

CSC4631: Final Project Proposal

Title: Balatro Decision Analysis

Team Members: Alec Nartatez, Tyler Schreiber

Motivation and traditional approaches

Balatro is a rogue-like card game developed by LocalThunk that is built around constructing poker hands to meet or exceed the chip requirement of each Blind (game round). Players are given an 8-card hand and a limited number of plays and discards (4 plays, 3 discards), which they must strategically manage to reach the blind's target score. Early game scoring uses fixed chips and multiplier values for poker hands, shown in Figure 1, and these values scale with the ranks of the cards used. For example, a pair of Kings yields more chips than a pair of Twos, since each card's rank contributes additional base score (blue numbers in Fig. 1) to the total. For simplicity, our implementation of Balatro will not include Jokers or other hand upgrades and will have decreased blind values to compensate for the lack of complicated multipliers. This creates a sufficiently complex sequential decision problem grounded in standard poker structure.

Straight Flush	100 X 8
Four of a Kind	60 X 7
Full House	40 X 4
Flush	35 X 4
Straight	30 X 4
Three of a Kind	30 X 3
Two Pair	20 X 2
Pair	10 X 2
High Card	5 X 1

Figure 1: Level 1 Balatro Hands Base Chips and Multipliers

The formula for the score of a hand is simply the numbers from Figure 1 with the rank of the cards used in the hand contributing to the base chips, which are the number highlighted in blue.

$$\text{Score} = (\text{Base Chips} + \text{Card Ranks}) \times \text{Base Multiplier}$$

For example, a Full House consisting of 3 Kings and 2 Aces would be

$$368 = (40 + (10 + 10 + 10 + 11 + 11)) \times 4$$

Human players typically evaluate all meaningful subsets of their hand, identify immediate high-value combinations (pairs, straights, flushes), and decide whether to

discard toward promising draws. These manual heuristics like: “keep pairs,” “draw for a flush,” “hope for a straight” are approximate greedy decisions that do not incorporate long term reasoning. They do not evaluate the expected benefit of using discards early versus preserving them, nor do they capture how to balance medium value plays against the risk of failing the blind. As a result, even human strategy is usually locally optimal but globally suboptimal.

Our project intends to use Q-Learning to select actions maximizing the probability of clearing the current Blind. To prove its effectiveness, we will be comparing the performance of Q-Learning to both random decision making and simple heuristics-based strategy in this constrained but representative game environment.

AI Algorithms

Q Learning: We will implement a tabular Q-Learning agent that learns action preferences through repeated experience. Since encoding raw 8-card hands in a table is infeasible, we will discretize the state using structural features: number of pairs, presence of triplets, straight or flush potential, highest-rank categories, remaining plays and discards, and progress toward the target score. Rewards are based on chips gained from playing a hand, with large terminal rewards for clearing the blind and penalties for failing it. Over many episodes, the Q-Learning agent learns when to discard, when to play conservative hands, and how to allocate limited plays.

Project Details and Analysis

In this project, we will build a simulator for the simplified Balatro gameplay loop. A blind will be considered “cleared” when the accumulated chip total reaches the blind’s target score before the player exhausts all plays. The Q-Learning agent will interact with the simulator through a unified state and action interface. Performance will be measured using blind-clear rate, average chips scored, number of plays used, sample efficiency, and computational cost. We will generate visualizations including reward-over-time curves, Q-value convergence plots, and comparison tables showing the performance of Q-Learning relative to random and heuristic strategies.

For comparison we are using both a random model and some simple heuristics like: “play only pairs” or “play only straights” to gauge the magnitude of performance increase the Q-learning offers.

Timeline

- Implementing and testing the problem
 - Implement the game loop starting with the representation of the 52-card deck and random card dealing. The hand by encoding 8 card hands, and

- selection of 5 or less cards for plays and discard operations with card replacement.
- Implement the poker-hand scoring module using the base chip and multiplier table shown in Figure 1.
- Track accumulated score, number of plays remaining, and number of discards remaining.
- Test the problem with random simulations and reproducibility with seeded runs.
- Implementing and testing the AI algorithm
 - Implement tabular Q-Learning: First, encode states using a discretization over hand structure (pairs, trips, straight/flush potential), resource counts (plays/discards remaining), and target-score progress.
 - Define a finite action set consisting of playable subsets and discard patterns.
 - Verify correct table updates by applying Q-Learning to small deterministic test environments (e.g., one-play blinds with limited actions).
 - Log Q table growth to ensure state-action pairs are visited and updated as intended.
- Bringing it together
 - Integrate algorithms into the full Balatro simulator.
 - Ensure consistent state and action interfaces for Q-Learning.
 - Run full blind simulations for each algorithm over many episodes using fixed seeds.
 - Evaluate Algorithm Performance.
 - Measure blind-clear rate, average chips scored, and number of plays used across episodes.
 - Measure runtime and sample efficiency (number of simulations required for stable behavior).
 - Plot Q-Learning performance over episodes (reward vs. time, Q-value convergence).
 - Include comparison tables or plots showing differences between algorithms