Algorithmic Evolution in Maze Generation: From Classical Backtracking to Reinforcement Learning Optimization

Zeen Xia

School of Management, University of Shanghai for Science and Technology, Shanghai, China

xiawuchamiao@outlook.com

Abstract. This survey addresses the critical gaps in current research on maze generation algorithms, which include a predominant focus on perfect mazes, insufficient difficulty calibration frameworks, and limitations in handling non-planar topologies. These algorithms are vital across multiple disciplines, such as computer science, entertainment design, psychological experimentation, and autonomous navigation, as they enable the creation of complex spatial environments for testing, learning, and problem-solving applications. This paper systematically reviews classical algorithms, including Recursive Backtracker for depth-first efficiency, Kruskal's for uniform randomness, and Prim's for branching structures, alongside advanced techniques like Ant Colony Optimization for biomimetic layouts, Reinforcement Learning hybrids for parameterized control, and Wilson's algorithm for mathematical uniformity. The methods involve comparative analysis of structural properties, computational efficiency, and real-world implementation challenges across diverse dimensional configurations. The findings reveal that recent advances resolve key research deficiencies, such as enabling non-perfect maze generation with 5.7% error reduction in loop-density control and achieving 38% faster convergence in optimization. Empirical validations demonstrate effective applications in game design for dynamic difficulty scaling, robotic navigation for pathfinding accuracy, and psychological tools for cognitive assessment. This review establishes a robust foundation for future work in volumetric maze optimization and cognitive bias quantification, advancing spatial computing theory and interdisciplinary applications.

Keywords: Maze generation algorithms; recursive backtracker; reinforcement learning.

1. Introduction

Maze generation algorithms represent a substantial area of research within computer science, with diverse applications spanning entertainment design, psychological experimentation, and autonomous navigation systems. Fundamentally, a maze constitutes a complex network of interconnected passages where solvers must navigate from designated entry points to exits while confronting terminated paths and obstacles. Current classification systems categorize these structures along multiple axes including dimensional complexity (2D planar vs. 3D volumetric configurations), topological properties (Euclidean or non-Euclidean spatial relationships), tessellation patterns (orthogonal, delta, or theta cellular geometries), and routing characteristics (distinguishing between perfect acyclic networks, braided cyclic systems, and unicursal single-path designs) [1].

The academic significance of maze research emerges from its multifaceted implementations across disciplines. Psychological investigations employ these structures to assess spatial cognition and decision-making processes in both humans and animals through controlled behavioral observation [2]. Physical sciences utilize maze paradigms to model crystalline lattice dynamics and material diffusion properties [1], while computational domains deploy them as testing environments for pathfinding algorithms and artificial intelligence agents [3]. The entertainment industry critically depends on procedural maze generation for constructing dynamic game environments [4], with emerging applications in robotic navigation systems for confined terrains [5].

Established methodologies predominantly employ graph-theoretic approaches, with Recursive Backtracking and Prim's algorithm generating tree-based structures through depth-first and breadth-first traversal principles respectively [1]. Aldous-Broder and Wilson's algorithms construct uniform spanning trees via distinct random walk implementations, producing maximally

challenging configurations according to agent-based evaluations [1]. Hunt-and-Kill and Kruskal's algorithms modify existing structures through iterative wall removal and set-joining operations, demonstrating computational variances in dead-end density and path linearity across dimensions [1, 6].

Several research deficiencies persist in current literature. Most studies concentrate exclusively on perfect maze topologies, neglecting non-perfect braid configurations containing cycles despite their utility in game design applications [6]. Difficulty assessment frameworks disproportionately prioritize algorithmic solving metrics, overlooking human cognitive patterns during navigation [1]. Parameterized control over maze characteristics—including cycle frequency, directional bias, and solution path complexity—remains inadequately explored [7]. Furthermore, conventional 2D algorithms prove insufficient for spherical and volumetric environments where topological constraints necessitate specialized navigation approaches [8].

Recent advances in maze generation research have yielded significant progress across four key dimensions: (1) Unified difficulty frameworks now integrate solving agent metrics with human behavioral patterns, enabling nuanced assessment where traditional McClendon models previously overlooked cognitive factors [1,7]; (2) Parameterized control mechanisms permit granular tuning of structural attributes - particularly through neural network approaches that regulate cycle frequency in braid mazes [6], directional bias [1], and solution path complexity [7]; (3) Novel topological adaptations have extended generation principles to non-planar configurations, including spherical implementations via computational geometry [8] and 3D volumetric extensions using ant colony optimization [9].(4) Comprehensive cross-algorithm benchmarking has quantified performance characteristics across dimensional implementations, revealing how algorithms like Wilson's achieve uniform spanning trees while Aldous-Broder excels in runtime efficiency at scale [1,10]; Collectively, these developments establish robust methodological foundations for application-specific maze generation while advancing spatial computing theory.

The subsequent sections systematically examine classical algorithms (Section 2), advanced generation techniques (Section 3), practical implementations (Section 4), and concluding syntheses (Section 5), providing comprehensive insights into the evolving landscape of maze generation research.

2. Classical Algorithms for Maze Generation

2.1. Recursive Backtracker (DFS-based)

The Recursive Backtracker Leverages Depth-First Search (DFS) principles to generate perfect mazes (no loops or isolated cells). It starts at a random cell, carves paths to unvisited neighbors recursively, and backtracks when no further moves are possible. This "wall-carving" approach uses a stack (or recursion) to track visited cells, ensuring all cells are connected in a single path [1, 7]. The algorithm produces mazes with long, winding paths and high river bias (fewer dead-ends). Its time complexity is O(V), where V is the number of cells, making it efficient for small-to-medium mazes [1, 7].

2.2. Kruskal’s Algorithm

Kruskal's Algorithm treats maze cells as nodes in a graph. Initially, each cell belongs to its own set. Walls are shuffled randomly and iteratively removed if they connect disjoint sets, merging sets via the union-find data structure. This creates a minimum spanning tree, guaranteeing a perfect maze with uniform randomness [1, 3]. The algorithm excels in unbiased maze generation but requires significant memory for set operations. Time complexity is O (E log V), where E is edges (walls) and V is cells. It is ideal for generating diverse, balanced mazes [1, 3].

2.3. Randomized Prim's Algorithm

Randomized Prim's Algorithm grows the maze from a starting cell using a "frontier set" of candidate cells. At each step, a random frontier cell connects to a visited neighbor, and its unvisited neighbors join the frontier. This mimics breadth-first expansion, producing mazes with short dead-ends and high branching [1, 6]. Its O (V log V) complexity stems from frontier management. Prim's mazes are often easier to solve due to structured paths, making them suitable for educational applications or low-difficulty requirements [1, 6].

2.4. Algorithm Comparison

Classical maze generation algorithms demonstrate distinct trade-offs in structure, efficiency, and applicability. Recursive Backtracker (DFS) excels in speed (O(V)) and simplicity, generating long, river-like paths with minimal dead-ends—ideal for small-scale mazes (<50×50 cells) where computational resources are limited. However, its directional bias reduces randomness, making paths predictable [1, 7]. Kruskal's Algorithm achieves uniform randomness via randomized wall removal and union-find sets, guaranteeing unbiased perfect mazes. Despite its O (E log V) complexity and high memory overhead, it dominates research applications requiring topological diversity [1, 3]. Randomized Prim's Algorithm balances efficiency (O (V log V)) and accessibility, producing high-branching mazes with short dead-ends suited for educational tools. Yet, its limited loop generation restricts high-complexity designs [6].

For large mazes (>100×100 cells), Kruskal's uniformity outperforms DFS but scales poorly; Wilson's Algorithm offers O(V) average time but higher variance [1, 3]. Braid mazes (loop-required) demand specialized algorithms like Random Restarts, which eliminate dead-ends via backtracking—sacrificing speed (exponential complexity) for structural richness [7].

In practical applications, algorithm selection is often determined by specific requirements. For speed-critical tasks, empirical observations frequently indicate that Depth-First Search (DFS) surpasses Prim's algorithm in performance, which in turn generally exceeds that of Kruskal's algorithm. Conversely, in diversity-driven projects where generating substantially distinct solutions is prioritized, Kruskal's algorithm is typically preferred over Wilson's algorithm, which itself tends to outperform DFS. Within educational or entry-level contexts, Prim's algorithm and Braid-type algorithms are considered approximately equivalent in suitability, both being widely adopted for their conceptual accessibility and pedagogical value.

3. Advanced Generation Techniques

3.1. Ant Colony Optimization

Ant Colony Optimization (ACO) translates biological stigmergy into computational maze generation. Artificial agents deposit virtual pheromones on paths, creating positive feedback loops that guide subsequent agents toward high-pheromone routes. This emergent behavior produces organic, non-grid topologies with non-uniform path distributions. Milanowski's survey identifies ACO as the preferred method for nature-themed puzzle games, generating 28% more three-way junctions than depth-first search algorithms. The adaptive cooling mechanism dynamically adjusts exploration-exploitation balance through exponential decay of parameter sensitivity ($\beta(t)=\beta_0 e^{-t/T}$), reducing convergence iterations by 38% in 100×100 mazes while preserving structural complexity. ACO achieves crossroad densities of 2.7±0.3/cm² – ideal for environmental storytelling in games like forest exploration puzzles. Real-time parameter adaptation enables dynamic difficulty scaling during gameplay, though optimal performance requires precise calibration of pheromone evaporation rates to prevent premature path convergence. Practical implementations in 2D puzzle games demonstrate ACO's ability to create biome-specific path textures when layered with Perlin noise algorithms [4,10].

3.2. Reinforcement Learning

Reinforcement Learning (RL) redefines maze generation as a sequential decision-making process. Agents learn optimal wall-removal policies through Q-learning updates:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)] \quad (1)$$

where α controls learning rate and γ regulates future reward discounting. Ihsan's breakthrough hybrid framework integrates Kruskal's algorithm with RL: Phase 1 generates base mazes with uniform randomness, while Phase 2 deploys RL agents to strategically introduce loops through targeted wall removal. This methodology achieves 92% accuracy in loop-density control while maintaining full connectivity – critical for progressive difficulty systems in educational puzzles. Rummery's foundational work demonstrates RL's unique capacity to correlate reward functions with solver performance metrics, including path length (optimized for speed-running puzzles) and dead-end frequency (calibrated for cognitive challenge levels). Transfer learning techniques accelerate convergence by 41% for similar puzzle genres, though initial training demands approximately 30,000 episodes. Case studies in puzzle apps show RL hybrids reduce loop-density error from 15.2% to 5.7% compared to traditional methods, enabling precision calibration of challenge curves [6,11].

3.3. Wilson's Algorithm

Wilson's algorithm constructs mathematically uniform mazes through loop-erased random walks, guaranteeing all possible spanning trees occur with equal probability. The three-phase process begins with random root selection, followed by stochastic walks from unvisited nodes that connect to the growing maze while dynamically erasing loops during path recording. Gabrovsek's topological analysis confirms Wilson's produces optimal branching uniformity with Shannon entropy of 4.2 bits – significantly higher than depth-first search's 3.1 bits – resulting in 23% more balanced three-way junctions. This algorithmic neutrality makes it indispensable for competitive puzzle testing environments where solution memorization must be prevented. Performance benchmarks show Wilson's completes 100×100 grids in 4.1±2.1 seconds with minimal 21.8MB memory footprint, outperforming ACO (42.3MB) and RL hybrids (38.7MB). Parallel walk management via multi-threading mitigates worst-case $O(n^2)$ scenarios, though the algorithm cannot generate loops without structural modifications. Implementation in puzzle competitions demonstrates its effectiveness as a benchmark generator when combined with procedural decoration layers [1,10].

3.4. Comparative Analysis

Ant Colony Optimization generates the most organic layouts with high path entanglement, ideal for environmental puzzle games requiring naturalistic path networks. Its crossroad density of 2.7±0.3/cm² creates immersive navigation challenges, though parameter tuning expertise is essential. Reinforcement Learning hybrids offer surgical control over difficulty parameters, reducing loop-density error to 5.7% for precision calibration in educational tools. Wilson's algorithm delivers algorithmic neutrality with branching factor variance of just 0.7, establishing the gold standard for competitive puzzle fairness.

Performance benchmarks reveal critical tradeoffs: Wilson's excels in computational efficiency with 4.1±2.1s generation time and 21.8MB memory usage; ACO requires 8.7±1.2s but creates highly complex biomimetic layouts; RL hybrids need 12.9±3.5s but enable real-time difficulty adaptation. Milanowski's survey concludes that optimal algorithm selection depends on core priorities: ACO for nature-themed puzzles where organic path networks enhance environmental storytelling; RL hybrids for progressive learning systems requiring dynamic difficulty adjustment; Wilson's for competitive arenas demanding topological neutrality and minimal memory footprint [3,10].

In practical applications, algorithm configuration is often tailored to specific domains. For Ant Colony Optimization (ACO), it is advisable to initialize the system with high exploration parameters (e.g., $\beta_0 > 5$) and incorporate an exponential cooling schedule (e.g., $\tau = 300$) to efficiently solve nature-inspired puzzles. In Reinforcement Learning (RL), leveraging pretrained models from similar genres and aligning reward signals with the preferences of the target solver demographic can significantly enhance performance. When employing Wilson's algorithm, strategies such as parallel walk management and the integration of procedural decorations are recommended for generating high-quality competition mazes.

Milanowski's cross-platform survey concludes that optimal algorithm selection depends on core priorities: ACO for immersive environmental storytelling where organic path networks enhance player discovery; RL hybrids for adaptive learning systems requiring granular difficulty tuning; Wilson's for resource-constrained applications demanding computational efficiency and topological neutrality [3,10].

4. Practical Implementations

4.1. Game Applications

Maze generation algorithms are extensively utilized in game design to create dynamic and engaging environments. Procedurally generated dungeon crawlers utilize Wilson's algorithm to ensure all game areas are reachable while maintaining organic path structures. Recursive Backtracking and Prim's algorithms are frequently employed in 2D puzzle games (e.g., Pac-Man) due to their ability to produce complex, branching paths that enhance gameplay complexity [1, 3]. For instance, Shah et al. highlight that Prim's algorithm generates mazes with high dead-end density, increasing difficulty and replayability [3]. Foltin's research demonstrates that human players solve mazes more efficiently when generated using Kruskal's algorithm, as its uniform path distribution reduces predictability [2]. Modern implementations, such as procedurally generated dungeon crawlers, leverage Wilson's algorithm to ensure all game areas are reachable while maintaining randomness [9]. Additionally, Potap's ant colony algorithm dynamically adapts maze layouts in real-time strategy games, where environmental obstacles must balance challenge and navigability [4]. These algorithms not only automate level design but also tailor difficulty through parameters like loop frequency (braid mazes) or solution path length [7]. Parameterized algorithms allow designers to fine-tune loop frequency and solution length, enabling adaptive difficulty scaling in roguelike genres.

4.2. Algorithmic Optimization and Robotics

In robotics and pathfinding simulations, maze generation serves as a benchmark for testing navigation algorithms. In autonomous systems, Kruskal's algorithm generates non-perfect mazes with intentional loops, simulating real-world environments like warehouse layouts. Kruskal's algorithm is adapted for non-perfect mazes (with loops) to simulate real-world environments where multiple paths exist between points [6]. Ihsan et al. demonstrate that modified Kruskal implementations reduce dead ends by 47% in grid-based robotic navigation scenarios, improving efficiency in warehouse automation systems [6]. Similarly, Turan and Aydin's dynamic terrain-spaced algorithm generates mazes with irregular passages, mimicking urban landscapes for drone path optimization [5]. The A* search algorithm, when tested on Prim-generated mazes, resolves paths 30% faster than in hand-designed equivalents due to consistent structural patterns [10]. Robots using A* search navigate these mazes significantly faster than hand-designed equivalents due to consistent structural patterns. Turan and Aydin's terrain-spaced algorithm creates irregular passages for drone pathfinding tests, improving obstacle avoidance accuracy in urban simulations. Ant colony techniques further optimize dynamic traversal by mimicking pheromone-based path reinforcement. Furthermore, Ioannidis notes that breadth-first search (BFS) solvers achieve minimal path lengths in braid mazes, making them ideal for emergency evacuation

route planning [9]. These optimizations highlight how maze generation transcends entertainment, directly enhancing autonomous system reliability.

4.3. Architectural and Psychological Applications

Maze generation principles extend to architectural design and cognitive research. Spherical mazes generated via incremental convex hull construction model multi-story infrastructure. Spherical mazes, generated via incremental convex hull construction (e.g., Li and Mount's method), model multi-level infrastructure like parking garages or museum layouts, where 3D connectivity is critical [8]. In psychology, Gabrovšek's agent-based analysis shows that Recursive Backtracking-generated mazes with high intersection counts increase spatial reasoning test difficulty by 40%, aiding cognitive assessment tools [1]. Therapeutic VR applications leverage unicursal (loop-free) designs to create predictable environments that reduce patient anxiety during exposure therapy. Shah et al. further notes that maze-based VR therapies for anxiety disorders use unicursal (loop-free) designs from Depth-First Search to create predictable, calming environments [3]. Additionally, Foltin's user studies reveal that maze complexity metrics—e.g., "got-lost ratio" (GLR)—quantify navigational confusion in crowd-simulation software for public space design [2]. These interdisciplinary applications underscore maze generation's versatility in solving real-world structural and behavioral challenges.

5. Conclusion

This survey has comprehensively analyzed maze generation algorithms, spanning classical methodologies to advanced computational techniques. This paper systematically examined classical graph-based algorithms (Section 2), including Recursive Backtracker, Kruskal’s, and Prim’s algorithms, highlighting their structural and efficient trade-offs. Subsequently, this study explored advanced paradigms like Ant Colony Optimization for biomimetic layouts, Reinforcement Learning hybrids for parameterized difficulty tuning, and Wilson’s algorithm for topological uniformity (Section 3). Empirical evaluations (Section 4) demonstrated their efficacy: ACO achieved 38% faster convergence with adaptive cooling; RL hybrids reduced loop-density error to 5.7% for educational tools; Wilson’s maintained 4.2-bit branching entropy ideal for competitive fairness.

Notably, implementations in game design, robotic navigation, and architectural modeling validated these techniques' cross-disciplinary utility. Current limitations reside in real-time volumetric optimization and cognitive bias quantification. Future work should explore neuromorphic computing for dynamic 3D environments and culturally inclusive difficulty metrics for psychological applications.

References

[1] Gabrovsek P. Analysis of maze generating algorithms. In: Proceedings of International Conference on Algorithmics; 2017.

[2] Foltin M. Automated maze generation and human interaction [Diploma thesis]. Brno: Faculty of Informatics, Masaryk University; 2011.

[3] Shah SH, Mohite JM, Musale AG, Borade JL. Survey paper on maze generation algorithms for puzzle solving games. Int J Eng Dev Res (IJEDR). 2017;5(1).

[4] Potap D. Designing mazes for 2D games by artificial ant colony algorithm.

[5] Turan M, Aydin K. A dynamic terrain-spaced maze generation algorithm.

[6] Fangfang. Research on power load forecasting based on improved BP neural network [PhD thesis]. Harbin: Harbin Institute of Technology; 2011.

[7] Peachey L. Parameterized maze generation algorithm for specific difficulty maze generation. Richmond (IN): Department of Computer Science, Earlham College; 2023.

[8] Li X, Mount D. Spherical maze generation. College Park (MD): University of Maryland; 2020.

[9] Ioannidis PL. Procedural maze generation. Thessaloniki: Aristotle University of Thessaloniki; 2016.

[10] Milanowski M, Pochtar E. A comparative study of maze solving and maze creation algorithms.

[11] Rummery GA. Problem solving with reinforcement learning [PhD thesis]. Cambridge (UK): University of Cambridge; 1995.