Performance Evaluation for Various Artificial Intelligence Algorithms in Pac-Man Game Environment

Eren Demirtaş
201401020
TOBB ETÜ
Ankara, Turkey
edemirtas@etu.edu.tr

***Abstract*—In this project, we present a simulation framework for the classic Pac-Man game, in which both Pac-Man and the ghost characters are controlled using different artificial intelligence (AI) search algorithms. The goal is to compare the performance of these algorithms under identical game conditions and evaluate their impact on the gameplay, particularly in terms of survival duration, coin collection efficiency, and win rates. The Pac-Man character is guided through the maze using algorithms such as A* Search, Breadth-First Search (BFS), Depth-First Search (DFS), Genetic Algorithm (GA), and a Decision Tree-based model, while the ghosts use a selection of these methods to chase Pac-Man. A real-time graphical user interface is implemented using the Pygame library, allowing for visual tracking of character movements and outcomes. Extensive simulations were conducted to benchmark each algorithm combination across multiple metrics. The results show that Pac-Man controlled by A* or BFS against GA-controlled ghosts achieved the highest win rates (63% and 61%, respectively), indicating the advantage of heuristic search when faced with stochastic adversaries. These findings provide insight into the strengths and limitations of each approach in dynamic pathfinding and reactive decision-making environments.**

I. Introduction

Artificial intelligence (AI) has become an essential component in modern game development, enabling non-player characters to exhibit intelligent behavior through various pathfinding and decision-making strategies. Classical games such as Pac-Man provide an ideal platform for evaluating and comparing the performance of these AI methods in dynamic, rule-based environments.

This project focuses on developing an AI-powered simulation of the Pac-Man game, where both the protagonist (Pac-Man) and antagonist characters (ghosts) are controlled by distinct search algorithms. The objective is to investigate how different algorithms behave under identical conditions and to evaluate their effectiveness based on metrics such as survival time, coins collected, and win rate.

A. Motivation and Objectives

The motivation behind this study is to provide a comparative analysis of various AI pathfinding algorithms in a grid-based game scenario. By analyzing the strengths and limitations of each algorithm, we aim to offer insights into their suitability for real-time applications such as autonomous navigation or game AI systems.

Specifically, our objectives include:

- Implementing multiple AI algorithms (A*, BFS, DFS, Genetic Algorithm, Decision Tree) to control Pac-Man and ghosts.
- Creating a dynamic, visual simulation environment using Pygame.
- Measuring algorithm performance across key criteria including path optimality, time efficiency, and decision quality.
- Visualizing simulation outcomes through graphs and heatmaps.

B. Problem Definition

The central problem addressed in this project is optimal pathfinding in a maze-like environment with dynamic obstacles (moving ghosts). Pac-Man must collect coins distributed across the grid while avoiding collision with ghosts. Each character makes decisions based on a selected algorithm. The challenge lies in ensuring fast, accurate, and adaptive decisions in a constrained environment where both objectives (collecting and avoiding) must be balanced.

C. Contribution

The key contributions of this study are:

- A modular simulation framework that allows easy integration and benchmarking of various AI algorithms.
- Detailed performance analysis of each algorithm under controlled experimental settings.
- Comprehensive visualizations including bar graphs, heatmaps, and path tracking for intuitive interpretation.
- Our experiments show that heuristic-based search (A* and BFS) significantly outperforms others when used against GA-driven ghosts, achieving up to 63% win rates.

D. PEAS Analysis of Agent Types

To formalize our agent design, we conducted a PEAS (Performance, Environment, Actuators, Sensors) analysis for both the Pac-Man and ghost agents:

1) Pac-Man Agent:

- **Performance Measure:** Number of coins collected, survival time (steps), win rate (collecting all coins without being caught), and path efficiency (minimal steps taken).
- **Environment:** Grid-based maze with walls (obstacles), coins (rewards), and ghosts (threats). The environment is partially observable, discrete, sequential, dynamic, and multi-agent.
- **Actuators:** Movement controls in four cardinal directions (up, down, left, right), with one move per time step.
- **Sensors:** Position detection of maze walls, coins, ghosts, and self. Limited visibility range that extends throughout the connected pathways of the maze.

2) Ghost Agent:

- **Performance Measure:** Time to capture Pac-Man (fewer steps is better), prevention of coin collection, and territorial coverage of the maze.
- **Environment:** Same grid-based maze as Pac-Man, but with different objectives. Environment characteristics remain consistent: partially observable, discrete, sequential, dynamic, and multi-agent.
- **Actuators:** Movement controls in four cardinal directions (up, down, left, right), with identical movement capabilities as Pac-Man.
- **Sensors:** Position detection of maze walls, Pac-Man's current location, and self. Depending on the algorithm, some ghosts have perfect information about Pac-Man's position, while others have limited sensing range.

This PEAS analysis guided our implementation of both agent types and helped ensure that each algorithm was evaluated against consistent criteria across all simulations.

II. Related Work

A. Pathfinding Algorithms in Grid-Based Games

Pathfinding in maze-like environments has been a classic benchmark for search algorithms such as BFS, DFS, and A*. In a recent comparative study, Elkari *et al.* examined Depth-First Search (DFS), Breadth-First Search (BFS), and A* in grid mazes to evaluate their efficiency in finding paths [1]. Their results highlight the trade-offs between these methods: DFS uses less memory by exploring one path deeply but does not guarantee an optimal solution, whereas BFS always finds the shortest path at the cost of higher memory usage. A* search, which integrates heuristic guidance, typically expands fewer nodes than BFS for similar optimal paths. Overall, Elkari *et al.* conclude that A* offers the best performance, consistently achieving lower path costs compared to BFS and DFS while keeping runtime reasonable.

Another study by Permana *et al.* compared A*, Dijkstra's algorithm, and BFS within a *Maze Runner* game environment to determine the most suitable pathfinder for an NPC character [2]. They measured each algorithm's path length, search time, and processed nodes. The outcomes similarly favored heuristic search: although BFS and Dijkstra guarantee optimal paths, their broader exploration led to longer computation times in complex levels. These studies corroborate that in grid-based games, BFS can serve as a baseline for shortest paths and DFS as a memory-efficient explorer, but an A*-style algorithm is often recommended for its balance of optimality and efficiency. Our project builds on this insight by implementing all three and comparing their performance in the Pac-Man maze.

B. Evolutionary Algorithms for Pac-Man

Beyond classical search, evolutionary algorithms have been applied to Pac-Man for both pathfinding and strategy optimization. One of the earliest efforts was by Lucas, who evolved a neural-network-based location evaluator for Ms. Pac-Man using a Genetic Algorithm (GA) [3]. The GA optimized the network's weights so that Pac-Man could better decide which direction to move, given the positions of walls, food, and ghosts. This approach demonstrated that an evolutionary method can learn effective strategies without explicit hard-coded rules.

More recently, GAs have also been used on the content side of Pac-Man-like games. Şafak *et al.* designed a GA to automatically generate maze levels for Ms. Pac-Man [4]. Their system encoded maze layouts as genetic chromosomes and defined a fitness function to reward mazes that are challenging yet winnable. The relevance to our project is twofold: First, it shows GA's versatility in game AI, capable of solving both behavior optimization and structural generation. Second, it suggests that if our project uses a GA for pathfinding, we should expect creative but non-deterministic paths that may trade optimality for novelty or adaptability.

C. Tree-Based and Decision Tree Strategies

Researchers have also explored tree-based decision strategies for Pac-Man, which relates to our project's implementation of a decision tree algorithm. Robles and Lucas introduced a simple tree search agent for Ms. Pac-Man that expanded a game tree of possible moves and evaluated the "danger" of each resulting state [5]. Even a shallow forward search significantly boosted Pac-Man's performance by reducing risky choices.

Foderaro *et al.* took this concept further with a model-based decision tree strategy that continuously updates during gameplay using predictions of ghost behavior and maze structure [6]. Their method enabled near-optimal decision-making in real time and demonstrated the potential of decision-tree-based planning. Our project draws from these works by implementing a simpler decision tree that uses key game-state features to make efficient and interpretable decisions under dynamic conditions.

III. Algorithms

This section outlines the artificial intelligence algorithms implemented in the project to control both Pac-Man and ghost agents. Five primary algorithms were utilized: A*, Breadth-First Search (BFS), Depth-First Search (DFS), Genetic Algorithm (GA), and a Decision Tree-based approach. Each algorithm offers a distinct trade-off between path optimality, computational cost, and adaptability in dynamic game environments.

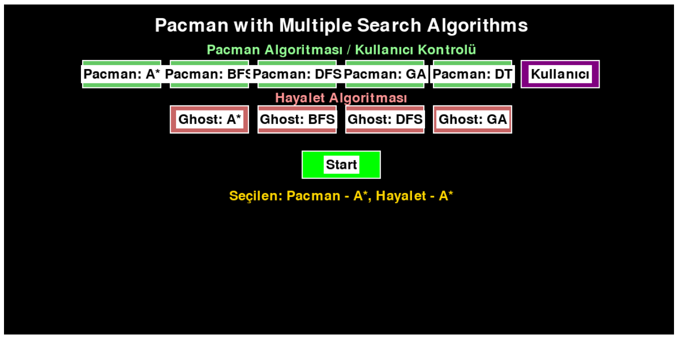

Fig. 1. User Interface of Game.

A. A* Search Algorithm

A* is a widely used informed search algorithm that combines the advantages of uniform-cost search and heuristic guidance. It calculates the cost of reaching a node as $f(n) = g(n) + h(n)$, where $g(n)$ is the path cost from the start node to n, and $h(n)$ is the heuristic estimate from n to the goal.

In this project, A* uses Manhattan distance as the heuristic due to the grid-based maze structure. Both Pac-Man and ghosts can utilize A* to find optimal paths avoiding walls and dynamically reaching targets (coins or Pac-Man, respectively). A* provided the most reliable and shortest paths among all algorithms, making it highly suitable for coin collection.

B. Breadth-First Search (BFS)

BFS is an uninformed search algorithm that explores all nodes at the current depth before moving deeper. It guarantees the shortest path in terms of the number of steps but can be computationally expensive in large search spaces.

In our implementation, BFS is used to guide Pac-Man or ghosts in a layer-wise manner. It performs well in simpler mazes but lacks directionality, making it inefficient in scenarios with multiple coins or moving enemies. However, its simplicity makes it a good benchmark for comparison.

C. Depth-First Search (DFS)

DFS explores as deep as possible along one branch before backtracking. It is memory-efficient but does not guarantee the shortest or even a feasible path in some cases. In the project, a limited-depth variant of DFS was implemented to avoid infinite loops and redundant exploration.

While DFS is fast and low on memory, its performance was inconsistent. In many simulations, it failed to reach all coins or avoided optimal routes. DFS served as a baseline for evaluating uninformed depth-based exploration strategies.

D. Genetic Algorithm (GA)

The Genetic Algorithm is an evolutionary search method inspired by natural selection. It maintains a population of candidate solutions (chromosomes), each representing a path or movement sequence. The fitness of each chromosome is evaluated based on coin collection and survival. Through selection, crossover, and mutation, better solutions evolve over generations.

In this simulation, GA was applied to guide Pac-Man's decisions across multiple steps. Although it did not always produce optimal paths, it showed high adaptability in stochastic settings. Notably, GA allowed Pac-Man to achieve the highest coin collection and win rates when ghosts used non-deterministic strategies. However, its performance was sensitive to parameter tuning and population diversity.

E. Decision Tree Strategy

The Decision Tree approach implements rule-based decision-making where each node evaluates a game-state feature (e.g., distance to ghosts or nearest coin). Based on these features, the agent selects an action from predefined branches.

In our implementation, the decision tree was first trained using state-action pairs generated by the A* algorithm running in controlled scenarios. This supervised learning approach allowed the decision tree to capture effective decision patterns from A*'s optimal pathfinding behavior. Training data was collected from over 200 simulated games where A* successfully navigated the maze to collect coins while avoiding ghosts.

The decision tree model was built using a recursive partitioning algorithm that identified the most discriminative features at each split. Key features included distance to nearest coin, distance to nearest ghost, number of available escape routes, and quadrant information. The resulting tree had a depth of 6 levels with 42 leaf nodes representing different action decisions.

The decision tree's structure was visualized using Graphviz (Figure 2) to provide insights into its decision-making logic. Despite its limitations in performance, the decision tree approach offers valuable transparency and computational efficiency compared to more complex algorithms.

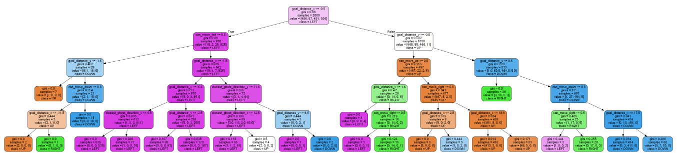

Fig. 2. Visualization of the manually constructed decision tree used for Pac-Man agent behavior. Each node represents a game state rule.

F. Comparative Observations

Each algorithm demonstrated strengths and weaknesses:

- A* achieved consistently optimal paths and high win rates against complex ghost behaviors.
- BFS performed reliably in structured mazes but lacked heuristic efficiency.

- DFS struggled with depth-limited vision and suboptimal route choices.
- GA performed best in adaptive environments, especially against unpredictable ghost algorithms.
- Decision Trees were interpretable but limited in scope without dynamic learning.

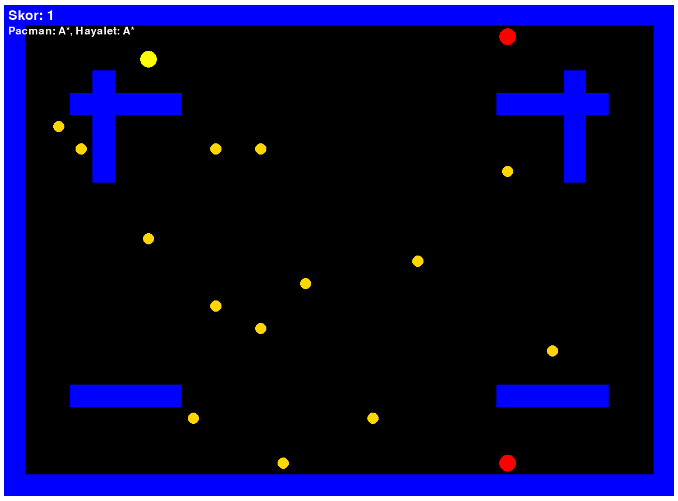

Fig. 3. A screenshot of the Pacman game interface used in the experiment. The environment includes walls (in blue), coins (yellow dots), the Pacman agent (large yellow circle), and ghosts (red circles).

IV. Performance and Results

To evaluate the effectiveness of the implemented AI algorithms, a comprehensive set of simulations was conducted. Each Pac-Man algorithm (A*, BFS, DFS, GA, and Decision Tree) was paired against ghost agents using one of A*, BFS, DFS, or GA. For each combination, 100 trials were run with 50 coins and a maximum of 250 steps per trial.

The performance was assessed based on three primary metrics:

- **Coins Collected:** Average number of coins collected per trial.
- **Survival Steps:** Average number of steps survived before game end.
- **Win Rate (%):** Percentage of trials where Pac-Man successfully collected all coins without being caught.

A. Overall Performance Trends

Among all algorithms, Pac-Man controlled by the A* and BFS algorithms demonstrated the highest performance when facing GA-controlled ghost agents. Specifically:

- A* vs GA achieved a win rate of **63%** and full coin collection.
- BFS vs GA achieved a win rate of **61%** with similar coin collection.

These results suggest that heuristic-based pathfinding algorithms (A* and BFS) outperform others in structured mazes when facing non-deterministic opponents like GA. The Genetic Algorithm, while not optimal in deterministic scenarios, showed adaptability in open and dynamic environments.

B. Detailed Evaluation of Full Results

To obtain a comprehensive understanding of the experimental outcomes, we analyze the complete set of simulation results presented in Appendix V. Each Pac-Man algorithm's performance is evaluated based on its interactions with different ghost strategies, offering insight into both general behavior and specific strengths or weaknesses.

- **A* Algorithm:** A* consistently performed best when paired with GA ghosts, achieving the highest coin collection (50.0), survival time (132.4), and win rate (63%). Against deterministic opponents like A* or BFS, however, its performance dropped significantly. This suggests that A* excels in environments where adversaries are less predictable, leveraging its heuristic efficiency to navigate optimally.

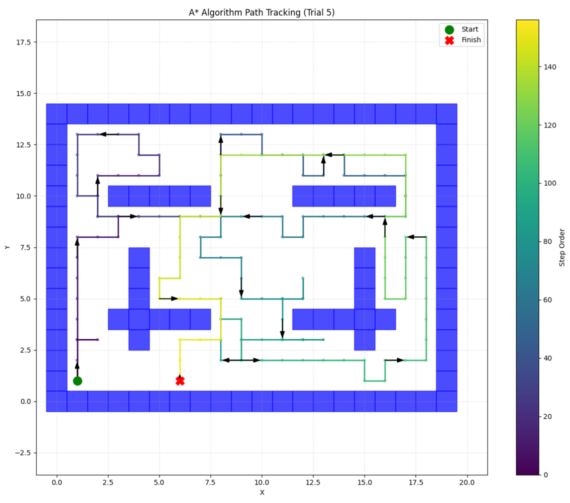

Fig. 4. Path tracking visualization of Pac-Man controlled by A* algorithm. Arrow direction and color indicate step order.

- **BFS Algorithm:** BFS mirrored A*'s performance pattern with GA ghosts, collecting 50 coins and achieving a 61% win rate. Although slightly behind A* in step efficiency, BFS proved to be a competitive alternative in scenarios where shortest path guarantees are beneficial. Its uninformed nature, however, made it vulnerable to more aggressive ghost strategies.
- **DFS Algorithm:** DFS underperformed across all metrics, except when facing GA ghosts, where it showed moderate success (30.6 coins, 114.4 steps, 5% win rate). The algorithm's depth-focused nature led to inefficient exploration and frequent cornering. This reinforces DFS's unsuitability for environments requiring complete state awareness.
- **Genetic Algorithm (GA):** As a Pac-Man controller, GA demonstrated remarkable survivability (up to 155.2 steps) but a very low win rate (1%). This reflects its adaptive but non-deterministic nature—it explores diverse paths and survives longer but struggles to consistently complete

the objective of collecting all coins. It's best suited for exploration rather than optimal navigation.

- **Decision Tree (DT):** DT-based control yielded the weakest results across all ghost pairings, with win rates fixed at 0% and low coin collection (maximum 11.5). This limitation is attributed to its hand-crafted rules and lack of real-time adaptability. While interpretable, DT lacks the flexibility and depth required to react to dynamic threats like moving ghosts.

In summary, heuristic search algorithms such as A* and BFS dominate in goal-oriented tasks with stochastic opponents, while evolutionary or rule-based methods lag due to either exploration bias or rigidity. These trends, clearly reflected in Table V, validate the strength of heuristics in reactive maze navigation.

C. Visualization of Results

To support the analysis, several comparative visualizations were generated:

- **Coin Collection Comparison:** Bar charts show that A* and BFS paired with GA ghosts reach the maximum average of 50 coins.
- **Survival Time:** GA achieved the longest survival in terms of steps, with both Pac-Man and ghost agents adapting gradually.
- **Win Rate:** A* and BFS dominate this metric with over 60% success against GA ghosts.
- **Heatmaps:** Highlight the most frequently visited locations by Pac-Man under different algorithms.
- **Path Tracking Visuals:** Illustrate movement efficiency and exploration behavior.

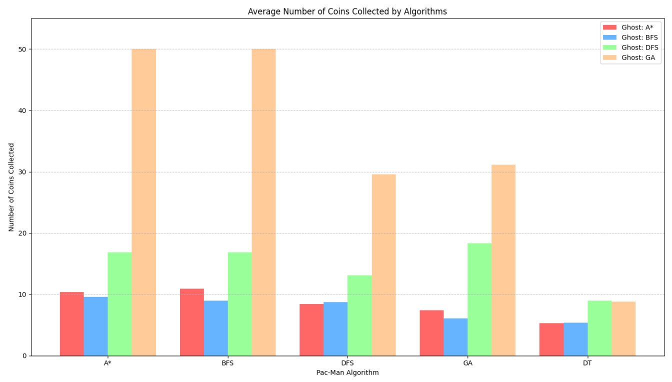

Fig. 5. Average number of coins collected by each Pac-Man algorithm against different ghost strategies.

These visual aids confirm the performance patterns and support the conclusion that heuristic-based algorithms (A* and BFS) offer superior performance in structured, adversarial game environments.

D. Tabular Analysis of Results

To further support the analysis, we include summarized tables presenting algorithm performance under selected conditions.

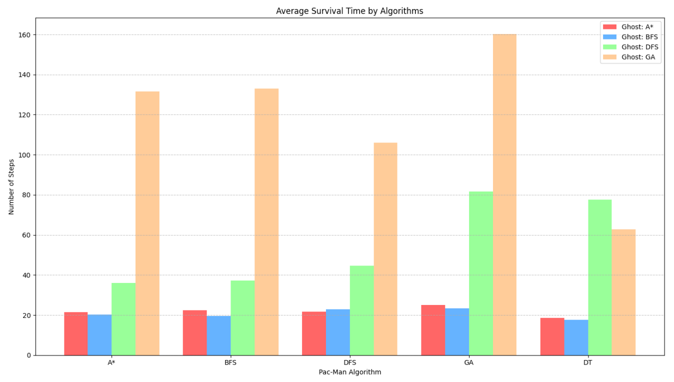

Fig. 6. Average survival time (number of steps) across different Pac-Man and ghost algorithm pairings.

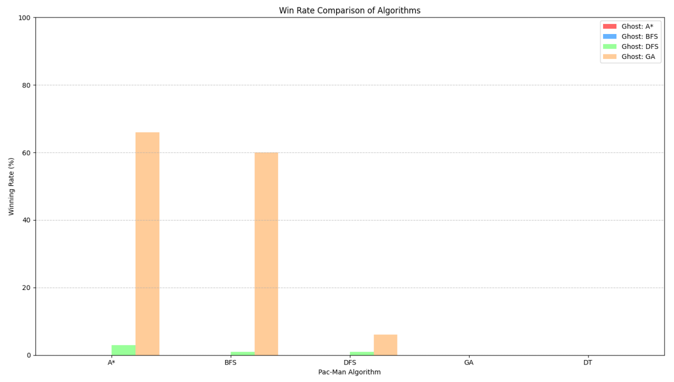

Fig. 7. Win rate comparison among algorithm combinations. A* and BFS exhibit the highest success rates against GA ghosts.

Analysis: Table I clearly highlights that heuristic-based algorithms (A*, BFS) significantly outperform others against GA ghosts. Although GA-controlled Pac-Man achieved the longest survival time, it did not translate into higher win rates due to its probabilistic and non-deterministic path planning. DFS performed poorly in both efficiency and success rate, while the Decision Tree failed to adapt dynamically to ghost movement.

Interpretation: Table II provides a comparative view of the best possible outcome achieved by each Pac-Man algorithm. Notably, all optimal cases occurred when ghosts were controlled by GA, suggesting that its probabilistic behavior, while unpredictable, may be exploitable by well-planned search algorithms. A* and BFS both achieved perfect coin collection and high win rates, demonstrating their robustness and goal-directed efficiency. On the other hand, Decision Tree strategies were ineffective in all scenarios, confirming their limitations in static rule-based design.

V. Conclusion

A. Summary

In this project, we developed a simulation-based framework to analyze the performance of five AI algorithms—A*, BFS, DFS, Genetic Algorithm (GA), and Decision Tree—in the

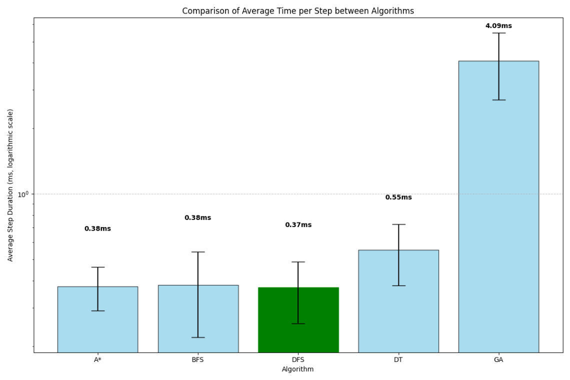

Fig. 8. Comparison of average time per step for each algorithm (log scale). Genetic Algorithm is the slowest due to iterative population updates.

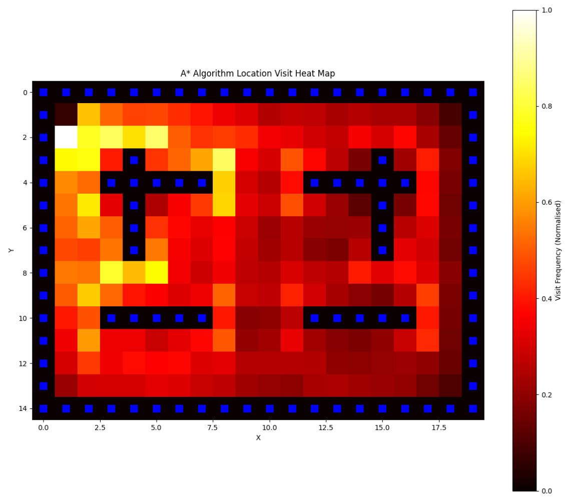

Fig. 9. Heatmap showing frequency of visited locations by A*-controlled Pac-Man. Bright areas indicate frequent exploration.

context of a Pac-Man-like game. The system enables agents to autonomously navigate a maze, either to collect coins (Pac-Man) or to pursue the player (ghosts), using search or decision-making techniques. Each algorithm was tested under identical game conditions, and their behavior was evaluated in terms of coin collection, survival time, and win rate.

B. Findings

The experimental results indicate that heuristic-based search algorithms such as A* and BFS outperform others in structured and adversarial environments. Notably, the A* and BFS agents achieved win rates of up to 63% and 61% respectively when paired against GA-controlled ghosts. The GA-based Pac-Man agent demonstrated long survival times but inconsistent win rates, highlighting its adaptability but lack of guaranteed optimality. In contrast, DFS showed weak performance due to its depth-biased exploration, while Decision Trees, though interpretable, lacked flexibility and failed to adapt to dynamic threats.

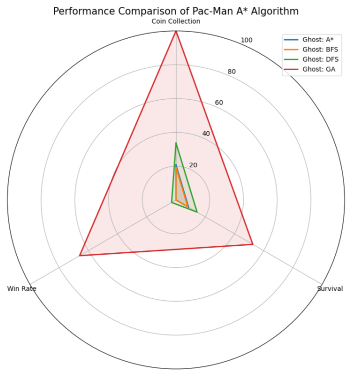

Fig. 10. Radar chart showing coin collection, survival, and win rate for A* algorithm against different ghost strategies.

TABLE I
PERFORMANCE OF PAC-MAN ALGORITHMS VS GA-CONTROLLED GHOSTS

Pac-Man Algorithm	Avg. Coins	Avg. Steps	Win Rate (%)
A*	50.0	132.4	63.0
BFS	50.0	131.4	61.0
DFS	30.6	114.4	5.0
GA	30.6	155.2	1.0
Decision Tree	11.5	64.5	0.0

TABLE II
BEST OBSERVED RESULTS FOR EACH PAC-MAN ALGORITHM (VS ANY GHOST)

Algorithm	Best Coins	Best Steps	Best Win %	Ghost Opponent
A*	50.0	132.4	63.0	GA
BFS	50.0	131.4	61.0	GA
DFS	30.6	114.4	5.0	GA
GA	30.6	155.2	1.0	GA
DT	11.5	83.1	0.0	DFS

C. Limitations

Despite its robustness, the system has several limitations:

- The Decision Tree logic was trained on A* data but could benefit from more diverse training examples across various game scenarios.
- Genetic Algorithm configurations (e.g., population size, mutation rate) were selected heuristically and may not be optimal.
- The game environment, though functional, is relatively simple and grid-based, which may not fully reflect the complexity of real-world navigation tasks.
- The simulations do not currently support learning over time or across episodes.

This project establishes a solid foundation for future research in game AI benchmarking, providing a versatile platform for both classical and learning-based algorithm comparisons.

REFERENCES

[1] B. Elkari, M. Zerrouki, and A. Elbaz, "Exploring Maze Navigation: A Comparative Study of DFS, BFS, and A* Search Algorithms," *Statistics, Optimization & Information Computing*, vol. 12, pp. 761–781, May 2024.

[2] S. D. H. Permana, K. B. Y. Bintoro, B. Arifitama, and A. Syahputra, “Comparative Analysis of Pathfinding Algorithms A*, Dijkstra, and BFS on Maze Runner Game,” *International Journal of Information System and Technology*, vol. 1, no. 2, pp. 1–8, 2018.

[3] S. M. Lucas, “Evolving a Neural Network Location Evaluator to Play Ms. Pac-Man,” in *Proc. IEEE Symposium on Computational Intelligence and Games (CIG)*, 2005, pp. 278–285.

[4] A. B. Şafak, E. Bostancı, and A. E. Soyluçiçek, “Automated Maze Generation for Ms. Pac-Man Using Genetic Algorithms,” *International Journal of Machine Learning and Computing*, vol. 6, no. 4, pp. 226–230, 2016.

[5] D. Robles and S. M. Lucas, “A Simple Tree Search Method for Playing Ms. Pac-Man,” in *Proc. IEEE Symposium on Computational Intelligence and Games (CIG)*, 2009, pp. 249–255.

[6] G. Foderaro, A. Swingler, and S. Ferrari, “A Model-Based Approach to Optimizing Ms. Pac-Man Game Strategies in Real Time,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, no. 2, pp. 153–166, 2017.

[7] E. Demirtaş, “Project codes, results and game.,” GitHub repository Available: https://github.com/agining/yap-441

Appendix A
Algorithm Parameters

Table III shows the parameter settings used in the Genetic Algorithm. These values were chosen heuristically to balance performance and computation time.

TABLE III
GENETIC ALGORITHM CONFIGURATION

Parameter	**Value**
Population Size	50
Chromosome Length	20
Mutation Rate	0.1
Elite Size	5
Number of Generations	5
Fitness Function	Coin collected + Survival time

Appendix B
Maze Layout Matrix

Table IV shows a simplified representation of the maze used in simulations. Walls are indicated with 1s and open paths with 0s.

TABLE IV
SIMPLIFIED MAZE MATRIX REPRESENTATION

1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1
1	0	1	1	1	1	0	1
1	0	1	0	0	1	0	1
1	0	1	0	0	1	0	1
1	0	1	1	1	1	0	1
1	0	0	0	0	0	0	1
1	1	1	1	1	1	1	1

Appendix C
Full Simulation Results Summary

Table V summarizes the complete simulation results, showing average coin collection, survival time (steps), and win rate for each combination of Pac-Man and ghost algorithms.

TABLE V
COMPLETE PERFORMANCE METRICS ACROSS ALL ALGORITHM COMBINATIONS

Pac-Man Algorithm	**Ghost Algorithm**	**Coins Avg**	**Steps Avg**	**Win Rate (%)**
A*	A*	12.1	25.3	0.0
A*	BFS	8.9	19.3	0.0
A*	DFS	17.4	38.0	0.0
A*	GA	**50.0**	**132.4**	**63.0**
BFS	A*	10.8	22.8	0.0
BFS	BFS	9.5	20.3	0.0
BFS	DFS	17.1	36.8	2.0
BFS	GA	**50.0**	**131.4**	**61.0**
DFS	A*	9.3	24.1	0.0
DFS	BFS	7.8	21.2	0.0
DFS	DFS	13.4	47.8	0.0
DFS	GA	30.6	114.4	5.0
GA	A*	7.2	25.5	0.0
GA	BFS	6.1	22.1	0.0
GA	DFS	15.8	71.4	0.0
GA	GA	30.6	**155.2**	1.0
Decision Tree	A*	5.8	18.8	0.0
Decision Tree	BFS	5.5	19.1	0.0
Decision Tree	DFS	10.1	**83.1**	0.0
Decision Tree	GA	11.5	64.5	0.0