Designing (Optimal) Multi-dimensional Blockchain Fees

Theo Diamandis
work with Guillermo Angeris, Alex Evans, Tarun Chitra, and Ciamac Moallemi

SBC 2024

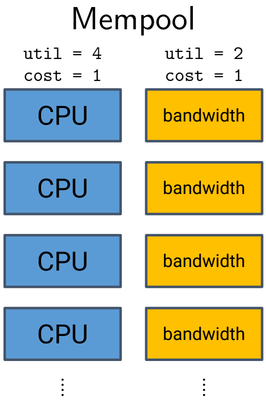
Fee markets with fixed relative prices are inefficient

Fee markets with fixed relative prices are inefficient

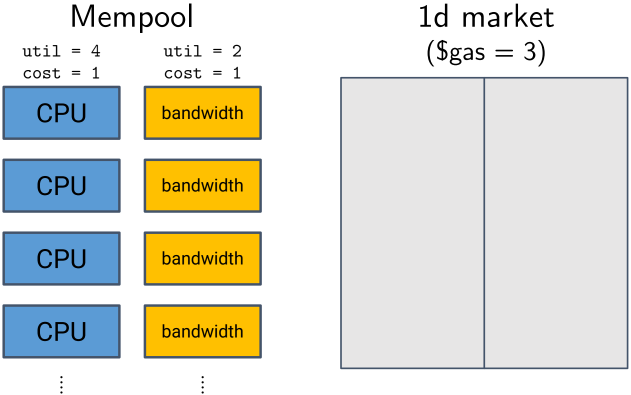
This talk: a framework to optimally set multi-dimensional fees for congestion control

Outline

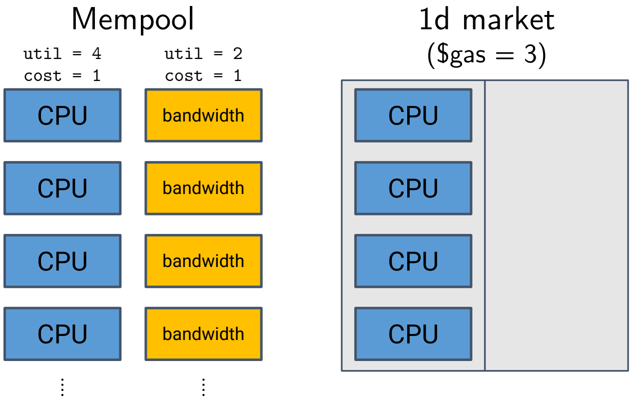
Fixed relative prices limit throughput



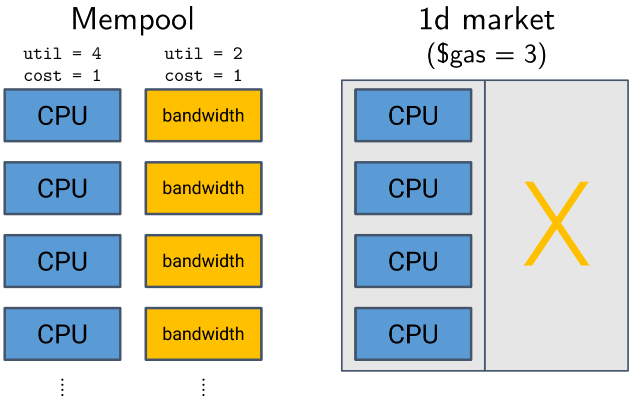
Fixed relative prices limit throughput



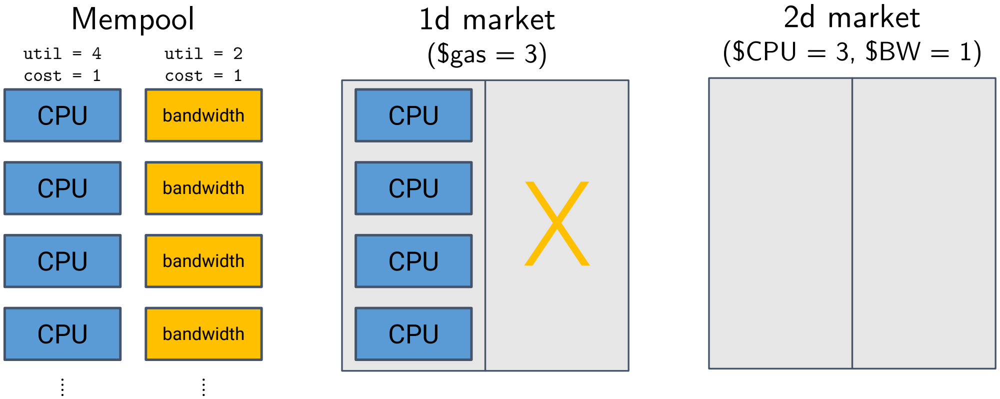
Fixed relative prices limit throughput



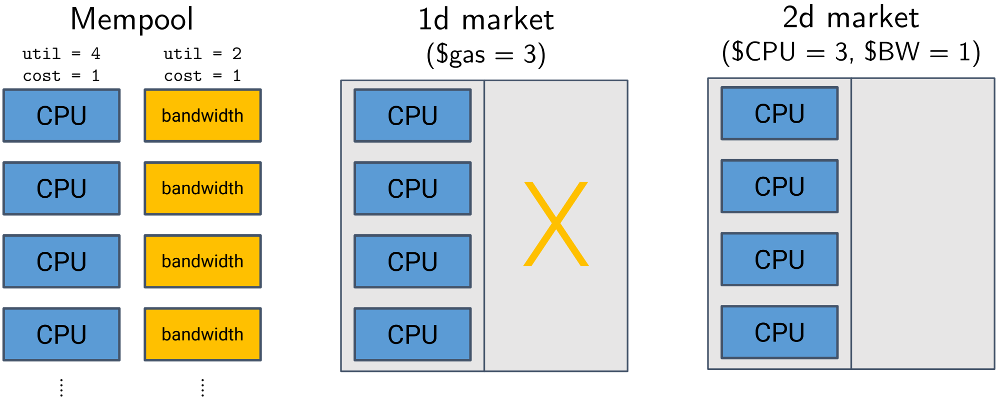
Fixed relative prices limit throughput



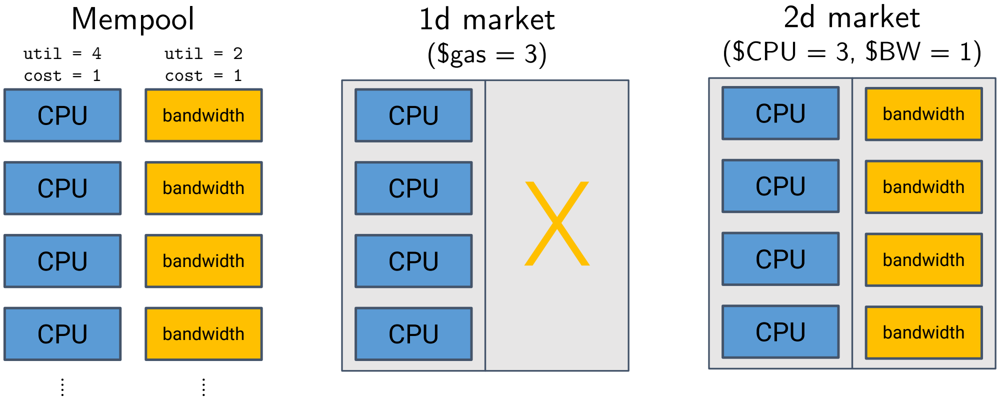
Fixed relative prices limit throughput



Fixed relative prices limit throughput



Fixed relative prices limit throughput



Orthogonal resources should be priced separately

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, *i.e.*, this is the base fee)

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

But how do we determine prices?

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

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 (multidimensional EIP-4844):

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

But how do we determine prices?

- We want a few properties:
  - $(Ax)_i = b_i^\star \rightarrow$ no update
  - $(Ax)_i > b_i^\star \rightarrow p_i$ increases
  - $(Ax)_i < b_i^\star \rightarrow p_i$ decreases
- Proposal (multidimensional EIP-4844):

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

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.*:

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

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

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

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

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

The resource allocation problem

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

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

The resource allocation problem

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

- **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

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

- 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

The resource allocation problem

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

- 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
- Goal: set prices so that this problem is solved optimally on average

Outline

Duality theory: relaxing constraints to penalties

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

- 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

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

- 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

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

- 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 producers and network

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

Dual decouples tx producers and network

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

Dual decouples tx producers and network

- Dual problem is to find the prices p that minimize dual function $g(p)$
- From before, p are the prices for violating prev. constraint $y = Ax$
  - Relaxing constraint to penalty $\rightarrow$ pay per unit violation
- Problem is separable, so $g(p)$ decomposes into two easily interpretable terms:

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

- Evaluating the 1st term is easy (conjugate function). Let's look at the 2nd...

Second term: block building problem

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

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

Second term: block building problem

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

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

- Exact problem solved by block producers! $\to$ 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 that 'supply' matches 'demand'

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

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)

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

Outline

Why are transactions so expensive?

Transactions and resources

The resource allocation problem

Setting prices via duality

Does gradient descent Just Work™?

Let's play a game

- Two players: network and block producers. At block t:

Let's play a game

- Two players: network and block producers. At block t:
  1. Network chooses prices p^t

Let's play a game

- Two players: network and block producers. At block t:
  1. Network chooses prices p^t
  2. Users submit txns (with utilities q^t, resources A^t), possibly adversarially!

Let's play a game

- Two players: network and block producers. At block t:
  1. Network chooses prices p^t
  2. Users submit txns (with utilities q^t, resources A^t), possibly adversarially!
  3. Network receives payoff $g_t(p^t)$ (from duality)

Let's play a game

- Two players: network and block producers. At block t:
  1. Network chooses prices p^t
  2. Users submit txns (with utilities q^t, resources A^t), possibly adversarially!
  3. Network receives payoff $g_t(p^t)$ (from duality)
- Metric: *regret* of the network ('welfare gap')

$$\frac{1}{T}\left(\sum_{t=1}^{T} g_t(p^t) - \min_{p^\star} \sum_{t=1}^{T} g_t(p^\star)\right)$$

- Interpretation: difference between dynamic update rule and the best fixed prices $p^\star$
  - Knowing $p^\star$ requires omniscience: assumes you know all future txns!

Main result:

- Gradient descent price update with fixed step size $\eta = M/B\sqrt{T}$ gives

$$\frac{1}{T}\left(\sum_{t=1}^{T} g_t(p^t) - \min_{p^\star} \sum_{t=1}^{T} g_t(p^\star)\right) \leq \frac{4MB}{\sqrt{T}}$$

where B and M are constants.

Main result:

- Gradient descent price update with fixed step size $\eta = M/B\sqrt{T}$ gives

$$\frac{1}{T}\left(\sum_{t=1}^{T} g_t(p^t) - \min_{p^\star}\sum_{t=1}^{T} g_t(p^\star)\right) \leq \frac{4MB}{\sqrt{T}}$$

  where B and M are constants.

- Regret is $O(1/\sqrt{T})$ and goes to 0 as T gets large!

Main result:

- Gradient descent price update with fixed step size $\eta = M/B\sqrt{T}$ gives

$$\frac{1}{T}\left(\sum_{t=1}^{T} g_t(p^t) - \min_{p^\star}\sum_{t=1}^{T} g_t(p^\star)\right) \leq \frac{4MB}{\sqrt{T}}$$

  where B and M are constants.

- Regret is $O(1/\sqrt{T})$ and goes to 0 as T gets large!

- This result does not assume any model or notion of stochasticity
  - No assumption that there exists a particular distribution for txns
  - Agents mess with your protocol! Need adversarial bounds.

Main result:

- Gradient descent price update with fixed step size $\eta = M/B\sqrt{T}$ gives

$$\frac{1}{T}\left(\sum_{t=1}^{T} g_t(p^t) - \min_{p^\star} \sum_{t=1}^{T} g_t(p^\star)\right) \le \frac{4MB}{\sqrt{T}}$$

  where B and M are constants.

- Regret is $O(1/\sqrt{T})$ and goes to 0 as T gets large!

- This result does not assume any model or notion of stochasticity
  - No assumption that there exists a particular distribution for txns
  - Agents mess with your protocol! Need adversarial bounds.

- Online convex optimization shines in this setting (common in blockchains!)
  - Note: does not require that we ever converge to the optimal fixed price $p^\star$

Main result II:

- This scheme is optimal in a certain sense: zero regret on average (with correct step size)
  - Directly from basic online convex optimization results
  - There exists a (stochastic) adversary that matches this bound
  - If utilization is stochastic, prices converge to clearing price

Main result II:

- This scheme is optimal in a certain sense: zero regret on average (with correct step size)
  - Directly from basic online convex optimization results
  - There exists a (stochastic) adversary that matches this bound
  - If utilization is stochastic, prices converge to clearing price
- This result is stronger than 'traditional' game theoretic results:
  - Does not require the adversary to be rational
  - Only requires adversary to be bounded (*e.g.*, have a budget or max block size)
  - Does not require playing to an equilibrium

Some simple examples:

Update rule

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

Loss function

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

Some simple examples:

Update rule	Loss function
$p^{t+1} = p^t - \eta(b^\star - Ax^\star)$	$\ell(y) = \begin{cases} 0 & y = b^\star \\ \infty & \text{otherwise} \end{cases}$
$p_i^{t+1} = p_i^t \cdot \exp\left(\eta(Ax - b^\star)_i\right)$	above with mirror descent

Some simple examples:

Update rule	Loss function
$p^{t+1} = p^t - \eta(b^\star - Ax^\star)$	$\ell(y) = \begin{cases} 0 & y = b^\star \\ \infty & \text{otherwise} \end{cases}$
$p_i^{t+1} = p_i^t \cdot \exp\left(\eta(Ax - b^\star)_i\right)$	above with mirror descent
$p^{t+1} = \left(p^t - \eta(b^\star - Ax^\star)\right)_+$	$\ell(y) = \begin{cases} 0 & y \leq b^\star \\ \infty & \text{otherwise} \end{cases}$

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

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

No difference between 'correctly' fixing prices with oracle knowledge of future and using online gradient descent algorithm.

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

No difference between 'correctly' fixing prices with oracle knowledge of future and using online gradient descent algorithm.

These results hold without assumptions of demand distributions or of market-clearing prices!

Extensions and future work

- What should the resources be?
  - How do you optimally trade-off between complexity & ease of use?
  - How do you design a loss function for desired performance characteristics?
  - Implementations by Avalanche and Penumbra teams may provide insights
  - Related to blob pricing and L1 vs L2 gas on rollups

Extensions and future work

- What should the resources be?
  - How do you optimally trade-off between complexity & ease of use?
  - How do you design a loss function for desired performance characteristics?
  - Implementations by Avalanche and Penumbra teams may provide insights
  - Related to blob pricing and L1 vs L2 gas on rollups
- What update rules are most useful? [Convergence behavior vs. complexity]

Extensions and future work

- What should the resources be?
  - How do you optimally trade-off between complexity & ease of use?
  - How do you design a loss function for desired performance characteristics?
  - Implementations by Avalanche and Penumbra teams may provide insights
  - Related to blob pricing and L1 vs L2 gas on rollups
- What update rules are most useful? [Convergence behavior vs. complexity]
- Likely relevant for many similar mechanisms...

For more info, check out our paper!

Paper

Thank you!

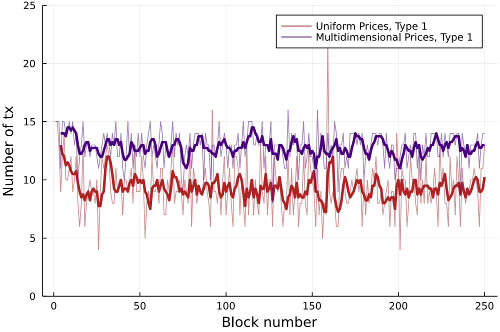
Theo Diamandis

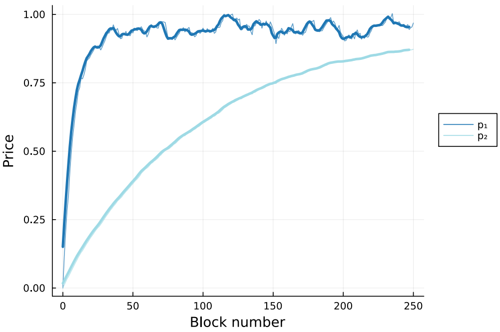
`tdiamand@mit.edu`

@theo_diamandis

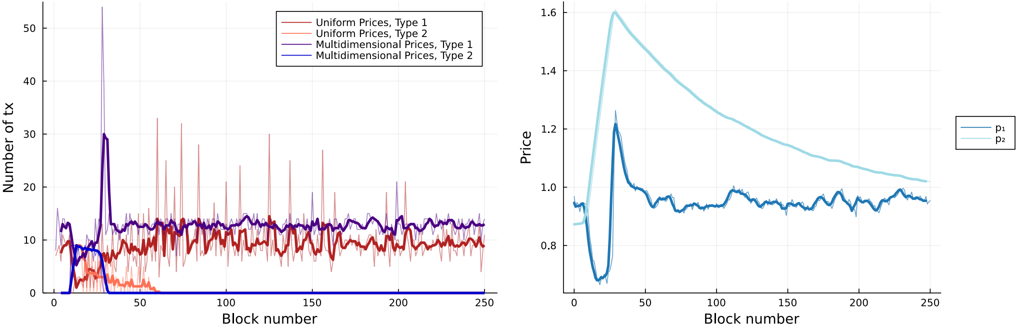
Appendix

Multidimensional fees increase throughput





Even when the tx distribution shifts



And resource utilitaztion better tracks targets