Best Strategy for Balatro Game

Ran Tao

https://github.com/TR1028/15618FinalProject

I am going to implement an algorithm to find the best solution to the current status of a game called Balatro, which was released in Feb, 2024, and gained 97% Positive feedback for over 25k players.

The game is a rouge-like game based on Texas Hold'em, with complex calculation systems to calculate the scores players with certain numbers of play hand or discard.

Challenges:

The primary challenge in parallelizing a decision support system lies in the game's inherent complexity. Although it is based on Texas Hold'em, it only uses the point system in it, but also implemented large amount of joker cards, tarot cards, planet cards, and even spectral cards, with different calculation methods, and each have their own system and usage. The decision-making process is highly dependent on non-deterministic elements, making it difficult to predict and parallelize due to:

Workload variability: Each game state can vary significantly, leading to unpredictable workload distribution among computing resources.

Data dependencies: Decisions are interdependent, complicating the task of dividing the workload into independent tasks that can be processed in parallel.

Divergent execution paths: Each player's decision can lead to a multitude of possible future game states, resulting in a branching problem space.

Goals and Deliverables:

Plan to Achieve:

- Develop a prototype that can analyze a static game state and suggest optimal moves in both sequential and parallel manner, this should include a valid algorithm that can compute the points earning in each situation with probability of sequential moves after this move, and hence predict an optimal solution.
- Have an interface to communicate with inputs and outputs given to the program, the input might be of form of file input, and output might include the optimal solution strategy.

Hope to Achieve:

- Integrate the program into the game environment, no need to manually input data, but can dynamically compute and give advise to the player.
- Further enhance the system behavior to even give advise of choosing special cards with special functionalities.

Demo:

The demo will be shown at final week would be at least a program that can take file input of certain format and generate output accordingly with the optimal solution.

Resources:

I am going to start from scratch with no other starter code. I will heavily rely on Fandom wiki with the stats and usage of each cards, as I do not currently discovered all special cards with their functionalities.

Platform choice:

The first choice of language might be C++, which we have been working on throughout this whole semester, and this language will also provide a sufficient performance for parallel computing.

The second choice might be Java, which has an object-oriented principle. As working with different kind of cards, the special functionalities might cause some trouble in C++, and Java might be helpful.

Schedule:

Week 1: Research through the Fandom Wiki with each cards' functionality.

Week 2: Develop an algorithm to compute the points earning during each situation, including the effects caused by special cards.

Week 3: Continue with the main algorithm design, with a functional algorithm that can handle and compute the solution at each stage, but not necessarily next steps.

Week 4: Having a functional sequential version for computing the optimal solution even considering further steps.

Week 5: Start implementing the parallel version of the algorithm

Week 6: Final week for tests and tuning for the parallel version program.