preference for growth over *Hatshepsut*, which corresponds to the coefficients assigned in the original model. Both equations for the expansionist period (turns 1:220), and its R^2 are presented below.

$$y_A = 0.03143x + 0.49439\ (97.17\%)$$
$$y_H = 0.02960x + 0.55610\ (96.80\%)$$

As previously said, a complete characterization, graphs and regressions for several indicators is presented in [Machado et al. 2011b], but without the discussion about the presented representation.

Finally, the last analyzed variable was G_o. We used two other agents in this task: *Louis XIV* and *Mansa Musa*, since *Alexander* and *Hatshepsut* had no preference for it in the presented models. Calling *Louis XIV* model as Pm_L and *Mansa Musa* as Pm_M we extracted the following model from CIVILIZATION IV's configuration files:

$$Pm_L = 5C_u + 0G_o + 0G_r + 2M_i + 0R_e + 0S_c$$
$$Pm_M = 0C_u + 2G_o + 0G_r + 5M_i + 0R_e + 0S_c$$

We were not able to differ agents preferences based on a unique indicator. We believe this is due to the fact that there are many interactions between gold and other game features. Another work that tried to classify agents preferences with machine learning techniques was [Spronck and den Teuling 2010] and their conclusions also corroborated with the fact that this preference is harder to be distinguished between agents.

This last result shows that, for some game variables, it may be necessary to use a different approach in order to obtain the weights. But it does not invalidate our claim that it is possible to use a simple agent model and infer some of its parameters from agents behaviors. In fact, we proposed an approach based on Binary Classification in [Machado et al. 2012] using Machine Learning and we were able to achieve an accuracy for *Gold* equals to 61.6% to completely unknown agents, what supports our claim.

Figure 5: *Snapshot of* COUNTER STRIKE.

5.2 Counter Strike: from models to behaviors

Once we were able to show that it is possible to infer weights for the discussed representation that are coherent to observed behaviors, we need to assure that we are capable of generating different behaviors from agent's models variation. As we already showed in Figure 1, we used the game COUNTER STRIKE to perform this task.

COUNTER STRIKE, shown in Figure 5, is a First-Person Shooter (FPS) game with two different teams that must defeat each other: Terrorists and Counter-Terrorists. The game is divided in rounds and each round is won or lost when its mission objective is achieved or when all members of the adversary team are killed. The mission goals can be (1) to rescue hostages or (2) to defuse bombs (this second goal was used in our experiments). Each round starts with both teams spawning in different parts of the environment and it lasts, at most, five minutes.

One of the reasons for choosing this game is the availability of a bot source code called *Ping of Death Metamod*[5] that implements AI behavior (the original game only works with human players, not permitting computer-controlled agents), allowing us to directly modify the source code. This feature is not common in commercial games, what makes COUNTER STRIKE very attracting for research purposes.

In COUNTER STRIKE matches, we can clearly see interactions in a multi-agent environment, allowing us to analyze the impact of different agents' models. It is interesting because, differently from CIVILIZATION IV, we are concerned with generating behavior from different models, not with inferring models from behaviors.

Once we had access to agents' behavior implementation in the bot source code, we were able to generate a specific model for them. In fact, as discussed in previous sections, we do not intend to generate the best possible model for agents in the game, but one that is capable to capture some relevant game characteristics.

We used the bot's source code to assist us on the modeling task since we base our model in the current implementation. The variables modeled were extracted from source code variables present in the bot's implementation. The final generated model is:

$$Pm = w_1 D_B + w_2 A_t + w_3 S_C + w_4 H_d,$$

where D_B is the priority to defuse bombs and A_t, S_C, H_d are, respectively, the agent's priority to attack, seek cover and hide. Note that the variables' level of abstraction is different when compared to those present in CIVILIZATION IV. This is mainly due to the interface offered. While we extracted high-level information in the CIVILIZATION IV *XML* files, COUNTER STRIKE offers to us a bot implementation, that is more focused on specific actions to be taken. These differences are interesting to support our claim that this representation can be used generically, for several different purposes.

Every agent in the game is represented by the equation above, terrorists and counter terrorists; despite D_B having no meaning for

[5] `http://podbotmm.bots-united.com/main_pb_page/index.htm`

the terrorist team. Using the pre-defined source code values of the variables, we obtain the following agent model, which represents an initial bot status.

$$Pm = 0.88D_B + 0.90A_t + 0.91S_C + 0.92H_d$$

After defining an agent model, we evaluated if different models generate different behaviors. To do this, we changed counter-terrorists model's weights and analyzed if this change caused any impact in gameplay. In order to objectively evaluate changes in gameplay we compared match results between two team's of virtual agents, with only one having its behavior varied. Besides that, we also compared bots' actions frequency.

Since we do not intend to perform a complete characterization of each variable and its impact in gameplay, but to show that different weights do cause a behavior variation, we decided to vary weights of one variable. We have chosen w_1, associated with D_B to validate our assumption due to its semantics, since defusing bombs priority is certainly an important task in COUNTER STRIKE.

The game may have up to thirty two players simultaneously, but we decided to run our experiments with teams composed of four members, the initial number defined by the bots implementation. We executed the game in the scenario *de_inferno*, which has two different locations where terrorists can plant the bomb.

Our experimental setup consisted in running the game ten times, with a fixed time of 30 minutes each, varying w_1 value and recording the game results. With this approach we were able to evaluate the average time of each round (each game contains several rounds) and the team's performance. We defined a w_1 starting value equals to 0.88 (initial pre-defined value) and added/subtracted 0.11 from this variable. The higher value was 0.99, representing a very high priority in defusing bombs. At the end, we generated the following set of values: $\{0.55, 0.66, 0.77, 0.88, 0.99\}$.

To analyze game result with absolute scores we have modeled each experiment of thirty minutes as a zero-sum game where each victory raises the terrorists team score. The following formula represents the score (S) presented in our discussion:

$$S = \frac{V_T - V_{CT}}{V_T + V_{CT}},$$

where V_T is the number of rounds won by Terrorist team and V_{CT} those won by Counter-Terrorist team. All the results presented below used analysis based on two-side intervals with a confidence of 95%.

After performing the experiments we calculated the average score of the counter-terrorist team for different values of w_1 and its confidence interval. These data are presented in Table 1.

w_1	Score $\pm$ C. Interval	Std. Dev.
0.55	0.73 ± 0.40	0.56
0.66	-0.37 ± 0.26	0.36
0.77	-0.61 ± 0.20	0.28
0.88	-0.42 ± 0.30	0.41
0.99	-0.44 ± 0.28	0.39

Table 1: *Performance of Terrorist Team varying w_1.*

To ease the data analysis and discussion, we also plotted the results of Terrorist team in Figure 6. The vertical lines are the confidence interval.

Table 1 and Figure 6 present interesting results about different agent models at the defined environment. At a first look, a result that is highlighted is the high variance of data. It was expected due to the randomness of bots both in terms of behaviors and goals during all matches. This random outcome is what keeps players interested, because every game becomes different.

Analyzing the data we can affirm that as w_1 increases, meaning that the counter-terrorists have a higher priority for defusing bombs,

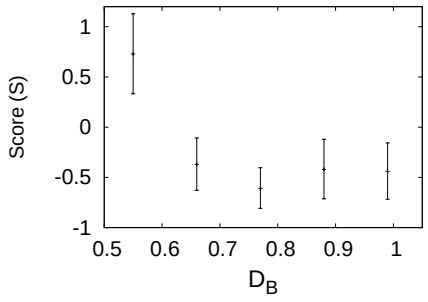

Figure 6: *Terrorist Team Performance varying w_1*

the score of the terrorist team decreases. In other words, by prioritizing defusing bombs, the counter-terrorist win more matches in this scenario. The w_1 confidence intervals for 0.88 and 0.99 overlap, meaning that we cannot state that we observed different results with statistically significance. Nevertheless, we were able to show that the score obtained when $w_1 = 0.77$ ($S_{0.77}$) is lower than those obtained by $w_1 = 0.88$ ($S_{0.88}$) and $w_1 = 0.99$ ($S_{0.99}$), with a confidence of 95%. We were also able to show that $S_{0.55} > S_{0.99}, S_{0.88}, S_{0.77}, S_{0.66}$; that $S_{0.66} > S_{0.77}$ and that $S_{0.66} < S_{0.88}, S_{0.99}$. Thus, we can observe that a simple variation in the agent model drastically impacts the general game result due to variations on agents behaviors.

To corroborate our hypothesis that different weights generate different behaviors, we also counted how many times the bots from the Counter-Terrorist team took each action. This experiment aims to explain the results presented above. We decided to analyze the Counter-Terrorist team because it is the team affected by the variation of the D_B weight.

There are dozens of actions available to each bot. Some of them are rarely performed, such as "throwing an smoke grenade". Due to that we decided to analyze the six most frequent actions taken, namely:

- *TASK_NORMAL* (Normal (roaming) Task)
- *TASK_MOVETOPOSITION*
- *TASK_CAMP* (If Bot is camping, he should be firing anyway and NOT leaving his position)
- *TASK_DEFUSEBOMB*
- *TASK_ATTACK*
- *TASK_SEEKCOVER*

We did not presented all actions descriptions because we believe its names describe their use. The descriptions presented between parenthesis were copied from source code comments.

After running the 10 rounds of 30 minutes we calculated the number of performed actions. To ease the comparison we normalized all the numbers between 0 and 1 (dividing the count by the number of actions), since each match has a different number of actions. The obtained results are plotted in Figure 7.

The total of actions taken in different weights configuration were compared with paired t-tests using a confidence of 90%. We also ordered these numbers considering the confidence interval obtained, we call this ordering as statistical ordering, since we do not just evaluate the average but also the confidence interval.

A first analyzed variable is *TASK_DEFUSEBOMB*. We were able to observe that the amount of attempts to defuse a bomb was lower when $w_1 = 0.55$, while we were not able to statistically distinguish the attempts frequency between $w_1 = 0.77$; $w_1 = 0.88$ and $w_1 = 0.99$. We have seen some unexpected results when $w_1 = 0.66$, since it presented a higher number of actions for defusing a bomb than teams with higher priority for this task. This is also reflected in the number of victories obtained since $S_{0.66} > S_{0.77}$, as previously said. We suspect that this game may have an optimum

parameter configuration and $w_1 = 0.66$ is the closer we were from it. This is a very complex question and must be further studied in the future. At the end, we were able to generate the following statistical ordering for the action frequencies when weight is equal to $w_1(F_{w_1})$: $F_{0.55} < F_{0.77} = F_{0.88} = F_{0.99} < F_{0.66}$.

Analyzing *TASK_NORMAL* actions taken in each configuration, we also observed a lower value for $w_1 = 0.55$ and a higher value for $w_1 = 0.66$. A final statistical ordering obtained was: $F_{0.55} < F_{0.99} < F_{0.88} < F_{0.77} \leq F_{0.66}$. It is hard to explain this behavior due to the genericity of this action, but we can observe that it is coherent with the *TASK_DEFUSEBOMB* pattern, and, most important, it is also affected by the weights variation.

Besides the frequency that bots try to defuse bombs, and default actions, we were also able to draw conclusions about other actions, not directly related to defusing bombs, showing that its variation do causes a behavior change. A first result that is clearly observable in Figure 7 is that the bot's team performed the action *TASK_MOVETOPOSITION* much more frequently when $w_1 = 0.55$. We were able to observe that for most of the w_1 values, a lower value implies in a higher count of this action. Table 2 summarizes all the obtained results with the average and standard deviation of each frequency. The generated statistical ordering was: $F_{0.99} < F_{0.77} < F_{0.88} \leq F_{0.66} < F_{0.55}$

We also observed an inverse behavior (higher weights implying in lower frequency) for the actions *TASK_ATTACK* and *TASK_SEEKCOVER*. With the following statistical ordering: $F_{0.55} < F_{0.66} \leq F_{0.77} < F_{0.88} < F_{0.99}$ for *TASK_ATTACK* and $F_{0.66} < F_{0.55} \leq F_{0.88} < F_{0.77} < F_{0.99}$ for *TASK_SEEKCOVER*.

Finally, the action *TASK_CAMP* kept a constant frequency among almost all weights. The final ordering by confidence intervals was: $F_{0.55} < F_{0.66} = F_{0.77} = F_{0.88} = F_{0.99}$. We present all the averages and standard deviations for all actions in Table 2.

Analyzing the presented results we can argue that a reduction in the priority for defusing bombs forces the counter-terrorist bots to search for the terrorist team, since they must kill all of them to win (increasing *TASK_MOVETOPOSITION*). A lower counting for attacking and seeking is probably due to the fact that they will not go, recklessly, to the place where the bomb is generally put. They do not fell the need of protecting this place.

An interesting topic is related to the generated game results. In general, the terrorists team present a much worse performance than counter-terrorists. We are able to make this claim because the reported score and its confidence interval is generally lower than zero. A zero score means an even game, so if zero is not in the confidence interval, we have a statistical confidence that the game is uneven between teams. This unbalance at the bots implementation may compromise player's satisfaction when using these bots to complete their teams.

To ease a graphical analysis of the discussed results we also plotted Figure 7 without the *TASK_NORMAL*, to allow a better visualization of the other frequencies. This result is presented in Figure 8.

One final note is that we did not observe any change in the duration of the game varying w_1. It seems that, in average, the time spent by a team to eliminate the adversary team or to plant/defuse the bomb is approximately the same.

6 Conclusion

In this work, we discussed and evaluated a generic representation to model virtual agents in digital games. This representation models agents behavior using a linear combination of different variables, which are used to represent specific game features. We performed experiments with different game genres and results show the effectiveness of this model. We have shown that the representation discussed is able to handle two opposing situations: the inference of models observing agents behaviors and the generation of behavior from models variation.

There are several paths for future work. First related to bots' behavior characterization and its parameters in a multi-agent environment, a future work is the attempt to find correlations between matches duration and actions count, trying to identify what type of action assures victory and how long does the team take to win.

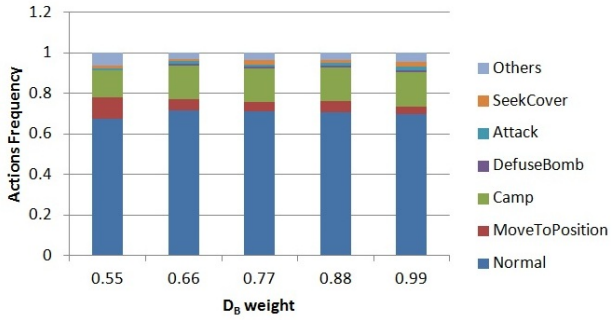

Figure 7: *Average frequency of actions performed by the Counter-Terrorist team during 10 turns of 30 minutes. The counters were normalized to be between 0 and 1.*

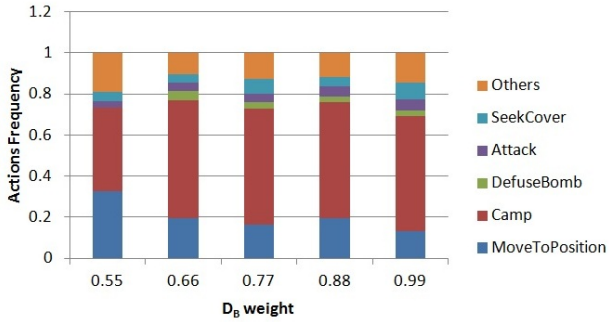

Figure 8: *Average frequency of actions performed by the Counter-Terrorist team during 10 turns of 30 minutes. The counters were normalized to be between 0 and 1, ignoring the action TASK_NORMAL.*

Related to the model discussed, a research branch is related to developing methods that are able to model humans instead of computer agents. Using this same representation, this task can be seen as an attempt to obtain the weights of each feature in a given model for different players.

A completely different topic that can also be extracted from this representation is the study of its application on game development process, evaluating its benefits for level designers and programmers developing the game.

Finally, we could also analyze how hierarchical representations would apply in practice. This topic was proposed in [Houlette 2003], where the author suggests the generation of weights that represent very low level actions such as *throw grenade* or *use rifle* and a hierarchical organization that extracts higher level information as the combination of its leaf nodes. Higher level representations could be *aggression* or *intelligence*, for example. Note that we have worked with both levels in this paper, while we used low level actions to represent agents in COUNTER STRIKE we used high level concepts to represent agents in CIVILIZATION IV.

Acknowledgments

We would like to thank Pieter Spronck who was always keen to solve any of our doubts related to the bots implementation for the game Counter Strike, what allowed us to use it as platform for validation of the discussed representation.

This work is partially supported by CAPES, CNPq and Fapemig.

Action Name / w_1	0.55	0.66	0.77	0.88	0.99
TASK_NORMAL	0.6753 (0.0303)	0.7168 (0.0361)	0.7124 (0.0288)	0.7063 (0.0368)	0.6972 (0.0326)
TASK_MOVETOPOSITION	0.1063 (0.0404)	0.0549 (0.0123)	0.0475 (0.0247)	0.0578 (0.0259)	0.0393 (0.0213)
TASK_CAMP	0.1324 (0.0424)	0.1632 (0.0339)	0.1622 (0.0240)	0.1648 (0.0363)	0.1699 (0.0436)
TASK_SEEKCOVER	0.0143 (0.0094)	0.0114 (0.0056)	0.0210 (0.0120)	0.0134 (0.0085)	0.0246 (0.0125)
TASK_ATTACK	0.0098 (0.0051)	0.0114 (0.0072)	0.0120 (0.0064)	0.0138 (0.0073)	0.0168 (0.0077)
TASK_DEFUSEBOMB	0.0000 (0.0000)	0.0127 (0.0032)	0.0090 (0.0051)	0.0088 (0.0044)	0.0089 (0.0044)

Table 2: *Frequency of actions for each weight configuration. The reported result is the average; standard deviation is between parenthesis.*

References

BAKKES, S., SPRONCK, P., AND VAN DEN HERIK, J. 2009. Rapid and Reliable Adaptation of Video Game AI. *IEEE Transactions on Computational Intelligence and AI in Games 1*, 2 (june), 93 –104.

BAKKES, S. C., SPRONCK, P. H., AND VAN LANKVELD, G. 2012. Player Behavioural Modelling for Video Games. *Entertainment Computing*.

BARD, N., AND BOWLING, M. 2007. Particle Filtering for Dynamic Agent Modelling in Simplified Poker. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, AAAI, 515–521.

BILLINGS, D., SCHAEFFER, J., AND SZAFRON, D. 1998. Opponent modeling in poker. In *Proceedings of the 15th National Conference on Artificial Intelligence*, AAAI Press, AAAI-98, 493–499.

CHARLES, D., AND BLACK, M. 2004. Dynamic Player Modelling: A Framework for Player-Centered Digital Games. In *Proceedings of International Conference on Computer Games: Artificial Intelligence, Design and Education*, 29–35.

CHARLES, D., KERR, A., MCNEILL, M., MCALISTER, M., BLACK, M., KCKLICH, J., MOORE, A., AND STRINGER, K. 2005. Player-Centred Game Design: Player Modelling and Adaptive Digital Games. In *Digital Games Research Conference*, Citeseer, vol. 285.

DAVIDSON, A., BILLINGS, D., SCHAEFFER, J., AND SZAFRON, D. 2000. Improved opponent modeling in poker. In *Proceedings of the International Conference on Artificial Intelligence*, AAAI Press, ICAI-2000, 493–499.

DOIRADO, E., AND MARTINHO, C. 2010. I Mean It!: Detecting User Intentions to Create Believable Behaviour for Virtual Agents in Games. In *International Conference on Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, no. Aamas, 83–90.

DRACHEN, A., CANOSSA, A., AND YANNAKAKIS, G. N. 2009. Player Modeling using Self-Organization in Tomb Raider: Underworld. In *Proceedings of the 5th International Conference on Computational Intelligence and Games*, IEEE Press, Piscataway, NJ, USA, CIG'09, 1–8.

FAIRCLOUGH, C., FAGAN, M., MAC NAMEE, B., AND CUNNINGHAM, P. 2001. Research Directions for AI in Computer Games. In *Irish Conference on Artificial Intelligence and Cognitive Science*, Citeseer, 333–344.

HOULETTE, R. 2003. *Player Modeling for Adaptive Games*. Charles River Media, Dec.

ISLA, D. 2005. Handling Complexity in the Halo 2 AI. In *Game Developers Conference*.

JAIN, R. 1991. *The Art of Computer Systems Performance Analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley.

LAIRD, J. E., AND LENT, M. V. 2001. Human-Level AI's Killer Application. *AI Magazine 22*, 2, 15–26.

LAVIERS, K., SUKTHANKAR, G., MOLINEAUX, M., AND AHA, D. 2009. Improving Offensive Performance Through Opponent Modeling. In *Artificial Intelligence and Interactive Digital Entertainment*, 58–63.

MACHADO, M. C., FANTINI, E. P. C., AND CHAIMOWICZ, L. 2011. Player Modeling: Towards a Common Taxonomy. In *Computer Games (CGAMES), 2011 16th International Conference on*, 50–57.

MACHADO, M. C., ROCHA, B. S. L., AND CHAIMOWICZ, L. 2011. Agents Behavior and Preferences Characterization in Civilization IV. In *X Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*.

MACHADO, M. C., PAPPA, G. L., AND CHAIMOWICZ, L. 2012. A Binary Classification Approach for Automatic Preference Modeling of Virtual Agents in Civilization IV. In *Proceedings of the 8th International Conference on Computational Intelligence and Games*, CIG'12.

MANOVICH, L. 2001. *The Language of New Media*. Massachusetts Institute of Technology, Cambridge, Mass.

NAREYEK, A. 2004. AI in Computer Games. *Queue 1* (February), 58–65.

SMITH, A. M., LEWIS, C., HULLETT, K., SMITH, G., AND SULLIVAN, A. 2011. An Inclusive View of Player Modeling. In *Proceedings of the 6th International Conference on the Foundations of Digital Games*, ACM, New York, NY, USA, FDG '11.

SOUTHEY, F., BOWLING, M., LARSON, B., PICCIONE, C., BURCH, N., BILLINGS, D., AND RAYNER, C. 2005. Bayes bluff: Opponent Modelling in Poker. In *In Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI*, 550–558.

SPRONCK, P., AND DEN TEULING, F. 2010. Player Modeling in Civilization IV. In *Proceedings of the 6th Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 180–185.

SPRONCK, P. 2005. A Model for Reliable Adaptive Game Intelligence. In *Workshop on Reasoning, Representation, and Learning in Computer Games*, IJCAI'05.

TAN, C., TAN, K., AND TAY, A. 2011. Dynamic Game Difficulty Scaling using Adaptive Behavioural Based AI. *IEEE Transactions on Computational Intelligence and AI in Games PP Issue:9*, 99.

TAYLOR, L. N. 2002. *Video games: Perspective, point-of-view, and immersion*. M.S. thesis, Grad. Art School, Univ. Florida, Gainesville, FL.