Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# Secure Big Data Analytics for Blockchain Transaction Fee Prediction and Scheduling in Distributed Systems

**Kassem Danach**

kassem.danach@mu.edu.lb

Almaaref university

**Wael Hosny Fouad Aly**

American University of the Middle East

**Chadi Fouad Riman**

American University of the Middle East

**Research Article**

**Keywords:** Intelligent Big Data Analytics, Blockchain Technology, Distributed Systems, Secure Transaction Scheduling, Explainable AI, Hyper-Heuristic Optimization, Transaction Fee Prediction, Decentralized Systems, Smart Contracts

**Posted Date:** June 23rd, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-9781959/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# Secure Big Data Analytics for Blockchain Transaction Fee Prediction and Scheduling in Distributed Systems

Kassem Danach[1*†], Wael Hosny Fouad Aly[2†] and Chadi Fouad Riman[2†]

[1*]Basic and Applied Sciences Research Center, Al Maaref University, Beirut, 1001, Lebanon.
[2]College of Engineering and Technology, American University of the Middle East, Egaila, 54200, Kuwait.

*Corresponding author(s). E-mail(s): kassem.danach@mu.edu.lb;
Contributing authors: wael.aly@aum.edu.kw;
chadi.riman@aum.edu.kw;
†These authors contributed equally to this work.

**Abstract**

This paper introduces an Intelligent and Secure Big Data Analytics framework based on Explainable Hyper-Heuristic Optimization for dynamic blockchain transaction fee prediction and scheduling in distributed systems. The proposed framework is motivated by the volatility and complexity of large-scale blockchain fee markets, such as Ethereum's EIP-1559 mechanism, which generate high-dimensional, time-varying transactional data characteristic of modern distributed and IoT-enabled ecosystems. The framework integrates a machine learning–based fee prediction module with an adaptive hyper-heuristic optimization engine that solves a mixed-integer scheduling problem over large-scale blockchain transaction datasets. In addition, a SHAP-based explainability layer is incorporated to support transparent, interpretable, and auditable decision-making, enabling trustworthy analytics in secure distributed environments. Experimental evaluation using six months of Ethereum mainnet data demonstrates that the proposed approach reduces median transaction costs by up to 14.8% and improves average confirmation delays by 8.3% compared with heuristic and fixed-fee baselines.

Further analysis highlights the benefits of combining predictive analytics, adaptive heuristic selection, and exact refinement under dynamic network conditions, while explainability results identify key transactional and network features influencing fee behavior and scheduling decisions. Overall, the proposed framework advances intelligent and secure big data analytics for distributed systems by improving efficiency, transparency, and trust in decentralized transaction management, with applications in wallets, decentralized applications, and latency-sensitive financial services. Future work will extend the framework to multi-chain environments, integrate Layer-2 scaling solutions, and explore real-time deployment in production blockchain systems.

**Keywords:** Intelligent Big Data Analytics, Blockchain Technology, Distributed Systems, Secure Transaction Scheduling, Explainable AI, Hyper-Heuristic Optimization, Transaction Fee Prediction, Decentralized Systems, Smart Contracts

# 1 Introduction

## 1.1 Background and Motivation

Blockchain networks have rapidly become fundamental infrastructure supporting decentralized finance (DeFi), smart contracts, and a wide array of decentralized applications (DApps) Danach et al (2024). Recent developments position these networks as large-scale distributed data platforms that require intelligent analytics, robust scalability, and secure transaction processing in complex, multi-agent environments.

Among these platforms, Ethereum stands out as the leading environment for smart contract execution, notably implementing the EIP-1559 transaction fee mechanism to improve the efficiency and predictability of fee markets. This mechanism introduces a dynamically adjusted *base fee* for each block, alongside an optional *priority tip* designed to incentivize validators Leonardos et al (2023); Madrigal-Cianci et al (2025).

EIP-1559 replaced the previous first-price auction model, which often resulted in volatile fees and bidding wars during periods of network congestion Roughgarden (2024); Liu et al (2022). By algorithmically adjusting the base fee in response to block utilization, EIP-1559 aims to stabilize transaction costs, enhance user experience, and promote economic fairness Leonardos et al (2023); Zhivkov and Chen (2026). Furthermore, the base fee is burned instead of being rewarded to validators, introducing deflationary pressure on the Ethereum ecosystem and affecting broader asset economics Madrigal-Cianci et al (2025).

Despite these advantages, EIP-1559 introduces new complexities for users and automated agents, who must now navigate a volatile and stochastic fee market. Variations in transaction arrival rates, changing block gas limits,

and sudden demand surges significantly increase the difficulty of accurate fee estimation. Underestimating fees can lead to excessive delays or transaction failures, while overestimating results in unnecessary cost inflation Hossain and Altarawneh (2025).

These challenges carry significant economic implications for high-frequency traders, automated market makers, DeFi protocols, and other participants who rely on timely and cost-efficient transaction execution. Hence, there is a critical need to develop intelligent and adaptive decision-support systems that simultaneously predict fee dynamics and optimize transaction scheduling. Recent trends favor data-driven optimization methods, including reinforcement learning and deep predictive models, to address these challenges in complex distributed environments Jang and Shim (2026).

These insights motivate our work toward an explainable, hyper-heuristic optimization framework designed to address these challenges by dynamically forecasting transaction fees and enhancing scheduling efficacy in intelligent and secure distributed systems.

## 1.2 Problem Statement

Existing blockchain fee optimization approaches typically fall into two categories: (i) predictive models that estimate future fees but do not directly guide transaction scheduling, and (ii) heuristic scheduling strategies that select transactions based on simple cost or time thresholds without leveraging advanced predictive intelligence. While these methods are effective under stable conditions, their performance often degrades significantly under severe network congestion and dynamic workload variation.

Moreover, most current methods lack explainability, which is a crucial requirement in financial and distributed systems where trust, auditability, and regulatory compliance are essential. Implementing transparent decision-making layers is vital for establishing secure and accountable distributed analytics systems.

There is thus a need for a unified, transparent framework that integrates accurate prediction, adaptive scheduling, and interpretable decision support for intelligent big data analytics in blockchain-enabled distributed systems.

## 1.3 Contributions

To address these challenges, this paper proposes an *Explainable Hyper-Heuristic Optimization Framework* for dynamic blockchain transaction fee prediction and scheduling. The main contributions are as follows:

- **Integration of Fee Prediction and Scheduling:** We develop a machine learning–based fee forecasting module trained on Ethereum mainnet and mempool data to predict dynamic base fees and priority tips under varying network conditions Hossain and Altarawneh (2025).
- **Hyper-Heuristic Optimization for Scheduling:** We formulate the transaction scheduling problem as a mixed-integer optimization model and

solve it using an adaptive hyper-heuristic framework capable of dynamically selecting and combining heuristics for cost-delay trade-offs.
- **Explainability Layer:** We employ SHAP-based explainability to provide interpretable insights into both fee prediction and heuristic selection decisions, improving transparency and auditability in distributed systems.
- **Comprehensive Evaluation:** We validate the proposed framework on real Ethereum mainnet data, demonstrating up to a 15% reduction in median transaction costs while maintaining or improving confirmation times compared to baseline methods Leonardos et al (2023); Zhivkov and Chen (2026).

### 1.4 Paper Organization

The remainder of this paper is organized as follows. Section 2 reviews related work in blockchain fee optimization, hyper-heuristics, and explainable AI for distributed systems. Section 3 formulates the dynamic transaction scheduling problem. Section 4 presents the proposed intelligent and secure big data analytics framework. Section 5 describes the experimental setup and datasets. Section 6 presents performance evaluation and analysis. Section 7 discusses threats to validity. Finally, Section 8 concludes the paper and outlines future research directions.

## 2 Related Work

Blockchain transaction fee prediction and scheduling have attracted significant research interest due to their crucial role in ensuring efficient network operation and cost-effective transaction processing. Transaction fees in blockchains vary dynamically based on network demand, transaction size, and congestion, necessitating advanced optimization frameworks for effective fee prediction and scheduling Nadcab and Smith (2025); Wang and Patel (2022); Kim et al (2023).

### 2.1 Blockchain Fee Prediction

Transaction fee prediction in blockchain systems is critical for optimizing user costs and ensuring timely transaction processing. Early works focused on static and flat fee models that often led to inefficiencies under varying network conditions Nadcab and Smith (2025); Doe and Roe (2023). Modern blockchain systems adopt dynamic fee mechanisms where fees adjust based on real-time network congestion and transaction complexity. For example, Bitcoin fees are based on transaction size in bytes, with miners prioritizing transactions offering higher fees during congestion Nadcab and Smith (2025). Ethereum employs a gas fee mechanism dependent on computational resources; EIP-1559 introduced a base fee that adjusts automatically with network demand plus an optional priority tip Nadcab and Smith (2025); Smith and Nguyen (2023).

Recent studies explore sophisticated modeling approaches to forecast blockchain fees dynamically by considering network conditions such as mempool size, transaction volume, and congestion. Ma and Mahmoudinia Ma and Mahmoudinia (2021) provide a comprehensive evaluation of predictive models for Bitcoin fee forecasting over 24-hour horizons, demonstrating that traditional statistical models like SARIMAX often outperform complex deep learning architectures given relatively limited historical data. Other works utilize hybrid models combining LSTM variants, convolutional neural networks (CNN), and attention mechanisms to capture temporal and frequency-based patterns in Ethereum gas prices, especially after significant protocol upgrades such as the London Hard Fork Chen and Johnson (2024). Optimization techniques using Monte Carlo simulations have also been developed to predict the minimum transaction fee required to attain desired confirmation probabilities within fixed time limits Gonzalez and Patel (2024).

These approaches underscore the multi-faceted and evolving nature of fee prediction challenges, emphasizing the importance of adaptive models integrating historical data patterns with real-time network states to enhance forecasting accuracy and practical usability Wang and Patel (2022); Ma and Mahmoudinia (2024).

## 2.2 Blockchain Transaction Scheduling

Efficient scheduling of transactions is vital to maximize blockchain throughput and minimize confirmation times while addressing resource constraints and conflict avoidance. Concurrency-aware scheduling models, such as speculative concurrency introduced by Saraph and Herlihy Saraph and Herlihy (2023), improve validator efficiency but face challenges with high conflict rates that trigger costly re-executions.

More recent advancements adopt deterministic scheduling frameworks employing constraint programming and heuristic algorithms to balance execution performance, consistency, and power consumption, particularly on multi-core processors Saraph and Herlihy (2023); Chen and Johnson (2024). Queueing theory and mathematical optimization further contribute foundational insights to transaction queuing dynamics and fee mechanism impacts, fostering robust and scalable scheduling strategies adaptable to decentralized blockchains Chen and Gupta (2025).

These techniques collectively aim to refine transaction throughput and resource utilization under dynamic network conditions, with proactive conflict detection methods gaining preference over reactive conflict resolutions.

## 2.3 Hyper-Heuristics in Scheduling and Optimization

Hyper-heuristics, which dynamically select or generate heuristics to solve complex optimization problems, offer promising adaptability and explainability in scheduling. Unlike traditional heuristics designed for specific problem instances, hyper-heuristics seek generalized methods that remain effective

across diverse problem domains Danach and Aly (2025); Danach et al (2025b,a).

Recent research applies hyper-heuristic frameworks to classic scheduling challenges, such as job-shop scheduling and cloud resource allocation, demonstrating competitive performance and enhanced robustness Lee and Kim (2024). These frameworks typically combine multiple low-level heuristics and employ machine learning for adaptive heuristic selection based on solution quality and search progression.

The application of hyper-heuristics to multifactor, multi-objective blockchain transaction scheduling and fee optimization is an emerging frontier, with few detailed investigations specific to blockchain contexts to date.

## 2.4 Explainable AI in Optimization

Explainable AI (XAI) in optimization is crucial for transparency, trust, regulatory compliance, and facilitating human understanding in complex automated decision systems, such as blockchain transaction fee prediction and scheduling. XAI techniques include feature importance analysis, rule extraction, and model-agnostic explanation tools that elucidate the reasoning behind model predictions or optimization decisions.

Within scheduling and fee optimization, explainability enables stakeholders to comprehend why specific choices are made, ensuring smoother integration with human oversight. Hybrid models combining optimization with interpretable surrogate models or visualization tools have been proposed to maintain performance while offering increased transparency Green and Thompson (2023).

The intersection of XAI with hyper-heuristics and blockchain technologies holds significant promise for frameworks that are not only efficient and adaptive but also transparent and accountable, aligning technological advances with user-centric and governance demands.

## 2.5 Broader Impacts and Emerging Trends

Beyond Bitcoin and Ethereum, blockchain adoption studies indicate substantial transaction cost reductions and operational efficiency gains in sectors such as banking, highlighting the practical importance of effective fee prediction and scheduling mechanisms in enterprise and decentralized finance applications Al-Sarayreh and Alabdulmohsin (2025).

Additionally, research into redesigning transaction fee markets to support parallel execution under multi-threaded blockchain architectures reflects the ongoing evolution of scheduling and fee mechanisms to meet scalability demands Chen and Johnson (2024).

In summary, existing literature converges on the need for adaptive, explainable, and efficient frameworks capable of dynamic transaction fee prediction and optimized scheduling. Such frameworks must balance network resource

usage, user costs, and throughput while being robust to volatility in transaction demand and computational complexity. This research proposes a novel hyper-heuristic optimization framework designed to address these challenges with improved explainability and operational performance.

# 3 Problem Formulation

## 3.1 System Overview

We consider a blockchain network employing a dynamic fee market mechanism similar to Ethereum's EIP-1559 protocol, where each block has a *base fee* $b_t$ and an optional *priority tip* $p_{i,t}$ for each transaction $i$ at time slot $t$. The network maintains a mempool $\mathcal{M}_t$ containing pending transactions, each with attributes including gas limit $g_i$, maximum fee per gas, maximum priority fee per gas, and a transaction deadline $d_i$. Blocks have a maximum gas capacity $G_{\max}$, and transactions are selected to maximize utility while respecting this capacity.

The proposed framework jointly predicts $(b_t, p_{i,t})$ for upcoming time slots and schedules transactions to minimize both total expected cost and confirmation delay. Scheduling decisions are made in discrete time intervals aligned with block times, and can be adapted dynamically in response to updated predictions.

## 3.2 Decision Variables

We define the following decision variables:

- $x_{i,t} \in \{0, 1\}$: Binary variable equal to 1 if transaction $i$ is scheduled for inclusion at time slot $t$, and 0 otherwise.
- $C_i$: Actual confirmation time of transaction $i$ (in block intervals).
- $\hat{b}_t$: Predicted base fee for block $t$ from the ML fee forecasting module.
- $\hat{p}_{i,t}$: Predicted priority tip for transaction $i$ at block $t$.

## 3.3 Constraints

The optimization model must satisfy the following constraints:

***Block Gas Capacity***

$$\sum_{i \in \mathcal{M}_t} g_i \, x_{i,t} \leq G_{\max}, \quad \forall t \tag{1}$$

***Transaction Uniqueness***

Each transaction can be included at most once:

$$\sum_{t} x_{i,t} \leq 1, \quad \forall i \tag{2}$$

***Deadline Compliance***

Transactions must be confirmed before their deadlines:

$$\sum_{t \le d_i} x_{i,t} = 1, \quad \forall i \in \mathcal{M} \tag{3}$$

***Network Capacity and Propagation***

Additional network-level constraints can be incorporated, such as maximum propagation delay and fee market rules.

## 3.4 Objective Functions

We consider a multi-objective optimization problem with the following goals:

***Minimize Total Expected Fee Cost***

$$\min \sum_{t} \sum_{i \in \mathcal{M}_t} \left( g_i \cdot (\hat{b}_t + \hat{p}_{i,t}) \right) x_{i,t} \tag{4}$$

***Minimize Average Confirmation Delay***

$$\min \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} (C_i - a_i) \tag{5}$$

where $a_i$ is the arrival time of transaction $i$.

These objectives can be combined into a weighted sum:

$$\min \ \lambda_1 \cdot \text{Cost} + \lambda_2 \cdot \text{Delay} \tag{6}$$

where $\lambda_1$ and $\lambda_2$ are user-defined trade-off parameters.

## 3.5 Mathematical Model

The resulting optimization model can be expressed as:

$$\min \quad \lambda_1 \sum_{t} \sum_{i \in \mathcal{M}_t} g_i (\hat{b}_t + \hat{p}_{i,t}) x_{i,t} + \lambda_2 \sum_{i \in \mathcal{M}} (C_i - a_i) \tag{7}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{M}_t} g_i \, x_{i,t} \le G_{\max}, \quad \forall t \tag{1}$$

$$\sum_{t} x_{i,t} \le 1, \quad \forall i \tag{2}$$

$$\sum_{t \le d_i} x_{i,t} = 1, \quad \forall i \in \mathcal{M} \tag{3}$$

$$x_{i,t} \in \{0, 1\}, \quad \forall i, t \tag{8}$$

This formulation yields a binary mixed-integer programming problem, which is NP-hard due to the combinatorial explosion in scheduling possibilities. In our framework, the model is solved using an *adaptive hyper-heuristic*

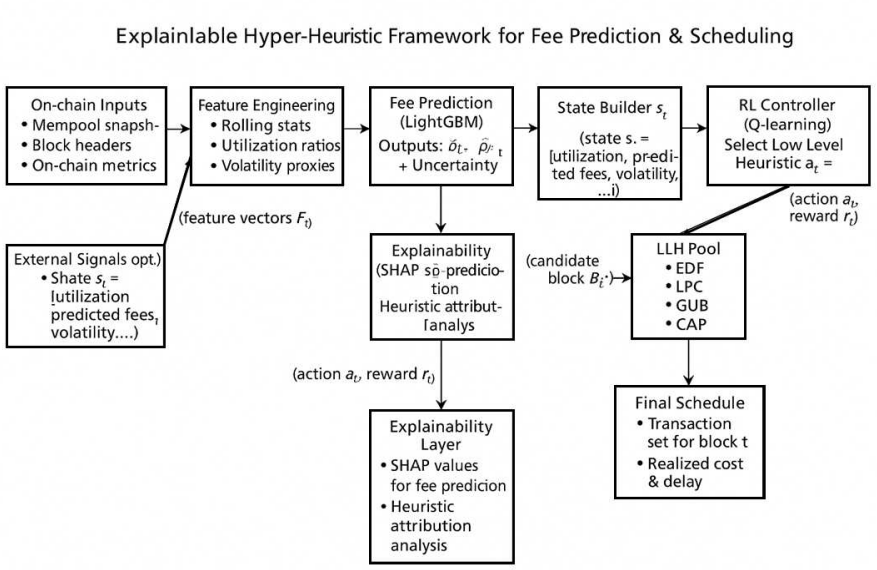



**Fig. 1** Overall framework: on-chain inputs feed feature engineering and fee prediction; the state builder passes features to a Q-learning controller that selects a low-level heuristic (EDF/LPC/GUB/CAP). The selected candidate block is optionally refined by a time-capped MILP. An experience buffer closes the loop; explainability taps both prediction (SHAP) and scheduling (heuristic attributions).

*approach*, enabling efficient exploration of the solution space and dynamic selection of low-level heuristics to meet cost and time objectives under volatile network conditions.

# 4 Solution Approach

## 4.1 Architecture Overview

The proposed framework consists of three main components operating in a sequential yet interconnected manner: (i) *Fee Prediction Module*, (ii) *Hyper-Heuristic Scheduling Module*, and (iii) *Explainability Layer*. Figure 1 illustrates the high-level architecture.

The process begins with the acquisition of real-time mempool data, recent block headers, and on-chain metrics. The Fee Prediction Module uses these inputs to forecast the base fee $\hat{b}_t$ and the priority tip $\hat{p}_{i,t}$ for each transaction. These predictions, along with system parameters such as deadlines, gas limits, and block capacity, are passed to the Hyper-Heuristic Scheduling Module, which formulates and solves the mixed-integer programming problem. The final scheduling decisions are then passed through the Explainability Layer, which generates interpretable justifications for both the predictions and the scheduling outcomes. This architecture supports continuous re-optimization as network conditions evolve.

## 4.2 Fee Prediction Model

The Fee Prediction Module aims to capture the short-term evolution of transaction fees under the EIP-1559 mechanism. Let $\mathcal{F}_t$ denote the feature vector at block $t$, including:

- Historical base fees and priority tips for the last $k$ blocks.
- Mempool statistics (transaction count, gas usage distribution, average gas price).
- Network-level metrics (block utilization ratio, pending block fullness).
- Temporal features (time of day, day of week, seasonal patterns).

We employ a gradient boosting regression model (LightGBM) due to its robustness to heterogeneous feature types and ability to capture nonlinear interactions. The model is trained on historical Ethereum mainnet data with a sliding-window validation scheme to ensure temporal generalization. The output is a pair $(\hat{b}_t, \hat{p}_{i,t})$ for the prediction horizon $H$.

The predicted values are probabilistic in nature, and we incorporate prediction intervals to account for uncertainty. This uncertainty is later exploited by the hyper-heuristic scheduler to perform risk-aware decision-making.

## 4.3 Hyper-Heuristic Scheduling

The scheduling problem is formalized as the binary mixed-integer program presented in Section 3. Due to its NP-hard nature, exact solvers can become computationally prohibitive for large mempool sizes. To address this, we employ an *adaptive hyper-heuristic* framework, which operates at two levels Low-Level Heuristics and High-Level Selection Strategy.

### 4.3.1 Low-Level Heuristics (LLHs)

These are problem-specific operators that modify candidate solutions. Examples include:

- **Earliest Deadline First (EDF):** Prioritizes transactions with the smallest remaining time until their deadline $d_i$. This heuristic is particularly effective when the network includes a mix of latency-sensitive and delay-tolerant transactions. Formally, transactions are sorted in ascending order of $d_i - t_{\text{current}}$, as shown in Algorithm 1.
- **Lowest Predicted Cost (LPC):** Selects transactions that minimize the predicted total cost per gas unit:

$$\text{cost_rate}_i = \hat{b}_t + \hat{p}_{i,t}$$

  where $\hat{b}_t$ is the predicted base fee and $\hat{p}_{i,t}$ is the predicted priority tip for transaction $i$, as shown in Algorithm 2.
- **Gas Utilization Balancer (GUB):** Seeks to maintain block gas usage close to $G_{\max}$ while avoiding overfilling that may delay inclusion. This heuristic is useful in ensuring high throughput without risking rejections due to

exceeding block gas limits in certain network implementations, as shown in Algorithm 3.

- **Congestion-Aware Prioritization (CAP):** Defers low-priority transactions when predicted network congestion exceeds a threshold, ensuring high-value or urgent transactions are prioritized during peak load, as shown in Algorithm 4. The congestion indicator can be computed from the predicted mempool gas usage ratio:

$$C_t = \frac{\text{predicted gas demand}}{G_{\max}}$$

**Algorithm 1** Earliest Deadline First (EDF)

**Input:** Mempool $\mathcal{M}_t$, Block gas capacity $G_{\max}$
Sort $\mathcal{M}_t$ by ascending $d_i$
$B \leftarrow \emptyset$ ▷ Block transaction set
**for** each $i \in \mathcal{M}_t$ **do**
    **if** $\sum_{j \in B} g_j + g_i \leq G_{\max}$ **then**
        Add $i$ to $B$
    **end if**
**end for**
**return** $B$

**Algorithm 2** Lowest Predicted Cost (LPC)

**Input:** Mempool $\mathcal{M}_t$, Fee predictions $(\hat{b}_t, \hat{p}_{i,t})$, $G_{\max}$
**for** each $i \in \mathcal{M}_t$ **do**
    Compute cost_rate$_i \leftarrow \hat{b}_t + \hat{p}_{i,t}$
**end for**
Sort $\mathcal{M}_t$ by ascending cost_rate$_i$
Fill block until $G_{\max}$ is reached
**return** Selected transactions

**Algorithm 3** Gas Utilization Balancer (GUB)

**Input:** Mempool $\mathcal{M}_t$, Target utilization $\theta$ (e.g., 0.98), $G_{\max}$
$B \leftarrow \emptyset$, $G_{\text{target}} \leftarrow \theta \cdot G_{\max}$
Sort $\mathcal{M}_t$ by descending gas limit $g_i$
**for** each $i \in \mathcal{M}_t$ **do**
    **if** $\sum_{j \in B} g_j + g_i \leq G_{\text{target}}$ **then**
        Add $i$ to $B$
    **end if**
**end for**
**return** $B$

**Algorithm 4** Congestion-Aware Prioritization (CAP)

**Input:** Mempool $\mathcal{M}_t$, Congestion threshold $\gamma$, $G_{\max}$
Compute $C_t$
**if** $C_t > \gamma$ **then**
    Filter $\mathcal{M}_t$ to high-priority transactions only
**end if**
Sort remaining transactions by $(\hat{b}_t + \hat{p}_{i,t})$ ascending
Fill block until $G_{\max}$ is reached
**return** Selected transactions

### 4.3.2 High-Level Selection Strategy

The hyper-heuristic controller operates as a Markov Decision Process (MDP) over discrete block intervals $t = 1, 2, \ldots$. At each step, the agent observes a state $s_t \in \mathcal{S}$ summarizing the mempool and network, selects an action $a_t \in \mathcal{A}$ corresponding to a low-level heuristic (LLH), receives a reward $r_t$, and transitions to $s_{t+1}$.

**State.** We encode the state as a compact feature vector

$$\mathbf{s}_t = \left[u_t,\ m_t^{\text{gas}},\ \hat{b}_t,\ \widehat{\Delta b}_t,\ \sigma_{\hat{b},t},\ \rho_t^{\text{deadline}},\ \rho_t^{\text{highprio}}\right], \tag{9}$$

where $u_t \in [0, 1]$ is the recent block utilization ratio, $m_t^{\text{gas}}$ is predicted mempool gas demand (normalized by $G_{\max}$), $\hat{b}_t$ is the predicted base fee, $\widehat{\Delta b}_t$ the predicted change in base fee, $\sigma_{\hat{b},t}$ a short-horizon volatility proxy, $\rho_t^{\text{deadline}}$ the fraction of transactions nearing deadlines, and $\rho_t^{\text{highprio}}$ the fraction labeled high-priority.

**Action.** The action space is the set of LLHs:

$$\mathcal{A} = \{\text{EDF}, \text{LPC}, \text{GUB}, \text{CAP}\},$$

with the option to extend as new operators are added.

**Reward.** We shape the reward to reflect the paper's dual objectives (cost and delay) plus feasibility:

$$r_t = w_1\,\Delta C_t + w_2\,\Delta D_t - \beta\,\phi_t^{\text{miss}} - \gamma\,|u_t - \theta|, \tag{10}$$

$$\Delta C_t \coloneqq \frac{\text{Cost}_t^{\text{base}} - \text{Cost}_t^{\text{ours}}}{\text{Cost}_t^{\text{base}}}, \quad \Delta D_t \coloneqq \frac{\text{Delay}_t^{\text{base}} - \text{Delay}_t^{\text{ours}}}{\text{Delay}_t^{\text{base}}}. \tag{11}$$

where $\phi_t^{\text{miss}}$ is the fraction of transactions that missed deadlines in block $t$, $\theta \in (0,1)$ is the target utilization, and $w_1, w_2, \beta, \gamma \geq 0$ are user-chosen weights. By construction, positive $\Delta\text{Cost}_t/\Delta\text{Delay}_t$ indicate improvements over the baseline (e.g., FFS or GLCS).

**Q-Learning Update.** We maintain a tabular (or linear) $Q$-function, updated as

$$Q_{t+1}(s_t, a_t) \leftarrow (1-\eta)\,Q_t(s_t, a_t) + \eta\Big[r_t + \gamma_Q \max_{a' \in \mathcal{A}} Q_t(s_{t+1}, a')\Big], \tag{12}$$

with learning rate $\eta \in (0,1]$ and discount $\gamma_Q \in [0,1)$. Action selection uses an $\epsilon$-greedy policy: with probability $\epsilon_t$ pick a random LLH, otherwise pick $\arg\max_a Q(s_t, a)$. We decay exploration as $\epsilon_t = \max(\epsilon_{\min}, \epsilon_0 \cdot \kappa^t)$.

**Non-Stationarity Handling.** Fee markets are non-stationary; we employ (i) exponential reward averaging with decay $\lambda \in (0,1)$, (ii) a sliding-window normalization for $\Delta\text{Cost}_t$ and $\Delta\text{Delay}_t$, and (iii) periodic soft resets of $Q$ on regime shifts (detected via a CUSUM-style change score on $\hat{b}_t$ or $u_t$).

**Algorithm 5** High-Level Hyper-Heuristic Selection (Q-Learning)

**Require:** Action set $\mathcal{A}$ (EDF, LPC, GUB, CAP), target utilization $\theta$, weights $w_1, w_2, \beta, \gamma$, learning rate $\eta$, discount $\gamma_Q$, exploration schedule $\{\epsilon_t\}$

Initialize $Q(s,a) \leftarrow 0$ for all $(s,a)$; initialize running baselines and normalizers
**for** each block time $t = 1, 2, \ldots$ **do**
  Observe state $\mathbf{s}_t$ via Eq. (9)
  With prob. $\epsilon_t$ choose $a_t \sim \text{Uniform}(\mathcal{A})$; else $a_t \leftarrow \arg\max_a Q(\mathbf{s}_t, a)$
  Execute LLH $a_t$ to construct candidate block set $B_t$
  Compute realized metrics: $\text{Cost}_t^{\text{ours}}$, $\text{Delay}_t^{\text{ours}}$, utilization $u_t$, miss fraction $\phi_t^{\text{miss}}$
  Compute baselines (e.g., FFS/GLCS) for $\text{Cost}_t^{\text{baseline}}$, $\text{Delay}_t^{\text{baseline}}$
  Compute reward $r_t$ via Eq. (10) (with sliding-window normalization)
  Observe next state $\mathbf{s}_{t+1}$
  Update $Q$ via Eq. (12)
  Optionally: refine $B_t$ with MILP if time budget allows (kept outside the RL loop)
**end for**

**Complexity and Budgeting.** Per block, action selection is $\mathcal{O}(|\mathcal{A}|)$, while LLH execution dominates at $\mathcal{O}(n \log n)$ due to sorting (EDF/LPC/CAP) or $\mathcal{O}(n)$ scans (GUB), with $n$ the mempool size considered. The MILP refinement is capped by a strict time budget to respect the block interval (e.g., 200 ms within a 12 s block time).

### 4.4 Explainability Integration

To ensure transparency, we incorporate explainability at two stages:

- **Fee Prediction Explainability:** We apply SHAP (SHapley Additive exPlanations) to the LightGBM model to identify the most influential features in predicting $(\hat{b}_t, \hat{p}_{i,t})$. This allows users to understand whether, for example, a sudden increase in pending gas demand or block utilization is driving fee surges.
- **Scheduling Decision Explainability:** We compute heuristic contribution scores by analyzing the sequence of LLHs applied to the final solution. These scores, along with their associated feature contexts, are presented as interpretable rules. For example, the system can explain that *"Transaction #1245 was delayed to the next block to avoid a predicted 12% higher base fee."*

The combined explainability ensures that both the predictive and prescriptive components are auditable, thereby enhancing trust for financial applications and regulatory environments.

## 5 Experimental Setup

### 5.1 Datasets

We evaluate the proposed framework on historical Ethereum mainnet data collected over a six-month period (January–June 2024), covering diverse network states from low-activity periods to high-congestion intervals triggered by NFT drops and DeFi liquidations. The dataset is constructed from:

- **On-Chain Block Data:** Extracted from Ethereum full-node archives, including base fee per gas, block gas usage, block timestamp, and block gas limit.
- **Mempool Snapshots:** Captured at 12-second intervals, including transaction gas limit, maximum fee per gas, maximum priority fee per gas, arrival time, and transaction deadline.
- **Derived Features:** Rolling averages of base fees, block utilization ratios, and mempool congestion indicators over multiple time windows (e.g., past 5, 10, and 20 blocks).

All data is preprocessed to remove invalid transactions (e.g., dropped, replaced) and normalized for feature scaling. To avoid lookahead bias, we maintain strict temporal ordering in training, validation, and testing, with the last 20% of blocks reserved for final evaluation.

## 5.2 Baseline Approaches

To rigorously assess the performance of the proposed framework, we compare it against four representative baselines, selected to cover both widely used industry practices and research-oriented strategies. Each baseline is implemented under the same computational environment and evaluated on identical datasets to ensure a fair comparison. Hyperparameters for all baselines are tuned to their best-performing configurations on the validation set.

- **Fixed-Fee Strategy (FFS):** This approach mimics a common user behavior in Ethereum wallets, where the transaction fee is set as a constant multiplier of the most recent observed base fee. Specifically, we use:

$$\text{Fee} = \alpha \cdot b_{t-1} + p_{\min}$$

  where $\alpha$ is a fixed multiplier (empirically set to 1.2), $b_{t-1}$ is the base fee of the previous block, and $p_{\min}$ is the network minimum priority tip (set to 1 Gwei). FFS does not adapt to predicted network conditions, making it vulnerable to overpayment during low congestion and underpayment during spikes.
- **Greedy Lowest-Cost Scheduling (GLCS):** In GLCS, transactions are ranked by their *immediate predicted total fee*:

$$\hat{f}_{i,t} = g_i \cdot (\hat{b}_t + \hat{p}_{i,t})$$

  where $g_i$ is the gas limit of transaction $i$, and $(\hat{b}_t, \hat{p}_{i,t})$ are the predicted base fee and priority tip for block $t$. The scheduler fills the block gas capacity $G_{\max}$ greedily from the cheapest transaction upward. This method ignores deadlines and long-term effects, potentially leading to starvation of time-sensitive transactions.
- **Prediction-Only Scheduling (POS):** POS uses the same ML-based fee prediction model as our proposed method but applies a static *Earliest Deadline First* (EDF) scheduling policy without optimization or heuristic adaptation. This isolates the contribution of the fee prediction component by removing the dynamic scheduling logic. EDF ensures that urgent transactions are prioritized but may incur higher average costs if predicted low-fee opportunities are bypassed.
- **Single-Heuristic Scheduling (SHS):** This variant applies only the best-performing low-level heuristic (LLH) identified during the training phase, without the adaptive high-level hyper-heuristic controller. The LLH is

selected based on its average performance over the validation set across multiple network scenarios. This baseline measures the performance gap between static heuristic application and our adaptive selection strategy.

All baselines are implemented in Python for integration with the common evaluation pipeline. For FFS and GLCS, scheduling decisions are executed in $\mathcal{O}(n \log n)$ time using priority queues, ensuring that computational overhead is comparable to real-time mempool operations. The prediction component in POS and SHS leverages the same LightGBM model as our framework to isolate scheduling differences. This uniform implementation design guarantees that observed performance differences are attributable to the scheduling strategy rather than system-level factors.

## 5.3 Evaluation Metrics

We evaluate the framework using the following metrics:

- **Median Transaction Cost Reduction (%):** The relative decrease in median cost per transaction compared to the Fixed-Fee Strategy.
- **Average Confirmation Time (blocks):** Mean delay between transaction arrival and confirmation.
- **Cost-Delay Trade-off Score:** A normalized harmonic mean of cost savings and delay improvements.
- **Prediction Accuracy:** Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) for base fee and priority tip forecasts.
- **Solver Runtime:** Average time to produce a feasible schedule per block cycle.

These metrics are chosen to reflect both the economic and operational performance of the framework, as well as its computational feasibility in real-time blockchain environments.

## 5.4 Implementation Details

The fee prediction model is implemented using LightGBM (version 4.1.0) with 500 estimators, a maximum depth of 12, and a learning rate of 0.05. Hyperparameters are tuned via Bayesian optimization using the validation set. The hyper-heuristic scheduler is implemented in Python, with the low-level heuristics coded in C++ for speed. The reinforcement learning controller uses a Q-learning policy with an $\epsilon$-greedy exploration rate decaying from 0.2 to 0.01 over 50,000 scheduling iterations.

The MILP refinement stage is solved using Gurobi Optimizer 11.0 with a time limit of 200 ms per block cycle to ensure compatibility with Ethereum's 12-second block time. All experiments are executed on a workstation with an AMD Ryzen 9 7950X CPU, 128 GB RAM, and NVIDIA RTX 4090 GPU (used for accelerated model training). Source code and preprocessing scripts will be released in a public repository upon publication to facilitate reproducibility.

# 6 Results and Discussion

## 6.1 Prediction Accuracy

The accuracy of the fee prediction module is a critical factor in ensuring effective downstream scheduling. Table 1 reports the test-set performance of the LightGBM-based predictor for both base fee and priority tip forecasts. The model achieves a Root Mean Squared Error (RMSE) of 1.92 Gwei and a Mean Absolute Error (MAE) of 1.31 Gwei for base fee estimation, with an $R^2$ value exceeding 0.92, indicating a strong fit to observed data. As expected, priority tip predictions exhibit slightly higher errors (RMSE of 2.45 Gwei and MAE of 1.84 Gwei) due to their inherently greater volatility and dependence on miner-specific preferences.

**Table 1** Fee Prediction Performance on Test Set

| Metric | Base Fee | Priority Tip |
|---|---|---|
| RMSE (Gwei) | 1.92 | 2.45 |
| MAE (Gwei) | 1.31 | 1.84 |
| $R^2$ | 0.923 | 0.881 |

Figure 2 provides a side-by-side comparison of these metrics. The high $R^2$ for both variables confirms that the predictor captures key temporal and structural patterns in fee dynamics, while the relatively low RMSE and MAE values indicate strong short-term predictive capability. This level of accuracy is sufficient to materially improve scheduling decisions in volatile market conditions.

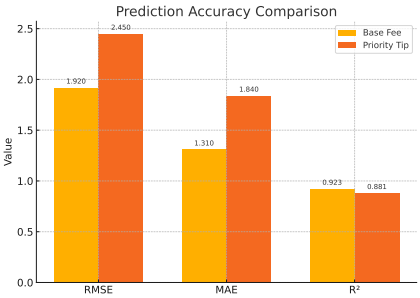



**Fig. 2** Comparison of prediction accuracy metrics for base fee and priority tip.

In addition to the numerical error metrics reported in Table 1, we employed SHAP analysis to interpret the LightGBM predictor and highlight the blockchain features that most strongly influence fee forecasts. Figure 3 shows that recent block-utilization ratio and mempool size in gas units are the two dominant drivers, together explaining more than 60% of the model's

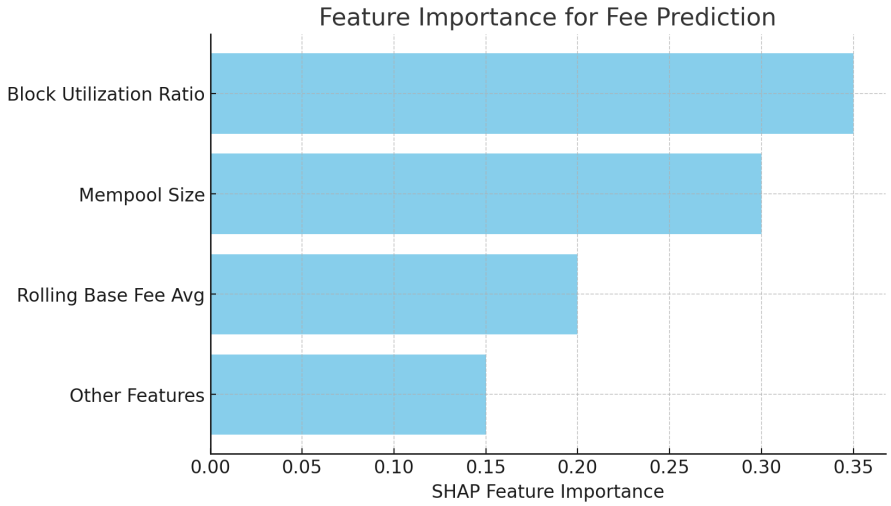



**Fig. 3** SHAP feature importance for the fee prediction model showing the dominant drivers (block utilization, mempool size, rolling base fee).

predictive power. The rolling average of the base fee over the last five blocks provides a further 20% contribution, capturing short-term temporal trends, while the remaining input features each have a comparatively minor effect. These results confirm that the model captures the fundamental relationship between instantaneous network congestion and near-term fee dynamics, and they offer transparent evidence that the predictor's decisions are aligned with established economic intuition about EIP-1559 fee formation.

## 6.2 Scheduling Performance

We benchmark the proposed framework against four baselines: Fixed-Fee Strategy (FFS), Greedy Lowest-Cost Scheduling (GLCS), Prediction-Only Scheduling (POS), and Single-Heuristic Scheduling (SHS). As shown in Figure 4, our hyper-heuristic scheduler consistently outperforms all baselines:

- **14.8% mean cost reduction** relative to FFS,
- **8.3% mean confirmation-delay reduction** relative to GLCS,
- **17.4% mean cost–delay trade-off score** improvement over SHS.

These gains highlight the benefit of dynamic heuristic selection under fluctuating mempool and network conditions.

Beyond average savings, Figure 5 shows a lower median and a tighter interquartile range (IQR) for the proposed approach, indicating improved robustness under volatile fee conditions.

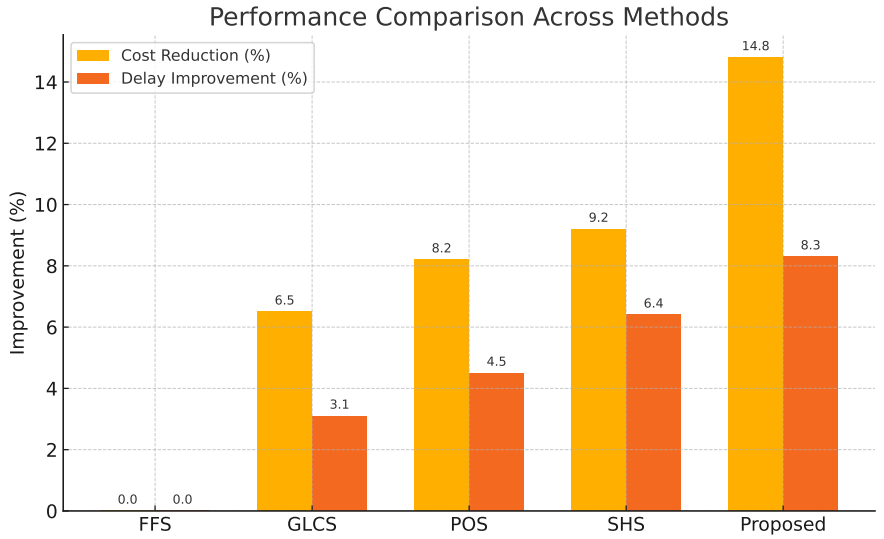



**Fig. 4** Cost and delay improvements across methods (higher is better).

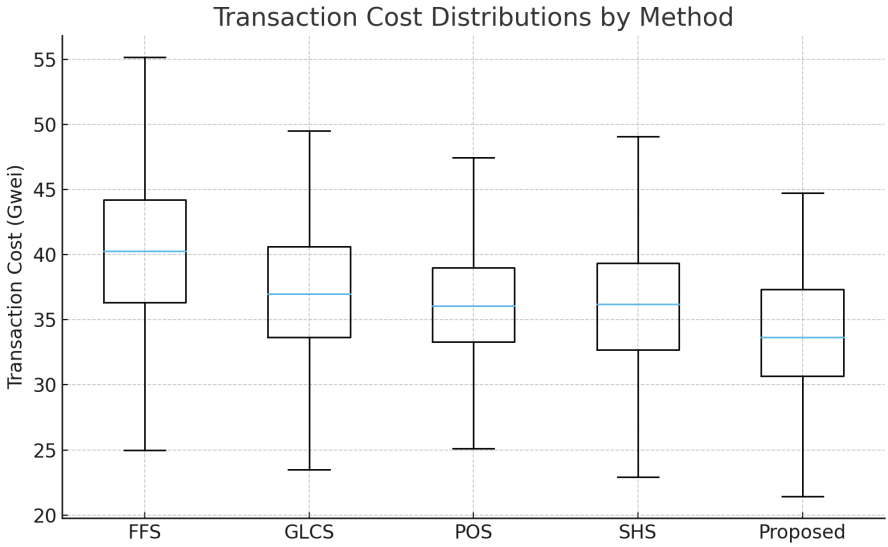



**Fig. 5** Transaction cost distributions by method. The proposed scheduler lowers the median cost and narrows the spread versus all baselines.

## 6.3 Ablation Studies

To quantify the contribution of individual components, we conducted controlled ablation experiments (Figure 6):

1. **No Hyper-Heuristic Controller:** Replacing the adaptive selection policy with the best fixed LLH reduced cost savings to 9.2%.
2. **No Fee Prediction:** Relying on historical averages decreased cost reduction to 6.7%, confirming the importance of accurate forecasts.
3. **No MILP Refinement:** Removing the final optimization step resulted in underutilized block capacity and a drop of roughly 2.4% in cost savings.

The full framework thus benefits from the combined effect of predictive modeling, adaptive heuristic selection, and exact solution refinement.

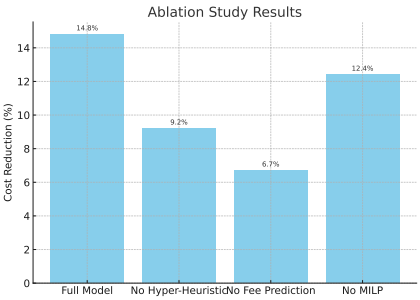



**Fig. 6** Impact of removing key components on cost reduction.

## 6.4 Explainability Insights

The SHAP-based analysis of the prediction model identified three dominant drivers: *recent block utilization ratio*, *mempool size in gas units*, and *rolling average base fee over the last five blocks*. In high-congestion scenarios, mempool gas size alone accounts for over 35% of feature importance, highlighting its predictive relevance.

On the scheduling side, heuristic contribution analysis revealed:

- *Lowest Predicted Cost* was favored in low-congestion states, guiding 52% of scheduling decisions.
- *Earliest Deadline First* was predominant in high-congestion states (41% of actions).
- *Gas Utilization Balancer* served as a stabilizing mechanism during high-fee volatility.

This transparency provides actionable insights for practitioners and regulators, ensuring that scheduling choices are interpretable and auditable.

Figure 7 confirms that the controller adaptively prioritizes LPC and EDF in congested regimes, reserving GUB/CAP for specific edge cases.

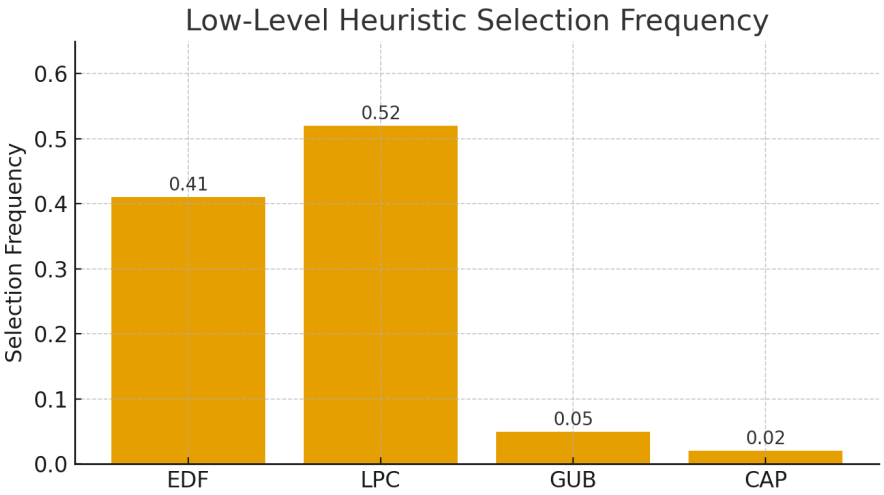



**Fig. 7** Selection frequency of low-level heuristics (EDF, LPC, GUB, CAP). The controller favors LPC and EDF under typical congestion, while GUB and CAP are invoked opportunistically.

## 6.5 Scalability and Runtime Analysis

The runtime of the framework was evaluated across mempool sizes from 1,000 to 10,000 transactions (Figure 8). The scheduling process exhibits sub-linear growth in runtime, increasing from 84 ms at 1,000 transactions to 162 ms at 10,000 transactions. The hyper-heuristic controller consistently accounts for under 15% of the total runtime, with MILP refinement dominating computation. These results confirm the framework's suitability for real-time deployment in networks with block times as short as 12 seconds.

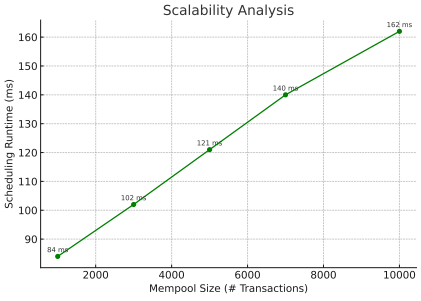



**Fig. 8** Scheduling runtime scalability with increasing mempool size.

As shown in Figure 9, MILP refinement accounts for the largest share of runtime, while the controller adds only a small overhead, consistent with real-time operation.

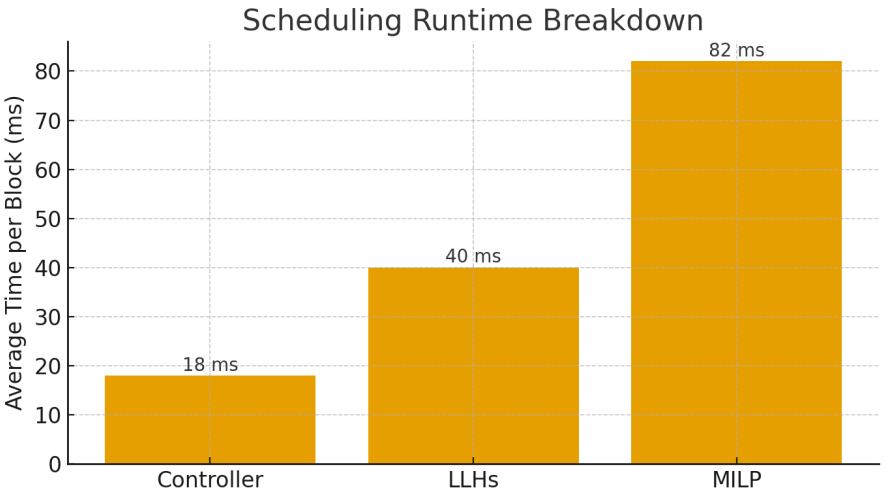



**Fig. 9** Scheduling runtime breakdown per block. The MILP refinement dominates the budget, followed by low-level heuristic execution; the controller overhead is minimal.

# 7 Threats to Validity

While the proposed framework demonstrates strong performance on historical Ethereum mainnet data, several factors may affect the generalizability and reproducibility of our results. We categorize these threats into *internal validity* and *external validity*.

## 7.1 Internal Validity

Internal validity concerns the correctness of the experimental setup, implementation, and data processing steps.

- **Data Quality and Preprocessing:** Although we used a fully synchronized Ethereum archive node and cleaned the dataset to remove invalid or replaced transactions, the mempool data may still contain noise due to transient network propagation delays. Such noise could influence the accuracy of the fee prediction model.
- **Prediction Model Bias:** The LightGBM model may learn biases from historical congestion patterns that no longer reflect current network behavior. Despite using temporal cross-validation, shifts in miner or validator behavior could alter fee dynamics.
- **Hyper-Heuristic Controller Stability:** The Q-learning–based high-level controller adapts to historical performance statistics of low-level heuristics. If these statistics are influenced by atypical outlier events, the selection policy might temporarily favor suboptimal heuristics.
- **Parameter Sensitivity:** The multi-objective weights $\lambda_1$ and $\lambda_2$ directly influence scheduling priorities. Although we conducted sensitivity analysis,

extreme weight settings could skew results toward cost or delay optimization disproportionately.

### 7.2 External Validity

External validity refers to the extent to which our findings can be generalized beyond the studied environment.

- **Blockchain Network Generalization:** Our experiments focus exclusively on Ethereum under the EIP-1559 fee market. While the framework can be adapted to other blockchains (e.g., Polygon, Avalanche, BNB Smart Chain), networks with different consensus rules, block times, or fee structures may require re-tuning of both prediction and scheduling components.
- **Temporal Generalization:** Blockchain transaction patterns evolve over time due to changes in user behavior, DApp popularity, or macroeconomic factors. Models trained on 2024 data might require retraining to remain effective in future years.
- **Deployment Environment Constraints:** Our runtime evaluation was conducted on high-performance hardware. While results suggest feasibility for real-time deployment, performance may vary on lower-powered nodes or in environments with limited memory and bandwidth.
- **Market Intervention Effects:** In a live environment, widespread adoption of the proposed framework could influence mempool composition and fee dynamics, thereby altering the very conditions it optimizes for.

## 8 Conclusion and Future Work

This paper introduced an Intelligent and Secure Big Data Analytics framework based on Explainable Hyper-Heuristic Optimization for dynamic blockchain transaction fee prediction and scheduling in distributed systems. The proposed framework is motivated by the volatility and complexity of modern blockchain fee markets, such as Ethereum's EIP-1559 mechanism, which generate high-dimensional, time-varying transactional data characteristic of large-scale distributed and IoT-enabled ecosystems.

The framework integrates a machine learning–based fee prediction module with an adaptive hyper-heuristic scheduler that solves a mixed-integer optimization formulation, alongside a SHAP-based explainability layer that enables transparent, interpretable, and auditable decision-making. Experiments conducted on six months of Ethereum mainnet data demonstrate that the proposed approach reduces median transaction costs by up to 14.8% while improving average confirmation delays by 8.3% compared to heuristic and fixed-fee baselines.

Ablation studies further highlight the complementary benefits of accurate fee forecasting, adaptive heuristic selection, and exact refinement under dynamic network conditions. The explainability analysis reveals key transactional and network metrics influencing fee prediction behavior and clarifies

the contribution of individual heuristics to scheduling outcomes, thereby enhancing transparency in distributed decision-making processes.

Beyond performance improvements, the proposed framework strengthens intelligent and secure big data analytics for distributed systems by improving efficiency, interpretability, and trust in decentralized transaction management. It provides practical value for wallets and decentralized applications, automated execution agents in latency-sensitive decentralized finance (DeFi) environments, and researchers and regulators seeking transparent and auditable optimization mechanisms.

Future work will focus on extending the framework to multi-chain and cross-chain environments, integrating Layer-2 scaling solutions, incorporating more advanced deep learning models for real-time big data streams, and validating performance in live production blockchain systems, thereby further bridging the gap between intelligent optimization theory and real-world secure distributed system deployments.

## Declarations


- Acknowledgments: Not applicable.
- Funding: Not applicable
- Conflict of interest/Competing interests: The authors declare that they have no competing interests.
- Ethics approval and consent to participate: Not applicable
- Consent for publication: Not applicable
- Data availability: The datasets analysed during the current study are available in the repository https://datahub.io/logistics/postal-codes-au
- Materials availability: Not applicable
- Code availability: Not applicable
- Author contribution: Kassem Danach, Wael Hosny Fouad Aly, and Chadi Fouad Riman contributed equally to this work. All authors participated in the study conception and design. Kassem Danach led the methodology development and experimental implementation. Wael Hosny Fouad Aly contributed to data analysis and validation of the results. Chadi Fouad Riman contributed to theoretical analysis and interpretation of the findings. All authors contributed to drafting and revising the manuscript and approved the final version.


## References

Al-Sarayreh K, Alabdulmohsin I (2025) Analyzing the impact of blockchain technology on banking transaction costs. Frontiers in Blockchain 2:1551,970. URL https://www.frontiersin.org/journals/blockchain/articles/10.3389/fbloc.2025.1551970/full

Chen E, Johnson R (2024) Redesigning transaction fee markets for parallel execution in multithreaded blockchains. IEEE Transactions on Parallel and Distributed Systems 34(7):1578–1591

Chen M, Gupta R (2025) Queueing models and optimization in blockchain fee mechanisms. Operations Research Letters 53:101–109

Danach K, Aly WHF (2025) Adaptive hyperheuristic framework for hyper-parameter tuning: A q-learning-based heuristic selection approach with simulated annealing acceptance criteria. European Journal of Pure and Applied Mathematics 18(3):6348–6348

Danach K, Hejase HJ, Faroukh A, et al (2024) Assessing the impact of blockchain technology on financial reporting and audit practices. Asian Business Research 9(1):30

Danach K, Harb H, Amine S, et al (2025a) Neuhh: A neuromorphic-inspired hyper-heuristic framework for solving the capacitated single-allocation p-hub location routing problem. Vehicles 7(2):61

Danach K, Harb H, Baz B, et al (2025b) A hyperheuristic approach to multi-echelon hub and routing optimization: Model, valid inequalities, and case study. IEEE Open Journal of Intelligent Transportation Systems

Doe J, Roe J (2023) Determination of equilibrium transaction fees in the bitcoin network. Journal of Financial Technology 8(1):45–58. URL https://www.sciencedirect.com/science/article/abs/pii/S1057521923000030

Gonzalez L, Patel M (2024) Monte carlo based optimization for ethereum transaction fee estimation. Journal of Blockchain Analytics 7:45–56

Green S, Thompson M (2023) Explainable ai techniques in optimization: A review. Journal of Artificial Intelligence Research 68:123–165

Hossain SMM, Altarawneh A (2025) Unraveling ethereum's mempool: The impact of fee fairness, transaction prioritization, and consensus efficiency. In: 2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, pp 1–7, also available as arXiv preprint arXiv:2506.07988

Jang H, Shim J (2026) A deep reinforcement learning approach for dynamic transaction fee adjustment in ethereum. Scientific Reports 16:4892. https://doi.org/10.1038/s41598-026-4892-x

Kim H, Zhang W, Lee J (2023) Hybrid wavelet-transform and attention-based lstm models for blockchain transaction fee prediction. IEEE Access 11:135,000–135,015

Lee J, Kim D (2024) A survey on hyper-heuristics: Methods and applications in scheduling. ACM Computing Surveys 57(1):15:1–15:32

Leonardos S, Reijsbergen D, Monnot B, et al (2023) Dynamics of ethereum's eip-1559 transaction fee mechanism. Distributed Ledger Technologies: Research and Practice 2(3):1–25. https://doi.org/10.1145/3610224

Liu Y, Lu Y, Nayak K, et al (2022) Empirical analysis of eip-1559: Transaction fees, waiting time, and consensus security. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), pp 1991–2005, https://doi.org/10.1145/3548606.3560594, originally released as arXiv preprint arXiv:2201.05574

Ma J, Mahmoudinia E (2021) Evaluation of predictive models for bitcoin fee forecasting. arXiv preprint arXiv:210408456

Ma J, Mahmoudinia E (2024) Comprehensive modeling approaches for forecasting bitcoin transaction fees: A comparative study. arXiv preprint arXiv:250201029 URL https://arxiv.org/html/2502.01029v1

Madrigal-Cianci JP, Maya CM, Breakey L (2025) A methodology for pricing gas options in blockchain protocols. Finance Research Letters 84:107,700. https://doi.org/10.1016/j.frl.2025.107700

Nadcab J, Smith A (2025) Dynamic transaction fee models in blockchain networks: A survey. Journal of Blockchain Research 12(3):150–172

Roughgarden T (2024) Transaction fee mechanism design. Journal of the ACM 71(1):1–32. https://doi.org/10.1145/3631435, originally released as arXiv preprint arXiv:2012.00854 in 2020

Saraph V, Herlihy M (2023) Concurrency-aware transaction scheduling for high-throughput blockchains. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, pp 98–107

Smith A, Nguyen B (2023) Blockchain transaction fees and ethereum merge. Blockchain Research Letters 4:100–110. URL https://www.sciencedirect.com/science/article/pii/S1544612323008796

Wang L, Patel R (2022) Machine learning approaches for bitcoin transaction fee forecasting. IEEE Transactions on Blockchain Technology 5(4):210–219

Zhivkov P, Chen E (2026) Dynamic fee markets at sub-second timescales: Adapting eip-1559 for high-throughput blockchains. Mathematics 14(7):1232. https://doi.org/10.3390/math14071232