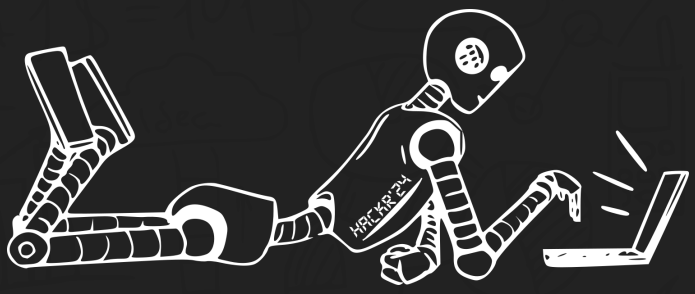
HackResearch

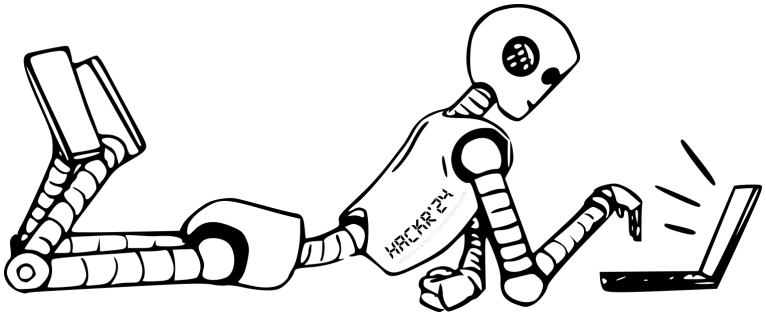
Proceedings of the 5th Hack Research Hackathon
University of Texas Rio Grande Valley

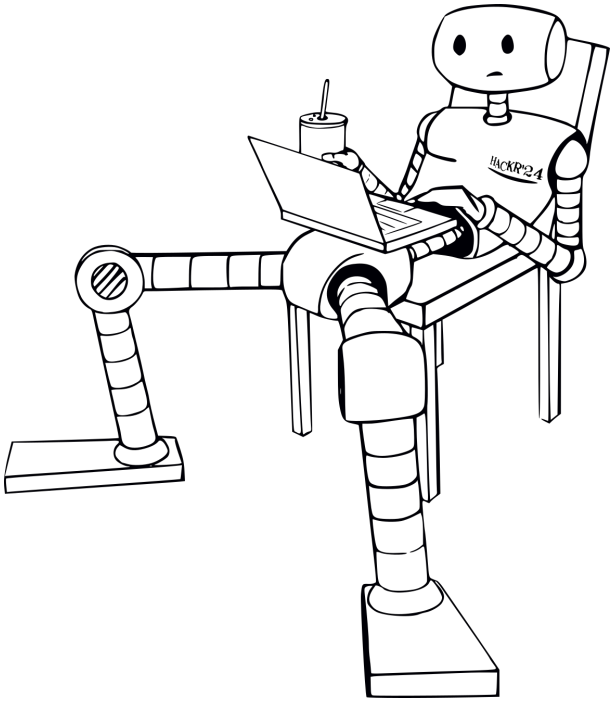


HackR 2024

Tim Wylie, Editor

HACKRESEARCH





Copyright © 2024 for the individual papers by the papers' authors. Copying permitted only for private and academic purposes. This volume is published and copyrighted by its editors.

Preface

Given the success of the previous year, this year kept a similar blueprint for the overal organization of the event. We kept the same venue and many smaller improvements from the previous year. We had breakout rooms for the problems, a great utilization of Discord, many faculty volunteers, and a lot of great SWAG.

We changed the prizes to be giftcards for simplicity, and we got rid of ordered winners. Given the different categories and types of research, it was too difficult to select who was the overall winner in terms of the best submission. Thus, we gave "first place" prizes to five different projects as winners in their respective categories. This was a great change that will be retained for future events.

This year a majority of students on papers were graduate students. This is partly due to UTRGV Computer Science starting a Ph.D. program in CS, and partly because of some of the graduate elective courses that are pushing research.

Although there was great engagement from the faculty for questions, many were unable to actually attend the event. All research questions were posed by faculty and students in various research groups. Finding and proposing problems requires a difficult balance between it being interesting and yet approachable. Attempting to find such open problems is a struggle when the problems should be nontrivial.

Many people contributed to the success of the event, and I have tried my best not to miss anyone in the acknowledgements section.

November 9-10, 2024
Edinburg, TX, United States

Tim Wylie, Editor

Acknowledgements

This event would not have been possible without the help of many people of whom we are extremely grateful to know. Without a doubt, the event would not have been possible without the understanding and support of the Computer Science department and the support staff. They have consistently supported and helped organize, fund, and create problems for the event.

The participating students also deserve acknowledgement for putting in many hours of hard work simply for the challenge and joy of it. The event would not function without their willingness to dedicate a weekend to being exhausted and asked to tackle problems they have no experience with.

I must also thank my family, who have always been supportive despite the amount of time this event takes. Their understanding and patience can not be overstated.

Finally, we are grateful for the sponsors who gave us the support we needed to make the event a reality. As mentioned, the Computer Science department at UTRGV backed the effort both monetarily and with the support of faculty and staff, and Google provided funding for outreach events through exploreCSR.

Sponsors

www.utrgv.edu/csci

research.google/outreach/explorecsr/

Organization

$\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024 was organized by the Algorithmic Self-Assembly Research Group (ASARG) with the help of the Department of Computer Science at the University of Texas Rio Grande Valley. The event was graciously hosted by UTRGV in Edinburg, TX, and funded by an exploreCSR grant from Google.

The Program Committee are the faculty that judged the submissions. Those faculty also submitted problems along with the faculty listed in Volunteers. The ASARG members are default volunteers.

Director

Tim Wylie	timothy.wylie@utrgv.edu	U. Texas Rio Grande Valley

Program Committee

Carlos Pena-Caballero	carlos.penacaballero01@utrgv.edu	U. Texas Rio Grande Valley
Robert Schweller	robert.schweller@utrgv.edu	U. Texas Rio Grande Valley
Emmett Tomai	emmett.tomai@utrgv.edu	U. Texas Rio Grande Valley

Volunteers

Marzieh Ayati
Sergei Chuprov
Erik Enriquez
Yifeng Gao
Qi Lu
Odette Perez
Li Zhang

ASARG Members

Alberto Avila-Jimenez
Divya Bajaj
David Barreda
Jose Luis Castellanos
Ryan Knobel
Aiden Massie
Adrian Salinas
Pablo Santos
Ramiro Santos

Results

Winners

This year we adopted a new format to give a prize to the top submission based on different research categories. Thus, prizes were given based on the categories of theory, data mining, and machine learning. An additional two prizes were given based on overall quality. The award winners were:

- Attacking a Game — Tak: A Beautiful Game!
  Adrian Salinas, Enrique García, Christian Peña, Angel Reyes
- Human Conflict via Simulative LLM
  Ramiro Santos, Mya Berlanga
- The Reachability Problem in (2,0) Void iCRNs with single inhibitor species is polynomial-time solvable
  Divya Bajaj, David Barreda, Ryan Knobel
- OrbitDrive: Reinforcement Learning for Circular Navigation
  Hector Lugo, Arturo Meza Canales, Jorge Orta
- How Hard can it be to Inscrypt a card game?
  Jose Luis Castellanos, Pablo Santos

Table of Contents

Evaluating the Accuracy of Large Language Models in Extracting Election Data 1
Damian Villarreal, Pigar Biteng, Miguel Garcia . . . . . .

Attacking a Game — Tak: A Beautiful Game! 4
Adrian Salinas, Enrique García, Christian Peña, Angel Reyes . . . . . .

Hack-Research: Data Extraction and Verification with Generative AI 6
Lillian Lopez, Chris Hinojosa, Koriel Lopez . . . . . .

Identification of Causal Relationships and Crosstalk Between Different Post Translational Modifications (PTMs) 8
Arely Solis, Belinda Alvarado . . . . . .

Arithmetic in dTAM 10
Alberto Avila Jimenez, Luis Martinez . . . . . .

Identifying Hidden Motifs 12
Oziel Sauceda, Andrea Garza, Mario Camarena . . . . . .

Improving the Performance of a Reinforcement Learning Agent in a Circular Road Environment: An Analysis of Optimization Strategies 14
Diana Castilleja, Alissen Moreno . . . . . .

Human Conflict via Simulative LLM 17
Ramiro Santos, Mya Berlanga . . . . . .

How Hard can it be to Inscrypt a card game? 19
Jose Luis Castellanos, Pablo Santos . . . . . .

Optimal Strategies in Tic-Tac-Toe Using Monte Carlo Tree Search 22
Robert Salazar, Joshua Hernandez-Noriega . . . . . .

OrbitDrive: Reinforcement Learning for Circular Navigation 25
Hector Lugo, Arturo Meza Canales, Jorge Orta . . . . . .

Leveraging Monte Carlo Tree Search: Enhancing Yavalath Gameplay with AI 28
Paul Hitchcox, Jaycee Sanchez . . . . . .

Ramsey Theory Analysis of Void Chemical Reaction Networks 30
Divya Bajaj, David Barreda, Aiden Massie . . . . . .

Reachability of (1,1) Conditional iCRN is NP-Complete and Equivalency over alternative CRN's under Void Rules 32
Jose Luis Castellanos, Pablo Santos . . . . . .

The Reachability Problem in (2,0) Void iCRNs with single inhibitor species is polynomial-time solvable **34**
Divya Bajaj, David Barreda, Ryan Knobel . . . . . .

Legend of Zelda: Echoes of Wisdom **36**
Ryan Knobel, Gaukhar Nurbek, Juan Manuel Perez . . . . . .

Data Extraction and Verification with Generative AI **38**
Ricardo Balvanera, David Medina, Patricio Rodriguez . . . . . .

Evaluating the Accuracy of Large Language Models in Extracting Election Data

Damian Villarreal * Pigar Biteng † Miguel Garcia ‡

Abstract

This paper investigates the effectiveness of large language models (LLMs) in extracting structured election data from unstructured text and compares the results against verified real-world data. Focusing on U.S. election information, we employed three specific prompts to extract key data points: early voting start and end dates with durations for each state, polling place hours for each state, and the dates of the U.S. general election for each past election term. The extracted data was then validated using publicly available resources from trusted sources, such as government websites and reputable organizations, to assess the accuracy and reliability of the LLMs. Our findings reveal both successes and discrepancies in the LLM-generated data, shedding light on the limitations of current AI models for precise data extraction in real-world applications. This study contributes valuable insights into improving the accuracy and utility of LLMs for structured data generation, particularly for time-sensitive and verifiable domains like elections.

1 Introduction

In recent years, large language models (LLMs) have demonstrated remarkable potential in natural language processing, especially when it comes to extracting and organizing information from vast amounts of unstructured text. This capability offers exciting possibilities for automating data collection and analysis in various fields, including the critical domain of elections. However, the accuracy and reliability of LLMs in extracting precise and verifiable information, particularly in high-stakes areas like elections, need careful evaluation. While LLMs can potentially streamline access to crucial election-related information and enhance transparency, it is essential to understand their limitations and potential biases to ensure responsible and ethical implementation.

This study aims to address this gap by assessing how effectively LLMs can extract structured election-related data from unstructured online sources and how closely that data aligns with verified information from trusted sources. We focus on U.S. election information, examining key data points such as early voting periods, polling hours, and historical general election dates. By comparing LLM-generated data with verified information from official sources, we aim to identify both the strengths and weaknesses of these models in handling factual and time-sensitive data.

To investigate this, we followed a multi-step process. First, we designed specific prompts to guide the LLM in gathering election-related data from unstructured text across the internet. The LLM was then tasked with organizing this information into a structured format for easier comparison and validation. Finally, we cross-referenced the LLM-generated data with verified information from reputable government organizations and other authoritative sources to evaluate its accuracy.

The primary goal of our study is to determine the reliability of LLMs in extracting and structuring election data and comparing their outputs with real-world information. By focusing on election data as our domain, we provide a concrete assessment of how LLMs perform in a field that demands precision and trustworthiness. This process allowed us to pinpoint both successful extractions and areas where the LLMs fell short, offering valuable insights into how these models might be improved for future use in time-sensitive and fact-based applications.

2 Related Work

While there is growing research on the applications of LLMs in various domains, studies specifically evaluating their accuracy in extracting structured data from unstructured text, particularly in elections, remain limited. However, several works provide relevant background and context for our study.

Bandi et al. [1] provide a comprehensive overview of generative AI, including LLMs, discussing their requirements, models, input-output formats, and evaluation metrics. This work highlights the challenges of implementing AI systems in real-world scenarios and emphasizes the importance of robust evaluation metrics to assess the quality and reliability of AI-generated outputs. Their insights into the potential pitfalls and evaluation frameworks for generative AI inform our approach to assessing the accuracy of LLMs in the specific context of election data extraction.

In addition to general evaluations of LLMs, research on information extraction and knowledge base population provides valuable insights. For example, studies on automatic extraction of factual information from text, such as relation extraction and event extraction, highlight the complexities involved in accurately identifying and structuring data from unstructured sources (e.g., [[5], [2]]). These works underscore the need to carefully evaluate LLM performance in similar tasks, particularly when

*Department of Computer Science, University of Texas Rio Grande Valley, damian.villarreal01@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, pigar.biteng01@utrgv.edu

‡Department of Computer Science, University of Texas Rio Grande Valley, miguel.garcia35@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

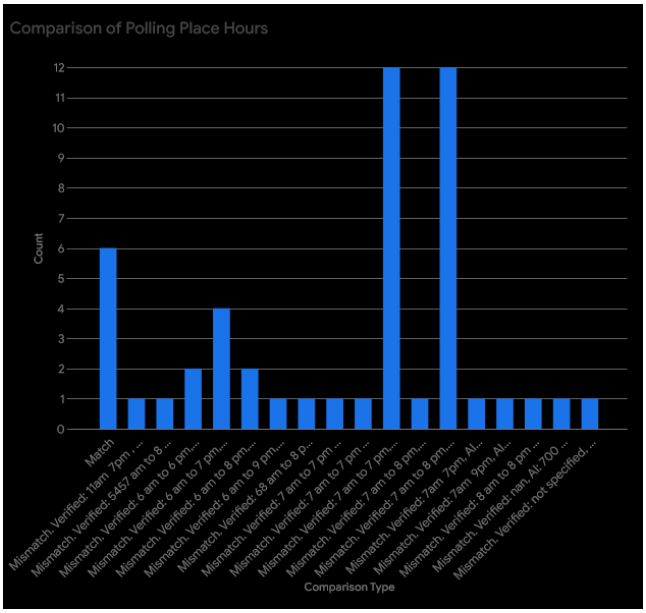


Figure 1: Comparison of polling place hours. There were only 6 matches between the verified dataset and the AI generated dataset.

dealing with dynamic and verifiable information like election data.

Furthermore, research on the use of AI in the electoral context, such as [[3], [4]], provides valuable perspectives on the potential benefits and risks of applying AI technologies in this sensitive domain. These studies emphasize the importance of transparency, accountability, and ethical considerations when deploying AI systems in elections, further highlighting the need for rigorous evaluation of LLM accuracy and reliability.

We observed that the AI model frequently generated dates that did not align with the verified data, particularly for 'Early Voting End Date'. In 10 instances, the AI model predicted an end date of November 5th, 2024, while the verified data indicated November 4th, 2024. The AI model also introduced several dates not present in the verified data, such as October 24th, 2024, for 'Early Voting Start Date'. Additionally, we found that the AI model often generated specific dates where the verified data did not have an exact date, indicating potential limitations in the AI's ability to discern nuanced or varied scheduling information.

In several instances, the AI model returned NaN values while the corresponding verified data contained specific dates, underscoring potential gaps. We observed that the AI model generated polling place hours that were different from the verified data. In most cases, the difference was in formatting. For example, the verified data frequently used formats like '7 am to 7 pm', while the AI model often opted for formats like '700 am to 700 pm'. While these discrepancies primarily involve stylistic choices, they underscore potential variations in how the AI model processes and presents time-related information.

We observed that the AI model generated general election dates that perfectly matched the verified data across all 12 entries. This complete agreement highlights the AI model's strong ability to accurately extract and reproduce standardized date-related information, particularly for events with fixed and well-defined schedules.

3 Future Works

Future work will focus on exploring additional avenues to refine and expand the application of LLMs in extracting structured information from unstructured text. A primary direction is to evaluate and compare the performance of various LLMs in extracting data across diverse topics, beyond election-related information. For instance, testing these models on other time-sensitive or fact-based domains, such as health policy updates, legal documentation, or real-time market data, could provide further insights into the versatility and limitations of different LLMs for structured data extraction. Moreover, investigating alternative prompting strategies and model fine-tuning may enhance the models' consistency and accuracy in domains where data precision is critical. Future research could also involve assessing hybrid approaches that combine LLMs with rule-based or knowledge-based systems to validate AI-generated outputs against verified sources. Expanding this work to include evaluations of model performance in multiple languages and regional contexts would provide a broader understanding of LLM utility in international data extraction tasks. These future studies aim to contribute to a robust methodology for developing and deploying LLMs in real-world applications where the accuracy of extracted information is essential for decision-making and public trust.

4 Conclusion

This study investigated the effectiveness of Large Language Models (LLMs), specifically the Gemini Advanced AI model, in extracting structured election data from unstructured text and compared the results against verified real-world data. We focused on U.S. election information, employing three specific prompts to extract key data points: early voting start and end dates with durations for each state, polling place hours for each state, and the dates of the U.S. general election for each past election term.

Our findings revealed a mixed bag of successes and discrepancies in the LLM-generated data. While the model demonstrated high accuracy in extracting standardized and fixed-schedule information, such as the dates of past U.S. general elections, it encountered challenges with more nuanced or variable data, like early voting periods and polling place hours. These challenges highlight the

limitations of current LLMs in handling real-world data that may be subject to changes, exceptions, or ambiguities.

Despite these limitations, this study contributes valuable insights into the potential of LLMs for automating data extraction and analysis in the election domain. By identifying specific areas where the model excelled or fell short, we provide guidance for future research and development aimed at improving the accuracy and reliability of LLMs for structured data generation.

Future research could explore several avenues to build upon our findings. One direction would be to investigate the impact of prompt engineering on the accuracy of LLM-generated election data. By refining the prompts used to guide the model, we may be able to elicit more precise and contextually relevant information. Another avenue would be to explore the use of multiple LLMs or sense mble methods to leverage the strengths of different models and potentially mitigate their individual weaknesses.

Additionally, future studies could delve deeper into the reasons behind the discrepancies observed between LLM-generated data and verified real-world data. This could involve analyzing the model's internal representations and decision-making processes to understand how it handles ambiguities and e xceptions in the data. Such analysis could lead to more targeted interventions for improving the model's performance.

In conclusion, this study provides a starting point for understanding the capabilities and limitations of LLMs in the context of election data extraction. While there are challenges to overcome, the potential benefits of this technology are significant. By continuing to refine and improve LLMs, we can harness their power to enhance transparency, accessibility, and efficiency in the electoral process.

References

[1] A. Bandi, P. V. S. R. Adapa, and Y. E. V. P. K. Kuchi. The power of generative ai: A review of requirements, models, input–output formats, evaluation metrics, and challenges. *Future Internet*, 15(8):260, 2023.

[2] H. Ji and R. Grishman. Knowledge base population: the story so far. In *Proceedings of the 2010 workshop on Knowledge base population*, pages 1–10, 2010.

[3] D. Kreiss and S. C. McGregor. Prototype politics: Technology-intensive campaigning and the data of democracy. *New Media & Society*, 18(11):2681–2699, 2016.

[4] H. Margetts, P. John, S. Hale, and T. Yasseri. *Political turbulence: How social media shape collective action.* Princeton University Press, 2017.

[5] F. Wu and D. S. Weld. A comprehensive study on relation extraction. *Proceedings of the 2010 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1199–1208, 2010.

Attacking a Game — Tak: A Beautiful Game!

Adrian Salinas * Enrique García † Christian Peña ‡ Angel Reyes §

Abstract

For this study we are trying to show how the complexity of a game can affect the ability for a simple AI to not only be created for it, but also be able to learn the game and its mechanics. In Theorem 1, we show the complexity of a restricted version of the game Tak. In section 3, we then show the process of creating a simple AI that can not only read how the game is played, but also be able to play the game against a human opponent to test competence of this simple AI.

1 Introduction

1.1 Restricted Tak

We initially tried to prove PSPACE-hardness for the generalized version of the game *Tak*, but it was difficult to start making progress, so we restricted the rules in Tak. In generalized Tak, you have a $n \times n$ board with square tiles, and the goal is to get your pieces (Red or Blue) to form a path from one side of the board to the other. In normal Tak, you can place, move, and stack 1×1 pieces, but in this restricted version of Tak, we are saying you cannot do this. You can only place 3×3 pieces, and they will remain unchanged for the rest of the game.

1.2 Hex

To prove PSPACE-hardness we are reducing the game *Hex* to the restricted Tak game. In Hex, you have a $n \times n$ board with hexagonal tiles, and the goal is to get your pieces (Red or Blue) to form a path from a specific side of the board to the other. For instance, in Figure 1, the Blue player can only win if they form a path from the left side of the board to the right. If blue formed a path from the top to the bottom, nothing happens. In Hex, you're only allowed to place your pieces.

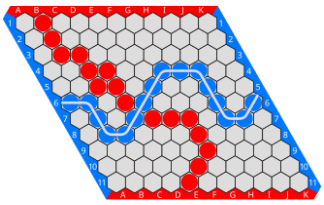


Figure 1: Example of Blue winning a game of Hex. (Image from Wikipedia)

2 Our Results

Theorem 1 *Determining if a player has a forced win in a restricted game of Tak is PSPACE-hard.*

Proof. Given the restrictions we put on the game Tak, it is easy to see the high level idea used to reduce Hex. The main differences between restricted Tak and Hex is the number of neighbors each piece has and the win condition. In Tak, each piece has four neighbors, whereas Hex pieces have six neighbors. So, to complete the reduction we need to create a Tak board where the pieces have six neighbors and each player's goal is to create a path across a specific side of the board. Figure 2 shows an example of how these conditions are simulated. The pattern throughout the board gives each 3×3 Tak piece six neighbor, and the empty spaces (white) ensure that no players can cheat. And the sides are lined with the respective players' pieces to enforce the win condition from Hex. A $4n + 1 \times 4n + 1$ Tak board is needed to simulate a $n \times n$ Hex board.

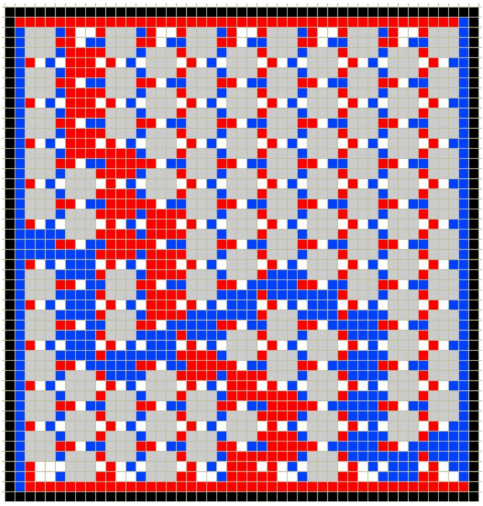


Figure 2: Example restricted 45×45 Tak simulating the 11×11 Hex example in Figure 1. The grey spaces represent the valid areas where 3×3 pieces can be placed. White spaces are just empty.

*Department of Computer Science, University of Texas Rio Grande Valley, adrain.salinas08@utrgv.edu
†Department of Computer Science, University of Texas Rio Grande Valley, enrique.garcia06@utrgv.edu
‡Department of Computer Science, University of Texas Rio Grande Valley, christian.pena02@utrgv.edu
§Department of Computer Science, University of Texas Rio Grande Valley, name2@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

□

3 Simple AI Code

This Python program implements a simplified version of the board game "Tak: A Beautiful Game". In this version of the game, it allows to play Tak either with and AI or with another player present using the 3x3 format of the game where you can use "flat stones" and "standing stones". The AI opponent uses a basic memory system to learn from previous games by saving winning and losing games in a file. Each game that the AI plays against the player, it learns and applies the strategies to future games.

GitHub Repository Here

https://github.com/AceMonke002/HackResearchTakGame

You can access our code as a .txt file by clicking HERE (you need to download the LaTex as a pdf first).

4 Tile Assembly Arithmetic

Theorem 2 *A dTAM system can perform unary addition, given a number n and m.*

Proof. You can perform unary addition using an dTAM system with two tiles, N and M. Both tiles have the same glues structure, where there is a glue labeled 1 on the left and right sides of the tiles. There will be n-amount of N tiles, and m-amount of M tiles. The output is the length of the string of tiles that was formed. You When the systems finishes running, the length of the string will be $n + m$. □

Theorem 3 *A dTAM system can perform unary subtraction, given a number n and m.*

Proof. You can perform unary subtraction with two tiles, N and M. The tiles should have a glue structure such that the N with form a horizontal line, and the M tiles will form a horizontal line below the N tiles. There will be n-amount of N tiles and m-amount of M tiles. The output is the length of empty space made □

Theorem 4 *A dTAM system can perform unary multiplication, given a number n and m.*

Proof. □

Theorem 5 *A dTAM system can perform unary division, given a number n and m.*

Proof. □

5 Conclusion

Using another similar game that is already proven to be PSPACE-hard, we were able to reduce it to a restricted version of Tak. Given our findings in how restricted Tak is PSPACE-hard, one could argue that generalized Tak is also PSPACE-hard since we showed this is true for a weaker model. Admittedly though, this is not a rigorous proof for generalized Tak, but it is an interesting observation. This is something that can also warrant future study with trying to identify the complexity of the original game of Tak where the rules create many more restrictions on both player and AI in our case. It can also be used to try and identify complexities of other types of games besides just board games or puzzle games. Given the fact that if these games are able to be proven as one of our complexity/difficulty designations, it is possible to create a form of learning AI for it along with measuring how quickly these AIs can learn from constant improvements.

Hack-Research: Data Extraction and Verification with Generative AI

Lillian Lopez * Chris Hinojosa † Koriel Lopez ‡

Abstract

Viewing data and verifying it from an unstructured text is a fundamental challenge in modern computer applications, particularly when using generative AI models that lack an inherent understanding of the structured data they manipulate. When using Large Language Models, it offers an approach by leveraging its foundation knowledge to perform one-shot and zero-shot learning. This study explores the capacity of LLMs to extract and structure data from unstructured text sources, such as articles, forums, and user interactions, transforming it into a format suitable for quantitative analysis. We address the challenge of data validation, investigating methodologies to verify the accuracy of AI-generated structured content. Leveraging LLMs with real-time access to current online sources, this study investigates their ability to extract data from an unstructured text, such as websites and documents. We will be highlighting the potential of LLMs data structure process while emphasizing the necessity for rigorous verification to ensure data reliability in AI-driven applications.

1 Introduction

The integration of structured data into computer applications has long been a critical challenge, central to their functionality and performance. From the early days of computing, the need for massive, well-organized datasets has been apparent, especially with the advent of Machine Learning techniques that require extensive training data. Whether it's a scheduling program managing events and tasks, a vehicle control system interpreting sensor data, or a game tracking the state of its simulated world, the manipulation and validation of data remain at the heart of every software application.

Generative AI models, particularly those designed to handle unstructured text, have emerged as powerful tools for bridging the gap between raw data and structured information. These models excel at predicting the next most probable term based on extensive human text corpora, enabling them to generate contextually appropriate responses. However, their utility often stops at generating unstructured text, which poses a challenge for further computational use.

Despite this, the versatility of Generative AI inputs and outputs offers significant potential. By refining prompts to include directives such as "format this as a table" or examples of desired output structures, these models can produce structured data from unstructured sources. Furthermore, they can generate executable code, facilitating validation and integration into various applications.

This paper explores the capabilities of Large Language Models (LLMs) in extracting structured data from unstructured text, suitable for computational tasks or programmatic integration. It involves selecting a domain rich in accessible text, extracting structured datasets for quantitative analysis or application use, and employing validation techniques to ensure data accuracy. The exploratory nature of this study aims to reveal the potential and limitations of AI-driven data extraction and verification, contributing to the broader understanding of AI applications in data management. [2]

We briefly highlight some related work in Section 3, and then provide the definitions and results of our work in Section 3. We then conclude in Section 4 and point towards the general research goals for this work [4].

2 Implementation Evaluation

Implementation: These computations were powered by the Microsoft's Copilot model, utilizing a dynamic database for searches due to it's limited access to the web. The generated Ai responses are based on the input of the user and the resources to ensure accurate and relevant answers.

Copilot is specifically designed to perform web searches, enabling it to access and extract dynamic information from web pages. It can provide real-time updates based on this information, as demonstrated by the image below.

Mind the fact that we gave it a simple task for searching information on it's own, and while the data given may have been relevant, it was not accurate. The temperature during the demonstration was around 84F, contrary to the 76F the AI claimed it to be. Leveraging advanced

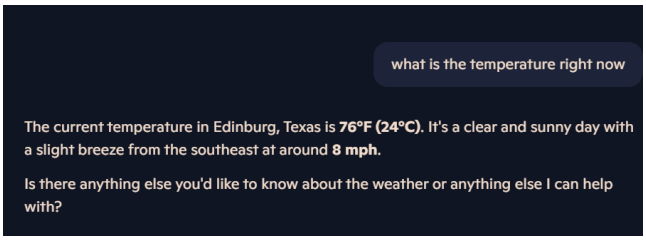


Figure 1: Weather Example w/ Copilot

capabilities of Copilot, which dynamically extracts information from both internal and external sources via simple link to a valid website, it provides real-time online access

*Department of Computer Science, University of Texas Rio Grande Valley, lillian.lopez02@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, Christopher.Hinojosa06@utrgv.edu

‡Department of Computer Science, University of Texas Rio Grande Valley, koriel.lopez01@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

to generate accurate, structured data from unstructured text.

Copilot was able to use the link to their advantage, but explored more to provide us additional information of the faculty member and their course. Check figure below.

This would be the ideal solution when it comes to our information getting the maximized use of what we fed it. It searched for more information of the professor with the given link.

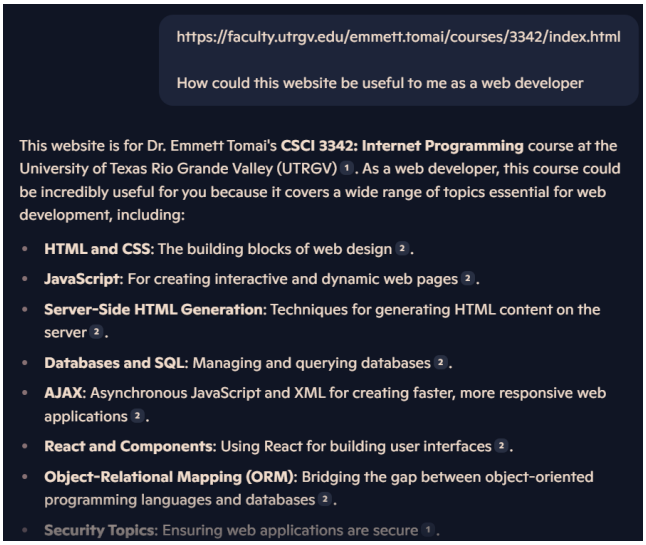


Figure 2: Tomai's website being recognized via link [3]

Copilot is indeed a powerful tool, but it has its limitations. One example is its inability to directly read and interpret graphs. Even if we try to focus on the graph, Copilot won't be able to analyze or provide insights from it. The limit is that there is no sort of image processing when it comes to reading the graph, so it won't be able to grab information directly from it. The image below shows the struggle to read from the graph. Evaluation: To validate our implementations, we selectively chose examples that demonstrate Copilot's results across different scenarios: a moderate one, an ideal one, and a challenging case that wasn't fully readable. Each example highlights unique strengths and limitations, offering valuable insights into how AI tools like Copilot can be effectively leveraged to maximize their benefits.

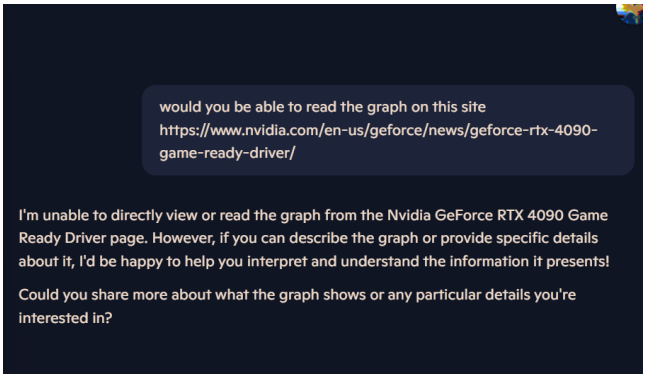


Figure 3: Copilot Struggles reading a graph [1]

3 Our Results

While Copilot's potential was not fully utilized, it effectively retrieves information from multiple webpages and summarizes the content provided. However, it may not capture all the details on each page, particularly if they include graphs, complex mathematical formulas, or other intricate data. In such cases, additional information would be required for a complete understanding.

4 Conclusion

Using an LLM like Copilot can be a powerful tool for putting together unstructured text into a structured format for the user. For our study we explored the limitations of LLMs by providing it several examples of unstructured resources to read from and produce a coherently summarized response from those. While LLMs like Copilot can do well in providing information from text based sources, they struggle with items that are image based. There's still a lot to discover in this area of computing, and it can open up a lot of opportunity for automation.

References

[1] Nvidia. News geforce rtx 4090 game ready driver, 2023.

[2] E. Tomai. Data extraction and verification w/ generative ai, 2024.

[3] E. Tomai. Emmett tomai web design website, 2024.

[4] T. Wylie. Why hack research is the best, 2019.

Identification of Causal Relationships and Crosstalk Between Different Post Translational Modifications (PTMs)

Arely Solis * Belinda Alvarado †

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

1 Introduction

Proteins are essential biomolecules that perform a diverse array of functions within cells, enabling processes ranging from structural support to signal transduction. Peptides, the building blocks of proteins, undergo numerous transformations that dictate the functional roles of their parent proteins. Among these transformations, post-translational modifications (PTMs) are particularly significant. PTMs are chemical alterations made to proteins after synthesis, introducing functional diversity and complexity by modifying amino acid residues. This diversity is vital in regulating cellular processes, including cell signaling, immune responses, and metabolic control. One characteristic feature of PTMs is their potential to interact with one another, a phenomenon termed "crosstalk." Crosstalk between different PTMs can modulate protein activity and function in a context-dependent manner, complicating our understanding of protein function. Identifying and understanding these PTM interactions are crucial for elucidating protein behavior in normal physiology and disease. However, investigating PTM crosstalk is challenging due to the complex and often transient nature of these modifications.To address this challenge, PeptideAtlas, an expansive proteomic database, provides a valuable resource for analyzing PTMs on a genome-wide scale. PeptideAtlas compiles data from high-throughput mass spectrometry experiments, providing a comprehensive catalog of peptides and their modifications across multiple species. Using PeptideAtlas, this study aims to uncover potential causal relationships and crosstalk between different PTMs. Such insights could improve our understanding of protein regulation, potentially revealing mechanisms relevant to disease and therapeutic intervention.

2 Methodology

2.1 Data Collection

The initial stage of this study involved downloading data from PeptideAtlas, a repository that archives peptides from proteomics experiments. Despite initial technical issues during data acquisition, such as download failures and file compatibility problems, the data was successfully retrieved in .mysql format. The dataset was then imported into MySQL Workbench for further analysis.

2.2 Data Structure Analysis

To effectively utilize PeptideAtlas data, it was necessary to understand the underlying schema of the database. The database comprises multiple interconnected tables, including *modified_peptide_instance* and *modified_peptide_instance_sample*, which detail essential attributes such as peptide sequences, charge states, and mass values. The relationships between these tables allow for the exploration of individual peptide characteristics as well as PTM occurrences in different samples, laying the foundation for downstream data analysis.

2.3 Data Pre-processing

Once the data was imported and the schema understood, a series of pre-processing steps were applied. These steps included:

- Cleaning and validating data entries to ensure integrity and consistency.
- Filtering the data to focus on peptides with multiple observed modifications, as these are key to studying PTM crosstalk.
- Normalizing fields like peptide charge and mass for cross-sample comparisons.

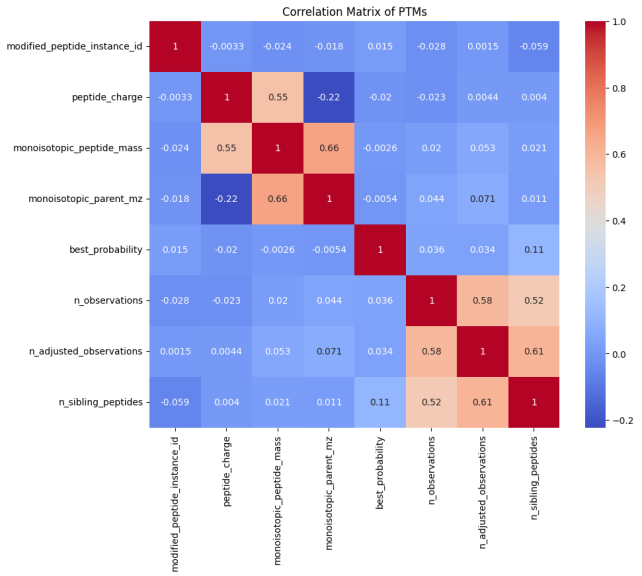
3 Our Results

The correlation matrix of post-translational modifications (PTMs) reveals several significant relationships between peptide properties and other parameters.

*Department of Computer Science, University of Texas Rio Grande Valley, arely.solis03@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, belinda.alvarado02@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.



Notably, there is a moderate correlation between *peptide_charge* and both *monoisotopic_peptide_mass* and *average_peptide_mass* (0.55), suggesting that larger peptides often exhibit higher charge states. This is consistent with the nature of peptides, where increased mass typically corresponds to additional residues or modifications, which can influence the charge.

Another critical observation is the strong correlation (0.66) between *monoisotopic_peptide_mass* and *monoisotopic_parent_mz*, as well as between *average_peptide_mass* and *average_parent_mz*. These relationships imply that the mass of peptides is closely linked to the mass-to-charge ratio of the parent ions, likely due to shared molecular characteristics that influence both attributes. Additionally, *n_observations*, *n_adjusted_observations*, and *n_sibling_peptides* exhibit strong correlations among each other, especially between *n_observations* and *n_adjusted_observations* (0.58), as well as *n_sibling_peptides* (0.52). This trend suggests that frequently observed peptides tend to have more adjusted observations and sibling peptides, indicating that peptides with multiple modifications or instances are repeatedly detected across samples.

4 Challenges and Limitations

Initial difficulties in downloading and importing PeptideAtlas data led to delays in data analysis, likely due to software and storage limitations on our laptops. With additional time and resources, we would have expect to find more evidence to help identify causal relationships and crosstalk between PTMs in humans.

5 Conclusion

This study leveraged PeptideAtlas data to explore potential crosstalk and causal relationships between PTMs. The high correlation between *monoisotopic_peptide_mass* and *average_peptide_mass* (close to 1) may suggest potential redundancy, as they likely represent similar measures. This insight could be useful for dimensionality reduction in further analyses to prevent multicollinearity issues. Overall, this matrix from figure 1 highlights important relationships within the dataset, particularly around peptide mass, charge, and observation patterns, which can guide further research into PTM characterization and analysis.

Arithmetic in dTAM

Alberto Avila Jimenez * Luis Martinez †

Abstract

In this paper, we introduce a discrete-count Tile Assembly Model (dTAM) system that performs binary addition and subtraction using finite tile types. Leveraging a limited tile inventory, we construct tile configurations that simulate binary arithmetic through self-assembly. Our design utilizes a sequential tile-binding approach to compute sums and differences without encoding results in the tile counts. Further, we analyze our approach to showcase efficient usage of tile resources and how it could be expanded into multiplication and division, demonstrating dTAM's potential for basic computational tasks within restricted environments.

1 Introduction

The discrete-count Tile Assembly Model (dTAM) is a variation of the abstract Tile Assembly Model (aTAM) that introduces tile count limitations, restricting the number of each tile type available in the system. This modification brings new challenges and opportunities, as it requires the system to achieve its goals within a finite resource environment. In this work, we extend the capabilities of dTAM by incorporating binary addition and subtraction rules to design a tile system that can perform these arithmetic operations. Our approach leverages a limited tile inventory to construct configurations that simulate binary arithmetic through self-assembly. By utilizing a sequential tile-binding approach, we compute sums and differences without encoding results in the tile counts. Further, we analyze our approach to showcase efficient usage of tile resources and how it could be expanded into multiplication and division, demonstrating dTAM's potential for basic computational tasks within restricted environments.

2 Related Work

The tile assembly model (TAM) has been extensively studied for its applications in DNA computing and nanotechnology. One notable work is by Winfree et al. [1], which formalizes the study of self-assembly and focuses on arithmetic computation within the TAM framework. Their research demonstrates the construction of tile systems capable of performing addition and multiplication with optimal tile types and linear assembly time relative to input size. We translate this into the discrete-count Tile Assembly Model (dTAM), where we extend these concepts to binary addition and subtraction under tile count constraints.

3 Our Results

In this section, we explain and analyze our dTAM constructions for binary addition and subtraction.

3.1 Addition

Given two positive integers m and n, we construct a dTAM system as follows:

We create tiles each with glues of strengths 1 or 2. In our figures, the blue glues represent a strength 2 glue and the purple glues represent a strength 1 glue. The temperature in our system is 2, meaning our tiles need a total sum of at least 2 in the glues to be able to bond together.

The tile types are as follows:

1. **Bit Tiles** Create a tile for each bit in the binary representation of m and n, shown in 2. For the sake of simplicity, we add buffer tiles at the leftmost bits. This is to account for any carry bits that might overflow out of our structure.
2. **Structure Tiles** create + tiles that help the create the structure for the result tiles
3. **Rule Tiles** create tiles for each of the binary addition rules as shown in 1. These will build our answer.

These tiles have specific counts. The initial bit tiles and the + tiles are unique, there is only one of each. We define k as the length of our binary sequence representing m in the tiles. We use this number k in the count of our rule tiles, as in the worst case we will need k of the same rule tile.

The glues on the north and south side of the m and n bit tiles have a unique indexing. For the glues labeled as m_j^i, i represents the index of the bit in the sequence and j represents the value of the bit in that tile. The glues labeled as n_l^j where j represents the value of the m bit tile above at the same index and l represents the value of the n tile

3.2 Subtraction

Subtraction follows closely to the addition construction of dTAM. Where

1. **Bit Tiles** We create bit tiles to represent m and n
2. **Structure Tiles** create − tiles that help the create the structure for the result tiles
3. **Rule Tiles** create tiles for each of the binary subtraction rules. These will build our answer.

*Department of Computer Science, University of Texas Rio Grande Valley, alberto.avilajimenez01@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, luis.martinez33@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

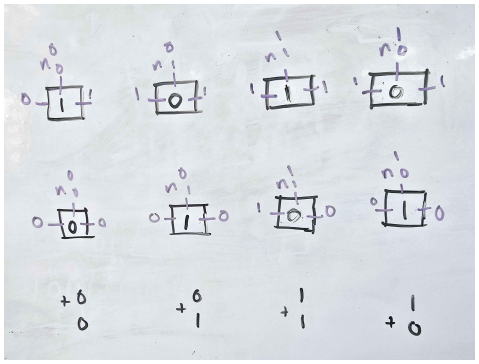


Figure 1: The tiles above have inputs on the north and east sides and an output at the west side of the tile. The north input represents a encoded glue of the values being added and the east input is the carry that happens in binary addition. The west side output also represents the carry. This covers all the binary addition permutations for two digits.

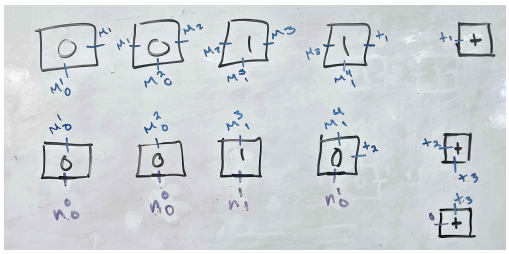


Figure 2: M and N are first encoded into their binary representations using tiles which are then glued using M type glues that ensure that the tiles are glued in the correct order while also encoding the inputs for the addition tiles.

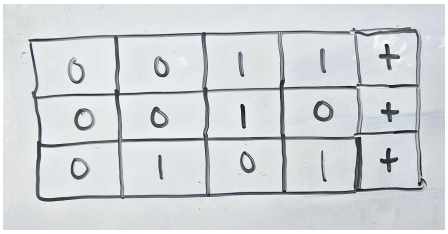


Figure 3: Final result after tiles are assembled.

3.3 Multiplication Division

Our dTAM constructions for binary addition and subtraction can be extended to perform multiplication and division. However, these extensions result in very naive approaches. For multiplication, we replicate the process of repeated addition, where each bit of the multiplier is used to conditionally add shifted versions of the multiplicand. Similarly, division is approached through repeated subtraction, where the divisor is subtracted from the dividend until the remainder is less than the divisor. While these methods work, they are highly inefficient, especially for large inputs, due to the multiple number of operations

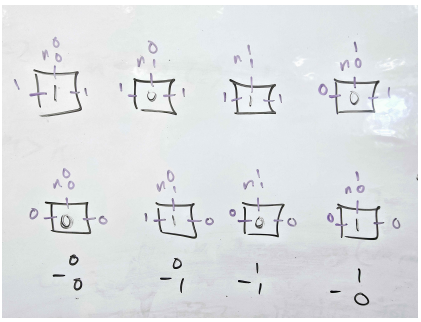


Figure 4: The tiles above show all the combinations that result from subtraction. The north side and east side represent inputs and the west side represents the tiles output. Refer to figure 1 for more information.

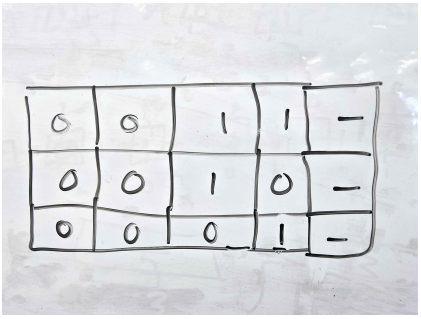


Figure 5: Final result of assembled subtraction tiles.

required. Consequently, we did not explore these operations in depth and leave them off for future work.

4 Conclusion

In this paper, we presented a discrete-count Tile Assembly Model (dTAM) system capable of performing binary addition and subtraction using a finite set of tile types. Our results demonstrate the feasibility of constructing tile configurations that simulate binary arithmetic through self-assembly. While our primary focus was on addition and subtraction, these results lay the groundwork for exploring more complex operations such as multiplication and division. However, due to the inherent inefficiencies and time constraints, we did not delve deeply into these operations. Future work could address these challenges and optimize the dTAM system for broader computational tasks.

References

[1] E. Winfree, F. Liu, L. Wenzler, and N. Seeman. Arithmetic computation in the tile assembly model: Addition and multiplication. *Theoretical Computer Science*, 410(15):1561–1571, 2006.

Identifying Hidden Motifs

Oziel Sauceda * Andrea Garza † Mario Camarena ‡

Abstract

The Time Series Pattern Hunter problem aims to derive similar patterns across a dataset. From electroencephalogram (EEG) brain waves, certain areas in their wave structure resemble similar patterns, even from different locations in that wave. To the regular eye, it is difficult to locate these similar structures, but what if we had an algorithm to help us find where these patterns take place and their length? We could identify these patterns throughout the dataset with the SCRIMP++ algorithm.

1 Introduction

To begin with, we will be referencing these recurring patterns as motifs. Our objective is to locate similar motifs and determine their position and length. SCRIMP++ helps calculate the matrix profile of a time series efficiently. Using this algorithm, we identified motifs and potential anomalies in our dataset. SCRIMP++ contains steps that ensure accurate and efficient time complexity and results. By incrementally computing a portion of the matrix profile initially and updating over time, the time complexity for this algorithm is O(nlogn).

2 Related Work

The SCRIMP++ algorithm, introduced in [1], is a recent advancement in exact motif discovery. This algorithm uses earlier methods, such as STOMP and STAMP, as a foundation on which to build a more efficient algorithm. Earlier methods contained a quadratic runtime of $O(n^2)$, where SCRIMP++ improves on this time complexity by incrementally calculating the matrix profile, allowing a linear runtime complexity of O(nlogn) along an exact matrix profile solution, making it suitable for large datasets.

3 Approach

For the Time Series Patter Hunter problem, we received a dataset of 10 million data points. To implement the SCRIMP++ algorithm, we initially extracted a subset of 50,000 points to balance computational requirements. The algorithm was implemented through a Python library named Stumpy, which efficiently computes a matrix profile in a time series. This library requires input data to be in the form of a 1-dimensional array, so we flattened the data points from a 2-dimensional array to ensure they were in the correct format. After flattening the array, we defined a sliding window to determine the length of each subsequence throughout the time series. This approach will affect patterns, and the algorithm will identify and provide the expected motif lengths we would like to find in our dataset.

Our approach continues to compute the matrix profile. We initialize our algorithm using stumpy.scrump() contains a percentage parameter of 0.1, initially computing only 10% of the matrix profile to achieve an approximate solution. An update method was utilized to incrementally refine the matrix profile by computing distance along the matrix.

Once the matrix profile is approximated, we will look for the most similar subsequences pair in the matrix. This is completed by successfully identifying the smallest value in that matrix profile and returning its index. The index represents the motifs as they contain the starting positions of the identified two closest segments.

Lastly, we apply Z-normalized to each motif segment to overlap both motifs into one graph to compare the two segments' shapes directly.

4 Our Results

In our analysis, we executed two distinct scenarios in the dataset. Given the dataset's size, we narrowed the subset to 50,000 values from a given starting point to capture a significantly large amount of data points while successfully looking for two similar segments.

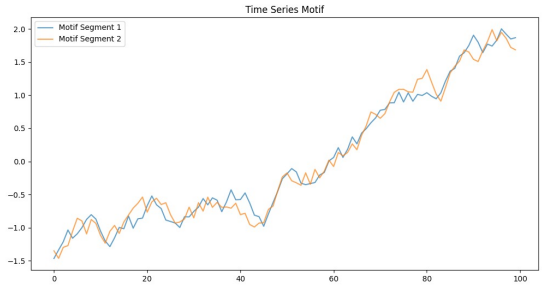


Figure 1: Similar Motifs: Range 0 - 50,000

Figure 1 contains a data point range from 0 to 50,000, while Figure 2 encompasses 1,000,000 to 1,050,000.

5 Conclusion

The SCRIMP++ algorithm was an effective approach for large-scale time series pattern recognition. Despite its large time complexity, the algorithm's incremental approach allowed us to process the large dataset within manageable computational limits. By breaking down the

*Department of Computer Science, University of Texas Rio Grande Valley, oziel.sauceda01@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, andrea.garza20@utrgv.edu

‡Department of Computer Science, University of Texas Rio Grande Valley, mario.camarena01@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

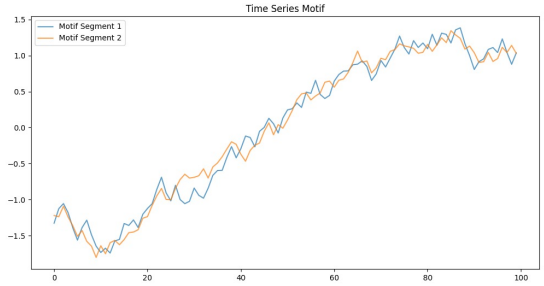


Figure 2: Similar Motifs: Range 1,000,000 - 1,050,000

large dataset into smaller portions, we were able to locate motifs, ultimately finding recurring patterns that might be invisible to the naked eye.

References

[1] Y. Zhu, C.-C. M. Yeh, Z. Zimmerman, K. Kamgar, and E. Keogh. Matrix profile xi: Scrimp++: Time series motif discovery at interactive speeds. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 837–846, 2018.

Improving the Performance of a Reinforcement Learning Agent in a Circular Road Environment: An Analysis of Optimization Strategies

Diana Castilleja, Alissen Moreno *

Abstract

In this paper, we analyze improvements to a reinforcement learning (RL) agent designed for autonomous navigation in a circular road environment. Initial training revealed consistent underperformance, as shown by frequent negative rewards and a lack of lap completions. We identified and implemented targeted modifications in the reward function, action space, exploration phase, and hyperparameter settings. These adjustments aimed to provide more effective learning feedback, smoother control, and prolonged exploration, with the goal of equipping the agent with better navigation capabilities.

1 Introduction

Reinforcement learning (RL) is increasingly applied to autonomous driving tasks, where agents must learn policies that balance efficiency, safety, and adaptability. In this study, we aim to improve an RL agent's navigation abilities in a circular road simulation where it initially struggled with effective learning, as shown by persistently negative rewards and a low lap completion rate. We identified areas for improvement in the reward function, action space, exploration duration, and hyperparameters, each of which has the potential to provide the agent with better learning feedback and adaptability in this environment.

Our analysis includes optimizations to encourage steady progress around the track, reduce penalty severity, and enhance the agent's control over its movement, creating a more balanced learning environment. These adjustments are anticipated to improve the agent's policy quality, yielding more consistent lap completions and fewer collisions.

2 Related Work

Previous research has shown that reward shaping is essential in RL applications for autonomous driving, particularly in guiding agents to prioritize efficient navigation and avoid collisions [1] . Studies in designing rewards for control tasks show that using specific rewards can help the agent make steady progress, stay on course, and balance penalties that discourage unsafe actions without stopping it from exploring. [1] .

New improvements in action settings highlight the importance of having more precise control over steering and speed, especially in situations that need accuracy. Researchers also point out that slowing down exploration helps the agent try out different options before focusing on certain strategies. [1] .

3 Our Results

Our modifications involved changes to four core components: the reward function, the action space, the exploration phase, and hyperparameters, all of which impacted the agent's learning and policy effectiveness.

3.1 Training Performance

Figure 1 shows the agent's reward progression over training episodes, highlighting the effects of the adjustments made.

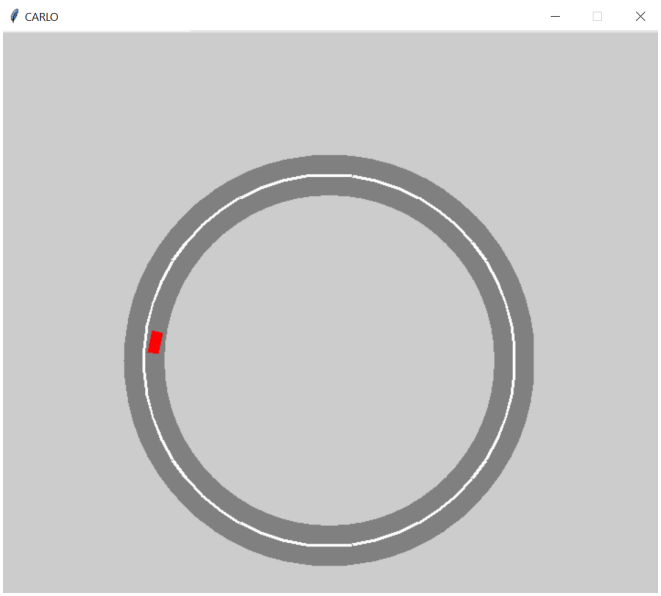


Figure 1: Training performance of the agent over episodes, showing cumulative rewards and learning stability.

3.2 Reward Function Optimization

The initial reward function imposed a large negative reward (-1000) upon collision, a small timestep penalty, and minimal positive feedback. These parameters limited the agent's exploration, leading to a high frequency of negative rewards and few lap completions.

Distance-Based Reward. To encourage the agent to maintain proximity to the track center, we added

*Department of Computer Science, University of Texas Rio Grande Valley, diana.castilleja01@utrgv.edu, alissen.moreno01@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

a distance-based reward. This additional reward provides consistent positive reinforcement for track alignment, which promotes smoother movement and helps the agent find the optimal position and trajectory.

Progress Reward. Forward progress around the circular track was incentivized by adding a reward for positive angular position changes. This reward motivated the agent to move steadily along the track, emphasizing forward movement rather than erratic or reverse behaviors, which were common in the initial setup.

Collision Penalty Adjustments. Initially, the large collision penalty (-1000) deterred exploration and penalized the agent heavily, often causing learning stagnation after a few collisions. Reducing this penalty to -500 allowed the agent to recover more easily and encouraged exploration, thus improving its resilience and willingness to experiment with different strategies.

3.3 Action Space Enhancement

The agent's action space was initially constrained, with limited options for steering and throttle control. This restricted the agent's ability to achieve smooth, precise control over its movement, which is critical in continuous navigation tasks.

Finer Steering Control. Adding intermediate steering angles (-0.2 and 0.2 radians) increased the granularity of the agent's turning capabilities. This adjustment enables smaller directional adjustments, leading to smoother navigation and reducing the need for abrupt turns, which can destabilize the agent's movement.

Additional Throttle Options. Similarly, an additional throttle level (0.05) offered finer control over speed adjustments. Previously, the agent's limited throttle range forced it to alternate between overly slow or overly fast speeds, reducing its control over acceleration. The new intermediate level allows the agent to navigate with smoother acceleration and deceleration, particularly around sharp turns and obstacles.

3.4 Extended Exploration Phase with Adjusted Epsilon Decay

In reinforcement learning, balancing exploration and exploitation is essential for effective policy learning. Our initial setup used a fast epsilon decay rate, limiting the agent's exploratory actions prematurely and leading to an incomplete policy. We slowed down the epsilon decay rate, extending the agent's exploration phase.

Slower Epsilon Decay. Increasing the epsilon decay period allowed the agent to maintain exploration for a longer duration, ensuring broader action space exploration. This slower transition to exploitation enabled the agent to develop a more comprehensive understanding of the environment before converging on specific strategies, reducing the risk of premature convergence on suboptimal actions and promoting better generalization across scenarios.

3.5 Hyperparameter Tuning for Stability and Performance

The learning rate and batch size are essential hyperparameters in reinforcement learning. Small changes to these values can have a significant impact on training stability and performance.

Learning Rate Adjustment. Lowering the learning rate from 0.001 to 0.0005 facilitated smaller, more stable updates to the Q-network. This adjustment is crucial in environments with sparse rewards, as it prevents sudden shifts in policy that may disrupt learning stability.

Increased Batch Size. We also considered increasing the batch size in experience replay to improve gradient stability. A larger batch size reduces variance in updates by averaging over more experiences, which stabilizes learning and allows the agent to make more informed adjustments to its policy.

4 Expected Impacts of Modifications on Agent Performance

The combined effect of these changes is expected to greatly improve how well the agent navigates the circular road. The updated reward system gives it more specific feedback on good behaviors, helping it complete more laps and avoid crashes. Adding finer control options allows for smoother, more precise navigation, and a longer exploration phase gives the agent more time to try different strategies.

Overall, these changes create an environment where the agent can balance speed and caution, both important for safely completing laps. By encouraging steady progress, fine-tuning its actions, and extending exploration, the agent is better prepared to learn a flexible, stable driving style that works well in different situations.

5 Conclusion

In summary, we made improvements to the reward function, action space, exploration phase, and hyperparameters to address the agent's initial challenges. These optimizations improved the agent's learning process, helping it make real progress in navigating the circular road environment. By carefully adjusting reward structures, control options, and exploration time, we created a strong base for further advancements in autonomous driving and showed how targeted changes in RL training can lead to better performance and stability.

However, due to time constraints and our beginner experience, there are still many details we need to fine-tune for the agent to consistently complete a full lap. With

more time and practice, we believe we can further refine these elements to help the agent navigate the road more reliably and safely when introducing other vehicles and/or obstacles.

References

[1] Z. Cao, E. Biyik, W. Z. Wang, A. Raventos, A. Gaidon, G. Rosman, and D. Sadigh. Reinforcement learning based control of imitative policies for near-accident driving. In *Proceedings of Robotics: Science and Systems (RSS)*, July 2020.

Human Conflict via Simulative LLM

Ramiro Santos * Mya Berlanga †

Abstract

This paper explores a dual-LLM system for interactive narrative games using Claude. The system employs two instances: a Response LLM that generates story-driven reactions to player actions, and a Verification LLM that validates these responses against established game rules. Through this architecture, the system maintains narrative coherence while enabling dynamic gameplay progression through verified state changes. Our findings demonstrate the effectiveness of LLMs in both activating game mechanics and ensuring their mechanical integrity through self-verification processes.

1 Introduction

In this paper, we present a narrative game system utilizing two instances of Claude, an advanced Large Language Model (LLM) developed by Anthropic. The system simulates academic interactions between students and professors, where conflicts arise from research-based scenarios that players must resolve through dialogue. The architecture employs two distinct LLM instances:

1. Response LLM (M_R): This primary model manages player interactions and simulates professor behaviors by generating contextually appropriate responses based on the current game state and player input.

2. Verification LLM (M_V): This secondary model evaluates responses from M_R and determines state transitions through a binary verification process. When verified, the system updates the game state, potentially resolving the current conflict and allowing progression.

The system enables dynamic storytelling while maintaining mechanical consistency through this dual-model approach, where successful conflict resolution leads to victory conditions and progression to subsequent challenges.

2 Enviroment

This experiment was conducted in a Unity environment, where C# was used to code the entire system. We utilized the model `claude-3-5-sonnet-20240620`, which has two responsive components that interact with the player. The game visually displays the text, and a state variable is used to determine whether the conflict can be ended and progress made, based on the verification status.

*Department of Computer Science, University of Texas Rio Grande Valley, ramiro.santos01@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, mya.berlanga01@utrgv.edu

The *Initial Prompt Responsive Generation* provides details such as the professor's name, research, hobbies, current position, academic background, and the specific conflict with the student (e.g., the student being late for a month or asking the professor to be their advisor for an interview).

Figure 1: Professors

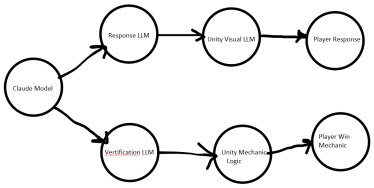


Figure 2: Diagram of the Logic

The *Initial Prompt Verification Generation* includes a set of criteria that the LLM must confirm. If all criteria are met, it returns a "yes" response; otherwise, it returns a "no". The model can only respond with either a "yes" or a "no" in this context.

3 Results

We conducted experiments on various conflict scenarios, including *Lying*, *Excuses*, *Interviews*, *Progression*, and *Plagiarism*. These conflicts were categorized by difficulty: *Lying* and *Excuses* were placed in the "easy" track, *Progression* and *Plagiarism* in the "medium" track, and *Interviews* in the "hardest" category due to their pass/fail nature.

In the easier conflicts (*Lying* and *Excuses*), the model could be easily resolved by admitting fault and attempting to redeem oneself. For *Excuses*, the model expects a clear and reasonable explanation for any absence. Our experiments showed that both the model's response and verification process were focused on ensuring that the player demonstrates the need to perform better than they have been, essentially seeking academic accountability. This is similar to how the *Lying* scenario is handled.

Plagiarism and *Progression* were categorized as medium difficulty because they involve persuasion: either defending your actions or demonstrating sufficient

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

progress in your work to convince the professor to allow you more time. Both scenarios can be resolved in two ways: by fully accepting the conditions or by arguing against them.

The *Interview* scenario was placed in the hardest category due to the complexity of the situation. Professors often conduct interviews based on their research background, and these may involve a series of questions that test the player's ability to gather information. By providing location-based hints related to the professor's work, such as pointing to specific research papers, the player must show they can access and apply relevant resources. This adds an additional layer of challenge to the interaction, testing the player's ability to navigate both academic and situational contexts.

4 Conclusion

In conclusion, this dual-LLM system successfully facilitates interactive, narrative-driven gameplay by using a Response LLM and a Verification LLM. The system maintains narrative coherence while allowing players to resolve conflicts in an academic setting. By categorizing conflict scenarios based on difficulty, we observe that the system can adapt to varying levels of complexity, from simple admissions of fault to more nuanced challenges requiring persuasion and critical thinking. Future work may involve refining the verification criteria and enhancing the system's responsiveness to dynamic player input.

References

How Hard can it be to Inscrypt a card game?

Jose Luis Castellanos * Pablo Santos †

Abstract

Inscryption is a 2021 rogue-like deck-building game developed by Daniel Mullins Games and published by Devolver Digital. With its depth of mechanics and charming art-style we not only show that determining whether there is a perfect win is NP-Complete, we also show how the game could be Turing Universal by including game mechanics introduced by the dedicated modding community.

1 Introduction

Inscryption is a 2021 rogue-like deck-building game developed by Daniel Mullins Games and published by Devolver Digital. The game is split into three different acts with distinctive game mechanics. However, in this paper we will turn out attention to the *Kaycee Mod*[3] which is unlocked upon completing the main game. In this mode, we can replay the game as a full rogue-like having access to game mechanics from all three acts as well as being able to fairly easily create custom cards to fit our needs. We make extensive use of the *Card Merge* feature and *Sigil* feature to show our reductions.

We show that finding if a *Perfect Win* (Win the match without dealing extra damage to the other player) is NP-Complete by reducing from Subset Sum in Section 3. We also show that with the inclusion of modded content, we are able to simulate a 3-Register Machine in Section 4.

2 Card Crafting

To understand how the game works we must first get some definitions out of the way.

Definition 1 *We define a Card as a 5-tuple* $C = (\mathcal{N}, \nu, \tau, \lambda, \Delta)$ *where*

- $\mathcal{N}$ *is a string with the name of the card.*
- $\nu \in \mathbb{Z}^+$ *which represents the sacrifice cost to play the card.*
- $\tau \in \mathbb{Z}^+$ *which represents the attack of the card.*
- $\lambda \in \mathbb{Z}^+$ *which represents the health of the card.*
- Δ *is a set of Sigils that affect how the card behaves.*

Definition 2 *Let* C' *be the set of all possible cards, we define the Hand as a 3-tuple* $H = (D, M, R)$ *where*

- D *is a multi-subset of* C'*, and* $|D| \geq 0$*.*
- $M \in \mathbb{Z}^+$ *is the current amount of sacrifices made.*
- $R \in \mathbb{Z}^+$ *is the number of bones the player currently has.*

Definition 3 *We define an empty Board* $B[x][y] = \varnothing$ *as a 2-dimensional Matrix of sizes* x *and* y*. Cards can be placed on the board such that* $B[x]'[y'] = C_{x'y'}$*, and only one card can be placed in each spot. Reserve* $B[x'][0]$ *for player cards, and* $B[x'][1]$ *for opponent cards.*

Definition 4 *To play a card means to take a Card from* H*, remove it from* H *and add it to* $B[x'][0]$ *such that* $x' \in \mathbb{Z}^+ \leq x$ *only if* $H_M \geq D_{x_\nu}$ *for the selected card* D_x*.*

Definition 5 *To destroy a card, means to select target card* $C_{x'y'} \in B[x'][y']$ *and remove it such that* $B[x'][y'] = \varnothing$*, we also increase* H_R *by 1.*

Definition 6 *When we sacrifice a card* $C_{x'y'}$*, we destroy it from* $B[x'][y']$ *and increase* H_M *by 1. This can only be done when a card is selected to be played that requires a sacrifice.*

Definition 7 *We define Sigils* Δ *as a subset of symbols* S *that give special properties to the card whenever it is played. The symbols used are the following:*

- *Stinky: Decrease the attack*(τ) *of the card in front by 1.*
- *Unkillable: Whenever this card is destroyed, return a copy to your hand.*
- *Enlarge: You can activate this card and spend 2 bones to increase its attack*(τ) *and defense*(λ) *by 1.*
- *Grotesque: Whenever this card would be attacked, it doesn't.*
- *Brittle: After this card attacks, destroy it.*
- *Terrain: This card cannot be sacrificed.*
- *Copy*[1]*: Whenever this card is played, you can select another card and copy its stats (Attack and Health).*

Definition 8 *A card attacking* $C_{x'y'}$ *always attacks the card above it* ($C_{x'y'+1}$) *such that* $C_{x'y'+1_\lambda} = C_{x'y'+1_\lambda} - C_{x'y'_\tau}$*. If the health of a card becomes 0, destroy it.*

*Department of Computer Science, University of Texas Rio Grande Valley, joseluis.castellanos01@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, pablo.santos01@utrgv.edu

[1] This mechanic is only available in a modded version of the game.

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

Definition 9 *At any given point, the player can pass the turn by hitting the bell in which all cards in play will attack, then it'll be the opponent's turn. In the case that either player does not have a viable move, all they can do is pass.*

Definition 10 *Finally, a Game is a 2-tuple* $G = (B[x][y], H)$ *where* B *is a board state and* H *is the players hand. The game ends when either player take a certain amount of damage.*

3 Winning Is Hard

Anyone who's ever played a card game, at some point they've wondered "Is there a way that I can win?". Depending on the game this can be a surprisingly hard question to answer, and for Inscryption it is NP-Complete.

Lemma 1 *We can create custom cards in a run of Inscryption.*

Proof. Showing this is fairly simple, as we are able to make use of the Mysterious Stones event in which we can permanently remove a card from our deck in order to transfer its sigil into another card. Since Kaycee's Mod is endless, we can achieve a stage where we can obtain enough cards with the sigils we want and merge them in order to obtain our desire card. □

Definition 11 *We define a Perfect Win as a win where we reduce the opponent's health to exactly 0 and not more.*

Theorem 2 *Achieving a Perfect Win in Inscryption is NP-Hard.*

Proof. We show a reduction from Subset Sum, a known NP-Complete Problem[2]. Let $P(S, T)$ be an instance of the subset sum problem where S is a multi-set and T is a target value. Create an instance of Game G with board $B[1][2]$. Create a hand H with cards C such that τ of card $C_i = S_i$, $\lambda = 1$, and $\Delta =$ Brittle. Let the opponent have health equal to T. Each turn we are able to play a single card on the board and pass in order to attack the opponent and slowly lower their health. Since the opponent holds no cards, all they can do is pass. We repeat this until we defeat the opponent, if we win and don't receive teeth(obtained from dealing extra damage) then there exists a valid solution for P. Thus we prove that if we can solve achieving a Perfect Win, we can solve Subset Sum. □

Theorem 3 *Achieving a Perfect Win in Inscryption is verifiable.*

Proof. Given a multi-set $S' \subseteq S$ as a solution for $P(S, T)$, create a Game G, Board B, and Hand H just as before in Theorem 2. Play each card in S' once per turn. Once we run out of cards and we if achieve a Perfect Win, then S' is a valid solution for $P(S, T)$. □

Corollary 4 *Achieving a Perfect Win in Inscryption is NP-Complete.*

Proof. Since we have shown Hardness and Membership, our problem of achieving a Perfect Win is NP-Complete. □

4 My Cards Can Count

Considering how complex and in-depth the game can be, one must wonder if there are more powerful things that the game could do. As a matter of fact it can! Due to the game being so beloved, there exists an extensive modding community for Inscryption[4]. Some of the mechanics added allow copying the stats of a card on command allowing the simulation of a 3-Register Machine to show Turing Universality[5].

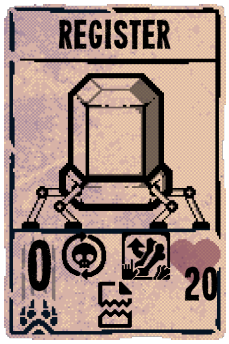


Figure 1: Example of a Register Card.

Definition 12 *A totem is a object where, given a head (defining type of card), and base (defining a sigil) it gives the chosen sigil to all cards of that type. This can be activated and deactivated at will at any moment.*

First, let G be a Game with board $B[4][2]$, Hand H with card C as shown on Figure 1[2] ($\Delta =$ Unkillable, Enlarge, Copy). Let $B[x'][1] = \{x'|x' \in \mathbb{Z}^+ \leq x-1\}$ be filled with Bear cards with , $\tau = 0$, λ being an arbitrarily large value, and $\Delta =$ Grotesque, Terrain, and $B[x'][0] = \{x'|x' \in \mathbb{Z}^+ \leq x-1\}$ be filled with the Register card as before. Also, let H_R be an arbitrarily large value.

Theorem 5 *Modded Inscryption can simulate a 3-Register Machine.*

[2]Custom cards were created using [1]

Proof. We have 3 cards each independent from each other and each one represents a register. In order to show that it can simulate a 3-register machine we must be able to perform the two following instructions:

- Increasing the count by 1.
- Decreasing the count by 1.
- If the count can't be decreased by 1, jump to the given instruction.

To increase the count by 1:

1. Use the card and spend 2 bones to increase the attack by 1.

To decrease the count by 1:

1. Activate the totem with Bear head and Stinky base.
2. Summon a register and copy the value of the chosen register to decrease.
3. Deactivate the totem previously used.
4. Summon a register and copy the value of the temporary register.

To jump to an instruction if the count can't be decreased:

1. If the card is unable to attack (attack is 0), then abort operation and skip to the specified instruction.

Since we have shown our game instance G can simulate a 3-Register machine, we have shown that Modded Inscryption is Turing Universal. □

5 Conclusion

Through clever use of in-game mechanics we have shown that winning in this card game can be deceivingly hard with the best approach being an NP-Complete solution in determining if you can win given a board state and deck. More so, we have proven that with extending the game to modded content we are able to show Turing Universality meaning that it's theoretically possible to compute things and even more theoretical to play Inscryption inside Inscryption!

References

[1] I. Community. Card creator. Accessed: 2024-11-10.

[2] D. o. C. S. Cornell University. Subset sum lecture notes, 2018. Accessed: 2024-11-10.

[3] Fandom. Inscryption wiki. Accessed: 2024-11-10.

[4] Thunderstore. Inscryption mods on thunderstore. Accessed: 2024-11-10.

[5] C. L. University of Cambridge. Computation theory lecture 2, 2016. Accessed: 2024-11-10.

Optimal Strategies in Tic-Tac-Toe Using Monte Carlo Tree Search

Robert Salazar * Joshua Hernandez-Noriega †

Abstract

Monte Carlo Tree Search (MCTS) is an effective algorithm for decision-making in game AI. This paper examines the use of MCTS to find optimal strategies in Tic-Tac-Toe, focusing on the effectiveness of different initial moves. Through 1,000 game simulations, we analyze the win rates for each possible initial move. Our findings show that certain starting moves significantly increase the AI's chances of winning, with a maximum win rate of 94.17% when starting in position 7 (bottom center). These results provide insights into MCTS's potential to guide strategic decisions in simple games.

1 Introduction

Monte Carlo Tree Search (MCTS) is a powerful algorithm for decision-making in game AI, capable of finding optimal moves through simulations. It has proven effective in complex games such as Go and Chess. In this study, we apply MCTS to Tic-Tac-Toe, a simple yet strategic game, to explore how different initial moves affect the AI's probability of winning. The primary objective is to determine the optimal starting move and sequence of moves that maximize the AI's chances of success.

We briefly highlight related work in Section 2, outline our methodology in Section 3, present the results in Section 4, and conclude with insights in Section 5.

2 Related Work

MCTS has been extensively studied in game theory and artificial intelligence. Prior work has demonstrated its effectiveness in games requiring complex decision trees, such as Go, Chess, and other board games. Research indicates that MCTS is particularly effective in games of perfect information where probabilistic simulations can approximate optimal strategies. This study contributes to the body of research by applying MCTS to a simpler game—Tic-Tac-Toe—to illustrate its effectiveness even in small-scale environments.

3 Methodology

To evaluate the effectiveness of different starting moves in Tic-Tac-Toe, we implemented an AI agent using Monte Carlo Tree Search (MCTS) in Python. The goal was to simulate 1,000 games where the AI, playing as 'O', attempts different initial moves to maximize its win rate. MCTS provides a probabilistic approach to strategy, balancing exploration of potential moves with exploitation of known favorable moves.

3.1 Monte Carlo Tree Search (MCTS) Implementation

MCTS operates through four key steps: *Selection*, *Expansion*, *Simulation*, and *Backpropagation*. **Selection**: The algorithm selects promising moves to explore, balancing high-potential moves with those needing more exploration (e.g., using UCB1) until it reaches an untested position.

Expansion: At an unexplored position, it randomly picks a new move, adds this position to memory, and continues.

Simulation: From this position, it plays out the game to get an estimate of the starting move's effectiveness, using either random or strategic moves.

Backpropagation: After each simulation, the algorithm updates the win count for moves in the game path, reinforcing successful moves for future decisions. The following code snippets illustrate the core components of our MCTS implementation.

3.1.1 Tic-Tac-Toe Board Setup

We created a Tic-Tac-Toe game board and initialized moves and states. The board setup function defines available moves and checks for a win state, as shown in the code snippet below:

```
class TicTacToe:
    def __init__(self):
        self.board = [' ' for _ in range(9)
            ]
        self.current_winner = None

    def available_moves(self):
        return [i for i, spot in enumerate(
            self.board) if spot == ' ']

    def make_move(self, square, letter):
        if self.board[square] == ' ':
            self.board[square] = letter
            if self.winner(square, letter):
                self.current_winner =
                    letter
            return True
        return False
```

Listing 1: Tic-Tac-Toe Board Setup

The `available_moves()` function returns open positions on the board, and `make_move()` updates the board with the current player's move.

*Department of Computer Science, University of Texas Rio Grande Valley, robert.salazar04@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, Joshua.hernandeznorieg01@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

3.1.2 MCTS Simulation and Move Evaluation

MCTS evaluates moves by simulating potential game outcomes for each possible action. For each simulation, MCTS chooses a random path to the end of the game, recording whether the move sequence results in a win for 'O'. The `simulate()` function below shows this process:

```
class MCTS:
    def __init__(self, game, iterations
        =100):
        self.game = game
        self.iterations = iterations

    def simulate(self, move, player):
        wins = 0
        for _ in range(self.iterations):
            game_copy = deepcopy(self.game)
            game_copy.make_move(move,
                player)
            if self.random_playout(
                game_copy, player):
                wins += 1
        return wins / self.iterations
```

Listing 2: MCTS Simulation and Move Evaluation

In the `simulate()` function, each move is evaluated by performing a number of random playouts (simulations) to estimate its likelihood of winning. The function uses deep copies of the game board to simulate independent move sequences for each trial.

3.1.3 Random Playouts and Move Selection

The `random_playout()` function completes each simulated game by choosing random moves for both players until the game reaches a terminal state (win, lose, or draw). The result is then used to update MCTS's strategy, as shown below:

```
    def random_playout(self, game_copy,
        player):
        turn = player
        while game_copy.empty_squares():
            available_moves = game_copy.
                available_moves()
            move = random.choice(
                available_moves)
            game_copy.make_move(move, turn)
            if game_copy.current_winner ==
                turn:
                return turn == player
            turn = 'O' if turn == 'X' else
                'X'
        return False
```

Listing 3: Random Playouts and Move Selection

After simulating each potential outcome, MCTS selects the move with the highest estimated win probability as the optimal choice for the AI agent. This iterative process allows MCTS to identify moves with the greatest likelihood of success based on simulated game outcomes.

3.2 Experiment Design

The experiment was conducted by simulating 1,000 games, each beginning with a different initial move by the AI. The AI selected the initial move with the highest calculated win rate, and this move sequence was stored as the optimal strategy.

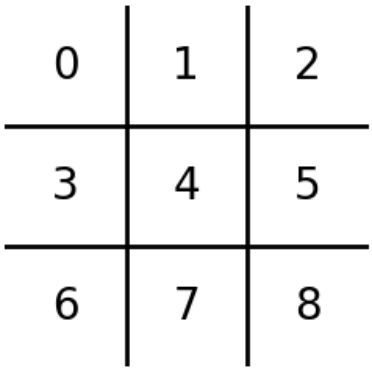


Figure 1: Tic-Tac-Toe board with labeled positions for initial moves.

4 Results

The win rates for each possible initial move are summarized in Table 1. Figure 1 provides a visual representation of the Tic-Tac-Toe board, with each position labeled according to its index number. In Table 1, each "Move" refers to the initial position chosen by the AI on the Tic-Tac-Toe board, as shown in the visual.

Move Position	Board Location	Win Rate (%)
Move 0	Top-left	85.58
Move 1	Top-center	83.16
Move 2	Top-right	88.68
Move 3	Center-left	83.93
Move 4	Center	86.73
Move 5	Center-right	88.99
Move 6	Bottom-left	92.97
Move 7	Bottom-center	94.17
Move 8	Bottom-right	92.92

Table 1: Win rates for each initial move in Tic-Tac-Toe.

4.1 Optimal Move Sequence

The optimal sequence of moves, starting with the best initial position (position 7), is outlined below. This path shows the AI's most effective strategy, capitalizing on advantageous board positions:

```
Move 1:          Move 2:          Move 3:
|   |   |   |    |   |   | X |    |   |   | X |
|   |   |   |    |   |   |   |    |   | O |   |
|   | O |   |    |   | O |   |    |   | O |   |
-------------    -------------    -------------
Move 4:          Move 5:
|   |   | X |    |   | O | X |
|   | O | X |    |   | O | X |
|   | O |   |    |   | O |   |
-------------    -------------
```

5 Conclusion

This study demonstrates that MCTS effectively identifies advantageous starting moves and winning strategies in Tic-Tac-Toe. The results indicate that starting in position 7 yields the highest probability of winning, with a success rate of 94.17%. However, the more iterations that were run, the longer the program took to compile, highlighting a trade-off in MCTS: finding a balance between speed and accuracy. MCTS thrives on tuning this balance to find optimal strategies quickly without exhaustive computation. This makes it especially valuable in scenarios where fast decision-making is crucial, even if absolute accuracy is sacrificed. Future research could explore scaling MCTS in more complex games with larger state spaces, where this balance becomes increasingly critical.

References

[1] AI and Games. Monte carlo tree search tutorial, 2024. Accessed: 2024-11-10.

[2] Jeff Bradberry. Introduction to monte carlo tree search, 2015. Accessed: 2024-11-10.

OrbitDrive: Reinforcement Learning for Circular Navigation

Hector Lugo * Arturo Meza Canales † Jorge Orta ‡

Abstract

This paper provides a comparative evaluation of three reinforcement learning algorithms—DQN, PPO, and Actor-Critic—to determine which performs best in navigating a Level 1 circular road environment. This paper presents the development of a reinforcement learning (RL) agent for autonomous navigation within a circular road environment. The environment, based on `circularroad.py`, consists of three levels of difficulty, with this study focusing on Level 1. The agent must learn to navigate safely from start to goal while avoiding collisions. Future work will involve scaling the agent's performance to Levels 2 and 3, which include dynamic obstacles and fine-tuning training methods. The aim is to create a robust RL solution that generalizes well across different scenarios.

1 Introduction

A major focus of this research is the comparative evaluation of three RL algorithms—DQN, PPO, and Actor-Critic—to determine which is most effective for this specific environment. Autonomous driving in controlled environments is a critical step towards achieving full autonomy in real-world scenarios. This research focuses on developing an RL agent capable of navigating a circular road environment. The project specifically targets Level 1, which involves basic trajectory maintenance, with Levels 2 and 3 reserved for future work to incorporate dynamic obstacles and increased complexity. Autonomous vehicle research benefits from such foundational studies to bridge the gap between simulated environments and practical applications [2, 3].

2 Problem Statement

The main objective is to evaluate the performance of three RL algorithms—DQN, PPO, and Actor-Critic—in training an agent to navigate a circular road under Level 1 conditions. Specifically, we aim to determine which algorithm achieves the best performance under the defined metrics. The main objective is to develop an RL algorithm that trains an agent to navigate a circular road under Level 1 conditions:

- **Level 1**: An empty road where the agent must maintain a safe trajectory and reach its goal.

Due to limited time, this study is restricted to Level 1. Future work will incorporate Levels 2 and 3, introducing dynamic obstacles and enhanced training strategies to improve agent robustness and performance.

3 Research Objectives

- **Comparative Evaluation**: Compare the performance of three RL algorithms—DQN, PPO, and Actor-Critic—in the Level 1 circular road environment to identify the most effective approach.
  - **Environment Understanding**: Analyze `circularroad.py` to understand the dynamics and constraints at Level 1.
  - **Algorithm Development**: Implement RL algorithms suitable for the environment (DQN, PPO, Actor-Critic).
  - **State and Action Space Design**: Define a state representation that captures essential information such as agent position, velocity, and heading.
  - **Reward Function Design**: Create a function that rewards goal-reaching efficiency while penalizing unsafe driving.
  - **Training and Evaluation**: Train the agent and evaluate performance using metrics such as success rate, average time to goal, and collision frequency.

4 Methodology

  - **Simulation Setup**: The `circularroad.py` environment was used as the base simulation. Modifications included enhancements for better agent feedback, allowing the model to gather relevant state information more effectively. Levels 2 and 3 were designed but not implemented due to time constraints.
  - **Algorithm Selection**: The study implemented three RL algorithms:
    * **DQN**: Utilized for discrete action control, featuring a neural network with three fully connected layers, experience replay, and target networks for training stability [3].
    * **PPO**: Chosen for continuous control, implemented using `stable-baselines3`, with policy gradient optimization and clipping to limit policy updates.

*Department of Computer Science, University of Texas Rio Grande Valley, `hector.lugo02@utrgv.edu`
†Department of Computer Science, University of Texas Rio Grande Valley, `arturo.mezacanales01@utrgv.edu`
‡Department of Computer Science, University of Texas Rio Grande Valley, `jorge.orta01@utrgv.edu`

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

  - **Actor-Critic**: Combined policy and value estimation using separate but interconnected networks, employing n-step rollouts and temporal difference learning for updates [2].

- **Implementation**: All models were built using PyTorch. The DQN architecture was optimized with experience replay, and PPO leveraged Generalized Advantage Estimation (GAE) for efficient policy updates. The Actor-Critic model featured distinct networks for policy evaluation and value estimation to balance training.
- **Training Strategy**: DQN training followed an epsilon-greedy strategy, while PPO utilized a clipped objective function to prevent large policy shifts. The Actor-Critic approach applied advantage-based learning to enhance stability.
- **Evaluation Metrics**: Success rate, average time to reach the goal, and collision frequency were key evaluation metrics. Training performance was visualized through plots of cumulative rewards and episode loss to highlight learning progress.

5 Results

DQN Performance: The DQN algorithm showed the most promising results for Level 1 training, achieving high cumulative rewards and stable learning curves. The award for this was around 50. The use of experience replay and target networks enhanced training consistency and convergence speed (see Figure 1).

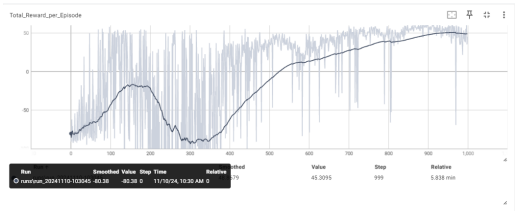


Figure 1: DQN Training Results: Total Reward per Episode.

PPO Performance: The PPO algorithm demonstrated a moderate learning curve, with slower improvement in mean rewards compared to DQN. The award for this was -57. PPO's clipped objective function provided a degree of stability but required more training epochs to achieve higher performance. (see Figure 2).

Actor-Critic Performance: The Actor-Critic model struggled with consistency, displaying significant reward fluctuations and slow convergence. The award for this was -350. While it effectively learned initial policies, maintaining stability across episodes was challenging due to high variance in updates. (see Figure 3).

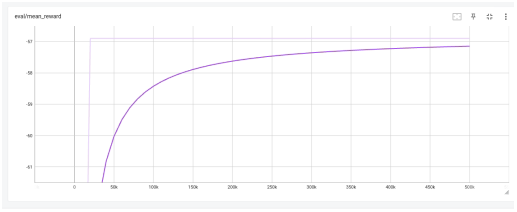


Figure 2: PPO Training Results: Mean Reward Over Time.

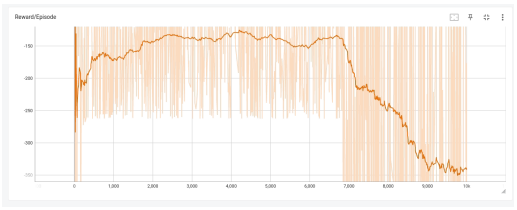


Figure 3: Actor-Critic Training Results: Reward per Episode.

6 Conclusion

This research successfully developed an RL agent capable of navigating a circular road environment at Level 1. Among the evaluated algorithms, DQN outperformed the others, demonstrating higher reward accumulation and faster convergence [3]. PPO showed moderate success, indicating its potential for complex control but requiring more extensive training. The Actor-Critic approach faced challenges with stability, highlighting the need for further refinement in training methods [2]. Future work will extend this study to include Levels 2 and 3, featuring dynamic obstacles and applying more sophisticated hyperparameter tuning and training strategies. These enhancements aim to improve the agent's generalization and robustness, crucial for transitioning RL solutions from simulation to real-world autonomous navigation.

References

[1] Cao, Zhangjie, Erdem Biyik, Woodrow Z. Wang, Allan Raventos, Adrien Gaidon, Guy Rosman, and Dorsa Sadigh. "Reinforcement Learning based Control of Imitative Policies for Near-Accident Driving." In *Proceedings of Robotics: Science and Systems (RSS)*, July 2020.

[2] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[3] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[4] Alexander Amini, Guy Rosman, Sertac Karaman, and Daniela Rus. Variational end-to-end navigation and localization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8958–8964. IEEE, 2019.

Leveraging Monte Carlo Tree Search: Enhancing Yavalath Gameplay with AI

Paul Hitchcox * Jaycee Sanchez †

Abstract

There have existed AI or computer opponents available to players in various games to help them practice against various levels of opponents. Traditional AI or computer opponents usually come in three variations of difficulty: easy, medium, and hard (though this is not always the case). With training and fine-tuning simulation time and depth of searching in decision trees, difficulties for AI agents can be much more precisely simulated for players to challenge themselves. Additionally, the ability to know all possible moves and predict the optimal path to win is usually seen in games with a limited number of decisions that a player can make and limited number of states that the game can have. However, some games like chess and Yavalath have substantial branching for possible moves in a Monte Carlo Tree, so it is only possible to look several moves ahead or down an arbitrary number of levels in a decision tree for chess and Yavalath, respectively. Therefore, Monte Carlo Tree Search and a strategy called UCB1 can be applied to predict best possible future moves using simple calculations that will serve as the basis for an AI agent. We propose that we can train the AI agent to learn the encoding for the Yavalath playable space, possible moves, and rules for winning and losing in python and will be able to use statistics generated by moves that result in wins or losses stored in a Monte Carlo Tree over a series of games to predict best moves for the AI agent to make to win the game.

1 Introduction

In this project, we utilized a variant of Monte Carlo Tree Search (MCTS) called Upper Confidence bound applied to Trees (UCT) in which statistical information about all possible moves in a two-player game are assigned empty values and new values are assigned as moves lead to wins over numerous runs of the simulations of the game. Using the UCB1 strategy, the upper bound of the statistical confidence intervals developed is chosen as the move with the upper bound fluctuates. These repetitive fluctuations over numerous simulations allow for confidence in best possible moves to ensure wins for players that utilize the AI agent. When different game possibilities are run, they become marked, and their statistical values such as times played and times won are updated.[1].

The four stages of UCB1 are selection, expansion, simulation, and back-propogation. Selection occurs when moves are made based on existing data stored in the program. Expansion occurs when moves are made randomly due to there being no valid marked paths to visit (marked nodes are nodes in which the move choices have been applied already). Simulation occurs when a move has been made and the playout of the game is done to completion by some criteria set by the programmer (or randomly). Back-propagation occurs when a playout or simulation ends so that the win count is updated if the player wins and all nodes visited along the path that terminated have their play counts incremented as well. This knowledge is necessary to understand other terminology found later in the paper and fully understand the mechanics behind Monte Carlo Tree Search and its application to AI training for games based on paths for choosing moves. Our game in particular, Yavalath, is usually played with two players where the color assigned to each player is either white or black. All initial spaces in the playing space are not marked, and the players alternate turns and mark free spaces with their respective colors. A player wins when they get four spaces in a line connected, but they lose if they get three spaces in a row. Due to the number of possible outcomes and slowly decreasing playing space, the resulting Monte Carlo tree gets very wide as every player makes a turn.

$$\bar{x}_i \pm \sqrt{\frac{2 \ln n}{n_i}}$$

Figure 1: Upper Confidence Bound (UCB1) formula used in Monte Carlo Tree Search, where $\bar{x}_i$ is the mean payout of move i, n_i is the number of simulations run for move i, and n is the total number of simulations run while the program executed.

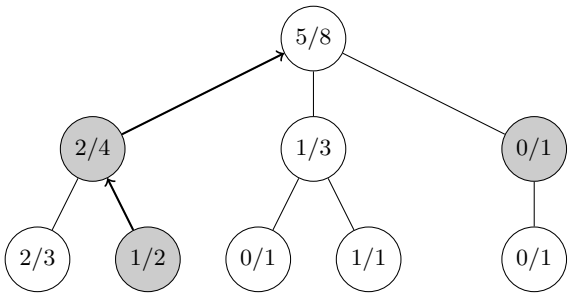


Figure 2: An example Monte Carlo Tree after some playouts during back-propagation.

*Department of Computer Science, University of Texas Rio Grande Valley, paul.hitchcox01@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, jaycee.sanchez01@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

```
Best move: (5, 7)
Total simulations run: 1500
Maximum depth reached in any simulation: 12
Move statistics:
Move (5, 7): 63.27% (950 / 1500)
Move (4, 6): 47.15% (707 / 1500)
Move (6, 8): 52.13% (782 / 1500)
Move (7, 7): 35.50% (533 / 1500)
Move (3, 5): 45.90% (688 / 1500)
Move (8, 6): 50.47% (757 / 1500)
Move (9, 5): 42.80% (642 / 1500)
Move (5, 6): 55.13% (827 / 1500)
Move (6, 5): 48.40% (726 / 1500)
```

Figure 3: Sample expected output for python code when running simulations on Monte Carlo Trees for Yavalath

2 Representing Game Boards with Python

Relying on the code provided for the Monte Carlo Tree search from Bradberry's article [1], the project was set up fairly quickly to get a code base running. The main technical part of this problem was then using python code to represent various aspects of the game in order for the Monte Carlo search class to take in the data when its member functions were called and then output reliable data about the efficiency of certain moves at differing lengths of simulation runs. To do this, we had member functions in class Board for init (sets up most basic game parameters), start (initializes a new game), is-terminal (checks whether someone has won or lost), check-winner (checks if a player has made 4 in a row), check-loser (checks if a play had made 3 in a row), get-available-spaces (returns all empty spaces on board), and next-state (updates turn, winner, and board positions for next turn). The majority of the time spent on this problem was spent on debugging the python code for the board configuration. The code was stored in Jupyter Notebook, and the closest attempt at a working solution was wiped after the page was inactive for a few minutes. Despite this, various observations were drawn from the problem and the attempted solution. With the closest attempt, the python code failed to terminate with a nice report similar to Figure 2, but with debugging statements, it was possible to see which moves resulted in wins and losses contributing to stats for the individual Cartesian coordinate pairs on the board.

3 Our Results

The results of the exploration and attempted solution of the problem were mostly observational. As seen in Bradberry's article[1], it is possible to represent a game board with python and use Monte Carlo Tree Search and strategies such as UCB1 to train AI agents to more efficiently play games of certain complexity classes. Since Yavalath falls into possible games for representation, great progress and a nearly-complete solution was achieved to track best moves to win the game. One main takeaway was that increasing the time that the program runs increases the overall number of simulated games run and, therefore, provides more accurate data than on a smaller dataset of simulated games. Additionally, it was found that there are potential disadvantages to the UCB1 strategy such as there potentially being no ideal moves or moves with very similar percentages of winning simulated games. Additionally, various debugging methods were discovered during the development process to measure the depth of the tree that it was reaching, program execution time, which steps were individually resulting in wins and losses even when the program terminated with some error, etc. These tools will streamline development in the future and make it much more efficient.

4 Future Work

After using limited time during the hackathon and seeing that developing an AI agent in python is possible for games, the goal is to complete the AI agent for Yavalath and develop more AI agents for a wider range of games. In addition, fine-tuning the length of program execution, optimal number of simulations, and adjusting these as the complexity of the games grow are all areas that could be expanded upon in further work. Additional work also involves developing a naive approach using AI and comparing its performance with the performance on the AI agent with no training and with various levels of training to understand the fine-tuning requirements for difficulties that we are interested in.

5 Conclusion

The progress that was made on the problem of developing an AI agent to play Yavalath well was not fully solved; however, many key observations were made about the Monte Carlo Tree Search process and how to develop it fully given more time. This result is encouraging for developing similar AI agents for other two-player games with decreasing sizes of playing space. Additionally, it provides insight into potentially automating the production of board configurations for various two-player games with python code given a set of rules and game format. Although we were unable to definitively accomplish the goal of the problem, major strides were made, and motivations were stirred to continue working on game complexity problems.

References

[1] J. Bradberry. Introduction to monte carlo tree search, 2015.

Ramsey Theory Analysis of Void Chemical Reaction Networks

Divya Bajaj * David Barreda † Aiden Massie ‡

Abstract

We analyze the Chemical Reaction Network (CRN) model under the branch of Ramsey theory combinatorics. Specifically, we focus on the following property: given a CRN that can only delete two unique copies of species, what is the smallest amount of distinct species n needed such that any reachable configuration in the system is guaranteed to have at least n species that share the same number of copies?

We first prove that, for $n > 2$, n^2 unique species are required to satisfy the problem. We have constructed a brute-force algorithm that also solves the question with n^2 for $n > 2$, verifying that our mathematical proof provides the optimal answer.

1 Introduction

Chemical Reaction Networks are well-studied models of chemistry in which a configuration of *species* of different counts is transformed and changed by a set of *rules*. Designed to abstract the real-world process of chemical interactions, CRNs have enjoyed an extensive research history and have been shown to be equivalent to important models in other fields such as Vector Addition Systems and Population Protocols.

Definition 1 (Reachability) *Reachability is concerned with the existence of a sequence of reactions such that the initial configuration can be turned in (or reach) to the target configuration.*

Definition 2 (Configuration) *A configuration refers to the amount of each species currently in the system. An **initial configuration** refers to how much of each species the system starts with.*

CRN studies have emphasized natural problems with the model's capabilities, such as reachability (given two configurations s and t, can s transform into t?) or coverability (given two configurations s and t, can s reach some configuration c s.t. all species counts in c are at least that of t's?). However, questions regarding the properties of the model themselves aren't as widely pursued.

We thus focus on the following question, inspired by the Ramsey theory field of combinatorics: given a CRN whose only rules are those that delete two *unique* species copies, what is the minimum number of species n required s.t. any reachable configuration in the CRN is guaranteed to have at least n species with the same amount of copies?

In Section 2, we prove that n^2 species is required to satisfy the problem for all non-trivial cases $n > 2$. A brute-force algorithm is also provided that matches the answers in the proof. We then conclude our results in Section 3.

2 Our Results

Definition 3 (Ramsey Pattern) *A CRN system has the **Ramsey pattern** if its current configuration and all other configurations reachable from it have at least n unique species that share the same amount of copies.*

Lemma 1 *We are given a (2,0)-size void CRN with rules of form $A + B \longrightarrow \phi$ and a starting configuration with n unique species of n copies each when $n > 2$. We can then reach a target configuration with exactly $n^2 - 1$ unique species s.t. $\forall j \in [0...n]$ we have $n-1$ unique species with j copies.*

Proof. Given n^2-1 unique species, we can always divide these species evenly into $n+1$ categories with $n-1$ unique species. Thus, $\forall j \in [0...n]$ we can have $n-1$ unique species with exactly j copies.

Now, we prove that configuration is reachable. Here, we look at two possible cases.

Let n be an odd number; then n^2-1 and $n-1$ are even numbers. If we have n^2-1 of unique species, then we can pair species evenly where for each pair (A, B), we can use the reaction $A + B \longrightarrow \phi$ to reduce the count of both species by 1 for every rule application. We can apply reactions for all such pairs, leading in $n-1$ species with count $j \forall j \in [0...n]$.

If n is an even number, then n^2-1 and $n-1$ are odd numbers. If we have n^2-1 of unique species, then we can pair species in a way so we have $n-2$ species with count $j \forall j \in [0...n]$. Then we are left with $n^2-1-((n-2)*(n+1))$ unique species which is equal to $n+1$ unique species. We don't need to reduce the count of 1 of the species as then we will have $n-2+1$ unique species with count of n. For the rest of our n unique species, which is an even count, we can recursively reduce them so we have 1 species each with count $j \forall j \in [0...n-1]$. We can then have $n-2+1$ unique species with count $j \forall j \in [0...n]$. □

Theorem 1 *Given a (2,0) void CRN with rules of form $A + B \longrightarrow \phi$ and a start configuration with n unique species of n copies each, the minimum number of unique species required to reach a configuration with at least n unique species with equal number of copies, such that any further reactions do not break the pattern, is n^2 for $n > 2$. The number is equal to n when n is $n \leq 2$.*

*Department of Computer Science, University of Texas Rio Grande Valley, divya.bajaj@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, david.barreda01@utrgv.edu

‡Department of Computer Science, University of Texas Rio Grande Valley, aiden.massie01@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

Proof. For $n = 1$ the proof is trivial by definition. Every configuration has at least one or more species with equal number of copies.

For $n = 2$, say you start with the configuration: $\langle 2A, 2B\rangle$. There is no need to add any new species, as the only possible (2,0) void rule will affect both species, resulting in them having equal count, maintaining the Ramsey pattern where at least 2 species will both have $0, 1 or 2$ copies each.

For $n > 2$, let the minimum number of species required to satisfy the Ramsey pattern be m. That is, if any further reactions occur, the pattern is not broken. Say the configuration with m unique species contain s_0 species with 0 copies, s_1 species with 1 copies, and so on, ending with s_n species with n copies. Then the following will be true:

$$m = s_0 + s_1 + ... + s_n$$

For any $m \leq n^2 - 1$ we can always reach a configuration where $\forall j \in [0...n], s_j <= n - 1$. This is because for $m = n^2 - 1$, we can have a reachable configuration with m unique species divided into $n+1$ categories with $n-1$ unique species where $\forall j \in [0...n], s_j = n - 1$. This configuration is reachable for any $n > 2$ as stated in Lemma 1.

However, when we add another species with n copies into the system, we make $m = n^2$. This will make it so that any further reactions do not allow a configuration where less than n species are equal. This would also increment the number of species s_l by 1 for any l less than n. Because $\forall j \in [0...n], s_j = n-1$, then this would always result in the Ramsey pattern. Therefore, the minimum number of species required so the pattern never breaks is n^2.

□

2.1 Brute Force Algorithm

We implement a brute-force algorithm to solve the Ramsey CRN problem and verify our hypothesis.

A formal description of the algorithm is as follows: Given an integer n, we first check if $n \leq 2$. If so, we only return n; otherwise we create a vector v of size n with each integer in v set to n. Note that v is a vector representation of our CRN. Each species s_i in the CRN is represented by the i-th index in v, with the value at i representing s_i's count. For example, a CRN with a configuration $\{A : 3, B : 3, C : 3\}$ is transformed into $[3, 3, 3]$.

We then input v into a new function that determines if all reachable configurations from v satisfy the Ramsey pattern. It works as follows: we first see if v satisfies the Ramsey pattern. If the check is satisfied, we then check if the valid configurations reachable from v also satisfy the pattern. If the check is again satisfied, we then continue recursively searching through every possible reachable configuration. Note that we can lower the running time here by observing that certain configurations are the same when ignoring order (i.e. $[2, 2, 3]$ and $[3, 2, 2]$) *and* break the Ramsey pattern. Therefore, when given a new configuration we perform a *reverse* sort on it and check if it has been observed before. If so, we simply ignore it, else we mark it down to be analyzed later.

If the function shows that v and all reachable configurations from v satisfy the Ramsey pattern, we return the length of v. If not, we perform the following operation: We first perform a reverse sort on v and check the number of indices m of value n present in v. We then append $n - m$ indices of n into v, re-satisfying the Ramsey pattern. We then input the new modified vector v' into the function. If v' and its reachable configurations is shown to satisfy the Ramsey pattern, we return v''s length; else we repeat the same operation on v' and so on. [1]

Testings on the brute force algorithm have shown that the algorithm provides the same answer as the mathematical proof for $n = \{0, 1, 2, 3, 4, 5\}$

3 Conclusion

We have thus shown that Ramsey's pattern can be observed in all configurations of a CRN if you have at least n^2 unique species, for some $n > 2$. For $n \leq 2$ where $n \in Z^+$, the minimum number of unique species is equal to n. We provided an algorithm to show this consistency with our calculation, proving that our formula for calculating this number is optimal.

References

[1] A. Massie. Ramsey crn brute force algorithm. https://github.com/aidenmassie/Ramsey-CRN-Solution, Accessed: 2024-11-10.

Reachability of (1,1) Conditional iCRN is NP-Complete and Equivalency over alternative CRN's under Void Rules

Jose Luis Castellanos * Pablo Santos †

Abstract

Chemical Reaction Networks have been studied for some time now, with many augmentations being created over the years. For this paper, we explore the iCRN model under (1,1) rules and show that its reachability problem is NP-Complete. We also look into Void Genesis, PVAS, and Single Inhibitor iCRN models under void rules and show their equivalency.

1 Introduction

Chemical Reaction Networks (CRNs) are a well-established model of chemistry, where chemical interactions are modeled as molecular species that react to create products according to a set of reaction rules. They have been studied since the 60's due to their complex computational power and have been proven to be Turing Decidable, while more complex models like iCRN are Turing equivalent through their simulation of register machines. Reachability for these models is another point of interest where we compute if a certain configuration is reachable through valid instructions from a starting configuration. This has been studied for many CRN models and has even been shown to be NP-Complete for iCRNs under (2,0) rules. In this paper, we continue this trend of finding the limitations of reachability by looking at (1,1) rules for iCRNs, proving its NP-Completeness as well as finding equivalency between other models similar to iCRN under void rules.

2 Reachability for (1,1) iCRNs is NP-Complete

For this proof, we use a simple reduction from the Hamiltonian Path problem.

Proof: Given any 3-sat formula $f = c_1 + c_2 + ... c_n$ create instance of iCRN with the following alphabet and ruleset.

Set notation:
$V = \{v \mid \text{v is a variable in at least one of the clauses}\}$
$T = \{T_x \mid x \in V\}$
$F = \{F_x \mid x \in V\}$
$D = \{d_x \mid x \in V\}$
$F = \{F_x \mid x \in V\}$
$C = \{C_i \mid C_i \text{ is a clause in the 3-SAT formula}\}$
$C_x = \{C_i \in C \mid C_i \text{ is true when } x \text{ is used and } x \in V\}$
$C_{\overline{x}} = \{C_i \in C \mid C_i \text{ is true when } \overline{x} \text{ is used and } x \in V\}$
$\Lambda = T \cup F \cup C \cup D \cup d$
Ruleset R :

$T_x \xrightarrow{d_x} d_x \quad | \quad x \in V$
$F_x \xrightarrow{d_x} d_x \quad | \quad x \in V$
$\{C_i \xrightarrow{C_x} d \quad | \quad x \in V \wedge C_i \in C_{\overline{x}}\}$

The starting configuartion would be one copy of each species in C, one copy of each species in T and F.

Given the rules and starting configuration of this iCRN, the reachability of configuration v, where you have 0 copies of each element in T and F, $|C|$ copies of d and two copies of each element in D. Because of the way this is made the reachability of this iCRN is equivalent to solving the original 3-SAT problem, making reachability NP-Hard.

Because all rules in this iCRN produce d and no rules have d as a reactant, the reachability for this problem can be verified in polynomial time, and therefore, the problem is NP-Complete.

3 Equivalency between PVAS, Single Inhibitor iCRN, and Single Counter Void Genesis (2,0) void rules

While these models are not equivalent in non-restricted rule sets, they share a certain equivalency under the (2,0) rule set.

Lemma 1 *If (2,0) Single Counter Void Genesis reachability is NP-Complete, then (2,0) PVAS is NP-Complete if species i = 1 always reduces to 0 first .*

Lemma 2 *If (2,0) Single Counter Void Genesis reachability is NP-Complete, then (2,0) Single Inhibitor iCRN is NP-Complete if single inhibitor always reduces to zero first.*

Proof. For any (2,0) Single Counter Void Genesis, there is some species which reaches zero first and is responsible for the value of z used throughout. Because we are working with (2,0) rules, this species can never become positive once it's reached zero, and z can only be greater than 1 after detecting zero. This implies the ability to simulate this void CRN with a Single Inhibitor CRN, where every Void Genesis rule $A + z \to \emptyset$ is replaced with $A \to \emptyset$ where X is one of the species in the Void Genesis CRN. □

Lemma 3 *If (2,0) Single Inhibitor reachability is NP-Complete, then (2,0) PVAS is NP-Complete.*

Proof. For any (2,0) Single Inhibitor CRN, you can simulate it with a PVAS model where every rule of the form $A + B \to \emptyset$ can be replaced with $A + B \to \emptyset$, where the configuration has I be the first species count. □

*Department of Computer Science, University of Texas Rio Grande Valley, joseluis.castellanos01@utrgv.edu
†Department of Computer Science, University of Texas Rio Grande Valley, pablo.santos01@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

4 Conclusion

References

The Reachability Problem in (2,0) Void iCRNs with single inhibitor species is polynomial-time solvable

Divya Bajaj * David Barreda † Ryan Knobel ‡

Abstract

An Inhibitory CRN is a variation of CRN with reactions that are inhibited by certain species. The reachability problem of (2,0) void iCRNs, (2,1) void iCRNs and (2,0),(2,1) void iCRNs is proven to be NP-hard. In this paper, we address the reachability problem in (2,0) void iCRNs with a single unique inhibitor species. We show that the reachability problem in this particular model is solvable in polynomial time, via a reduction to the maximum b-matching problem. Our results contrast with the general NP-hardness of the reachability problem in other types of iCRNs.

1 Introduction

A *Chemical Reaction Network (CRN)* is a model of molecular computation. While the model itself is highly studied, much work is yet to be done across other practical variations. One such model variation is the *Inhibitory Chemical Reaction Network (iCRN)* model [1], which is the focus for this paper.

Definition 1 (Inhibitory CRN) *An iCRN $\zeta = (S, R)$ is a set of species S and rules R. These rules dictate how species interact with each other. Also, each rule may also have **inhibitors**, which are species that if present, prevent the rule from running.*

Definition 2 (configuration) *A configuration refers to the amount of each species currently in the system. An **initial configuration** refers to how much of each species the system starts with.*

Definition 3 (The Reachability Problem) *Given an iCRN with species S and rules R, an initial configuration C_a, and target configuration C_b, does there exist a sequence of reactions such that the initial configuration can be turned into the target configuration (yes or no)?*

A (2,0) void iCRN with single inhibitor reactions with n unique inhibitor species, can be represented as a set of rules of the form:

independent reactions:

$$A + B \longrightarrow \phi$$

reactions using inhibitors:

$$A + I_x \longrightarrow \phi$$

inhibited reactions:

$$A + C \xrightarrow{I_y} \phi$$

We show that the reachability problem in (2,0) void iCRNs with the restriction of single type of inhibitor species is polynomial-time solvable. We briefly highlight some related work in Section 2, and then provide the definitions and results of our work in Section 3. We then conclude in Section 5 and point towards the general research goals for this work.

2 Related Work

It has been proven that the reachability problem of (2,0) void iCRNs is NP-hard with a polynomial number of unique inhibitor species. Another interesting result is that the reachability problem of (2,0),(2,1) void iCRNs is NP-hard. T. Wylie reduced Vertex cover, an NP-complete problem, into an instance of the reachability problem of (2,0),(2,1) void iCRNs.

3 Our Primary Results

For our primary result, we rely on the b-matching problem, a generalization of the matching problem. It takes the form of a traditional matching when all b-values are 1 and an uncapacitated b-matching occurs when all edge capacities are assigned u(e) = ∞.

Definition 4 *(b-matching). Given a graph $G = (V,E)$ and some edge capacity function $u : E \to \mathbb{N} \cup \{\infty\}$ and a b-value function $b : V \to \mathbb{N}$, find a maximum assignment $f : E \to \mathbb{N}$ s.t. $f(e) \leq u(e) \forall e \in E$ and $\sum_{e \in \delta(v)} f(e) \leq b(v) \forall v \in V$. We call it a perfect b-matching if $\sum_{e \in \delta(v)} f(e) = b(v) \forall v \in V$.*

Theorem 1 *Reachability for (2,0) void iCRN with binary encoded species and only one inhibitor species is solvable in polynomial-time.*

Proof. For any start configuration C_a and target configuration C_b, C_b is reachable from C_a if the empty configuration C_0 is reachable from $C_b - C_a$. This is because it is a void iCRN that only deletes species.

*Department of Computer Science, University of Texas Rio Grande Valley, divya.bajaj@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, david.barreda01@utrgv.edu

‡Department of Computer Science, University of Texas Rio Grande Valley, ryan.knobel01@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

Given any target configuration C_b, $C_b[I]$ is the count of inhibitor species I in the target configuration. We need to look at two possibilities.

In the case where $C_b[I] \neq 0$, i.e. there are a positive number of copies of the inhibitor species left in C_b, the inhibited reactions will not run. We can reduce this instance of reachability of (2,0) void single inhibitor iCRN with R rules to a reachability problem of (2,0) void CRN with R' rules where $R' = R \setminus \{r_k \text{ where } r_k \text{ is of the form } s_i + s_j \xrightarrow{I} \phi\}$. ASARG [2] shows that we can reduce the reachability problem of (2,0) void CRN to a maximum b-matching algorithm which is solvable in polynomial-time. There exists a valid sequence in (2,0) void iCRN where $C_b[I] \neq 0$ iff there exists a valid sequence in (2,0) void CRN as constructed above. This is correct by construction. In the forward direction, if there exists a valid sequence in (2,0) void CRN, we also have a valid sequence in (2,0) void iCRN because the iCRN only has a few extra inhibition rules that won't be in the sequence. For the other direction, this is true because the inhibitor count is > 0, therefore, the inhibition rules in iCRN will never run, and therefore need not be included in the void CRN.

For the case where $C_b[I] = 0$, i.e. there are zero copies of inhibitor species left in the target configuration, we can reduce this to an instance of void CRN with no inhibition. For any reaction of form $A + C \xrightarrow{I} \phi$ in the (2,0) void iCRN, we substitute a rule of the form $A+C \longrightarrow \phi$ in the (2,0) void CRN. We know that the reachability problem in (2,0) void CRN is polynomial time solvable.

If there exists a valid sequence in (2,0) void iCRN, there will always be a valid sequence in the (2,0) void CRN. This is because a (2,0) void CRN is a more relaxed version with no inhibition, and if a target configuration is reachable with inhibition, it can also be reached without inhibition by running rules that use inhibitors before inhibited rules. A valid sequence of a (2,0) void CRN might not always be a valid sequence of a (2,0) void iCRN. This is because a valid sequence of form $R_1, R_2, ...$ could have a reaction R_i and R_j s.t. $i < j$ where R_i is an inhibited reaction in the iCRN model and R_j is a reaction with the inhibitor as a reactant. We argue that there is always a possible rearrangement of the reaction sequence such that a valid sequence in (2,0) void CRN can be transformed into a valid sequence in (2,0) void iCRN. This is possible because the void CRNs only have deletion rules, and because we have a fixed count of $C_a[i] - C_b[i]$ for any species i which is reachable. We can rearrange the subsequence R_j, R_k to R_k, R_j where each reaction has species i as a reactant. This is because only a fixed number of species will be used up by each reaction, and if we can reach the empty configuration from a start configuration with sequence $..., R_i, ..., R_j, ...$, then we can also reach the target configuration with sequence $..., R_j, ..., R_i, ...$ With void rules, no reaction generates a new species. Hence no reaction execution will ever enable another reaction. Therefore, if we have a valid sequence of a (2,0) void CRN, we can rearrange it into a valid sequence of (2,0) void iCRN with single inhibitor species by moving all inhibition reactions after the last occurrence of reactions that use the inhibitor.

From the above constructions, given a (2,0) void iCRN with single inhibitor species, we can always reduce the reachability problem in this model into an instance of the reachability problem in a version of (2,0) void CRN which is polynomial time solvable. □

Corollary 1 *Reachability for (2,0) PVAS with single priority is solvable in polynomial time.*

The above corollary is apparent because Priority Vector Addition Systems (PVAS) can be simulated by single inhibitor species iCRN, when they are restricted to one priority ($k = 1$).

4 Further Results

Following the same logic as in (2,0) void iCRN with only one inhibitor species, we can extend the idea of using b-matching to apply it to (2,1) void iCRN with the same contingency of only one inhibitor species. This implies the lemma that follows.

Lemma 1 *Reachability for (2,1) void iCRN with binary encoded species and only one inhibitor species where $C_b[I] \neq 0$ is in NC.*

Proof. The reduction is similar to 1, but applied to (2,1) void iCRN with single inhibitor species type. For $C_b[I] \neq 0$ we reduce this problem to the reachability problem of (2,1) void CRNs with reduced rule set with no inhibition. ASARG [2] shows that this instance of the problem is in NC. There exists a valid sequence for reachability problem of (2,1) void iCRNs iff there exists a valid sequence for the reachability problem of (2,0) void CRNs. This is true by construction. □

5 Conclusion

We have thus shown that the reachability problem in (2,0) void iCRNs with a single inhibitor species can be solved in polynomial time by reducing the problem to the reachability problem in corresponding void CRNs. We still do not know about the complexity of the problem with constant number of unique inhibitor species.

References

[1] K. Calabrese and D. Doty. Rate-independent continuous inhibitory chemical reaction networks are turing-universal. In D.-J. Cho and J. Kim, editors, *Unconventional Computation and Natural Computation*, pages 104–118, Cham, 2024. Springer Nature Switzerland.

[2] A. Lab. Step-void crns.

Legend of Zelda: Echoes of Wisdom

Ryan Knobel * Gaukhar Nurbek † Juan Manuel Perez ‡

Abstract

This paper shows NP-hardness for dungeon traversal using 3 of the new mechanics found in Legend of Zelda: Echoes of Wisdom.

1 Introduction

The Legend of Zelda, first introduced in the 1980's, quickly rose to the forefront of the video game industry. In combination with this rise to fame, the introduction of new mechanics and its puzzle solving nature lead to several interesting theoretical questions. Among these is the complexity of dungeon traversal: Given some dungeon, a starting point and an ending point, can Link/Zelda make it from the start to finish?

[2] extensively covers this question across all Zelda games up until the year 2017. Our work focuses on extending [2] to include the 3 new mechanics from the latest Zelda game, showing NP-hardness for dungeon traversal.

This paper is organized as follows. We start by providing definitions and background information in Section 2. In Section 3, we detail the gadgets and the polynomial time reduction. We then conclude in Section 4.

2 Definitions

Following the convention of [2], we refer to **Generalized 3D Zelda** as the abstraction of rooms into three-dimensional spaces bounded by solid polygons with fixed-point coordinates, as well as dynamic objects with polygon-defined collision boundaries. We refer to this room abstraction as a **dungeon**. Zelda and other dynamic objects cannot pass through solid polygons, and are pulled downwards by gravity, which means that they will fall if they are not on a polygonal surface. Crystals are an exception, as they are not affected by gravity.

In Echoes of Wisdom, Zelda has 3 new abilities. Her **Swordfighter form** enables her to use a sword for the duration of her energy. Once this energy is fully consumed, her swordfighter form is disabled. Energy can be reattained through *energy crystals*, which appear in dungeons and can also be dropped by enemies when they get killed. Zelda has the ability to **bind/unbind** to certain objects, such as moveable platforms. Once binded, Zelda follows the movement of the object (or vise versa) until unbinded. This is possible through her **Tri Rod**, which also allows her to spawn up to 3 **echoes** at a time. These echoes can be objects or captured enemies.

*Department of Computer Science, University of Texas Rio Grande Valley, ryan.knobel01@utrgv.edu

†Department of Physics, University of Texas Rio Grande Valley, gaukhar.nurbek01@utrgv.edu

‡Department of Computer Science, University of Texas Rio Grande Valley, juan.m.perez02@utrgv.edu

Figure 1: Starting gadget.

Figure 2: Checking gadget.

The **Dungeon Traversal** problem asks the following: given a starting location, an ending location and a dungeon, can Zelda traverse from the start to the end without dying?

3 Our Results

This section uses a reduction from 3SAT to show NP-hardness for the 3 new mechanics described in Section 2. The reduction follows a similar framework to that of [1], first providing assignments for each variable and then ensuring each clause is visited with a checking gadget.

We start by first describing the gadgets used, followed by the result. For this reduction, we abstract Zelda's movements to the following: she can jump 0.5 units high and 1 unit long, can place new echos up to 1 unit high (0.5 units higher than the jump distance) and has 2 hearts. We also assume that defeating an enemy in her swordfighting form consumes 1 energy crystal.

Starting Gadget. Directly to the right of Zelda's starting location is a 1 unit drop into a hole 2 units long, with the opposing wall being 3 units high. On top of this wall is the exit for the dungeon. As Zelda can place at most 3 echos, there is no way for Zelda to directly scale the wall. Instead, Zelda must place 3 echos into a column 3 units high that is 1 unit away from the opposite wall. This prevents Zelda from using echos in other parts of the dungeon while enabling her to reach the exit if she ever makes it on top of the column of echoes. To the left, there are enemies which Zelda must land on that take away 1 heart, which prevent Zelda from revisiting this part of the map (see Figure 1).

Variable Gadget. Each variable gadget is a choice between 1 of 2 drops into a 1 unit deep hole which extends and acts as a wire. Each of the choices represents either setting variable $X = True$ or $\bar{X} = True$ (see Figure 3).

Variable Connecting Gadget. Between 2 variables X_i and X_{i+1}, the variable connecting gadget is used to join the X_i and $\bar{X}_i$ wires into a single wire that feeds in to the X_{i+1} variable gadget to prevent unwanted traversal. This is done by including another 1 unit drop from both wires that then joins into a single wire (see Figure 4).

Clause Gadget. The clause gadget is an empty $7 \times 7 \times 7$ room with an energy crystal located 1 unit above

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

Figure 3: Variable gadget.

Figure 4: Variable connecting gadget.

Figure 5: Clause gadget.

the center (4 units from the floor). On 3 of the walls exists a central opening 4 units from the ground. From each of these openings a platform moves from the ceiling to the center of the room and back, stopping directly over the energy crystal. This makes it so the crystal can be attained by using Zelda's binding ability to first bind to a floating platform, then use the platforms movement to retrieve the energy crystal (see Figure 5).

Wires. Wires are composed of a path closed by 1 unit high walls.

Crossover Gadget. Wires can cross over by using stairs to elevate one wire over the other.

Checking Gadget. The checking gadget consists of a narrow corridor of enemies. In order to defeat these enemies, Zelda must use her swordfighter form. Only if she has enough energy can she defeat all the enemies and jump on the 3 column high echoes placed in the start gadget, where she can then jump to the exit of the dungeon (see Figure 2).

Using these gadgets, we now show NP-hardness through a reduction from 3SAT.

Theorem 1 *Dungeon Traversal in Generalized 3D Zelda with swordfighter form, binding/unbinding and echoes is NP-hard.*

Proof. Given a 3SAT formula Φ with m clauses and n variables:

1. For each clause C_i, create a clause gadget. The 3 openings in the walls connect to wires from variables which satisfy the clause.
2. For each variable X_i, create a variable gadget. Each half of the gadget extends to the clauses which the variable assignment satisfies.
3. For each variable X_i with wires for X_i and $\bar{X}_i$, use the variable connecting gadget to connect these wires to variable X_{i+1}.
4. For variable X_n, use the variable connecting gadget to connect to the finish gadget.

Forward Direction. If Φ is satisfiable, then Zelda is able to traverse to the end of the dungeon. For each assignment in Φ, drop in the tunnel corresponding to the choice and retrieve each energy crystal. Since all clauses are satisfied, every clause gadget can be visited, resulting in enough energy to defeat the enemies in the checking gadget.

Backwards Direction. If the exit to the dungeon is reachable, then Φ is satisfiable. For each choice made at the variable gadgets, assign the corresponding variable the respective assignment in Φ. Since Zelda had enough energy to make it to the end of the dungeon, all clause gadgets were visited, meaning all clauses of Φ can be satisfied.

Why it works. With only 2 hearts, Zelda can only walk over the enemies from the starting gadget once, losing a heart in the process. This prevents Zelda from revisiting the variables once they are assigned. Additionally, when choosing an assignment for each variable, by dropping into the 1 unit hole Zelda is unable to jump back out without using an echo. However, by using an echo, Zelda removes one of the echoes placed at the start, making it impossible to make it to the exit. Thus, once an assignment is chosen, Zelda must stick to it.

To retrieve the energy crystals per gadget, Zelda must bind to the platforms and use it to float over the hole. The platforms are timed such that no 2 overlap, meaning Zelda can not jump or bind from one platform to another. The room being 7 units high is an additional preventative measure to ensure Zelda can not drop to the floor and bind to a different platform.

The 2 wires corresponding to $Truth$ and $False$ assignments for each variable come together as input to the variable connecting gadget, which acts as input to the next variable. Since both wires drop into another 1 unit hole, it is not possible to traverse the $False$ wire from the $Truth$ wire, and vise versa.

Finally, the narrow corridor of enemies ensures that Zelda must pick up all of the energy crystals to clear the path. Without enough energy, Zelda would not be able to jump to the column of 3 unit high echoes, which leads to the exit of the dungeon. □

4 Conclusion

In this paper, we show that 3 of the new mechanics in the latest Legend of Zelda game lead to NP-hardness for dungeon traversal. However, it is still unknown how these new mechanics influence previously known results, which we leave as a treatment for future work.

References

[1] G. Aloupis, E. D. Demaine, A. Guo, and G. Viglietta. Classic nintendo games are (computationally) hard, 2015.

[2] J. Bosboom, J. Brunner, M. Coulombe, E. D. Demaine, D. H. Hendrickson, J. Lynch, and E. Najt. The legend of zelda: The complexity of mechanics, 2022.

Data Extraction and Verification with Generative AI

Ricardo Balvanera* David Medina† Patricio Rodriguez‡

Abstract

The ability to extract structured data from unstructured text is essential for applications across domains, from scheduling systems to autonomous vehicle control. Generative AI offers a promising solution by transforming raw text into structured formats usable for quantitative analysis and programmatic functions. This paper explores the potential of large language models (LLMs) to perform one-shot and zero-shot learning for data extraction, enabling structured outputs directly from text prompts. By examining the application of LLMs in creating usable data from product reviews, we discuss methodologies for prompting, data validation, and reliability analysis. Our findings contribute insights into the strengths and limitations of using Generative AI for structured data extraction and verification, emphasizing the model's versatility and the complexities of ensuring data integrity.

1 Introduction

The rapid growth of data has increased the need for structured information in applications like machine learning and decision support. While traditional algorithms rely on structured data, much of the world's information exists in unstructured text, creating challenges in fields like NLP and customer analytics. Conventional methods for structuring text, such as manual labeling or rule-based systems, face limitations in scalability and flexibility.

Advancements in generative AI, particularly large language models (LLMs) like OpenAI's GPT, offer new possibilities for transforming unstructured text into structured formats through contextual understanding and prompt engineering. LLMs can produce structured outputs directly from unstructured input, which holds promise for applications across diverse industries. However, using LLMs for this purpose brings challenges related to accuracy and consistency, as results can vary with different prompts or contexts. In this study, we focus on using product reviews as a test case to explore methods for structuring data with LLMs, emphasizing prompt engineering and validation strategies to improve data reliability.

2 Experiment

To develop an effective approach for leveraging a LLM for product recommendation, we carefully structured our setup, including the selection of model, prompt design, and dataset curation. We used ChatGPT-4 as our model, selected for its high level of language understanding and contextual capabilities. Our dataset consisted of sampled customer reviews from Amazon.com products, focusing specifically on negative reviews to challenge the model's recommendation capabilities under real-world conditions. These negative reviews varied widely in tone and detail, with some describing specific product issues while others conveyed more generalized dissatisfaction. This variation allowed us to assess how well the model could interpret and respond with suitable alternative recommendations, even when key issues were either explicitly stated or only implied. By designing a range of prompts and scenarios, we aimed to gauge the model's ability to adapt to different levels of detail and context within user feedback.

3 Our Results

In our testing phase, we explored multiple prompt designs, passing them through the ChatGPT API to assess response quality and relevance. Initial attempts that involved simply passing raw review data resulted in inconsistent and sometimes generic recommendations, indicating that the model needed additional context to understand the task. To enhance performance, we revised our approach by creating context-based prompts. These prompts explicitly instructed the model that it would receive a user review and was tasked with providing alternative product suggestions tailored to the specific issues or preferences expressed in the review. This adjustment proved highly effective, as it significantly improved the relevance and accuracy of the recommendations generated.

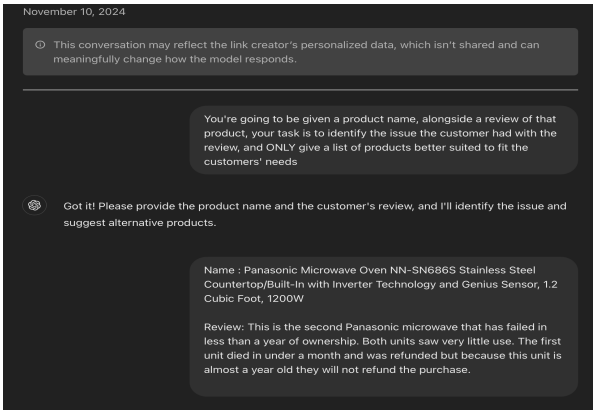


Figure 1: Example of an Effective Prompt

With this context-driven prompt structure, ChatGPT-4 began to consistently produce responses aligned with

*Department of Computer Science, University of Texas Rio Grande Valley, ricardo.balvanera01@utrgv.edu

†Department of Computer Science, University of Texas Rio Grande Valley, david.medina05@utrgv.edu

‡Department of Computer Science, University of Texas Rio Grande Valley, patricio.rodriguez04@utrgv.edu

This is an abstract of work done at $\mathcal{H}\alpha c\kappa\mathbb{R}$ 2024. It should be considered a workshop preprint.

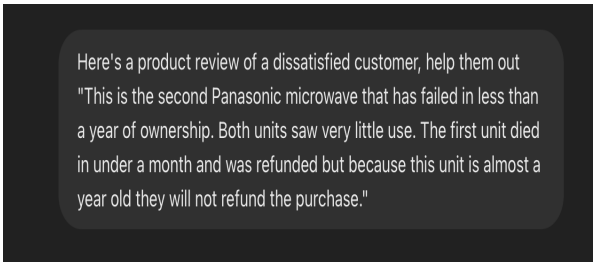


Figure 2: Example of an Ineffective Prompt

our overall goal. The model's recommendations were returned as a numbered list of alternatives. This outcome demonstrated that providing a clear task helps to maximize the LLM's utility in providing a structured answer with user-focused suggestions in mind.

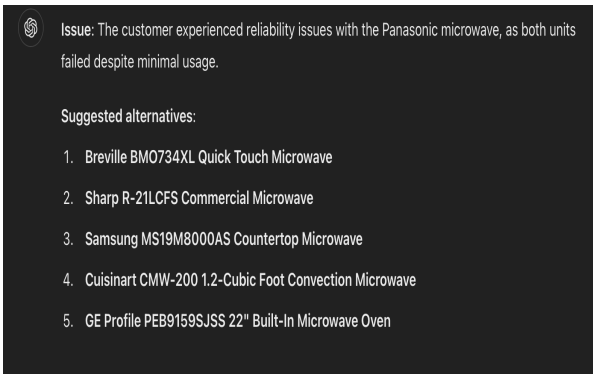


Figure 3: Result from Effective Prompting

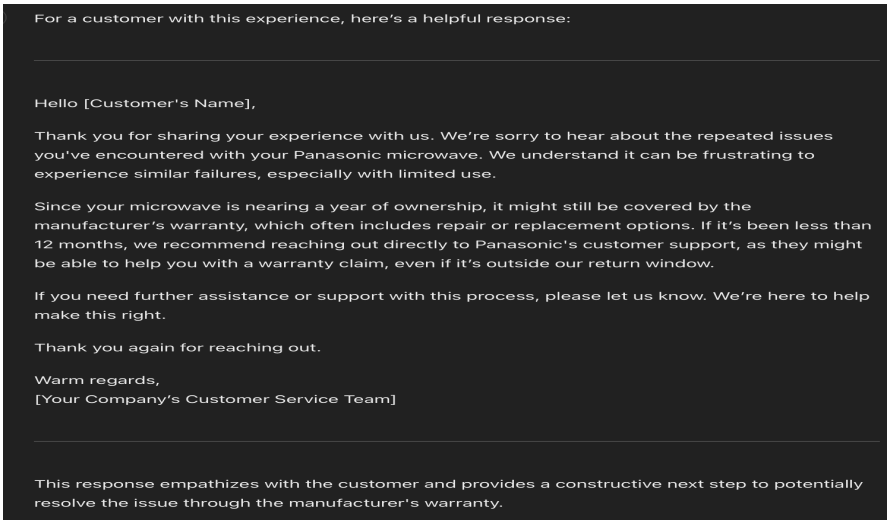


Figure 4: Result from Ineffective Prompting

4 Conclusion

Through these findings, we observed the model's potential for effective recommendation generation when equipped with the right contextual cues. By using carefully designed prompts, we were able to guide ChatGPT-4 to produce relevant recommendations based on the specific issues and preferences in customer feedback. This approach underscores the importance of clear instructions to improve the quality of the model's responses. Future work could explore ways to automate the validation process and create more flexible prompts for different types of text, supporting a wider range of uses for structured data generation across various fields.

References

[1] Jamie. Panasonic microwave oven nn-sn686s, 2023.

[2] OpenAI, R. Balvanera, D. Medina, and P. Rodriguez. Gpt 4 prompt engineering examples, 2024.

[]

Author Index

Alvarado, Belinda, 8
Avila Jimenez, Alberto, 10

Bajaj, Divya, 30, 34
Balvanera, Ricardo, 38
Barreda, David, 30, 34
Berlanga, Mya, 17
Biteng, Pigar, 1

Camarena, Mario, 12
Castellanos, Jose Luis, 19, 32
Castilleja, Diana, 14

Garcia, Enrique, 4
Garcia, Miguel, 1
Garza, Andrea, 12

Hernandez-Noriega, Joshua, 22
Hinojosa, Chris, 6
Hitchcox, Paul, 28

Knobel, Ryan, 34, 36

Lopez, Koriel, 6
Lopez, Lillian, 6
Lugo, Hector, 25

Martinez, Luis, 10
Massie, Aiden, 30
Medina, David, 38
Meza Canales, Arturo, 25
Moreno, Alissen, 14

Nurbek, Gaukhar, 36

Orta, Jorge, 25

Perez, Juan Manuel, 36
Peña, Christian, 4

Reyes, Angel, 4
Rodriguez, Patricio, 38

Salazar, Robert, 22
Salinas, Adrian, 4
Sanchez, Jaycee, 28
Santos, Pablo, 19, 32
Santos, Ramiro, 17
Sauceda, Oziel, 12
Solis, Arely, 8

Villarreal, Damian, 1