GPT-Powered Blockchain Simulation Modeling: Enhancing Accessibility and Flexibility

***Abstract*—The rapid evolution of blockchain technology has transformed various industries, offering decentralization and security. However, experimenting with real-world blockchain scenarios involves substantial temporal and economic costs. This paper presents a novel approach that leverages Generative Pre-trained Transformers (GPT) to enhance accessibility and flexibility in blockchain simulation modeling. GPT, known for its Natural Language Processing (NLP) capabilities, is utilized to generate blockchain simulation models based on user-provided experimental setups described in natural language. This streamlines the simulation model creation process, lowering barriers for researchers and addressing adaptability needs in the evolving blockchain landscape. We demonstrate the feasibility of our approach by applying GPT-3.5 to replicate an existing blockchain simulator, detailing the modeling process and discussing its limitations.**

***Index Terms*—Blockchain, Simulation, GPT, Generative AI**

I. Introduction

In recent years, blockchain technology has rapidly evolved, introducing innovative changes across various domains. The inherent decentralized and secure nature of blockchain has made it a cornerstone in enhancing transparency, traceability, and efficiency in various industries. This dynamic progression in blockchain necessitates extensive experimentation in the research and development processes, particularly in dealing with larger-scale network infrastructures and data sets. However, creating a real-world experimental environment and conducting experiments in real-world scenarios incurs significant temporal and economic costs.

A computer simulation can serve as a viable solution to address these challenges. It enables cost-effective testing of novel ideas in environments closely resembling real-world scenarios, leveraging a single machine and minimizing financial investments. Yet, creating simulation environments for large-scale distributed ledger technologies like blockchain remains complex and requires substantial effort, often exceeding the capabilities of individuals without expertise in simulation design. Moreover, existing blockchain simulators often come with constraints on flexibility, particularly in configuring experimental conditions, limiting their adaptability. Adapting blockchain simulators to be flexible and constructible on-the-fly based on situational demands becomes crucial. This is where Generative Pre-trained Transformers (GPT) [1] can play a significant role.

GPT, at the forefront of generative AI, are revolutionizing artificial intelligence with their exceptional Natural Language Processing (NLP) capabilities. As a type of generative AI, GPT leverages a pre-trained model, such as GPT-3 [2], [3], to understand and generate human-like text based on the input it receives. This technology excels in tasks like language translation, text completion, and question answering. By employing a vast dataset during its training phase, GPT acquires a broad understanding of linguistic nuances, enabling it to generate contextually relevant and coherent responses. Beyond traditional language-related tasks, GPT showcases its extraordinary versatility by extending its capabilities to the field of code generation. Remarkably, GPT can synthesize source code, offering a notable solution for developers and industries seeking automated programming assistance.

This paper proposes an innovative approach for blockchain simulation modeling by harnessing the capabilities of GPT. GPT is employed to generate a blockchain simulation model capable of executing specific experimental setups provided by users through detailed description texts. Traditionally, constructing blockchain simulation models involved intricate technical knowledge and considerable effort. However, our proposed approach leverages the NLP capabilities of GPT to streamline and facilitate the simulation model creation process. Users can now input detailed descriptions of their experimental setups, and GPT autonomously translates this textual information into a functional blockchain simulation model. By employing GPT for blockchain simulation modeling, we aim to enhance accessibility and flexibility in the experimentation process. This approach not only reduces the barrier to entry for researchers with varying levels of technical expertise but also addresses the need for adaptability in the rapidly evolving blockchain landscape. The integration of GPT into the simulation workflow marks a significant step towards creating more user-friendly and responsive tools for blockchain research and development. To validate our proposed approach, we utilized GPT-3.5 [4] to model blockchain simulation using the description of an existing blockchain simulator. We present the modeling process and discuss the limitations of our approach.

The remainder of the paper is organized as follows. In Section II, we cover related work, exploring existing blockchain simulators and the application of GPT in simulation modeling. Section III provides a comprehensive technical background, and we present our methodology in Section IV. Further, we validate the proposed methodology in Section V and provide discussions in Section VI. Finally, we conclude this paper in Section VII.

II. Related Work

Our work, to the best of our knowledge, represents the first attempt to integrate GPT into blockchain simulation modeling.

Previous studies in the blockchain domain have focused on diverse evaluation methods for new technical approaches or the development of specialized blockchain simulators tailored to specific purposes. This section provides an overview of existing blockchain simulators and explores the experimental objectives behind their development. Additionally, we introduce research in simulation modeling utilizing GPT beyond the blockchain domain.

A. Existing Blockchain Simulators

Existing blockchain simulators play a role in assessing the security and performance of different blockchain systems. Gervais et al. [5] conducted a comprehensive analysis of Proof-of-Work (PoW) blockchains, focusing on security aspects. They introduced a quantitative framework to analyze the security implications of various consensus and network parameters, considering real-world constraints such as network propagation, block sizes, and generation intervals. Aoki et al. [6] developed SimBlock, a versatile blockchain network simulator that allows easy modification of node behaviors. This enables the investigation of how nodes' actions influence blockchain dynamics. The simulator was validated by comparing simulation results with measured values from actual blockchains, and practical experiments explored the impact of neighbor node selection algorithms and relay networks on block propagation time. Faria and Correia [7] presented BlockSim:Faria, a discrete-event simulator designed to evaluate different blockchain implementations. The simulator proved flexible in modeling and simulating Bitcoin and Ethereum, providing insights into performance under varied conditions. Notably, the study found that doubling the number of transactions per block had a minimal impact on block propagation delay, while encrypting communication significantly affected the delay. Alharby and Van Moorsel [8] introduced BlockSim:Alhardy as a simulation framework for blockchain systems, organized into incentive, connector, and system layers. The emphasis was on modeling and simulating block creation through the PoW consensus algorithm, providing a valuable tool for investigating diverse configuration choices in blockchain design and deployment.

Paulavičius et al. [9] conducted a systematic review and empirical analysis of blockchain simulators, recognizing their importance in evaluating complex distributed systems. The study highlighted the extensibility of simulators, allowing testing at large scales with different settings and parameters. The comprehensive review provided insights into the features and limitations of selected simulators, offering valuable guidance for future research directions in the field. They also found that most simulators primarily focused on simulating PoW blockchain, and instead of fully implementing the entire blockchain system, they specifically developed essential layers for experimental purposes. The rest of the components were simplified.

B. GPT in Simulation Modeling

As GPT is a relatively recent and emerging technology, there has been limited research on the use of GPT in the specific field of simulation modeling. Jackson et al. [10] presented a study that investigates the automation of simulation modeling for logistics systems using GPT, particularly focusing on the collaboration between human experts and AI-based systems. Their framework, built upon the advanced GPT-3 Codex, demonstrated the ability to automatically generate functionally valid simulations for queuing and inventory management systems when provided with verbal explanations. The refined language model exhibited proficiency in producing simulation models for inventory and process control, ultimately simplifying the traditionally complex process of simulation model development. This work established a technological foundation for human-AI collaboration in simulation modeling, providing valuable guidelines for building simulation models of logistics systems automatically based on natural language descriptions.

On the other hand, in the opposite direction, Jackson and Rolf [11] concentrated on the intersection of NLP models and simulation understanding, specifically applying GPT-3 to translate simulation source code into English. Recognizing the critical need for modeling inventory management systems, the study addresses the inherent complexities of simulation modeling, including technical intricacies and the iterative exchange of information between domain experts and simulation engineers. Their exploration emphasizes the potential of state-of-the-art NLP systems, such as GPT-3, in comprehending the core principles and domain-specific context behind simulations of inventory management systems. By providing a proof of concept, the study underscores the capability of advanced NLP models to facilitate the development of simulation models for complex systems, mitigating challenges associated with technical complexity and communication gaps between experts and engineers.

Together, these studies contribute to the evolving landscape of simulation modeling, showcasing the important role of GPT-based models in automating and enhancing the development process, from generating simulations based on natural language descriptions to translating simulation source code into a more understandable and accessible format.

III. Technical Background

In this section, we provide a comprehensive technical background to contextualize the proposed approach of utilizing GPT for blockchain simulation modeling. We begin by offering a structural overview of the blockchain simulation model, explore the technical intricacies of GPT, and emphasize the significance of prompt engineering in the context of this innovative fusion.

A. Structural Overview of Blockchain Simulators

Blockchain simulators are essential tools for understanding, validating, and optimizing the intricacies of blockchain systems. Their architectures comprise several layers essential for emulating the multifaceted nature of blockchain systems and understanding these layers is crucial for a comprehensive understanding of blockchain system dynamics.

- *Data Layer*: This layer simulates data storage and management on the blockchain, considering block structure, transaction validation, and storage mechanisms.
- *Consensus Layer*: Replicating the chosen consensus algorithm (e.g., Proof-of-Work, Proof-of-Stake) [12] allows researchers to assess its impact on network security and efficiency.
- *Network Layer*: Simulating communication protocols, network topology, and peer interactions provides insights into the scalability and resilience of the blockchain network.
- *Smart Contract Layer*: The inclusion of a smart contract layer enables the simulation of decentralized application (DApp) [13] execution and interactions between smart contracts, facilitating the analysis of contract-based functionalities.

In addition to the layered architecture, it is essential to investigate the structural overview of a typical blockchain simulation model. A fundamental representation involves the following key components:

- *Node Configuration*: Describing the characteristics and behaviors of individual nodes within the simulated blockchain network, including attributes such as mining power, transaction processing capacity, and consensus participation.
- *Transaction Generation*: Detailing the process by which transactions are created and propagated through the network, considering factors such as transaction types, frequency, and complexity.
- *Consensus Mechanism*: Expanding on the simulation of the consensus algorithm chosen for the blockchain model, highlighting how nodes reach agreement on the state of the blockchain.
- *Smart Contract Execution*: Simulating the execution of smart contracts within the network, encompassing the life cycle of smart contracts from deployment to invocation.

This structural overview provides a comprehensive understanding of the simulated blockchain environment, setting the stage for the subsequent discussion on the integration of GPT.

B. GPT and Code Generation

GPT (Generative Pre-trained Transformer), a state-of-the-art language model built on the transformer architecture, has revolutionized NLP tasks. Pre-trained on diverse language corpora, GPT shows an outstanding ability to understand and generate human-like text across a wide range of contexts. At its core, GPT relies on a transformer neural network architecture, which allows it to capture intricate patterns and dependencies in language data. The model's architecture is characterized by self-attention mechanisms, enabling it to consider the contextual relationships between words in a given sequence. This capability facilitates the generation of coherent and contextually relevant text.

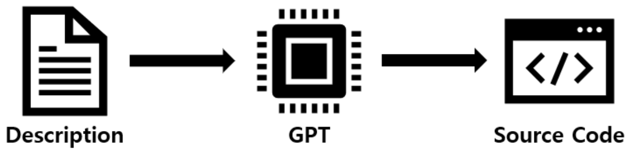


Fig. 1. Code Generation Using GPT

The versatility of GPT extends beyond natural language understanding to include code generation (Fig. 1). GPT is trained on vast repositories of code from various programming languages, enabling it to understand the syntactic and semantic structures inherent in coding practices. When prompted with a coding-related query or context, GPT can generate code snippets that align with the specified requirements.

C. Prompt Engineering

Prompt engineering is a critical aspect of leveraging GPT for specific tasks. Crafting precise and well-structured prompts enables GPT to generate outputs aligned with user intent. In the context of blockchain simulation modeling, prompt engineering becomes essential for translating textual descriptions of experimental setups into functional simulation models. The nuanced design of prompts empowers GPT to navigate the intricacies of blockchain technology and produce accurate and context-aware simulation models.

Effectively instructing language models like GPT for code generation involves thoughtful prompt engineering. Considerations for crafting successful prompts include providing clear and specific task descriptions, incorporating example input-output pairs to illustrate desired behavior, structuring prompts with placeholders for systematic code generation, and embedding descriptive comments for better semantic understanding.

Furthermore, influence the model's choices by including specific variable and function names in prompts, offering contextual information about the programming environment, and adopting an iterative refinement process based on model outputs. Break down complex tasks into manageable subtasks, address potential ambiguities, and include error handling and edge cases for robust code generation.

Through continuous experimentation and evaluation, the iterative nature of prompt refinement enhances the accuracy and effectiveness of code generation, contributing to improved model performance over time.

By delving into these technical aspects, we lay the groundwork for understanding how the proposed integration of GPT into blockchain simulation modeling addresses the challenges posed by traditional simulators, offering a more accessible and flexible approach for researchers and developers.

IV. Methodology

In this section, we propose a novel methodology that integrates GPT into the intricate process of blockchain simulation modeling. The methodology, illustrated in Fig. 2, delineates a systematic approach wherein researchers collaboratively engage with GPT to iteratively refine and generate a sophisticated blockchain simulation model.

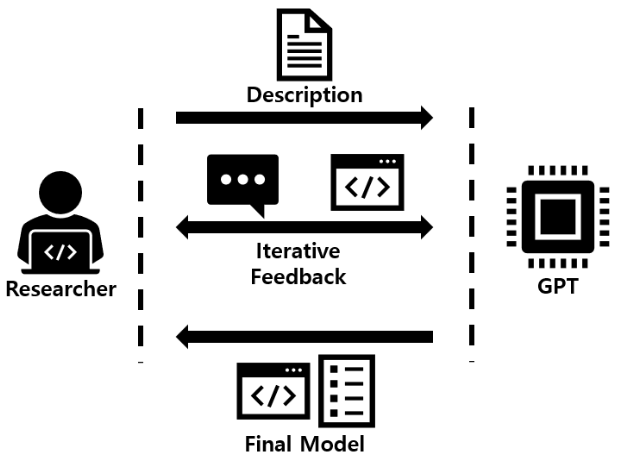


Fig. 2. GPT-Powered Blockchain Simulation Modeling Approach

A. Prompt Design

The initiation of the process demands the formulation of meticulously crafted prompts. These prompts serve as a conduit for articulating the intricate requirements and specifications governing the desired blockchain simulation model. The precision and clarity in prompt design are paramount, as they directly influence GPT's capacity to comprehend the researcher's intent and subsequently generate a model of high completeness.

In this study, these prompts are designed to encompass essential details, including the specific code generation task for blockchain simulation modeling, the chosen programming language, parameter specifications, output format preferences, and the potential provision of initial parameter values. Additionally, researchers have the autonomy to specify the composition and nomenclature of the resulting source code files. By embedding such comprehensive information within the prompts, researchers exercise precise control over the code generation process. This step not only bolsters the accuracy of the generated models but also affords researchers a nuanced and customizable experience, enhancing their ability to tailor simulation models to specific experimental needs.

B. Input into GPT Model

The formulated prompts undergo input into a pre-trained GPT model, renowned for its proficiency in natural language understanding. GPT scrutinizes the prompts, extracting nuanced details pertaining to the experimental setup, and produces an initial simulation model reflective of the researcher's input.

C. Model Verification and Iterative Feedback Loop

Researchers rigorously scrutinize the initially generated simulation model to ascertain its fidelity to the specified requirements. The meticulous examination involves an in-depth assessment of the simulation model's adherence to the outlined parameters and intended functionalities. Identification of any discrepancies or areas necessitating refinement prompts researchers to actively engage in an iterative feedback loop with GPT. This iterative refinement process is characterized by a dynamic exchange between researchers and the model, wherein researchers provide detailed feedback, articulate necessary adjustments, and iteratively guide GPT towards generating an increasingly refined simulation model. Such an iterative approach ensures continual improvements, supporting a collaborative and adaptive process that aligns seamlessly with the evolving objectives and expectations of the researchers involved in blockchain simulation modeling. This process persists until the researcher attains a level of satisfaction with the model's completeness, accuracy, and alignment with the overarching research objectives.

Explicit modifications to the prompt also provide a nuanced means for researchers to guide the model's understanding and execution. This iterative refinement not only underscores the adaptability of the proposed methodology but also empowers researchers to iteratively tailor the simulation model to their evolving research inquiries, thereby enhancing the model's effectiveness and relevance in the context of blockchain experimentation.

D. Model Finalization

The model finalization stage delivers the refined simulation model with a comprehensive documentation that introduces configurations, relevant code snippets, and specifications integral to the simulation. The documentation provides researchers with a detailed and insightful overview of the generated model, offering insights into the intricacies of the code, the underlying parameters, and the overall structure of the simulation.

E. Experimentation and Parameter Tuning

This stage allows researchers to examine the intricacies of the simulation, undertaking nuanced adjustments to algorithms, parameters, and other pertinent facets. The flexibility inherent in the proposed methodology enables researchers to systematically explore a spectrum of scenarios within the blockchain landscape. This experimentation phase is characterized by the iterative testing of diverse configurations and scenarios, providing researchers with the opportunity to fine-tune the simulation model based on specific research inquiries. By engaging in systematic parameter tuning and iterative experimentation, researchers can gain valuable insights into the behavior and performance of the blockchain simulation model under varying conditions.

The proposed methodology, characterized by a cooperative interaction between researchers and GPT, aspires to streamline the intricate process of blockchain simulation modeling. Rooted in the principles of accessibility and flexibility, this approach empowers researchers to tailor simulation models to their evolving research inquiries. By providing researchers with a customizable and responsive framework, the methodology contributes to supporting a more inclusive and adaptive paradigm within the field of blockchain experimentation. This adaptability not only enhances the accessibility of blockchain simulation modeling but also underscores the potential for a broader range of researchers to actively participate and contribute to advancements in blockchain research and development.

V. Experiments

In this section, we demonstrate the generation of a blockchain simulation model through the proposed methodology. We aimed to validate our approach by developing a simulator capable of conducting equivalent experiments to those presented in a technical paper presenting SimBlock [6], one of the existing blockchain simulators that were introduced in Section II. Our simulation model, called GPT-SimBlock, was constructed based on the description outlined in the SimBlock paper.

To initiate the validation process, we created an initial prompt for the GPT model, drawing inspiration directly from the descriptions provided in the SimBlock paper. The formulated prompt was designed to encapsulate the key parameters and scenarios relevant to blockchain simulation modeling. Leveraging the GPT-3.5 model developed by OpenAI [14], specifically the ChatGPT [15] variant, we executed the model through a web interface. The prompt, created in the context of the SimBlock experiments, was input into the ChatGPT interface to initiate the simulation model generation process.

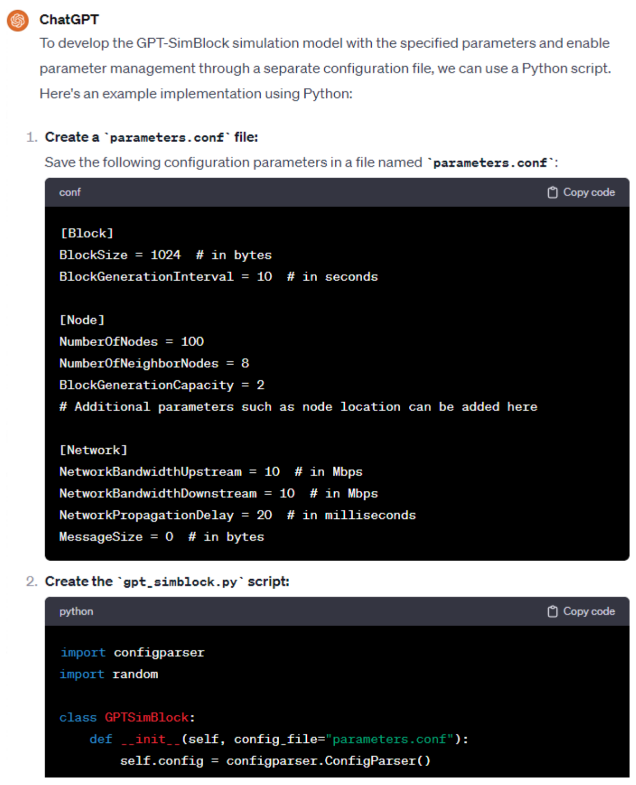
The designed initial prompt is as follows:

Develop an event-driven Proof-of-Work blockchain simulation model named GPT-SimBlock. It simulates the exchange of Block and Inv messages among nodes located in six different regions, modeling the process of mining blocks. The simulation aims to derive values for the median block propagation time and the rate of fork. Researchers can execute the simulation by adjusting the following parameters. Enable the management of parameters by creating a separate "parameters.conf" file.

Block parameters:
- Block Size: Size of the block generated by the node.
- Block Generation Interval: Targeted time for generating a block

Node parameters:
- Number of Nodes: Total nodes in the blockchain network.
- Number of Neighbor Nodes: Neighboring nodes for each node.
- Location of the Node: Geographical location.
- Block Generation Capacity: Capacity of each node to generate blocks.

Network parameters:
- Network Bandwidth: Upstream and downstream bandwidths for each region.
- Network Propagation Delay: Average propagation delay between regions with dispersion.
- Message Size: Simulated as 0 bytes for most messages except block messages.
- Transmission Time: Determined by message size and bandwidth.
- Message Arrival Time: Calculated using propagation delay and bandwidth.


ChatGPT

To develop the GPT-SimBlock simulation model with the specified parameters and enable parameter management through a separate configuration file, we can use a Python script. Here's an example implementation using Python:

1. **Create a `parameters.conf` file:**

   Save the following configuration parameters in a file named `parameters.conf`:

```conf
[Block]
BlockSize = 1024  # in bytes
BlockGenerationInterval = 10  # in seconds

[Node]
NumberOfNodes = 100
NumberOfNeighborNodes = 8
BlockGenerationCapacity = 2
# Additional parameters such as node location can be added here

[Network]
NetworkBandwidthUpstream = 10  # in Mbps
NetworkBandwidthDownstream = 10  # in Mbps
NetworkPropagationDelay = 20  # in milliseconds
MessageSize = 0  # in bytes
```

2. **Create the `gpt_simblock.py` script:**

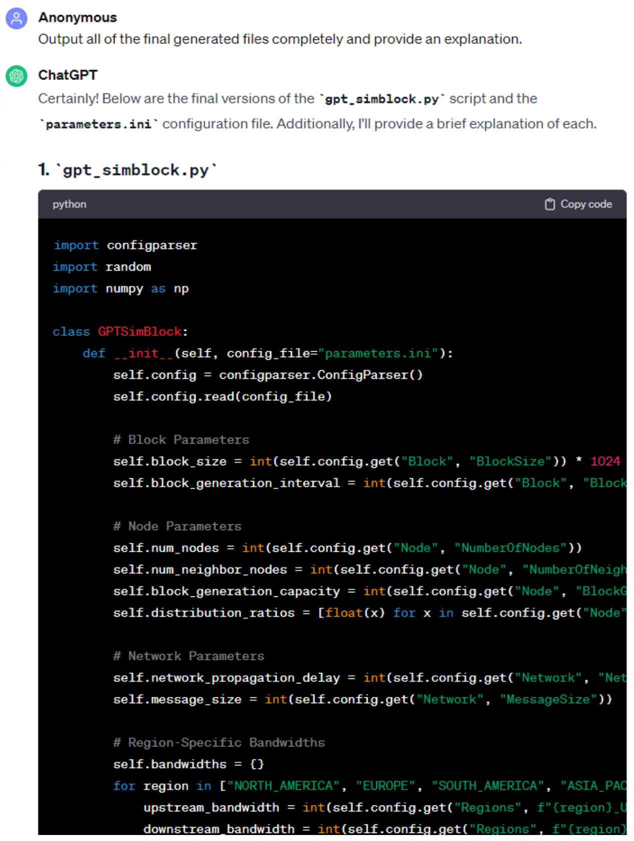
```python
import configparser
import random

class GPTSimBlock:
    def __init__(self, config_file="parameters.conf"):
        self.config = configparser.ConfigParser()
```

Fig. 3. GPT's Output in Response to the Initial Prompt

As a result, GPT generated a simplified version of GPT-SimBlock in Python, incorporating several files and classes. Fig. 3 is a part of GPT's Output in Response to the Initial Prompt. The Python script includes a class for blockchain simulation, along with a parameter configuration file requested from the prompt. While the basic structure for blockchain simulation was established, only a subset of the requirements from the prompt were accurately reflected. Notably, details such as the class for blocks and the geographical distribution of nodes were not included at this stage. Recognizing the challenge of overwhelming GPT with extensive requirements in a single prompt, we adopted an iterative approach, providing GPT with prompts containing one requirement at a time to progressively refine the simulation model.

Subsequently, specific details were introduced, specifying the geographical distribution of nodes into six regions and adjusting bandwidth and latency values accordingly. Additionally, the block size was changed to KiB units, and the simulation was configured to generate a total of 10,000 blocks. All of these details were intended to follow the experimental setups of SimBlock. Moreover, the concept of difficulty was incorporated to simulate block creation, and the parameters file was transformed into an INI file format. Furthermore, the propagation of blocks in a P2P network was considered in terms of node distribution, bandwidth, and latency. The concept of the fork and the calculation of the fork rate were introduced. A detailed dialogue of this interactive process can be found in [16].



Anonymous
Output all of the final generated files completely and provide an explanation.

ChatGPT
Certainly! Below are the final versions of the `gpt_simblock.py` script and the `parameters.ini` configuration file. Additionally, I'll provide a brief explanation of each.

1. `gpt_simblock.py`

```python
import configparser
import random
import numpy as np

class GPTSimBlock:
    def __init__(self, config_file="parameters.ini"):
        self.config = configparser.ConfigParser()
        self.config.read(config_file)

        # Block Parameters
        self.block_size = int(self.config.get("Block", "BlockSize")) * 1024
        self.block_generation_interval = int(self.config.get("Block", "Block

        # Node Parameters
        self.num_nodes = int(self.config.get("Node", "NumberOfNodes"))
        self.num_neighbor_nodes = int(self.config.get("Node", "NumberOfNeigh
        self.block_generation_capacity = int(self.config.get("Node", "BlockG
        self.distribution_ratios = [float(x) for x in self.config.get("Node"

        # Network Parameters
        self.network_propagation_delay = int(self.config.get("Network", "Net
        self.message_size = int(self.config.get("Network", "MessageSize"))

        # Region-Specific Bandwidths
        self.bandwidths = {}
        for region in ["NORTH_AMERICA", "EUROPE", "SOUTH_AMERICA", "ASIA_PAC
            upstream_bandwidth = int(self.config.get("Regions", f"{region}_U
            downstream_bandwidth = int(self.config.get("Regions", f"{region}
```

Fig. 4. GPT's Final Response with Finalized Model

The final simulation model (Fig. 4), encapsulated in Python and parameter configuration files, showed a well-structured framework for simulating PoW blockchains. However, its direct usage for research purposes proved challenging, as still some variable types were inaccurately defined, and internal simulation logic deviated slightly from real-world scenarios.

We compared the features of SimBlock and GPT-SimBlock in Table I. We utilized the SimBlock source code release version 0.7.0 [17] available on GitHub at the time of publication for the comparison. We confirmed that most of the data structures and parameters required for simulation were appropriately designed in GPT-SimBlock as well. In the case

TABLE I
COMPARISON OF SIMBLOCK AND GPT-SIMBLOCK

Feature	**SimBlock**	**GPT-SimBlock**
Programming Language	Java	Python
Block Data Class	O	O
Block Interval	O	O
Block Size	O	O
Final Block Height	O	O
Node Data Class	O	X
The Number of Nodes	O	O
Regional Node Distribution	O	O
Latency Between Regions	O	O
Bandwidth Between Regions	O	O
P2P Network Construction	O	O
Block Propagation Over P2P Network	O	X
Block Propagation Time Calculation	O	△
Fork Occurrence Checking	O	O

of node data, unlike SimBlock, GPT-SimBlock did not create a separate data class but managed the necessary attributes through multiple variables in a list structure. The most significant difference and issue in GPT-SimBlock compared to SimBlock were observed in the simulation of block propagation. While GPT-SimBlock could properly construct the P2P network according to the given parameters, the process of propagating blocks generated by nodes was developed to directly transmit from the node that successfully mined the block to all other nodes in a single hop, rather than following the paths in the P2P network. Therefore, the block propagation time could not be accurately calculated, despite providing an appropriate calculation logic utilizing the provided latency and bandwidth between regions. We concluded that this was due to a lack of GPT's understanding of blockchain system.

Nevertheless, the source code of the generated simulation model serves as a robust starting point for constructing a complete simulator. By adapting and debugging certain simulation logics to suit specific requirements, a reliable simulation experiment environment can be established.

In addition to addressing the specified requirements outlined in the experiment prompts, our findings reveal the remarkable versatility of GPT in providing support for a diverse array of tasks. Notably, GPT demonstrated proficiency in generating outputs for simulation results in various formats, such as simple numerical values or graphical representations. Moreover, the model exhibited the capability to generate code facilitating the visualization of simulation outcomes, showcasing its adaptability in translating abstract requirements into concrete implementation.

Furthermore, GPT showed its ability to handle intricacies beyond the given prompts. Specifically, the model demonstrated competence in generating code to integrate specific libraries into the source code. This extended functionality included the provision of instructions on utilizing designated libraries or incorporating pre-existing libraries into the generated code. These results affirm the versatility of GPT as a powerful tool for enhancing the accessibility and flexibility of blockchain simulation modeling, transcending limitations commonly associated with conventional approaches.

VI. Discussion

This section provides insights garnered through experimentation and emphasizes key points for consideration.

A. GPT-Powered Blockchain Simulation Modeling

In this work, we validated the viability of our approach by modeling GPT-powered blockchain simulation, iteratively providing requirements for blockchain simulation models to GPT. Simultaneously, we also identified inherent limitations of the approach. The process of leveraging GPT for the generation of blockchain simulation models proved to be a promising starting point for easily and flexibly configuring simulation environments according to researchers' needs. However, it is important to note that relying solely on GPT for the creation of complete simulation models presents challenges.

The iterative nature of our methodology allowed us to refine the interaction between GPT and the specifications for blockchain simulation, revealing the potential for GPT to serve as a valuable tool in constructing the foundational components of simulation models. Nevertheless, limitations emerged because GPT, by design, lacks a comprehensive understanding of intricate technical details and domain-specific nuances that are integral to the precise modeling of blockchain systems.

Therefore, while GPT contributes significantly to the accessibility and flexibility of blockchain simulation environments, researchers should approach its use judiciously. Yet, the integration of domain-specific expertise remains crucial for refining and validating the simulation models generated through GPT. Future advancements may address these limitations, but our current findings emphasize the importance of a collaborative approach, combining the strengths of AI models like GPT with human expertise, to ensure the accuracy and reliability of blockchain simulation outcomes.

B. Importance of Blockchain Simulation

While the aforementioned limitations persist, further research on blockchain simulation modeling akin to this study is imperative. Blockchain simulation holds paramount importance in the field of blockchain research and development. While mainnet, testnet, and private networks offer valuable insights into real-world scenarios, the inherent challenges associated with these environments necessitate the integration of simulation methodologies.

Accessing comprehensive information about the entire network on the mainnet can be challenging. Moreover, the costs associated with network transactions, including fees and resource consumption, can hinder extensive experimentation. Simulations provide a controlled environment to study network behaviors without incurring these financial and operational burdens. Testnets, while serving as testing grounds, often present difficulties in obtaining precise and comprehensive data. Furthermore, differences between testnet and mainnet introduce uncertainties, making it challenging to entirely trust performance experiment results. Simulations bridge this gap by offering a controlled and adjustable environment, enhancing the reliability of experimentation outcomes. Moreover, creating large-scale, intricate networks in private settings demands substantial time and economic investment from researchers. The manual configuration of such networks introduces complexities and limitations. Simulations mitigate these challenges by enabling researchers to model diverse scenarios economically and efficiently, facilitating a more agile and cost-effective research process.

C. Integrating GPT into Blockchain Research

The integration of GPT into blockchain research presents a promising avenue for exploration. Beyond the scope of blockchain simulation, GPT's capabilities extend to areas such as smart contract creation and auditing.

1) Smart Contract Creation: GPT's NLP and code generation ability position it as a valuable tool for enhancing smart contract creation. Researchers can leverage GPT to streamline the process of generating smart contract code, potentially reducing development time and improving code quality. We could find several studies [18], [19] that have worked on this.

2) Smart Contract Audit: GPT's ability to understand and generate human-like text extends to auditing processes. Utilizing GPT for auditing smart contracts can enhance the efficiency and accuracy of code reviews, providing an additional layer of security in blockchain applications. Many studies [20]–[24] have already been conducted.

The integration of GPT into blockchain research represents a synergy between a novel technology and established research directions. This collaboration opens avenues for innovative approaches, demonstrating the adaptability and versatility of blockchain research methodologies. Embracing GPT in blockchain research is not just an evolution but a paradigm shift. It introduces a novel perspective, allowing researchers to explore uncharted territories and redefine the boundaries of what is achievable in blockchain development. As GPT continues to advance, its integration into blockchain research stands as a testament to the dynamic synergy between cutting-edge technologies and the ever-evolving blockchain landscape.

VII. Conclusion

In this work, we proposed an innovative method for blockchain simulation modeling, harnessing the power of GPT. Our approach enables users to describe experimental setups in natural language, with GPT autonomously translating this information into functional blockchain simulation models. This not only streamlines the model creation process but also enhances accessibility for researchers with varying technical expertise. By leveraging GPT's NLP capabilities, we address the complexity of large-scale distributed ledger simulations, offering a cost-effective and flexible solution. Our experimentation with GPT-3.5 showcased the feasibility of our approach, providing insights into the modeling process and acknowledging its limitations.

As part of future work, we envision expanding the capabilities of our approach by integrating more advanced and powerful GPT models. Additionally, considering the dynamic nature of blockchain technology, we propose exploring the development of a GPT model pre-trained specifically on blockchain-related knowledge. This dedicated knowledge base can potentially enhance the accuracy and relevance of GPT-generated simulation models in the blockchain domain. By advancing the integration of state-of-the-art GPT models and domain-specific knowledge, we aim to further empower researchers and developers in the blockchain space, creating more sophisticated and adaptive simulation tools.

References

[1] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] GPT-3, Available at https://platform.openai.com/docs/models/gpt-3, accessed: 2023-12-11.

[4] GPT-3.5, Available at https://platform.openai.com/docs/models/gpt-3-5, accessed: 2023-12-11.

[5] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.

[6] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "Simblock: A blockchain network simulator," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 325–329.

[7] C. Faria and M. Correia, "Blocksim: blockchain simulator," in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 439–446.

[8] M. Alharby and A. Van Moorsel, "Blocksim: a simulation framework for blockchain systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 135–138, 2019.

[9] R. Paulavičius, S. Grigaitis, and E. Filatovas, "A systematic review and empirical analysis of blockchain simulators," *IEEE access*, vol. 9, pp. 38 010–38 028, 2021.

[10] I. Jackson, M. Jesus Saenz, and D. Ivanov, "From natural language to simulations: applying ai to automate simulation modelling of logistics systems," *International Journal of Production Research*, pp. 1–24, 2023.

[11] I. Jackson and B. Rolf, "Do natural language processing models understand simulations? application of gpt-3 to translate simulation source code to english," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 221–226, 2023.

[12] B. Sriman, S. Ganesh Kumar, and P. Shamili, "Blockchain technology: Consensus protocol proof of work and proof of stake," in *Intelligent Computing and Applications: Proceedings of ICICA 2019*. Springer, 2021, pp. 395–406.

[13] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. Leung, "Decentralized applications: The blockchain-empowered software system," *IEEE access*, vol. 6, pp. 53 019–53 033, 2018.

[14] OpenAI, Available at https://openai.com/, accessed: 2023-12-11.

[15] ChatGPT, Available at https://openai.com/chatgpt, accessed: 2023-12-11.

[16] GPT-SimBlock-Dialogue, Available at https://chat.openai.com/share/5953ac14-aa17-4e55-aafd-94a2bd5dd6ad, accessed: 2023-12-11.

[17] SimBlock-0.7.0, Available at https://github.com/dsg-titech/simblock/tree/v0.7.0, accessed: 2023-12-11.

[18] N. Petrović and I. Al-Azzoni, "Model-driven smart contract generation leveraging chatgpt," in *International Conference On Systems Engineering*. Springer, 2023, pp. 387–396.

[19] R. Karanjai, E. Li, L. Xu, and W. Shi, "Who is smarter? an empirical study of ai-based smart contract creation," *arXiv preprint arXiv:2308.02955*, 2023.

[20] G. Ibba, M. Ortu, R. Tonelli, and G. Destefanis, "Leveraging chatgpt for automated smart contract repair: A preliminary exploration of gpt-3-based approaches," *Available at SSRN 4474678*.

[21] Y. Sun, D. Wu, Y. Xue, H. Liu, H. Wang, Z. Xu, X. Xie, and Y. Liu, "When gpt meets program analysis: Towards intelligent detection of smart contract logic vulnerabilities in gptscan," *arXiv preprint arXiv:2308.03314*, 2023.

[22] I. David, L. Zhou, K. Qin, D. Song, L. Cavallaro, and A. Gervais, "Do you still need a manual smart contract audit?" *arXiv preprint arXiv:2306.12338*, 2023.

[23] S. Hu, T. Huang, F. İlhan, S. F. Tekin, and L. Liu, "Large language model-powered smart contract vulnerability detection: New perspectives," *arXiv preprint arXiv:2310.01152*, 2023.

[24] C. Chen, J. Su, J. Chen, Y. Wang, T. Bi, Y. Wang, X. Lin, T. Chen, and Z. Zheng, "When chatgpt meets smart contract vulnerability detection: How far are we?" *arXiv preprint arXiv:2309.05520*, 2023.