Lanturn: Measuring Cryptoeconomic Smart Contract Security through Learning

Kushal Babel

Cornell Tech, IC3

SBC'23

with Mojan Javaheripi, Mahimna Kelkar, Yan Ji,
Farinaz Koushanfar, Ari Juels

Smart Contract Security

JESSE COGHLAN JUN 17, 2022

Inverse Finance exploited again for $1.2M in flash loan oracle attack

No user funds have been affected by the exploit, but Inverse Financ offered the attacker a bounty to return the stolen funds.

09 May 2022 00:53 GMT-7 · 2 min read

DeFi Lending Protocol Fortress Loses All Funds in Oracle Price Manipulation Attack

Business

Solana-Based Decentralized Finance Platform Hit by $100 Million Exploit

Mango's MNGO token was down over 40% after suffering from the latest mas decentralized finance exploit.

By Sam Kessler Oct 11, 2022 at 7:21 p.m. EDT Updated Oct 12, 2022 at 1:24 p.m. EDT

Tech

Solana DeFi Protocol Nirvana Drained of Liquidity After Flash Loan Exploit

The price of the protocol's ANA token fell almost 80% following the attack.

By Shaurya Malwa Jul 28, 2022 at 4:41 a.m. PDT Updated Jul 28, 2022 at 8:06 a.m. PDT

Complex Interactions

- Smart-Contracts can interoperate very easily.
- Examples:
  1. Lending contract can use Decentralized Exchange(DEX) contract as price oracle.
  2. Multiple DEX contracts can be aggregated together.
  3. FlashLoans + DEX
  4. FlashLoans + Lending Contract ...

MEV, Informally…

MEV = Maximal Extractable Value (or Miner Extractable Value)

Ability of miners/validators/bots to extract value by reordering, inserting or censoring transactions

***EV* =** Value extracted in a given transaction sequence

MEV = max(*EV* for *any* transaction sequence)

We can use MEV as the measure of Cryptoeconomic Security, as it models the worst case adverserial advantage.

Talk Outline

- Previous approaches
- Lanturn
  - Overview
  - Optimization Module
  - Simulation Module
- Evaluation
- Limitations
- Conclusion

Prior Approaches: Heuristics Based

- Most works encode a specific attack strategy, essentially looking for patterns in transaction data and blockchain state
- Highly efficient at finding patterns such as arbitrage, sandwich attacks, but…
- These approaches do not generalizable to new contracts or new transaction types
- As a result, these approaches do not attempt to find the worst-case adversarial advantage

- L. Zhou, K. Qin, C. F. Torres, D. V. Le and A. Gervais, "High-Frequency Trading on Decentralized On-Chain Exchanges" 2021 IEEE Symposium on Security and Privacy (SP).

Prior Approaches: Clockwork Finance

- Leverage formal verification to find the optimal value of the inserted transactions and optimal order of transactions that maximize MEV.

- Formal guarantees on the optimality of obtained MEV.

- Not scalable for complex contracts such as contracts with *loops* (Curve Finance), or large number of transactions (>10)

K. Babel, P. Daian, M. Kelkar and A. Juels, "Clockwork Finance : Automated Analysis of Economic Security in Smart Contracts" 2023 IEEE Symposium on Security and Privacy (SP).

Prior Approaches

Heuristics Based

Efficient but not generic

Formal Methods

Generic but not scalable

Lanturn

Generic and scalable learning-based tool to find MEV opportunities and understand the economic security risks of smart contracts and their composition

Lanturn Properties

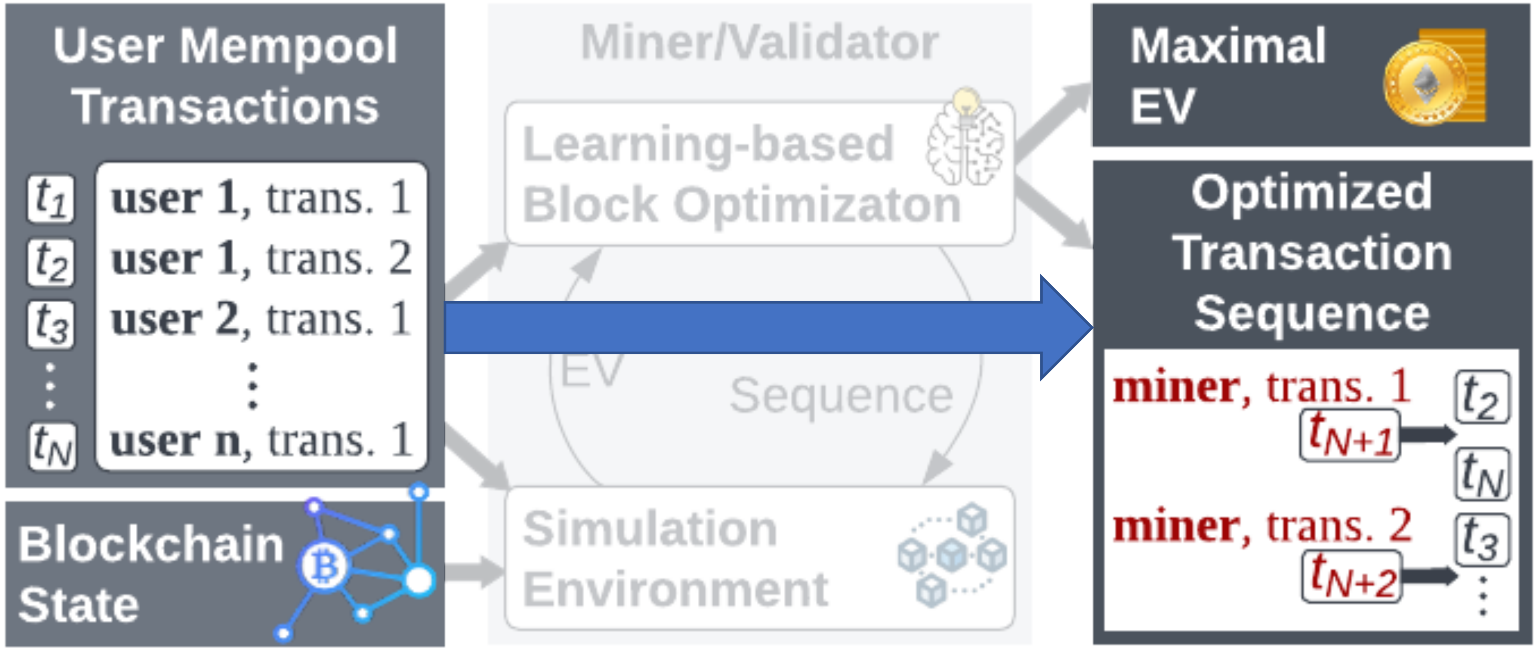
- **Generalizability** : No encoding of strategies and contract-specific heuristics, black box execution of smart contract, but requires templates for insertions based on the interface
- **Native Smart-Contract Execution** : Simulates smart contract bytecode, profit-yielding strategies are directly executable on-chain
- **Scalability** : Scales with both contract complexity and number of transactions (>50 transactions, >10 insertions)
- **Adaptability to computation budget** : Optimization algorithm can be tuned to match the computation budget and amenable to parallelization

Lanturn Applications

- **Developers and researchers** : Directly enables developers and researchers to understand the cryptoeconomics of smart contracts
- **Users**: Understand the value that can be extracted from their transactions
- **Strategic Agents** : Use Lanturn to extract value and discover new strategies. Real-time usage can be supported by parallelization across servers

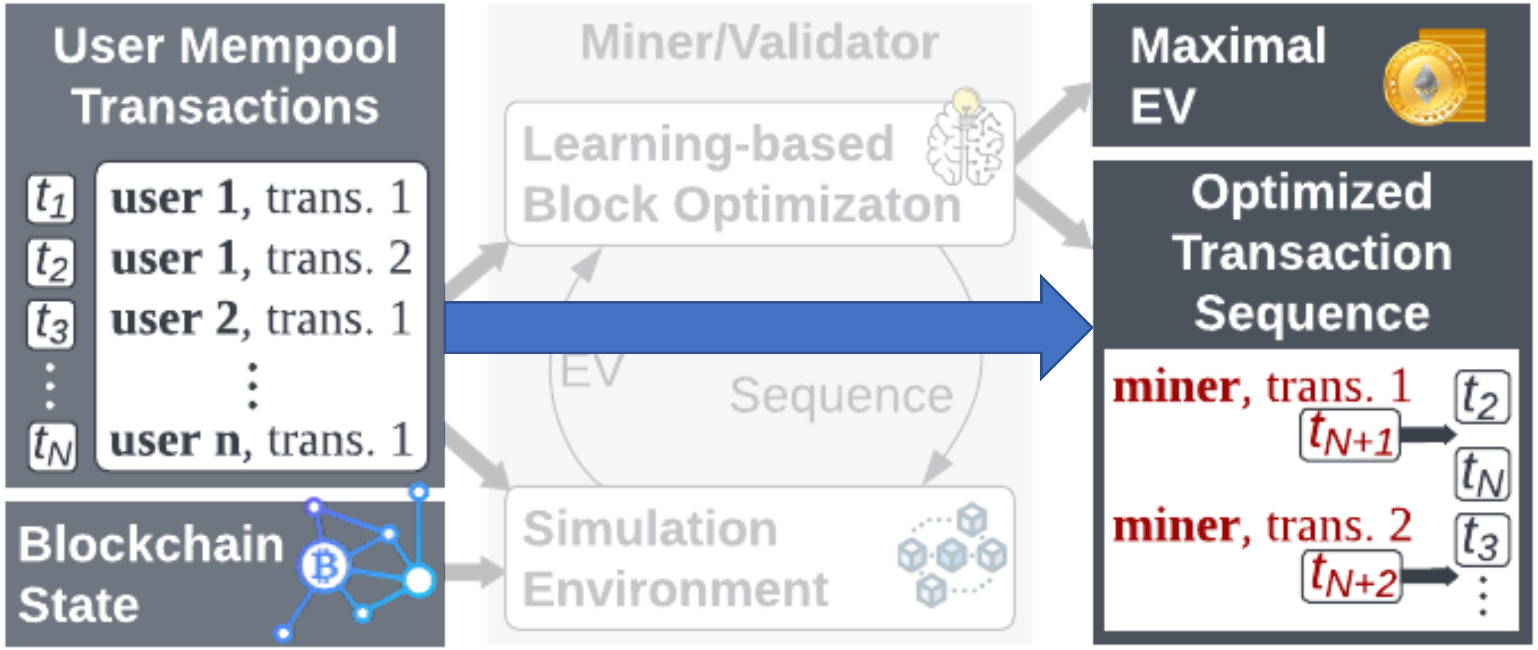
Problem Overview

- Given a pool of user transactions and the current blockchain state, Lanturn *automatically* learns to maximize the validator's EV.



Problem Overview

- The strategies include:
  - How to order the transactions in the block?
  - What transactions can the validator insert to take advantage of the block?



Problem Overview

- The strategies include:
  - How to order the transactions in the block?
  - What transactions can the validator insert to take advantage of the block?

- For the latter question, the validator transactions are selected from a set of templates like:

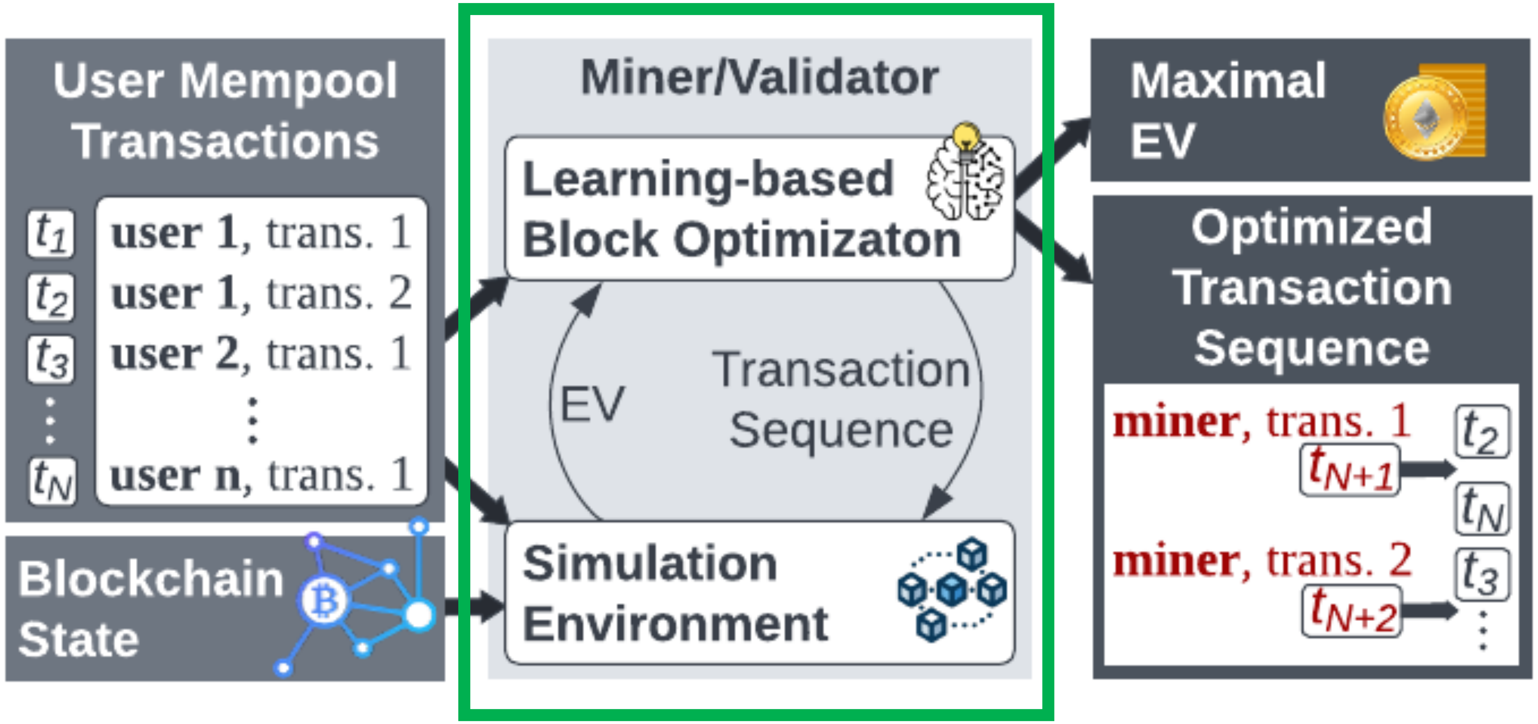
$$\boxed{\text{swap } \boldsymbol{\alpha_0} \times \texttt{token}_0 \text{ with } \boldsymbol{\alpha_1} \times \texttt{token}_1}$$

In this example,

- The tokens are filled with those that user transactions have interacted with.
- The alpha values (template-variables) are found by Lanturn such that they maximize EV.

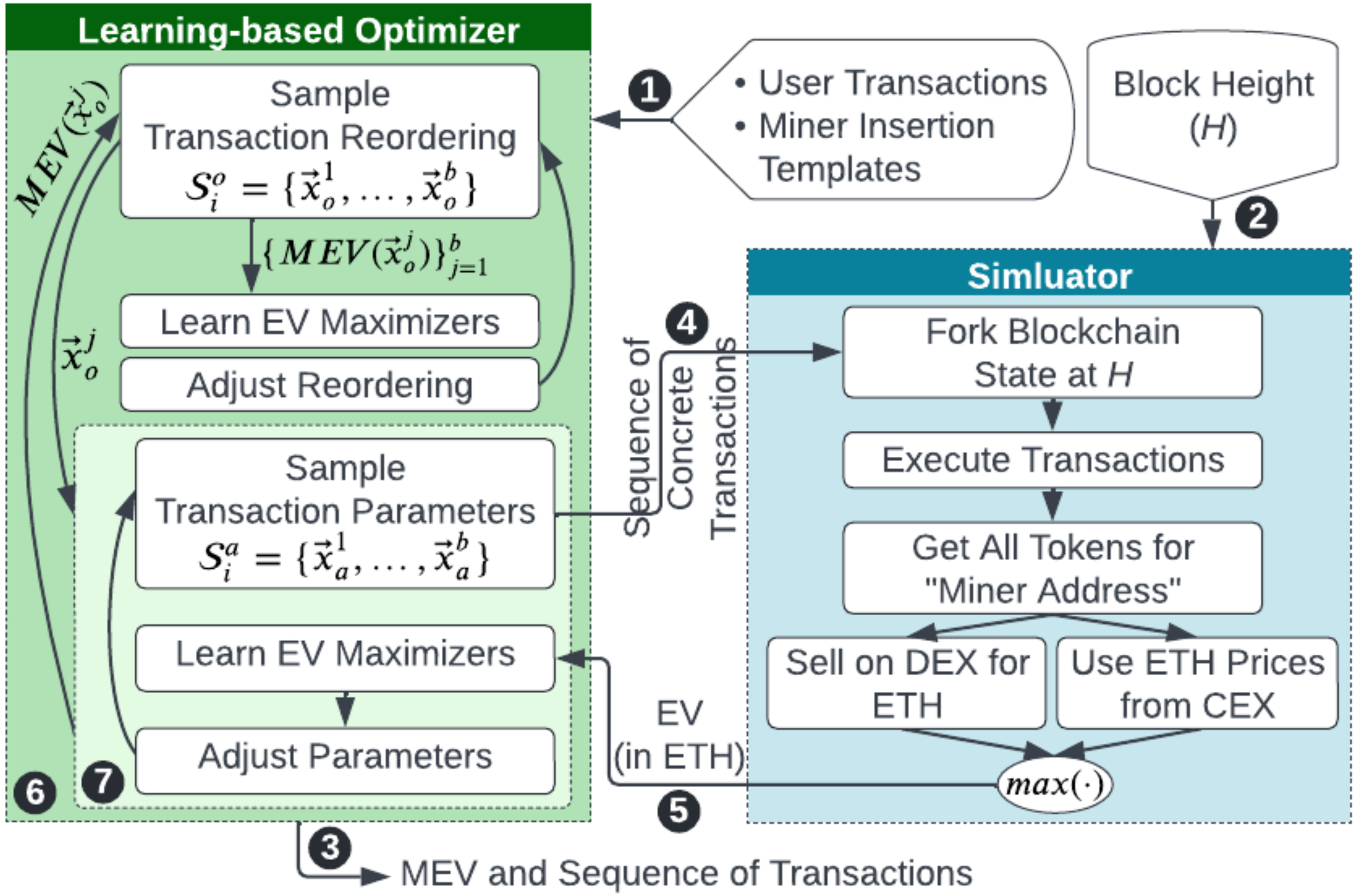
System Overview

- Learning is done through iterative interactions between our optimization module and simulator.



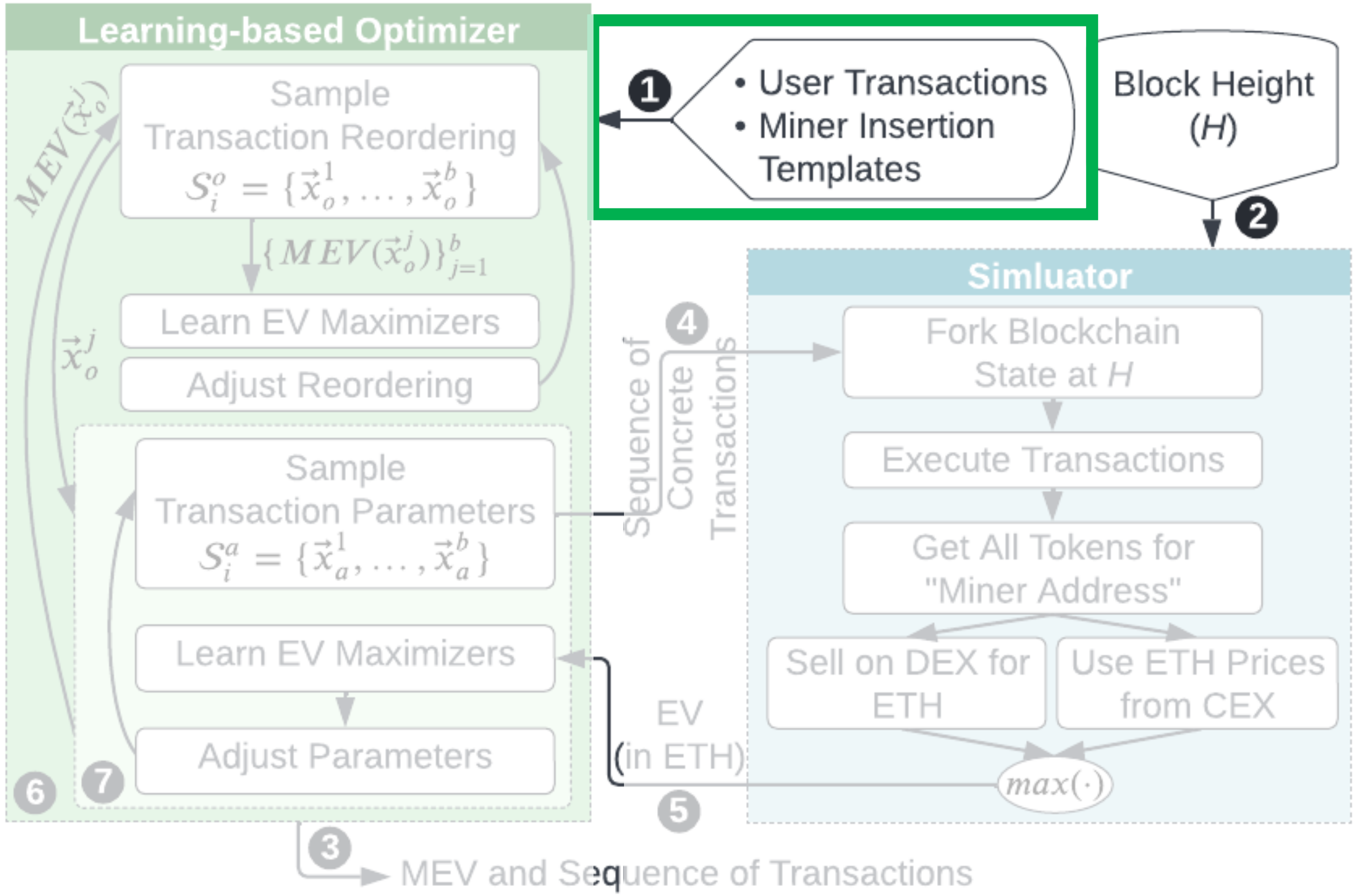
Deeper Look Inside

- High-level view of the internals of Lanturn components:



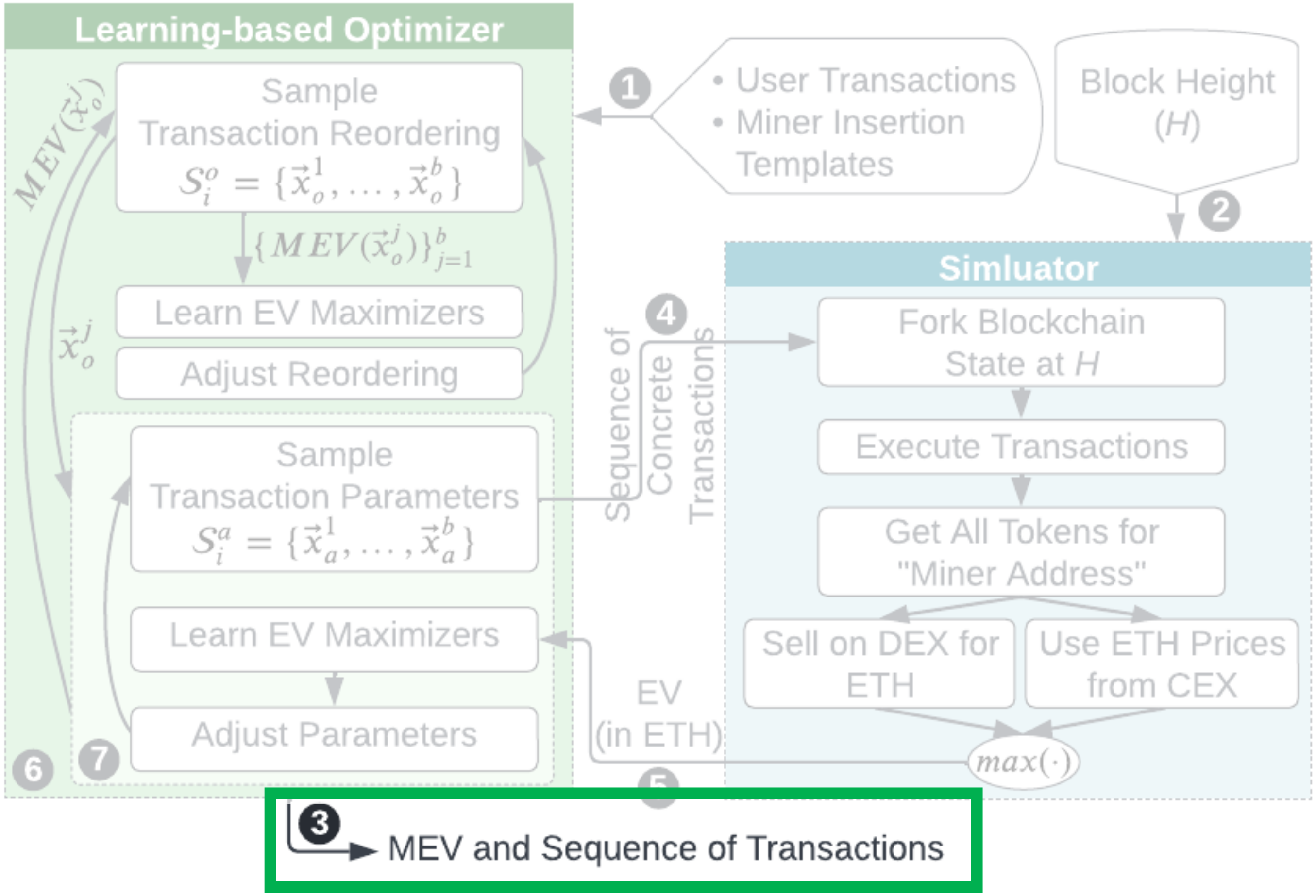
Deeper Look Inside

- Inputs to the optimizer (1):
  - User transactions to be ordered
  - Templates for validator-inserted transactions



Deeper Look Inside

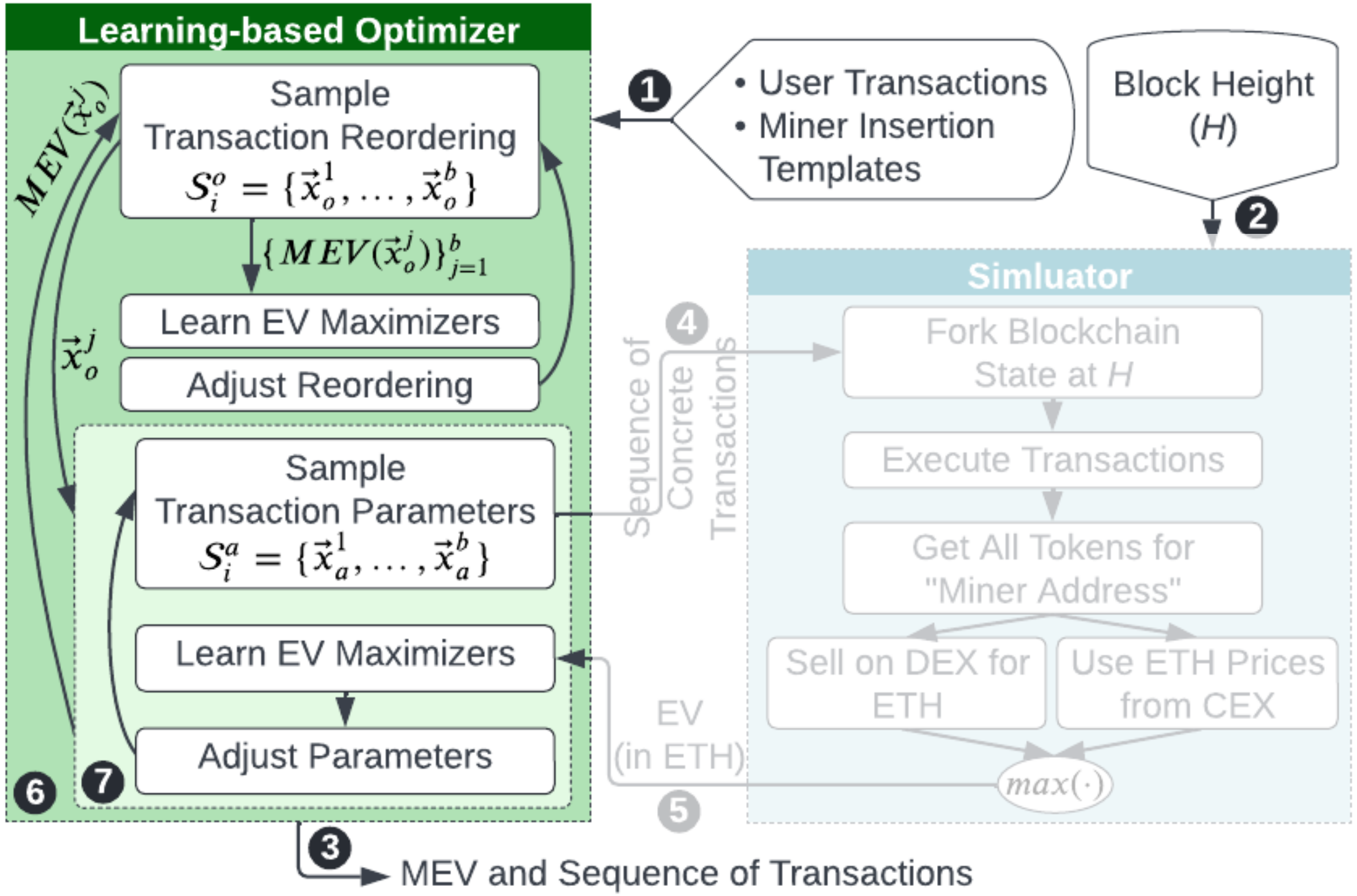
- Output of the optimizer (❸):
  - Maximal EV
  - Optimal transaction order



Deeper Look Inside

- Optimization has two hierarchical learning loops:

1. The outer loop learns the optimal order of transactions (❻).
2. The inner loop learns the optimal template-variables for the validator (❼).



Lanturn Learning-based Optimizer

- Problem Formulation: $$\max_{\vec{x} \in X} F(\vec{x})$$

1. Objective function (*F*): Validator's EV in terms of Eth
2. Design Variables ($\vec{x}$): Order of transactions (x_o), transaction amounts (x_a)

Bi-level optimization:

$$\max_{\vec{x_o} \in X_o} F(\vec{x_o})$$ Find the best transaction order

$$\text{where } F(\vec{x_o}) = \max_{\vec{x_a} \in X_a} f(\vec{x_o}, \vec{x_a})$$ Find the best values for template-variables

3. Optimization Bounds (X_o, X_a): Accepted range of values for each design variable, a.k.a, the search space

Lanturn Learning-based Optimizer

Goal: empirically search for optimal vector:

$$\vec{x}^* = \operatorname*{argmax}_{\vec{x} \in X} \boxed{F(\vec{x})}$$

What does optimality mean?

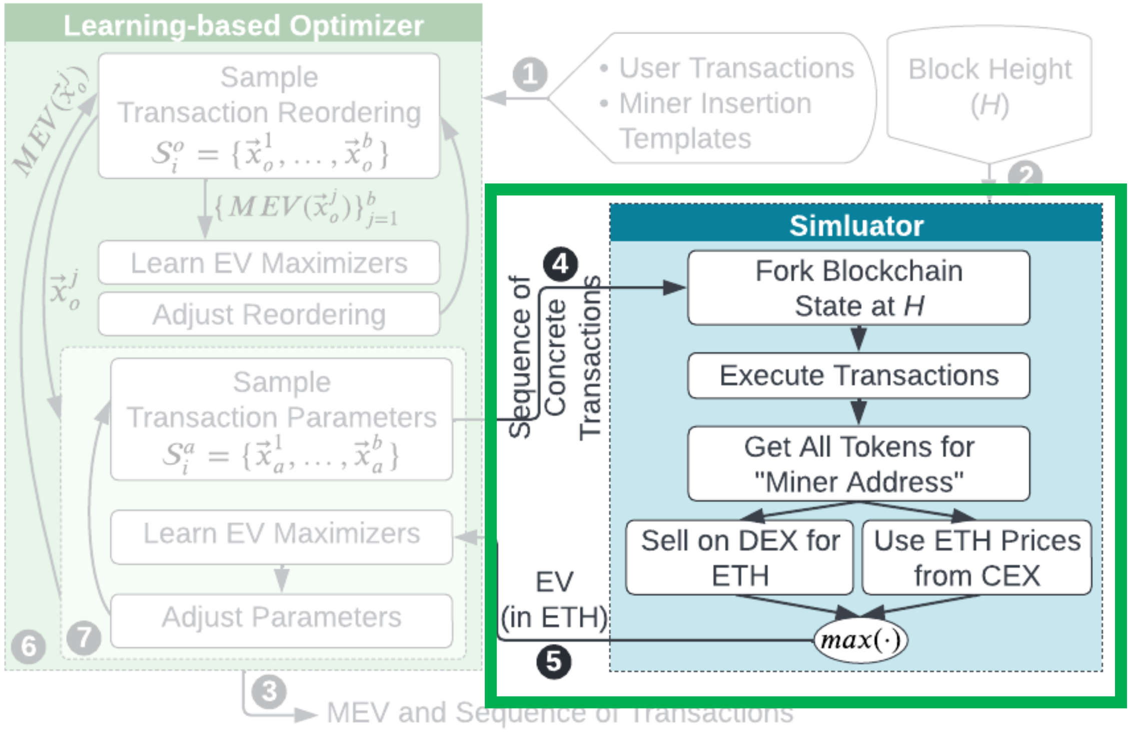
- High EV for the validator

Lanturn Learning-based Optimizer

Goal: empirically search for optimal hyperparameter vector:

$$\vec{x}^* = \underset{\vec{x} \in X}{\operatorname{argmax}}\ F(\vec{x})$$

What does empirical mean?
- No analytical solution
- Relies on accurate simulations of EV



Lanturn Learning-based Optimizer

Goal: empirically search for optimal hyperparameter vector:

$$\vec{x}^* = \underset{\vec{x} \in X}{\operatorname{argmax}} \; F(\vec{x})$$

Search-space is not small, even for the below toy problem with only 4 transactions, the search space covers almost 30 billion possibilities.

How to search the large space?

-Fast convergence

-Computational efficiency

-Parallelizable

User Transactions:

- User1 swaps 100 usdc for eth
- User2 swaps 4 eth for usdc

Template Symbolic Transactions:

- Validator adds α_0 liquidity units in price range α_1 to α_2
- validator swaps α_3 usdc for eth

Template-variables:

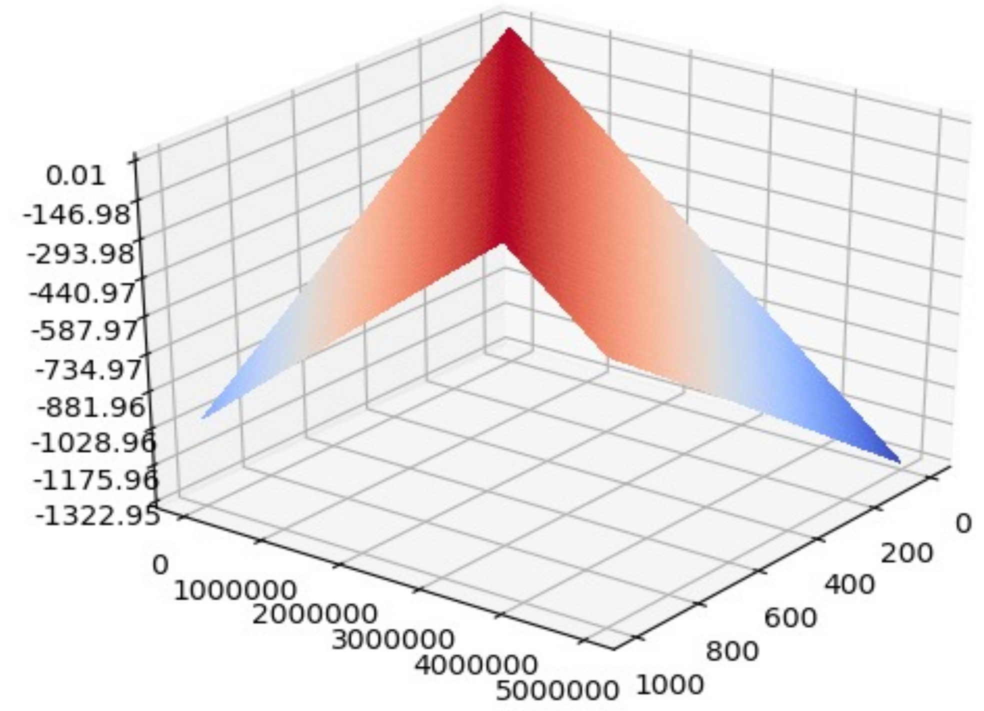
α_0 : `uint128`; α_1, α_2 : `int24`; α_3 : `uint256`

Constraints:

$\alpha_0 \in \{1, 10, 100\}$; $1000 < \alpha_1 < 1500$;
$2000 < \alpha_2 < 2500$; $400 < \alpha_3 < 2000$

Search-space Geometry

- Visualization for finding transaction amounts for an example problem with validator executing a "Just-in-time" liquidity strategy.
- **Search goal:** Identify the peak region.
- Challenges: In general -
  - Non-monotonic
  - Non-convex
  - High correlations between variables
  - High-dimensional space



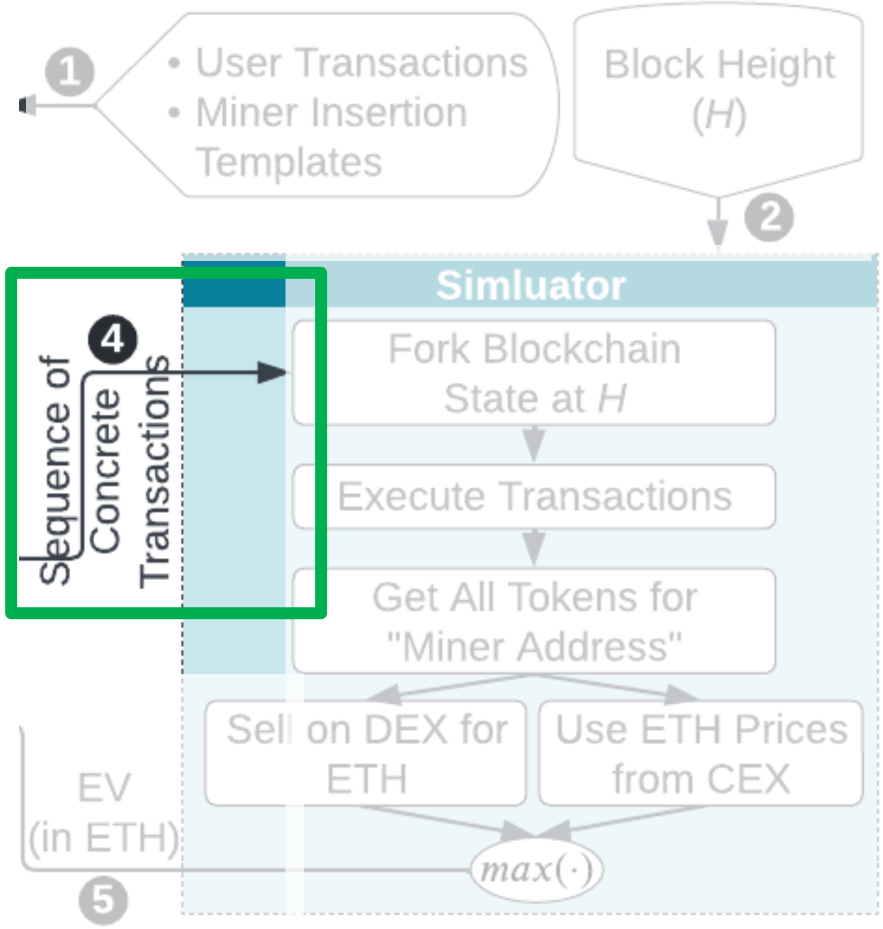
- horizontal plane: Transaction amounts
- vertical axis: objective function (EV)

Algorithm High Level Sketch

1. Initialize distribution D *to* Uniform
2. *res* = 0
3. For $i = 1$ *to* N:
4. $\quad S$ = Sample(D)
5. $\quad$ *res* = max(*res*, EV(S))
6. $\quad G$ = Good_Samples(S)
7. $\quad D$.update(G)
8. return *res*

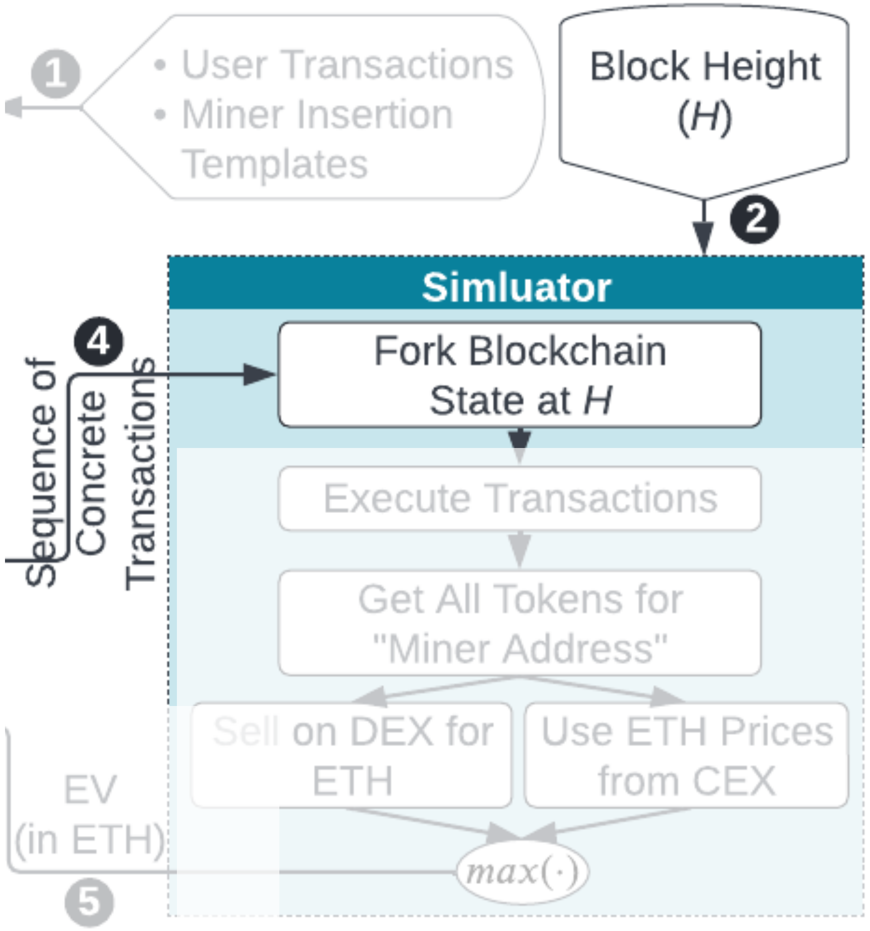
Simulation Module

1. Receive a concrete transaction sequence



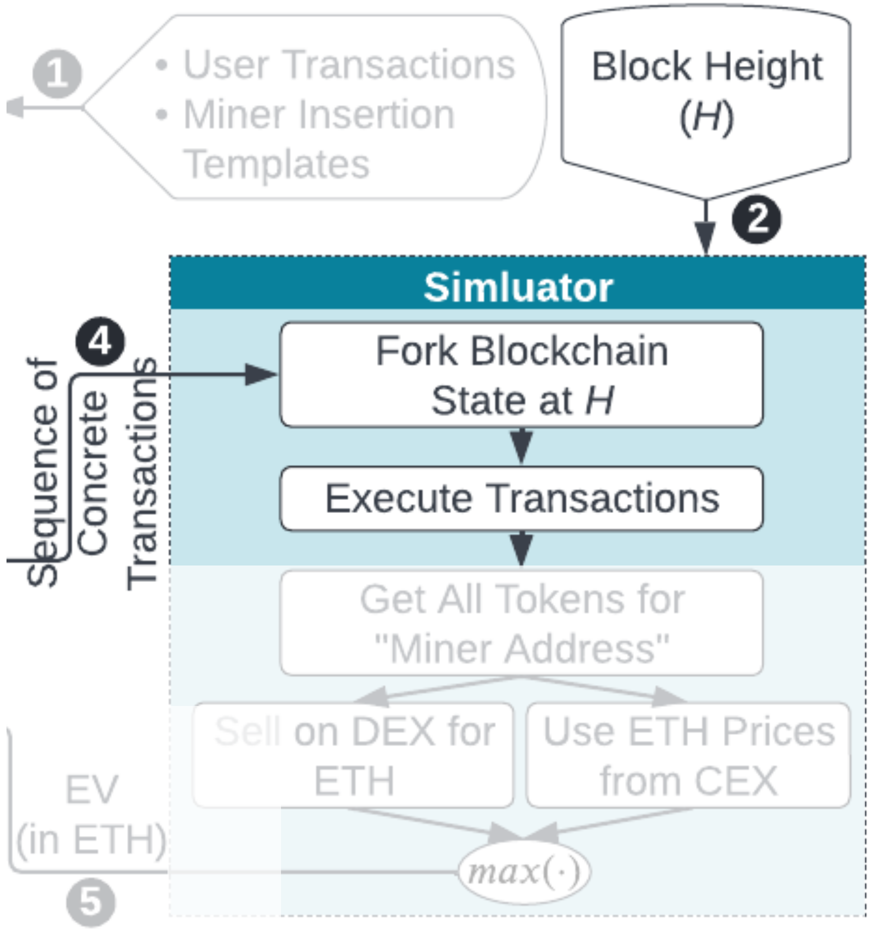
Simulation Module

1. Receive a concrete transaction sequence
2. Fork the blockchain state at height *H*
3. Give validator an initial capital (e.g. 10m Eth)



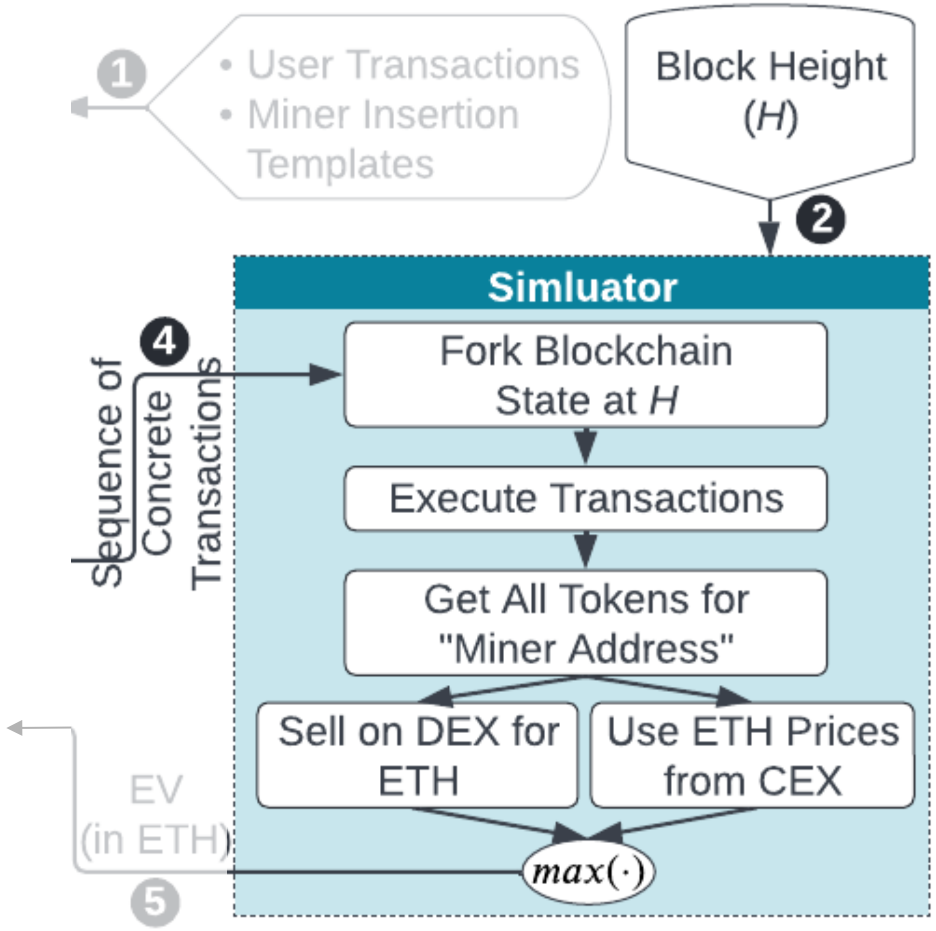
Simulation Module

1. Receive a concrete transaction sequence
2. Fork the blockchain state at height *H*
3. Give validator an initial capital (e.g. 10m Eth)
4. Execute transaction sequence



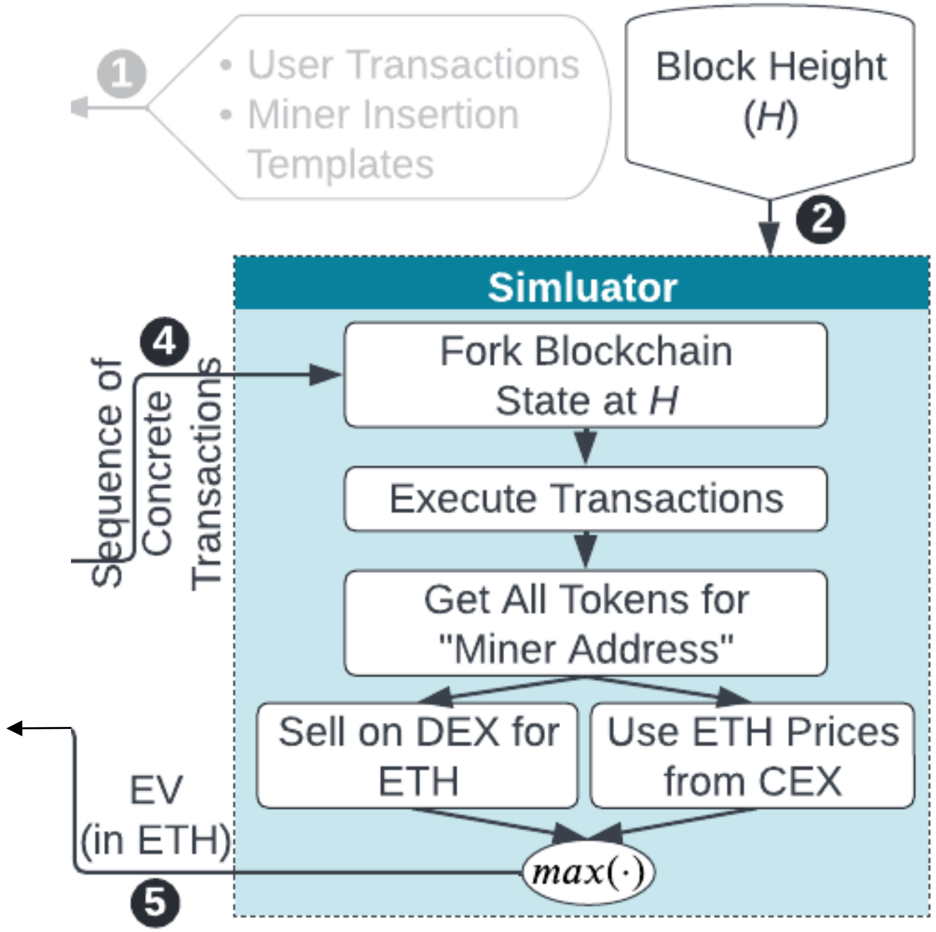
Simulation Module

1. Receive a concrete transaction sequence
2. Fork the blockchain state at height *H*
3. Give validator an initial capital (e.g. 10m Eth)
4. Execute transaction sequence
5. Convert any non-Eth tokens to Eth by executing DEX trades or using historical prices from CEX



Simulation Module

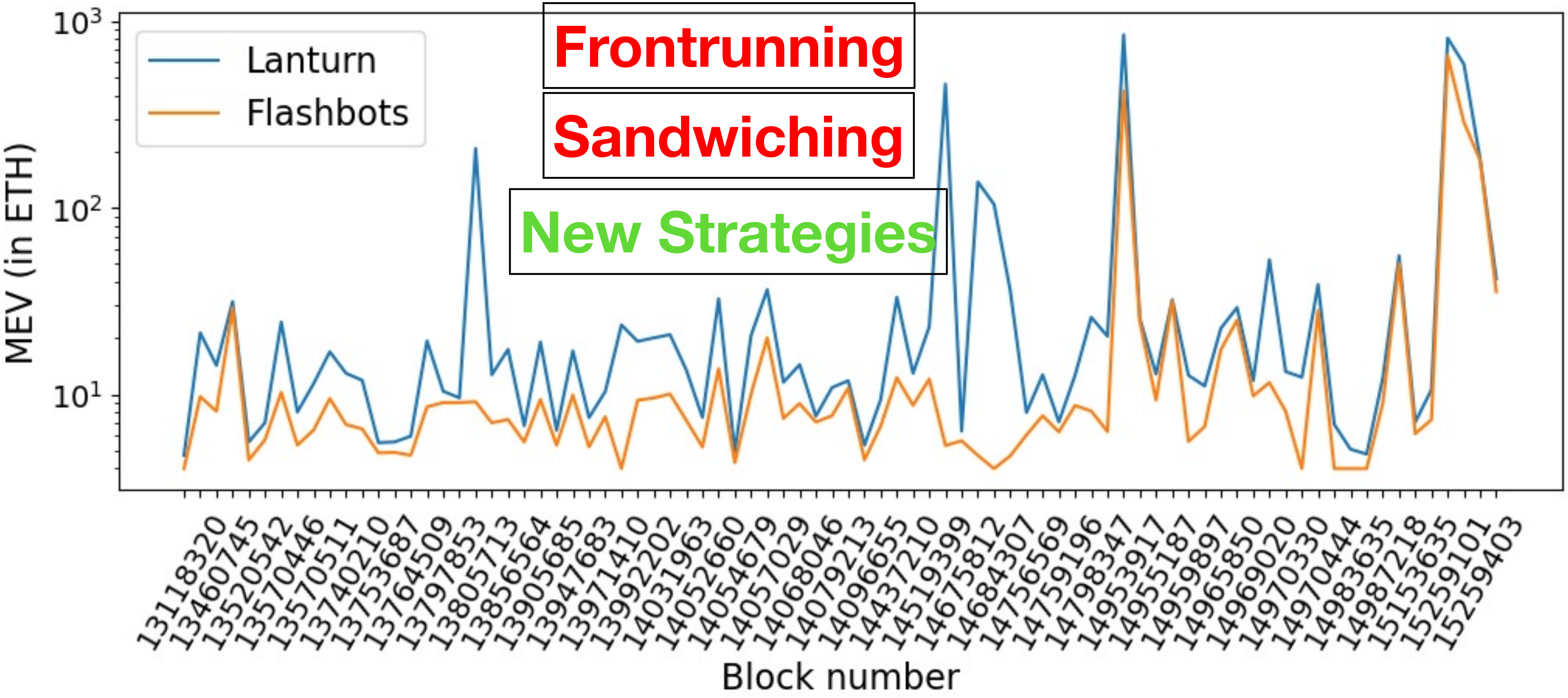
1. Receive a concrete transaction sequence
2. Fork the blockchain state at height *H*
3. Give validator an initial capital (e.g. 10m Eth)
4. Execute transaction sequence
5. Convert any non-Eth tokens to Eth by executing DEX trades or using historical prices from CEX
6. Mine block *H+1*
7. Return the validator's final Eth balance less initial balance.



Evaluation

- UniswapV3, UniswapV2, Sushiswap, Aave
- Dataset
  - Filter blocks that have more than 500 ETH trade on Sushiswap or UniswapV2, 1000 ETH on UniswapV3 or Liquidation event on Aave
  - CEX prices: Freely available historical minute-level price data (Binance)
  - Baseline: Flashbots data for MEV bribes paid to the validators per-bundle (pre-MEV-Boost), through gas fees and direct transfers
- Template Transactions: Symbolic transactions for swapping in either direction, liquidity provision and removal, liquidation on Aave

AMM Trading – UniswapV2



Lazarus Strategy

Bring a dead transaction back to life

Transaction Hash: 0x367e86b305d5a722b39a332ddfc203a4

Status: Fail

Block: 14954940 3073023 Block Confirmations

From: 0xbc1c16b50Ecf01bD1e4f6C2fE21887A67aC2eC33

To: 0x793FF66B5435A6De9D6d7e70C4C68522E8cD24E1 (MEV Bot: 0x793...4E

Warning! Error encountered during contract execution **[Reverted]**

Reorder, make it succeed, but then frontrun it!

Gas-leeching Strategy

- MEV Bots usually have safeguards in their transactions
- But some don't!
- Many transactions, which pay a high gas fees, run into an infinite loop when reordered, and end up consuming all the gas in the block --- transaction fees goes to the validator!

Transactions

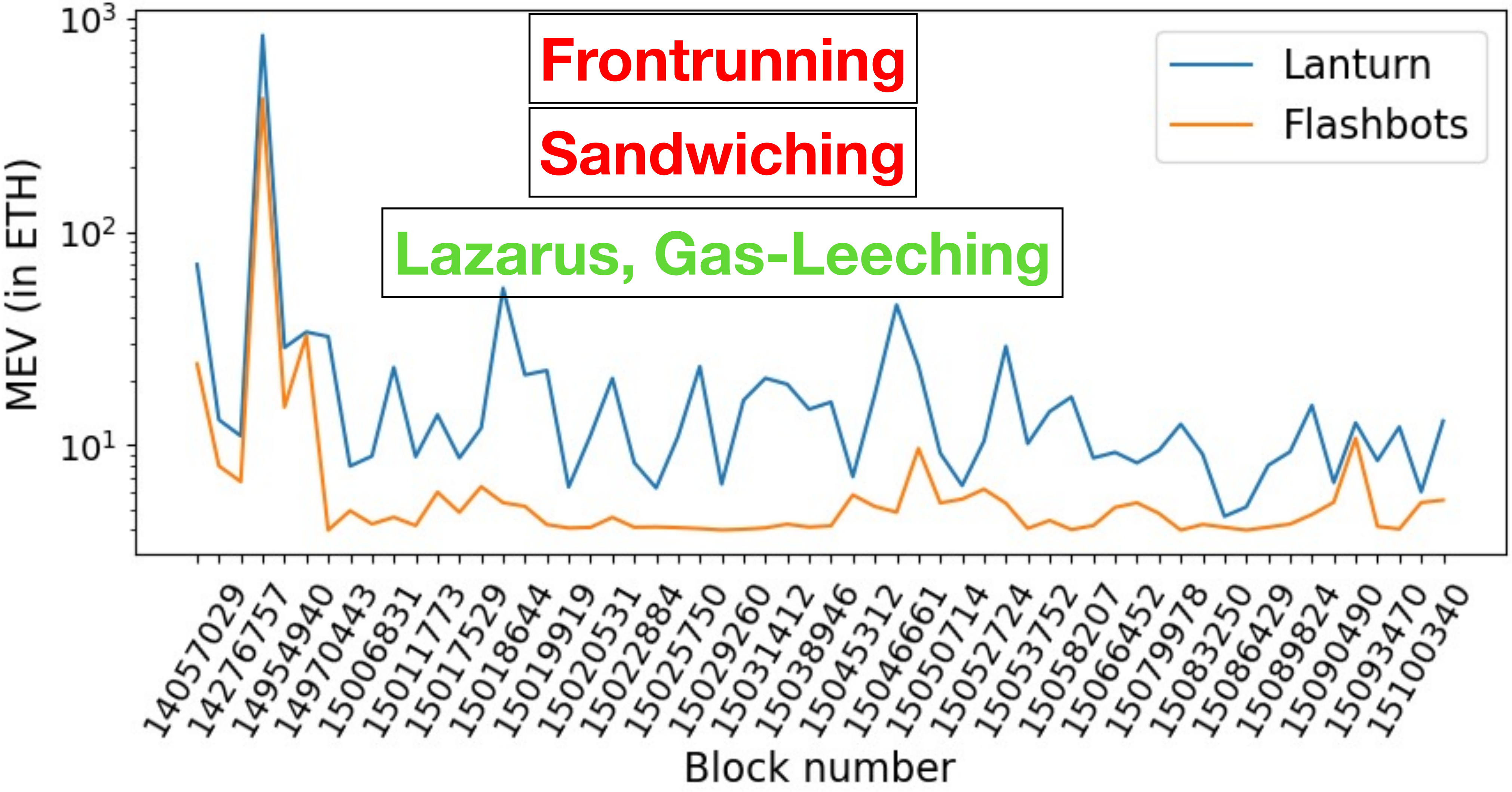
For 0x00000000003b3cc22af3ae1eac0440bcee416b40 MEV Bot: 0x000...B40

A total of 378,509 transactions found
(Showing the last 100k records)

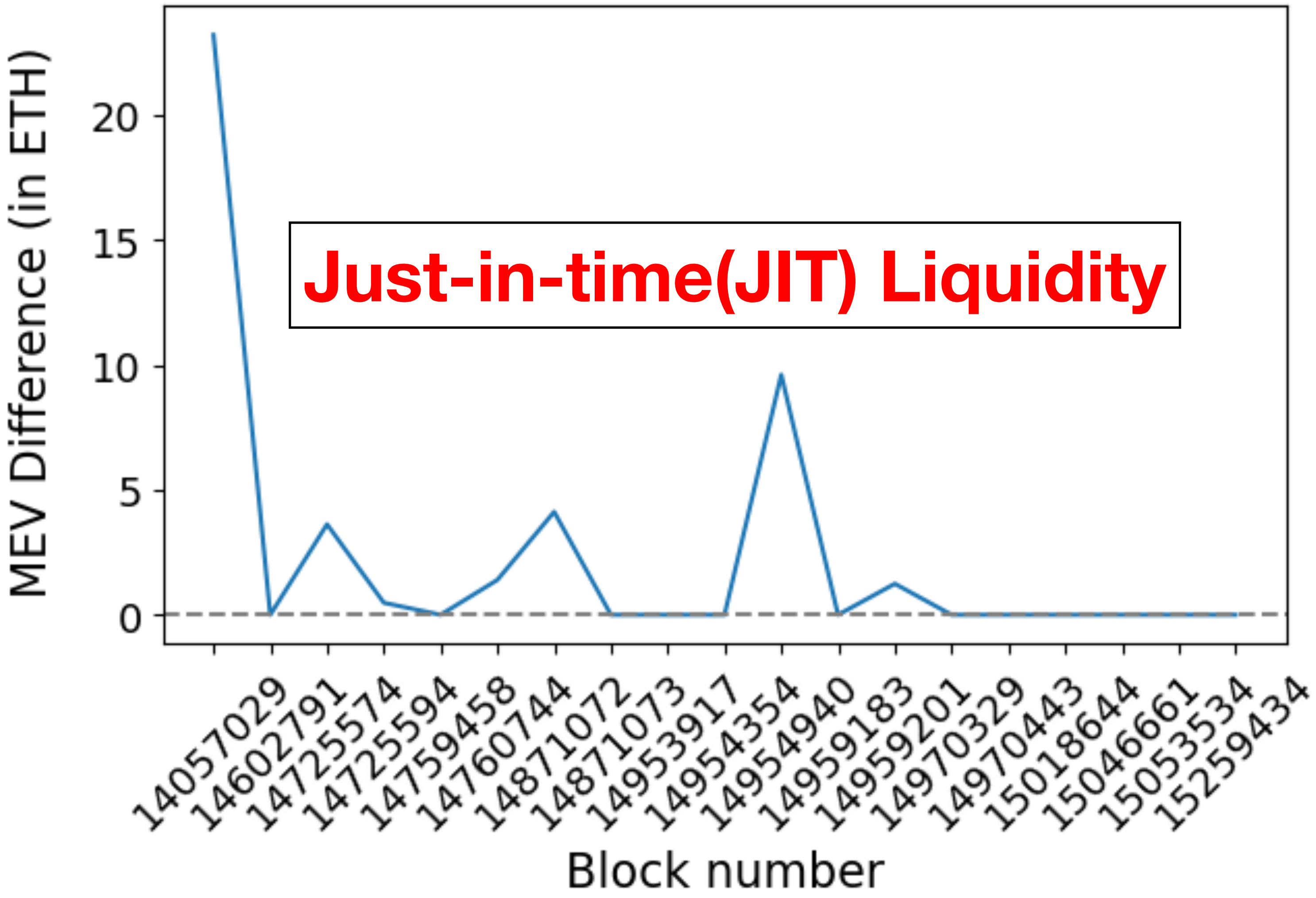
First

	Txn Hash	Method	Block	Age	From		To
	0xc9c02cd9bfc4dfb1a...	0x01f3cd17	18027974	7 mins ago	0xb7aB1D...F3798538	IN	MEV Bot: 0x000...B40
	0xcb0483d3a0a13de5...	0x00f3cd17	18027974	7 mins ago	0xb7aB1D...F3798538	IN	MEV Bot: 0x000...B40

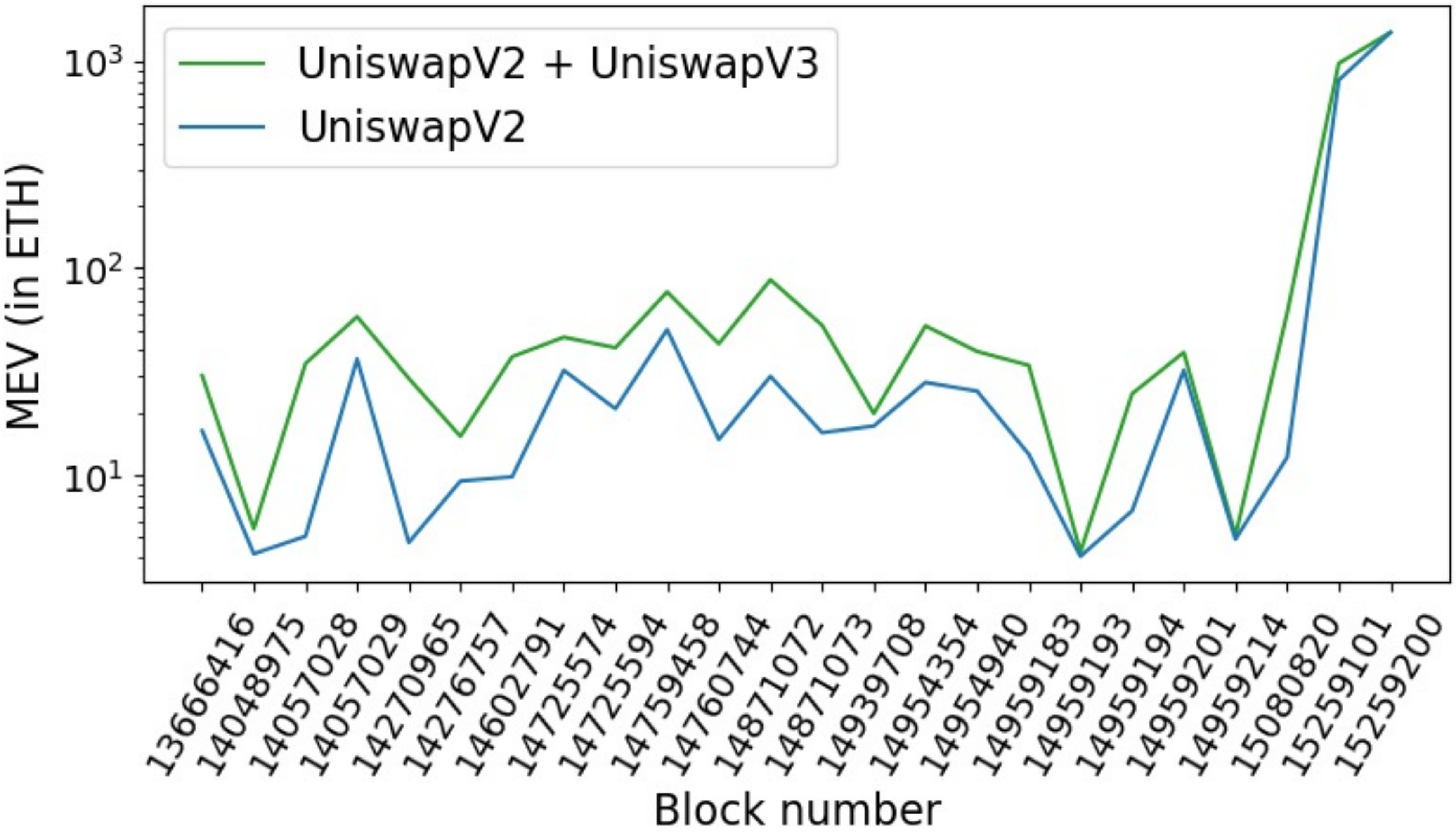
AMM Trading – UniswapV3



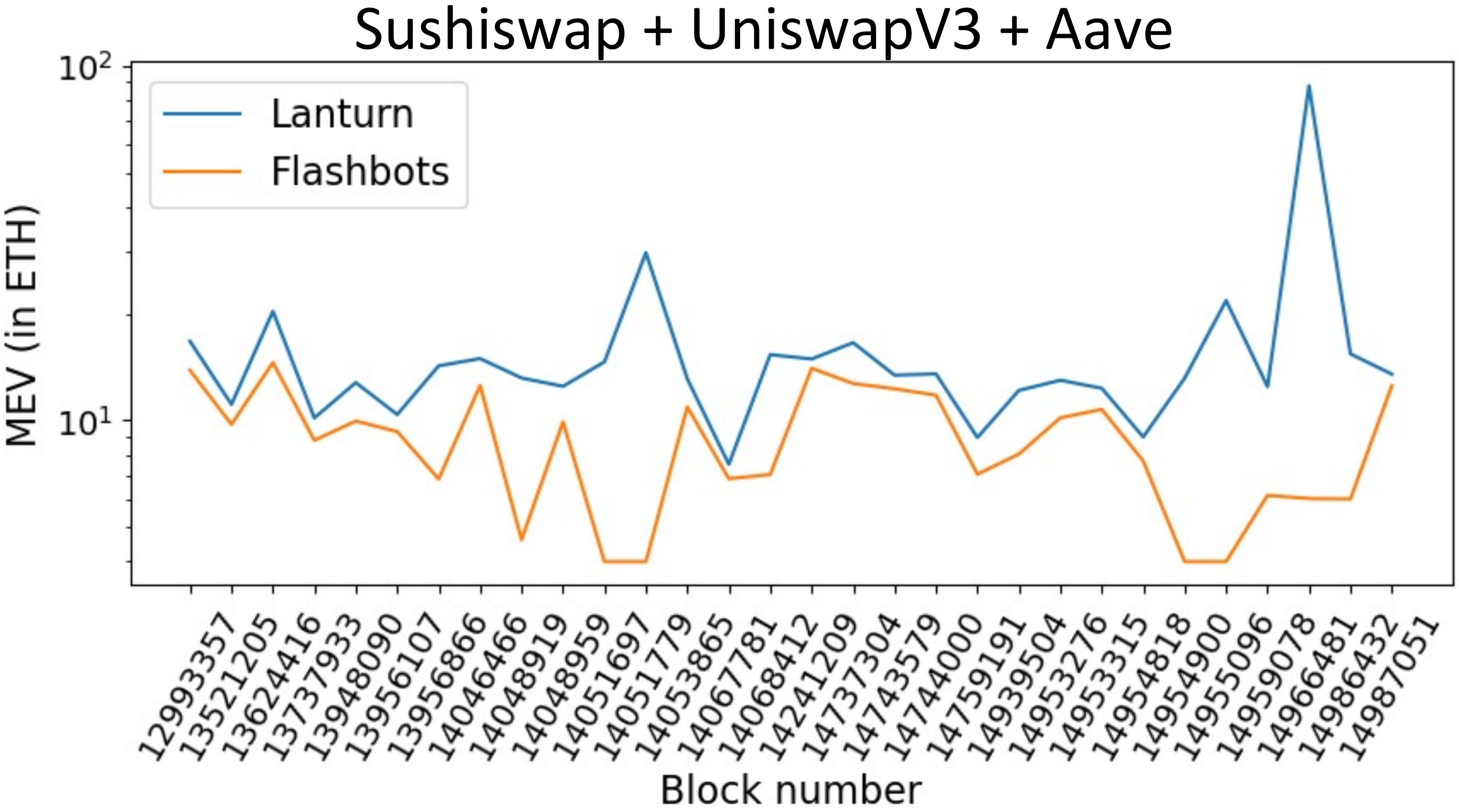
UniswapV3 Liquidity Provision



Multiple AMM Composition

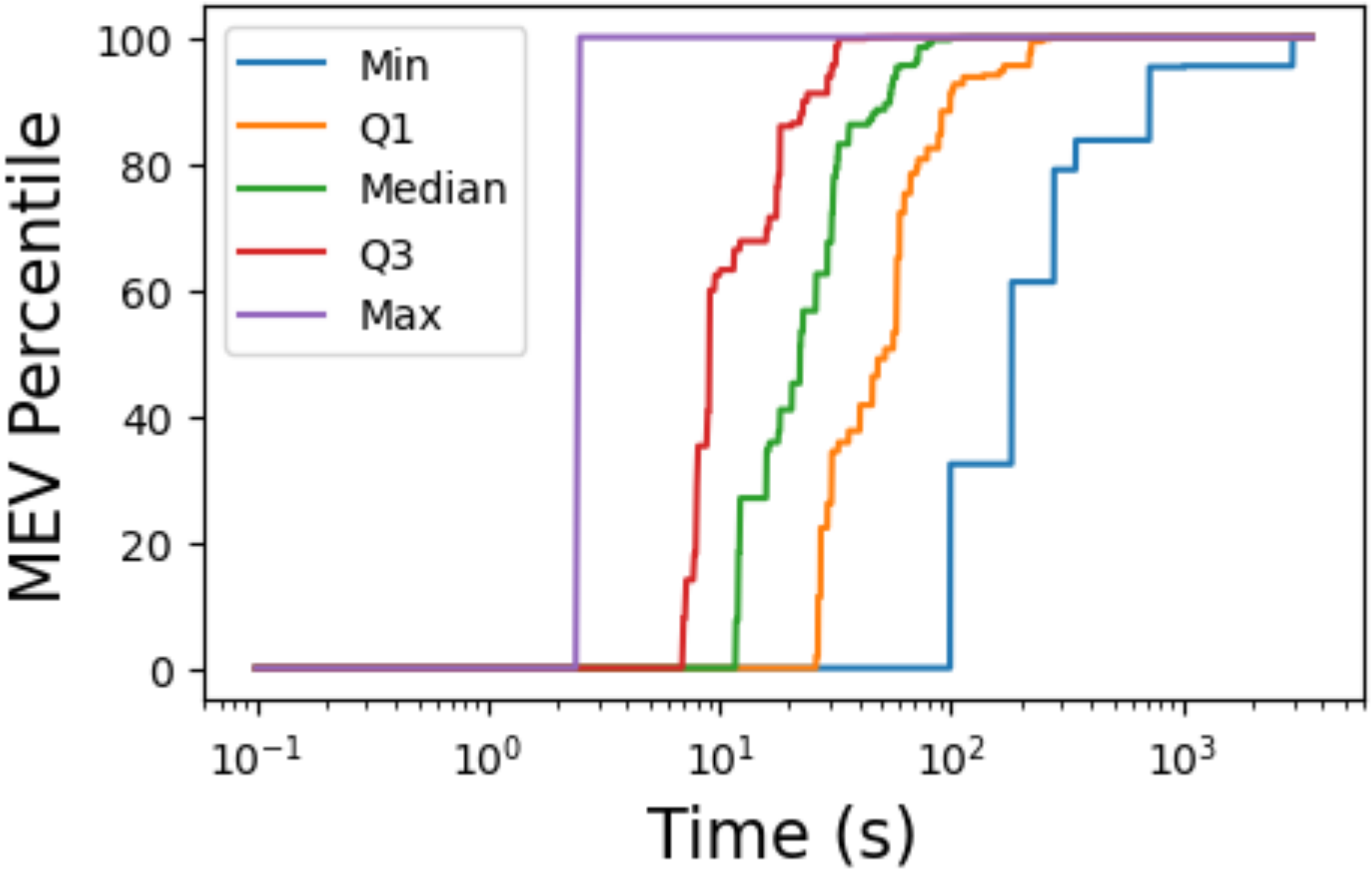


AMM+Lending Contract Composition



Execution Time

- Single Server
- AMD Ryzen Threadripper 3960X, 48 CPU threads, 128 GB RAM and SSD storage
- 44 parallel simulations



Limitations

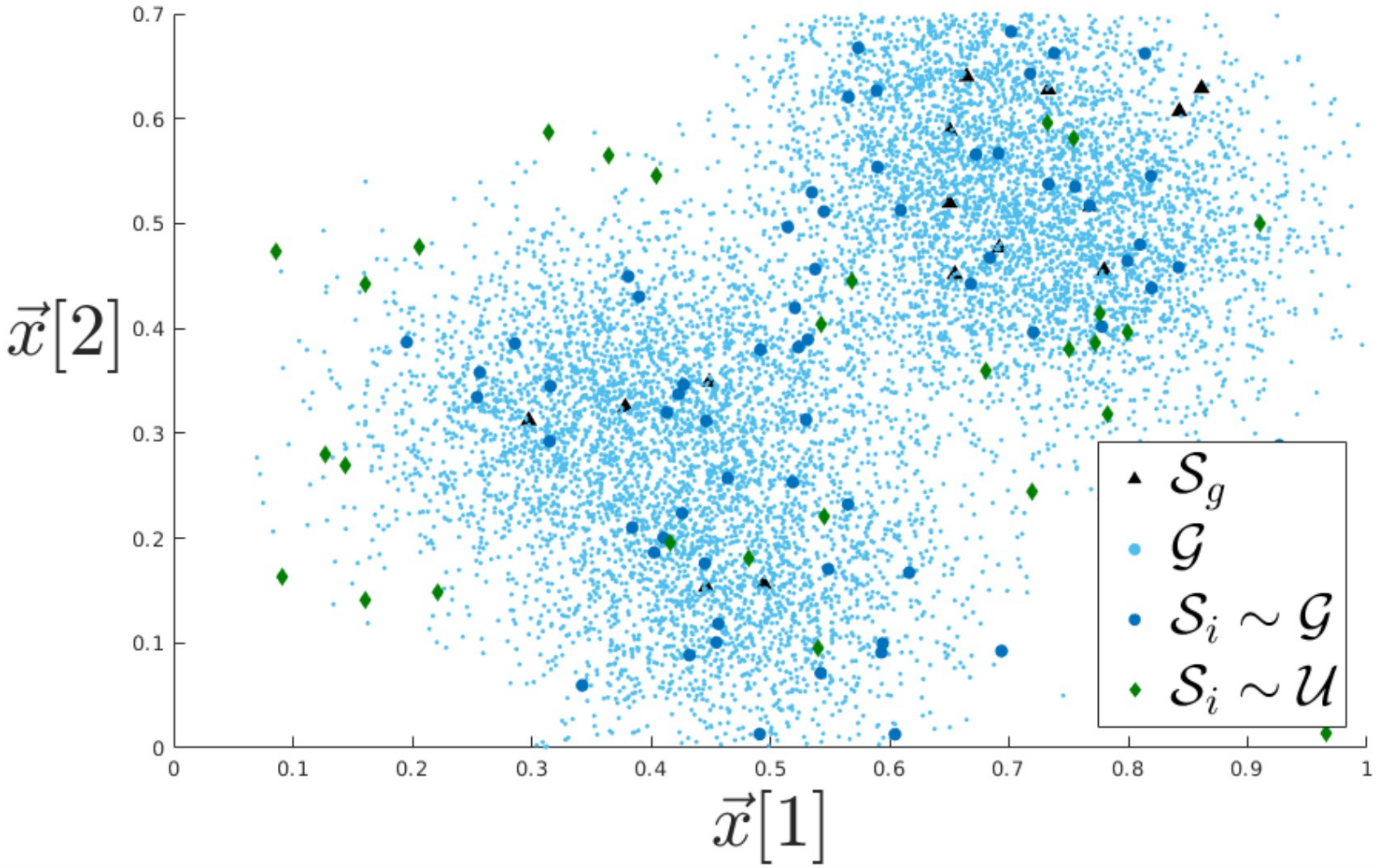
1. Mild regularity conditions needed for learning
2. Manual specification of templates

Conclusion

- Formulation of cryptoeconomic smart-contract security as a learning task.
- The approach can generalize to *any* smart-contract by treating smart-contract bytecode as black box simulation environment, output executable as is on-chain.
- **Lanturn** can scale to MEV strategies spanning over large (>50) number of transactions efficiently, and without modelling (complex) smart-contract behavior.
- **Lanturn** uncovers significant MEV by discovering not only well-known strategies but new strategies as well

Paper: www.cs.cornell.edu/~babel/papers/lanturn.pdf
Github: www.github.com/lanturn-defi/lanturn

Mutations – Gaussian+Uniform Distribution



Mutations - Permutations