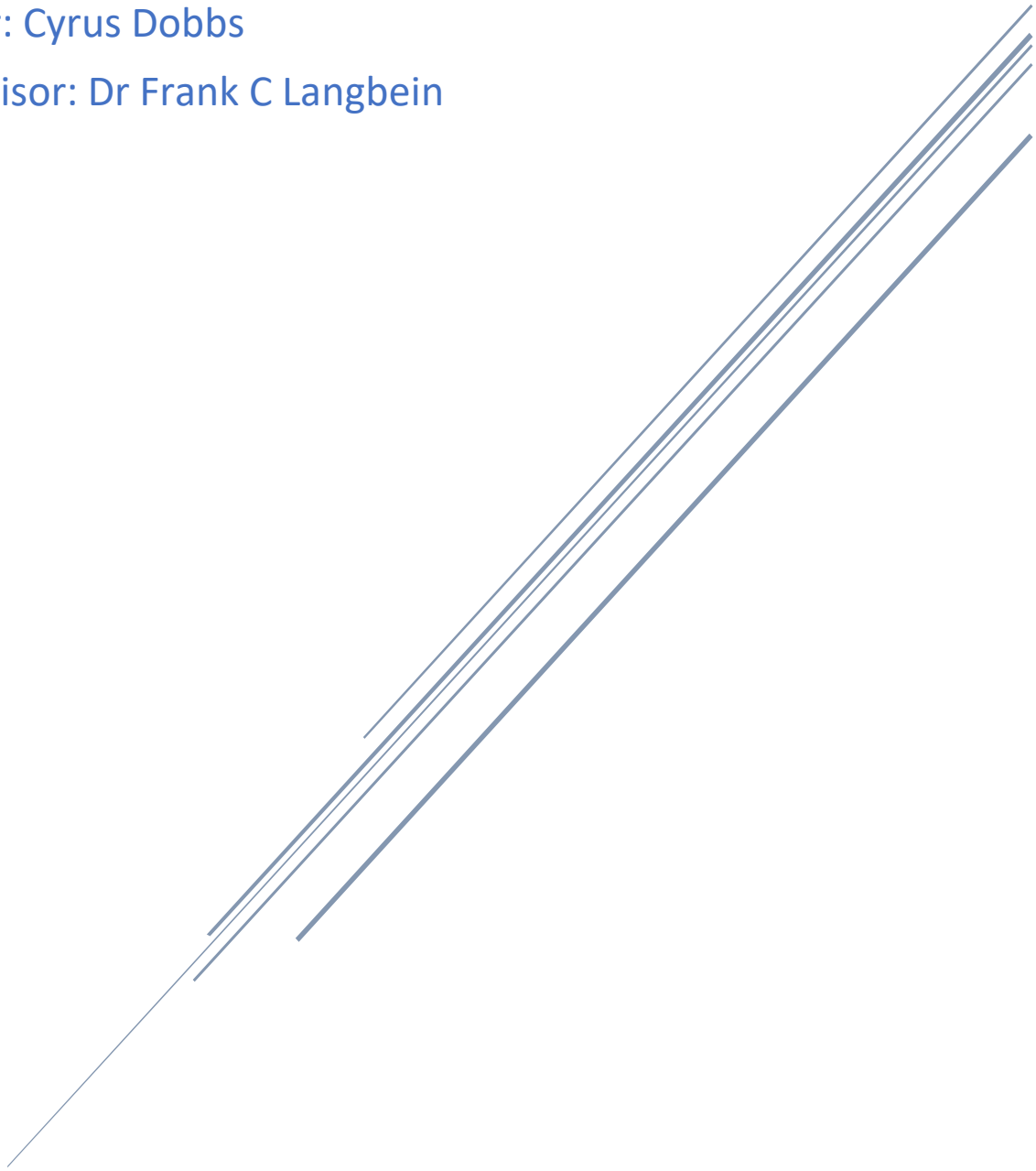
INITIAL PLAN: EFFECTIVENESS OF USING AN INFORMATION SET MONTE CARLO TREE SEARCH AI FOR THE CARD GAME OF 'CASTLE' IN AN ONLINE MULTIPLAYER SETTING

Author: Cyrus Dobbs

Supervisor: Dr Frank C Langbein



CM3203: One Semester Individual Project
40 Credits

Table of Contents

The aim of this project

For this project I have chosen to implement the game of Castle[1]; a card game between 2-5 players taking turns sequentially with the aim of disposing of all their cards. I would like to focus more resources into the AI aspect of the project where I hope to produce at least an ISMCTS based player as this is where most of my interest lies. I also think that it is the part of the project from which I will to develop the most interesting analysis from.
However, I also want to tackle the networking aspect to the point of at least allowing two players to connect with ease and without latency issues during gameplay. I aim to do this by implementing an authoritative game server that utilises Websockets and TCP to connect the players together.
The human players will also be able to play the game against another human or an AI player through a dedicated client-based graphical user interface. I will also produce a text-based interface to run from the console for use when implanting and testing new features as well as to quickly produce stats on the AI's play.
After producing a working system, I hope to evaluate the effectiveness of my AI implementations and networking solution. I will write the game logic, GUI and game server in Java.

Why is Castle interesting for AI?

I think Castle will provide interesting learning opportunities due to its properties of imperfect information and randomness.

- Deterministic & Stochasticity
    - Randomness is present due to the shuffled deck.
    - Imperfect information from the shuffled deck, the opponent's hands and their facedown cards.
- A Heuristic based approach is unlikely to be effective.
    - It is hard to produce heuristics for Castle because it's hard to rank how a particular move will increase a player's chance of winning & decrease the chance of an opponent from winning.

Why ISMCTS?

Due to the very large possible state space of Castle (52 card deck) and the time limit set on a player's turn, an anytime algorithm that can run for as long as specified and return the best move found at that time is very suitable.
Imperfect information games often make game trees more costly to search, increasing both the branching factor and depth of the tree. Many algorithms deal with this by using determinisation - a process of guessing the hidden information and making a decision based on searching separate trees for each guess in the sample of guesses. An example of this is Perfect Information Monte Carlo (PIMC) which treats each determinisation as if it were a game of perfect information and analyses that state.
A more applicable technique to the game of Castle is the use of ISMCTS which utilises Information Sets (IS). An Information Set is the current game state using the information available to the player.
In my Castle implementation, I will build the IS using the cards in the player's hand, the cards discarded by each player and also the discard pile that is added to opponents' hands

[1] en.wikipedia.org/wiki/Castle_(card_game)

when they have no legal moves available to them. Although only some of the cards will be used as information because it isn't realistic that a player would remember every single card played. This could be a factor in how I produce different difficulty levels of the AI.
ISMCTS uses the current information set to limit the combinations of determinisations to the possible states of the game – it then uses a single tree to run an iteration of MCTS[2] for each determinisation (up to a time limit). A move is then decided based off the proportion of times a move is traversed over all the simulations.
However, there are some potential issues with ISMCTS. For one, classic ISMCTS does not utilise any game specific knowledge and this could lead to some AI moves looking unrealistic/illogical to human players. Additionally, in order employ a winning strategy, ISMCTS may need to run for longer than is considered to maintain the fun aspect of the game for human opponents.

Networking

I will use client-server architecture and run the primary game state on the server. The server will be the only trusted part of the system and will validate all moves made by the clients to ensure they aren't cheating. The server will work using WebSockets and will send/receive packets over TCP.
I considered using a RESTful web server as a game server but due to the speed of which the game is played, I decided not to. In order to ensure the validity of moves chosen by opponents the server also needs to check them – this would be hard in a stateless RESTful system.

1. Server sends possible moves and information to be displayed by the GUI to players' clients.
2. The UI will tell players of invalid moves if they try and select them, e.g clicking the 9 of Diamonds when the last card to be played was a 5. The players will select a move send that information back to the server.
3. Server validates the move by checking it against the list of valid moves for that game state.
4. If valid, shares the chosen move with the other players, updates game state and tells the next player to move

[2] https://en.wikipedia.org/wiki/Monte_Carlo_tree_search

Requirements

Primary Requirements

These are the bare minimum I would like to achieve.

- Game Logic
    - An expandable, object-orientated game logic library with high-cohesion & low-coupling.
        - New editions to logic and gameplay should be able to be added with ease.
    - Should play a standard game of Castle.
    - The game should be playable between AI & Human, Human & Human and AI & AI.
    - 2 player capability
- Artificial Intelligence
    - AI player that uses ISMCTS.
    - Online players should be able to play against the AI.
- User interface
    - Should be intuitive for people of all levels of computer literacy to use.
    - I don't aim for the graphics to be to a very high level. However, it must be aesthetically pleasing for its level of basic graphics.
- Networking
    - Players should be able to connect to the game server from their client in order to create/join a game lobby.
    - The game should play without interruptions or errors from the system.

Secondary Requirements

Once all the primary aims have been completed, I will move on to adding some these additional features.

- Game Logic
    - More than 2 players
    - Players can change certain game settings
        - Time given for a turn, castle size, hand size etc
    - Add friends/chat
- User Interface
    - Add animations
    - Produce and implement a more complete theme to the UI
- Networking
    - More than 2 players
    - In game chat
- Artificial Intelligence
    - Difficulty levels to the AI
    - Add knowledge to the ISMCTS AI
    - Add different AI algorithms to compare and contrast

Evaluation

I plan to evaluate my project by comparing the performance of different AI's.

- ISMCTS
- Random decisions
- AI with a rule-based strategy of always picking the lowest ranking legal card
- ISMCTS with game-specific knowledge

Tests could include:

- Win rates with varying time allowed for each move.
- Human evaluation:
    - How realistic are the AI's moves?
    - How fun is the AI to play?

From these tests I will produce tables and graphs to compare, contrast and make assumptions about the different AI's. My evaluation will also test the robustness of the networking aspect.

Time plan

	Focus	Delivered by the end of the week	Meeting with project supervisor (Tuesday 12-1pm)
Week 1	Write plan	Submitted plan.	✔ (30 minutes)
Week 2	Core game logic and console interface	Finished core game logic. Human vs. basic AI player using a basic text-based user interface runs via the console.	✔ (30 minutes)
Week 3	ISMCTS AI Player	Implemented ISMCTS player. (See	✔ (30 minutes)
Week 4	GUI	Game playable through GUI.	✔ (30 minutes)
Week 5	Networking	Server code written and logic added.	✔ (30 minutes)
Week 6	Networking	2 clients can connect to the server and play. (See	✔ (30 minutes)
Week 7	Buffer for catching up or Secondary Requirements	All of primary requirements are completed. System works in full.	✔ (30 minutes)
Week 8	Evaluation	Network stress tests & stats-based AI tests ran, and the results visualised.	✔ (30 minutes)
Week 9	Evaluation	Human user gameplay tests completed, and the results visualised.	✔ (30 minutes)
Week 10	Report	Written most of the first draft.	✔ (30 minutes)
Week 11	Report	First draft submitted for review and feedback.	✔ (30 minutes)
Week 12	Report due 7/5/20	Submitted final report.	✔ (30 minutes)