

# Artificial Intelligence In Sports: A Study Upon American Football

Gregory S. Nilsen  
Computer Science Department  
Hiram College  
Hiram, OH 44234  
nilsengs@hiram.edu

## Abstract

*Sports have long been more of a realm of real-world application, but the growth of the video gaming industry has changed that in the past decade. Players are continually looking for the added bit of realism to make the gaming experience more authentic, and coaches may one day actually design plays and test them in a virtual environment. We will explore the methods of creating a realistic American football simulation including knowledge representation and strategy, specifically using a two-player minimax methodology. We demonstrate the potentials of such a simulation, as well as the difficulties in creating the simulation.*

## 1. Introduction

Throughout the life of artificial intelligence, games have been an interest to the field because of the way games engage the intellectual faculties of humans [7]. Sports, which are simply games with a lower level of abstraction has, however, been relatively untouched by the field. Due to their less abstract nature, these sports have been found to be more difficult to represent and intelligently engineer.

However, some sports lend themselves better to abstraction due to their play format and rule systems. One of these sports is American football. With very clear and distinct rules, accessible nature and a play-by-play format, the sport lends itself much better to representation for use in artificial intelligence. One of the more intelligent parts of the game of football is defense. The ability of a defense to read and react to an offense can be a difficult task for humans, but it can also be represented in the structures of a computer.

There are many uses for the development of a defensive football agent as well. It can be used for standard game applications, increasing the level of realism in them. It can also be used as a coaching aid.

As Percell [8] states, “computer football games give consumers hands on coaching opportunities”. In this manner, the defense can be used to develop new offensive plays, or even test new defensive formations. It is these reasons why this paper will focus upon the study of artificial intelligence in the defense of American football.

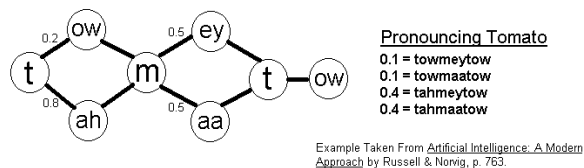
## 2. The Task

The task of this project was to create a team of defensive football players. This team should be able to respond to a programmed offense, where plays are set and run and the offense does not adapt to the defense. This was done in order to easily recreate results and to ease debugging procedures. The defense should also use various formations and packages in order to minimize the advancement of the football by the offense. The use of various formations also allows for increased realism, since in football many different formations are used, and forces defensive players to react to similar plays in different ways.

## 3. Paradigms for Game Playing

With games being such a large part of artificial intelligence, there are many paradigms for game play. However, there are five that seemed especially applicable to the sport of football. These are the fuzzy logic approach, stochastic game approach, Markov model, the Approachability-Excludability theory, and Minimax search with Alpha-Beta pruning.

Dubois [1] states, “It is the formalization of some forms of commonsense reasoning that has motivated the development of fuzzy logic”. Since football has many situations that involve “commonsense reasoning”, this seemed like a solid approach. However, football is also a very



**Figure 1: Markov model – an example**

accessible game, leaving a limited amount of uncertainty.

The Stochastic game paradigm, which Haller [3] tells us was introduced by Shapely in 1953, creates models of possible moves along with a probability of each move. As Shimkin [11] states, “the game matrix which is played at each stage depends on some ‘state’ variable which may change from stage to stage”. This was also another reasonable approach in writing, but the prediction of moves by the offense proves to be a daunting mathematical task; one that could not be completed within the scope of this project.

The Markov model, according to Norvig [7], is a method of describing a process of creating all possible paths, from start to finish, and assigning each a probability. Again, this paradigm contained a very complicated mathematical model. Also, the creation of all paths through the entire process also seemed to limit the possibility of reactions in game play.

The Approachability-Excludability theory of Shimkin [11] provides a belief that a player can force the average payoff in a game in their advantage regardless of the moves made by the other player. However, this theory is relatively limited in research and unverified.

The final paradigm considered was the Minimax search with Alpha-Beta pruning. This method searches through all possible moves to find the best move for each player using a zero-sum method. This standard recursive algorithm is very generic and easily allows for manipulation towards many facets of game play.

## 4. Approach

Realizing the case that it is possible to represent football in a two-player model and the decision to use the LISP programming language (which optimizes recursion), made the Minimax search with Alpha-Beta pruning the most logical choice for the situation. Also, already having a familiarity due to research found concerning the generic minimax function [5] helped to make the decision as well. The research that was found dealt with the game of tic-tac-toe. Since this game is fairly simple to represent, a lot of work with data manipulation was required in order to get this approach to work with football.

- Field
  - Array
  - Set of Players
  - Ball
- Players
  - Team
  - Location on Field
  - Position on Team (i.e. “running back”)
  - Blocked?
  - Open?
  - Ball?
- Ball
  - Location on Field
  - Height

**Figure 2: Structures Used to Play Football**

As part of a bottom-up design, the program was developed in many pieces that eventually wove together into one application. Each step in the process was developed and tested individually before being added into the rest of the application. This design resulted from the recognition that many larger pieces depended on the development of smaller, or less significant pieces.

### 4.1. Structures and Representation

The main structure needed to play football is a field, simply because you cannot play if there is nowhere to do so. In this case, the decision was made to use a “practice field” representation, to minimize space used. The field began as a simple thirty-by-thirty array, with each position of the array representing a square yard on the field. These positions could then take in the team of the player structure “occupying” that position. Unfortunately, this structure was flawed and had problems being used in the minimax function. Later on in the project, the field encompassed not only an array, but a set of player structures and a ball structure as well (see Figure 2). This allowed for easy duplication later on in the process.

According to Freeborn [2], defense comes down to two main objectives: guard and defend. Working off of this simple defensive model, the player structure developed into the model in Figure 2. The player structure is used for both offensive and defensive players, though both sides do not use all fields. Some properties are specifically for defensive and some for offensive players.

The team field is occupied by either an X (defense) or an O (offense), which is used in the printing of the field. Two fields, row and column represent the location. These provide the location of the player on the field array. The position of the player is occupied by a position description, such as “right linebacker” or “running back”. The blocked?

property indicates if there is an offensive player directly in “front” of a defensive player. Open? indicates whether or not a receiver, tight end or running back has a defender in one of the eight field positions surrounding them. Lastly, ball? is true if the player has the ball.

The final structure is the ball. The ball has a location, which is represented in the exact same manner as in the player structure. The ball also has the added structure of height. The height contains a zero, one or two as a value. A value of one means that the ball is “in hand” for a player and still in play. A height of zero indicates that the ball is on the ground, due to either a tackle or incomplete pass. Lastly, a height of two means that the ball is in the air and cannot be deflected by any player.

The other important function here was the printing of the field. To keep things simple, a text-based representation was used. This system used X to represent defenders, O to represent offensive players, B to represent the ball, and horizontal and vertical lines to draw the borders of the field.

#### **4.2. Movement Functions**

In order to play the game of football, some sort of player and ball movement is necessary. Due to the representation of the field as an array, this left five basic moves: moving right, left, up, down, and stay in the same position. The stay move was not originally included in the movement designs. However, it proved necessary for refreshing the properties of a player when a pass was thrown. Each player movement function, except stay, includes a check to see if the destination was occupied to avoid confusion. If a position is found to be occupied, then the move is changed to stay instead. However, all players have the same speed using such a representation. Separate directional movement functions were also written for the ball, however the occupation checks were not found to be necessary for the structure.

#### **4.3. Offense Definitions**

The next step in the development process was to define the offense. This was done before the creation of the defense so that results of offensive moves could be found, allowing this information to be used for the implementation of the defense. Seven plays were created, including four runs and three passes, in order to test the abilities of the defense in multiple situations. Each play is executed in a separate function for functionality purposes, and ease of reading.

In order to complete the running plays, a handoff function was necessary to change the ball

location and player ball? properties. This allowed the ball to be handed off to another player in a fashion typical of football.

The passing plays also required special functions. The handoff function could not be used because it did not realistically represent a typical pass. Instead, two functions were written, one for throwing and one for catching. The throwing functions put the ball in the air, with no player having possession. The catching function did the opposite of the throwing function, bringing the ball down to standard level and in possession of the receiver was open. Otherwise the height of the ball would be set to zero and the pass would be considered incomplete.

Once all of these functions were complete, the offense was tested for validity. When run alone, no difficulties were encountered. Then a defensive formation, one that would not move, was added to see if this affected the plays. When the passing plays were run, the receivers ran into the defenders in “front” of them and did not move for several turns. Then when the ball was thrown to a position that was several spaces away, the receivers still managed to catch the ball. This was physically impossible in actual football, so a change had to be made. Alterations were made so that a receiver had to be in the same position as the ball in order to make a catch. When this change was made, the offense worked as desired.

Also included in the offensive plays was a statement that checked a move counter, which is initialized in the main function. This value lets the play know how far it has progressed and what moves to make in the functions next call.

#### **4.4. Defense Definitions**

After having the offense in place, the time had come to work on the heart of the task: the defense. Howland [4] declares that artificial intelligence in games is “not interested in creating a perfect reactionary machine in a game enemy”. Instead it is “interested in providing a challenge for the player”. This is very applicable to football. A defense in the sport is never perfect, instead it tries to minimize the losses it incurs. McCune [6] refers to this as a “good” or “near-optimal” solution, and notes they are especially applicable to knowledge-based systems concerning defense.

The first part of the defense that was approached was the defensive formations. In order to get a reasonable amount of differential, four different formations were chosen. Two traditional formations were chosen for the defense, the 4-3 and the 3-4, which Percell [8] goes into more detail about. The

other two formations used were variations on the 4-3, shifting the linebackers to the right or left.

Once the formations were complete, the next task needed was to define the property checks. These are functions that checked if a player was open, blocked, or tackled. The open and blocked functions set the player properties as mentioned above. The tackled function was a little different because it did not affect a property of a player, but of the ball. The function was given a defensive player and then determined that if they were not blocked and if there was an offensive player with the ball in one of the eight surrounding locations, then the ball height was set to zero and the player was considered down.

Once the main defensive functions were defined, it came time to add the essential Minimax search with Alpha-Beta pruning function. This function itself required several other functions to be written in order to get full utility from the minimax function:

- ❑ Static Function—The static function tells the minimax function how “good” a move or position is to the current player (or team of that player). Our function checked several different properties to determine this value for the defense. The main influence on the value was the position of the ball on the field, just as in real football. Every yard, or space, past the line of scrimmage, or starting point, was taken and multiplied by 20 for the initial static value. This value was set so that it had more impact than any other. Other values were checked, which affected this initial value. These other values dealt with the openness of receivers (subtracting from one to three depending on the receiver) and the blocked properties of defenders (subtracting one for every player blocked). The final check was for the height of the ball at the position, and if the ball was down the static value was set to 9999. The static evaluation for the offense was the exact opposite of that for the defense, in order to maintain a zero-sum game.
- ❑ Copy Functions—Normally in LISP, when a structure is created, a copy function is automatically created for that structure. The problem with these automatically created functions is that they often do not create exactly what the programmer desires. This was the case with the copy function for this field. Instead a function that copied array positions from one array to another was included. This allowed for the exact duplication of the playing field. Then the players and ball were copied from one field

to another, creating a proper copy of the field.

- ❑ Move Generation Function—One of the abstractions that the minimax function uses is a move generation function. This function takes the current player and determines all the various moves a player can make. This returns a list of fields where these moves have been made.
- ❑ Value Manipulation Functions—Due to the nature of this minimax function, it needed several functions that took results and returned specific parts of them. These are most used in the pruning portion of the minimax function.
- ❑ Result Interpretation Functions—These were the last part of the program dealing with the minimax functions that were written. Once the Minimax function was completed, it was noted that it returned a static value and a field. In order for the current field to be changed to the one resulting from the minimax function, it had to be examined by another function that recognized the differences between the two fields and made the proper move.

Once these functions were completed, the actual Minimax function could be plugged into the code and be fully operational.

#### 4.5. Main Function

The final portion of the code was the main function. This is called with a desired search depth as the only input to it. First, the function creates a new field on which the play is to be executed. It then asks the user for an offensive play between one and seven, and for a defensive play between one and four. Next, a loop is begun, which will only halt when twenty moves have been executed or the ball height is zero. It then takes the play information, along with an initialized move counter, and calls the offensive function. After the offense has executed their moves, then the defense does a set of property checks on the receivers and defensive players. Then the main function calls an update on the current field. After that is completed, the defensive function, which includes the minimax function, is executed. This process continues until a stopping condition is reached.

#### 4.6. Testing

The final step in the development process was testing the overall application. Because of the design of the program, the main portion of testing

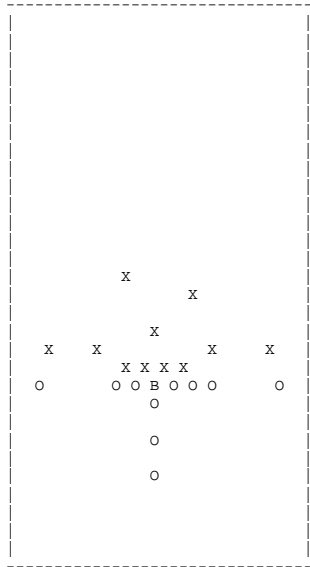


Figure 3: Initial field position

focused on how well the program played defense. Several slight changes to the static evaluation and move generation functions were made during this process, but no major changes were required at this point.

## 5. Experimental Results

In our attempts to create a realistic football defense, we noticed several points when running our experiments. For the most part, these observations dealt with the speed and effectiveness of the algorithm.

The first thing that was noticed was that the algorithm is very slow. Even when compiled in LISP, which greatly increases the speed of execution for the program, the program ran for five to ten minutes using a depth of one on a 233 MHz Linux machine. More extensive depths caused the program to run at unreasonable lengths. One test case with a depth of three ran for twenty-four hours before completion, and another case at depth five ran for 72 hours and only completed two moves.

The speed efficiency issue makes the application of this program to video games or real-time simulation unlikely. Results need to come in quickly for these applications. As Reddy [10] states, “long-term success in AI will also require advances in all of computer science”. Though this may help the algorithm, it is still too lengthy for these applications. However, the program may still be usable for applications such as a play developer or tester.

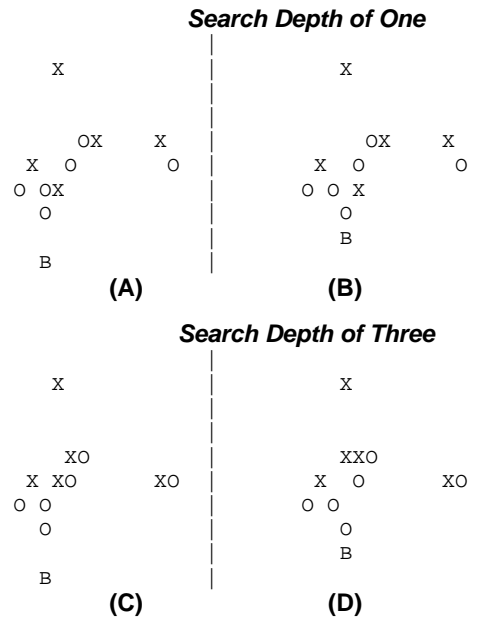


Figure 4: Difference in aggression based on search depths (right side of field only)

When the effectiveness of the algorithm is examined, it is noted that the algorithm runs effectively, not perfectly, against running plays. The same held true for the passing plays, though the occasional lapses seemed to occur more often. Most of these lapses were moves that were made appropriately to the algorithm implemented, but the moves were unrealistic in actual application. Just as Pitrat [9] noted, “the successes of many AI systems are not due to their own intelligence, but to the intelligence of their designers”. Apparently, more expert input would be necessary to cause more improvements in the system.

One of the more important things noted in experimentation was the effect of depth on moves. In Figure 4, we see the same play being run against the same defense on different search depths. (A) and (C) are both at the same point in the play, and it can already be seen that depth one causes the defense to be more aggressive. When we look one step ahead at (B) and (D), we see the trend continue. This has resulted because at a search depth of one, the defense sees the best move of the offense to be a forward one. However, at a depth of three, the defense anticipates the runner to move to the side to avoid being tackled by a move forward. Hence, the trend is that at a lower search depth, the defense is more aggressive and willing to take chances. This is effective against a set offense, but may not be against one that adapts to the defense.

## 6. Future Changes/Improvements

At the point of completion of this project, there are many changes that were desired. Some changes would be made from where the project stands now. Unfortunately, there were also some that would come from starting the project anew.

One thing that would definitely be changed in the project as it currently stands is the static evaluation. The function needs to be more realistic, perhaps including a component of intuition. As it is now, the defensive players do not all attack well, and many just stand in place. These attributes are not very realistic to actual football play.

Another change would be in the coverage representation. The defender cannot cover a receiver from any side because, in actual application, the ball can still be caught if the defender is on the wrong side of the receiver. A change to make the defender stay between the receiver and quarterback in coverage would be preferable.

A few other changes that would be made are in the efficiency of the game, creating an artificially intelligent offense, and creating full game play instead of the "practice field" atmosphere.

If the project were to be done again, choosing another defensive algorithm for searching would be the greatest desire. An algorithm that is more efficient and provided more realistic choices would be preferable.

A change in the representation would also be desired. The moves could be more realistic in a less limiting field environment, and possibly with a graphical display. Again here, an artificially intelligent offense and full game play would be desired.

## 7. Conclusion

We have developed here the processes of an American football defense, consisting of strategy and game play. In this paper, it has been demonstrated that it is possible to represent sports as less abstract games. However, the process is difficult and complex. Though this project has been successful in the goals that was set forth and in the time frame given, it still leaves much to be desired. This is only the first step in the process, and much has been learned already. Having learned more about different types of artificial intelligence game strategies and the importance of the algorithm efficiency in large problems will be a great asset in the continuation of this project. Also, the importance of good design was a key component to this project coming together smoothly, thanks to the bottom-up design. Overall,

the system accurately completed its goals in a mostly effective manner.

## References

- [1] Dubois, Didier and Henri Prade. "What Does Fuzzy Logic Bring to AI?" ACM Computing Surveys Sept. 1995: 328-30. ACM Digital Library. On-line. ACM. 3 Feb. 2000.
- [2] Freeborn, Roger A. "Defense". Football Faxuals. n.pag. On-line. Internet. 18 Feb. 2000. Available: <http://jvm.com/coachfree/DefenseIntro.html>
- [3] Haller, Hans and Roger Lagunoff. "Genericity and Markovian Behavior in Stochastic Games". Journal of Economic Literature. Revised: Sept. 19 May 1999.
- [4] Howland, Geoff. "Introduction to Learning in Games". Gamedev.net n.pag. On-line. Internet. Available: <http://www.gamedev.net/reference/programming/ai/article547.asp>
- [5] "Laboratory Resources on Search and Game Playing". n.pag. On-line. Internet. Available: <http://yoda.cis.temple.edu:8080/UGAIWWW/resources/search-resources.html>
- [6] McCune, Brian P. "The Potential for AI in Strategic Defense" 1997. ACM Digital Library. On-line. ACM. 3 Feb. 2000.
- [7] Norvig, Peter and Stuart Russell. Artificial Intelligence: A Modern Approach. Upper Saddle River, NJ: Prentice Hall, Inc., 1995.
- [8] Percell, Justice Casey. "NFL Defensive Strategy". Basic NFL Strategy: n.pag. On-line. Internet. Available: <http://www.idscorral.com/corfbid.shtml>
- [9] Pitrat, Jacques. "AI Systems Are Dumb Because AI Researchers Are Too Clever". ACM Computing Surveys Sept. 1995: 349-50. ACM Digital Library. On-line. ACM. 3 Feb. 2000.
- [10] Reddy, Raj. "Grand Challenges in AI". ACM Computing Surveys Sept. 1995: 301-3. ACM Digital Library. On-line. ACM. 3 Feb. 2000.
- [11] Shimkin, N. and A. Shwartz. "Guaranteed Performance Regions for Markov Models". The Institute for Systems Research TR 91-8: 1991. The Institute for Systems Research Technology Reports. On-line. The Institute for Systems Research. 18 Feb. 2000.