High-Low

Cameron Burwell
Bryce Cheng-Campbell

Computer Science
California Polytechnic State University, San Luis Obispo
Advisor Franz Kurfess

June 2016

Table of Contents

- Table of Contents
- 1. Abstract
- 2. Introduction
- 3. Background & Technologies
- 4. Requirements
- 5. Design
- 6. Development
  - Winter 2016 - Week 2 - Background Research
    - High-Low Card Game Rules
    - Similar Game AI Agents
  - Winter 2016 - Week 4 - Implementation
  - Winter 2016 - Week 6 - Design
  - Winter 2016 - Week 8 - Networking
  - Winter 2016 - Week 10 - Functioning Online Game
  - Winter 2016 Quarter Summary
  - Plan for Spring 2016
  - Week 2 - Planning
  - Week 4 - AI Development
  - Week 6 - Full AI Integration
  - Week 8 - Testing, finalize integration, tweaking
- 7. Challenges
- 8. Conclusions
- Appendix
  - Winter 2016 - Proposed Schedule and Deliverables
    - Meetings
    - Milestones
    - Schedule
  - Spring 2016 - Proposed Schedule and Deliverables
    - Meetings
    - Milestones
    - Schedule
  - User Testing
  - Testing Data
  - Source Code

1. Abstract

The card game High-Low is a cooperative game that provides a unique problem of where all players have partial information and build off other players guesses to win the game. This problem lends itself to creating an AI bot that can play the game with other human players. By making decisions that change as the other players make guesses through the rounds, we have created an AI bot that plays the game with a good degree of accuracy.

2. Introduction

High-Low is a card game where each player receives a playing card that everyone else can see. The game has two rounds where in the first round each player guesses their rank across all the cards from high to low and the second round the player guesses their rank and the value of their card. The game is won if all players guess their rank correctly, with the potential for everyone to guess both their rank and values correctly, leading to a perfect game. This is a logic game because each person can see everyone's card but their own, so they need to use other player's guesses to determine what their card is. We made a web interface for people to play the game online with each other and with optionally added AI players. This is an interesting AI project because the computer will need to remember other player's guesses and try to help the group by making decisions like guiding another player away from their initial guess if it's detrimental to the group.

3. Background & Technologies

There are existing online games that are either card games or logic games similar to this one, but none for High-Low that we could find. We used JavaScript, HTML, and CSS along with JS frameworks Express and Node.js to make a web server and client for people to play the game. The communication is all done through websockets using Socket.io. Players can add AI's to their game if they want more players, or even want to play a game solely against AI. The AI are all controlled on the server side whenever it is their turn.

4. Requirements

4.1 The server shall accept incoming connections and allow them to join the game lobby.

4.2 In order for a game to start, four or more players must be present with at least one human player.

4.3 When the game starts, the system will assign a card to each player.

- 4.3.1 Players will be able to see everyone's card but their own.
- 4.3.2 All cards will be unique, none of the same value

4.4 A random player is selected to start the game.

4.4.1 After the first player, players turns will proceed based on when they joined.

4.5 On their turn, a player will enter a number between 1 and the number of players as a guess of what their card rank is.

4.5.1 This continues for one full round of the players.

4.6 On the second round, players will guess their rank and card value.

4.7 If every player correctly guesses their rank, the game is won.

4.7.1 If every player guesses their rank and value correctly, the players achieve a perfect game.

This system will be low performance, and can be hosted with reasonable performance using the hardware present in most computers. Privacy and security need not be concerns, as the game only uses an anonymous name and the server is not hosting anything important.

5. Design

The basis of the client and server are done through the Node.js platform and the Express library built off that. Node mainly handles the package that is our project, and the addition of libraries like Express, Socket.io, and build tools. Express handles the distribution of the webpage and client javascript. We use Socket.io to connect the client to the server so the lobby and game states can be communicated.

To handle the UI, all of the components are made using React. Components are broken down into parts like Lobby, Game, Player, etc. Actual visual design of our game is all functional, we are not designers and could definitely make major improvements to user inputs, layout, and graphics.

6. Development

Winter 2016 - Week 2 - Background Research

High-Low Card Game Rules

- Each player receives a card that they cannot see but is visible to every other player
- Each player gets a turn to first take a guess at what rank their card is relative to all players
    - Based on information gained from looking at the cards of other players as well as the guesses of other players
- After this round, each player guesses what the rank and value of their card is
    - If everyone guesses correctly, the game is won
    - If one person guesses incorrectly, the game is lost

Similar Game AI Agents

- Monte Carlo Method
    - Used in games such as Bridge, Briscola, Go, Four In A Row, Chess, Tic Tac Toe
        - Example of Four In A Row - Monte Carlo Method
    - Sample use case for Monte Carlo
        - The search space is very huge (i.e Go board game)
        - You don't know how to build a strong heuristic
    - Seems great for the examples, not so great for our application
        - Limited by selecting “best move” from a tree of moves where it is presumed that the probability of guessing your card correctly in High-Low is entirely dependant on the first round of guesses. Since this is entirely new on every play-through, it seems hard to generalize what the best move would be.
- Minimax / Alpha-Beta Pruning
    - High-Low is *not* a Zero-Sum Game
    - Figuring out “which move is worse” after eliminating obvious capabilities may depend largely on chance
- Paper on General Card Game Playing (for AI)
    - Holds some potentially interesting information about figuring out heuristics and other features of the AI in games with both perfect and imperfect knowledge of the game.

Winter 2016 - Week 4 - Implementation

Disclaimer: For this task, we accidentally did Week 6’s task in place of Week 4.

This task marks the completion of Milestone 1 where we have a functioning game where one user plays as all the players.

To implement this in the JavaScript command line, we had to use the NPM library called ‘readline-sync’. This library allowed us to take continuous user input, also allowed for prompting questions and waiting on input.

This is the bare bones implementation of the game and shows the logic behind setting up the game with N players, and then proceeding through the two rounds of the game. Based on the user guesses, it will say who guessed right and if the whole game was won or not. This is essentially our prototype that puts the game into JavaScript for future reference.

Winter 2016 - Week 6 - Design

Disclaimer: For this task, we did Week 4's task in place of Week 6 due to an earlier mix-up.

In this task, we created the initial design of the system. This includes UI and general architecture that we will use. The codebase is written in Node.js and we will continue to expand the packages used as well as adding Express which will provide us with various HTTP calls in order to set up networking.

The initial plan for networking is to have a user select if they wish to host a game or join a game. Hosting the game will spin up a server on their machine and connect them to it. Anyone who selects join a game and selects this server from the list will also be added. Express should allow us to do this fairly easy and the majority of this phase will be ensuring that proper conditions and error handling are in place so that the system remains robust.

Winter 2016 - Week 8 - Networking

For week 8, we implemented network connectivity for our game. While the gameplay mechanics are not entirely implemented and tested, networking is an important part of the game design. Setting up connectivity requires a client and server, which we've decided to set up using Node.js, Express, and Socket.io. Our server is hosted on one of Bryce's domains and anyone is able to connect to it. Once the lobby system has been fully implemented, users should be able to go to the domain and create a lobby or join one. Currently we have a placeholder screen and debugging prints going to console to ensure that the websockets are working. The next milestone should be a fully implemented and playable game of High-Low that also allows users to join or create lobbies using any internet connected device.

Completion of this task marks the completion of milestone 2 - server setup. Milestone 3 will also be completed by the end of Winter quarter, which is finishing the UI and client interface.

Winter 2016 - Week 10 - Functioning Online Game

Week 10 marks the completion of milestone 3 which is having the client setup as well as the UI elements and I/O methods. At this point, our game is fully playable over a network with 4-13 human players.

ReactJS was used to create the client UI and allowed for rapid development of dynamically scalable code, permitting us to allow 4-13 players. While the UI is rudimentary, the game is fully functional, which was the main goal for this week.

Winter 2016 Quarter Summary

As the first quarter of our senior project comes to a close, our project has been going fairly smoothly as we have met all of our deliverables and milestones by the expected time and have attended all meetings with our advisor, Professor Kurfess. We have a fully functioning game of High-Low that is playable over the internet, hosted on Bryce's server (highlow.nubsrevenge.seedr.io). The UI is simple but functional, providing intuitive controls for anyone who wants to connect to the server and play the game, and also allowing lobbies to start a new game within the same lobby.

Plan for Spring 2016

The first action item for Spring quarter is to figure out a meeting time to meet with our advisor and provide updates. After this, we will plan out the remainder of our project, giving concrete deliverables and milestones with due dates, similar to what we did for the first quarter. While this isn't specified in the guidelines for senior projects, it helped us stay on task and keep our advisor up-to-date on the status of the project.

The next major goal of our project is to implement Artificial Intelligence into our game in the form of a computer player that can be added to any lobby. We hope to be able to allow 1-12 bots to be added to any lobby, so long as the player limit stays within 4-13 and there is always at least one human player.

Before we start developing the AI, we will attempt to recruit some subjects to test our game for us. Having real world test runs will help us find and fix any bugs that may be in our game. It can also give us an idea of how people play, perhaps providing us with insight into how we should develop our AI.

Neither of us have significant experience with front-end web development, so our UI lacks beauty. However, we hope to improve its looks by the end of the project, should we have time after implementing the AI.

Week 2 - Planning

Our proposed meeting time for Spring 2016 is every even week at 4pm on Wednesday. We planned out the remainder of our project, giving concrete deliverables and milestones with due dates, similar to what we did for the first quarter.

The next major goal of our project is to implement Artificial Intelligence into our game in the form of a computer player that can be added to any lobby. We plan to be able to allow 1-12 bots to be added to any lobby, so long as the player limit stays within 4-13 and there is always at least

one human player. This requires us to at least sketch out a plan for implementing the AI as well as how to integrate it into the game.

Week 4 - AI Development

For this week, we hope to meet our milestone 1 which is having the AI developed. We've experienced some setbacks in implementation details but we've reached a point where we can at least add AI players into a game. Our next step is to implement AI functionality so that the game is playable with the AI. This can likely be completed by the end of the weekend, if not, it will be completed by the end of next week for certain. Hooking up the AI should have been the most difficult obstacle so AI development should be able to progress smoothly from here.

Our goal by the end of this weekend is to have a functional AI player that operates with at least a rudimentary level of intelligence, possibly with user assistance, if necessary. For Week 6, we plan to have the AI fully implemented at a reasonably functionally complex level and the game should be playable with AI players without user assistance.

Week 6 - Full AI Integration

For this milestone, we finished our AI integration. At this point, the game is fully functional and playable with and without AI players. The AI for the game is fairly simple, as it's hard to determine heuristics for this game. The nature of the game is such that each game is only two rounds (i.e. the AI player has one turn to "learn" about the game) and it can't make inferences to the next game because the previous game has no weight on future games.

Previously, the AI just made random guesses based solely on other players' cards and the likelihood that the bot's card is in the larger range (i.e. if it sees the highest is 8, it'll guess highest with 9, 10, etc.) and pick a number at random from there. For this week, we modified the AI to make it slightly more complex. The AI now uses the best guess in the range (i.e. if highest is 8, it'll guess 9, if lowest is 3, it'll guess 2, etc.) which helps to keep the game information more accurate and therefore more winnable for the players. It now takes other players' guesses into account as well.

If a player that played prior to the bot guesses the rank that the bot was going to guess, it looks at their card and sees if their guess is practical based on the other cards that the bot can see. If the player's guess isn't practical, then it disregards the guess and plays how it would have before taking the guess into account. If it is a possible guess, then the bot will choose the second most likely number in its calculated range. (i.e. if highest is 6 and someone guesses highest 7, the bot will guess highest 8 if the previous player's guess checks out).

At this stage, our AI is fairly competitively implemented and should be able to play the game effectively enough to help the other players win, since the goal is for everyone to guess correctly. This update completes milestone 2. The next goal in our project is to have the game user tested and round out any bugs found through testing, as well as work on any outstanding optional features like improved UI.

Week 8 - Testing, finalize integration, tweaking

By week 8, our project has essentially come to an end in terms of development. At this point, we have completed milestones 2 and 3, leaving only the final documentation and submission remaining for our project to be complete. Most of the game's features are rounded out and the game is fully playable with both AI and human players of any number. The last two weeks have consisted of user testing that uncovered a few bugs, some of which we are unable to fix given our time constraints. We weren't able to update the quality of our UI but aside from that, we're content with how the project turned out.

We ran our tests with combinations of new players, experienced players, and AI players. A few notable averages of the games we had (given three games per player configuration and taking whichever result was most common):

2 experienced players, 2 AI - Win
2 exp. players, 2 new players - Win
4 exp. players - Win
1 exp player, 3 AI - Win
1 exp player, 3 new players - Lose
4 new players - Lose
1 new player, 3 AI - Lose

Our results were interesting and it seems that the bots are only as useful as the human players' ability to strategically use them. In our 4 new player testing, the third game was almost won because the players had begun to learn the rules by that point. Every game with experienced players and AI was won, presumably because the experienced players were able to use the bots' guesses to their advantage whereas the new players were still learning the rules. The tests involving 1 experienced player with 3 AI and 1 experienced player with 3 new players are of particular interest because it seems to indicate that the AI's capability lies somewhere between that of a new player and an experienced player, which was the level we were initially aiming for.

There were a few bugs uncovered in testing that include:

- If a player refreshes the game or goes back a page, the game will freeze and everyone will have to refresh in order to continue playing the game.
- If there is an active game and someone disconnects, it should automatically kick them to the lobby or give a message that notifies players that a player has left.

- User input can be a little messy or unexpected, leading to other issues.

Of these known bugs, we only had time to fix the last one so that our input mechanism is a little more robust.

7. Challenges

Most of the early challenges were figuring out how to make the website, server, and client. We had to learn different javascript libraries to help with that and got networking down. Once we had a prototype I was able to put it up on a “production” server to make sure it still worked. Then the challenge was building off that prototype to get the full game working with the server doing most of the work. Figuring out how the game state was passed around was pretty challenging and took a lot of debugging on both the client and server side of things.

One of the biggest challenges we faced was deciding how the bots were going to be handled. We did not know the capabilities of our game server and possibly wanted the bots to be attached to the client. We initially set this up and had the bot’s turn get tied into the same time as the player’s turn. But this proved unwieldy and had more problems with multiple bots. We then converted it over to the usual server sided bot concept where it detects if the next player’s turn is actually an AI player, then it runs through our decision making code using the current game state to perform that AI player’s turn.

8. Conclusions

The primary goals of this project were for us to create a software version of a card game that we enjoy playing with our friends and to further explore techniques for implementing AI in games, motivated by our CPE 480 - Artificial Intelligence class.

We initially thought that the Monte Carlo method would be the best fit for our application as it’s used in many software board games. However, this method was not appropriate for our game because there is no “memory” that can persist between games, rendering this method useless. Methods we researched did not seem to lend themselves to our application, so we had to create our own method. Our method is fairly simple and allows AI the play somewhere between beginner-advanced human player level, depending on how the human players decide to use the AI in their strategies. The more experienced the player, the more helpful the AI is.

The technologies we used made it easy to implement and host our game on the web. Express, Node, and Socket.io worked well together and also allowed us to easily inject AI players into a game with minimal modification to our game that originally only support human players.

Appendix

Winter 2016 - Proposed Schedule and Deliverables

Meetings

Every even week, Thursday at 2pm for Winter 2016.

Milestones

1. Player only games working
2. Server setup - concept of player names and lobbies
3. Client setup - user interface of cards, other players, and the rank/value inputs
4. AI developed
5. Integrate AI and game fully working

Schedule

Week 2
Background research

- High-Low card game rules
- AI agents that have been implemented in games similar to this one

Week 4
Continued research and initial design of the system

Week 6
Functioning High-Low Card Game implemented and playable without AI

- One person can play as multiple players to ensure usability of all features

Week 8
Functioning web server and client

- Set up network connectivity

Week 10
Functioning game over web server

- Players can join and play a full game over the network

Spring 2016 - Proposed Schedule and Deliverables

Meetings

Every even week, Wednesday at 4pm for Spring 2016.

Milestones

1. AI developed
2. Integrate AI and game fully working
3. User tested with human and AI players
4. Final documentation completed and submitted

Schedule

Week 2
Plan out remainder of quarter

- Milestones, deliverables, biweekly schedule, etc.

Begin planning out AI integration

Week 4
Crude AI integration

- Initial implementation, simplistic functionality, potentially user-assisted or strictly using probabilities (not learning)

Week 6
Full AI integration

Week 8
Testing, finalize integration, tweaking

- Improve UI, fix bugs, user testing

Week 10
Finish up documentation and submit!

User Testing

Below is a table of testing results when playing the game with some friends we recruited. First we defined what a “win” was, and this was if a player got a correct rank guess for the game. This is because if someone gets the correct rank guess, they have helped the other players as best as they could and also performed well themselves. We compared a number of games with various configurations in number of human players that could be either experienced or inexperienced, and up to 3 AI bots. We saw that the bots were able to perform at a decent level, that of a player that fully understood the game but might still make some mistakes interpreting the information given to them.

Some notable things we observed is that the game is quite complicated and for four new players, 3 games is not enough to learn how to work together and read into the intention of what other player's guesses are going for. With the addition of an experienced player to explain, it takes a couple games and then they will figure it out.

Testing Data

Configuration	Game 1	Game 2	Game 3	Average
2 exp. players, 2 AI	Lose	Win	Win	Win
2 exp. players, 2 new players	Lose	Win	Win	Win
4 exp. players	Win	Win	Win	Win
1 exp. player, 3 AI	Win	Win	Win	Win
1 exp. player, 3 new players	Lose	Lose	Win	Lose
4 new players	Lose	Lose	Lose	Lose
1 new player, 3 AI	Lose	Lose	Win	Lose

Source Code

https://github.com/bryceacc/high-low