AUTHORS

Sam James, Sam Medwell, Lukas Rutkauskas and Katie Yang

COUPON FOR TRAVEL

Dynamic Difficulty Adjustment** for **Imperfect Information Games** using **Monte-Carlo Tree Search

AFFILIATIONS

Supervised by Professor Till Bretschneider
University of Warwick

INTRODUCTION

Games are typically at their most engaging when a suitable level of challenge is provided. Dynamic Difficulty Adjustment (DDA) aims to facilitate this by adjusting to the skill level of the player as the game progresses. In this project, we have built an agent based on the Monte-Carlo tree search (MCTS) algorithm and implement DDA modifications, using the popular board game Ticket to Ride as our test bed.

OBJECTIVE

Modify the Monte-Carlo tree search algorithm to develop AI models for two-player Ticket to Ride, a game of imperfect information; implement an application to allow players to play against it.

SYSTEM DESIGN

An adaptation of Ticket to Ride was built in C++, which we named C◎upon for Travel. It allows the player to play against the DDA agent in a riveting two-player game of Ticket to Ride, with two map choices. The back end has been designed to be highly performant in order to keep the game simulation run time to a minimum. The application has been designed to be accessible, intuative and enjoyable to play with.

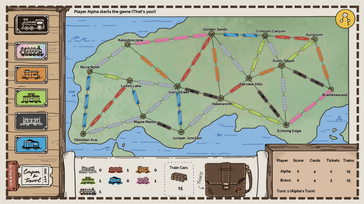

Figure 1. User interface for game application, *Coupon for Travel*.

PROJECT MANAGEMENT

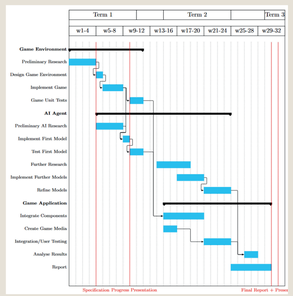

Figure 2. Gantt chart for project schedule.

We followed the Scrum methodology as a general guideline, employing the use of product and sprint backlogs. The team worked in two tracks of development (AI and game dev.) and in 2-week sprints, holding kickoffs at the start and reviews at the end of each sprint.

MONTE-CARLO TREE SEARCH

Monte-Carlo Tree Search (MCTS)

- To adjust to the player's skill level, we first have to be able to beat the player. We chose to implement the MCTS algorithm, which is a tree search algorithm that has been shown to perform well in board games [1]. The algorithm works as follows:

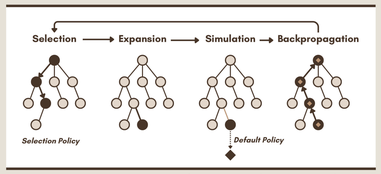

Figure 3. One iteration of Monte-Carlo tree search.

Selection: Starting at the root node, apply the selection policy until the first expandable node.
Expansion: Add child to the expandable node.
Simulation: Simulate the game from the new node using default policy until the end of game.
Backpropagation: Outcome propagates up the tree, updating each node in its path.

- To balance exploration and exploitation, Upper Confidence Bound for Trees (UCT) uses the following equation in the selection policy to select the best child:

$$UCT = \bar{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}}$$

Xj = average payoff at node j
n = number of times j's parent has been visited
nj = number of times j has been visited
Cp = exploration constant

REFINED MCTS STRATEGIES

Modifications for MCTS

T2R is an imperfect information game, but the base MCTS algorithm applies to perfect information games like chess, or go. We implemented two variants of MCTS to deal with this [2]:

- **Cheating MCTS** treats the game as a perfect information game, by accessing normally hidden information ("cheating"). Cheating is OK because we're trying to implement DDA.
- **Information Set MCTS** uses information sets, which are sets of states indistinguishable from the player's point of view, instead of single game states as nodes in the search tree.

Dynamic Difficulty Adjustment (DDA)

- The base DDA algorithm uses **Reactive Outcome-Sensitive Action Selection (ROSAS)** which aims for a score of zero, hence maximising the likelihood of a draw [3].
- **Proactive Outcome-Sensitive Action Selection (POSAS)** builds on this by adding a threshold around zero in which actions are selected at random, promoting proactive play.
- We propose **momentum** as a further modification which aims to reduce the exploitabilty of the agent and create more natural gameplay. It penalises larger swings in game state score to help maintain smooth, consistent strength.

HEURISTIC AGENTS

A number of heuristic agents were developed for evaluating our DDA agent against. These heuristic agents take actions using a ruleset developed based on expert knowledge.

- **Base agent**: Based on the CFBaseAI agent [4], finds the most optimal path by evaluating route costs, then tries to complete that path.
- **Modifications**: Addition of randomness, prioritisation of certain actions (e.g. hoarding cards), as well as mixed agents which switch strategies during the game, resulting in arbitrary changes in its skill level.

TESTING & RESULTS

Round Robin

First, the heuristic agents are tested against each other to produce a relative ranking of difficulty. 100000 simulations are ran for each pairing and start sequence, given that bots can start first or second in a game.

DDA Against Heuristic Agents

Then, we match our DDA against heuristic agents of differing skill levels. We produced some example results for the **ISMCTS** agent using **POSAS** and **momentum**:

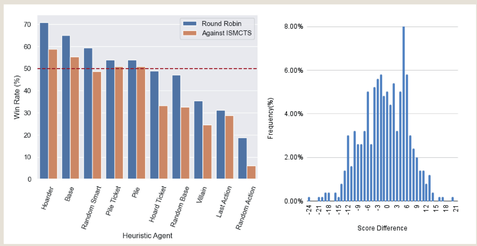

Figure 4. *Left*: Average win rate of each heuristic agent in round robin format in blue + win rate against ISMCTS DDA agent in orange.

Right: Score difference distribution between ISMCTS and the base heuristic agent.

User Testing

Based on results from games played by testers of varying skill level:

- ISMCTS proves to be too weak for DDA to be effective against humans.
- CMCTS DDA results in close matches even for skilled players (within 10 points). However, it typically loses as it tries to keep the score even.

Performance

We achieved average MCTS iteration* times of 11371.57ns for CMCTS variants and 15421.24ns for ISMCTS variants, benchmarked using the DCS cpu-batch partition (nodes with a Intel Xeon E5-2660 @2.6GHz).

*an MCTS iteration is one application of the steps 1-4, as outlined in Figure 3.

CONCLUSION & FUTURE WORK

Cheating MCTS proves to be strong enough against human players for DDA to be effective, and testers found the games enjoyable to play. We argue that giving the agent extra information is acceptable, as its purpose is to maximise enjoyment and not to win. We investigated modifications that would create more human-like gameplay using POSAS and momentum. In theory, the same modifications can be applied to ISMCTS if its performance ceiling can be raised.

REFERENCES

[1] A Survey of Monte Carlo Tree Search Methods, Browne et al., 2012
[2] Information Set Monte Carlo Tree Search, Cowling et al., 2012
[3] Monte Carlo tree search based algorithms for dynamic difficulty adjustment, Demediuk et al., 2017
[4] Reinforcement Learning Agents Playing Ticket to Ride—A Complex Imperfect Information Board Game With Delayed Rewards. Yang et al., 2023
[5] Ticket to Ride, Days of Wonder, 2004