Automatic Computer Game Balancing: A Reinforcement Learning Approach

Gustavo Andrade, Geber Ramalho, and Hugo Santana
Universidade Federal de Pernambuco
Centro de Informática
Caixa Postal 7851, CEP 50732-970, Recife, Brazil
{gda,glr,hps}@cin.ufpe.br

Vincent Corruble
Université Paris 6
Laboratoire d'Informatique de Paris VI
Bôite 169 - 4 Place Jussieu
75252 PARIS CEDEX 05
Vincent.Corruble@lip6.fr

ABSTRACT
Designing agents whose behavior challenges human players adequately is a key issue in computer games development. This work presents a novel technique, based on reinforcement learning (RL), to automatically control the game level, adapting it to the human player skills in order to guarantee a good game balance. RL has commonly been used in competitive environments, in which the agent must perform as well as possible to beat its opponent. The innovative use of RL proposed here makes use of a challenge function, which estimates the current player's level, as well as changes on the action selection mechanism of the RL framework. The technique is applied to a fighting game, Knock'em, to provide empirical validation of the approach.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]:Distributed Artificial Intelligence – *Intelligent Agents*; I.2.6 [**Artificial Intelligence**]: Learning – *Parameter learning, Reinforcement learning*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems – *Games*.

General Terms
Algorithms, Human Factors.

Keywords
Game Balancing, Adaptive Agents, Reinforcement Learning.

1. INTRODUCTION
Machine learning (ML) techniques have been widely used in competitive domains, with the focus on finding an optimal strategy which maximizes the payoffs for the agent on most scenarios of competition. It means that the agent must perform as well as possible. Computer games can be seen as competitive environments, however, in this case, it is necessary to achieve a balanced behavior. Game balancing is related to ensuring a good level of challenge in a game, which implies avoiding the extremes of getting the player frustrated because the game is too hard or becoming bored because the game is too easy [2]. Balancing is recognized by the game development community as a key characteristic for a successful game [1]. The goal is to keep the game level adapted to the performance of the human player, no matter his or her skill level, which can vary widely from novices to experts. Unfortunately, fixing a few pre-defined and static difficulty levels (e.g., beginner, intermediate and advanced) is not sufficient In fact, maintaining the adequate level is a dynamic process, because of the evolution of the player's behavior, as a natural consequence of the experience acquired in playing the game. On the other hand, as user skills can regress (for instance, after a long period without playing the game), regressions of the level are also needed.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.
Copyright 2005 ACM 1-59593-094-9/05/0007 …$5.00.

2. DYNAMIC GAME BALANCING
Dynamic game balancing is a process which must satisfy at least three basic requirements. First, the game must, as quickly as possible, identify and adapt itself to the human player initial level. Second, the game must track as close and as fast as possible the evolutions and regressions in the player's performance. Third, in adapting itself, the behavior of the game must remain believable, since the user is not meant to perceive that the computer is somehow facilitating things (e.g., by decreasing the agents physical attributes or executing clearly random and inefficient actions).

There are many different approaches to address dynamic game balancing. One can control the game environment settings in order to make challenges easier or harder [4]. Although this approach may be effective, its application is constrained to game genres where such particular environment manipulations are possible. Another approach to dynamic game balancing is to modify the behavior of the Non-Player Characters (NPCs), characters controlled by the computer and usually modeled as intelligent agents. This approach has been innovatively implemented employing genetic algorithms techniques to keep alive agents that best fit the user level [5]. However, it shows some limitations considering skilled users or users with uncommon behavior, as it takes a long time until the agents reaches the user level.

3. OUR APPROACH
We propose a novel use of Q-Learning, a popular Reinforcement Learning (RL) algorithm [3], to address dynamic game balancing. As directly learning to play at the user level can take a long time, we faced game balancing as two separate problems: learning (building agents that can learn optimal strategies) and adapting

(providing action selection mechanisms for providing game balance, possibly using sub-optimal actions). Due to the requirement of being immediately able to play at the human player level, including expert ones, offline training is needed to bootstrap the learning process. Then, online learning is used to adapt continually this initially built-in intelligence to the specific human opponent, in order to discover the optimal strategy to play against him or her.

Standard Q-Learning, when not doing exploration, selects in the action selection mechanism the action whose value is maximal for the current state. In fact, the agent chooses the best action for each situation and keeps learning in order to improve its performance.

In our case, we cannot simply keep the agent acting as best as possible. The agent must choose an action, possible sub-optimal, according to the challenge function [5], a function that maps a game state into a value that specifies how easy the game is perceived by human users. For a given situation, if the game level is too hard, the agent does not choose the optimal action (the one with highest value, as given by the action value function constructed in Q-Learning), but chooses progressively sub-optimal actions until its performance is as good as the player's. This entails choosing the second best action, the third one, and so on, until it reaches the player's level. Similarly, if the game level becomes too easy, it will choose actions whose values are higher, possibly until it reaches the optimal one.

This approach uses the order relation naturally defined in a given state by the action-value function, which is automatically built during the learning process. As these values estimate the individual quality of each possible action, it is possible to have a strong and fast control on the agent behavior and consequently on the game overall level. It is important to notice that this technique changes only the action selection mechanism, while the learning process, including the updates of the action value function, is the same as standard Q-Learning. The agent keeps learning during the entire game.

4. CASE STUDY

As a case study, the approach was applied to Knock'Em, a real-time fighting game where two players face each other inside a bullring. The main objective of the game is to beat the opponent. A fight ends when the life points of one player (initially, 100 points) reach zero, or after 1min30secs of fighting, whatever comes first. The winner is the fighter which has the highest remaining life at the end. The environment is a bidimensional arena in which horizontal moves are free and vertical moves are possible through jumps. Some possible attack actions are to punch, to kick, and to launch fireballs. Some defensive actions are blocking or crouching. The challenge function used is based on the life difference during each fight.

We compare the performance of two agents: a traditional reinforcement learning (playing as best as possible), and the adaptive agent (implementing the proposed approach). The evaluation scenario consists of a series of fights against different opponents, simulating the diversity of human players strategies: a state-machine (SM, static behavior), a random (RD, unforeseeable behavior) and a traditional RL agent (TRL, intelligent and with learning skills). Each agent being evaluated plays 30 fights against each opponent. The performance measurement is based on the final life difference in each fight. Positive values represent that the evaluated agent wins, and negative ones that the agent loses. The results are graphically displayed beyond.

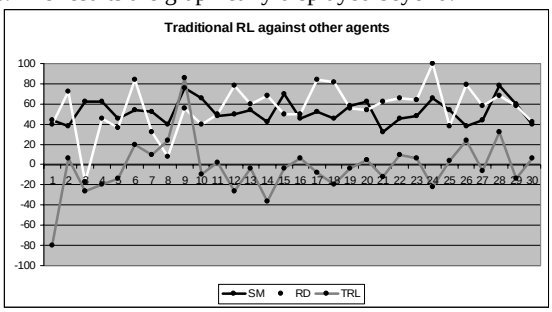

Figure 1: Traditional RL agent performance

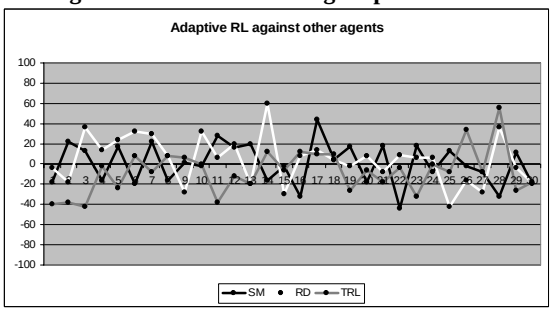

Figure 2: Adaptive RL agent performance

Figure 1 shows the traditional RL agent performance. The positive values of the black and white lines show that the agent beats quite easy the state-machine and the random opponents. The gray line shows that two TRL fighters have a similar performance while fighting against each other. Figure 2 shows the adaptive RL agent performance. Although this agent has the same capabilities as traditional RL, because their learning algorithms and their initial knowledge are the same, the adaptive mechanism forces the agent to act at the same opponent level. As a result, the performance varies between wins (positive points) and defeats (negative points), independently of opponent skills.

5. CONCLUSIONS

These results indicate the effectiveness of our approach, concerning the three requirements of dynamic game balancing. In fact, although the adaptive agent could easily beat their opponents, its performance level is adapted to be close to the opponent level. Moreover, the adaptive agent has shown to fast adapt, acting at the same opponent level already in the initial fights. The approach can be applied to different game styles, requiring only adequate RL representation and challenge functions, which can be based on rate of successful shots or hits, the numbers of won and lost pieces, time to complete some task, or any metric used to calculate a game score.

6. REFERENCES

[1] Falstein, N., The Flow Channel, *Game Developer Magazine*, May Issue, 2004.

[2] Koster, R., *Theory of Fun for Game Design*, Paraglyph Press, Phoenix, 2004.

[3] Sutton, R., Barto A., *Reinforcement Learning: An Introduction*, Cambridge, MA, 1998.