An Efficient Algorithm for Optimal Routing Through Constant Function Market Makers

Theo Diamandis (MIT), Max Resnick (Rook), Tarun Chitra (Gauntlet), and Guillermo Angeris (Bain Capital Crypto)

Financial Cryptography 2023

tl;dr: It's all convex optimization

Routing (multi-DEX swaps, etc.) is a convex[1] optimization problem, so it can be *very efficiently* solved to *verifiable* global optimality.

[1] when we ignore gas

tl;dr: It's all convex optimization

Routing (multi-DEX swaps, etc.) is a convex[1] optimization problem, so it can be *very efficiently* solved to *verifiable* global optimality.

This talk: the 'very efficiently' part

[1] when we ignore gas

Outline

Review: Constant Function Market Makers

- Most DEXs are implemented as *constant function market makers* (CFMMs)
- CFMMs are defined by their trading function $\varphi : \mathbb{R}^n_+ \to \mathbb{R}$

Review: Constant Function Market Makers

- Most DEXs are implemented as *constant function market makers* (CFMMs)
- CFMMs are defined by their trading function $\varphi : \mathbb{R}^n_+ \to \mathbb{R}$
- Maps reserves $R \in \mathbb{R}^n_+$ to a real number

Review: Constant Function Market Makers

- Most DEXs are implemented as *constant function market makers* (CFMMs)
- CFMMs are defined by their trading function $\varphi : \mathbb{R}^n_+ \to \mathbb{R}$
- Maps reserves $R \in \mathbb{R}^n_+$ to a real number
- Is concave and increasing

Review: Constant Function Market Makers

- Most DEXs are implemented as *constant function market makers* (CFMMs)
- CFMMs are defined by their trading function $\varphi : \mathbb{R}^n_+ \to \mathbb{R}$
- Maps reserves $R \in \mathbb{R}^n_+$ to a real number
- Is concave and increasing
- Accepts trade $\Delta \to \Lambda$ if $\varphi(R + \gamma\Delta - \Lambda) \geq \varphi(R)$.

Most DEXs are CFMMs

- Geometric mean trading function (Balancer, Uniswap, etc...):

$$\varphi(R) = \left(\prod_{i=1}^{n} R_i^{w_i} \right)^{1/n}$$

where w_i are nonnegative weights that sum to 1.

Most DEXs are CFMMs

- Geometric mean trading function (Balancer, Uniswap, etc...):

$$\varphi(R) = \left(\prod_{i=1}^{n} R_i^{w_i}\right)^{1/n}$$

where w_i are nonnegative weights that sum to 1.

- Curve:

$$\varphi(R) = \mathbf{1}^T R - \alpha \prod_{i=1}^{n} R_i^{-1}$$

where $\alpha > 0$.

Problem: fragmented liquidity

- Most CFMMs are swap pools (trade asset A for B)

Problem: fragmented liquidity

- Most CFMMs are swap pools (trade asset A for B)
- For n assets, can have $\sim n^2$ swap pools

Problem: fragmented liquidity

- Most CFMMs are swap pools (trade asset A for B)
- For n assets, can have $\sim n^2$ swap pools
- If I want to trade ETH for DAI, there are many routes I can take:
  - $\mathrm{ETH} \to \mathrm{DAI}$
  - $\mathrm{ETH} \to \mathrm{USDC} \to \mathrm{DAI}$
  - $\mathrm{ETH} \to \mathrm{wBTC} \to \mathrm{DAI}$
  - ...

Problem: fragmented liquidity

- Most CFMMs are swap pools (trade asset A for B)
- For n assets, can have $\sim n^2$ swap pools
- If I want to trade ETH for DAI, there are many routes I can take:
  - $\mathrm{ETH} \rightarrow \mathrm{DAI}$
  - $\mathrm{ETH} \rightarrow \mathrm{USDC} \rightarrow \mathrm{DAI}$
  - $\mathrm{ETH} \rightarrow \mathrm{wBTC} \rightarrow \mathrm{DAI}$
  - ...
- **Problem:** How to split trade?

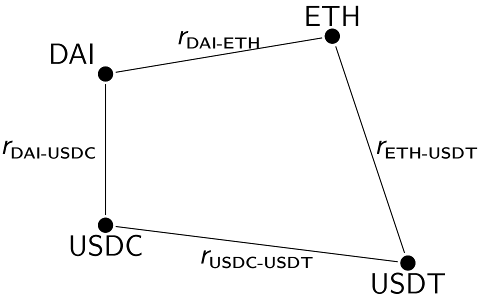
Problem: fragmented liquidity

- Most CFMMs are swap pools (trade asset A for B)
- For n assets, can have $\sim n^2$ swap pools
- If I want to trade ETH for DAI, there are many routes I can take:
  - $\mathrm{ETH} \to \mathrm{DAI}$
  - $\mathrm{ETH} \to \mathrm{USDC} \to \mathrm{DAI}$
  - $\mathrm{ETH} \to \mathrm{wBTC} \to \mathrm{DAI}$
  - ...
- **Problem:** How to split trade? **Solution:** build a router

Outline

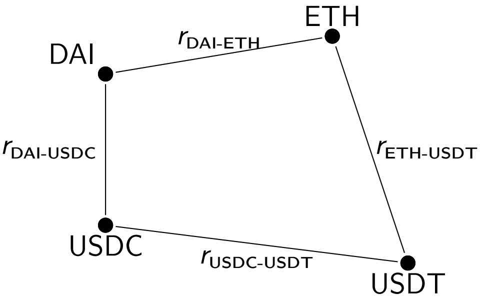
Networks of CFMMs

- Common representation: undirected graph with exchange rates



Networks of CFMMs

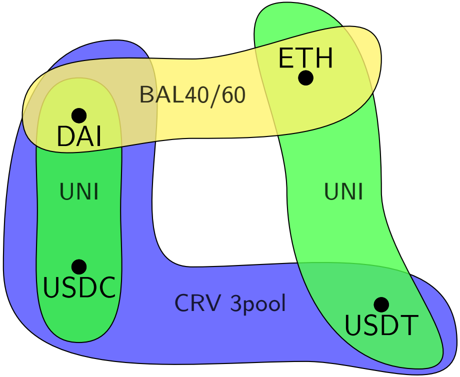
- Common representation: undirected graph with exchange rates



- But how to handle three pools? Multiple CFMMs?

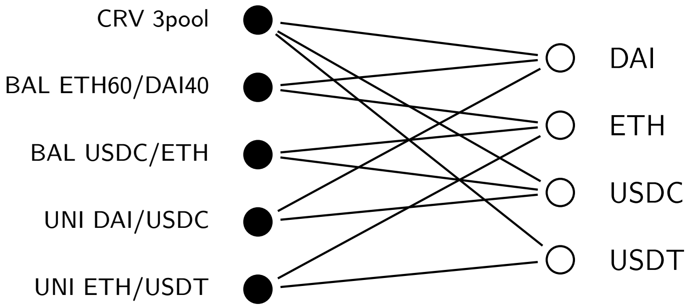
Networks of CFMMs

- The token-CFMM network is a hypergraph: edges can connect more than 2 vertices



Networks of CFMMs

- The token-CFMM network is a hypergraph: edges can connect more than 2 vertices



- Good bookkeeping is essential!

Networks of CFMMs

- Label the tokens $1, 2, \ldots, n$
- Label the CFMMs $1, 2, \ldots, m$

Networks of CFMMs

- Label the tokens $1, 2, \ldots, n$
- Label the CFMMs $1, 2, \ldots, m$
- CFMM i has n_i tokens, with *local* indices $1, \ldots, n_i$

Networks of CFMMs

- Label the tokens $1, 2, \ldots, n$
- Label the CFMMs $1, 2, \ldots, m$
- CFMM i has n_i tokens, with *local* indices $1, \ldots, n_i$
- Trade (Δ_i, Λ_i) with CFMM i, where $\Delta_i, \Lambda_i \in \mathbb{R}_+^{n_i}$

Networks of CFMMs

- Label the tokens $1, 2, \ldots, n$
- Label the CFMMs $1, 2, \ldots, m$
- CFMM i has n_i tokens, with *local* indices $1, \ldots, n_i$
- Trade (Δ_i, Λ_i) with CFMM i, where $\Delta_i, \Lambda_i \in \mathbb{R}_+^{n_i}$
- Trade accepted if $\varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \geq \varphi_i(R_i)$

Networks of CFMMs

- Matrices A_i map token's *local* index in CFMM i to global index, *e.g.*, ,

Token	Local Index	Global Index
DAI	1	3
ETH	2	1

$$A_i \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

Networks of CFMMs

- Matrices A_i map token's *local* index in CFMM i to global index, *e.g.*, ,

Token	Local Index	Global Index
DAI	1	3
ETH	2	1

$$A_i \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

- The overall net trade with the network is

$$\Psi = \sum_{i=1}^{m} A_i(\Lambda_i - \Delta_i)$$

Simplifying the Model

- We ignore gas fees
- We don't worry about transaction execution ordering
- We can return to these later...

The Routing Problem

- We choose some utility function $U(\Psi)$ of the net trade Ψ

The Routing Problem

- We choose some utility function $U(\Psi)$ of the net trade Ψ
- The optimal routing problem is then

$$
\begin{array}{ll}
\text{maximize} & U(\Psi) \\
\text{subject to} & \Psi = \sum_{i=1}^{m} A_i(\Lambda_i - \Delta_i) \\
& \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1, \ldots, m \\
& \Delta_i \geq 0, \quad \Lambda_i \geq 0, \quad i = 1, \ldots, m.
\end{array}
$$

The Routing Problem

- We choose some utility function $U(\Psi)$ of the net trade Ψ
- The optimal routing problem is then

$$
\begin{array}{ll}
\text{maximize} & U(\Psi) \\
\text{subject to} & \Psi = \sum_{i=1}^{m} A_i(\Lambda_i - \Delta_i) \\
& \varphi_i(R_i + \gamma_i\Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1, \ldots, m \\
& \Delta_i \geq 0, \quad \Lambda_i \geq 0, \quad i = 1, \ldots, m.
\end{array}
$$

The Routing Problem

- We choose some utility function $U(\Psi)$ of the net trade Ψ
- The optimal routing problem is then

$$\begin{array}{ll} \text{maximize} & U(\Psi) \\ \text{subject to} & \Psi = \sum_{i=1}^{m} A_i(\Lambda_i - \Delta_i) \\ & \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1, \ldots, m \\ & \Delta_i \geq 0, \quad \Lambda_i \geq 0, \quad i = 1, \ldots, m. \end{array}$$

- Each individual CFMM is defined by trading constraints

$U(\Psi)$ encodes what we want to do

- Utility function U gives our satisfaction with the net trade
- We can also use U to encode constraints

$U(\Psi)$ encodes what we want to do

- Utility function U gives our satisfaction with the net trade
- We can also use U to encode constraints
- **Arbitrage:** Find the most profitable nonnegative net trade

$$U(\Psi) = c^T \Psi - \mathbb{I}(\Psi \geq 0)$$

  - The vector c is a positive price vector
  - Indicator function $\mathbb{I}(\Psi \geq 0) = 0$ if $\Psi \geq 0$ and $+\infty$ otherwise

Swaps: trade token i for j

- Goal: maximize output of token j given fixed input of token i
- Constraints: input exactly Δ^i of token i and only get token j

$$U(\Psi) = \Psi_j - \mathbb{I}(\Psi_{[n]\setminus\{i,j\}} = 0,\ \Psi_i = -\Delta^i)$$

Swaps: trade token i for j

- Goal: maximize output of token j given fixed input of token i
- Constraints: input exactly Δ^i of token i and only get token j

$$U(\Psi) = \Psi_j - \mathbb{I}(\Psi_{[n]\setminus\{i,j\}} = 0,\ \Psi_i = -\Delta^i)$$

- More generally, we can optimally purchase or liquidate a basket of tokens
- Capturing "arbitrage" opportunities as part of the swap

Outline

Duality provides an alternate view of the problem

- The primal problem: finding the optimal trades

Duality provides an alternate view of the problem

- The primal problem: finding the optimal trades
- The dual problem: finding the optimal prices

Duality provides an alternate view of the problem

- The primal problem: finding the optimal trades
- The dual problem: finding the optimal prices
- Idea: your utility function induces personal "shadow" prices (marginal utilities) at which you value each token

Duality provides an alternate view of the problem

- The primal problem: finding the optimal trades
- The dual problem: finding the optimal prices
- Idea: your utility function induces personal "shadow" prices (marginal utilities) at which you value each token
- Given these prices, you can arbitrage each CFMM independently & in parallel

Duality provides an alternate view of the problem

- The primal problem: finding the optimal trades
- The dual problem: finding the optimal prices
- Idea: your utility function induces personal "shadow" prices (marginal utilities) at which you value each token
- Given these prices, you can arbitrage each CFMM independently & in parallel
- Strong duality $\implies$ dual problem has the same optimal value

Duality provides an alternate view of the problem

- The primal problem: finding the optimal trades
- The dual problem: finding the optimal prices
- Idea: your utility function induces personal "shadow" prices (marginal utilities) at which you value each token
- Given these prices, you can arbitrage each CFMM independently & in parallel
- Strong duality $\implies$ dual problem has the same optimal value
- Strong duality $\implies$ certificate of optimality (very cheap to check)

The dual problem is much easier to solve

- The dual problem is

$$\text{minimize} \quad g(\nu) = (-U)^*(-\nu) + \sum_{i=1}^m \mathsf{arb}_i(A_i^T \nu)$$

The dual problem is much easier to solve

- The dual problem is

$$\text{minimize} \quad g(\nu) = (-U)^*(-\nu) + \sum_{i=1}^m \mathrm{arb}_i(A_i^T \nu)$$

- The conjugate function is typically easy to evaluate

The dual problem is much easier to solve

- The dual problem is

$$\text{minimize} \quad g(\nu) = (-U)^*(-\nu) + \sum_{i=1}^m \mathrm{arb}_i(A_i^T \nu)$$

- The conjugate function is typically easy to evaluate
- $\mathrm{arb}_i(A_i^T \nu)$ is the optimal arb on CFMM i with global token prices ν

$$\begin{array}{ll} \text{maximize} & (A_i^T \nu)^T (\Lambda_i - \Delta_i) \\ \text{subject to} & \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \geq \varphi_i(R_i) \\ & \Delta_i \geq 0, \quad \Lambda_i \geq 0 \end{array}$$

The dual problem is much easier to solve

- The dual problem is

$$\text{minimize} \quad g(\nu) = (-U)^*(-\nu) + \sum_{i=1}^{m} \mathrm{arb}_i(A_i^T \nu)$$

- The conjugate function is typically easy to evaluate
- $\mathrm{arb}_i(A_i^T \nu)$ is the optimal arb on CFMM i with global token prices ν
- This is an unconstrained convex problem $\implies$ fast to solve!

The dual problem is much easier to solve

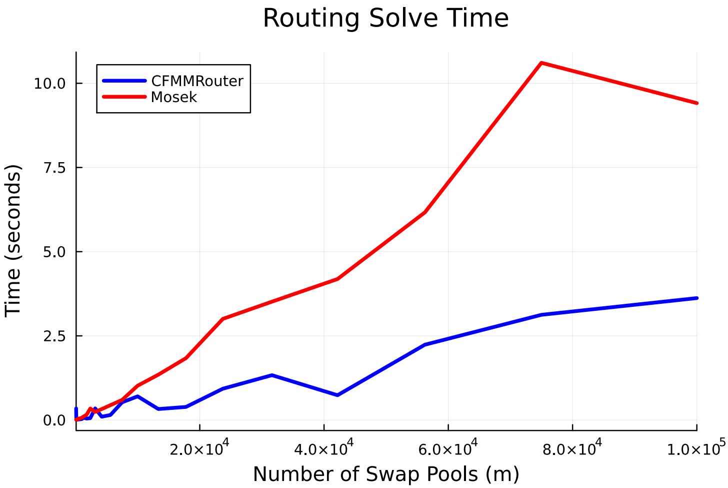
- The dual problem is

$$\text{minimize} \quad g(\nu) = (-U)^*(-\nu) + \sum_{i=1}^m \mathrm{arb}_i(A_i^T \nu)$$

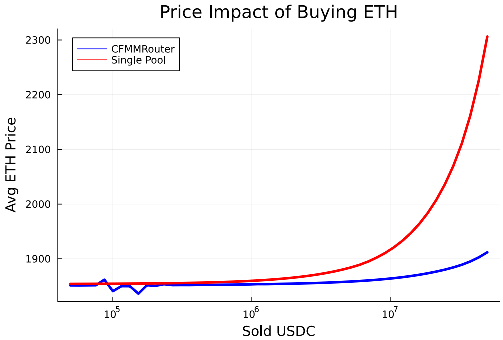
- The conjugate function is typically easy to evaluate
- $\mathrm{arb}_i(A_i^T \nu)$ is the optimal arb on CFMM i with global token prices ν
- This is an unconstrained convex problem $\implies$ fast to solve!
- To add a DEX, **only need to define this arbitrage function**

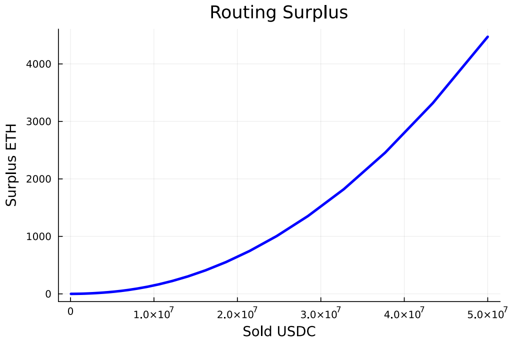
Outline

Our solver `CFMMRouter` is faster than commercial convex solvers

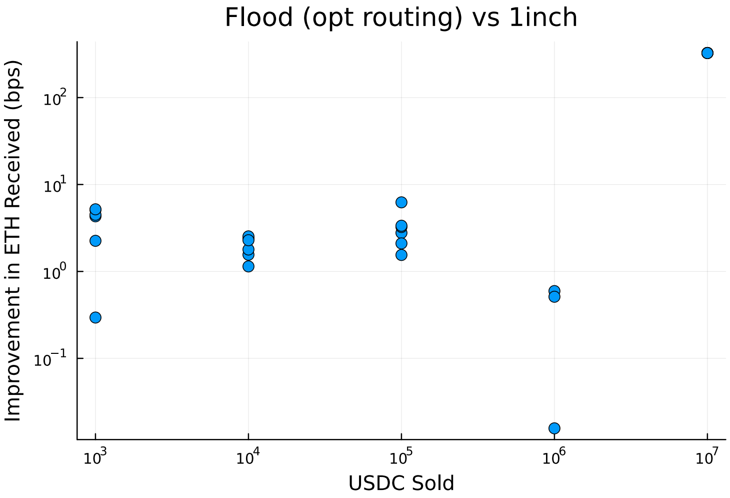


We see way less price impact for large txns





And it beats 1inch in production on Arbitrum (flood.bid)



Routing package on Github: `CFMMRouter.jl`

Flood in beta on Arbitrum: `flood.bid`

Outline

Summary

- Routing with no gas fees is a convex optimization problem

Summary

- Routing with no gas fees is a convex optimization problem
- This means it can be solved quickly to global optimality

Summary

- Routing with no gas fees is a convex optimization problem
- This means it can be solved quickly to global optimality
- And we can prove a feasible point is optimal

Summary

- Routing with no gas fees is a convex optimization problem
- This means it can be solved quickly to global optimality
- And we can prove a feasible point is optimal
- We construct an efficient algorithm using convex duality

Summary

- Routing with no gas fees is a convex optimization problem
- This means it can be solved quickly to global optimality
- And we can prove a feasible point is optimal
- We construct an efficient algorithm using convex duality
- This algorithm is implemented in `CFMMRouter.jl`

Future work includes expanding this framework

- Routing with gas fees (nonconvex—need good heuristics)
- Routing through liquidations
- Routing with probabilistic constraints when TXs may fail (*e.g.*, cross-chain)

For more info, check out our paper & **`CFMMRouter.jl`**

Paper

Thank you!

Theo Diamandis

`tdiamand@mit.edu`

Appendix

Optimality conditions

For the primal problem

$$
\begin{array}{ll}
\text{maximize} & U(\Psi) \\
\text{subject to} & \Psi = \sum_{i=1}^{m} A_i(\Lambda_i - \Delta_i) \\
& \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1, \ldots, m \\
& \Delta_i \geq 0, \quad \Lambda_i \geq 0, \quad i = 1, \ldots, m
\end{array}
$$

The optimality conditions are

$$
\lambda_i \gamma_i \nabla \varphi_i(R_i + \gamma_i \Delta_i^\star - \Lambda_i^\star) \leq A_i^T \nu^\star \leq \lambda_i \nabla \varphi_i(R_I + \gamma_i \Delta_i^\star - \Lambda_i^\star), \qquad i = 1, \ldots, m
$$

Routing with gas fees

- Gas cost for CFMM i is q_i
- New variable $\eta \in \{0, 1\}^m$
- $\eta_i = 1$ if CFMM i is used in the trade

Routing with gas fees

- Gas cost for CFMM i is q_i
- New variable $\eta \in \{0, 1\}^m$
- $\eta_i = 1$ if CFMM i is used in the trade

$$
\begin{array}{ll}
\text{maximize} & U(\Psi) - q^T \eta \\
\text{subject to} & \Psi = \sum_{i=1}^m A_i(\Lambda_i - \Delta_i) \\
& \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1, \ldots, m \\
& \eta_i \Delta^{\max} \geq \Delta_i \geq 0, \quad \Lambda_i \geq 0, \quad i = 1, \ldots, m \\
& \eta \in \{0, 1\}^m
\end{array}
$$

Routing with gas fees

$$
\begin{array}{ll}
\text{maximize} & U(\Psi) - q^T \eta \\
\text{subject to} & \Psi = \sum_{i=1}^m A_i(\Lambda_i - \Delta_i) \\
& \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1, \dots, m \\
& \eta_i \Delta^{\max} \geq \Delta_i \geq 0, \quad \Lambda_i \geq 0, \quad i = 1, \dots, m \\
& \eta \in \{0, 1\}^m
\end{array}
$$

- **Issue:** this problem is nonconvex...

Routing with gas fees

$$
\begin{array}{ll}
\text{maximize} & U(\Psi) - q^T \eta \\
\text{subject to} & \Psi = \sum_{i=1}^m A_i(\Lambda_i - \Delta_i) \\
& \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1, \ldots, m \\
& \eta_i \Delta^{\max} \geq \Delta_i \geq 0, \quad \Lambda_i \geq 0, \quad i = 1, \ldots, m \\
& \eta \in \{0, 1\}^m
\end{array}
$$

- **Issue:** this problem is nonconvex...
- ...but we have good heuristics for this type of problem

One Heuristic: ℓ_1 penalty

- Use ℓ_1 norm to approximate cardinality of trade vectors Δ_i
- ℓ_1 norm: $\|x\|_1 = \sum_i |x_i|$

One Heuristic: ℓ_1 penalty

- Use ℓ_1 norm to approximate cardinality of trade vectors Δ_i
- ℓ_1 norm: $\|x\|_1 = \sum_i |x_i|$
- Approximate gas cost: $\sum_{i=1}^{m} q_i \|\Delta_i\|_1 / n_i$

One Heuristic: ℓ_1 penalty

- Use ℓ_1 norm to approximate cardinality of trade vectors Δ_i
- ℓ_1 norm: $\|x\|_1 = \sum_i |x_i|$
- Approximate gas cost: $\sum_{i=1}^m q_i \|\Delta_i\|_1 / n_i$

$$
\begin{array}{ll}
\text{maximize} & U(\Psi) - \sum_{i=1}^m q_i \|\Delta_i\|_1 / n_i \\
\text{subject to} & \Psi = \sum_{i=1}^m A_i(\Lambda_i - \Delta_i) \\
& \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1, \ldots, m \\
& \Delta_i \geq 0, \quad \Lambda_i \geq 0, \quad i = 1, \ldots, m
\end{array}
$$

One Heuristic: ℓ_1 penalty

- Use ℓ_1 norm to approximate cardinality of trade vectors Δ_i
- ℓ_1 norm: $\|x\|_1 = \sum_i |x_i|$
- Approximate gas cost: $\sum_{i=1}^m q_i \|\Delta_i\|_1 / n_i$

$$
\begin{array}{ll}
\text{maximize} & U(\Psi) - \sum_{i=1}^m q_i \|\Delta_i\|_1 / n_i \\
\text{subject to} & \Psi = \sum_{i=1}^m A_i(\Lambda_i - \Delta_i) \\
& \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1, \ldots, m \\
& \Delta_i \geq 0, \quad \Lambda_i \geq 0, \quad i = 1, \ldots, m
\end{array}
$$

How does Uniswap v3 fit in?

- **Answer 1:** if solving the dual, only need to define $\mathrm{arb}(\cdot)$
- This is relatively easy: simple algorithm & closed form solution within a tick

How does Uniswap v3 fit in?

- **Answer 1:** if solving the dual, only need to define $\mathrm{arb}(\cdot)$
- This is relatively easy: simple algorithm & closed form solution within a tick
- **Answer 2:** The φ constraint is a bit of a lie...

How does Uniswap v3 fit in?

- **Answer 1:** if solving the dual, only need to define $\mathrm{arb}(\cdot)$
- This is relatively easy: simple algorithm & closed form solution within a tick
- **Answer 2:** The φ constraint is a bit of a lie...
- Only need a convex reachable reserve set (or, equivalently, trading set):

$$\varphi(R + \gamma\Delta - \Lambda) \geq \varphi(R) \iff R + \gamma\Delta - \Lambda \in S(R)$$