Azrieli
College of Engineering
Jerusalem

עזריאלי
מכללה אקדמית להנדסה
ירושלים

המחלקה להנדסת תוכנה

פרויקט גמר – ה'תשפ"ה

רשתות נוירונים ניתנות לפירוש עבור בעיית מסילה המילטונית

Interpretable Neural Networks for Tackling Hamiltonian Path Problems

מאת

אנדרי טמס

ת.ז. 322341173

טוני בגאלי

ת.ז. 322342205

מנחה אקדמי: ד"ר לוריא צור.

04/07/2025

חתימת מנחה:

מערכות ניהול הפרויקט:

#	מערכת	מיקום
1	מאגר קוד	Github link
2	יומן	Issues link
3	קישור לסרטון **דוח אלפא**	Alpha Video link
4	קישור לסרטון **דוח הסופי**	Final Video link

מידע נוסף:

סוג הפרויקט	מחקרי ממרצה במכללה
אם זה פרויקט ממשיך?	זה פרויקט חדש

הצהרה:

העבודה נעשתה בהנחיית **ד"ר לוריא צור**, עזריאלי המכללה האקדמית להנדסה ירושלים - המחלקה להנדסת תוכנה.

החיבור מציג את עבודתנו האישית ומהווה חלק מהדרישות לקבל תואר ראשון בהנדסה.

תקציר בעברית

פרויקט זה עוסק בהתמודדות עם בעיות מסילה המילטוניות (Hamiltonian Path Problems) בעיות קומבינטוריות קשות מסוג (NP-complete) תוך דגש על **פרשנות, שקיפות והבנה של תהליך קבלת ההחלטות**. הפרויקט מתמקד בבעיית סיבוב הפרש (Knight's Tour), שבה על הפרש לעבור על כל לוח השחמט מבלי לחזור על משבצת.

במהלך הפרויקט פיתחנו מערכת מקיפה הכוללת מספר מצבי פעולה ואלגוריתמים מתקדמים:

אלגוריתם וורנזדורף (Warnsdorff) – אלגוריתם היסטורי הבוחר את המהלך עם מספר האפשרויות הקטן ביותר להמשך.
חיפוש עצים מונטה קרלו (MCTS) – אלגוריתם הסתברותי המעריך מסלולים אפשריים לבחירת מהלך מיטבי, ומאפשר ניתוח של אופן קבלת ההחלטות לאורך ההרצה.
PUCT בסגנון AlphaZero – שילוב רשת נוירונים עם חיפוש עץ הסתברותי ליצירת AI חזק במיוחד ובעל יכולת למידה עצמית, תוך **חקר דפוסי ההתנהגות והאסטרטגיות** שנוצרות בתהליך.
שחקן עץ החלטה (Decision Tree) – סוכן סמלי ניתנת לפרשנות, המאמן עץ החלטה על פי תוצאות PUCT, ומאפשר **שקיפות מלאה בהבנת הכללים וההחלטות** שנוצרו בתהליך הלמידה.
מצבי Elo – השוואת ביצועי האלגוריתמים בתחרויות הוגנות להסקת חוזק יחסי והבנת יתרונותיהם.

המטרה המרכזית הייתה לגלות **היוריסטיקות חבויות חדשות שה-AI עשוי לפתח במהלך ההתמודדות עם בעיות אלו**, ולבחון האם ניתן להמיר את הגישות מבוססות ה-NN וה-AI להיוריסטיקות סמליות פשוטות וברורות, אשר יאפשרו פתרונות ניתנים להבנה מבלי להסתמך על רשתות נוירונים או בינה מלאכותית כשחורים לא ניתנים לפירוש.

Abstract

The **Knight's Tour problem**, a special case of the Hamiltonian Path Problem (HPP), involves finding a path for a knight to visit every square on an n-by-n chessboard exactly once using valid L-shaped moves. Due to its **NP-complete nature**, efficient solutions are unattainable for all general cases, especially with the addition of obstacles that mimic real-world constraints. This project tackles the Knight's Tour by exploring multiple approaches, progressively increasing both **performance and interpretability**.

We first implemented **Warnsdorff's heuristic** as a classical baseline, followed by **Monte Carlo Tree Search (MCTS)** to incorporate probabilistic exploration.
To further improve performance, we integrated a **PUCT (Predictor + UCT) neural network-guided search**, inspired by AlphaZero, which combined policy priors and value predictions to guide tree search more effectively.
However, recognizing the opacity of neural-based methods, we developed a **Rule-Based Logic-Only Player** that approximates neural policies using **handcrafted symbolic formulas** based on move degrees and board accessibility.
Finally, a **Decision Tree model** was trained on data collected from neural PUCT runs to extract **explicit, human-readable if-else rules**, enabling symbolic decision-making without neural inference.

Throughout this progression, we leveraged transparency tools such as **heatmaps, Q-value visualizations, and feature correlation analyses** to uncover hidden heuristics within AI decisions. Our findings demonstrate that neural-guided approaches often build upon **Warnsdorff-like degree heuristics enhanced by strategic long-term planning**, which can be approximated symbolically for practical and interpretable solutions.

This project showcases how **AI decision-making processes can be reverse-engineered** into transparent heuristics, bridging the gap between powerful black-box models and human-understandable logic, with implications for combinatorial optimization, robotics, and explainable AI research.

Table of Contents

Key Terms

1. **Knight's Tour**
   The classical puzzle of moving a knight across a chessboard so that it visits every square exactly once. This can be viewed as a special case of the Hamiltonian Path Problem in graph theory.

2. **Hamiltonian Path Problem (HPP)**
   A fundamental graph problem of finding a path that visits every vertex exactly once. The Knight's Tour can be modeled as finding a Hamiltonian path (or cycle) on the board's graph.

3. **Monte Carlo Tree Search (MCTS)**
   A search algorithm that builds a game tree incrementally through random simulations ("rollouts") and uses statistical outcomes to determine the most promising moves.

4. **Warnsdorff's Algorithm**
   A heuristic for the Knight's Tour that always moves to the square with the fewest onward moves (lowest degree), aiming to minimize early dead ends.

5. **Brute Force / Backtracking**
   A systematic search approach that tries every possible path. If a path leads to a dead end, the algorithm backtracks to explore other branches.

6. **Reinforcement Learning (RL)**
   A machine-learning paradigm where an agent learns optimal actions through rewards and penalties while interacting with an environment.

7. **Q-Learning**
   A popular RL method that maintains a Q-table (state-action values). The agent updates these values based on rewards received, converging toward an optimal policy over time.

8. **PUCT (Predictor + UCT)**
   An algorithm combining neural network policy predictions with Monte Carlo Tree Search (MCTS) to guide move selection efficiently. It balances exploration and exploitation using prior probabilities from the neural network, as used in AlphaZero.

Elevator speech

In this project, we aim to develop **interpretable neural networks** that tackle **Hamiltonian path problems:** A class of complex combinatorial puzzles that include the Knight's Tour and the Traveling Salesman Problem (TSP). These problems are **NP-complete** and grow exponentially in complexity.

Our goal in this project is to determine whether general-purpose reinforcement learning algorithms can discover new heuristics and algorithms that don't involve neural networks at all. Focusing on interpretability means we want to shine a light on the network's decision-making process, extracting insights into the "why, and how" behind each move rather than just offering unexplained solutions. Our motivating example is the discovery of the min-conflicts heuristic, which is a powerful and widely used heuristic to solve constraint satisfaction problems that was discovered by a neural network algorithm.

Introduction

Combinatorial optimization problems play a pivotal role in numerous real-world domains, from routing in logistics to path planning in robotics. One particularly fascinating class of these problems revolves around **Hamiltonian paths** routes that must visit every node (or square) exactly once. Among the most iconic examples is the **Knight's Tour**, where a knight in chess traverses an n-by-n board, trying to visit each square a single time. Another well-known instance is the **Traveling Salesman Problem (TSP)**, in which a traveller must visit each city once and minimize the total travel distance.

Over the years, many methods have emerged to tackle Hamiltonian path problems: classical heuristics, brute-force backtracking, and various AI-driven algorithms. Yet, as problem sizes grow, these approaches often become computationally expensive. Furthermore, sophisticated algorithms particularly machine learning methods tend to function as **"black boxes,"** offering limited insight into why a given move or route is chosen.

This project addresses the gap between **efficient pathfinding and interpretability**. Existing solutions for problems like the Knight's Tour either rely on well-known heuristics **(e.g., Warnsdorff's rule)** or exploit more advanced AI techniques **(e.g., Monte Carlo Tree Search, PUCT with neural networks)**, but rarely provide a clear explanation of their reasoning. Such transparency is crucial in high-stakes environments where human operators need to trust and verify the solutions, solutions where it might be in autonomous vehicle navigation, warehouse robotics, or complex logistics planning.

During this project, we developed a comprehensive system with multiple approaches to tackle the Knight's Tour problem, including:

- **Warnsdorff's Algorithm** for quick heuristic-based tours.
- **Monte Carlo Tree Search (MCTS)** to explore move possibilities probabilistically.
- **PUCT with Neural Networks (AlphaZero-style)**, combining deep learning with tree search to develop powerful policies, while providing **full transparency into decisions via Q-values, priors, visit counts, and move degrees**. This approach also included the use of **heatmaps**, visually showing how the AI "views" the board at each step, allowing humans to intuitively understand its focus and decision rationale.
- **Rule-Based PUCT Logic-Only Player**, designed to function **entirely without neural networks**. Instead of relying on learned priors or value functions from a NN, this player calculates move probabilities using **symbolic rules based on Warnsdorff-like degree heuristics and inverse move degrees**, combined with counts of future legal moves. It effectively mimics the behavioural patterns seen in neural-guided PUCT by **calculating what the NN would have prioritized using purely handcrafted mathematical formulas**, offering fully interpretable decisions and replicating strong performance without any AI training or model inference.
- **Decision Tree-based AI**, which takes transparency a step further by training a symbolic decision tree on logs collected from the neural PUCT agent. The decision tree uses input features such as **Q-values, visit counts, priors, and Warnsdorff degrees** extracted from previous AI runs to **learn classification rules** that decide which move is best in each scenario. After training, it functions **entirely without neural networks or simulations**, relying only on its learned if-else rules to calculate the optimal path decisions. This enables **transparent, human-readable explanations** of each move while reproducing high-quality AI-like performance without the computational demands or opacity of neural inference.

Additionally, we implemented **Elo match evaluations** to compare the performance of these approaches under identical conditions and determine their relative strengths.

This progression **from neural-network-guided PUCT, to a purely logic-based symbolic mimic, and finally to a decision tree explanation** demonstrates how insights from powerful AI models can be distilled into **simple symbolic heuristics**. The core goal was not to only tackle NP problems outright, but rather to **discover hidden heuristics that AI models might have found while tackling these problems**, and **replace opaque neural network decisions with transparent, interpretable strategies** understandable by humans and usable without reliance on AI black boxes.

Problem description

The **Hamiltonian Path Problem (HPP)** involves finding a path that visits each vertex (or cell) exactly once, making it especially challenging in large or obstacle-laden environments. In this project, we focus on the **Knight's Tour**, a specific instance of HPP where a knight must visit every square on a chessboard exactly once using valid L-shaped moves. However, the Knight's Tour extends beyond its classical puzzle framing in several key ways:

1. NP-Completeness and Insolvability

The Knight's Tour belongs to the broader class of **NP-complete problems**, meaning there is no known efficient algorithm to guarantee a solution in polynomial time for all cases. This makes the problem **effectively unsolvable** for arbitrary board sizes and obstacle configurations. Therefore, the challenge lies in **finding the most optimal and extreme paths possible for each specific board configuration**, striving for maximal coverage even when a full tour is unattainable.

2. Obstacle Handling

Adding blocked squares to the board simulates real-world constraints where certain nodes or paths are unavailable. This significantly increases problem complexity, creating scenarios where dead ends occur earlier and requiring robust decision-making to maximize path length before failure.

3. Transparency and Interpretability Challenges

Traditional algorithms and AI approaches often provide solutions without explaining **why each move was selected**. This lack of transparency limits human understanding and prevents the extraction of reusable heuristics or general strategies that could be applied to other combinatorial problems.

4. General Hamiltonian Path Challenges

Due to its NP-completeness, the Knight's Tour encapsulates the broader challenge of developing algorithms capable of **tackling extremely difficult problems effectively**, under varying constraints and conditions, while ensuring that decisions are rational and explainable.

Problem Requirements and Characterization

1. Flexible Board Configurations

The system must support **various board sizes and customizable obstacle placements**, enabling diverse and realistic test scenarios for algorithm evaluation and benchmarking.

2. Decision Explanation

Each move should be accompanied by a **clear and interpretable rationale**, indicating how it contributes towards reaching a complete tour or avoiding premature dead ends.

3. Performance vs. Interpretability

While computational efficiency is important for exploring larger boards, **interpretability remains central**, ensuring that the logic behind decisions is understandable and traceable.

4. Training Data and Tracking Decisions

To extract heuristics and develop interpretable models, the system must be capable of **tracking AI decisions, logging features, and generating training data** that accurately represents how algorithms approach the problem.

The Problem from a Software Engineering Perspective

Implementing an interpretable Hamiltonian path tackler for the Knight's Tour involves several software engineering challenges:

1. Modular Architecture

Designing a clean codebase where different algorithms and heuristics can be implemented in **separate, reusable modules**, enabling testing, evaluation, and extension without unintended side effects.

2. Extracting Hidden Heuristics from Neural Networks

Neural networks can learn powerful policies, but their internal reasoning remains opaque. A significant challenge is to **figure out the hidden heuristics embedded within neural network decisions**, and convert them into transparent, symbolic rules or simpler models that can replicate their performance without relying on AI black boxes.

3. Finding a Way to Achieve Transparency and Tracking

Developing tools and methodologies to **track, log, and visualize** AI decision-making processes, including features for importance of analyses, to provide human-understandable insights into each move.

Similar Works & Comparison

Several research efforts have focused on integrating neural networks, heuristic algorithms, or both to tackle combinatorial optimization problems such as the **TSP** and the **Knight's Tour**. The table below highlights a few key publications, summarizing each work and its relevance to our project's emphasis on interpretability, AI-driven heuristics, and efficiency.

Article	Summary	Relevance to the Project
Learning Improvement Heuristics for Solving Routing Problems	This paper explores how neural networks can be used to solve the Traveling Salesman Problem (TSP), including methods for training and optimizing NN models for combinatorial optimization tasks.	The paper provides insights into using neural networks for combinatorial problems like TSP, which is directly related to our goal of solving Hamiltonian path-type problems using neural networks.
NASA Technical Paper on Min-Conflicts Algorithm	This document discusses the min-conflicts algorithm for constraint satisfaction problems, highlighting heuristic-based approaches to problem-solving.	The min-conflicts algorithm's heuristic nature is relevant for our project's focus on discovering and extracting efficient heuristics for problems like Knight's Tour and TSP.
A Survey on Neural Network Interpretability	This paper reviews deep learning approaches applied to NP-hard problems, offering techniques for improving the efficiency of solving such problems.	The methods explored in this paper could guide the neural network architecture and training strategies to efficiently solve Hamiltonian path problems, aligning with our goal to optimize and interpret neural network solutions.
Acquisition of Chess Knowledge in AlphaZero	The paper focuses on making AI models, especially deep learning, interpretable in the context of optimization, explaining the decision-making process.	This paper aligns with our project's emphasis on interpretability, offering methods and frameworks to explain neural network decisions in solving complex optimization problems.

Solution Description

Our solution to the Knight's Tour problem was developed through a systematic pipeline, where **each approach built upon insights from the previous stage**, gradually increasing both performance and interpretability.

1. **Warnsdorff's Algorithm – Classical Heuristic Baseline**

   We began with **Warnsdorff's algorithm**, a traditional heuristic that selects the next move based on the **fewest onward options** (degree heuristic).

$$score(move) = degree(move)$$

   where **degree(move)** is the number of legal moves available from the target square.

   The algorithm iteratively chooses the move with the **lowest degree**, prioritizing routes that reduce the chance of early dead ends. While this approach is fast and effective on open boards, it fails when obstacles disrupt move availability, motivating the need for adaptive algorithms.

2. **Monte Carlo Tree Search (MCTS) – Probabilistic Exploration**

   To tackle limitations of fixed heuristics, we implemented **Monte Carlo Tree Search (MCTS)**. MCTS builds a search tree by performing many simulations, each consisting of:

   - **Selection**: Traversing nodes based on UCT (Upper Confidence Bound applied to Trees):

$$UCT = Q + c \cdot \sqrt{\frac{\ln N_{parent}}{N_{child}}}$$

     where:
     - **Q** is the average reward of the child node
     - **N** are visit counts
     - **c** is an exploration parameter balancing exploitation and exploration
   - **Expansion**: Adding new nodes to the tree when unexplored moves are found
   - **Simulation**: Running a random rollout from the new node to estimate its value
   - **Backpropagation**: Updating Q-values up the tree based on simulation results

MCTS enabled us to **explore move sequences probabilistically**, offering better adaptability compared to deterministic heuristics.

3. **PUCT with Neural Networks – AlphaZero-Style Guided Search**

To further improve performance, we integrated a **PUCT (Predictor + UCT) approach using neural networks**, inspired by AlphaZero. Here, a **policy and value neural network** predicts:

- **Policy (P)**: The probability distribution over possible moves
- **Value (V)**: The estimated outcome (e.g., coverage percentage)

The **PUCT formula** combines these predictions with tree search statistics:

$$PUCT = Q + c_{puct} \cdot P \cdot \frac{\sqrt{N_{parent}}}{1 + N_{child}}$$

- **Q** is the average value of the child node
- **P** is the prior probability from the neural network
- **N** are visit counts
- **c_{puct}** controls exploration strength

This approach enabled powerful decision-making, as the neural network guides the tree search towards promising moves, while maintaining **transparency** via:

- **Q-values, priors, visit counts, and move degrees** logged for each move
- **Heatmaps** visualizing the board from the AI's perspective, enhancing human interpretability

4. **Rule-Based PUCT Logic-Only Player – Symbolic Mimic of Neural Policies**

From the transparency of the neural-guided PUCT runs, we observed consistent heuristic patterns. We then designed a **Rule-Based PUCT Logic-Only Player** to replicate these patterns **without any neural network inference**. This player calculates move scores using:

$$score(move) = \frac{1}{degree(move) + 1e-5}$$

where **degree(move)** is the number of onward moves, and **1e-5** prevents division by zero. The scores are normalized into a probability distribution:

$$P(move) = \frac{score(move)}{\sum score(all\_moves)}$$

These handcrafted formulas mimic the behavioural tendencies learned by the neural PUCT agent, producing high-quality decisions purely from **symbolic rules**, thus achieving interpretability and eliminating reliance on opaque neural inference.

5. **Decision Tree-Based AI – Extracting Symbolic Heuristics**

To further enhance transparency, we trained a **Decision Tree model** on data logged from the neural PUCT runs. Input features included:

- **Q-values**
- **Visit counts**
- **Prior probabilities**
- **Warnsdorff degrees**

The decision tree learned **if-else classification rules** to select optimal moves in various board states. After training, it operates **entirely without neural networks or simulations**, relying purely on these symbolic rules. This approach combines:

- **High performance**, as it mimics neural-guided decisions
- **Full interpretability**, as humans can read and understand each decision rule

6. **Elo Match Evaluations – Comparative Performance Testing**

Finally, we implemented **Elo match frameworks** to evaluate the relative strengths of:

- **Warnsdorff's algorithm**
- **MCTS baseline**
- **PUCT with neural network**
- **Rule-based logic-only player**
- **Decision tree player**

This allowed us to quantify the **effectiveness and trade-offs between performance and interpretability** across all developed methods.

Solution Summary

Our solution pipeline progressed systematically:

1. Started with **classical heuristics (Warnsdorff)**
2. Advanced to **probabilistic exploration (MCTS)**
3. Integrated **neural-guided PUCT with full transparency**
4. Designed a **logic-only symbolic mimic** based on neural insights
5. Extracted **decision tree heuristics** for ultimate interpretability
6. **Benchmarked all approaches** to understand their strengths and weaknesses

Conclusions

Throughout this project, we explored multiple approaches to tackle the Knight's Tour problem, progressing from classical heuristics to advanced AI-driven methods and finally to interpretable symbolic models. This journey yielded several important conclusions and lessons:

1. **AI-Guided Search is Powerful but Opaque**
   Our AlphaZero-style PUCT with neural networks demonstrated high performance and adaptability across diverse board configurations. However, its **black-box nature** limits human understanding of the underlying decision-making process, highlighting the interpretability gap in modern AI.

2. **Transparency Enables Heuristic Discovery**
   By logging detailed AI decisions – including Q-values, priors, visit counts, and degrees – and visualizing them via heatmaps and correlation matrices, we gained insights into **how the AI prioritizes moves**. For example, we observed that the neural-guided PUCT tends to:
   - Prefer moves with **lower onward degree**, similar to Warnsdorff's heuristic
   - Occasionally choose higher-degree moves when it predicts **long-term coverage improvement**, showing strategic sacrifice
   - Avoid early edges or corners unless necessary, indicating spatial prioritization patterns

3. **Logic-Only Heuristics Can Mimic AI Behaviour**
   The handcrafted rule-based player, which calculated inverse degree scores to approximate neural policies, achieved strong performance without relying on trained models. This demonstrates that **interpretable symbolic heuristics can replicate AI-guided strategies**, suggesting that much of the neural network's policy can be approximated by:
   - **Inverse degree heuristics weighted by future move counts**
   - Prioritization of moves maintaining maximal board accessibility

4. **Decision Trees Provide Human-Readable Strategies**
   Training a decision tree on neural-guided decisions allowed us to extract explicit, human-readable rules for move selection. The decision tree revealed:
   - **Onward degree, visit counts (N), and neural prior (P)** were top features influencing decisions
   - Certain board configurations consistently resulted in the same move choice, indicating **underlying heuristics such as edge avoidance or centre prioritization**

5. Hidden Heuristics and Behavioural Insights

During our project, several hidden heuristics and behavioural patterns were uncovered through tracking and analysing the AI models decisions:

1. **Beyond Warnsdorff's Minimum Degree Selection**
    - **Warnsdorff's heuristic** always chooses the move with the **fewest onward options (lowest degree)** to avoid early dead ends.
    - **PUCT-guided AI**, however, showed more nuanced behaviour:
        - Sometimes it **selects higher-degree moves** if the neural value predicts better long-term coverage, prioritizing future board openness over immediate minimal degree.
        - It balances **degree-based safety** with **strategic exploration**, unlike Warnsdorff's purely greedy approach.

2. **Feature Importance and Prioritization**
    - From the **correlation matrix and decision tree** analysis:
        - **Visit counts (N)** were a top determinant of move selection, implying that moves frequently explored during tree search gained higher confidence.
        - **Prior probabilities (P)** from the neural policy significantly influenced decisions, even when degree was not minimal.
        - **Q-values** played a secondary but impactful role in tie-breaks between moves with similar degrees or visit counts.

3. **Spatial Heuristics and Board Awareness**
    - Heatmaps revealed AI often **avoids corners or edges early** unless strategically forced, preferring moves that maintain central board accessibility for as long as possible.
    - In contrast, **Warnsdorff's heuristic is unaware of spatial layout patterns**, relying solely on onward move counts without considering future reachability or board openness.

4. **Emergent Long-Term Planning**
    - The AI's decision patterns indicated **implicit long-term planning**:
        - Choosing moves that preserve **board connectivity**, avoiding isolated regions that cannot be revisited.
        - Sacrificing short-term degree optimality to ensure continued mobility in later stages.

Summary of Differences from Warnsdorff

Aspect	Warnsdorff	PUCT-based AI / Logic / Tree
Selection Criterion	Lowest degree	Combination of **degree**, **prior**, **visit count**, and **Q-value**
Planning Horizon	Myopic (greedy)	Strategic lookahead via **tree search** and **NN** value estimates
Spatial Awareness	None (degree only)	Avoids corners early, prioritizes central mobility
Flexibility	Fixed heuristic	Adaptable based on learned **policies** and **board state**
Interpretability	Fully interpretable	AI opaque; logic-only and tree models provide interpretability

Overall Behavioral Insight

The AI-guided algorithms demonstrated that **optimal Knight's Tour paths are not purely minimal-degree sequences**, but rather involve a **balance between degree heuristics, board connectivity, and strategic foresight**. Our extraction of decision trees and symbolic rules captured parts of this complex policy, bridging performance with transparency.

Note: Examples of the correlation matrix, transparency heatmaps, decision tree visualizations, and performance estimates of each approach are provided in the appendices for detailed reference.

Helpful Generative AI Tools

ChatGPT

- **Idea Generation** & **Research:** ChatGPT was invaluable for brainstorming new approaches, clarifying complex algorithmic concepts such as **MCTS**, **PUCT**, and **decision tree heuristics**, and refining our understanding of AI interpretability frameworks. It felt like having a real-time assistant, continuously validating our ideas and providing alternative perspectives during both implementation and report writing phases.

- **Explanations** & **Documentation:** ChatGPT provided clear, concise explanations of advanced AI methods, summarized theoretical concepts, and assisted in structuring our documentation effectively. Its ability to contextualize discussions within **Hamiltonian path** problems and **reinforcement learning** applications allowed us to produce thorough, well-structured, and technically sound documentation, saving considerable time in final report preparation.

GitHub Copilot

- **Collaboration** & **Team Efficiency:** During collaborative development in our shared repository, GitHub Copilot maintained a consistent coding style, reduced repetitive boilerplate, and allowed us to focus review discussions on algorithmic logic rather than syntax. This improved the quality and speed of our code reviews and pull requests.
- **Inline Code Suggestions:** Copilot provided real-time code snippets for routine tasks, such as initializing data structures, implementing standard loops, and setting up class templates. This functionality helped maintain coding flow and productivity, reducing interruptions to look up routine syntax.

Cody AI (in Visual Studio)

- **Code Assistance:** Within Visual Studio, Cody AI effectively generated boilerplate code, class outlines, and method stubs, allowing us to allocate more time to implementing complex algorithms and refining project logic rather than repetitive code writing.
- **Contextual Recommendations:** Cody AI provided context-aware suggestions and refactor recommendations, streamlining implementation and enhancing coding efficiency by anticipating next logical steps based on existing code structures.

Appendices

Risk Table

Risk ID	Risk Description	Likelihood (L/M/H)	Impact (L/M/H)	Mitigation Strategy	Contingency/Response Plan
R-01	**MCTS Complexity and Time Consumption** **Monte Carlo Tree Search** may become too computationally expensive for larger boards or extensive simulations, potentially slowing progress.	Medium	High	- Start with smaller board sizes for initial testing. - Optimize or prune search if runtime grows too large.	- Switch to a more efficient approach or a reduced-iteration **MCTS** if performance is unacceptable.
R-02	**Decision Tree Generalization Limitations** Decision trees may overfit to training data and perform poorly on unseen board configurations or obstacle layouts.	Medium	Medium	- Train on diverse datasets with varied obstacles and board sizes. - Prune the tree to avoid overfitting.	- Revert to logic-only heuristic or hybrid approaches if decision tree accuracy is insufficient.
R-03	**Warnsdorff Algorithm Limitations** Relying solely on **Warnsdorff's** heuristic might not yield the insights or results needed, especially in heavily obstructed or specialized boards.	Medium	Medium	- Investigate multiple heuristics beyond **Warnsdorff** (e.g., alternative move-order strategies). - Compare different heuristics in parallel.	- Abandon or augment **Warnsdorff** if it proves consistently weak, adopting a newly discovered or proven heuristic.
R-04	**Team Communication Gaps** With two members working on the project, misunderstandings could arise about new ideas, potential solutions, or analysis of AI behavior.	Low	High	- Hold frequent check-ins and brainstorming sessions. - Document all algorithmic changes and testing outcomes in a shared repository.	- If misalignment is detected, schedule dedicated sync sessions to realign goals and maintain consistent progress.
R-05	**Ensuring Correct Learning** & **Transparency** Even if the AI (or backtracking/heuristics) performs well, we must illuminate *why* it works and *how* new algorithms emerge, aligning with our core research goal of discovering novel heuristics.	Medium	High	- Capture the system's decision-making rationale (e.g., state-action pairs, path expansions). - Integrate logging and post-run analysis tools for interpretability.	- If clarity is insufficient, develop more robust explanation frameworks or conduct deeper manual trace analyses.
R-06	**Neural Network Implementation Challenges** Implementing **neural networks** for policy and value predictions was complex and time-consuming, requiring additional integration work with **PUCT**.	Low	Medium	- Allocate focused time for model development and testing. -Leverage **PyTorch's** debugging tools to streamline training.	- If **NN** complexity overruns schedule, refocus on logic-only or decision tree approaches to maintain project goals.

User Requirements Document (URD)

Functional Requirements

Req. ID	Requirement Description	Priority (L/M/H)
FR-01	The simulation shall allow creating an n-by-n board with customizable obstacle placement, displaying how each algorithmic decision is made to reveal its underlying heuristic logic.	High
FR-02	Different approaches and algorithms should be easy to implement in the simulation code to be able to build up our approaches.	Medium
FR-03	The ability to provide parameter tuning capabilities for algorithms (e.g., exploration constants, heuristic weights) to facilitate debugging and performance optimization.	High
FR-04	The system shall display or output a result if no valid tour is found under current settings with clear view of its approach and path.	Low
FR-05	Backtracking algorithms should be traceable and easy to follow to understand its prioritizing path decisions.	Medium

Non-Functional Requirements

Req. ID	Requirement Description	Priority (L/M/H)
NFR-01	Performance: The system should handle large boards within a reasonable time limit.	High
NFR-02	Memory Usage: The system shall manage memory efficiently to accommodate computationally intensive algorithms such as **MCTS**, **PUCT**, or **Decision Tree** evaluations.	Medium
NFR-03	Computational Resources (**CPU/GPU**): We can leverage different approaches, such as parallelizing **MCTS** threads, exploring multiple paths simultaneously and reducing overall runtime. However, this must be implemented with caution, as performance gains will vary based on each system's hardware specifications.	Medium
NFR-04	Reliability: Critical functions (board generation, move validation) must be tested to minimize errors and ensure consistent results.	High
NFR-05	**Maintainability:** Given that AI can learn different strategies over time, these approaches should be learned so it must be easily accessible for future reference, especially when refining or replicating high-performing training runs for tracing different new heuristics.	Medium
NFR-06	**Expandability:** New algorithms and heuristics should be **traceable** and **easily accessible**, ensuring that they can be added to the existing system without major refactoring and located or referenced quickly for further analysis.	Medium

Project planning

Step	Task	Timeframe	Description & Goals
1	**Investigating MCTS with Warnsdorff-Like Behavior**	~1 Month	- **Verify Hidden Heuristics**: Dive into the **MCTS** code to confirm whether it truly follows **Warnsdorff's** rule or if it's discovering an even more efficient heuristic. - **Comparative Analysis**: Compare its results directly to the original **Warnsdorff's** algorithm, examining why the **MCTS**-based approach is performing better. - **Algorithmic Insights**: Attempt to identify or extract any "hidden" algorithm that might be driving these unexpected improvements.
2	**Tune Q-Learning & Integrate Neural Network Approaches**	~1 Month	- **Q-Learning Improvements**: Adjust hyperparameters, reward structures, or exploration strategies to enhance the Q-Learning agent's performance, which so far has been suboptimal. - **Neural Network Exploration**: Incorporate a basic NN to see if it can boost results or uncover unique strategies. - **Heuristic Discovery**: Investigate whether the NN-augmented or tuned Q-Learning reveals hidden heuristics worth spotlighting.
3	**Begin AlphaZero-Style Methods**	~1 Month	- **AlphaZero Integration**: Combine a valuable network with MCTS in a self-play setting. - **Heuristic Generation**: Observe if AlphaZero's iterative approach yields new heuristics or improves upon existing methods. - **Performance Tracking**: Evaluate how well this approach scales relative to the classic Warnsdorff or MCTS-only solutions.
4	**Performance & Further Enhancements**	If Time Allows	- **Code Optimization**: Reduce runtime overhead across all algorithms (MCTS, Q-Learning, neural models). - **Hybrid Approaches**: Investigate combining different heuristics or algorithms for a more robust final solution. - **Final Integration**: If feasible, finalize an approach that balances interpretability (uncovering hidden heuristics) and performance (covering larger boards efficiently).

References

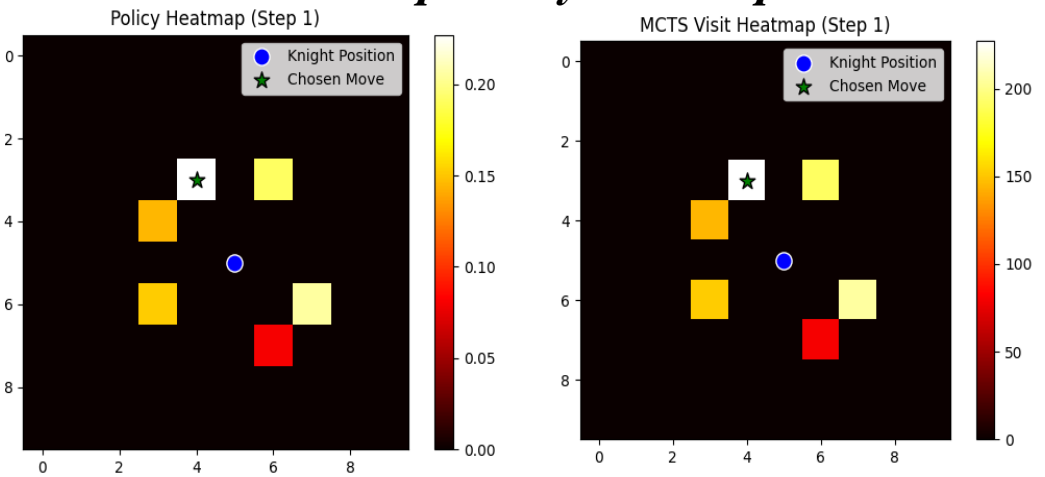
1. Wu, Y., Song, W., Cao, Z., Zhang, J., & Lim, A. (2019). *Learning Improvement Heuristics for Solving Routing Problems*. Retrieved from https://arxiv.org/pdf/1912.05784

2. Minton, S., Johnston, M. D., Philips, A. B., & Laird, P. (1992). *Minimizing conflicts: A heuristic repair method for constraint-satisfaction and scheduling problems*. NASA Technical Memorandum 108121. Retrieved from https://ntrs.nasa.gov/api/citations/19930006097/downloads/19930006097.pdf

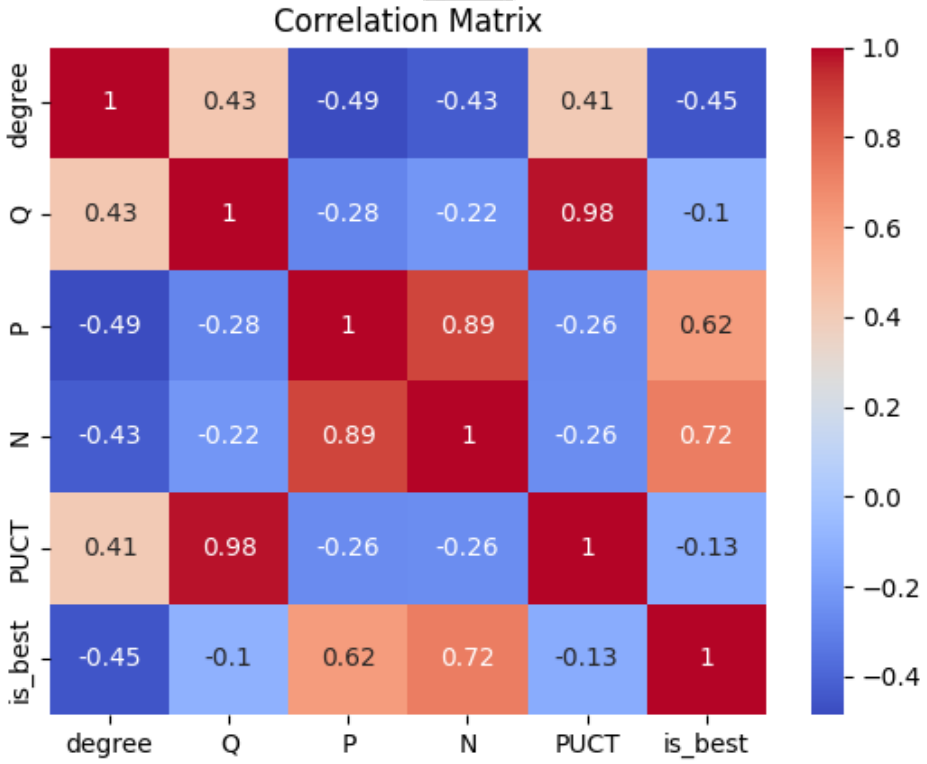
3. Zhang, Y., Tiňo, P., Leonardis, A., & Tang, K. (2021). *A Survey on Neural Network Interpretability*. Retrieved from https://arxiv.org/pdf/2012.14261

4. Cruz, C. M., & Mitra, B. (2023). *Exploring neural activity and behavior through real-time analysis*. *Proceedings of the National Academy of Sciences*, 120(2). Retrieved from https://www.pnas.org/doi/pdf/10.1073/pnas.2206625119

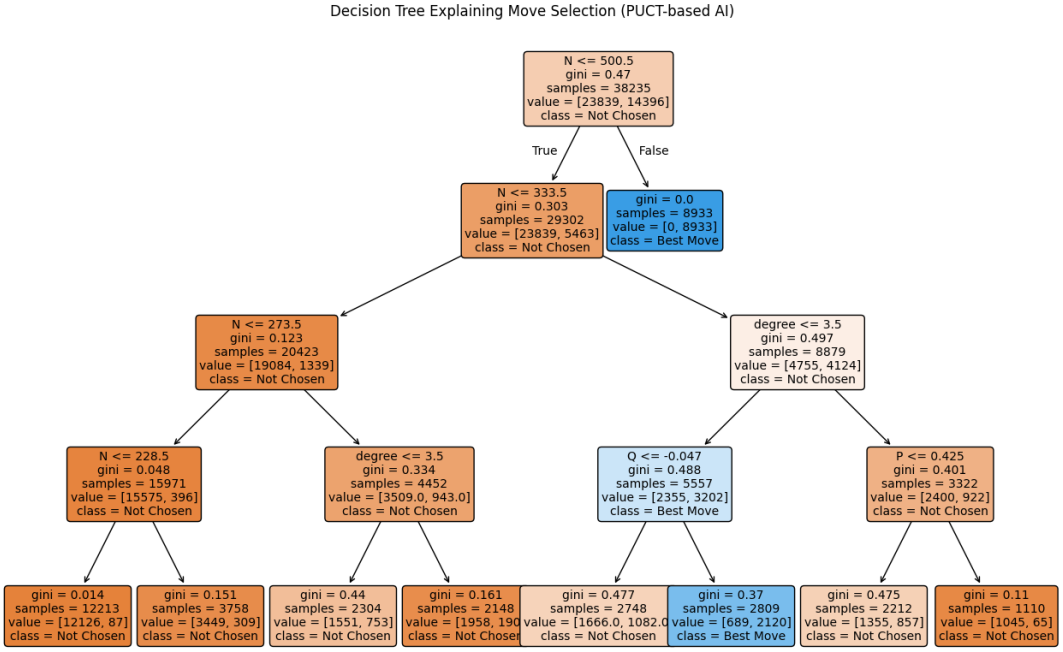
Transparency Heatmaps



Correlation matrix of PUCT Logic-Only features and best move selection



Decision tree showing feature-based move classification learned from PUCT AI data



Extracted decision tree rules guiding move decisions on the board

```
Decision Tree Rules:

|--- N <= 500.50
|   |--- N <= 333.50
|   |   |--- N <= 273.50
|   |   |   |--- N <= 228.50
|   |   |   |   |--- class: 0
|   |   |   |--- N >  228.50
|   |   |   |   |--- class: 0
|   |   |--- N >  273.50
|   |   |   |--- degree <= 3.50
|   |   |   |   |--- class: 0
|   |   |   |--- degree >  3.50
|   |   |   |   |--- class: 0
|   |--- N >  333.50
|   |   |--- degree <= 3.50
|   |   |   |--- Q <= -0.05
|   |   |   |   |--- class: 0
|   |   |   |--- Q >  -0.05
|   |   |   |   |--- class: 1
|   |   |--- degree >  3.50
|   |   |   |--- P <= 0.43
|   |   |   |   |--- class: 0
|   |   |   |--- P >  0.43
|   |   |   |   |--- class: 0
|--- N >  500.50
|   |--- class: 1
```

While decision trees may not capture the full generalization power of neural networks, they provide **high transparency and deterministic reproducibility**, supporting our goal of interpretable solutions.

Example Performance Estimates of Each Approach Across Obstacle Levels

Number of Obstacles	*Warnsdorff Coverage*	*MCTS Coverage*	*PUCT NN Coverage*	*Logic-Only Coverage*	*Decision Tree Coverage*
0	*~99%*	*~100%*	*~95%*	*~100%*	*~99%*
5	*~95%*	*~98%*	*~90%*	*~98%*	*~95%*
10	*~88%*	*~95%*	*~85%*	*~95%*	*~90%*
15	*~80%*	*~90%*	*~80%*	*~90%*	*~80%*
20	*~70%*	*~85%*	*~75%*	*~85%*	*~70%*
25	*~60%*	*~80%*	*~65%*	*~80%*	*~60%*

"The knight stands as both piece and machine; a silent AI whose hidden strategies we seek to uncover, revealing logic behind every move"