International Journal of Automation and Computing 9(2), April 2012, 122-134
DOI: 10.1007/s11633-012-0625-5

Improving Behavior of Computer Game Bots Using Fictitious Play

Ushma Kesha Patel Purvag Patel Henry Hexmoor Norman Carver
Department of Computer Science, Southern Illinois University, Carbondale IL 62901, USA

Abstract: In modern computer games, "bots" – intelligent realistic agents play a prominent role in the popularity of a game in the market. Typically, bots are modeled using finite-state machine and then programmed via simple conditional statements which are hard-coded in bots logic. Since these bots have become quite predictable to an experienced games′ player, a player might lose interest in the game. We propose the use of a game theoretic based learning rule called fictitious play for improving behavior of these computer game bots which will make them less predictable and hence, more a enjoyable game.

Keywords: Agents, computer game bots, fictitious play, game theory, learning, Nash equilibrium.

1 Introduction

First person shooter (FPS) games, online card games, and massively multiplayer online playing games (MMORPGs) are popular and extremely diverse online game genres. Our research in this paper is based on one of the most vibrantly used FPS games, the Counter-Strike (CS). Encouraged by game Artificial Intelligence (AI), we have incorporated computer game bots for our simulation of the CS game. Computer game bots are non-player characters (NPCs) that simulate human game players and display human-like behavior while playing against humans. We propose a state-of-the-art method towards game design using a game theory based learning algorithm called fictitious play[1]. The bots (non-player characters), in this method, assume that the opponents are playing a fixed strategy and hence based on their past experiences they plan their next move. This research explores how a computer game bot adapts to the dynamic human player′s behavior while playing the simulated game of CS, thereby leading to its unpredictable behavior. Here we focus only on an individual agent at a time when it comes to experimentation. All remaining alternative evaluations, such as team behavior, lie outside the scope of the current research.

It is very common to add a module to computer games called AI that guides behavior of NPCs. This AI module is inspired by the field of Artificial Intelligence. However, it differs from it. Game AI aims to speed up games and encapsulates rules of thumb to NPC and not necessarily employs methods which may be actually classified as AI technique by the researchers in academic AI. Although a crisp definition for research discipline of AI is difficult to state, as researchers have different viewpoint/point of view/context/angles to define the term. Some researchers like[2] explain AI as a study of measuring performance against humans by thinking like humans and acting like humans, and a measurement against ideal performances like thinking rationally and acting rationally; others like define it is an effort towards making the computers think, and some suggest that AI is the automation of activities, such as decision-making, learning, problem solving, etc., that we associate with human thinking. Even though with disagreement over the definition of AI, the goal of the researchers is to design intelligent algorithms for various purposes, and in our case for improving behavior of computer game bots.

Computer games have improved exponentially in graphics and sound capability, more CPU is power available for implementing AI and become competitive in the games market. For the users, it is very important to have the ability of discerning various game character actions and hence improved and enhanced graphical representation of those characters came into existence. Moreover, far advanced game development techniques with high-end graphics are employed by Graphics Processing Unit with higher CPU processing power. High standards in terms of remarkable AI techniques are a must along with excellent graphics, sound capability and powerful CPUs that would account for the success of a game. These marked changes came into existence with the use of AI in games[3], also refered to as computational intelligence in games[4]. Technically, AI in computer game design has been treated differently in three communities: 1) combat oriented AI, 2) non-combat oriented AI, and 3) analytical and efficiency oriented AI. Game AI is combat oriented in FPS games and is in its nascent stages, and lacks the ability to dynamically interact with the physical game environment like terrains. Non-combat oriented AI is a much more complicated implementation form of game AI and is therefore generally not preferred by game programmers. The analytical and efficiency oriented AI is the most rapidly evolving class of games and has been explored but needs to be fully implemented. Our research and experiments fall into this, namely analytical and efficiency oriented AI, category of game AI with the help of game theory based learning algorithm. Our game bots analyze the observations of the opponent′s actions, learn and respond efficiently resulting into a better game performance, and we get significant bots improvements.

Modern day games pose problems for AI with the most common features like real time, dynamicity of game environments, game bots having incomplete knowledge of the world, and restricted game resources for game bots[5]. Fur-

Manuscript received April 17, 2011; revised June 13, 2011

thermore, game AI has challenges to face for different genres of games. In role playing games (RPGs), more realistic and engaging NPCs are needed[6]. FPS games face a challenge with AI agents when these agents have to assume possibilities according to their current strategy for reasoning and selecting their behavior at higher levels of the computer games[7]. Even though AI controlled NPCs are successful in avoiding the adverse game environment actions, such as to shoot like an experienced skilled player, human combatants have four major things to offer over AI: 1) instinct for survival, 2) better knowledge of their respective environments, 3) their hunting ability, and 4) teamwork. A detailed study shows that these expected AI challenges clearly indicate the utmost need and efficient use of AI in computer games that can be achieved by maximizing the goal and selecting the most optimal actions as per the given information in [8, 9].

AI in game bots has a purpose of presenting several expected NPC capabilities that include learning, decision-making, prediction, reasoning, planning, and scheduling[8]. Our research encompasses the capabilities like learning in line with prediction and decision-making capability with the game bots and the results are quite promising. Machine learning in AI provides algorithms like reinforcement learning, supervised learning, and unsupervised learning. Fictitious play is a learning rule that we have incorporated in our research based on which we discover how learning makes game bots more intelligent.

AI aspect of game design is the driving force for computer game developers as it can be used to control characters, give commentary in sports games, or change parameters dynamically to make the game more challenging[10]. They find it difficult and time-consuming to hard-code parameters for encoding bots strategies to tune the NPCs to suit the game environment. In addition, there is a lack of infrastructure for building and maintaining dynamic multi-agent realistic research environments. To overcome these intricate issues, Adobbbati et al.[11] suggested multi-user virtual environments (MUVEs) like AI test-beds, e.g., Phoenix[12] and RoboCup[13], where game bots are tested in complex multi-agent environments. The biggest advantage of these test-beds is that they are invariant to specific computer games and therefore could be used across many. For research in AI and multi-agent systems (MAS), gamebots provide several previously unavailable opportunities, such as: 1) supporting multiple tasks, 2) providing an extension for built-in scripting language, 3) allows creating multiple environments, 4) supporting humans-as-agents, and 5) public availability in USA and overseas[11].

Neumann with Morgenstern published a groundbreaking text giving birth to an interdisciplinary research field of game theory, which until recently has largely been overlooked by the computer scientists[14, 15]. Game theory has become a branch of applied mathematics which is assimilated by various disciplines, such as social sciences, biology, political science, philosophy, and engineering. As with other computer science applications, the game theory approach in game development is yet to be explored to its capacity. Game theory concepts in AI can have an important role to play in the implementation of numerous computer game strategies that can provide a number of mathematical tools to understand the possible opponent strategies. This not only provides the game developers with a useful tool for implementing various complex strategies for the computer games, but can also contribute towards their marketing value. One such learning algorithm in game theory is called fictitious play (FP)[1].

Fictitious play was introduced as a solution to find Nash equilibrium. An agent using fictitious play considers that opponents are playing a fixed strategy and keeps track of the strategy being played by them[1]. Based on these observations, agents execute their future actions. Fictitious play is a simple and efficient algorithm. Fictitious play being a light weight algorithm can be used in internet based virtual environment for three-dimensional games, such as [16]. This paper explores the use of fictitious play in game AI, specifically for computer game bots.

In Section 2, we discuss the background of computer games bots, the game of CS, game theory concepts and fictitious play thus creating the foundation to understand how our research is related to these concepts. In Section 3, we discuss some of the related works by [17] and [11] in the field of game AI and look into their detailed work. In Section 4, we propose a state-of-the-art method as our approach towards game design using a game theory based learning algorithm called fictitious play. Also, we present our experiments and promising results, and finally discuss the future work. In Section 5, we discuss conclusion on our research. To our knowledge, the attempt to infuse game theoretic concepts in designing the computer game bots is a novel frontier.

2 Background

In the computer game bot or game agent research community, there are many other characterizations for autonomous agents. To exemplify, Yildirim and Stene[8], Franklin and Graesser[9] state that game agents are said to be autonomous if they possess their own agenda and select the corresponding necessary actions. Russell and Norvig[2] suggests that an agent is said to be autonomous if its own experience can determine his behavior provided that it has the ability to learn and adapt. In robotics, independence of control is autonomy.

Technically, AI in computer game design has been treated differently in three camps: 1) combat oriented AI, 2) non-combat oriented AI, and 3) analytical and efficiency oriented AI. Game AI is combat oriented in FPS games and is only in the beginning stages. Since its implementation in Half-Life, combat oriented AI emerged as an unseen game giving the players a realistic awareness of the weapons, the environment, and AI comrades. Game designers realized the importance of this AI aspect of game design after Half-Life. It lacks the ability to dynamically interact with the physical game environment like terrains. Non-combat oriented AI is a much more complicated implementation form of game AI and is therefore generally not preferred by game programmers. One of its philosophical implications can be found in RPG with mainly NPC. The analytical and efficiency oriented AI is the newest frontier of games and has to be reached to its full potential. Analytical AI intends to reinforce the game experience; e.g., by controlling number of combat AI opponent agents, keeping records of opponent

agents reaction time and thus analyzing the damage levels. Efficiency AI intends to make an efficient use of all the available CPU resources at a given time. For example, in the game of CS, coloring the critical areas of the map and leaving the rest in grey shades during a particular phase of a round could create a possibility of disguise for the terrorist agent. Our research and experiments fall into this category of game AI with the help of a learning rule in game AI called fictitious play (FP)[1]. Our game bots in FP analyze the observations of the opponent actions, learn and respond efficiently resulting into a better game performance, and we get significant bots improvements.

For the current research, we use the game of CS as a classic example for first-person shooter games and our simulation environment is inspired from it. Following section discusses the game.

2.1 The game of counter-strike

It is required to study the environment for bots interaction and their usage in computer games. Hence, we selected a modification (i.e., mod for the popular game Half-Life), the game of CS, as a case study for our project. This is one of the most popular open source games online. There are thousands of game players that play it simultaneously on the Internet.

This first-person-shooter (FPS) game is team-based[18]. It incorporates two teams, counter-terrorist team (CT) and a terrorist team (T) playing against each other in a variety of exotic and urban environments. The game agents are addressed as CT agents and T agents for the respective teams. Fig. 1 is a snapshot of CS as viewed by a player after a few rounds of the actual game. The game status values shown are two rounds won by the counter-terrorists and two by the terrorists at the top black panel of the window. On the grey panel of the window at the bottom, the name of the agent playing this round is Tim (i.e., a terrorist). The landscape in the figure is a two-dimensional scene with concrete and wooden walls throughout as seen by the player (a T agent in this case). T agent is holding a gun. A continual flashed map is provided on top of this two-dimensional scene for the T agent to check the position of its teammates beyond the walls of the area. The teammates are depicted as small white points on this area map. This map shows counter-terrorist camp (CTC) on the top left corner and the terrorist camp (TC) on the bottom right corner in the figure. These camps are the starting points of both the teams agents to initiate their locomotion. Bomb site A (BCA) shown as =A, and bomb site B (BCB) shown as =B are the target bomb sites either for the terrorists to plant the bomb or for the counter-terrorists to diffuse it. The area map has faint white lines drawn between the TC and BCA which suggests that the T agents have already played a few rounds and have visited BCA by adopting different paths. This game in general features three game modes[19]:

1) Diffusion where T players plant a bomb, and CT players diffuse it,

2) hostage rescue where CT players attempt to rescue hostages, and

3) assassination where T players attempt to kill a VIP that the CT players must protect.

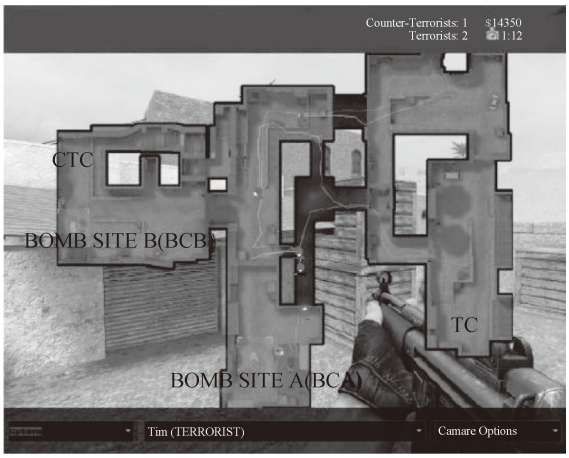

Fig. 1 A snapshot of Counter-Strike

Here, we are more concerned with the first feature of the game, i.e., diffusion. Elaborating this, in the diffusion mode, the T agent plants the bomb, while the CT diffuses it. However, usually the round ends beforehand because one of the teams would win the round by eliminating the opponents. The team winning more number of rounds is the winner.

This scenario can be played using different preloaded maps available in the game. Usually, on each map there are two sites labeled A, and B which are called "bomb sites" where a T agent plants the bomb. CTs make efforts to defend these "bomb sites", and if the bomb gets planted by T, CTs try to diffuse the bomb before it explodes. In the beginning of each round, both the teams are located at designated locations on the map. For example, the location where CTs start is called "counter terrorist camp" (CTC) and similarly, Ts′ camp is called "terrorist camp" (TC). Once the round starts, they start moving around the map, fighting with each other and trying to achieve their respective goals. Each map in the game has many paths that CTs and Ts may use to go from their base camps to the targeted places and usually they use different paths for each round. On these paths, there are certain positions that are critical which act as battle grounds. We will call these positions "critical positions". Both the teams need to pass through these "critical positions" to reach the bomb sites and hence, both the teams possessing the knowledge of these positions would plan to exploit them.

There is also a facility for playing this game offline. In that case non-human players are needed to replace human players. Following section provides brief overview of such artificial game characters.

2.2 Bots in computer games

Bots in counter-strike, also called NPCs, are used to replace human players. Bots play as a part of the team and achieve goals similar to humans. Bots simulate human players and are aimed to give game players the "illusion" of playing against actual human player similar to the illusion that the computer is playing the role of a human in the turing test. Currently, bots used in counter-strike are programmed to find the path, attack opponent players, or run away from the site if they have heavy retaliation or if their

energy is less, providing an illusion that they are intelligent. Similar species of bots are also used in many other FPS games, with similar method of programming. Bots are usually pre-programmed according to the requirements of a game and play for or against human players. Bots usually display two kinds of behaviors, namely static behavior, and dynamic behavior. Static behavior of a bot reflects that it has been hard-coded for a predefined set of outcomes. And, dynamic behavior is a result of handling bots with the help of AI learning techniques such as applying game-theoretic approaches.

Human players while playing against or with computer players, which are used to replace humans, have a few expectations which should be incorporated in the design of game agents. Bob Scott listed out these expectations which may act as guidelines for game developers[20]:

1) Predictability and unpredictability: Human actions are unpredictable, and at the same time they can be very predictable. These kinds of actions tend to provide surprise elements to the game. For example, in FPS this means that throwing granite for the purpose of distraction is unpredictable. An example of predictable behavior is using the same route repeatedly. Also, human actions are very inconsistent, and sometimes they tend to behave stupidly. Mimicking this form of behavior in a game can be difficult.

2) Support: Often, a computer player has to provide support to fellow human players. This can be as simple as guiding human players through the game, or providing cover during combat. An efficient method of communication is necessary between computers and human players which is hard to achieve.

3) Winning, losing and losing well: As we said earlier that it is relatively easy to engineer a computer player that always wins or loses against a human player. The main focus herein is believability of the opponents. Difficulty settings provided in almost every game allow game players to set the level of computer player whether it will be challenging or not. Also, there are few games where the computer players change their difficulty levels based on number of wins and loses.

4) Cheating: Whether a game AI should be allowed to cheat or not has always been debatable. Objections are that AI will have unfair advantage over humans, but it also true that computers are already at a disadvantage because they are impassive. From our point of view, cheating is acceptable as long as it does not get detected by a human player keeping computer player interesting and challenging.

Nareyek[5] argues that modern day games pose problems for AI with the most common features like there is a very limited time given for reasoning, dynamicity of computer game environments is very high and competitive, many game bots have incomplete knowledge of the world, and game character or environment resources might be restricted[5]. Furthermore, game AI has challenges to face for different genres of games. In RPGs, there is a need for more realistic and engaging NPCs. The complexity level in reaching a common goal can account for determining what type of intelligence is required for the agents to collaborate or cooperate, e.g., what kind of collaboration strategies they follow, what kind of resources the team or team members possess, and so on[6]. A challenge that FPS games face with AI agents is in controlling the game character by the player. These games employ a layered structure of the artificial intelligence system which consists of lower level and higher level layers. Functions like creating maps for the terrain, generating sequential character animation, and generating a list of weapons are handled by the lower level layers. Responsibility for the higher level layers could include agents assuming possibilities according to their current strategy for reasoning and selecting their behavior for example, whether the agent should run through the generated map in search of the opponent, or should it indulge into combat, etc.[7]

Usually, bots in computer games are modeled using a FSM, as shown in Fig. 2, where rectangle represents a possible state whereas leading edges show transition between states. It is just a miniature representation of an actual bot where many more such states exist with more complicated transitions. FSM for bots is quite self-explanatory where the bot begins by making initial decisions like game strategies, buying weapon(s), and then start finding the enemies. Once an enemy is found, it makes a transition to attack state in which it fires bullets at the enemy. A bot may kill an enemy; therefore, in that case it will again start searching for the enemy as shown in Fig. 2. Also, a bot could be in any of the above mentioned states and might get killed by the enemy.

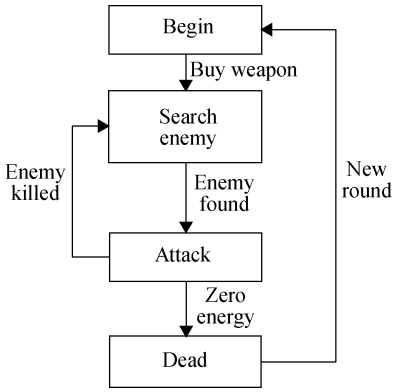

Fig. 2 A prototypical FSM for bots

Bartish and Thevathayan[21] suggest that using state machines in game development results into its design complexity and code complexity as compared to belief-desire-intention (BDI) agents. BDI agents are rational agents, as [22] defines them, having mental attitudes of belief, desire and intention that determine the agent-oriented systems behavior. They used interplay as their test-bed for the atomic bomberman game and found that state machines had higher complexity as compared to agents in their experiments. They explored that on adding more behaviors to the original game, code complexity increases such that the rate of change in complexity was on a significant (quadratic) increase for the state machines, and linear increase for the BDI.

FSM for bots are implemented using simple if-else or switch-case statements usually using C/C++ programming languages. Problem with this style of programming is that their behavior becomes very predictable to even novice play-

ers. Players will be able to predict their action and what path they will use. The reason for this is that a game player spends hours playing the same game and at same level and therefore reaches a threshold at which the player can predict the bots behavior because they repeatedly perform similar behaviors due to their hard-coded logic. This makes the game less interesting to an experienced game player and eventually he may lose interest in the game.

There have been efforts to solve this problem using reinforcement learning algorithm such as Q-learning[23]. Reinforcement learning (RL) is a machine learning technique in which agents learn to solve problems by taking actions based on their interactions with the environment so as to achieve maximum rewards (i.e., payoffs in game theory). This learning approach is significant to the community of computer game developers[24]. Many game applications such as [25], and more have provided us with successful real-time game strategies and algorithms in which bots evolved are intelligent and more realistic. Problem on using reinforcement learning or other machine learning techniques is that they are computationally expensive, difficult to implement and can lead to very random outcomes.

We address the learning problem with bots differently with a simple, computationally efficient and easy to implement algorithm – fictitious play, which is described in Section 4. Following section (Section 2.3) provides overview of game theory that leads to the explanation and working algorithm of fictitious play.

2.3 Game theory

Game theory aims to model the situations in which participants interact or affect each other′s outcome[26]. Each individual is intelligent and capable of making rational decisions. Each participant has an individual objective in a shared system, wherein his decisions influence another′s welfare. This leads to analyses of competition among these individuals that is termed as "games". Therefore, the science that studies the mathematical models of conflict and cooperation between the intelligent rational decision-makers can be defined as "game theory"[27].

2.3.1 Basic concepts

A "Game" = N players, is a set of strategies for each player, where $\{0|0$ is the outcome of each "play" of the game$\}$, and $0 \rightarrow \{P|P$ is a payoff for each player$\}$. A game-matrix, also known as payoff-matrix is represented in Fig. 3.

In Fig. 3, player I and player II are the two players with their payoff values as a_{ij} (typically for Player I) where $i = 1, 2, \cdots, m$, and $j = 1, 2, \cdots, n$, and S_i, S_j = Set of strategies that player II and player I play, respectively.

A "zero-sum game" is one in which the summation of all the players′ payoffs is zero, in other words, a player wins if and only if the opponent loses. For example, the game of Matching Pennies shown in Fig. 4 is a zero-sum game. Here in Fig. 4, H = heads, and T = tails. Both the players have to write down either H or T on a piece of paper. If both of them turn out to be the same, then the second player has to pay 1 (unit is dollar in this particular case) as a payoff to the first one, and if the pennies written on the piece of paper do not match, then the first one ends up paying the same amount to the second player[28].

The representation of games can be done primarily in two forms: first, the normal form (matrix form as in Figs. 3, 4, and 6), and second is the extensive form. For example, the extensive form for the game of Matching Pennies (Fig. 4) is shown in Fig. 5. In Fig. 5, player I will play the first move of the game (i.e., either he will play H heads or T tails) and then player II will play the second move.

		Player II			
		S_1	S_2	$\cdots$	S_n
Player I	S_1				
	$\cdots$		a_{ij}		
	S_n				

Fig. 3 Payoff matrix

		Player II	
		H	T
Player I	H	+1, –1	–1, +1
	T	–1, +1	–1, +1

Fig. 4 Game of Matching Pennies

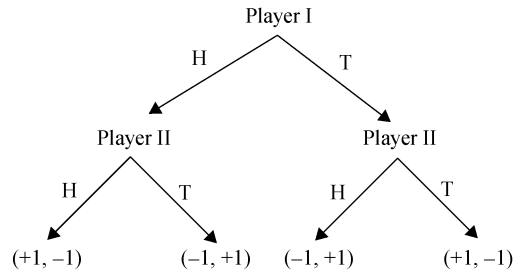

Fig. 5 Matching Pennies in extensive form

2.3.2 Prisoner′s dilemma

Game theory is best demonstrated with the classical example called "the prisoners dilemma". Suppose that police arrested Rob and Bob from the crime site and placed both of them in an isolated cell. In lack of enough evidence police gave a fair deal to each of them. "You may choose to confess (defect) or remain silent. If you confess and other remains silent then you will be set free and other will get 10 years of prison. If you both remain silent then in lack of evidence you both will get 6 months of prison and then you both will be set free. But, if both of you confess then both of you will get 5 years of prison. Finally, we will make sure that both of you will not have any clue what choice the other one has made"

Dilemma faced by both Rob and Bob is whether to confess or remain silent in order to minimize their own time in jail without the concern of other player. This scenario can be summarized with payoff matrix in Fig. 6.

In Fig. 6, the "dilemma" faced by prisoners is that both are better off when they confess, but in that case outcome is worse than what they would have received if both of them have remained silent. This puzzle reflects the conflict between individual and group rationality[29]. There are various forms of this game which are being extensively studied

by the game theorists. This is because this form of situation arises in everyday interactions of human beings, and hence, is studied extensively in economics, political science, and sociology. For example, the two prisoners can be replaced by two computer users who are competing for network usage. A good strategy for both users would be to back off for few minutes; in that case delay will be 1 min. If none backup, then delay will go up to 5 min. If one of them backs up and other does not, then the one which backs up might experience delay of 10 min while other might not have any delay. Furthermore, the nature of prisoner dilemma and its family as 2×2 games has aptly been discussed by Robinson and Goforth[30].

A/B	B remains silent	B defects
A remains silent	A & B: 6 months jail	A: 10 years jail B: set free
A defects	A: set free B: 10 years jail	A & B: 5 years jail

Fig. 6 Prisoner′s dilemma

Game theory is not only used in developing computer games, but also resolving disputes. One such attempt is an online integration support environment which is described with the help of argumentation techniques of Lodder combined with the artificial/game theory approach of Belluci and Zeleznikow that helps in making decision as to whether or not the court case should be settled outside the court or not[31]. This would lessen the burden of innumerous court cases yet pending.

As a computer scientist we want to solve problems we face in computer science using game theory while modeling them as agents and multi-agents. We will use "non-cooperative" game theory, a branch of game theory, for this research.

2.3.3 Mixed strategies

Mixed strategy has been formally defined by various game theorists, such as [28]. Moulin[32] considers the game of the battle of the sexes shown in Fig. 7.

H/W	Football	Opera
Football	(3, 0)	(0, 1)
Opera	(0, 3)	(1, 0)

(a)

H/W	Football	Opera
Football	(3, 0)	(0, 1)
Opera	(0, 3)	(1, 0)

(b)

Fig. 7 Battle of sexes

Husband (H) and wife (W) have to decide as to whether they want to go to the football or to the opera, but they are unable to decide. The husband prefers to go the football together with his wife and the wife prefers to go to the opera together with her husband. Now, let us suppose that the husband is unaware of his wife′s preferences, then in this case, there can be two outcomes: either the wife prefers to be with her husband (i.e., Fig. 7 (a)), or she prefers going alone to either of the two events (i.e., Fig. 7 (b)). This is when we can define what a mixed strategy is. We can conclude that there are two choices to be made: either go to the football, or go to the opera. These are called pure strategies. There exists another choice of strategies, such as tossing a coin and then deciding what should be done if tail comes (like going to the football), and what should be done if head is up (like going to the opera). The third choice of tossing a coin is known as a mixed strategy where it is equally likely that they might decide to go for the football or to the opera together[28]. Although this mixed strategy is not stable and it is difficult to reach equilibrium. Fictitious play is one of the methods of finding mixed strategy equilibrium as discussed in the "fictitious play" section.

2.3.4 Fictitious play

John Nash proposed the concept of Nash Equilibrium in 1950. Nash equilibrium in game theory is a solution concept that follows a strategy wherein a player gives a best response to another player′s strategy if there is no other strategy that could be played that would yield a higher payoff in any situation in which the other player′s strategy is played. In other words, the Nash equilibrium is the best possible choice for a player to respond to opponent current best possible action in the game as the player has no other choice that changes the opponent payoff as a result[33].

Brown[1] first introduced fictitious play as an explanation for Nash equilibrium play. He imagined that a player would imulate play of the game in his mind and update his future play based on this simulation; hence the name "fictitious play"[1]. Monderer and Sela[34] explains fictitious play in very simple terms that a belief-based learning process is a game where a player selects the best response according to his beliefs for the opponent and these beliefs get updated based on the past observations. This helps in determining the future behavior of the player. He says that a belief-based learning process is like fictitious play (FP) process in which each player has a belief that the opponent player plays a stationary mixed strategy based on its past behavior. If fictitious play converges to any fixed distribution of probabilities then these probabilities are the Nash equilibrium for that particular game.

Formally, for the two agents (i and j) case, we say that i maintains a weight function, $K_i : S_i \rightarrow \mathbf{R}^+$. The weight function changes over time as the agent learns. The weight function at time t is represented by K_i^t which keeps a count of how many times each strategy has been played. When at time $t-1$ opponent j plays strategy S_i^{t-1}, then i updates its weight function with

$$k_i^t(s_j) = k_i^{t-1}(s_j) + \begin{cases} 1, & \text{if } s_j^{t-1} = s_j \\ 0, & \text{if } s_j^{t-1} \neq s_j. \end{cases} \quad (1)$$

Using this weight function, agent i can now assign a probability to j playing any of its $s_j \in S_j$ strategies with

$$Pr_i^t[s_j] = \frac{k_i^t(s_j)}{\sum\limits_{s_j \in S_j} k_i^t(s_j)}. \quad (2)$$

Player i then determines the strategy that will give it the highest expected utility given that j will play each of its strategy $s_j \in S_j$ with probability $Pr_i^t[s_j]$. That is, player i determines its best response to a probability distribution over player j′s possible strategies. This amounts to player i assuming that player j′s strategy each time is taken from some fixed but unknown probability distribution[35].

We demonstrate the working of fictitious play in Fig. 8. Consider that the game of CS is being played by two types of players: 1) terrorists represented as green players, and 2)

counter-terrorists represented as blue players. Green players are using fictitious play algorithm ((1) and (2)) for predicting blue players′ moves. Fig. 8 shows the mental state of the green agent which keeps track of blue players (A and B). Blue players have two strategy options, going to BCA or BCB. BCA and BCB are the bomb sites as described in Section 2.1. The green player will observe the blue players′ strategy in each step and update its beliefs. Initially, green player is indifferent about blue players strategy, therefore K-Table consists of all zeros (Fig. 8 (a)). In first step of the game, player A chooses strategy BCA and player B chooses BCB. The green player using (1) updates its K-Table and correspondingly can calculate Pr-Table using (2); results are shown in Fig. 8 (b). After the first step, the green player believes that the chances of player A executing strategy BCA is 1, while that of player B is zero. Similarly, green agent computes the K-Table and updates corresponding Pr-Table for next two steps, as shown in Fig. 8 (c) and (d). At the end, green player observes that player A always go to BCA while player B is playing mixed strategy with higher chances of going to BCA. The green player can use these observations for making future decisions. Suppose the goal of the green player is to kill as many opponents as possible, then it should prefer to go to BCA location where it can find more enemies, but if the goal is to plant the bomb, then it would prefer to go to BCB location with less number of enemies. Hereby, we demonstrated the working of fictitious play and its practical application. Note that the algorithm is computationally efficient. Only dataset we need to maintain in memory is the K-table, and the Pr-table can be calculated on the fly while making decisions. Detailed explanation about the approach and results are presented in Section 4.

K-Table		
Blue player	BCA	BCB
A	0	0
B	0	0

Pr-Table		
Blue player	BCA	BCB
A	0	0
B	0	0

(a) Initial

K-Table		
Blue player	BCA	BCB
A	1	0
B	0	1

Pr-Table		
Blue player	BCA	BCB
A	1	0
B	0	1

(b) After: A → BCA, B → BCB

K-Table		
Blue player	BCA	BCB
A	2	0
B	1	1

Pr-Table		
Blue player	BCA	BCB
A	1	0
B	0.5	0.5

(c) After: A → BCA, B → BCA

K-Table		
Blue player	BCA	BCB
A	3	0
B	2	1

Pr-Table		
Blue player	BCA	BCB
A	1	0
B	0.66	0.33

(d) After: A → BCA, B → BCA

Fig. 8 Example of fictitious play

3 Related work

For computer games like Counter-Strike (CS), Half-Life, and Quake which use bots, we find ourselves getting bored by repeated playing of the same game because of the predictable behavior of the bots used in these games. In this section, we are illustrating related application aspects of computer game bots, and different approaches to game problems.

Boumgarten et al.[36] implemented a bot that outperforms the existing automated players of a multiplayer real-time strategy game. The bots evolved to perform game actions including synchronizing attacks, assigning targets with intelligence, and usage of influence maps, along with using large-scale fleet movements. These bots examine previously played games in order to find similar games. It does not matter whether these previous games ended in success or failure. It then builds a decision tree based on the factors that differentiate good and bad games. A fitness-proportionate traversal is applied to this decision tree to plan for the current game by finding a branch of the tree which acts as a partial plan, and then the missing branches are filled up randomly[36].

Cole et al.[37] argues that in game AI, programmers spend a lot of time hard-coding the parameters of FPS robot controllers (bots) logic and consequently, a considerable amount of time is consumed in the computation of these hard-coded bot parameter values. Therefore, to save the computation and programmer time, he presented an efficient method of using a genetic algorithm that evolves sets of parameter values for the task of tuning up these parameters, and demonstrated that these methods resulted in bots that were highly competitive when playing against bots tuned by humans with expert knowledge about the game. His team selected the combination of parameters to tune, allowed them to tune while running the genetic algorithm, evolved bots against each other while playing, and finally, interfaced the best bots against the ones tuned by an expert player to acquire the performance results[37].

Gamebots is a multi-agent system infrastructure derived from an Internet-based video game which is used as a test-bed for different multi-player games. The Gamebots domain provides several previously unavailable unique opportunities for multi-agent research. It supports multiple multi-agent competitive tasks. Also, it provides an extension for built-in scripting language so that agents can be faced with ever-changing multiple tasks to support long-term research in continuation. Additionally, it allows creating multiple environments to appeal to a wider user community. Moreover, it helps researchers to study human problem solving and investigating human-AI involving scenarios by supporting humans-as-agents. And it public availability in USA and overseas makes it convenient for game players to access it across the globe[11].

An approach similar to ours was proposed with the use of BDI agents by [38]. They suggested a model of bots displaying human-like behavior using BDI agents by designing two algorithms that are made to run simultaneously. The first algorithm allows the BDI agent to perform the task of observing the environment, and the second one is responsible for examination of the reasons towards the game playing

strategies. To specify the bots′ beliefs, desires, and intentions while playing the game, this model took inputs from the actual game players. Also, they claimed that the agent which would replace the bot would possess the four basic properties of an agent: 1) autonomy: as no authority can control the agent, its decision making on the basis of its beliefs would be autonomous, 2) proactiveness: goals like winning the round are very "definite" for an agent which makes it proactive, 3) reactiveness: an immediate change in plan as the need arises shows how fast an agent can react to situations with time-crunch, and 4) social ability: audio or video communication with teammates is required[38].

Broome[17], Cole et al.[37], and Zanetti and Rhalili[39] independently experimented with different algorithmic approaches including genetic algorithms and neural networks to evolve bots and let them evolve against the original ones to get performance results with many computer games like Half-Life, Quake-3, etc. Although a number of attempts have been made to use machine learning techniques for bots, yet all of them employ offline learning.

Wooldridge and Jennings[40] discuss some important theoretical and practical issues associated with intelligent agent design and construction. These issues are arbitrarily divided into three categories, namely agent theory, agent architectures, and agent languages. Through this paper, the needs of academia and industry have been equipped with an insight of what an agent is and in what ways could the agent be represented to study its properties. They discuss the methods as to how can agents autonomously plan and practice their actions while coordinating and cooperating with each other in an environment where they negotiate with dynamic and unpredictable situations. This review provides us with a richer understanding of the questions that are put forth in the following paragraph[40].

Wooldridge[41] discusses two key problems in detail where he finds the following as challenges of game theory and the theory of multi-agent systems through his vision – for a particular game.

A similar effort was made by Temenholtz[42] who proposes fundamental problem solutions for the agent to select its action in a particular game with the help of competitive safety analysis, and agents adopting desired behaviors for that game with the help of theories of mediators which were documented in this proposal[42].

In addition to this, we know that fictitious play generally converges to a pure Nash equilibrium for some types of the games when played repeatedly, but it is not always necessary that we culminate with one[43]. Therefore, Gerding et al.[44] made certain analyses of equilibrium strategies for the intelligent bidding agents which when participated in multiple, simultaneous second-price auctions empirically showed that using best-response fictitious play, these strategies did not converge to pure Nash equilibrium. This result drove their attention towards mixed Nash equilibria, and by applying a learning approach called smooth fictitious play, they were able to approximate the equilibrium numerically. Furthermore, when expected utility equations were combined with this learning algorithm, they were successful in computing mixed strategies without any auction-simulations. The final results that they derived showed the ϵ-Nash mixed strategies convergence with their strategies[44].

Reinforcement learning (RL) is a machine learning technique where an agent learns to solve problems while interacting with the environment[24]. McPartland and Gallagher[25] suggested a learning algorithm to investigate the extent to which RL could be used to learn basic FPS bots behaviors. The team discovered that using RL over rule-based systems rendered a number of advantages, such as: 1) game programmers used minimal code for this particular algorithm, and 2) there was a significant decrease in the time spent for tuning up the parameters. Also, the applied algorithm was used to successfully learn the bots behaviors of navigation and combat, and the results showed that by changing its planning sets of parameters, different bots personality types could be produced. Thus, the paper suggested how an agent can learn to be a bot with the help of RL in shooter games.

Patel[23] selected Q-learning, a reinforcement learning technique, to evolve dynamic intelligent bots, as it is a simple, efficient, and online learning algorithm. Machine learning techniques, such as reinforcement learning, are known to be intractable if they use a detailed model of the world, and also require tuning of various parameters to give satisfactory performance. Therefore, they opt to examine Q-learning for evolving a few basic behaviors viz. learning to fight, and planting the bomb for computer game bots. Furthermore, they experimented on how bots would use knowledge learned from abstract models to evolve their behavior in more detailed model of the world. Their results demonstrate the feasibility of learning techniques for controlling the behavior of bots and at the same time making them unpredictable to game players.

In the following section, we present our approach towards improving the behavior of game bots.

4 Approach

To avoid use of hard-coded bots with static behavior, one needs to use a learning technique, which can adapt bots′ behavior to the play of human players. We propose using the game theoretic based learning algorithm called fictitious play for developing sophisticated bots. For testing fictitious play on bots we needed a simulation environment. We developed a scaled-down abstraction of Counter-Strike in Java and have simulated bots in this very environment. The following section outlines the experimental simulation of the CS game and corresponding results.

4.1 Simulation

The miniature version of Counter-Strike, as shown in Fig. 9, is our simulation for the game. This simulation is extended from [23] to use fictitious play. Here we have attempted to build an environment that simulates a classic FPS game where two sets of autonomous agents are involved in a combat.

In Fig. 10, the boundary of the map restricts the agents′ movement to the confined space. We have different color agents: blue agents and green agents representing the CT

and T agents respectively, and they navigate through the brick walls running through the boundary of the map, and the white paths in between. These agents imitate the behavior of counter-terrorist and terrorist in the CS game. Also, there are two bomb sites: BCA and BCB at the diagonally opposite corners of the map which represent the bomb sites A and B from the CS game in Fig. 1. In addition to this, TC and CTC are the terrorist camp and counter-terrorist camp, respectively, for the green and blue agents. From this point, we called terrorist as green agents and counter-terrorist as blue agents. There are three alternative scenarios in which blue agent and green agents can play (see Section 2.1). We choose them to emulate diffusion scenario for the agents. Before the game starts, we will decide the number of blue agents and green agents. The goal of green agents would be to plant a bomb at either one of the two bomb sites or to kill all the blue agents. And the blue agent′s goals would be to diffuse the bomb if planted and to kill all the green agents. During the course of achieving their respective goal both the sets of agents will fire missiles at each other, if they encounter.

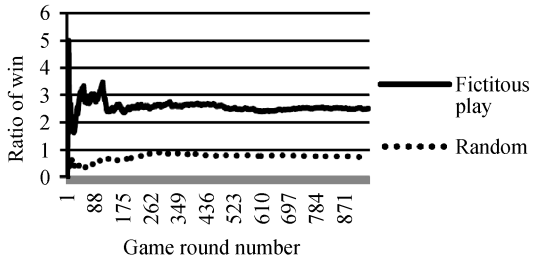

Fig. 9 Initial result

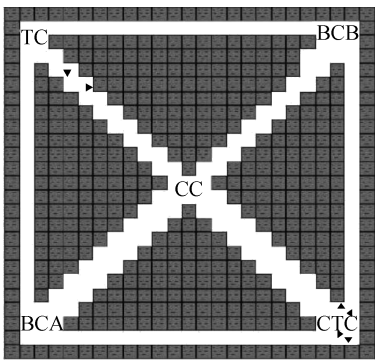

Fig. 10 Simulated map of Counter-Strike

Results of the game can be determined by following cases:

1) Green team wins if all the blue agents are dead, or bomb is planted.

2) Blue team wins if all the green agents are dead, or bomb is not planted.

4.2 Experimentation

In an actual game of CS, an experienced player while playing against bots will eventually get bored after playing the game for long periods as the player can easily predict the next move of the opponent agents which accounts for a static playing strategy. Hence, we propose using the game theoretic based approach called fictitious play wherein the agents dynamically change their strategy of playing based on its opponents′ past observations thus giving the human player a feeling of an active opponent similar to the player. So, for this, we have prepared a simulation, described in the Section 4.1, of the game of CS in which there are two teams, blue and green.

We experimented to improve the behavior of green agents while controlling blue agents′ behavior manually. Blue agents have fixed strategic probabilities of visiting the bomb sites, for instance, 80 % of the times go to BCA and 20 % of the times go to BCB. Green agents having no knowledge of these fixed strategies of blue agents use fictitious play to learn their opponents′, i.e., blue agents′ behavior. They use the following (3) as fictitious play algorithm (also described in Section 2.4):

$$k^t_{\text{green}}(s_{\text{blue}}) = k^{t-1}_{\text{blue}}(s_{\text{blue}}) + \begin{cases} 1, & \text{if } s^{t-1}_{\text{blue}} = s_{\text{blue}} \\ 0, & \text{if } s^{t-1}_{\text{blue}} \neq s_{\text{blue}}. \end{cases} \quad (3)$$

Using this weight function from (3), green agents assign a probability to blue agents playing any of its $s_{\text{blue}} \in S_{\text{blue}}$ where $S_{\text{blue}} = \{\text{BCA}, \text{BCB}\}$ strategies with probabilities shown in (4).

$$Pr^t_{\text{green}}[s_{\text{blue}}] = \frac{k^t_{\text{green}}(s_{\text{blue}})}{\sum\limits_{s_{\text{blue}} \in S_{\text{blue}}} k^t_{\text{green}}(s_{\text{blue}})}. \quad (4)$$

Based on these learned probabilities $Pr^t_{\text{green}}[s_{\text{blue}}]$, green agent decides his own strategies, i.e., whether to go to BCA or to BCB.

4.3 Results

In our simulation, we ran experiments with five green agents (i.e., terrorists) and five blue agents (i.e., counter terrorists).

For the initial experiment the green agents stored the opponents′ strategic probabilities and took the intuitive decision of going to the bomb sites with lower probabilities. This is similar to what human players would have done, i.e., going to the bomb site with fewer number of enemies, and therefore we call this the "intuitive" strategy. Purpose of this experiment was to test the effectiveness of fictitious play for bots, which to our knowledge is not done before. Hence, our initial experiments test the performance of green agents using fictitious paly against the same green agents selecting strategy randomly.

Fig. 10 is a graph plotted with game round number against the ratio of win; i.e., ratio of number of rounds won by the green agents to the number of rounds won by the blue agents. Simulation was run for approximately 1000 rounds. A round is a game played to completion with outcomes of a bomb planted, defused, or agents sacrificed. The graphs shows 1) the green agents and the blue agents play at random (i.e., not mindful of opponents′ strategies), and 2) the green agents are playing fictitious play against blue agents playing randomly. The figure imply that when green agents play at random, they win fewer rounds than when they follow fictitious play. It is also evident from this graph that more number of games is won on the average, and there

is a significant improvement when the green agents play fictitious play against randomly playing blue agents. When fictitious play is played by green agents, the graph in Fig. 10 shows the highest recorded ratio value of 5 at round number 6, and second highest recorded ratio value of 3.48 to be precise for the 112th round of the game. After approximately 560 rounds, the result of the game becomes consistent. Table 1 shows values of the ratio and game round numbers that are shown in Fig. 11. After initial fluctuations, value of ratio of the rounds won remains approximately 2.5, suggesting that there are no further significant improvements.

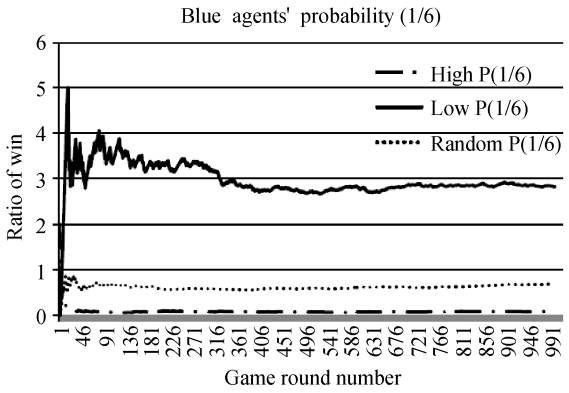

Fig. 11 Playing against 1/6 probability of blue agents

Furthermore, we conducted nine more experiments in groups of three. For each group of experiments, the ratio of blue agents′ visitation probability to the bomb sites is changed. It is 1/6, 2/6, and 3/6 respectively for each of the three groups. That is, for the first three sets of experiments, blue agents will directly go to BCA with 1/6th probability, and for rest of the time it will go to BCB. During each of these experiments, green agents are using fictitious play for deciding which bomb site to visit. Furthermore, we also tested whether the intuitive decision made by the agents was actually the best decision. In order to test that, we ran three simulations in each set by altering the intuitive decision, hence total of nine. Instead of going to the bomb site with lower probability, now the agents will go to bomb site with higher probability. And finally, we compared these results against the blue agent′s decision of going to each bomb sites with equal probability.

4.3.1 Playing against 1/6 probability of blue agents

In this group of experiments, blue agents′ visitation probability was set to 1/6, i.e., they first navigate to BCA 1/6 times and to BCB 5/6 times. We conducted three experiments by programming green agents, such that 1) select bomb site with high value of $Pr^t_{\text{green}}[s_{\text{blue}}]$, 2) select bomb site with low value of $Pr^t_{\text{green}}[s_{\text{blue}}]$, and 3) select both bomb site with equal probability. Hereby, we conducted three experiments altering the decision of green agent.

Fig. 11 shows the results of the experiment. Agents show highest performance while going to bomb site where the probability, $Pr^t_{\text{green}}[s_{\text{blue}}]$, of blue agents was lower. These results support our initial assumption that agents are better off going to bomb site with lower probabilities values. It is also evident from the results that agent is performing poorly while going to bomb site with higher probability value. This is because green agents will face high resistance from blue agents on the bomb site with high probability value. These results signify that the agents are learning to make decisions which humans consider as a matter of common sense.

4.3.2 Playing against 2/6 probability of blue agents

Similar to the experiment in Section 4.3.1, we set the blue agents visiting probability to 2/6, i.e., 1/3 for this set of experiments. Again we experiment with three decisions for green agents, as listed in Section 4.3.1.

Results shown in Fig. 12 are similar to previous results where agent is performing better while going to bomb site with lower probability and worst while going to bomb site with higher probability. There is significant drop in average performance of the agents in all three cases due to decrease in randomness of blue agents. In previous experiment blue agents′ visitation probability was 1/6 compared to 2/6 in current experiment. Agents gave better performance in the former experiment because they got more opportunity to plant the bomb; 1/6 time better chances, and now they might be getting involved in combat more often than in former case. Note that we are still getting better results by playing fictitious play then playing randomly.

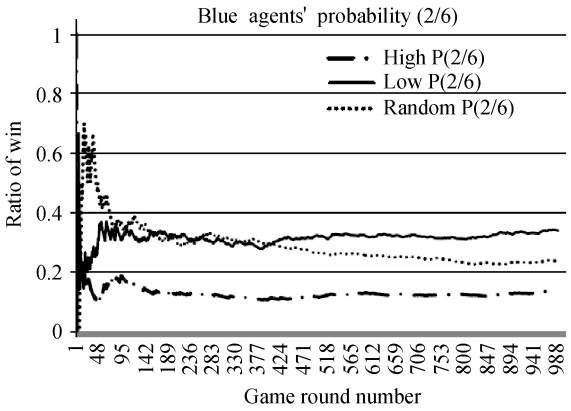

Fig. 12 Playing against 2/6 probability of blue agents

4.3.3 Playing against 3/6 probability of blue agents

For the last set of experiments, we set the blue agents visitation probability to 3/6 (=1/2) which is equivalent to say that they are playing randomly. Results for three different decisions by green agents are shown in Fig. 13. As seen in previous experiments, agents visiting the locations with lower probability show better performance. Average performance of agents was less than previous two experiments because blue agents are visiting both the location with equal probability, and hence lesser number of chances for green agents to plant the bomb. A difference in the pattern of bots′ evolution is seen in this experiment, wherein the performance of FP playing agents is close to random agents, since the agents observing opponent′s random probability will also learn to behave randomly.

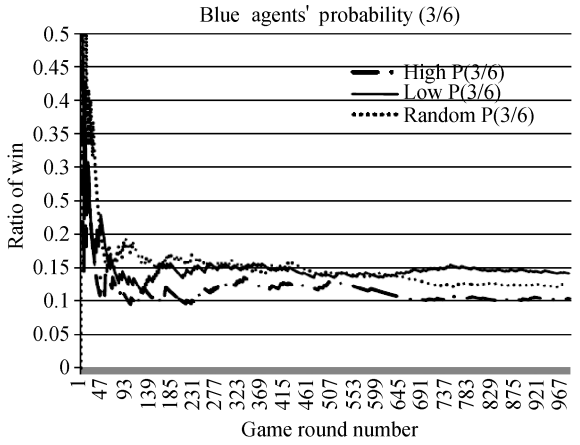

Fig. 13 Playing against 3/6 probability of blue agents

Nevertheless, all the experiments show that irrespective of opponents′ strategy, there is a dominating strategy for agents playing FP. One of the goals of using FP for evolving green agents was to make the agents play more unpredictably which is not directly visible from the results. Although, if one considers the blue agents are three different players having three different strategies, then it can be inferred that agents using FP will exhibit different behavioral patterns. Even for a single player, if it changes strategy for subsequent games, fictitious play algorithm will compute probabilities accordingly and hence, different behaviors for different strategies will be highlighted. Therefore, we can conclude from these results that our initial problem definition that over the time game players gets bored is addressed. Every time a player would play a game, our agents playing fictitious play will evolve differently.

There are many possible extensions. In the following section, we outline one possible approach. However, this extension lies outside the scope our current experimental framework.

4.4 Future work

$Pr^t_{\text{green}}[s_{\text{blue}}]$ is still the probability of only one agent at a time. But, bots will be facing groups of agents and their decision should be based on group, not just one individual agent. Hence, to predict the group behavior, we will take an average of $Pr^t_{\text{green}}[s_{\text{blue}}]$ for all opponent agents. So, the group probability will be:

$$Pr^t_{\text{green}}[s_{\text{group}}] = \frac{\sum_{i=0}^{n} Pr^t_{\text{green}}[s_i]}{n} \tag{5}$$

where n = number of blue agents.

Based on this probability , $Pr^t_{\text{green}}[s_{\text{group}}]$, an agent can, up to some extent, predict what is the major visitation site of the opponents. So, based on these, the agent can have following two strategies which it can adopt:

1) Going to the bomb site with lower $Pr^t_{\text{green}}[s_{\text{group}}]$, or

2) Going to the bomb site with higher $Pr^t_{\text{green}}[s_{\text{group}}]$.

We do not know for certain which of these two strategies might be better for an agent. To test these strategies we will run three experiments in each of which agent will use different strategies as done in Section 4.3. Following are the three experiments:

1) Generating the bomb sites randomly.

2) Taking into consideration the bomb site with lower $Pr^t_{\text{green}}[s_{\text{group}}]$.

3) Taking into consideration the bomb site with higher $Pr^t_{\text{green}}[s_{\text{group}}]$.

Another aim of the experiment would be to see how fast a bot can adapt to the opponents changing strategies using fictitious play. For this, we will change the blue agents′ strategy after a cycle of few rounds and will observe the change in strategy of green agents based on this. As an end product, we expect to generate a training graph with a spike in it whenever there is change in opponents′ strategy. This experiment will decide level of unpredictability in an evolved bot using fictitious play.

5 Conclusions

Modern computer games emulate various forms of artificial behaviors in game characters so that they appear intelligent to game players. Traditionally, one of the most common methods of achieving such artificial behavior is cheating, such as increasing the enemy′s energy or some other power to increase the difficulty levels. Most of the AI rules are hard coded in the game′s logic. Once understood by an experienced game player, the game becomes very predictable to him. On the other hand, game developers had dedicated considerable time in configuring these hard coded parameters. As a solution to this problem, this paper presents the use of the state-of-the-art game theory based learning technique called fictitious play in computer games.

In order to limit the domain, we concentrated on improving the behavior of a type of an artificial character called bots, specifically in first-person shooter game of CS. Bots in CS are used to replace human players. Bots play as a part of the team and achieve goals similar to humans, i.e. fighting against enemies or plant the bomb at the bomb sites. We presented our approach on the use of the learning rule of fictitious play for improving behavior of bots. To test our proposed methodology, we developed a virtual simulation environment, similar to the game of CS, consisting of two agent teams-blue and green. Blue agents were controlled manually while green agents used fictitious play to understand the opponent′s strategy. With this approach we have been able to evolve bots (green agents) that will select future strategies based on their past experiences. This has made the bots more unpredictable to a game player and thereby making the game more interesting. The bots using fictitious play also showed better performance compared to bots playing randomly. This work focuses on improving behavior of bots using fictitious play, but in fact this is a very simple strategy, which can be applied to all the problems for which predicting the opponent′s behaviors is required.

References

[1] G. W. Brown. Activity analysis of production and allocation. *Iterative Solutions of Games by Fictitious Play*, T. C. Koopmans, Ed., New York, USA: Wiley, pp. 374–376, 1951.

[2] S. Russell, P. Norvig. *Artificial Intelligence: A Modern Approach*, New Jersey, USA: Prentice Hall, 2003.

[3] D. Johnson, J. Wiles. Computer games with intelligence. In *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, IEEE, Melbourne, Australia, pp. 1355–1358, 2001.

[4] S. M. Lucas. Computational intelligence and games: Challenges and opportunities. *International Journal of Automation and Computing*, vol. 5, no. 1, pp. 45–57, 2008.

[5] A. Nareyek. Intelligent agents for computer games. In *Proceedings of the 2nd International Conference of Computers and Games*, Springer, vol. 2063, pp. 414–422, 2000.

[6] C. Fairclough, M. Fagan, B. Namee, P. Cunningham. Research directions for AI in computer games. In *Proceedings of the 12th Irish Conference on Artificial Intelligence and Cognitive Science*, CiteSeer, NUI, Ireland, pp. 333–344, 2001.

[7] J. Grzyb. Artificial intelligence in games. *Software Developer′s Journal*, [Online], Available: http://www.cs.rochester.edu/ brown/242/assts/termprojs/games.pdf, August 14, 2011.

[8] S. Yildirim, S. B. Stene. A survey on the need and use of AI in game agents. In *Proceedings of the 2008 Spring Simulation Multi-conference*, ACM, Ottawa, Canada, pp. 124–131, 2008.

[9] S. Franklin, A. Graesser. Is it an agent or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, Springer, pp. 21–35, 1997.

[10] J. E. Laird. Using a computer game to develop advanced AI. *IEEE Computer*, vol. 34, no. 7, pp. 70–75, 2001.

[11] R. Adobbati, A. N. Marshall, A. Scholer, S. Tejada, G. Kaminka, S. Scfaffer, C. Sollitto. Gamebots: A 3D virtual world test-bed for multi-agent research. In *Proceedings of the 2nd International Workshop on Infrastructure for Agents MAS and Scalable MAS*, Mendeley, Montreal, Canada, vol. 45, pp. 47–52, 2001.

[12] R. Calder, J. Smith, A. Courtemarche, J. Mar, A. Ceranowicz. ModSAF behavior simulation and control. In *Proceedings of the 2nd Conference on Computer Generated Forces and Behavorial Representation*, STRICOM-DMSO, pp. 347–356, 1993.

[13] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, M. Asada. The RoboCup synthetic agent challenge 97. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, ACM, Nagoya, Japan, pp. 62–73, 1997.

[14] J. Neumann. First draft of a report on the EDVAC. *IEEE Annals of the History of Computing*, vol. 15, no. 4, pp. 27–75, 1993.

[15] J. Neumann, O. Morgentstern. *Theory of Games and Economic Behavior*, New Jersey, USA: Princeton University Press, 1944.

[16] M. W. Lee, J. M. Lee. Generation and control of game virtual environment. *International Journal of Automation and Computing*, vol. 4, no. 1, pp. 25–29, 2007.

[17] J. Broome. Botman′s bots Half-Life bot development, [Online], Available: http://hpb-bot.bots-united.com/, August 14, 2011.

[18] C. Melo, R. Prada, G. Raimundo, J. Pardal, H. S. Pinto, A. Paiva. Mainstream games in the multi-agent classroom. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, ACM, Hong Kong, PRC, pp. 757–761, 2006.

[19] CSNation.net. Counter-Strike, [Online], Available: http://csnation.totalgamingnetwork.com/, August 11, 2011.

[20] B. Scott. The illusion of intelligence. *AI Game Programming Wisdom*, Steve Rabin Ed., Massachusetts, USA: Charles River Media, pp. 16–20, 2002.

[21] A. Bartish, C. Thevathayan. BDI agents for game development. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-agent Systems*, ACM, Bologna, Italy, pp. 668–669, 2002.

[22] A. Rao, M. Georgeff. BDI agents: From theory to practice. In *Proceedings of the 1st International Conference on Multi-agent Systems*, Mendeley, California, USA, pp. 312–319, 1995.

[23] P. Patel. Improving Computer Game Bots′ Behavior Using Q-Learning, Ph. D. dissertation, Southern Illinois University Carbondale, USA, 2009.

[24] R. Sutton, A. Barto. *Reinforcement Learning: An Introduction*, Massachusetts, USA: MIT Press, pp. 3–10 and 51–75, 1998.

[25] M. McPartland, M. Gallagher. Learning to be a bot: Reinforcement learning in shooter games. In *Proceedings of the 4th Artificial Intelligence and Interactive Digital Entertainment Conference*, California, USA, pp. 78–83, 2008.

[26] N. Nisan, T. Roughgarden, E. Tardos, V. Vazirani. *Algorithmic Game Theory*, New York, USA: Cambridge University Press, pp. 3–27, 2007.

[27] R. Myerson. *Game Theory: Analysis of Conflict, Massachusetts*, USA: Harvard University Press, pp. 1–31, 1997.

[28] P. Dutta. *Strategies and Games — Theory and Practice*, Massachusetts, USA: MIT Press, pp. 37, 103–115, 139–148, 1999.

[29] S. Kuhn. Prisoner′s Dilemma, [Online], Available: http://plato.stanford.edu/entries/prisoner-dilemma/, August 14, 2011.

[30] D. Robinson, D. Goforth. *The Topology of 2×2 Games — A New Periodic Table*, USA and Canada: Routledge Publication, pp. 73–75 and 81–83, 2005.

[31] J. Zeleznikow, E. Bellucci, A. Lodder. Integrating artificial intelligence, argumentation and game theory to develop an online dispute resolution environment. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, IEEE, Boca Raton, USA, pp. 749–754, 2004.

[32] H. Moulin. *Game Theory for the Social Sciences — Studies in Game Theory and Mathematical Economics*, New York, USA: New York University Press, pp. 148–177, 1986.

[33] J. Nash. Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[34] D. Monderer, A. Sela. Fictitious Play and No-Cycling Conditions, [Online], Available: http://www.sfb504.uni-mannheim.de/publications/dp97-12.pdf, August 14, 2011.

[35] M. Vidal. Learning in Multiagent Systems: An Introduction from a Game-theoretic Perspective. [Online], Available: http://jmvidal.cse.sc.edu/papers/vidal03a.pdf, August 14, 2011.

[36] R. Baumgarten, S. Colton, M. Morris. Combining AI methods for learning bots in a real-time strategy game. *International Journal of Computer Games Technology*, vol. 2009, Article ID 129075, 2009.

[37] N. Cole, S. J. Louis, C. Miles. Using a genetic algorithm to tune first-person shooter bot. In *Proceedings of the International Congress on Evolutionary Computation*, IEEE, vol. 1, pp. 139–145, 2004. [Online], Available: http://www.cse.unr.edu/ sushil/pubs/newpapers/2004/cec /cole/paper.pdf, August 14, 2011.

[38] P. Patel, H. Hexmoor. Designing bots with BDI agents. In *Proceedings of International Symposium on Collaborative Technologies and Systems*, ACM, Maryland, USA, pp. 180–186, 2009.

[39] S. Zanetti, A. E. Rhalili. Machine learning techniques for FPS in Q3. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACM, Singapore, pp. 239–244, 2004.

[40] M. Wooldridge, N. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.

[41] M. Wooldridge. *An Introduction to Multiagent Systems*, England: John Wiley & Sons, 2002.

[42] M. Tennenholtz. Game-theoretic recommendations: Some progress in an uphill battle. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, ACM, Estoril, Portugal, vol. 1, pp. 10–16, 2008.

[43] D. Moderer, L. Shapley. Fictitious play property for games with identical interests. *Journal of Economic Theory*, vol. 68, no. 1, pp. 258–265, 1996.

[44] E. Gerding, Z. Rabinovich, A. Byde, E. Elkind, N. Jennings. Approximating mixed Nash equilibria using smooth fictitious play in simultaneous auctions. In *Proceedings of the 7th International Conference on Autonomous Agents and Multi Agent Systems*, ACM, Estoril, Portugal, vol. 3, pp. 1577–1580, 2008.

Ushma Kesha Patel received the B. Tech. degree in computer engineering from the Kurukshetra University, India in 2007, and the M. Sc. degree in computer science from the Southern Illinois University, Carbondale, IL, USA in 2011. At present, she is an MBA student at the Southern Illinois University, Carbondale, IL.

Her research interests include game theory, multiagent systems, and financial management.

E-mail: ushma@siu.edu (Corresponding author)

Purvag Patel received the B. Eng. degree in computer engineering and M. Sc. degree in computer science from the Gujarat University, India in 2007 and the Southern Illinois University, Carbondale, IL, USA in 2009, respectively. He is currently working towards the Ph. D. degree in computer science at the Southern Illinois University, Carbondale, IL, USA.

His research interests include fuzzy logic, social network analysis, search engines, computer games, and multiagent systems.

E-mail: purvag@siu.edu

Henry Hexmoor received the M. Sc. degree from Georgia Tech, Atlanta, USA, and the Ph. D. degree in computer science from the State University of New York, Buffalo, USA in 1996. He taught at the University of North Dakota, USA before a stint at the University of Arkansas, USA. He is currently an assistant professor with the Department of Computer Science, Southern Illinois University, Carbondale, IL, USA. He has published widely in artificial intelligence and multiagent systems.

His research interests include multiagent systems, artificial intelligence, cognitive science, and mobile robotics.

E-mail: hexmoor@cs.siu.edu

Norman Carver is an associate professor of computer science at Southern Illinois University Carbondale.

His research interests include distributed problem solving (a subfield of multiagent systems) and machine learning. His DPS work includes theoretical approaches (such as DEC-MDPs), simulations, and tools to support human engineering of DPS systems. His machine learning work currently involves music evaluation and multiagent learning.

E-mail: carver@cs.siu.edu