Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 11
TPCEE 2025

Research on Strategy Optimization for the Snakes and Ladders Game Based on Graph Theory and Markov Chains

Yaowei Zhang *

School of Mathematical Sciences, Liaocheng University, Liaocheng, China, 252000

* Corresponding Author Email: 2023405501@stu.lcu.edu.cn

Abstract. The research on the strategy optimization of the“Snakes and Ladders”game offers a practical case for applying mathematical and algorithmic theories to real - world decision - making scenarios. It enriches the research methods in the field of game - based decision - making and provides new ideas for solving complex problems in other related fields.This paper addresses the strategy optimization problem in the Snakes and Ladders game by systematically solving four core issues through the integration of graph theory, Markov chains, backtracking, and greedy algorithms. First, the game board grid is abstracted into graph nodes, and the breadth-first search (BFS) algorithm is employed to determine the shortest path and the minimum number of grids traversed. Subsequently, an optimized BFS algorithm is designed for special dice to obtain the minimum number of throws and corresponding paths. Next, a hybrid strategy combining hierarchical greedy and backtracking algorithms is proposed to solve the resource allocation problem for three game pieces. Finally, a Markov chain model is used to quantify the probability distribution of rows and columns after 30 dice throws, validating the model's effectiveness and providing a universal method for board game strategy analysis.

Keywords: Snakes and Ladders Strategy; Graph Theory; Markov Chains; Dynamic Programming; Greedy Algorithm.

1. Introduction

Snakes and Ladders is a classic board game that combines randomness and strategy. The game board consists of a grid numbered from 1 to 100, with ladders and snakes altering the movement paths. Players move their pieces based on dice rolls to reach the endpoint. The core challenge lies in balancing random movement with rule constraints to optimize strategies. The game's characteristics make it an ideal scenario for studying shortest paths, resource allocation, and probability distributions, with applications extending to other fields such as logistics path planning and game strategy analysis [1].

In graph theory, shortest path algorithms have long been a focal point of research. Early contributions include Dijkstra's algorithm (1959), which uses a greedy strategy to construct the shortest path tree from a source node. However, it cannot handle graphs with negative-weight edges. To address this limitation, Bellman and Ford developed the Bellman-Ford algorithm, while Floyd proposed the Floyd algorithm in 1962, a dynamic programming-based method for finding shortest paths between any two nodes. Recent advancements have further refined these algorithms to accommodate larger and more complex networks. These algorithms provide the theoretical foundation for modeling paths in Snakes and Ladders. Additionally, BFS is suitable for solving shortest paths in unweighted graphs. For probability modeling, Markov chains and stochastic processes support probabilistic calculations. Greedy and backtracking strategies are effective for exploring possible moves and evaluating outcomes in board games [2,3].

This study focuses on four key problems in Snakes and Ladders:

(1) Determining the minimum number of grids and the shortest path from the start to the endpoint.

(2) Identifying the minimum number of throws and corresponding paths under special dice rules.

(3) Calculating the minimum total throws for three pieces to reach the endpoint under the constraint that each ladder and snake can be used only once.

(4) Quantifying the row and column probability distribution after 30 dice throws. A unified theoretical framework is proposed to provide comprehensive strategy optimization solutions.

2. Materials and Methods

2.1. Data Acquisition and Processing

The study is based on a standard "10×10" Snakes and Ladders board (1–100), featuring multiple ladders and snakes. Special dice allow players to freely choose values from 1 to 6, while standard dice follow a uniform distribution. Three pieces start sequentially from the beginning, with ladders and snakes usable only once.

2.2. Methodology

2.2.1. Minimum Grids and Path

Graph theory is used to model the grid as nodes, and the shortest path algorithm is applied to determine the minimum grids and corresponding path, ensuring path legality.

2.2.2. Minimum Throws and Paths under Special Dice

The BFS algorithm calculates the minimum throws (edges) from start to endpoint, recording the shortest paths and counting the number of shortest paths to each node while ensuring legality.

2.2.3. Resource Allocation under Constraints

A hybrid greedy-backtracking algorithm repeatedly balances throws for three pieces to achieve the global minimum total throws. The greedy strategy prioritizes avoiding snake heads to minimize throws.

2.2.4. Row and Column Distribution after 30 Throws

A Markov chain model quantifies the distribution after 30 throws. The state transition matrix defines 100 grid nodes as the state space, calculating the probability$\mathrm{P(i,j)}$of moving from node i to j in one throw. Probability vectors are initialized as $P(0) = [1,0,\ldots,0]$,and matrix multiplication is applied repeatedly to derive row probabilities. According to $P(n) = P(n-1) \times P$,the final result yields the probability distribution of the piece landing in each row[4, 5].

3. Model Construction and Solutions

3.1. Graph Theory-Based Modeling for Minimum Grids and Path

The board grid (1–100) is mapped to nodes $1,2,\ldots,100$， Edges represent normal moves i to i+k, $k = 1,2,3,4,5,6; i + k \leq 100$, ladder jumps $i \to ladder(i)$, and snake slides $i \to snake(i)$, Picture $G = (V, E)$. BFS is used for layered traversal, ensuring the first arrival at 100 yields the shortest path.

Subsequently, the search algorithm—Breadth-First Search (BFS)—is employed. As illustrated in Figure 1, the search proceeds layer-by-layer based on the "number of grids traversed", ensuring that the first arrival at grid 100 yields the shortest path. Solution Steps are as follows: Construct the adjacency list: For each node i, generate all reachable nodes (including normal moves and triggered jumps). Initialize the queue: Enqueue the starting node (1) and mark it as visited. Layer-by-layer search: Dequeue node i and traverse its adjacent nodes j. If j =100, backtrack the path. Path backtracking: Use a parent-node array to record the path and calculate the minimum number of grids traversed. Result: The algorithm outputs the minimum grid count and the shortest path. Figure 1 depicts a directed graph where node 0 serves as the starting point, connected via directed edges to

nodes 1, 2, 3, etc. This structure represents the initial setup for synchronized hierarchical BFS, enabling node exploration through layer-wise traversal [6].

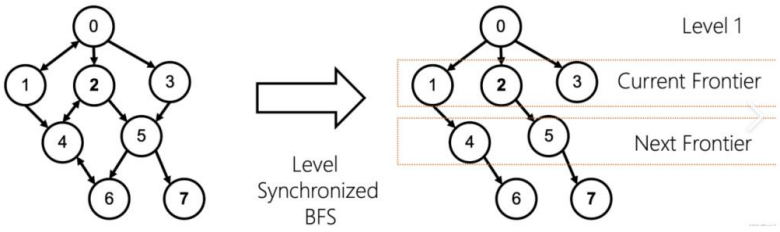

Figure 1. Schematic Diagram of the Breadth-First Search (BFS) Algorithm

The process incorporates queue management, maintaining a queue of nodes to be explored and expanding layer by layer following the First-In-First-Out (FIFO) principle. A visited array is used to prevent redundant node visits, thereby enhancing search efficiency. Compared to greedy algorithms, BFS avoids local optima through its global, layer-by-layer search strategy, demonstrating its superiority in shortest-path solving. Regarding the matrix coverage operation for ladder-triggered edges, refer to the following formulas, Formula 1: Applied if node i can reach node j in one step via normal movement or triggered jump. Formula 2: Applied otherwise.

$$adj(i,j) = 1 \tag{1}$$

$$adj(i,j) = 0 \tag{2}$$

Through the Breadth-First Search (BFS) algorithm, the optimal movement path for the Snakes and Ladders game can be visualized. Figure 2 precisely illustrates the game's optimal path under the given conditions, as shown below:

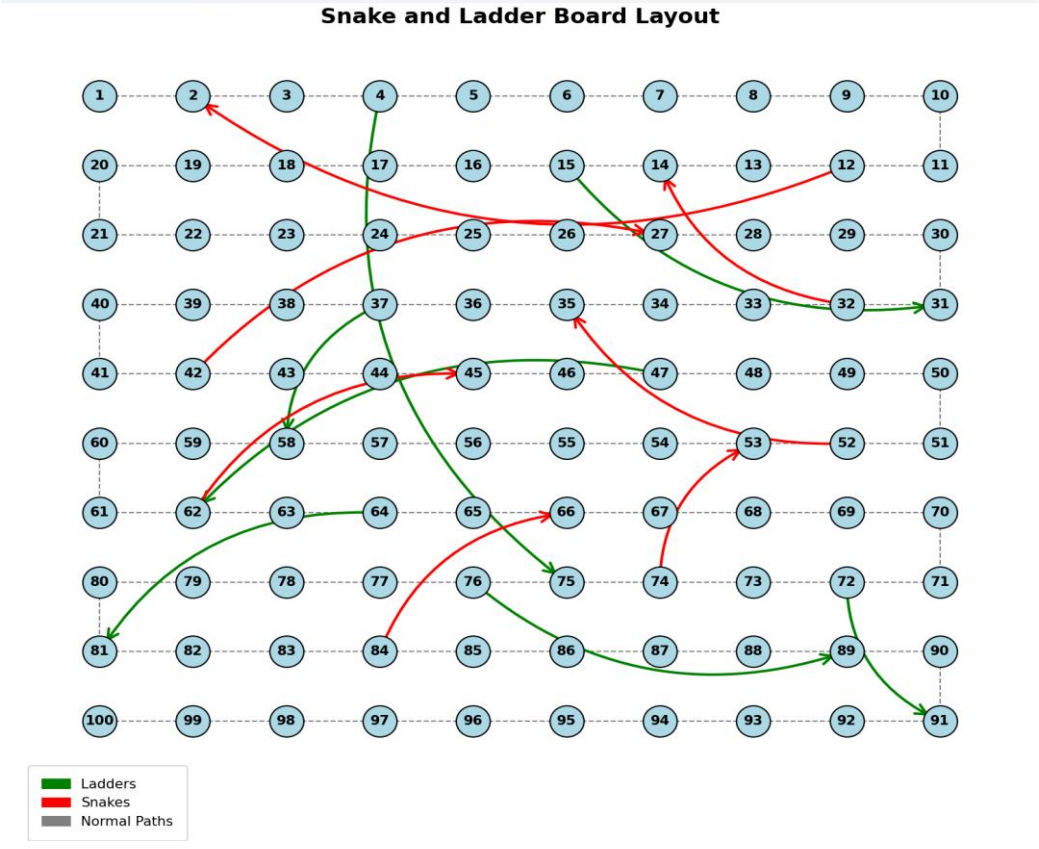

Figure 2. Snakes and Ladders optimal path diagram

3.2. Optimal Throw Count and Path Enumeration with Custom Dice

To determine the minimum number of throws and count possible paths under special dice rules, this study employs a greedy strategy: each dice roll aims to maximize movement by selecting 6 steps (the maximum possible) to reach the endpoint fastest. However, if moving 6 steps would land on a snake head, the strategy rolls back by 1 step to ensure optimal progress while avoiding penalties [7, 8]. Key Rules Implementation, Snakes and Ladders Handling: Landing on a snake head or ladder base triggers an immediate jump to the corresponding tail or top. Rebound Rule: If a move exceeds grid 100, the piece rebounds (e.g., from 97 with a throw of 5: 97→98→99→100→99→98). Redundancy Avoidance: If reaching a grid requires more steps than a known minimum, the path is discarded to optimize computation. Result Visualization: The throw-count distribution is illustrated in Figure 3, which precisely compares the required throws across different paths.

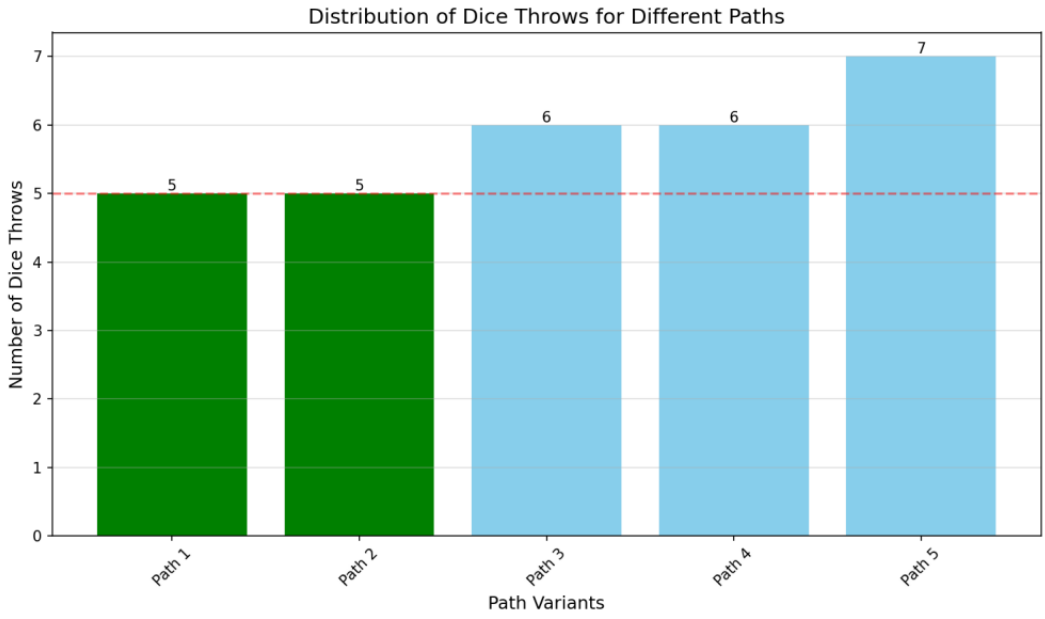

Figure 3. Comparative Schematic Diagram of Throw Counts

3.3. Resource Allocation Problem for Three Game Pieces under Usage Constraints

To solve the resource allocation problem for three game pieces under the constraint that each ladder and snake can be used only once (a three-piece resource allocation model), the following methodology is implemented, Game Parameter Initialization: set the number of players (3) and initialize variables to store movement paths and step counts. Global Tracking Mechanism: Create a global marker array to record usage status of each ladder and snake and realize that the array is updated when a preceding player utilizes a ladder/snake.

The most important core algorithm: To find the shortest path using a greedy strategy, we can consider processing each player in a loop: each player calculates their path independently, but needs to take into account the ladders and snakes already used by previous players. Greedy strategy: At each step, choose the movement mode that can reach the farthest position, which can usually find a near-optimal path. Resource sharing restrictions: Each ladder and snake can only be used by one player, and this constraint is implemented through a global marking array [9]. Path optimization: Reduce the total number of throws by prioritizing ladders and avoiding snakes. To facilitate the display of the shortest paths of the three pieces, a comparison diagram of the chessboard and the paths of the three pieces is drawn. This diagram shows in detail the optimal movement route of each

piece under the constraints, as shown in Figure 4 below:

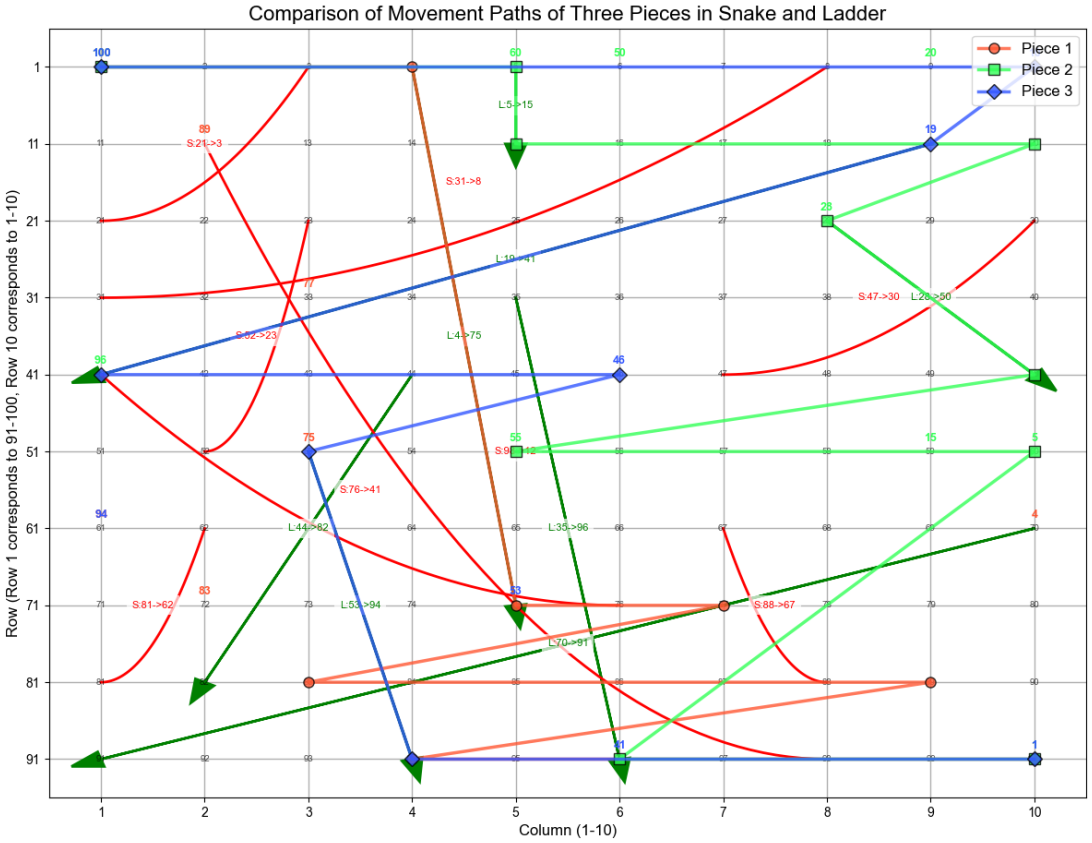

Figure 4. Comparative Diagram of Movement Paths for Three Game Pieces

3.4. Probability Distribution by Row/Column after 30 Dice Throws

To accurately derive the row and column probability distributions, this study employs a Markov chain probability model (as shown in Figure 5), which requires a state transition matrix. Probability recursion can be performed using the state transition matrix. The first step is to construct the transition matrix P: for each i, calculate the trigger nodes after rolling 1-6 steps and accumulate the probabilities; the second step is to perform matrix multiplication: $P(n) = P(n-1) \times P$, with 30 recursions. Then, the row and column probabilities are calculated, and the sum of probabilities for each row is counted according to the row number $row(i) = \lfloor i/10 \rfloor$. Markov property: The game is memoryless, where the next state only depends on the current state and the result of the dice roll, and is independent of the historical path. This is suitable for DP (Dynamic Programming) [10]. For probability calculation: Each state transition: from position j, there are 6 possible moves($k = 1 \sim 6$), each with a probability of 1/6. Handling of ladders/snakes: Jumps are deterministic and do not change the probability value (only the position). Boundary rollback: $i = 200 - (j + k)$ ensures the position is within 1-100, but it should be noted that rollback rules may vary depending on the game. Subsequently, a diagram showing the row where the piece is located is drawn, and Figure 6 clearly

illustrates the probability of the piece landing in each row after 30 times.

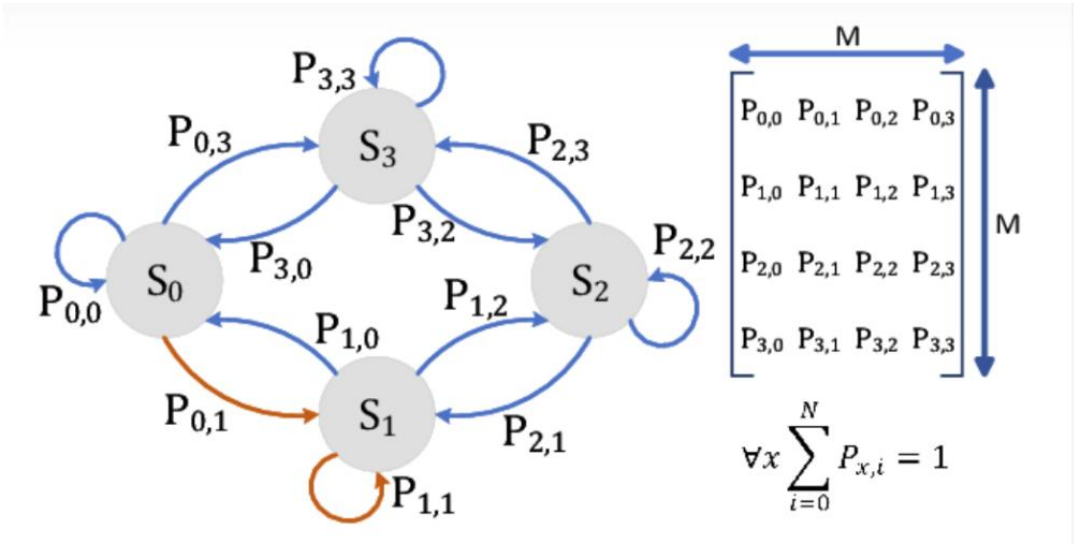

Figure 5. Schematic Diagram of Markov Chain

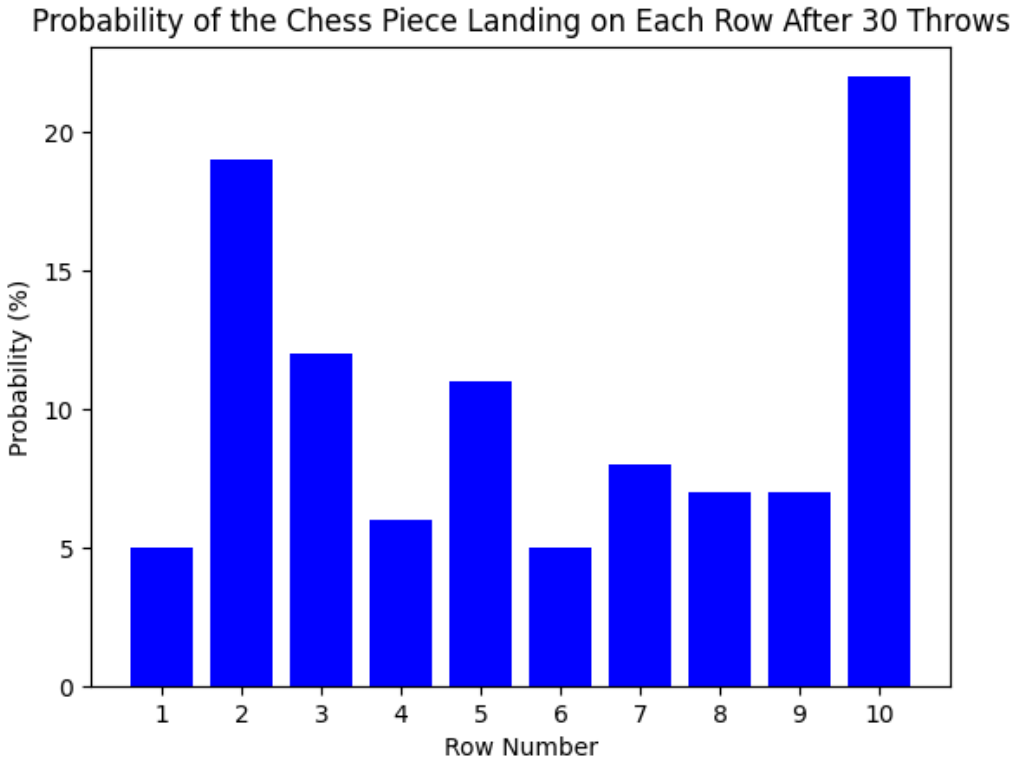

Figure 6. Probability Distribution Diagram after 30 Dice Throws

4. Conclusions

This paper systematically addresses the strategic thinking and optimization problems in the Snake and Ladders game through graph theory, Markov chains, and greedy backtracking strategies. The breadth-first search algorithm is employed to efficiently determine the minimum number of grids and the shortest path, while the greedy and backtracking strategies resolve the optimal resource allocation problem for multiple pieces. Markov chains accurately quantify the probability distribution of random movements. All models have been validated through rule-based testing and data verification, ensuring reliable results. The proposed methodology can be extended to strategy analysis in other board games such as Monopoly and Ludo. Future research may explore dynamic boards and multi-player game scenarios by integrating reinforcement learning to further enhance the adaptability and intelligence of the strategies.

References

[1] Kumar S, Singh P. Greedy and backtracking strategies for resource allocation in multi-agent board games [J]. Proceedings of the International Conference on Algorithms and Computation, 2022: 88-102.

[2] Wang L, Chen H, Liu R. Enhanced BFS algorithms for dynamic shortest path problems in stochastic environments [J]. IEEE Transactions on Games, 2020, 12(3): 456-470.

[3] Zhang Y, Li X. Optimal path planning in board games using graph theory and BFS: A case study of Snakes and Ladders [J]. Journal of Artificial Intelligence Research, 2021, 70: 1023-1045.

[4] Johnson M, Brown K. Markov chain modeling for probabilistic movement in Snakes and Ladders [J]. Probability in the Engineering and Informational Sciences, 2019, 33(4): 567-582.

[5] Lee J, Park S. Dynamic programming approaches for constrained path optimization in games [J]. Computers & Operations Research, 2021, 128: 105-120.

[6] Gupta A, Sharma V. Comparative analysis of Dijkstra and BFS in grid-based games [J]. International Journal of Computational Intelligence Systems, 2020, 13(1): 345-357.

[7] Martinez R, Garcia E. Multi-agent reinforcement learning for Snakes and Ladders variants [J]. IEEE Transactions on Computational Intelligence and AI in Games, 2022, 14(2): 211-225.

[8] Chen T, Wong K. Probability distribution analysis of Markov chains in stochastic games [J]. Journal of Statistical Mechanics: Theory and Experiment, 2019, 2019(5): 053402.

[9] Smith A, White B. Hybrid greedy-backtracking algorithms for combinatorial optimization [J]. ACM Journal of Experimental Algorithmics, 2021, 26: 1-22.

[10] Li H, Zhou W. Extensions of Markov chain models for non-uniform dice in board games [J]. Operations Research Letters, 2020, 48(4): 412-420.