[Cross posted from Efficient L2 batch posting strategy on L1 - Arbitrum Research]

Posting data batches of layer two (L2) rollup chains on layer one (L1) constitutes most of the costs the former chains incur. We try to find an optimal strategy for when to post transaction batches as the L1 price of posting data fluctuates. The tradeoff is clear: we want to avoid delaying the posting of batches, but at the same time, we want to avoid posting when the price is high. The question is motivated by historical experience with the Ethereum base fee, which fluctuates in a partially predictable way, and occasionally has intervals of very high base fee.

We decompose the cost in two parts: a posting cost that can be directly observed when posting batches, and a delay cost that is not directly measurable. Delay cost has several components. The first is psychological: users do not like when the batches are not posted for long period, as it may suggest to them that components of the system are down or unavailable. The second part is related to delayed finality: L2 transactions are not fully final until they are posted as part of a batch and that batch posting has finality on L1. Until that time, users must either wait for finality or trust the L2 system's sequencer to be honest and non-faulty. Delayed finality imposes costs on some applications. The third part is related to the specific technical nature of transaction fee computation on L2 rollups. Namely, a transaction fee is calculated when transactions are created, not when they are posted on L1. Therefore, more delay causes less precise estimates of the L1 cost to attribute to L2 transactions, increasing the risk of unfair or inefficient pricing of L2 transactions.

We model the problem as a Markov decision process. In each round, we calculate total costs independently from the past rounds. Each round is characterized by the current queue size and price, which gives a state. The price in the next round is a random variable, which depends on the current price. Depending on the strategy in the current random and the random variable indicating the price in the next round, we move to the next state. To solve the optimization problem of finding the optimal strategy in each round, we use tools from dynamic programming, in particular, q-learning. The structure of the solution allows us to design a practical algorithm, that we test against benchmarks on the previous year's Ethereum base fee data.

See details in the paper:https://arxiv.org/pdf/2212.10337.pdf

Any feedback on modeling assumptions? Any suggestions on the functional form of delay costs or how else can we optimize it together with posting costs instead of simply adding them? Suggestions for related literature?