Parker Coady, Esteban Ramirez, Treksh Marwaha
CS4100 Term Project Writeup

What We Did

For the term project, we created a tool to help users play UNO at an inhumanly optimal level. The tool does not simulate an UNO game within itself, and nor does it provide a graphical interface to the user or detect illegal plays- however, if given the appropriate input, it will keep track of a real-life UNO game's current state and suggest plays to its user, as well as indicate facts about the gamestate (like the exact number of cards in each player's hand and what cards they were not able to play on with their current hand) that a human might not be able to remember. This tool works for only games of UNO with four players and a standard UNO deck (108 cards, no house rules). It deals with single games, but its suggestion method factors in attempting to score the maximum number of points per game and minimizing point gain for the winning player, so it can be used for proper rounds of point-based games.

The tool is to be used in combination with a player, not as a substitute for one. The player should follow the instructions that the tool outputs and input cards appropriately, but also is the one responsible for making the actual plays. The tool will supply information to the player, but it is ultimately up to the player's own judgement to regard or disregard the bot's advice (for example, the bot is unable to read bluffs or make alliances with other players, for obvious reasons).

The tool is created in java.

Why We Did It

The main goal of the assignment was, from the beginning, to create a useful tool that users can go back and revisit, bringing it above the simple distraction of a novelty. Thus, our tool provides a number of systems that make it useful for a user, like the features mentioned above. On top of that, the idea of creating an AI that played card games was what initially brought our team together.

Although a tool that not only simulated but *played* a near-perfect game of UNO in a player's stead would have been a good compromise between these two initial design concepts, we quickly ran into two main roadblocks. The first was one of scale- we were only a three-man team, and the duration where the project was assigned without other work from this class on top of it had shrunk significantly due to various delays. Even from the beginning, it wasn't feasible to make a tool which simulated the actual playing of the game, and we cut back on a visual interface in favor of a text-based one due to further constraints (although it was specified in our proposal that one of our goals was not to make the most streamlined UI, favoring functionality over polish). The second roadblock was that, as far as our research showed, there *was* no AI

which played UNO to a perfect degree, as it is a *highly* unsolved game. The most we could find was a similar project which employed a large amount of high-level mathematics and even deep learning, but still only had a winrate around 50%.

How It Relates To Things We Read/How It Works

The main logic that powers the suggestion mechanism is adapted from an online strategy written in psuedocode, but upon closer examination, is really a decision tree! During the player's turn, the tool analyzes the top card on the deck, then makes a number of decisions based on the gamestate- the number of cards in other players' hands, the types of cards that have been played, and the colors and numbers of cards in the user's hand. It makes three observations about the gamestate, and if any of those are true, begins testing if the user has cards in their hand in the priority of which to play them. This process is also very similar to traversing a highly unbalanced decision tree, where seeing a card in the user's hand makes the engine suggest it and end the decision tree, but seeing the player does not have a card continues to move down the tree.

If it determines that the player is about to lose, it heavily prioritizes playing draw fours, wilds, and other high cost cards.

Otherwise, if the player has more than 1 wild or draw 4, it prioritizes playing all but one remaining. This is to try and get the player towards a state of having a wild card as the last card in their hand.

It then does the same thing for draw 2s, getting rid of all excess but keeping one in case of emergency (having a player next to you with "uno").

Otherwise, the tool prioritizes recommending skips, then reverses, then the spare draw 2, then number cards, then wild draw 4s.

In rare scenarios when the player cannot play any cards but their only wild, the AI prioritizes passing rather than spending it.

Lastly, the AI plays their one wild left.

When playing wild cards, the suggestion AI will also choose colors based on a very small decision tree. It will see what colors of cards the player has, and then will count the number of cards that have been played before as well as the cards in its hand, and choose the color that has been played most frequently. This is to make it unlikely that other players will be able to play.

Important of note is that this is not an optimal strategy by any means- simply a strong, logically consistent set of suggestions.

Parker Coady, Esteban Ramirez, Treksh Marwaha

What Our Results From Evaluation Are/What They Mean

Although our UI has been in various states of completeness throughout the course of the project, our logic has been working since its implementation. Of our tests with complete logic, the bot-assisted player won six out of nine times- a stunning result, by all means. The suggestion tool nor the UI ever stuttered or lagged in any of the games or during periods of copious user input, indicating that our chosen algorithm was also optimal enough not to cause significant delay in computation. Although more testing opportunities would have been appreciated to get a greater sense of just how effective the tool is, according to our evaluation criteria outlined in the proposal, our tool passes evaluation with flying colors.

These results tell us that not only is a decision tree efficient for an algorithm of this size, but the particular decision tree we chose- however simple it is- is extremely effective at helping players beat human opponents. Although the bot's only superhuman abilities were its precise memory of played cards and cards that other players had to pass on (in the latter case, not even interpreting those results into its decision making process, simply displaying them to the player), its instant decision-making, and its extreme logical consistency, when augmenting a human player, the results are undeniable. It seems likely that the bot's shortcomings- such as not being able to predict players' mental states and bluffs based on gestures, or being unable to form alliances with other players- were covered by the human's second judgement, forming a team far stronger than the bot by itself could have been.

What We Learned

More than anything, this project was a lesson in carefully controlled scope. It took a lot of effort to decide what parts of UNO were game logic we could trust the user to handle, and what parts of the game we wanted the tool to figure out by itself. For example, the tool keeps track of the turn order, and changes the turn order appropriately when the normal procedure is interrupted by an irregularity such as a draw two, skip, or reverse. However, it does not automatically draw cards for opponent players, or even determine if cards played by opponents are legal. Implementing *all* of the features of a full UNO simulation would have made the project incompletable with the time and manpower we were given, but, with careful planning around what features to include and exclude, we were able to complete the tool in a relatively small amount of time.

We also learned that, when a perfect algorithm for an optimal play in an unsolved game does not exist, a truly helpful assistant tool for a game like this simply needs to provide information to the player. Simply knowing a list of cards an opponent couldn't play on or what

play the bot would *suggest* in an environment without readable human players gave the humans a significant advantage, even if the tool's suggestion algorithm was far from giving a complex, optimal solution for every situation.