2012

Comparing AI Archetypes and Hybrids Using Blackjack

Robert Edward Noonan
Minnesota State University - Mankato

Comparing AI Archetypes and Hybrids Using Blackjack

by

Robert E Noonan

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in

Computer Science

Minnesota State University, Mankato

Mankato, Minnesota

July 2012

Comparing AI Archetypes and Hybrids Using Blackjack

Robert E Noonan

This thesis has been examined and approved by the following members of the thesis committee.

Dr. Rebecca Bates, Advisor

Dr. Dean Kelley

Dr. Christophe Veltsos

Abstract

The discipline of artificial intelligence (AI) is a diverse field, with a vast variety of philosophies and implementations to consider. This work attempts to compare several of these paradigms as well as their variations and hybrids, using the card game of blackjack as the field of competition. This is done with an automated blackjack emulator, written in Java, which accepts computer-controlled players of various AI philosophies and their variants, training them and finally pitting them against each other in a series of tournaments with customizable rule sets. In order to avoid bias towards any particular implementation, the system treats each group as a team, allowing each team to run optimally and handle their own evolution. The primary AI paradigms examined in this work are rule-based AI and genetic learning, drawing from the philosophies of fuzzy logic and intelligent agents. The rule-based AI teams apply various commonly used algorithms for real-world blackjack, ranging from the basic rules of a dealer to the situational "rule of thumb" formula suggested to amateurs. The blackjack options of hit, stand, surrender, and double down are supported, but advanced options such as hand splitting and card counting are not examined. Various tests exploring possible configurations of genetic learning systems were devised, implemented, and analyzed. Future work would expand the variety and complexity of the teams, as well as implementing advanced game features.

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

Two hundred dollars are sitting on a green felt table. The player looks down at the cards in front of him, an ace of clubs and an eight of spades. The player looks across the table to where the dealer is watching with an expression of detached serenity, patiently waiting for the choice to be made. Before him is a queen of hearts and another card lies face down beside it. The other players are waiting as well, though nowhere near as patiently. One choice would double that money, but which? One choice would take it all away, but which? Which would you choose? Making decisions in blackjack can be modeled by rules and other learning approaches. This work asks the question of whether genetic learning can develop an artificially intelligent player that can play as well as – or better than – the rules devised by various dealers and experts.

As card games go, blackjack is one of the simplest. Every card has a value: number cards are worth whatever number they display, face cards (jacks, queens, and kings) are worth ten points each, and aces are either worth one point or eleven, whichever suits the player's hand best. The goal of the game is to collect a hand of cards whose total value is greater than that of the dealer's hand, yet no greater than twenty one.

A game of blackjack follows a simple pattern. Each player places a bet on the table. The dealer deals two cards to each player at the table as well as two for the dealer. Each of these cards is visible to everyone at the table, with the exception of the first card the dealer drew, which is held face down. Once the cards have been dealt, each player takes turns deciding whether to draw another card, known as "hitting", leave the cards alone for the rest of the game, known as "standing", or pursue one of the more rare, situational options available in the game. Once each player has finished drawing cards, the dealer draws the rest of his or her hand, drawing cards until the hand passes a predetermined total. Any player who has a total that

surpasses the dealer's, yet remains under twenty one, wins the game, while anyone who matches the dealer's total neither wins nor loses anything. If the dealer's total goes over twenty one, then anyone whose total is twenty one or less wins. How much the winners receive for their victory depends on the dealer, but the most basic version is that for every unit (e.g., dollar, chip, or point) a player bets, they get two back if they win [1].

Besides the standard hit and stand options, there are some additional options that have situational uses. The first of these is the "double down", which can only be used during a player's first turn. The player receives one more card and then stands, and whatever bet they originally made is doubled. The second option is "surrender", where the player automatically loses but whatever bet they originally made is halved. This limits the cost of losing a hand, but does so at the expense of any potential positive outcome. A third option, called "split", is only available if the player starts with a pair of identical cards, allowing them to turn the two cards into two separate hands, draw another card for each, and play both hands at the same time. Additional information about blackjack may be found online, including on professional gambling advice websites such as Wizard of Odds [2] and Beating Bonuses [1]. This work will focus on the standard rules of blackjack as well as the situational rules of double down and surrender. While many players may attempt to "count cards", a blackjack strategy where the player keeps track of which cards have been played so far and uses this information to adjust their predictions as to which card the dealer is hiding, this will not be explored here.

Blackjack is a useful game for testing artificial intelligence (AI) approaches. The rules are straight forward; the possible interactions are tightly defined; the risks and the rewards are clearly stated, as are the situations in which risks and rewards apply. All of these make it an easy game to implement and test. The most important aspect of the game, however, is the fact that the

information in it is incomplete. When playing chess, a player may not know what their opponent is thinking, but they know everything about the battlefield and can plan accordingly. When playing blackjack, that hidden card in the dealer's hand makes all the difference in the world, but it is unknown to all players except the dealer. As a result, players are forced to guess, to develop an intuition as to whether it is worth the risk to take a given action while that key piece of information is still missing [3]. The goal of this work is to find AI approaches that can rival the rules and strategies established by real world dealers and players.

This thesis describes the fundamental concepts used in the execution of this work, outlines the software created for it, and presents and analyzes the results of the tests conducted using the software. The thesis concludes by discussing the lessons learned and some suggestions for advancing this work in the future.

Chapter 2 - Background

This chapter includes a description of the artificial intelligence concepts of intelligent agents, rule-based AI, genetic algorithms, and fuzzy logic. The chapter closes with a discussion of how AI approaches have been used to play card games.

2.1. Intelligent Agents

Artificial intelligence agents have been defined in multiple ways, with the key idea being the encapsulation of an artificial intelligence into a single entity that is distinct from, yet able to interact with, its environment [4]. A simple example of a simple reflex agent given by Rudowsky is of a thermostat, which monitors the ambient temperature of a room and determines whether or not the furnace needs to be engaged [4].

The key word in that description is “encapsulation”. By containing an artificial intelligence as a single entity separate from the environment around it, one agent can be swapped for another. As long as both agents interact with the environment in the same manner, the environment does not need be customized to suit each individual, regardless of any differences in the agent functions. This also means that new forms of artificial intelligence can be introduced without needing to modify the environment, and that the environment can be relied upon to treat each agent equally, so that the only difference in a system is how each agent responds to environmental inputs. In this work, all forms of artificial intelligence were implemented as “player” agents. The blackjack program, i.e., the game environment, does not know or care which types of players are in the game; it treats each one in the same manner.

2.2. Rule-Based AI

Rule-based AI makes decisions based on a list of rules, implemented in a priority-based order. These rules dictate courses of action and include conditions in which the prescribed action

is applicable. In other words, it can be implemented as a chain of if-then statements an agent can use to determine how to respond to its current situation; it observes the circumstances, selects the rule (or rules) that best apply, and implements them [5]. Rule-based AI is used as a tool for classifying [6], mining [7], and modeling data [8]. It is often combined with other AI paradigms in research, such as genetic learning [9], fuzzy logic [10], and neural networks [8].

In the case of blackjack, each player is given two pieces of information: the cards they have in their hand and the displayed card in the dealer's hand. In a rule-based system, for each potential combination of those two facts, there is a predetermined rule that dictates what action the player should take for their turn.

In this work, a rule-based structure serves as the foundation of each player in the form of a 17x10 decision matrix. This covers the ten possible dealer card values, face cards having the same value as a ten, and the 17 possible scores a player can have, from a pair of twos to any hand with a score of 20. Since a score of 21 is a success and anything above a 21 is an automatic failure, these values need not be included in the matrix. Each cell of a decision matrix has a value between 0 and 4, with each value representing a course of action: 0 for surrender, 1 for stand, 2 for hit, and 3 and 4 for double down otherwise stand or hit, respectively. The only exception to this approach is when fuzzy logic is used, which will be described in Section 2.4.

2.3. Genetic Algorithms

Derived from the theory of evolution and the old axiom of "survival of the fittest" [11], a genetic algorithm attempts to find a solution by keeping what works, getting rid of what does not, and creating variations of what works. It accomplishes this by maintaining a population of possible solutions to a given task. Each solution is tested and given a score based on how well it performed when attempting the task. Those that score well are kept while those that do not are

discarded and replaced by new solutions. By repeating this cycle of testing, destruction, and creation, this archetype slowly builds up a collection of the best performing solutions, each one tried by multiple generations of tests.

The first step in genetic learning is deciding on which aspects of the problem must be optimized. These are the "genes" of the problem. In blackjack, the genes can be defined as the actions prescribed by each rule in the rule-based structure used for the players. The conditions of each rule can remain static, ensuring that all possible permutations are accounted for. Their respective solutions are all that need to be modified. Once the genes are determined, a population of initial solutions is created. These can be randomly generated solutions, but known solutions can also be placed in the population in order to give the population a head start.

Once this initial population is tested, the lowest scoring portion of the population are discarded and replaced, either by randomly generating new solutions, referred to as mutation, or by combining the genes of two high performing solutions, referred to as breeding. When breeding two solutions, a random subset of genes from one "parent" is used by the new "child" solution, while the remaining genes are supplied by the other parent. Ideally, this new solution should perform better than either parent, though the actual result is a matter of chance. The other advantage this provides is that the genes of effective solutions are spread throughout the population, making it easier to find the best combination. In theory, given enough time, better solutions will present themselves, and the process will result in the best solution possible.

Genetic algorithms have been developed as tools for many practical applications, ranging from the artificial generation of art in the form of line drawings [12] and the composition of music [13] to the arrangement of construction sites [14] and cancer research [15]. It is also used in topics such as scheduling [16] and software testing [17].

2.4. Fuzzy Logic

First explored by Lofti A. Zadeh in 1965 [18], fuzzy logic attempts to mathematically describe concepts that are vague or open to interpretation. Common examples of fuzzy logic are the question of what the descriptor "warm" means with regards to temperature or how tall a person needs to be to qualify as "very tall" [19]. In order to describe such concepts, values are assigned a likelihood that they qualify for a given descriptor. A person with a height of 5 feet and 6 inches, for instance, could be considered very tall by the standards of a kindergarten class but would never be classified as such by a professional basketball team. If a group of people were asked to make the judgment about whether the person is very short, short, average, tall, or very tall, only a small percentage would say that the individual was very tall. In fuzzy logic terms, that could mean that individual was 15% in the category "very tall".

For the example categories, the percentages for a given person do not have to sum to 100%. A subject's "membership" in a certain classification is independent of that subject's membership to another classification. In this example, 15% of people would consider 5'6" to be very tall, while 85% would not. At the same time, perhaps 45% of people may consider 5'6" to be simply "tall", and 55% would consider this as still too short to fit the description.

A fuzzy logic-based blackjack hand would have membership values defined for each of the available actions. For example, it might be a 10% good idea to surrender, a 50% good idea to stand, a 70% good idea to hit, and a 35% good idea to double down. The most correct answer would be to hit in that circumstance, but all of the available options are viable, and the player may select any of them.

In this work, fuzzy logic is implemented through expanding the 17x10 decision matrix into a third dimension, dividing each of the matrix's 170 cells into 5, one for each available course of action as defined in Section 2.2. Each of these new cells contains an integer with a value between 1 and 100, representing the likelihood that a course of action will be suggested. When a fuzzy logic player is required to make a decision, the dealer's display card and the value of the player's hand are used to identify the correct set of values. Five integers between 1 and 100 are then randomly generated, and each is compared to the value in one of the cells. If the number is less than or equal to the value held by its respective cell, the action is suggested. One of the suggested actions is selected at random and then implemented. If none of the actions is suggested, the player will stand.

Fuzzy logic players go through the same genetic learning process described in Section 2.3. A population of players are randomly generated and then tested over a number of hands of blackjack. The best performers are kept, while the rest are discarded and replaced either by breeding the remaining players or by randomly generating new ones. For the purpose of breeding, the five cells for each combination of dealer card and player hand are treated as a single gene. Additionally, some experiments require pre-defined genetics for the initial population. While these particular players do not necessarily qualify as "fuzzy", randomly generated players will continue to be introduced to the population with each generation as in genetic learning.

2.5. Games and Artificial Intelligence

When seen through the framework of artificial intelligence, games can be divided into two groups: those that allow the player to know everything related to the game (referred to as "perfect information") and those that leave certain, often critical, information hidden (referred to

as “imperfect information”) [20]. Board games are a good example of perfect knowledge, because the player is able to see all of the pieces on the board and know how each one is able to move. Examples are simple games like tic-tac-toe, complicated ones like chess [21], and logic challenges such as the Eight Queens puzzle, where eight queens have to be arranged on an 8x8 chess board so that none of them are able to threaten any other [22]. The crucial thing is that everything about the game is readily apparent except the minds of any other players. Perfect information games have long been a popular topic for artificial intelligence experiments and significant successes have been achieved, such as winning games against world champions in checkers (Marion Tinsley, 1994) [23], chess (Garry Kasparov, 1997) [24], and Othello (Takeshi Murakami, 1997) [25].

Card games, however, are often based on imperfect information. Whether the game is solitaire, poker, blackjack, or old maid, the value of some or most of the cards is hidden from opposing players’ sight, either face down on the table or fiercely guarded in the hand of another player. While it can and has been argued that many perfect information games can be played with simple brute force [24], using implausibly large decision trees to chart the possible ways the game could progress, imperfect information games require something rather more abstract. Whether you call it instinct, intuition, or raw probability, it is a difficult thing to do right.

Works using card games in artificial intelligence have explored several varieties of game, such as bridge [26] and solitaire [27], with poker in particular being a popular game to employ [28,29]. Blackjack serves as a simplified alternative to the more complicated poker, boasting simple rules, limited player options, clearly defined win and loss conditions, and minimal player interaction as each player is only ever in direct competition with the dealer. Blackjack has been used as a tool to explore neural networks [28,29], genetic learning [3,21,28], hidden Markov

models [32,33], and even how humans responded when playing beside intelligent agents [34]. The project most closely related to this work, however, was presented in 2005 by Curran and O'Riordan [30], which ran simulations of blackjack with various AI paradigms such as neural networks and genetic learning.

Chapter 3 - Methodology

This chapter describes the software developed for this work, as well as the primary functions the software was designed to accomplish and the player types implemented for the software's use. It concludes with a description of the method of evaluating different decision-making approaches

3.1. The Program

The blackjack simulator for this experiment was written in Java and uses object-oriented design to provide a simple, reliable, and easily upgradeable tool for experimentation. The simulator consists of two halves, the table and the players. Players can be organized into teams to support genetic learning. This section describes the simulator, referred to as the table, and the players and teams created to work within the simulator.

3.1.1. The Table

The table refers to the half of the program that manages the simulation of blackjack itself, and is a nested collection of the four objects, the table, the deck, the hand, and the card. These are illustrated in Figure 3.1.

The table runs the game and serves as a container for the rest of the objects. It possesses a deck of cards, an array of players taking part in the current game, and an array of hands, one for each player. The key function of the object is play(), which runs one whole hand of blackjack for the players in it. During a play operation, the table will use the call() function on each player as their turn arises. The inputs of the function are the current hands of all the players at the table and the dealer's displayed card, while the output is the action the player in question would like to take given that information.

The deck consists of an array of cards and a counter which points to the next card to be drawn. When a new deck object is created, it is given a number that denotes how many sets of 52 cards are to be included in the array. The key functions of the object are shuffle(), which randomizes the order of the cards, and deal(), which returns whatever card object is next in the array and increments the counter.

The hand consists of an array of cards that were dealt to it from a deck. The key functions of the object are deal(), which adds a new card to the hand, and score(), which calculates the value of the hand. The score() function assumes any aces in the hand to have a value of 11 unless that puts the score over 21, in which case it assumes a value of 1.

The card is simple object that contains a suit (clubs, spades, diamonds, hearts) and a value (ace, 2 through 10, jack, queen, king).

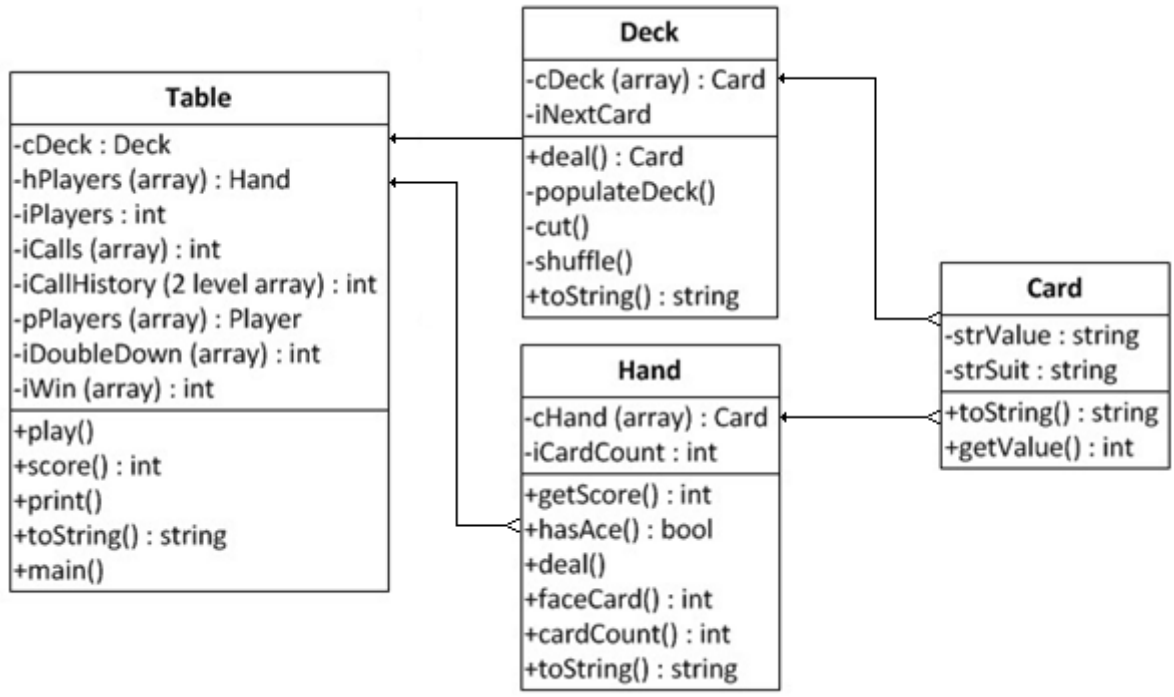

Figure 3.1: Core Classes for Blackjack Simulator: The Table

3.1.2. The Player

In the second half of the program, as depicted in Figure 3.2, all player classes inherit from a single Player superclass that defines all the critical functions that a player must have. This design is intended to take advantage of polymorphism. Since all players inherit from the Player superclass, the Table is able to treat them all equally and uniformly, so that no player is given special treatment. New types of players can be implemented and used without modifying the Table half of the program. The key function of all players is call(), which is called on a player by the table and supplies the players with the current information regarding the circumstances of the game and requests that it selects an action to take for this round. The selected action is carried back to the table as the return value of the call() function. Player also contains getScore() and updateScore(), both minor functions that either return or modify the player's current score.

Genetic learning players include an additional constructor method that takes in two existing players of that type and uses them to generate a new player using whatever strategy is defined for that type of player. The various types of players will be described in more detail in Section 3.2.

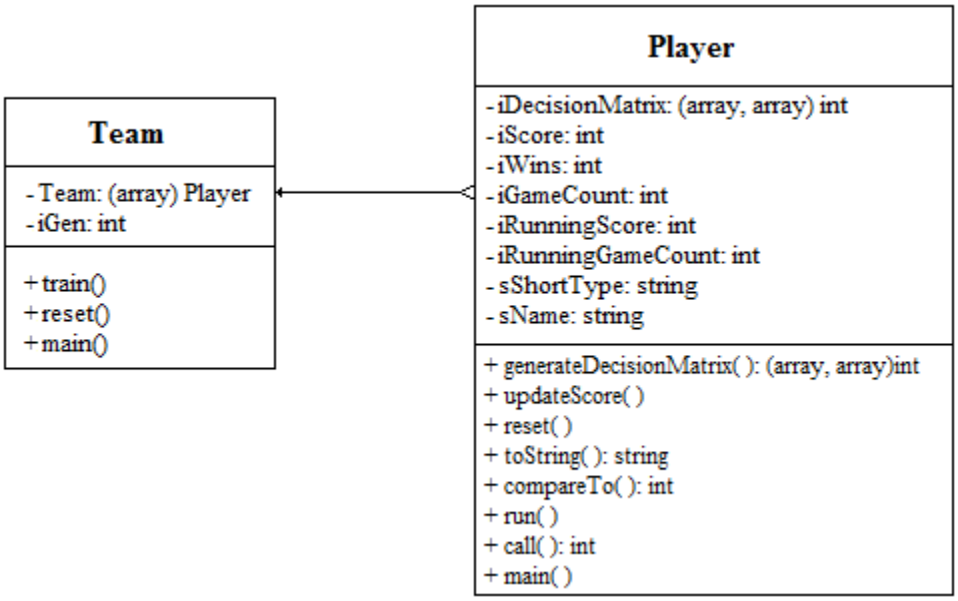

Figure 3.2: Core Classes for Blackjack Simulator: The Players

3.1.3. The Team

The element that allows genetic learning players to evolve is the existence of a team of players. Each team is designed for a specific type of player. In general, each team consists of an array of players as well as the functions required to accommodate the algorithm described in Section 3.3. Each team also includes a static "main" method designed to automate the training process.

3.2. The Player Types

With the testing environment complete, the next step was to define the player learning types that would be tested within it. Eight types were designed, four rule-based learners and four genetic learners. The rule-based learning types were divided into long-established dealer rules and expert rules. The genetic learning teams were standard genetic learning, genetic learning with fuzzy logic, genetic learning gene-by-gene, and genetic learning with a head start.

3.2.1. Dealer Rules

Of all the players in a game of blackjack, the dealer is the one with the least freedom. The choice of whether to hit or stand is only based on whether the current total of their hand is less than or greater than 17. If the value is less than 17, they have to hit, and if it is greater than 17, they have to stand. There are two versions of this rule set, which differ in how they handle the case where the value is exactly 17. One version, where the dealer must stand on a 17, is referred to as the S17 rule. The other version, where the dealer must hit on the value, is referred to as the H17 rule [1]. Either way, this approach is a rule-based AI at its most basic: a single question with a simple answer. While both H17 and S17 rule sets were employed as players during this research, the dealer always used S17 rules.

3.2.2. Expert Rules

While the dealer rules are simple, they are perhaps too simple to be useful for a player. For one thing, they do not take the dealer's card into account while deriving their action. For another, they do not make use of the additional options available to the player. Fortunately, there are general strategies known as "expert rules" available to help the amateur gambler.

Unlike the dealer rules, which operate on a single question, expert rules use a decision matrix, or a table of potential courses of action. Using a combination of the value of the player's hand and the dealer's displayed card, one can find a suggested course of action for the given situation. Since a player cannot have a value lower than 4 (a pair of twos) and does not have any viable alternatives with a value of 21 or greater, this means that there are 17 possible player scores and 10 possible dealer card values (the ace and nine number cards, with all jacks, queens, and kings treated as 10). As only one card is showing at this point, the value of an ace cannot yet be ascertained and is treated simply as an ace rather than a 1 or an 11.

The decision matrix is represented as a two dimensional array of integers with each suggested course of action given a value between 0 and 4: "Surrender", "Stand", "Hit", "Double down (Hit)", and "Double down (Stand)", respectively. The matrix is populated using the suggestions offered by experts on various gambling advice websites. Two players were generated in this manner: "Wizard of Odds" [2] and "Rule of Thumb" [1]. The matrices used are shown in Tables 3.1 and 3.2.

The suggestions on these sites also included additional advice for specific combinations, such as the presence of an ace in the player's hand or when the player is originally dealt a pair of matching cards. Since the "Split" option was not implemented in this experiment, the second half of this additional advice was ignored. Special consideration for hands containing aces was

implemented in early tests, but resulted in inferior performance and was thus abandoned during further development.

Table 3.1: Wizard of Odds Decision Matrix [2]

Player Score \ Dealer's Displayed Card	2	3	4	5	6	7	8	9	10	A
4	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
5	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
6	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
7	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
8	Hit	Hit	Hit	Double Down	Double Down	Hit	Hit	Hit	Hit	Hit
9	Double Down	Double Down	Double Down	Double Down	Double Down	Hit	Hit	Hit	Hit	Hit
10	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Hit	Hit
11	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down
12	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
13	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
14	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
15	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
16	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Surrender	Surrender
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

Table 3.2: Beating Bonuses Decision Matrix [1]

Player Score \ Dealer's Displayed Card	2	3	4	5	6	7	8	9	10	A
4	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
5	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
6	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
7	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
8	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit	Hit
9	Hit	Double Down	Double Down	Double Down	Double Down	Hit	Hit	Hit	Hit	Hit
10	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Hit	Hit
11	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Double Down	Hit
12	Hit	Hit	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
13	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
14	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Hit	Hit
15	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Hit	Surrender	Hit
16	Stand	Stand	Stand	Stand	Stand	Hit	Hit	Surrender	Surrender	Surrender
17	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
18	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
19	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand
20	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand	Stand

3.2.3. Standard Genetic Learning

Players used genetic learning to generate new decision matrices, in the hopes of finding an improved solution through repeated trial and error. Training for this team began by generating a number of new players, each with a different set of random values in their decision matrix. Each

player then played through a number of hands of blackjack. The number of hands played has a tradeoff between speed and performance. The fewer hands played, the faster each generation is completed, but random chance has a larger impact on the results. It takes significantly longer to play more hands per player, but random chance becomes less of a factor as more games are played. Here, a compromise between the two extremes was selected and one thousand hands per player were dealt. Once the round of games ends, the players were ranked by their average score. Only the top 40% were kept, while 40% were replaced via breeding and 20% were randomly generated.

The approach for breeding was based on that used when shuffling cards. Two of the top 40% were selected at random for each breeding. Bias towards higher scoring players was introduced by giving them larger shares of the selection pool. The lowest ranking player of the surviving population was given one share, and each rank gained one more share than the rank below it. Given a population of 100, 40 of which were kept each generation, this meant that the highest ranking player had 40 shares while the lowest had one, and thus the highest ranking player was forty times more likely to be selected for breeding. If the same player was selected to be both parents, the second parent is dropped and reselected until the two are not the same player.

With 17 potential player totals and 10 potential dealer card values, this means that there are 170 fields, or "genes", to consider. Ten random numbers between 1 and 65534 ($2^{16} - 2$) are generated and assigned to each potential dealer card value. 0 and 65535 were excluded in order to ensure that at least *one* gene from each parent was utilized. Then they are run through the following algorithm, with *n* being the number generated:

For int $i = 0$; $i < 17$; i++:

a) If n is odd, the value from the stronger parent is used when the player total = $i + 4$ and $n = (n - 1)/2$

b) Else (if n is even), the value from the weaker parent is used when the player total = $i + 4$ and $n = n/2$

This new generation of players is then run through another set of hands, culled, and repopulated in the same way. This happens for a set number of generations, ideally generating the best courses of action through the application of genetic learning.

3.2.4. Genetic Learning with Fuzzy Logic

Modifying the basic genetic learning team to implement fuzzy logic required the addition of a third dimension to the decision matrix used by the expert rules players and genetic learning teams. As a result, rather than returning a single answer when provided with the value of the player's current hand and the dealer's displayed card, this matrix returned an array of five numbers, with values between 1 and 5, each representing the proportional likelihood of a course of action being suggested: Surrender, Stand, Hit, Double down (Hit), and Double down (Stand), respectively. An example of such an array would be [1, 3, 5, 4, 2]. In this case, the player in question would have a 1 in 15 chance, roughly 7%, of selecting Surrender, the course of action assigned to the first number. On the other hand, the player would have a 5 in 15 chance, or roughly 33%, of selecting Hit, the third course of action in the list.

In order to accommodate this third dimension in the decision matrix, modifications were made to this team's breeding algorithm. The decision was made to treat each of the five numbers in the new array as a gene rather than the array as a whole. Future variants of this team could explore the effectiveness of other design choices, including treating the whole array as a single

gene and allowing the likelihood for a course of action to be a larger range of values such as zero or numbers greater than 5, to better define the likelihood of that action being the best choice.

3.2.5. Gene-by-Gene Genetic Learning

The approaches described in Section 3.2.3 and 3.2.4 had very poor initial results. The use of large randomly generated integers to provide direction as to which parent would provide each gene was judged the most likely problem. Although each number in a given range should have had an equal probability of occurring, it seemed that Java's pseudo-random number generation functionality was not handling such large ranges appropriately. Numbers on the upper extremes of these ranges appeared to be extremely rare, which introduced a parental bias in certain genes and skewed the results.

In order to address this concern, a new genetic learning team was created to compete with the existing players. This "gene-by-gene" design determined inheritance for each gene individually based on a random number between 0 and 1, with a 0 inheriting from the first parent and a 1 inheriting from the second. This new team's performance was used to gauge what impact, if any, this design choice and the limitations of Java random number generation had on the ultimate results.

3.2.6. Genetic Learning with a Head Start

Another concern raised during initial experiments using the genetic learning team was the slow pace of improvement. Although the team did provide better players on average over time, the quality of these players was far below that of the most basic rule-based player and the rate of improvement suggested it would likely not reach such a point within the 25 generations allotted for the training. This was by no means unexpected; the rules outlined within the Expert Rules teams, and even the simpler Dealer Rules teams, had evolved through decades of experience. A

system based on random numbers alone may be unable to derive a player able to rival experts. However, one of the questions this work asked was whether genetic learning *could* derive a better solution to the game.

As with the question of modes of breeding, this concern was explored through the generation of a new team that would take part in the competitions. This team was identical to the original genetic learning team except that one player in this team's population was generated using the decision matrix of one of the Expert Rules teams. The expectation was that this team could answer the question: Assuming genetic learning could, given time, generate a player as effective as the Wizard of Odds player, would it be able to then evolve one that was even better?

3.2.7. Genetic Learning With Memory

A concept that was explored in the later stages of this research was that of implementing memory within a genetic learning team. In the previous genetic learning teams genes were selected at random, with the child having an equal chance of inheriting a given gene from either parent. No preference was given to tried and true genes over those that had performed poorly as long as the parents' overall scores were sufficient to reach the top 40%. Introducing a bias for particular genes was expected to provide the best results of any genetic learning team.

In order to implement this team, three new 17x10 matrices were added to the new player. One of these, the "Trigger Count" matrix, would record how many times a particular gene had triggered, such as holding a hand worth 15 while the dealer is showing an ace. The second, the "Win Count" matrix, would hold how often that gene had been involved in a winning hand. A third temporary matrix, whose values were reset to zero after each hand, kept track of which genes were triggered in a given game so that the other two could be updated accurately.

When a call() function was called on this player, the first thing this player would do was add one to the correct cell in the temporary matrix. This allows the same gene to trigger multiple times in a hand, as it is possible to have a 5 and an ace (a value of 16) and then draw a jack (also a value of 16). The player then looks up the correct cell in the decision matrix and declares its action. Once the hand is over, the player then goes through the temporary matrix. For every cell with a non-zero value, the player then adds that value to the TriggerCount matrix and, if the hand was a winner, to the WinCount matrix as well. Once the appropriate cells are updated in the other matrices, the cell in the temporary matrix is reset to zero for the next game.

When called to breed, this team will use these new matrices to introduce bias into decisions of inheritance. Both parents' genes are rated using the equation:

$$Gene\ Rating = \begin{cases} 25, & If\ TriggerCount = 0 \\ \dfrac{100\ x\ WinCount}{TriggerCount}, & else. \end{cases}$$

If one gene rates higher than the other, that one is selected; otherwise, the gene is selected at random. The default value of 25 is meant to provide a threshold at which a poorly performing gene can be supplanted by an untested gene. Without it, a gene that succeeded once in a million games would be rated higher than a fresh gene. With it, a tested gene has to win at least a quarter of the time in order to have a chance of being inherited. The WinCount and TriggerCount values of the inherited gene are placed in the appropriate matrices of the child, so that a gene is rated based on its entire lifespan and not merely that of the player.

3.3. The Algorithms

There were three primary functions this project needed to accomplish. The first was to accurately simulate a hand of blackjack using interchangeable players. The second was to

effectively operate and train a genetic learning team. The third was to gauge and rank each approach. The algorithms are presented here.

3.3.1. Playing a Hand of Blackjack

1) An array of players is created. Any type of player can be used, as long as the player's class extends the Player superclass. The first player in the array (player zero) is the dealer and must be a "dealer rules" style player. All players can be generated via a constructor, but team-based players created in such a way will simply be randomly generated, and should usually be drawn from a Team instead. Player zero is the dealer and player one is the player to be tested. By default, the dealer uses S17 dealer rules, but can be set to use the H17 variant.
2) A new Table object is created using this array of players. It creates a new deck of cards (comprised of however many decks are desired), shuffles the deck, and then deals a hand of two cards to each player on the table.
3) Each player is then asked, in turn, what action they will take, with each available action designated a number. For example, a response of "1" would mean a decision to stand.
    a. The player is allowed to see their own hand, the hands of any other player at the table, and the displayed card of the dealer before selecting an action.
    b. If a player chooses to hit or double down they are dealt an additional card for their hand.
    c. If a player chooses to surrender, stand, double down, or has a hand with a value exceeding 20, they are removed from the rotation for the rest of the game.
    d. If a player obtains a hand with a value of exactly 21, they automatically stand.
    e. Step 3 is repeated until all players are removed from the rotation.

4) Each player is scored for the game. Players start with zero points and have their score reset after every generation.
    a. If the player opted to surrender, the player loses 500 points.
    b. If the player and dealer score the same, but do not surpass 21, the player gains nothing.
    c. If the player hits and goes over 21, the player loses 1000 points regardless of the dealer's score.
    d. If the player starts with 21 (an ace and either a ten or a face card) and the dealer does not, the player gains 1500 points. If the dealer starts with 21 as well, the player gets nothing.
    e. If the player opted to double down (only available in the first round of a hand), scores 21 or less, and the dealer either scored lower or went over 21, then the player gains 2000 points.
    f. If the player opted to double down and either goes over 21 or scores less than the dealer (who also scored 21 or less), the player loses 2000 points.
    g. If the player opted to stand and scores higher than the dealer without exceeding 21, the player gains 1000 points.
    h. If the player opted to stand, but scores less than the dealer (who also scored 21 or less), the player loses 1000 points.
5) Each player keeps a running tally of their total score and number of games played and will provide an average of their running score upon request, until they are directed to reset their scores.

3.3.2. Training a Genetic Learning Team

1) The team creates an array of players large enough to accommodate the population and generates new players of the selected type to fill the array. Some teams may have special rules that demand that one or more of the initial population be created through specific means, such as the Genetic Learning with Head Start team.
2) The population is initialized and ranked.
    a. The scores for each player are reset.
    b. Each player plays 1,000 hands.
    c. The team is re-sorted according to their new scores, with the highest score winning the first slot.
3) The population is culled and repopulated.
    a. The team is saved as a serialized file as a measure against premature termination of the program.
    b. The top 40% of players are kept.
    c. The next 40% of players are generated via breeding, a process which is specific to each type of team.
    d. The last 20% of players are randomly generated.
4) The new population is then tested and ranked.
    a. The scores for each player are reset.
    b. Each player plays 1,000 hands.
    c. The team is re-sorted according to their new scores, with the highest score winning the first slot.

5) The two highest scoring players from each team are selected for an evaluation match, detailed in Section 3.3.3.
6) The number of generations is incremented by one. If the number of generations is less than 25 (250 in the Extended Training Test), return to step 4.

3.3.3. Testing and Scoring

1) The two top ranking players from each genetic learning team are selected for an evaluation match. For rule-based approaches, two players that use the given rule set are used and no changes are expected across generations.
2) Each player plays 100,000 hands of blackjack. These scores are averaged to determine a player's final score for the generation, which is recorded in the team's log.
3) The final scores from step 2 are then averaged to determine each team's overall performance. This is the metric used to evaluate most of the tests described in Chapter 4.

3.4. Evaluating the Approaches

During the implementation of the simulator and its players, certain design choices posed additional questions to be addressed. The first of these was the question of how large an impact the genes in the initial population had when training genetic learning teams. Three tests, randomized, initialized, and H17-initialized, were devised to test this by altering how the initial populations were defined. The second question was regarding what number of hands each player should play in a generation, for which the Volatility test was created. The third question, whether allowing more training generations would provide better results, resulted in the Extended Training test. Additionally, questions were raised as to whether the use of multiple decks in a game or a dealer applying H17 dealer rules would affect each team's results, leading to the

creation of the Multideck and H17 Rules tests, respectively. Table 3.3 shows the tests and the altered settings implemented in each, for the algorithms described in Section 3.3.

Table 3.3: Summary of Tests

Test Name	Teams Involved	Gene Initialization	No. of Hands per Report	Dealer Type	No. of Decks	No. of Generations
Randomized Test	All	Random	100,000	S17 Rules	1	25
Initialized Test	All	S17 Rules	100,000	S17 Rules	1	25
H17-Initialized Test	All	H17 Rules	100,000	S17 Rules	1	25
Volatility Test	Memory GL	Random	100 and 100,000	S17 Rules	1	25
Extended Training Test	Memory GL	Random	100,000	S17 Rules	1	250
Multideck Test	All	Random	100,000	S17 Rules	6	25
H17 Rules Test	All	Random	100,000	H17 Rules	1	25

The standard bet used for this game is 1,000 points. 1,000 points are gained if the player wins a hand, and 1,000 points are lost when a hand is lost. If the player achieves a score of 21 from dealt cards alone (an ace and either a ten or a face card), the player gains 1,500 points. If the player chooses to surrender, they only lose 500 points. If the player chooses to double down, they effectively double their bet and gain or lose 2,000 depending on the next card they are dealt.

Players begin each generation with zero points and win or lose points with every hand of blackjack played, with the average score of these hands being used to rank players in genetic learning teams. Players are allowed to go below a score of zero and continue playing. After each generation, an “exhibition match” is then played, taking the two highest ranking players of each type from the last generation and running both through 100,000 hands of blackjack. The scores of these hands are then averaged and stored for that team, along with the generation number and

the method used (“initialization”, “breeding”, “mutation”). With the exception of the Volatility test, the results of these two representatives are then averaged together to determine the overall score of the player type for that generation. In the Volatility test, the scores of the representatives are sorted by the higher score of a generation and the lower score of a generation, and the two sets of scores are plotted separately. The Extended Training test includes the maximum, minimum, and average of the Wizard of Odds Expert Rules performance. As the Head Start team used a player with Wizard of Odds genetics in its initial population, these statistics can provide benchmarks for determining any improvement in the Head Start team.

Chapter 4 - Experiments and Results

This chapter presents and analyzes the results of the eight tests outlined in Section 3.4. These are the randomized test, initialized test, H17-initialized test, volatility test, extended training test, multideck test, H17 rules test, and variation test. The first six tests assess various Player and Team configurations while the seventh test evaluates changes in the dealer and the eighth test makes no changes to the configuration but assesses the impact of variation in the learning process.

4.1. The Randomized Test

In this test, the genetic learning teams were created using completely randomized settings for their original populations. The only exception to this was the first player of the Head Start genetic learning team, which remained initialized to the same values as a Beating Bonuses player. For each of 25 generations, with a generation consisting of 1000 hands per player, the two top ranked members of each genetic learning team and two each of the rule-based teams are run through 100,000 hands of blackjack. Each game is played as a single player against a dealer using S17 rules and a single deck of cards.

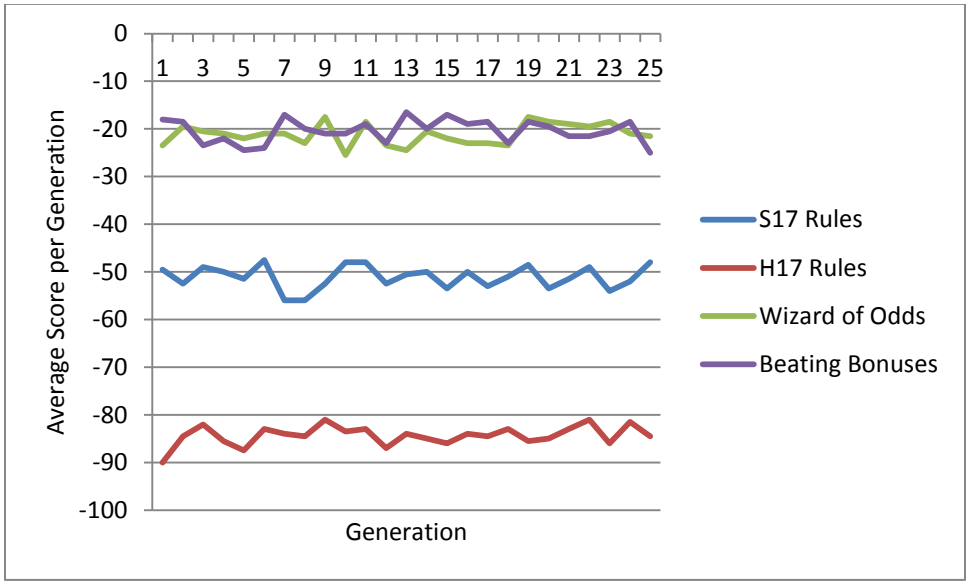

Figure 4.1: Rule-based Team Results of Randomized Test

The first and most immediately visible aspect of this test's results is also the reason it is best displayed in two graphs (Figures 4.1 and 4.2), and that is the vast difference in performance between the genetic learning and rule-based players in this test. While the worst of the rule-based players remained relatively stable around the -80 mark, only two of the genetic learning teams managed to improve above -100. The second aspect is that, while the Fuzzy GL team did improve its original score by more than 10% over the course of the test, it did not improve as quickly as the other teams.

As seen in Figure 4.1, the expert rules teams are consistently capable performers, both maintaining a -20 without either displaying a clear advantage. Of the dealer rules teams, the S17 team scored a reliable 30 to 40 points higher than its H17 counterpart. These results were consistent across other tests, and will not be discussed in further sections. Risk taking was not rewarded in this context.

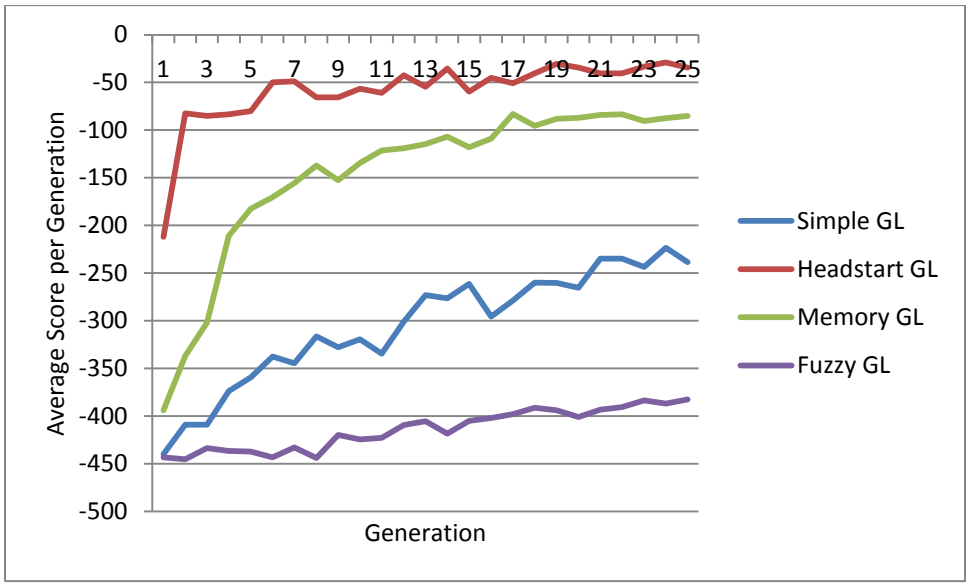

Figure 4.2: Genetic Learning Team Results of Randomized Test

Figure 4.2 provides an interesting set of results about genetic learning. Initializing GL players using random values resulted in notably poor results early on, with scores starting around -400. With a single Wizard of Odds player in its roster, the Headstart GL team was able to

average that player's strong score with that of a score as bad as that of any other team, resulting in an average of roughly half of the rest of the pack. Despite these rocky beginnings, or more accurately because of them, this experiment gave each team ample opportunity to grow. The Headstart GL team started strong and held the top rank throughout the experiment, ending up only 10 points below the expert rules teams. The Memory GL team, despite starting at -394, learned very quickly. It ended up with scores in the -80s, putting it on par with the H17 dealer rules. The Simple GL team, while not able to compete with the rule-based teams, improved its score by more than 45% over the course of the experiment. The Fuzzy GL team, on the other hand, improved very little over the same stretch of time, increasing its score by only 14%.

4.2. The Initialized Test

In the Initialized Test, the genetic learning teams were created using the settings for the S17 dealer rules for their initial population, with the exception of the first player of the head start team, which was initialized using the Wizard of Odds expert rules. The Fuzzy GL team was set to 100% for hit or stand according to S17 rules, removing the fuzzy nature from the initial population but not players generated at a later point. As with the Randomization test, this test was run for 25 generations, with the two highest ranking representatives from each team playing 100,000 hands each generation. The settings of the blackjack game are identical as well: single player, against a S17 dealer using a single deck.

By initializing the genetic learning teams to use S17 rules, the goal was to start all the teams with effective (if not optimized) genes. The teams started from a better position with the expectation that they would improve further than during the Randomized test, but were expected to grow at a slower rate, as the players have less room to improve.

With the exception of the Headstart GL team, the results shown in Figures 4.3 and 4.4 were not quite as anticipated. The other three GL teams remained quite stable at the -50 mark, remaining on par with their S17 origins, but not improving much beyond that point. For its part, the Headstart GL team did improve, but only slightly above what it achieved in the Randomized test (Figure 4.2).

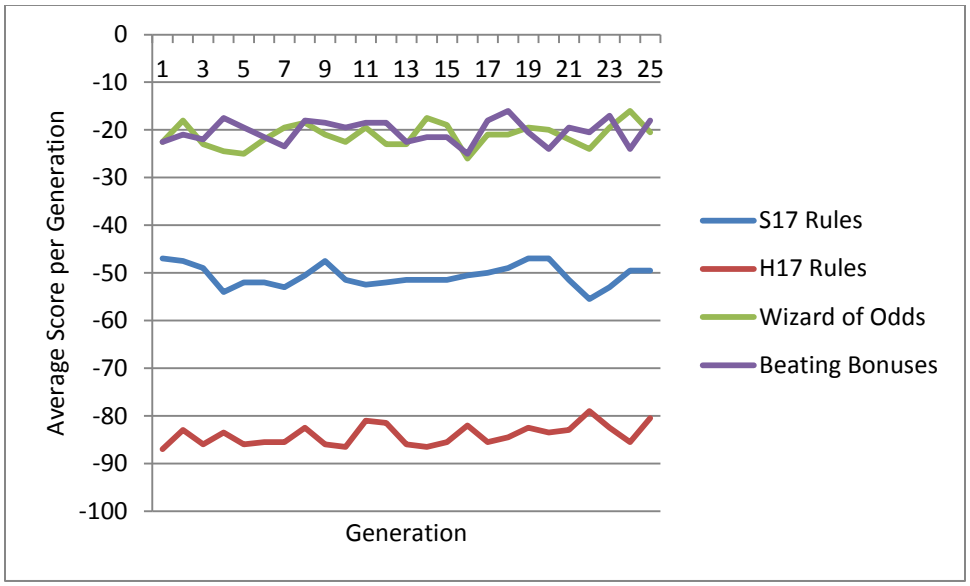

Figure 4.3: Rule-Based Team Results of Initialized Test

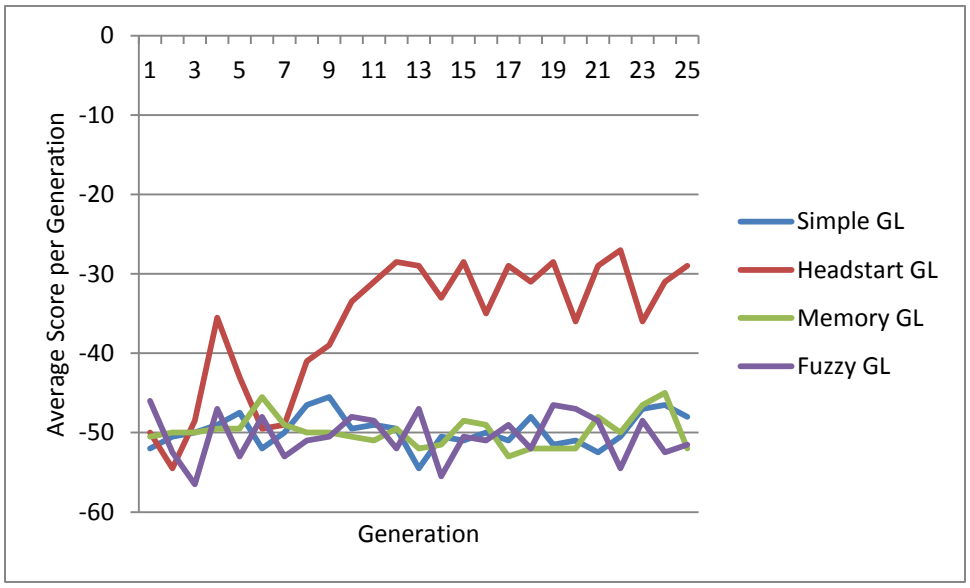

Figure 4.4: Genetic Learning Team Results of Initialized Test

While these results were not expected, they are explainable. As a team's initial genetics improve, the number of potentially superior gene combinations decreases. Starting with a solid set of genetics, such as S17 Dealer Rules, means it is substantially harder to generate a better player, and as a result many more generations are required to make even a minor improvement, if indeed any notable improvement occurs. Such improvement did not occur in this test. Whether an increased mutation rate or a higher number of generations could have affected the expected results is unknown, and was not explored in this work.

4.3. The H17-Initialized Test

A reasonable comparison for the initialization test is the use of H17 dealer rules rather than S17. This addresses the question of whether the use of S17 dealer rules to initialize the genetic learning teams was going too far, and whether a more noticeable growth rate could be obtained using a poorer performing, but valid, rule-set such as H17. Rather than run a full test using all 8 teams to explore this question, only the gene-by-gene genetic learning team was given the new rule-set and run through 25 generations. As with the Initialized test, the two top ranking players of each generation were run through 100,000 hands of blackjack, using the same settings.

With more room to grow, the team was expected to exhibit a better growth rate during this experiment. Since even H17 rules outperformed the simple genetic learning team during the Randomized test, however, it was expected that it would have limited room for improvement, similar to that seen in the Initialized Test, and exhibit negligible improvement.

The results of this experiment, as seen in Figure 4.5, are similar to those observed in Section 4.2, and for the same reasons. The Headstart GL team once again managed to achieve scores in the -30s, while the other teams never evolved beyond the expected range of their initial genes. While this was not what was hoped for, it is worth noting that the only GL team that

outperformed the dealer rules was the one to display improvement in both initialized tests. This consistency in final performance suggests that there is an upper bound to a given team's performance, and that the initial genes do not alter its ability to evolve beyond what they can naturally derive.

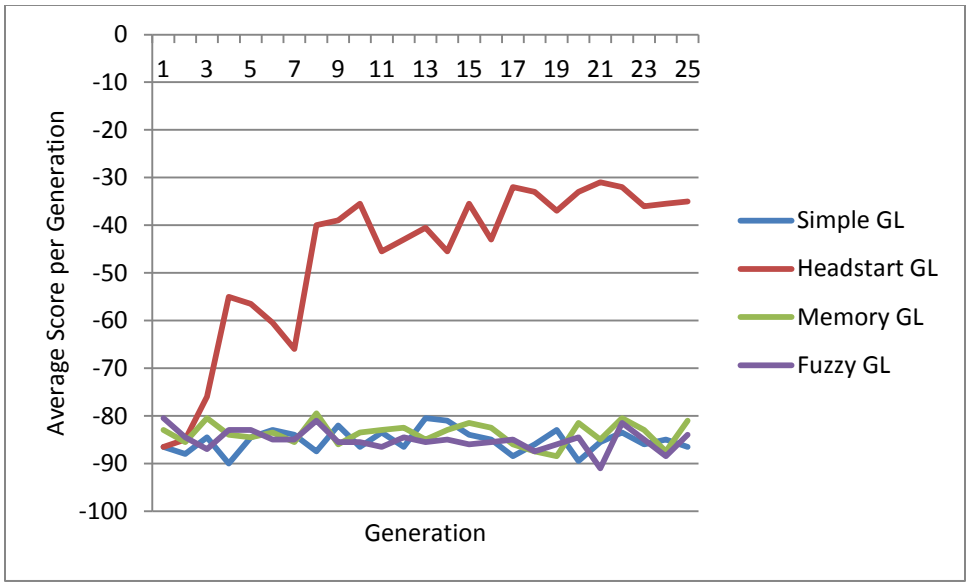

Figure 4.5: Genetic Learning Team Results of H17-Initialized Test

4.4. The Volatility Test

The Volatility Test explored the impact of the number of hands played on our ability to gauge the performance of a player, be it for ranking players or comparing team results. The question is whether these tests could be run with fewer hands, reducing the time required to evaluate each generation. Teams were trained using the same initial conditions in the Randomized Test (Figure 4.2). During evaluation, representatives of each team only played 100 hands rather than the 100,000 used in other tests. Additionally, the metric we examined for this test was the difference between the scores of the top two players rather than the average of their scores. The assumption was that the larger the difference between the scores over many generations, the lower the reliability of the results.

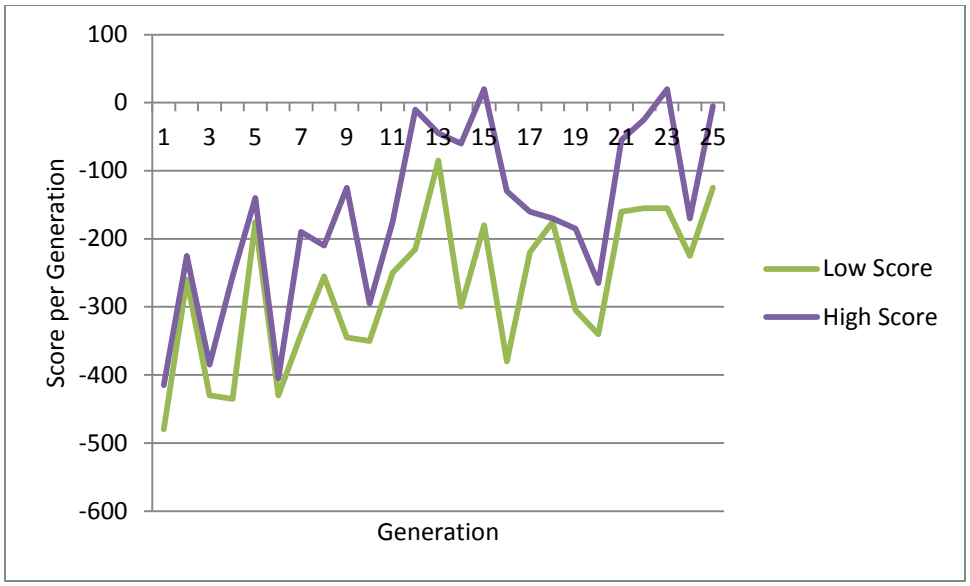

Figure 4.6: Using 100 Hands per Generation with Memory GL

The results of the Volatility experiment gave the test its name. The scores were all over the board for all eight teams. Rather than attempt to put all of the results in one graph or include a graph from each, the graphs for this experiment show the Memory GL team as a representative. In order to demonstrate the difference between two runs of the experiment, Figures 4.6 and 4.7 show the higher and lower scores for each generation as separate lines rather than averaging the scores to create a single line. It should be noted that, despite the fact that the same players were used in the 100 Hand and 100,000 Hand evaluations, the 100 Hand evaluation final scores are an average of 72 points higher than those of its 100,000 Hand counterpart, -65 as opposed to -137. Given the theory precipitating this test, that fewer hands during evaluation result in less accuracy, this is not a surprise. Indeed, the 100,000 Hand evaluation never went as low as 200 again after generation 7, while the 100 Hand went below that threshold as late as generation 24. It is interesting to note that the final score for the 100,000 hand evaluation is -137, which is noticeably worse than the -85 the team achieved in Figure 4.2, despite starting from almost the same point (-396.5 vs -394 respectively), a deviation that inspired the Variation Test detailed in Section 4.8. Figure 4.8 displays the difference between the high and low scores for both 100

Hand and 100,000 Hand evaluations, calculated as (High Score – Low Score). The difference between scores for the 100 Hand run ranged from nearly zero to 250, while the 100,000 Hand run ranged from zero to just over 50. The results of this experiment suggest that, despite requiring more time to achieve, running more hands per generation provides results with less variation and thus greater accuracy. Figures 4.7 and 4.8 also show that 100,000 hands result in a

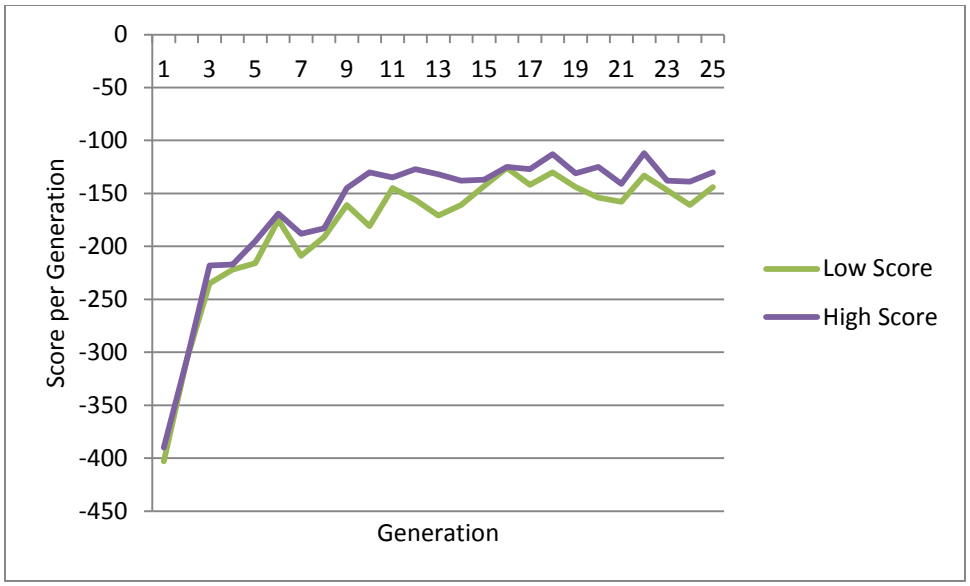

Figure 4.7: Using 100,000 Hands per Generation with Memory GL

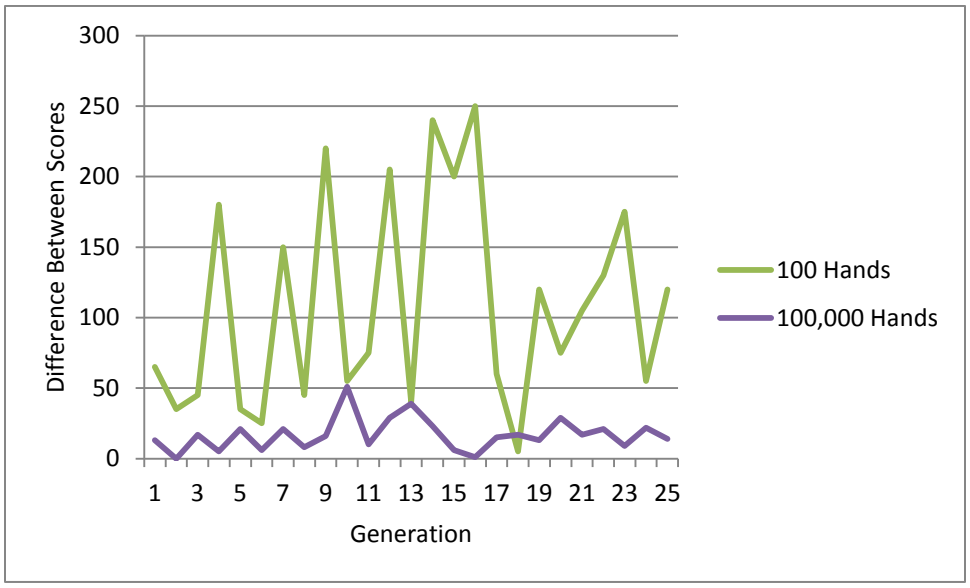

Figure 4.8: Difference between High and Low Scores by Generation with Memory GL

satisfactory evaluation of the performance of AI approaches, as the difference between the two scores were closer to zero.

These results also suggest an interesting aspect about blackjack; when using set strategies, it is possible to either win or lose big in the short run, but in the long run you always lose just a little bit. Even the best strategies available lost an average of twenty points per hand after playing 100,000 hands.

4.5. The Extended Training Test

Another question that was explored was whether going beyond 25 generations in training would result in better results than were already attained. To answer this question, a team using Memory GL players ran for 250 generations. With the exception of the number of generations to be run, the settings for this test were identical to those used in the Randomized Test (Figure 4.2).

Given the largely flat growth rate the teams demonstrated in the latter generations of the first two tests, it was doubted that 225 further generations would result in significantly improved performance. This experiment would be considered a positive success if the team finally succeeded in at least closing the gap between itself and the expert rules players.

The results of this experiment, shown in Figure 4.9, provide some corroboration for the theory offered in Section 4.3, that a given team will achieve roughly the same result regardless of its initial genes. This is because the Memory GL Team achieved the same scale of results by the 25^{th} generation as seen in the Randomized test, and did not improve in the 225 generations that followed, holding at a score of -90±10 throughout the extended testing, comparable to the values seen in Figures 4.2 and 4.5. This also suggests that adding additional generations to a run is less likely to provide an improved result.

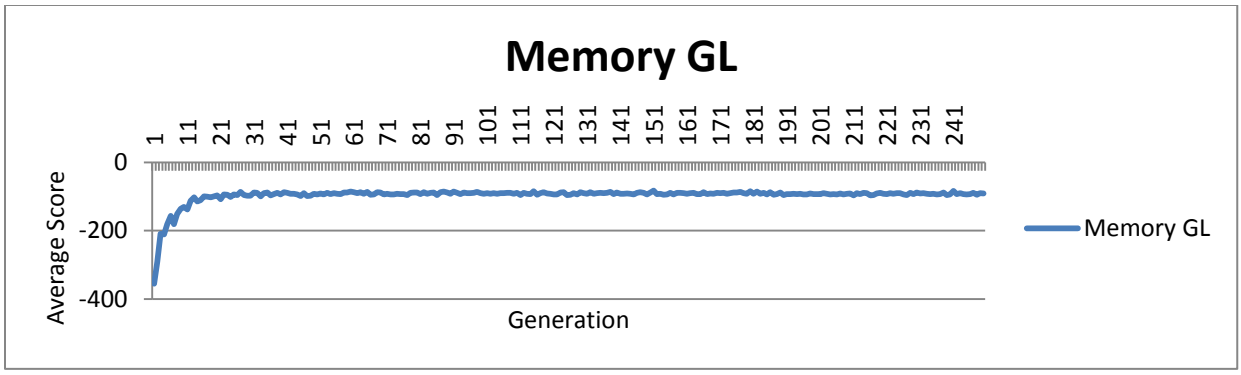

Figure 4.9: Extended Test of Memory GL Team (250 Generations)

4.6. The Multideck Test

Although the previous tests explored the various aspects of the various player designs and simulator settings, they did not address one element of the game. In the real world, blackjack is not played with a single deck of cards. In fact, the standard rules in American casinos call for eight decks [2]. The question must be asked: how would the performance of these various player differ in an eight deck game of blackjack? This is a question easily answered, as the design of the simulator allows for any number of decks. The remaining settings for this test are the same as those used in the Initialized Test.

The only notable difference between this experiment (results in Figures 4.10 and 4.11) and the Randomized Test (results in Figures 4.1 and 4.2) was the time required for it. Using multiple decks instead of just one resulted in the shuffle operations taking longer. Because the decks were reshuffled for every hand, the amount of extra time required for each generation was approximately tripled. As a result, a compromise of six decks was used to limit the extra time required while reducing the effect previously drawn cards have on the probability of what card

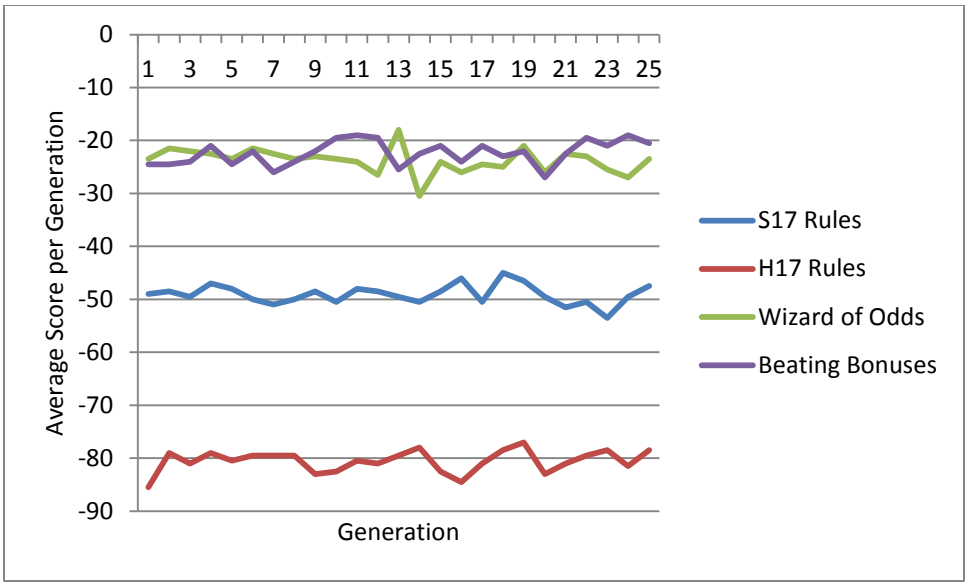

Figure 4.10: Rule-Based Team Results of Multideck Test

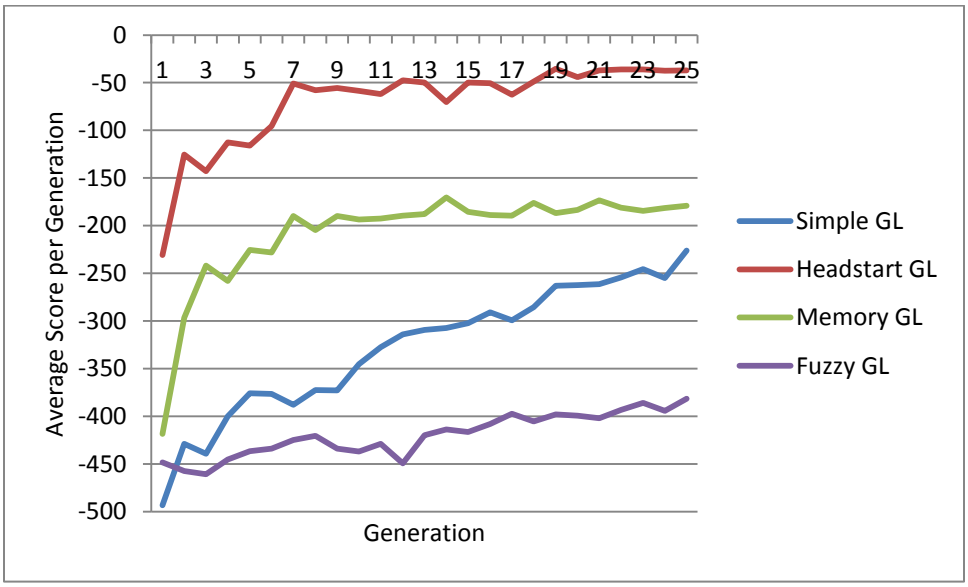

Figure 4.11: Genetic Learning Team Results of Multideck Test

would be drawn next. The actual results were similar to those seen in the Randomized test, though the Simple GL team started from a markedly worse position. These results suggest that using multiple decks to gauge performance in these experiments is not time efficient.

4.7. H17 Rules Test

A capability of the simulator that has not been explored in previous tests is the dealer's ability to use the H17 rules in games. In order to test how a change in dealer strategy could affect the players, the H17 Rules Test was run. With the exception of the dealer using H17 rules, this test used the same settings as the Randomized Test. It is worth noting that, according to Wizard of Odds, a dealer using H17 rules puts players at a slight disadvantage when compared to S17 rules [2], although genetic learning players using H17 did not perform better than S17 GL players.

As has been the case for many of these experiments, the GL teams achieved the same general level of success when left to their own devices. Although their performance was slightly weaker than in previous experiments, as seen in Figures 4.12 and 4.13, and this is most clearly seen through the previously consistent dealer rules teams, the shift was not more than ten to twenty points and the performance ranking of the teams remained much the same.

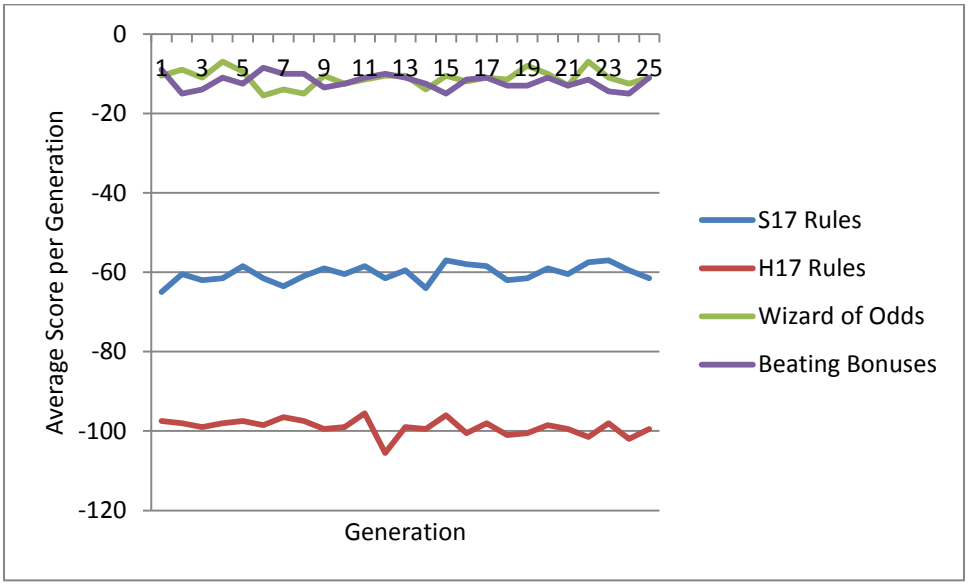

Figure 4.12: Rule-Based Team Results of H17 Rules Test

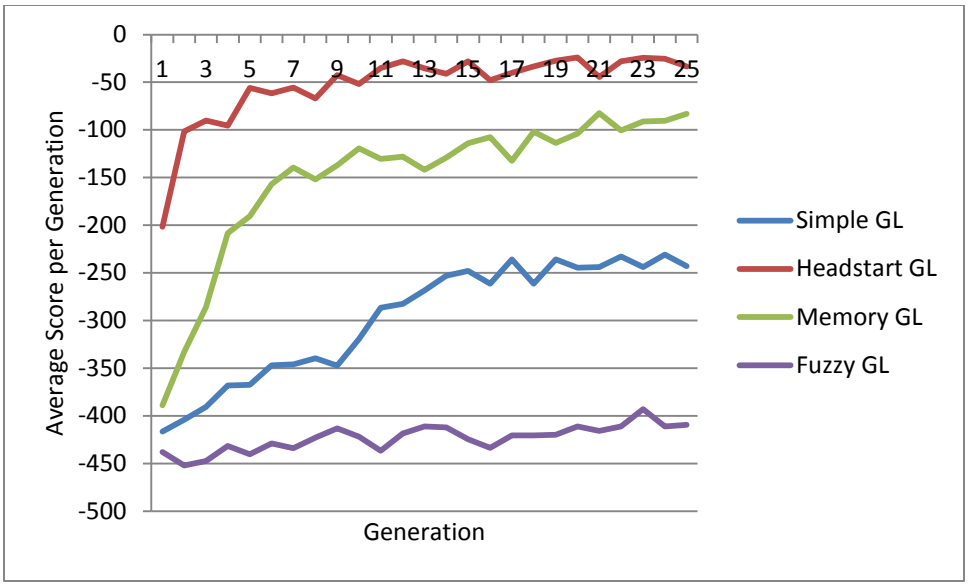

Figure 4.13: Genetic Learning Team Results of H17 Rules Test

4.8.Variation Test

While reviewing the results of the previous tests, another question arose that needed to be addressed: how reliably does a team reach a given final score? During the Randomized and Volatility Tests, the same team, with the same test settings, obtained final scores over fifty points apart. To answer this question, the Variation Test was devised, in which the Memory GL team was run through the Randomized Test from Section 4.1 an additional five times. In order to obtain a wider range of results and reduce the potential impact of limitations in Java's random number generator, each test was run on a different machine with the same memory and processor configuration and initiated at different times.

As seen in Figure 4.14, there is an eighty point difference between the highest final score and the lowest. Although this is a relatively large margin, all seven tests scored between the established scores for the Headstart and Simple GL teams, though there would be some overlap if similar margins were assumed for those teams. While the Volatility Test's results (Figure 4.12) remain the lowest of the seven, the distance between any individual test is not very large. This

suggests that, while the learning rate for a particular approach remains essentially the same, the randomness involved results in variable final outcomes.

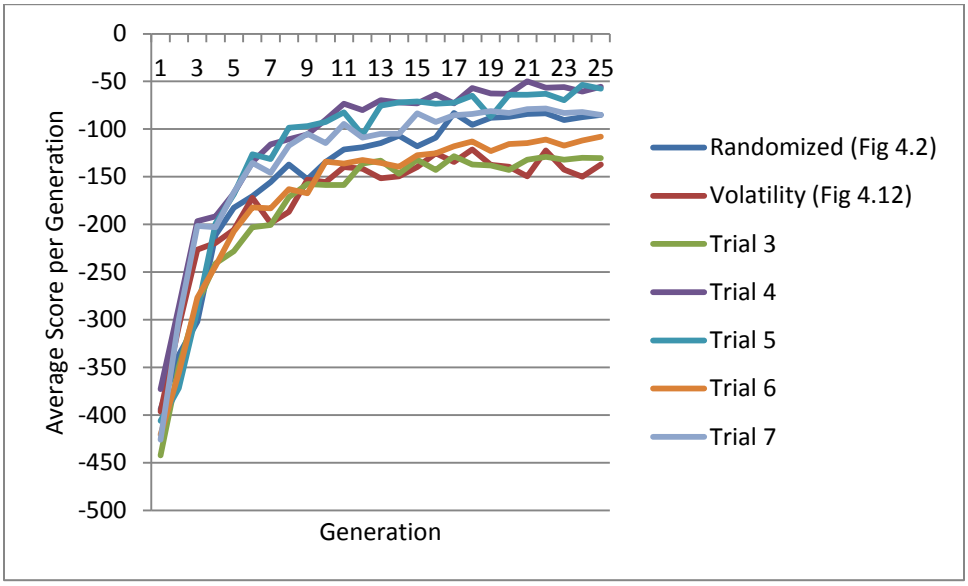

Figure 4.14: Seven GL Memory Trials with Randomized Initialization

4.9. Impact of Initial Settings

One consistent detail that appeared throughout these tests was each team tended to reach the same range for their average scores regardless of their starting initialization settings or changes in their testing variables. This can be seen through a comparison of all the tests for one approach,

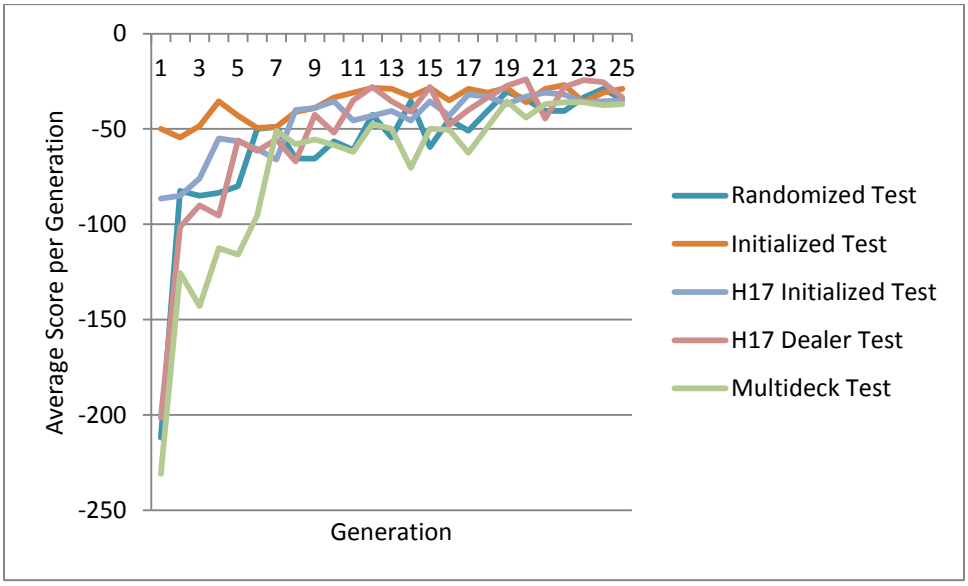

Figure 4.15: Headstart Team Progress Across Multiple Tests

the Headstart GL team, as seen in Figure 4.15. Although each test began from a different starting point, their ending scores ranged from -29 to -37, a difference of only 8 points.

4.10. Comparison to Previous Work

In order to build a comparison between this work and that of Curran and O'Riordan [30], one additional metric needed to be recorded in these experiments, as Curran and O'Riordan reported their findings based on the percentage of games won rather than the average number of points won or lost. An example of this work's results, those of the Randomized Test (seen in Figures 4.1 and 4.2), may be found in Figures 4.16 and 4.17. Three observations can be derived from this metric. The first is Curran and O'Riordan's results wound up in the same 40-45% range as seen in the majority of this work. The second is that, in terms of straight wins versus losses, the rule-based teams showed surprisingly little deviation, while the genetic learning teams were at best able to match their win rates while never surpassing them. Finally, the Memory GL Team in Figure 4.17 clearly wins more hands on average than the Headstart GL Team, while the Headstart GL Team ranked consistently higher in terms of average score in Figure 4.2. This suggests that, while the Memory GL Team managed to evolve better strategies to win a hand, it never managed to learn how to make effective use of the advanced actions available to a player, surrender and double down, in order to maximize the its scores. Thus, if any future work is done in this line of research, it is suggested that both metrics be examined, because both highlight different aspects of an AI's progress.

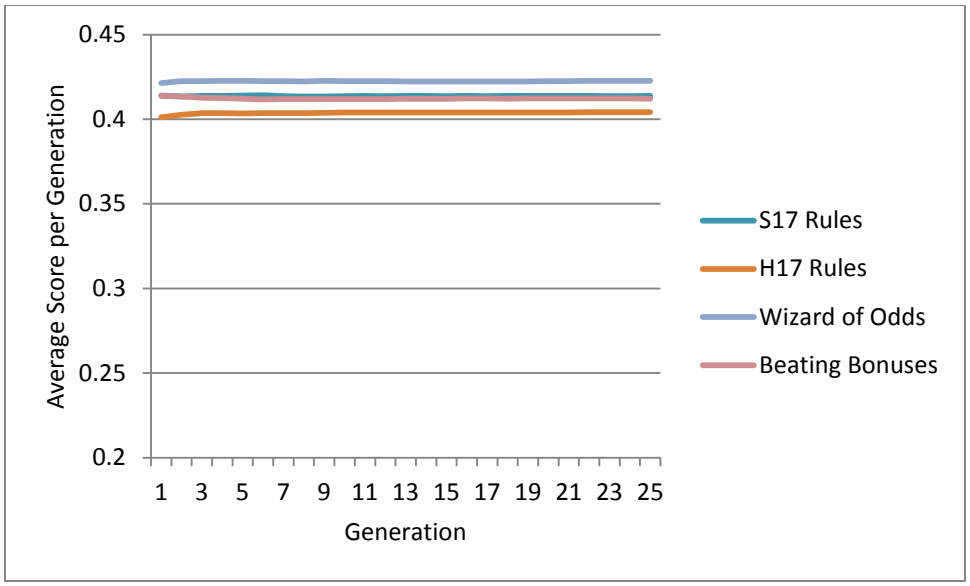

Figure 4.16: Rule-Based Team Win Percentage for Randomized Test

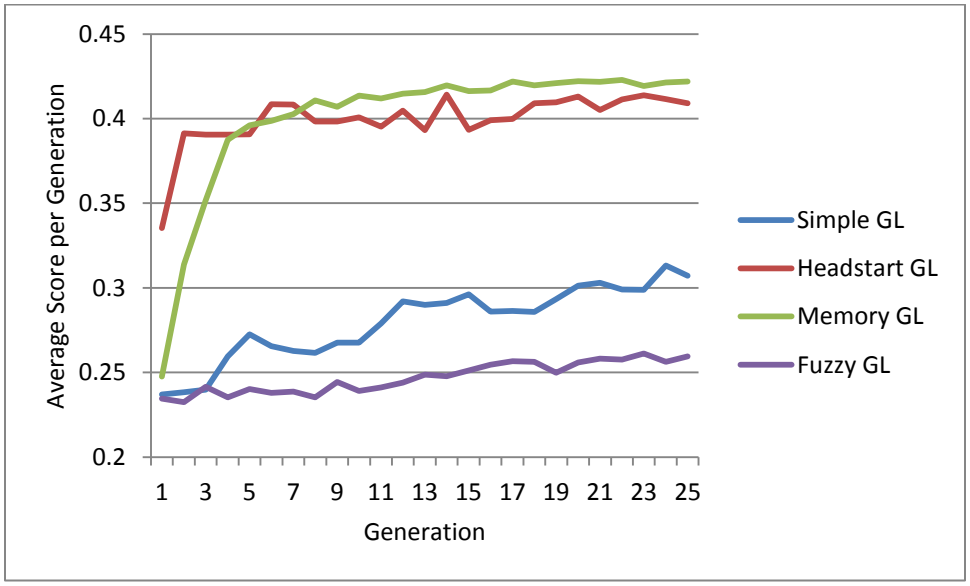

Figure 4.17: Genetic Learning Team Win Percentage for Randomized Test

Chapter 5 - Conclusions

The results of this work provide an interesting glimpse into the implementation and growth of an artificial intelligence agent based on genetic learning. The first observation is the apparent limits of genetic learning algorithms, as each team managed to achieve the same general results regardless of environmental settings such as deck size, dealer strategy, or even initial genetics. Approaches that start with superior solutions in their genes trumped the same approaches in terms of final performance. In the end, the conclusion that was drawn from these experiments was that the wrong variables were focused upon. Rather than exploring the effect of the various changes in how the game is played, it would appear that more could be learned by exploring the variables of how a genetic learning team operates its own evolution, such as how it manages its population or breeds new members.

In the pursuit of this work, several questions were addressed as to what impact evaluation design choices might make on the ultimate result. Although the modification of initial genes for genetic learning teams, as seen in the Initialized and H17 Initialized tests (Sections 4.2 and 4.3 respectively), did give them a better initial score, it had no appreciable impact on their final scores for each team. Indeed, if the initial genes allowed the team to outperform the team's standard scores, no additional growth was seen. The only case in which a team improved was the Headstart genetic learning team, which was never able to surpass the expected scores of the Wizard of Odds player that the team started with. The number of hands played during an evaluation round, on the other hand, had an impact on our ability to accurately gauge the performance of a player, with more hands giving more consistent results. Generations past the twenty-fifth provided little improvement, indicating that simply extending the duration of the test is not an effective way to refine the results. The use of multiple decks proved easy to implement

and had little impact on the results, but added immensely to the time required to run the test. Using the H17 dealer rules instead of the S17 rules did not influence the outcome much, either, with near identical results whether the dealer stood or hit on a hand of 17. In summary, the variations in the evaluation setup had different, though not large, effects on the results of the four rule-based learners and four genetic algorithm learners examined here.

While chance and circumstance did shape the various genetic learning players, it was the design of the players that ultimately had the most impact on their ultimate performance. With the exception of the initialized tests, where certain players were effectively tied, the rankings of the genetic learning teams never changed and time, which theoretically should have allowed the memory team to improve even further in the Extended Test, did not demonstrate the influence it possibly should have. So the lesson learned becomes this: dumb luck and patience cannot supplant intelligent design, it can only supplement it. The better players, i.e. the Wizard of Odds and Beating Bonuses rule-based ones and the Headstart genetic learning approach, provided the better results, regardless of circumstance.

Future work could include implementing the Split function in the blackjack simulator and designing a player that can count cards and/or including multiple players in each hand. Additionally, other varieties of player could be implemented. First among these would be neural networks, which was used in Curran and O'Riordan's work [30], as well as other variations and hybrids of the archetypes already implemented in this experiment. In particular, additional genetic learning teams with variations in population size and repopulation proportions could be included. Of the 60% of the population that was dropped and repopulated, currently 20% was randomly generated and 40% was generated through breeding. Experimenting with these numbers in order to find more ideal settings would extend the breadth of this work.

References

[1] Nolan. (2011) Beating Bonuses. [Online]. http://www.beatingbonuses.com/blackjack.htm

[2] Shackleford, M. (2011, June) Wizard of Odds. [Online]. http://wizardofodds.com/blackjack/strategy/1deck.html

[3] Kendall, G. and C. Smith, "The Evolution of Blackjack Strategies," in *2003 Congress on Evolutionary Computation*, 2003, pp. 2474-2481.

[4] Rudowsky, I., "Intelligent Agents," in *Proc. Americas Conference on Information Systems*, New York, NY, August 2-6, 2004.

[5] D'Hondt, M. and V. Jonkers, "Hybrid Aspects fro Weaving Object Oriented Functionality and Rule-Based Knowledge," in *AOSD*, Lancaster, UK, 2004, pp. 132-140.

[6] B. Qin, Y. Xia, S. Prabhakar, and Y.C. Tu, "A Rule-Based Classification Algorithm for Uncertain Data," in *IEEE International Conference on Data Engineering 2009*, Washington DC, 2009, pp. 1633–1640.

[7] V.S. Rao, "Multi Agent-Based Distributed Data Mining: An Over View," *International Journal of Reviews in Computing*, pp. 83-92, 2010.

[8] Chang, P. C., Liu, C. H., "A TSK Type Fuzzy Rule Based System for Stock Price Prediction," *Expert Systems with Applications*, vol. 34, pp. 135-144, 2008.

[9] Orriols-Puig, A.; Bernado´-Mansilla, E., "Evolutionary Rule-Based Systems for Imbalanced Datasets," *Soft Computing*, vol. 3, no. 13, pp. 213-225, 2009.

[10] Fernández, A.; del Jesus, M.J., Herrera, F., "Hierarchical Fuzzy Rule Based Classification Systems with Genetic Rule Selection for Imbalanced Data-Sets," *International Journal of Approximate Reasoning*, vol. 50, pp. 561-577, 2009.

[11] Whitley, D., "A Genetic Learning Tutorial," *Statistics and Computing*, vol. 4, pp. 65-85, 1994.

[12] Baker, E. and Seltzer, M., "Evolving Line Art," in *Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993.

[13] Gartland-Jones, A., and Copley, P., "The Suitability of Genetic Algorithms for Musical Composition," *Contemporary Music Review*, vol. 22, no. 3, pp. 43-55, 2003.

[14] Osman, H., Georgy, M., and Ibrahim, M., "A Hybrid CAD-based Construction Site Layout Planning System using Genetic Algorithms," *Automation in Contruction*, vol. 12, pp. 749-764, 2003.

[15] Shah, S. and Kusiak, A., "Cancer Gene Search with Data-Mining and Genetic Algorithms," *Computers in Biology and Medicine*, vol. 37, pp. 251-261, 2007.

[16] Bierwirth, C., and Mattfield, D.C., "Production Scheduling and Rescheduling with Genetic Algorithms," *Evolutionary Computation*, vol. 7, pp. 1-17, 1999.

[17] Singh, K., Rani, R., Rani, S., and Singh, V., "Effective Software Testing Using Genetic Algorithms," *Journal of Global Research in Computer Science*, vol. 2, no. 4, April 2011.

[18] Hellmann, M., "Fuzzy Logic Introduction," *Laboratoire Antennes Radar Telecom*, vol. 1, 2001.

[19] Jantzen, J., "Tutorial On Fuzzy Logic," Technical University of Denmark, Technical Report 98-E-868 (logic), 1998.

[20] Schaeffer, J. and H. J. van den Herik, "Games, Computers, and Artificial Intelligence," *Artificial Intelligence*, vol. 134, pp. 1-7, 2002.

[21] Kendall, G. and S. Lucas, "Evolutionary Computation and Games," *IEEE Computational Intelligence Magazine*, pp. 10-18, February 2006.

[22] Norvig, P. and S. Russel, *Artificial Intelligence: A Modern Approach*, 2nd ed. Upper Saddle River, New Jersey, USA: Pearson Education, Inc, 2003.

[23] Schaeffer, J., *One Jump Ahead: Challenging Human Supremacy in Checkers*. New York, United States of America: Springer-Verlag, 1997.

[24] Epstein, S., "Game Playing: The Next Moves," in *Sixteenth National Conference on Artificial Intelligence*, Menlo Park, California, USA, 1999, pp. 987-993.

[25] Buro, M., "From Simple Features to Sophisticated Evaluation Functions," in *First International Conference on Computers and Games*, Tsukuba, Japan, 1998, pp. 126-145.

[26] Ginsberg, M., "GIB: Steps Toward an Expert-Level Bridge-Playing Program," in *IJCAI-99*, Stockholm, Sweden, 1999, pp. 584-589.

[27] Bjarnason, R., P. Tadepalli, and A. Fern, "Searching Solitaire in Real Time," *ICGA Journal*, vol. 30, no. 3, pp. 131-142, 2007.

[28] Barone, L. and L. While, "Adaptive Learning for Poker," in *Genetic and Evolutionary Computation Conference*, 2000, pp. 566-573.

[29] Sandven, A. and B. Tessem, "A Case-Based Learner for Poker," in *Ninth Scandinavian Conferance on Artificial Intelligence*, Helsinki, Finland, 2006.

[30] Curran, D. and C. O'Riordan, "Evolving Blackjack Strategies Using Cultural Learning," *Adaptive and Natural Computing Algorithms*, pp. 210-213, January 2005.

[31] Chen, Y., Jensen, C.D., Gray, E., and Seigneur, J., "Risk Probability Estimating Based on Clustering," in *4th IEEE Annual Information Assurance Workshop*, West Point, NY, 2003, pp. 229-233.

[32] Popyack, J.L., "Blackjack-playing Agents in an Advanced AI Course," in *14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, New York, NY, 2009, pp. 208-212.

[33] Moore, D. and Essa, I., "Recognizing Multitasked Activities using Stochastic Context-Free Grammar," in *AAI*, 2002, pp. 770-776.

[34] Blascovich, J., Loomis, J., Beall, A., Swinth, K., Hoyt, C., & Bailenson, J.N., "Immersive Virtual Environment Technology as a Methodological Tool for Social Psychology," *Psychological Inquiry*, no. 13, pp. 103-124, 2002.

Supplemental Files

File Manifest for Robert Noonan's Thesis Source Code.ZIP

a. Card.java
b. Deck.java
c. Hand.java
d. Table.java
e. Player.java
f. PlaerBeatingBonuses.java
g. PlayerGeneticLearning.java
h. PlayerGeneticLearningFuzzy.java
i. PlayerGeneticLearningHeadstart.java
j. PlayerGeneticLearningMemory.java
k. PlayerH17.java
l. PlayerS17.java
m. PlayerWizardofOdds.java
n. TeamGeneticLearning.java
o. TeamGeneticLearningFuzzy.java
p. TeamGeneticLearningHeadstart.java
q. TeamGeneticLearningMemory.java
r. README.txt

The contents of the README.txt file give directions for using the software and are included here:

To run a rule-based player experiment:
Run the command "java <Player Class> <Generations> <Iterations>
Where:

<Player Class> is the java class of the rule-based player in question.

Possible Values for Player Class:

PlayerBeatingBonuses
PlayerWizardofOdds
PlayerH17
PlayerS17

<Iterations> is the number of hands played per generation. (Default 100,000 hands)
<Generations> is the number of generations run. (Default 25 generations)
The program will display the results on the screen and record them in text files.

To run a genetic learning team experiment:
Run the command "java <Team Class> <Generations> <Iterations> <Training Iterations>"
Where:

<Team Class> is the java class of the genetic learning team in question.

Possible Values for Team Class:

TeamGeneticLearning
TeamGeneticLearningFuzzy
TeamGeneticLearningHeadstart
TeamGeneticLearningMemory

<Iterations> is the number of hands per exhibition match. (Default 100,000 hands)
<Generations> is the number of generations run. (Default 25 generations)
<Training Iterations> is the number of hands per player during training (Default 1000 hands)

The program will display the results on the screen and record them in text files.

To change the initialization of a genetic learning team for an experiment:

1) Edit the java file for the team
2) In the "main" method, find the line "<Team Type> TeamGL = new <Team Type>()"
3) Inside the parentheses enter one of the following:
   a. "S17" to start with Dealer S17 rules.
   b. "H17" to start with Dealer H17 rules.
   c. "WoO" to start with Wizard of Odds rules.
   d. "BB" to start with "Beating Bonuses" rules.
   e. If it's left blank, or anything else is used, it will randomly generate the starting values.
4) Save.
5) Comple ("javac <Team Class>.java")
6) Run