Events vs. Score based — Optimizing Splendor AI Agents via Value Networks and MCTS Integration

Team Members: Yehao Yan, Qianyun Shen, Xinyan Guo

Splendor Game Rules

Game Components:

1. Gem tokens: 5 colors (white, blue, red, green, black) + gold .
2. Development cards: Three level (I, II, III). Each card has a cost, a permanent gem bonus, and possibly victory points.
3. Noble tiles: Provide victory points when a player meets specific bonus requirements.

Turn Actions:

On each turn, a player may take one of the following actions:

1. Take gem tokens

- Take 3 different colors, or
- Take 2 tokens of the same color (if at least 4 are available).

2. Buy a development card

- Purchase a face-up card from the market or a previously reserved card.

3. Reserve a card

- Reserve a face-up card (or top of a deck) and take 1 gold token.

Scoring and Winning:

1. Development cards and nobles provide victory points.
2. Permanent gem bonuses reduce future card costs.
3. The game ends when a player reaches 15 or more points(Prestige); the player with the highest score wins.

1. Introduction & Motivation

Splendor is a turn-based strategy card game characterized by resource management, partial observability (hidden reserved cards), and stochasticity (card shuffling). Unlike perfect information games like Chess or Go, Splendor presents unique challenges for Artificial Intelligence:

1. Sparse Reward Problem(Score-Based Limitation): Players often receive zero prestige points for the first 10–20 turns while building their engine. Traditional agents struggling to evaluate intermediate states often fail to distinguish between a "good 0-point state" (rich in resources) and a "bad 0-point state".
2. Stochasticity & Planning Horizon: Pure search algorithms like Monte Carlo Tree Search (MCTS) struggle with the high branching factor caused by random card replenishments, leading to short-sighted gameplay.

Objective: This project aims to develop a high-performance Splendor agent by bridging the gap between Statistical Forward Planning and Deep Learning. Crucially, we propose to solve the sparse reward problem by implementing Event-Based Feature Engineering, moving beyond traditional Score-Based representations.

2. Problem Definition & Baseline

2.1 Baseline

A standard MCTS agent that relies entirely on a hand-crafted linear evaluation function:

$$V_{baseline}(S) = w_1 \cdot \text{Prestige} + w_2 \cdot \text{GemCount} + w_3 \cdot \text{DevelopmentPoints}$$

- This is a pure Score-Based agent. It reacts fast based on static board scores but lacks the ability to learn or perceive dynamic tactical shifts.
- It serves as the perfect "Control Group" to measure the effectiveness of the RL model.

2.2 The Proposed Method: Event-Augmented RL Agent

Our approach is inspired by the AlphaZero framework, but differs in that we explicitly incorporate event-based inductive bias to address sparse rewards and stochasticity in Splendor.

We will design a Deep Reinforcement Learning (Deep RL) agent.

1. Architecture: The Neural Network receives not only the State Vector (S_t) but also a dedicated Event Vector (E_t) describing the dynamic changes from the previous turn.
2. Training: The agent will learn the Value Function $V(S, E)$ via self-play or by training against the Baseline agent.

3. Methodology

3.1 Data Generation: The "Mixed-Policy" Simulation

To ensure our dataset covers both diverse scenarios (Exploration) and high-quality gameplay (Exploitation), we will use a existing gym-splendor to simulate 50,000 games using a mixed strategy:

- Agents: Two agents playing against each other.
- Policy: ϵ-Greedy Strategy.
    i. 80% probability: Greedy Move (prioritize buying cards/points).
    ii. 20% probability: Random Move (introduce noise and recovery scenarios).

3.2 Feature Engineering

We will design a feature vector to capture "Game Tempo" that the Baseline misses:

1. Scarcity Event:

- Logic: if Gem_Pile[Color] <= 2
- Significance: Triggers hoarding strategies; increases the valuation weight of that specific color.

2. Aggression Event:

- Logic: if Opponent_Reserved_Card in My_Wishlist
- Significance: Identifies targeted blocking by the opponent; triggers a "Pivot Strategy" signal.

3. Lethal Proximity Event:

- Logic: if Opponent_Score >= 13
- Significance: Forces the MCTS to prioritize defensive branches (e.g., sacrificing a turn to reserve a card the opponent needs).

3.3 Network Architecture

We will utilize a NN (Neural Network) as the Value Network to ensure fast convergence during training:

- Input Layer: Concatenation of $[State\_Vector, Event\_Vector]$.
- Hidden Layers: 2-3 Fully Connected (Dense) Layers with ReLU activation.
- Output Layer: A single scalar $V \in [-1, 1]$ representing the Win Probability.

3.4 MCTS Integration

In the MCTS Selection phase, the Neural Network output will guide the search:

$$UCT(s,a) = \underbrace{Q(s,a)}_{\text{Exploitation}} + \underbrace{C \cdot P(s,a) \cdot \frac{\sqrt{N(s)}}{1+N(s,a)}}_{\text{Neural Exploration}} + \underbrace{\lambda \cdot \text{EventBias}}_{\text{Tactical Innovation}}$$

Where

- λ : A dynamic weight that forces the AI to explore specific branches when high-priority Events are detected.
- $Q(s,a)$: The Mean Action Value.
- $P(s,a)$ (Prior Policy): The probability output by your Neural Network. It represents the AI's "Intuition" about which move looks good before searching.
- $\frac{\sqrt{N(s)}}{1+N(s,a)}$ (Visit Count Decay): $N(s)$ is the total visits to the parent node, and $N(s,a)$ is visits to this specific action. This term decreases as a node is visited more often, encouraging the agent to explore less-visited branches.
- C: An exploration constant (hyperparameter) that balances the influence of the prior (P) against the empirical value (Q).
- EventBias: A signal derived from your Event Logic.

4. Evaluation Plan

4.1 Win Rate Analysis

- Target: The Proposed events-based RL Agent should achieve a >60% win rate against the score-base Baseline.
- Setup: 500 simulated matches (250 as First Player, 250 as Second Player) to eliminate First-Move Advantage bias.

4.2 Ablation Study

To prove that performance gains come from the Event Features and not just the neural network, we will compare:

- Score-based: Trained using only State-Vector.
- Events-based: Trained using State-Vector + Event Vector.We expect this to demonstrate faster convergence speed and a higher final win rate.

Reference:

1. Human-level control through deep reinforcement learning (Mnih et al., Nature 2015)
2. Rinascimento: Optimising Statistical Forward Planning Agents for Playing Splendor (Bravi et al., 2019
3. Rinascimento: using event-value functions for playing Splendor (Bravi & Lucas, 2020)
4. Mastering the game of Go without human knowledge (Silver, D., et al.2017)