1 Introduction

This project looks to simulate and analyze naive artificial intelligence agents playing the game *Chutes and Ladders*. An extension of the game simulated in two-dimensions, this project simulates *Chutes and Ladders* in three-dimensional space where players move not one, but an arbitrary number of tokens around the playing board. Unlike the previous simulation, the game winner is not determined by the first finisher but as a function of treasure totals along with a bonus for the first finisher. To ascertain a dominant strategy given these new rules, six AI personalities were created to play the game including Random, Anti-Hold, Treasure Hunter, Sprinter, Smart Sprinter, and Tortoise players. After dominant strategies from these six personalities were uncovered, a Hybrid personality was designed to merge the best known strategies. Descriptions of these personalities can be found in Section 2.3. We hypothesize that players who value both attaining treasures and finishing first will win on a more frequent basis than players that do not. That is, we hypothesize that Sprinters, Smart Sprinters, Treasure Hunters, and Hybrids will all win more frequently than the remaining personalities. In addition, we hypothesize that as the value of finishing first increases, Sprinters and Smart Sprinters will win more frequently as opposed to Treasure Hunters. The experimental design to test these hypothesizes are discussed in Section 3. Along with this dominant strategy analysis, an analysis of a game termination condition we call *Infinite JStack* was completed to understand the likelihood of this type of board configuration. We hypothesize that this board configuration will become more likely as the die number increases. All analysis of these hypotheses can be found in Section 4.

2 Approach

2.1 Game Configuration Summary

From the preliminary simulation, a modified version of *Chutes and Ladders* was constructed. This project is an extension of the initial simulation. To benefit the reader, we offer a summary of the game structure from the initial project to motivate the remainder of the section. The game board is represented by a three dimensional Array List[1] in which the inner most array is a collection of spaces. These spaces include aids and obstacle spaces such as treasure pots and hold queues. New to this list of spaces is a *JStack* space (see Section 2.2) which serves as a trampoline moving players to different levels of the game board. A player class tree manages the players participating in the game. Each player inherits a *TokenHolder* object (see Section 2.3) as well as a personality that will be used when moving tokens around the board. Additional classes, such as a *Reader* and *Statics* class manage the necessary input files and high level management of the simulation. Classes were also constructed for exporting data and managing the game interface via the terminal. For a more in depth description of these classes and their functionality, we advise reading the report given for the initial simulation.

2.2 Board Modifications

In this project, the game board has been generalized to three-dimensional space. A three-dimensional Array List[1] was created to manage these changes. The inner most list contains objects of type Space. Upon generation of a random integer via a uniform distribution and the given probabilities given in the configuration file, the board constructs and stores the space in the inner most list. Included in the possible spaces for construction were the original spaces from the base simulation and a new space called a *JStack*. The only modification to the original spaces included a new implementation of the *Priority Hold* space. Instead of using a *Priority Queue* to manage this space, a *Treemap*$^{(2,6)}$ was implemented. No functionality was changed.[4]

The *JStack* space simply holds a multiplier of -1 or 1. Once a player lands on this space, the product of this multiplier and the roll value is mapped via the division algorithm to a value between zero and the z - dimension of the board. The player is then moved to that level of the board. One consequence of this space is the possibility that the player could potentially land on the same *JStack* or some combination of *JStack* spaces that have the player moving from level to level without ever stopping. This condition we call an *Infinite JStack*. Section 4.1 analyzes how frequently this problem arises as well as purposes a further adjustment to the game to remove this end condition.

2.3 Player Tokens

In the original simulation of this game, players held a single token which they moved around the board. In this project, players now contain several tokens. To account for this update, a new *TokenHolder* class was created to effectively store and manage this set of tokens. On instantiation of the game, each player constructs a *TokenHolder* object to manage their tokens. This class also hosts all functionality for decision processes of a given player. Based on the player's personality and the current roll value, a player assess her *TokenHolder* set and decides which token to move. Additional functionality to aid this decision algorithm was added to the *Token* class including a "look ahead" method that allows the player to attain the future space of each token given a die roll. Once the decision algorithm is complete, the given token is moved via the division algorithm (once for the z-value and once for the y-value) and is then placed on the appropriate space. All other functionality of the game remains unchanged.

2.4 AI Personalities

To most effectively play this modified version of *Chutes and Ladders*, AI personalities were constructed. Initially, six personalities were created. These included Random, Anti-Hold, Treasure Hunter, Tortoise, Sprinter, and Smart Sprinter players. The **Random** player randomly selects a token to move on any given round. The **Anti-Hold** player prioritizes having all tokens out of holds. Therefore, if a token is stuck in a hold and is allowed to move, this player will always move that token. If the player's tokens are stuck in holds or no tokens are in holds, this personality will simply choose a token at random from the set of tokens not in a hold. The **Treasure Hunter** will always

try to attain the most treasure on any given turn. Using a query method for future space treasure totals, this player will move the token that attains the most treasure on any given round. The **Tortoise** player will attempt to move their entire team to the finish. Therefore, this player will always move the token that is furthest behind on the board. The **Sprinter** is the exact opposite of the Tortoise. Prioritizing finishing first, this player will always move the token furthest along the board. The **Smart Sprinter** is a modification on the Sprinter player that ensures that the furthest token does not land on a hold space. If there is no way to avoid this situation, the player chooses a token to move at random.

Upon initial analysis, discussed in section 4.2, we see that the dominant strategies are Treasure Hunters, Sprinters, and Smart Sprinters. By calculations made in the analysis section, we see that Treasure Hunters hold the dominant strategy for low values of the finishing bonus and the Sprinters hold the dominant strategy for the high values of the finishing bonus. These results will be discussed in depth in section 4.2. The **Hybrid** personality chooses to play a certain strategy as a function of the value of the finishing bonus. If the finishing bonus is less than 625 then it will play the Treasure Hunter strategy. If the finishing bonus is greater than 675, the player will play the Smart Sprinter strategy. If the bonus is between these two values, it choose a strategy randomly for each turn.

3 Methods

As discussed in the introduction of this paper, this project looks to find the dominant game playing strategy for this version of *Chutes and Ladders* as well as answer the question posed in Section 2.2 regarding *Infinite JStack*. To first address the *Infinite JStack* problem, we run a simulation with the original implementation of the game across several values of die number, number of players, and number of tokens on the default configuration file. The default configuration file is summarized in **Table 1**.

board	10	10	5
values	33	1.5	
HoldQ	5	2	5
Hold	10	-3	-1
treasurePotA	15	5	50
PriorityHold	5	-2	2
treasurePotB	10	60	
JStack	10		

Table 1: Default configuration file

The dice number was varied from 2-9, the number of players was varied from 2-8 by two, and the number of tokens was varied from 10-40 by ten. Each combination was simulated 40 times. From this experiment, finishing data was recorded including finisher personality, winner personality, treasure totals, and round totals.

After the removal of the *Infinite JStack* termination condition (see Section 4.1 for details), this

experimental design was replicated several times each with a differing number of finishing bonus values. Experiments were completed with finishing bonuses of 33, 75, 150, 250, 500, 600, and 700. From here, an analysis of the dominant strategy was completed.

The final experiment was completed with the Hybrid personality implemented. 5000 games were simulated with only Sprinters, Smart Sprinters, Treasure Hunters, and Hybrids with a random finishing bonus between 0 and 1000. Finally, this experiment was replicated with a finishing bonus randomly selected from a normal distribution (mean 650, standard deviation 100). These experiments attempt to see how the Hybrid personalities compete against the original strategies with an unknown finishing bonus.

4 Data and Analysis

4.1 Infinite JStack

The *Infinite JStack* ending condition occurs when a player lands on the same *JStack* or some combination of *JStack* spaces that have the player moving from level to level without ever stopping. An initial analysis was completed to see how many games finished and how many games ended via this termination condition. Consider **Figure 1** below.

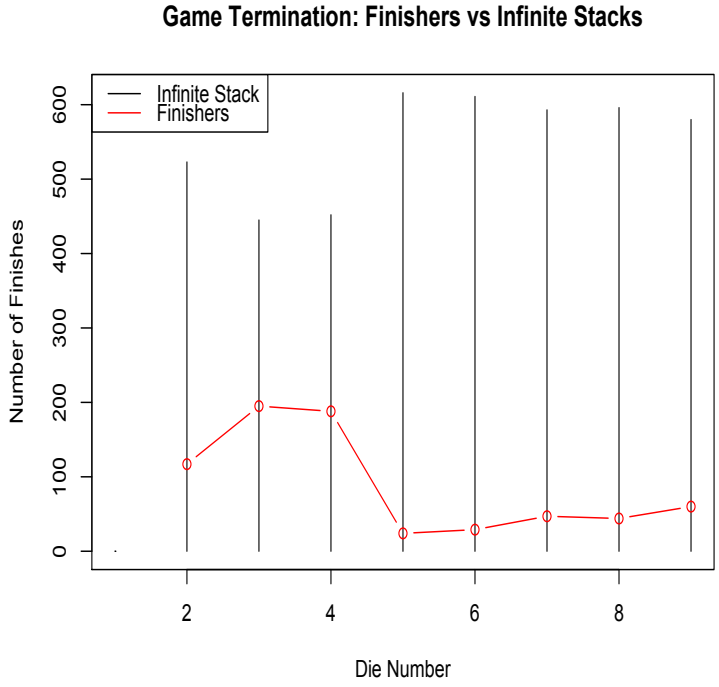

Figure 1: Game Termination: Games finishing due to an *Infinite Stack* vs games finishing due to a finisher or round total termination.

Figure 1 shows the number of game terminations due to an *Infinite Stack* against the number of games finished due to a legitimate finisher or a round total termination. Notice that the number of games finished due to an *Infinite Stack* is much greater than the number of natural finishers for all die numbers. Further notice that there is a sharp decline in natural finishers when the die number is greater than 5. This is due to the modular implementation of the *JStack*. If a player lands on an *JStack* with a roll of a five in this experiment, the game will certainly fail as the token will just be cycling back to the same *JStack* after every iteration. This supports our hypothesis proposed in the Introduction section of this paper. We expect a cyclic patter if we allowed the die number to grow larger than 10 as players would have this same issue for any number that is equivalent to 0 $\mod 5$.

This analysis clearly shows that this implementation would not be appropriate for an analysis of dominant strategy. To remove this end game condition, we offer a modification to this game. Instead of having a player jump a second time if it lands on a *JStack*, we propose that a player can only be moved by a *JStack* once per round. This removes this game termination condition and allows for more players to finish. For the remainder of the experiments, this implementation will be used to remove end game conditions that may inhibit a fair analysis of the AI personalities.

4.2 Dominant Personalities

To test which strategy was most dominant, a series of experiments varying finisher bonus was completed. Consider **Figure 2**, a compilation of the number of wins by personality across all seven experiments outlined in Section 3.

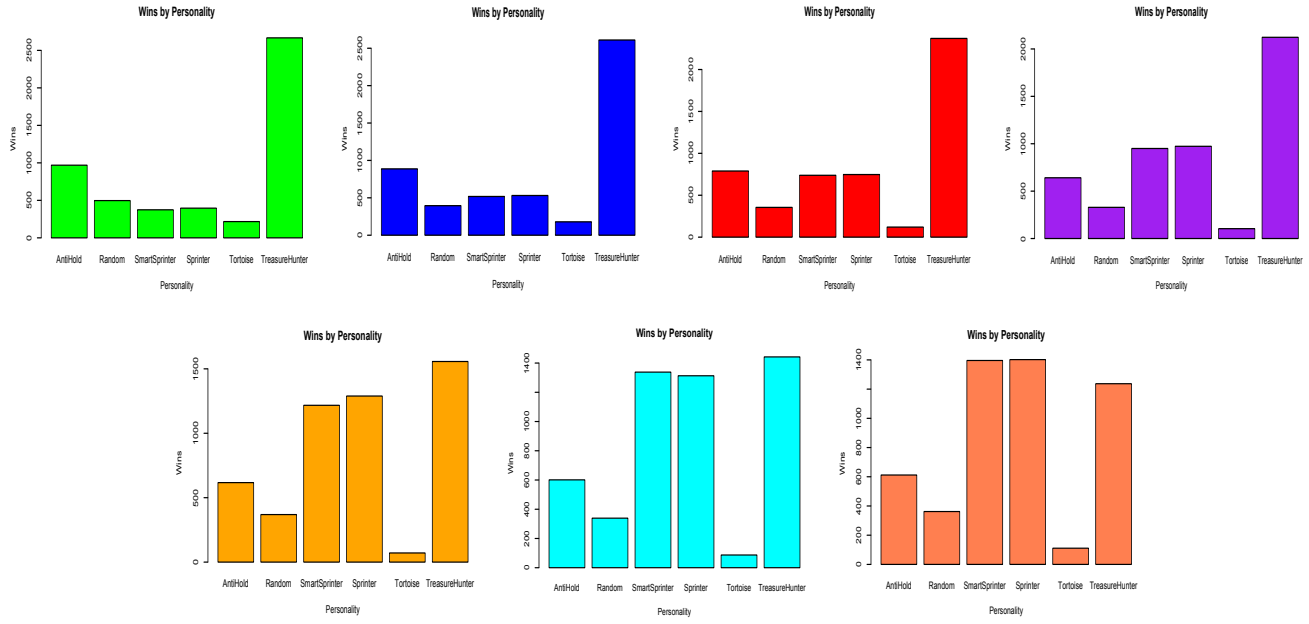

Figure 2: Dominant Strategies: Number of wins by personality

Notice how in all simulations, excluding the final experiment, that the Treasure Hunter was

the dominant strategy. Especially in experiments with low finishing bonuses, the Treasure Hunter out performed all other strategies combined with 53% of total wins. Furthermore, as the finishing total rises the two Sprinting personalities rise almost identically. This suggests that there is little difference between the smart and standard sprinter. The Tortoise personality is consistently the lowest performing personality in the group, even behind the Random personality. Both the Random and Anti-Hold players appear to perform at the same rate regardless of the finishing bonus. This could be due to the fact that neither finish first frequently and therefore do not gain the finishing bonus.

These results both contradict and support the purposed hypotheses. The Treasure Hunter was the dominant strategy, supporting our hypothesis. The Sprinter players, however, were not successful, in comparison to the constant personalities, until the finishing bonuses were above 150. This suggests that all players should play the Treasure Hunter strategy unless the finishing bonus is remarkably high. But how high is "remarkably high"? Consider **Figure 3**, a visualization of the Sprinter players against the Treasure Hunter.

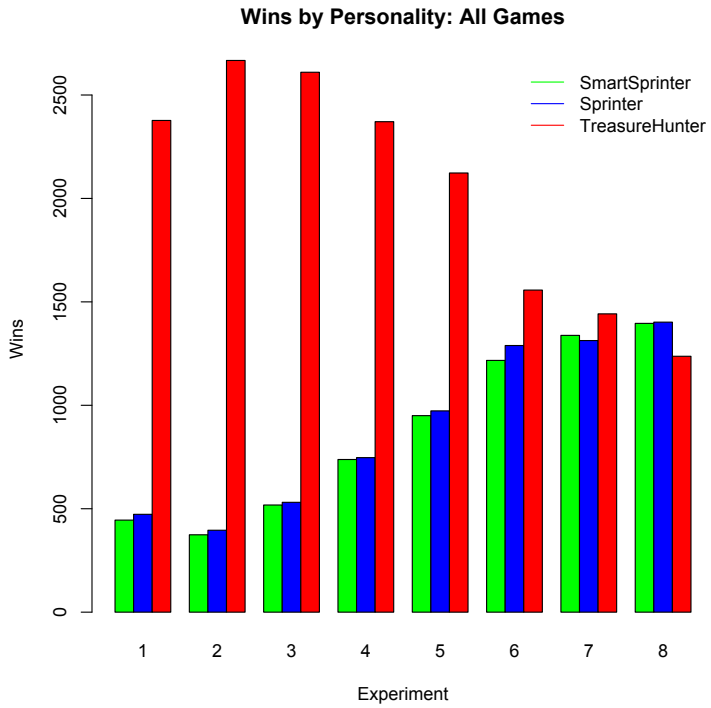

Figure 3: Comparing Strategies: Number of wins by personality

Notice as the finisher bonus rises, the Sprinters begin winning more games as the win totals for the Treasure Hunter's wins are steadily decreasing. The first instance where the Sprinters win more games than the Treasure Hunter is when the finisher bonus is 700. Note, however, that the wins for the three personalities are comparable when the finisher bonus is 600. Therefore, we estimate that, holding others constant, that the Sprinter strategy is dominant only when the finishing bonus is greater than 650. For all values of the finishing bonus less than this number, the Treasure Hunter is the dominant strategy.

4.3 Hybrid Personality

In an attempt to create a truly dominant strategy with the information given in the previous section, a Hybrid player was created to adjust strategies as a function of the finishing bonus. The details for this personality were discussed in Section 2.4. An experiment was completed to compare this personality to the original, dominant personalities. This experiment assigned a random finishing bonus between 600 and 700. **Figure 4** contains the results of this experiment.

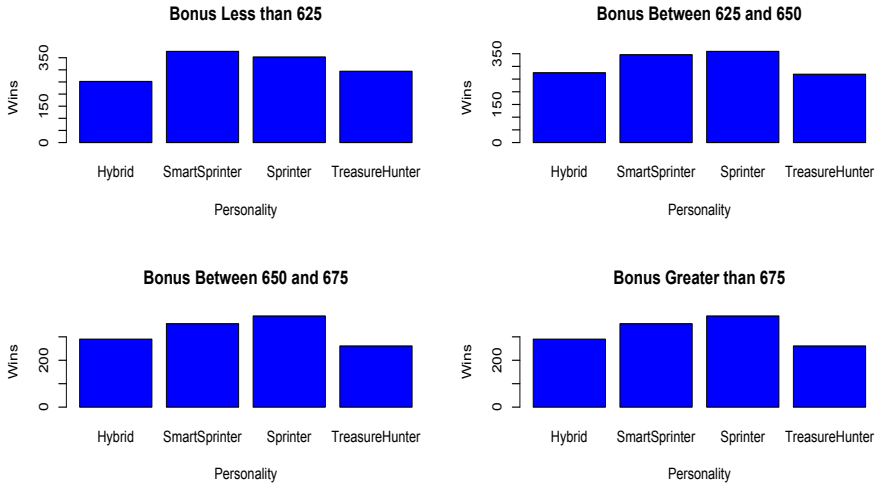

Figure 4: Hybrid Analysis: Varying finisher bonus to see affects of winning personalities

Notice that it appears that the Hybrid player does not affect the Sprinter player's win totals while diminishing the Treasure Hunter's win totals. A replication of this experiment was completed without constricting the finishing bonus to the $[600, 700]$ range. Finishing bonuses were sampled from a normal distribution with mean 650 and standard deviation $100^{(3,6)}$. The results are displayed in **Figure 5**.

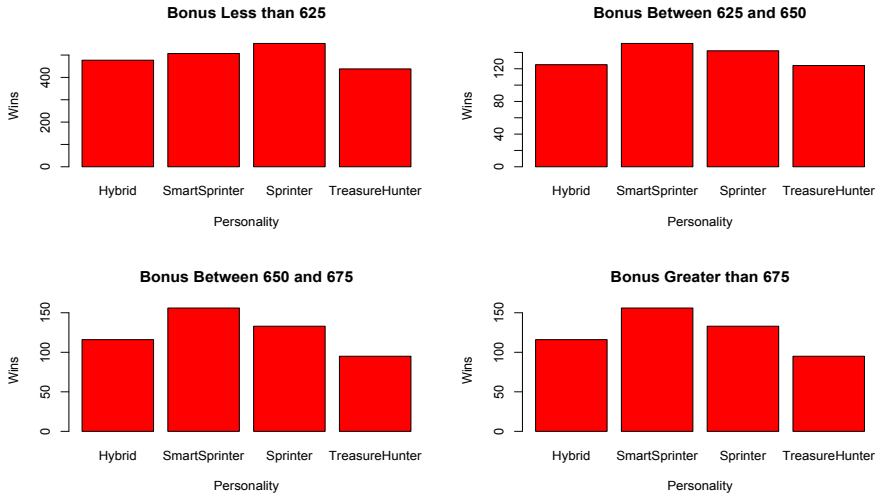

Figure 5: Hybrid Analysis: Varying finisher bonus via normal distribution

Again, we see that the hybrid player's are not affecting the Sprinter's win totals but only hindering the Treasure Hunter's win count. This suggests that for high values of the finishing bonus, a player should try to finish as fast as possible if playing with either a Treasure Hunter or a Hybrid to account for the dynamic decisions made by the Hybrid personality.

A final experiment was conducted in the same fashion as the previous experiments. This experiment, however, was completed with a random finisher bonus selected from a uniform distribution over the range $[0, 1000]$. The results of this experiment can be found in **Figure 6**.

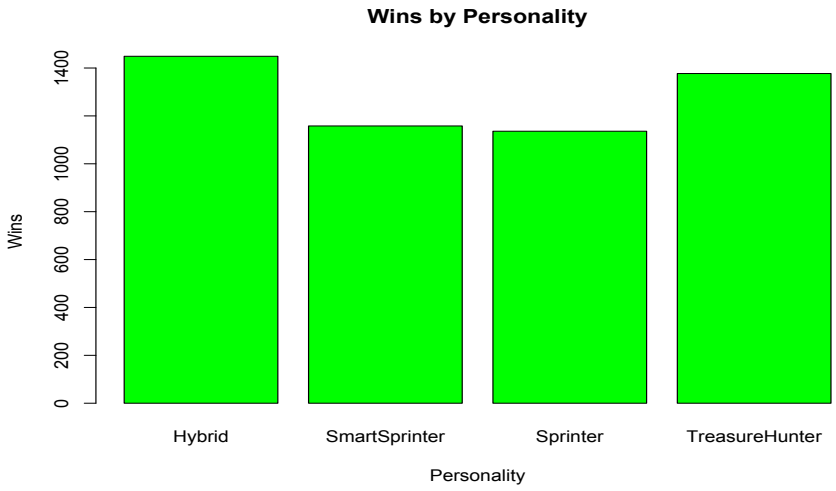

Figure 6: Hybrid Analysis: Varying finisher bonus $[0, 1000]$

Here we see that with a random value of the finisher bonus, the Hybrid and Treasure Hunter personalities are still winning the majority of the games. Therefore, this analysis recommends the Hybrid personality, unless playing against another Treasure Hunter. These results neither support or contradict our hypothesis. This Hybrid was neither worse nor better than any of the other dominant strategies in the original personalities. For this reason, a more advanced AI needs to be developed to better merge the Treasure Hunter and Sprinter personalities.

5 Conclusion

This project extended the initial simulation of *Chutes and Ladders* to three-dimensional space. This paper completed analysis to address concerns with the current board configuration and suggested alternative board configurations to avoid future problems. With new game components, naive AI decision algorithms were designed to intelligently play the game. The Treasure Hunter and Sprinter personalities preformed the best as the Random, Tortoise, and Anti-Hold players all

struggled to compete. The final personality, Hybrid, competed with the original dominant personalities, but failed to merge the two personalities into one superior strategy. Further work will reveal a hybrid personality that effectively combines these personalities to achieve a true superior agent.

6 References

1. "ArrayList (Java Platform SE 8)." *ArrayList (Java Platform SE 8)*. `https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html` N.p., n.d. Web. 19 Feb. 2016.

2. "Comparator (Java Platform SE 8)." *Comparator (Java Platform SE 8)*. `https://docs.oracle.com/javase/8/docs/api/java/util/Comparator.html` N.p., n.d. Web. 17 Feb. 2016.

3. Degroot, Morris and Mark Schervish. *Probability and Statistics*. Boston, MA: Addison Wesley, 2002. Print.

4. "Project Description 2." `http://cs.lafayette.edu/~liew/courses/cs150/` N.p., n.d. Web. 17 Feb. 2016.

5. "Random (Java Platform SE 8)." *Random (Java Platform SE 8)*. `https://docs.oracle.com/javase/8/docs/api/java/util/Random.html` N.p., n.d. Web. 12 Feb. 2016.

6. "TreeMap (Java Platform SE 8)." *TreeMap (Java Platform SE 8)*. `https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html` N.p., n.d. Web. 24 March. 2016.

7. Weiss, Mark Allen. *Data Structures and Problem Solving Using Java*. ed 4. Reading, MA: Addison Wesley, 2010. Print.