Dynamic Pricing for Non-fungible Resources

Designing Multi-dimensional Blockchain Fee Markets

Theo Diamandis, MIT & Bain Capital Crypto

Based on work by T. Diamandis, A. Evans, T. Chitra, G. Angeris

CryptoEconDay @ ETHDenver 2023

Fee markets with fixed relative prices are inefficient

Fee markets with fixed relative prices are inefficient

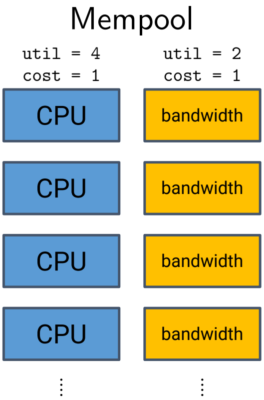
Our work: a framework to optimally set multi-dimensional fees

Outline

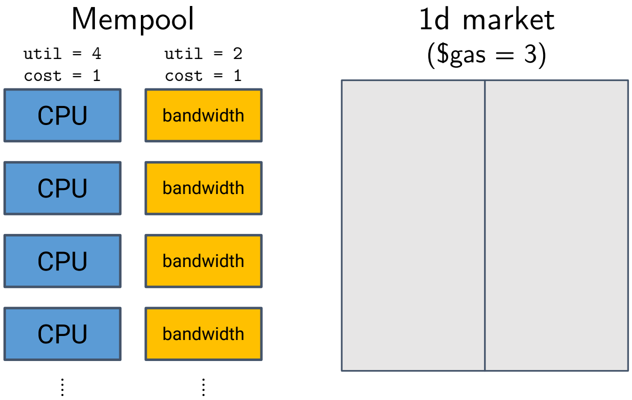
Fixed relative prices lead to DoS attacks

- All opcodes have fixed **relative prices** to each other (measured in gas)
- Potential mismatch between relative prices & resource usage leads to **resource exhaustion attacks** (DoS attacks)
  - EXTCODESIZE attack in 2016 exploited disk read mispricing
  - Opcode prices had to be manually adjusted (EIP-150)

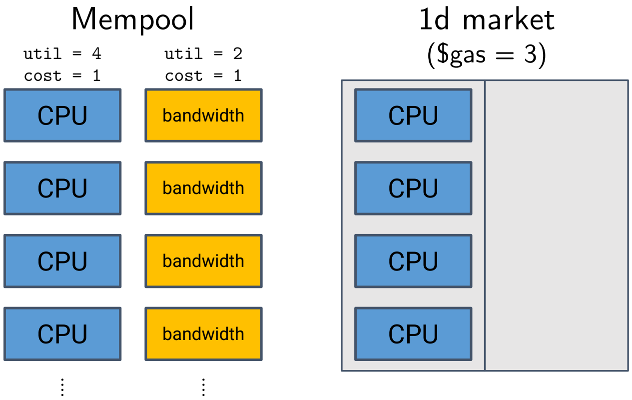
Fixed relative prices limit throughput



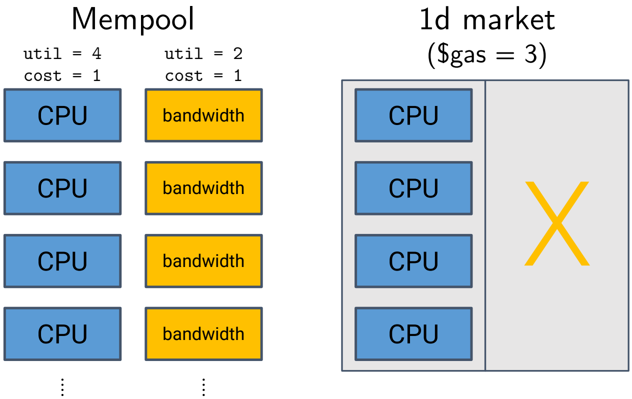
Fixed relative prices limit throughput



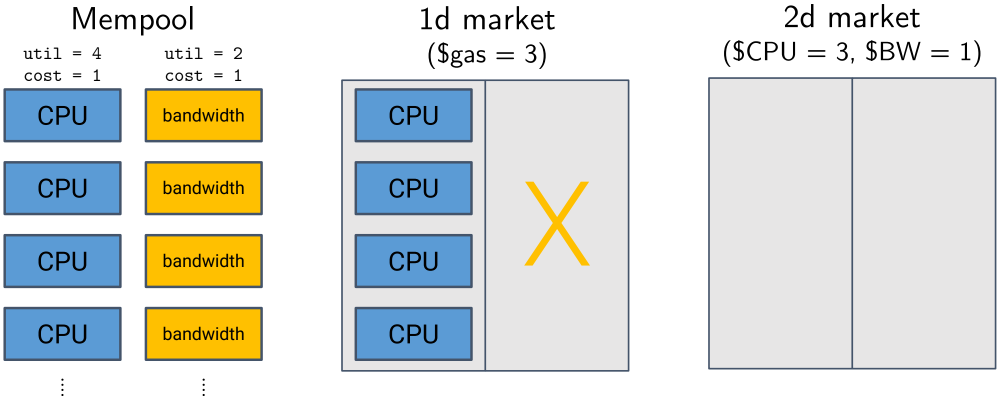
Fixed relative prices limit throughput



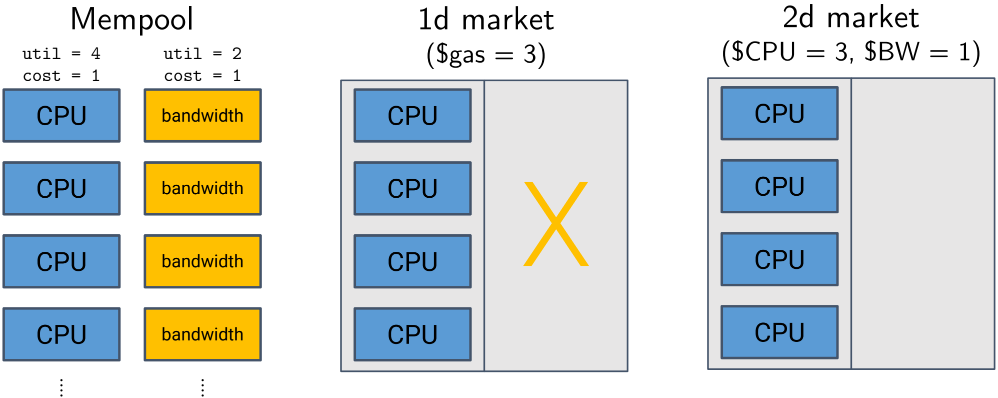
Fixed relative prices limit throughput



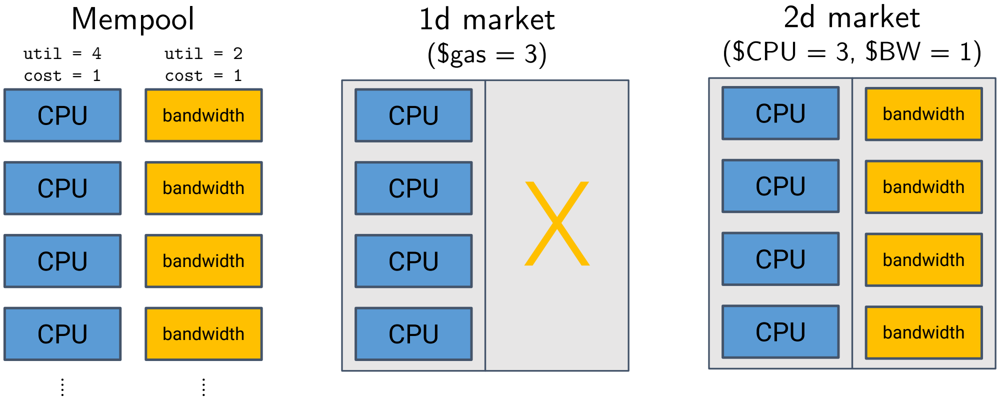
Fixed relative prices limit throughput



Fixed relative prices limit throughput



Fixed relative prices limit throughput



Orthogonal resources should be priced separately

Orthogonal resources should be priced separately

We need a mechanism to design **fee markets**

Outline

But what is a resource?

But what is a resource?

- Anything that can be metered!

But what is a resource?

- Anything that can be metered!
- Blobs (EIP-2242 & EIP-4844)

But what is a resource?

- Anything that can be metered!
- Blobs (EIP-2242 & EIP-4844)
- Compute, memory, storage

But what is a resource?

- Anything that can be metered!
- Blobs (EIP-2242 & EIP-4844)
- Compute, memory, storage
- Opcodes

But what is a resource?

- Anything that can be metered!
- Blobs (EIP-2242 & EIP-4844)
- Compute, memory, storage
- Opcodes
- Sequences of opcodes

But what is a resource?

- Anything that can be metered!
- Blobs (EIP-2242 & EIP-4844)
- Compute, memory, storage
- Opcodes
- Sequences of opcodes
- Compute on a specific core

But what is a resource?

- Anything that can be metered!
- Blobs (EIP-2242 & EIP-4844)
- Compute, memory, storage
- Opcodes
- Sequences of opcodes
- Compute on a specific core
- ...

Let's formalize this

- A **transaction** j consumes a vector of resources $a_j \in \mathbb{R}^m_+$
  - Entry $(a_j)_i$ denotes the amount of resource i consumed by tx j

Let's formalize this

- A **transaction** j consumes a vector of resources $a_j \in \mathbb{R}_+^m$
  - Entry $(a_j)_i$ denotes the amount of resource i consumed by tx j
- The vector $x \in \{0,1\}^n$ records which of n possible txns are included in a block
  - Entry $x_j = 1$ if tx j is included and 0 otherwise

Let's formalize this

- A **transaction** j consumes a vector of resources $a_j \in \mathbb{R}_+^m$
  - Entry $(a_j)_i$ denotes the amount of resource i consumed by tx j
- The vector $x \in \{0, 1\}^n$ records which of n possible txns are included in a block
  - Entry $x_j = 1$ if tx j is included and 0 otherwise
- The quantity of resources consumed by this block is then

$$y = \sum_{j=1}^{n} x_j a_j = Ax$$

We constrain & charge for each resource used

- Define a **resource consumption target** $b^\star$
  - Deviation from the target is $Ax - b^\star$
  - In Ethereum, $b^\star = 15M$ gas

We constrain & charge for each resource used

- Define a **resource consumption target** $b^\star$
  - Deviation from the target is $Ax - b^\star$
  - In Ethereum, $b^\star = 15M$ gas
- Define a **resource consumption limit** b
  - Txns included must satisfy $Ax \leq b$

We constrain & charge for each resource used

- Define a **resource consumption target** $b^\star$
  - Deviation from the target is $Ax - b^\star$
  - In Ethereum, $b^\star = 15M$ gas
- Define a **resource consumption limit** b
  - Txns included must satisfy $Ax \leq b$
- Charge for usage of each resource (*e.g.*, EIP-1559)
  - Prices p, mean that transaction j costs (this is burned)

$$p^T a_j = \sum_{i=1}^{m} p_i (a_j)_i$$

But how do we determine prices?

- We want a few properties:
  - $(Ax)_i = b_i^\star \to$ no update
  - $(Ax)_i > b_i^\star \to p_i$ increases
  - $(Ax)_i < b_i^\star \to p_i$ decreases

But how do we determine prices?

- We want a few properties:
  - $(Ax)_i = b_i^\star \rightarrow$ no update
  - $(Ax)_i > b_i^\star \rightarrow p_i$ increases
  - $(Ax)_i < b_i^\star \rightarrow p_i$ decreases
- Proposal:

$$p_i^{k+1} = p_i^k \cdot \exp\left(\eta(Ax - b^\star)_i\right)$$

But how do we determine prices?

- We want a few properties:
  - $(Ax)_i = b_i^\star \to$ no update
  - $(Ax)_i > b_i^\star \to p_i$ increases
  - $(Ax)_i < b_i^\star \to p_i$ decreases
- Proposal:

$$p_i^{k+1} = p_i^k \cdot \exp\left(\eta(Ax - b^\star)_i\right)$$

Is this a good update rule?

Update rules are implicitly solving an optimization problem

Update rules are implicitly solving an optimization problem

Specific choice of objective by network designer $\implies$ specific update rule

Outline

Setting (for now):

Network designer is omniscient and determines txns in each block

Loss function is network's unhappiness with resource usage

- Network designer determines **loss function** for resource allocation problem; *e.g.*:

$$\ell(y) = \begin{cases} 0 & y = b^\star \\ \infty & \text{otherwise} \end{cases}$$

Loss function is network's unhappiness with resource usage

- Network designer determines **loss function** for resource allocation problem; *e.g.*:

$$\ell(y) = \begin{cases} 0 & y = b^\star \\ \infty & \text{otherwise} \end{cases}$$

$$\ell(y) = \begin{cases} 0 & y \leq b^\star \\ \infty & \text{otherwise} \end{cases}$$

We encode all tx constraints in set S

- $S \subseteq \{0,1\}^n$ is the set of allowable transactions
  - Network constraints, *e.g.*, $Ax \leq b$
  - Interactions among txns, *e.g.*, bidders for MEV opportunity

We encode all tx constraints in set S

- $S \subseteq \{0,1\}^n$ is the set of allowable transactions
  - Network constraints, *e.g.*, $Ax \leq b$
  - Interactions among txns, *e.g.*, bidders for MEV opportunity
- We consider the convex hull of S: $\mathrm{conv}(S)$
  - This means j can be 'partially included'
  - $x_j \in (0,1) \implies$ tx j included after roughly $1/x_j$ blocks

Transaction producers get utility from each included tx

- Tx producers = users + validators

Transaction producers get utility from each included tx

- Tx producers = users + validators
- If tx j is included, tx producers get (joint) utility q_j

Transaction producers get utility from each included tx

- Tx producers = users + validators
- If tx j is included, tx producers get (joint) utility q_j
- We almost never know q in practice

Transaction producers get utility from each included tx

- Tx producers = users + validators
- If tx j is included, tx producers get (joint) utility q_j
- We almost never know q in practice
- But we will see that the network does not need to know q!

The resource allocation problem

$$\begin{array}{ll} \text{maximize} & q^T x - \ell(y) \\ \text{subject to} & y = Ax \\ & x \in \mathrm{conv}(S). \end{array}$$

The resource allocation problem

$$\begin{array}{ll} \text{maximize} & q^T x - \ell(y) \\ \text{subject to} & y = Ax \\ & x \in \mathrm{conv}(S). \end{array}$$

- **Objective:** Maximize utility of included txns minus the loss incurred by the network

The resource allocation problem

$$\begin{array}{ll} \text{maximize} & q^T x - \ell(y) \\ \text{subject to} & y = Ax \\ & x \in \mathrm{conv}(S). \end{array}$$

- **Objective:** Maximize utility of included txns minus the loss incurred by the network
- **Constraints:** Utilization y is resource usage of included txns, and x is in the set of allowable txns $S \subseteq \{0, 1\}^n$ (can be very complex/hard to solve!)

The resource allocation problem

$$\begin{array}{ll} \text{maximize} & q^T x - \ell(y) \\ \text{subject to} & y = Ax \\ & x \in \mathrm{conv}(S). \end{array}$$

- But network designer cannot solve this in practice!
  - Doesn't decide which txns are in a block (block builders do this)
  - Doesn't know utilities q
  - Cannot include fractional txns ($x_i \in (0, 1)$)

Outline

Duality theory: relaxing constraints to penalties

$$\begin{array}{ll} \text{maximize} & q^T x - \ell(y) \\ \text{subject to} & y = Ax \\ & x \in \mathrm{conv}(S). \end{array}$$

- Network designer cares about utilization y, based on txns x
- Block builders only care about which txns they can include

Duality theory: relaxing constraints to penalties

$$\begin{array}{ll} \text{maximize} & q^T x - \ell(y) \\ \text{subject to} & y = Ax \\ & x \in \mathrm{conv}(S) \end{array}$$

- Network designer cares about utilization y, based on txns x
- Block builders only care about which txns they can include
- We will 'decouple' utilization of network and that of tx producers

Duality theory: relaxing constraints to penalties

$$\begin{array}{ll} \text{maximize} & q^T x - \ell(y) \\ \text{subject to} & y = Ax \\ & x \in \mathrm{conv}(S) \end{array}$$

- Network designer cares about utilization y, based on txns x
- Block builders only care about which txns they can include
- We will 'decouple' utilization of network and that of tx producers
- Correctly set penalty $\rightarrow$ dual problem $=$ original problem & utilizations are equal

Dual decouples tx produces and network

- Problem is separable, so $g(p)$ decomposes into two easily interpretable terms:

$$g(p) = \underbrace{\ell^*(p)}_{\text{network}} + \underbrace{\sup_{x \in \mathrm{conv}(S)} (q - A^T p)^T x}_{\text{tx producers}}$$

Dual decouples tx produces and network

- Problem is separable, so $g(p)$ decomposes into two easily interpretable terms:

$$g(p) = \underbrace{\ell^*(p)}_{\text{network}} + \underbrace{\sup_{x \in \mathrm{conv}(S)} (q - A^T p)^T x}_{\text{tx producers}}$$

- Dual problem is to find the prices p that minimize $g(p)$

Dual decouples tx produces and network

- Problem is separable, so $g(p)$ decomposes into two easily interpretable terms:

$$g(p) = \underbrace{\ell^*(p)}_{\text{network}} + \underbrace{\sup_{x \in \mathrm{conv}(S)} (q - A^T p)^T x}_{\text{tx producers}}$$

- Dual problem is to find the prices p that minimize $g(p)$
- From before, p are the prices for violating prev. constraint $y = Ax$
  - Relaxing constraint to penalty $\rightarrow$ pay per unit violation

Dual decouples tx produces and network

- Problem is separable, so $g(p)$ decomposes into two easily interpretable terms:

$$g(p) = \underbrace{\ell^*(p)}_{\text{network}} + \underbrace{\sup_{x \in \mathrm{conv}(S)} (q - A^T p)^T x}_{\text{tx producers}}$$

- Dual problem is to find the prices p that minimize $g(p)$

- From before, p are the prices for violating prev. constraint $y = Ax$
  - Relaxing constraint to penalty $\to$ pay per unit violation

- Evaluating the 1st term is easy: can be done on chain! Let's look at the 2nd term

Second term: block building problem

- Maximize net utility (utility minus cost) subject to tx constraints

$$
\begin{array}{ll}
\text{maximize} & (q - A^T p)^T x \\
\text{subject to} & x \in \mathrm{conv}(S).
\end{array}
$$

Second term: block building problem

- Maximize net utility (utility minus cost) subject to tx constraints

$$\begin{array}{ll} \text{maximize} & (q - A^T p)^T x \\ \text{subject to} & x \in \mathbf{conv}(S). \end{array}$$

- Same optimal value if we use S instead of $\mathbf{conv}(S)$

Second term: block building problem

- Maximize net utility (utility minus cost) subject to tx constraints

$$\begin{array}{ll} \text{maximize} & (q - A^T p)^T x \\ \text{subject to} & x \in \mathrm{conv}(S). \end{array}$$

- Same optimal value if we use S instead of $\mathrm{conv}(S)$
- Exact problem solved by block producers! $\rightarrow$ Network can observe $x^\star$

What do we get at optimality?

- Let $p^\star$ be a minimizer of $g(p)$, *i.e.*, prices are set optimally
- Assume the block building problem has optimal solution $x^\star$
- The optimality conditions are

$$\nabla g(p^\star) = y^\star - Ax^\star = 0$$

where $y^\star$ satisfies $\nabla \ell(y^\star) = p^\star$

Key results

1. Prices that minimize g charge the tx producers exactly the marginal costs faced by the network:

$$\nabla \ell(Ax^\star) = p^\star$$

Key results

1. Prices that minimize g charge the tx producers exactly the marginal costs faced by the network:

$$\nabla \ell(Ax^\star) = p^\star$$

2. These prices incentivize tx producers to include txns that maximize welfare generated $q^T x$ minus the network loss $\ell(Ax)$

Cool. So how do we minimize $g(p)$?

- We can compute the gradient:

$$\nabla g(p) = y^\star(p) - Ax^\star(p)$$

Cool. So how do we minimize $g(p)$?

- We can compute the gradient:

$$\nabla g(p) = y^\star(p) - Ax^\star(p)$$

- Network determines $y^\star(p)$ (computationally easy)

Cool. So how do we minimize $g(p)$?

- We can compute the gradient:

$$\nabla g(p) = y^\star(p) - Ax^\star(p)$$

- Network determines $y^\star(p)$ (computationally easy)
- Network observes $x^\star(p)$ from previous block (block building problem soln)

Cool. So how do we minimize $g(p)$?

- We can compute the gradient:

$$\nabla g(p) = y^\star(p) - Ax^\star(p)$$

- Network determines $y^\star(p)$ (computationally easy)
- Network observes $x^\star(p)$ from previous block (block building problem soln)
- Then network applies favorite optimization method (*e.g.*, gradient descent)

$$p^{k+1} = p^k - \eta \nabla g(p^k)$$

Some simple examples:

Update rule

$$p^{k+1} = p^k - \eta(b^\star - Ax^\star)$$

Loss function

$$\ell(y) = \begin{cases} 0 & y = b^\star \\ \infty & \text{otherwise} \end{cases}$$

Some simple examples:

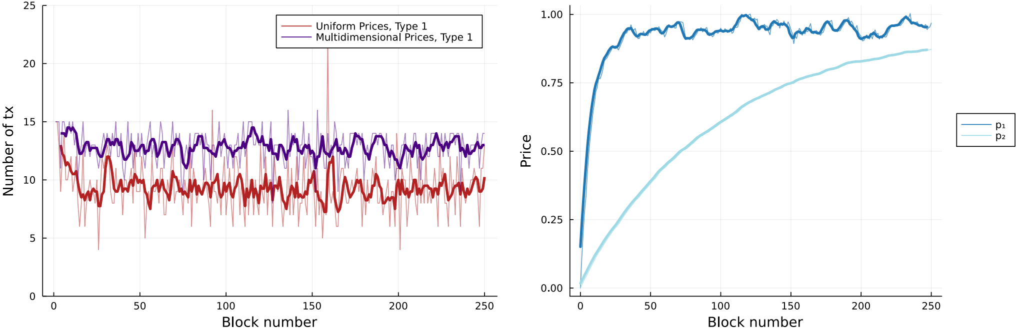
Update rule	Loss function
$p^{k+1} = p^k - \eta(b^\star - Ax^\star)$	$\ell(y) = \begin{cases} 0 & y = b^\star \\ \infty & \text{otherwise} \end{cases}$
$p^{k+1} = \left(p^k - \eta(b^\star - Ax^\star)\right)_+$	$\ell(y) = \begin{cases} 0 & y \leq b^\star \\ \infty & \text{otherwise} \end{cases}$

Some simple examples:

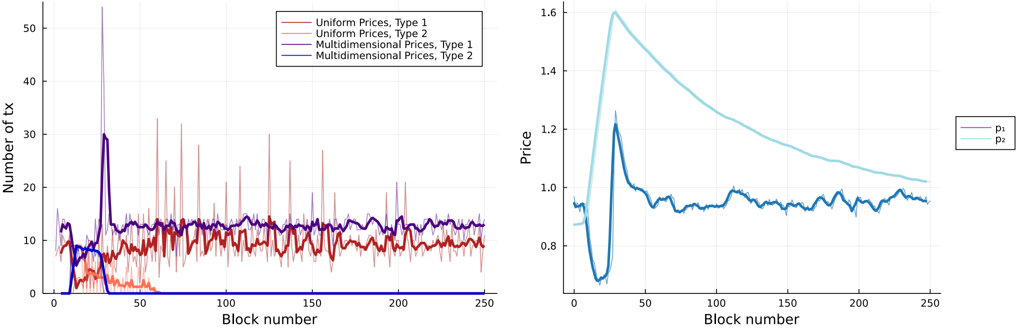
Update rule	Loss function
$p^{k+1} = p^k - \eta(b^\star - Ax^\star)$	$\ell(y) = \begin{cases} 0 & y = b^\star \\ \infty & \text{otherwise} \end{cases}$
$p^{k+1} = \left(p^k - \eta(b^\star - Ax^\star)\right)_+$	$\ell(y) = \begin{cases} 0 & y \leq b^\star \\ \infty & \text{otherwise} \end{cases}$
$p_i^{k+1} = p_i^k \cdot \exp\left(\eta(Ax - b^\star)_i\right)$	See paper, appendix C

Outline

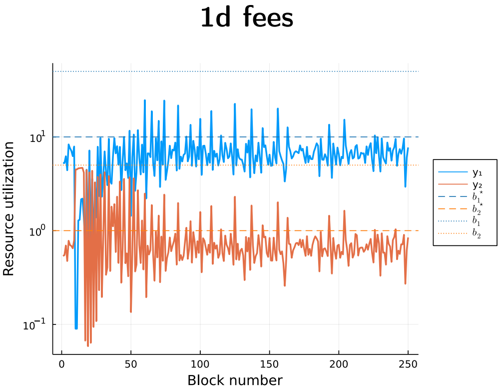
Multidimensional fees increase throughput



Even when the tx distribution shifts



And resource utilitaztion better tracks targets



Conclusion: choose your objective, not the update rule!

Choice of **objective function** by network designer yields an "optimal" price update rule via our optimization-based framework

For more info, check out our paper!

Paper

Thank you!

Theo Diamandis

Bain Capital Crypto & MIT

@theo_diamandis