Optimization Under Uncertainty

Take-home project: Ludo

Ludo is a dice-based race game related to the French game *petits chevaux*. Model the game, compute optimal strategies for small instances, and investigate how far your methods scale. Parts 1 and 2 form the core project. The mathematical proofs and Part 3 are optional bonuses.

Rules and board size

Choose and describe a precise rule set: entering the board, movement, captures, shared or blocked squares, reaching home, extra rolls, safe squares, and cases with no legal move. Use independent rolls of a fair six-sided die; players observe the roll before choosing a move. Keep your rules consistent across parts and document any simplifications.

Let N be the number of squares on the shared track. Specify the starting squares and home-lane lengths, including how they change with N. The full-size reference has four horses per player, 52 shared-track squares, and a private lane of five squares followed by the final home position.[1] You may begin with smaller boards or fewer horses. State the parameters of every experiment.

Submission

Work in **groups of three people**.

Submit your source code and a report covering the rules, models, algorithms, results, and conclusions by **Wednesday, 6th of January, in a public Git repository**.

In the repo, include a README with dependencies, commands, parameters, and random seeds for reproducing the main experiments.

1 One player

Remove all opponents. Start with all horses in the player's base and finish when all have reached home. **Minimize the expected number of die rolls needed to finish**, counting extra rolls and rolls with no legal move.

1. **Build an MDP model.** Define the states, actions, transition probabilities, cost, initial state, and terminal states of a Markov decision process. Derive the Bellman optimality equation and terminal condition.
2. **Compute an optimal policy.** Solve the Bellman equation numerically for a board of size N. Describe your state representation, algorithm, and stopping criterion.
3. **Study the policy and the computational limits.** Increase N, aiming for the full-size, four-horses game if feasible. Identify properties of the computed optimal policy, supported by examples or numerical evidence.
4. **Bonus: prove a policy property.** Try to recover mathematically the properties of the optimal policy.

[1] An example board and rule set are available at Print & Play Game: Ludo rules.

2 Two players

Introduce a second player, with all horses initially in their respective bases. Stop when one player brings all their horses home. Use opposite payoffs: $+1$ for a win, -1 for a loss, and 0 for nonterminating play. Each player maximizes their expected payoff. If the game terminates with probability one, this is equivalent to maximizing the probability of winning.

1. **Build a stochastic game model.** Formulate a turn-based, zero-sum stochastic game. Define the state, legal actions, chance transitions, and player to move. Express values from the first player's perspective. Account for extra rolls and discuss whether your rules ensure almost-sure termination, or how your method handles possible nontermination.
2. **Solve small instances by dynamic programming.** Derive the optimality equations. Compute the game value and a policy for each player. Captures and extra rolls can create cycles: explain how your method handles them and how you assess convergence. If you impose a finite horizon, state how this changes the problem and how unfinished games are scored.
3. **Explore larger instances.** Increase N and, if initially reduced, the number of horses. Report the largest instances solved, computational costs, and initial-state values. Illustrate the strategies with representative positions.

3 Bonus: four-player self-play

Use the full-size board with four players and four horses per player, stopping at the first winner.

1. Choose a reinforcement learning method, for example deep Q-learning or a policy-gradient method such as PPO.
2. Train your agent through self-play, by having it play against copies of itself. Present your approach and discuss the results.

Hint: Shaping the rewards may help make training easier. You may also report unsuccessful approaches and explain what you learned from them.