Optimal Routing for Constant Function Market Makers

GUILLERMO ANGERIS, Stanford University
ALEX EVANS, Bain Capital Crypto
TARUN CHITRA, Gauntlet Networks
STEPHEN BOYD, Stanford University

We consider the problem of optimally executing an order involving multiple crypto-assets, sometimes called tokens, on a network of multiple constant function market makers (CFMMs). When we ignore the fixed cost associated with executing an order on a CFMM, this optimal routing problem can be cast as a convex optimization problem, which is computationally tractable. When we include the fixed costs, the optimal routing problem is a mixed-integer convex problem, which can be solved using (sometimes slow) global optimization methods, or approximately solved using various heuristics based on convex optimization. The optimal routing problem includes as a special case the problem of identifying an arbitrage present in a network of CFMMs, or certifying that none exists.

CCS Concepts: • **Mathematics of computing** → **Convex optimization**; • **Applied computing** → **Operations research**.

Additional Key Words and Phrases: convex optimization, optimal routing, decentralized finance, blockchains

ACM Reference Format:
Guillermo Angeris, Alex Evans, Tarun Chitra, and Stephen Boyd. 2022. Optimal Routing for Constant Function Market Makers. In *Proceedings of the 23rd ACM Conference on Economics and Computation (EC '22), July 11–15, 2022, Boulder, CO, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3490486.3538336

1 INTRODUCTION

Decentralized exchanges (DEXs) are a popular application of public blockchains that allow users to trade assets without the need for a trusted intermediary to facilitate the exchange. DEXs are typically implemented as *constant function market makers* (CFMMs) [5]. In CFMMs, liquidity providers contribute reserves of assets. Users can then trade against these reserves by tendering baskets of assets in exchange for other baskets. CFMMs use a simple rule for accepting trades: a trade is only valid if the value of a given function at the post-trade reserves (with a small adjustment to account for fees collected) is equal to the value at the pre-trade reserves. This function is called the *trading function* and gives CFMMs their name. A common example of a trading function is the constant product popularized by Uniswap [16], wherein a trade is only accepted if it preserves the product of the reserve amounts.

CFMMs have quickly become one of the most popular applications of public blockchains, facilitating several billion dollars of trading volume per day. As DEXs have grown in popularity, so have the number of CFMMs and assets offered, creating complexity for traders who simply want to maximize their utility for trading one basket of assets for another. As a result, several "DEX aggregators" have emerged to route orders across multiple CFMMs on behalf of users. These

ACM ISBN 978-1-4503-9150-4/22/07…$15.00
https://doi.org/10.1145/3490486.3538336

aggregators currently execute several billion dollars per month across all DEXs on Ethereum [1]. At the same time, CFMM platforms such as Uniswap offer software for routing orders across the subset of CFMMs they support [14].

While the properties of individual CFMMs have been studied extensively (see, *e.g.*, [4, 5]) it is more common for users to want to access liquidity on multiple CFMMs to minimize their total trading costs. In this paper, we study the optimal routing problem for CFMMs. We consider a user who can trade with multiple CFMMs in order to exchange one basket of assets for another and ask how one should perform such trades optimally. We show that, in the absence of fixed costs, the optimal routing problem can be formulated as a convex optimization problem, which is efficiently solvable. As an (important) sub-case, we demonstrate how to use this problem to identify arbitrage opportunies on a set of CFMMs. Our framework encompasses the routing problems considered in prior work [8, 12] and offers solutions in the more general case where users seek to trade any basket of assets for any other basket across any set of CFMMs whose trading functions are concave and not necessarily differentiable. When including transaction costs, we show that the optimal routing problem is a mixed-integer convex problem, which permits (potentially computationally intensive) global solutions as well as approximate solutions using heuristics based on convex optimization.

Outline. We describe the optimal routing problem in §2, ignoring the fixed transaction costs. In §3 we give the optimality conditions for the optimal routing problem, and give conditions under which the optimal action is to not trade at all. We give some examples of the optimal routing problem in §4, including as a special case the detection of arbitrage in the network. We give a simple numerical example in §5. In §6 we show how to add fixed transaction costs to the optimal routing problem, and briefly describe some exact and approximate solution approaches.

2 OPTIMAL ROUTING PROBLEM

Network of CFMMs. We consider a set of m CFMMs, denoted $i = 1, \dots, m$, each of which trades multiple tokens from among a universe of n tokens, labeled $j = 1 \dots, n$. (We note that a 'token,' following common convention, is better understood as a currency, rather than an individual unit of the currency.) We let n_i denote the number of tokens that CFMM i trades, with $2 \leq n_i \leq n$. We can think of the m CFMMs as the vertices of a network or hypergraph, with n hyper-edges, each corresponding to one of the n assets, adjacent to those CFMMs that trade it. Alternatively we can represent this as a bipartite graph, with one group of m vertices the CFMMs, and the other group of n vertices the tokens, with an edge between a CFMM and a token if the CFMM trades the token.

A simple example is illustrated as a bipartite graph in figure 1. In this network there are $m = 5$ CFMMs, which trade subsets of $n = 3$ tokens. CFMM 1 trades all three tokens, so $n_1 = 3$; the remaining 4 CFMMs each trade pairs of tokens, so $n_2 = \cdots = n_5 = 2$.

Global and local token indices. We use multiple indices to label the tokens. The *global* index uses the token labels $1, \dots, n$. CFMM i, which trades a subset of n_i of the n tokens, has its *local* token index, $j = 1, \dots, n_i$. To link the global and local indexes, we introduce matrices $A_i \in \mathbf{R}^{n \times n_i}$, $i = 1 \dots, m$, where $(A_i)_{jk} = 1$ if token k in the CFMM i's local index corresponds to global token index j, while $(A_i)_{jk} = 0$ otherwise.

As an example, consider the simple network shown in figure 1. CFMM 1 trades all three tokens, with its local indexing identical to the global indexing, so A_1 is the 3×3 identity matrix. As a more interesting example, consider CFMM 4, which trades two tokens, labeled 1 and 2 in the local

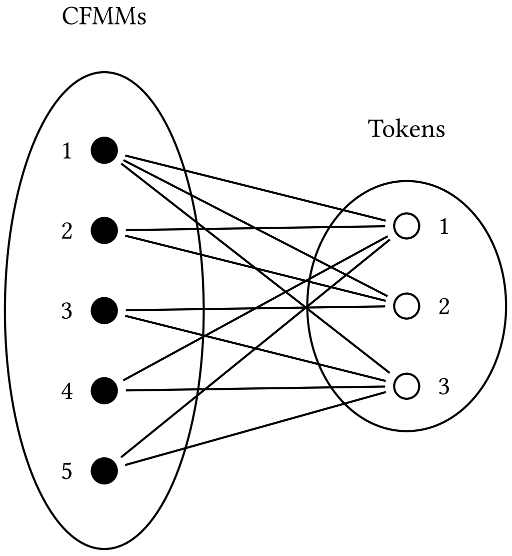


Fig. 1. Example CFMM network with $m = 5$ CFMMs and $n = 3$ tokens.

indexing, but 1 and 3 in the global indexing, so

$$A_4 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

CFMM tendered and received baskets. For CFMM i we denote the tendered and received baskets as $\Delta_i,\ \Lambda_i \in \mathbf{R}_+^{n_i}$. These are quantities of tokens (in the local indexing) that we give to, and receive from, CFMM i in a proposed trade.

CFMM trading semantics. The proposed trade (Δ_i, Λ_i) is valid and accepted by CFMM i provided

$$\varphi_i(R_i + \gamma\Delta_i - \Lambda_i) = \varphi_i(R_i),$$

where $\varphi : \mathbf{R}_+^{n_i} \to \mathbf{R}$ is the trading function, $R_i \in \mathbf{R}_+^{n_i}$ are the current reserves, and $\gamma_i \in (0, 1]$ is the trading fee for CFMM i. We will assume that the trading functions φ_i are concave and increasing functions. (For much more detail, see [4].)

Examples of trading functions are the sum function, with

$$\varphi_i(R) = R_1 + \cdots + R_{n_i},$$

the product or geometric mean function

$$\varphi_i(R) = (R_1 \cdots R_{n_i})^{1/n_i},$$

and a generalization, the weighted geometric mean function

$$\varphi_i(R) = R_1^{w_1} \cdots R_{n_i}^{w_{n_i}},$$

with weights $w > 0$, $\mathbf{1}^T w = 1$, where $\mathbf{1}$ is the vector with all components one.

Network trade vector. The CFMM tendered and received baskets Δ_i and Λ_i can be mapped to the global indices as the n-vectors $A_i\Delta_i$ and $A_i\Lambda_i$, which give the numbers of tokens tendered to and received from CFMM i, using the global indices for the tokens. Summing the difference of received and tendered tokens over all CFMMs we obtain the *network trade vector*

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

This is an n-vector, which gives the total net number of tokens tendered to the CFMMs in the network. We interpret Ψ as the net network trade vector. If $\Psi_k \geq 0$, it means that we do not need to tender any of token k to the network. This does not mean we that do not trade token k; it only means that in our trading with the CFMMs, we receive at least as much as token k as we tender.

Network trade utility. We introduce a utility function $U : \mathbf{R}^n \to \mathbf{R} \cup \{-\infty\}$ that gives the utility of a trade Ψ to a trader as $U(\Psi)$. We will assume that U is concave and increasing. Infinite values of U are used to impose constraints; we consider a proposed trade with $U(\Psi) = -\infty$ as unacceptable. As an important example, consider the constraint $\Psi + h \geq 0$, where $h \in \mathbf{R}^n_+$ is the vector of a user's current holdings of tokens. This constraint specifies that the post-trade holdings $\Psi + h$ should be nonnegative, *i.e.*, we cannot enter into a trade that requires more of any token than we currently have on hand. To express this in the utility function, we define $U(z)$ to be $-\infty$ when $z + g \not\geq 0$. (This modified utility is also concave and increasing.)

There are many possible choices for U. Perhaps the simplest is the linear utility function $U(z) = \pi^T z$, with $\pi \in \mathbf{R}_{++}$, where we interpret π_i as the trader's internal or private value or price of token i. Several other choices are discussed in [4, §5.2].

Optimal routing problem. We wish to find a set of valid trades that maximizes the trader's utility. This optimal routing problem can be expressed as

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

The variables here are $\Psi \in \mathbf{R}^n$, $\Lambda_i \in \mathbf{R}^{n_i}_+$, $\Delta_i \in \mathbf{R}^{n_i}_+$, $i = 1, \dots, m$. The data are the utility function U, the global-local matrices A_i, and those associated with the CFMMs, the trading functions φ_i, the trading fees γ_i, and the reserve amounts $R_i \in \mathbf{R}^{n_i}_+$.

Convex optimal routing problem. Unless the trading functions are affine, the optimal routing problem (1) is not a convex optimization problem. We can, however, form an equivalent problem that is convex. To do this we replace the equality constraints with inequality constraints:

$$\begin{array}{ll} \text{maximize} & U(\Psi) \\ \text{subject to} & \Psi = \sum_{i=1}^m A_i(\Lambda_i - \Delta_i) \\ & \varphi_i(R_i + \gamma_i\Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1, \dots, m \\ & \Delta_i \geq 0, \quad \Lambda_i \geq 0, \quad i = 1, \dots, m. \end{array} \tag{2}$$

This problem is evidently convex [7, §5.2.1] and can be readily solved.

We will show that any solution of (2) is also a solution of (1). This the same as showing that for any solution of (2), the inequality constraints hold with equality. Suppose that $\Delta_i^\star$ and $\Lambda_i^\star$ are feasible for (2), but $\varphi_k(R_k + \gamma_k\Delta_k^\star - \Lambda_k^\star) > \varphi_k(R_k)$ for some k. This means we can find $\tilde{\Lambda} > \Lambda_k^\star$ which satisfies $\varphi_k(R_k + \gamma_k\Delta_k^\star - \tilde{\Lambda}) \geq \varphi_k(R_k)$. The associated trade vector $\tilde{\Psi} = \Psi^\star + A_k\Lambda_k$ satisfies

$\tilde{\Psi} \geq \Psi^\star$, $\tilde{\Psi} \neq \Psi^\star$, *i.e.*, at least one component of $\tilde{\Psi}$ is larger that the corresponding component of $\Psi^\star$. It follows that $U(\tilde{\Psi}) > U(\Psi^\star)$, so $\Psi^\star$ is not optimal and therefore cannot be a solution. We conclude that any solution of (2) satisfies the inequality constraints as equality, and so is optimal for (1).

A similar statement holds in the case when U is only nondecreasing, and not (strictly) increasing. In this case we can say that there is a solution of (2) that is optimal for (1). One simple method to find such a solution is to solve (2) with objective $U(\Psi) + \epsilon \mathbf{1}^T\Psi$, where ϵ is small and positive. The objective for this problem is increasing. In principle we can let ϵ go to zero to recover a solution of the original problem; in practice choosing a single small value of ϵ works.

Consequences of convexity. Since the problem (2) is convex, it can be reliably and quickly solved, even for large problem instances [7, §1]. Domain specific languages for convex optimiztion such as CVXPY [3, 9] or JuMP [11] can be used to specify the optimal routing problem in just a few lines of code; solvers such as ECOS [10], SCS [15], or Mosek [6] can be used to solve the problem.

Implementation. The solution to (2) provides the optimal values (Δ_i, Λ_i) that one must trade with each CFMM i. Note that we do not explicitly consider the ideal execution of these trades, as these will depend on the semantics of the underlying blockchain that the user is interacting with. For example, users may wish to execute trades in a particular sequence, starting with an initial portfolio of assets and updating their asset composition after each transaction until the final basket is obtained. Users may alternatively use flashloans [2] to atomically perform all trades in a single transaction, netting out all trades before repaying the loan at the end of the transaction.

3 OPTIMALITY CONDITIONS

Assuming that U is differentiable (which it need not be), the optimality conditions of problem (2) are feasibility, and the dual conditions

$$\nabla U(\Psi) = \nu, \tag{3}$$

and

$$\gamma_i \lambda_i \nabla \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i) \leq A_i^T \nu \leq \lambda_i \nabla \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i), \quad i = 1, \dots, m, \tag{4}$$

where $\nu \in \mathbf{R}^n$, $\lambda_i \in \mathbf{R}_+$, $i = 1, \dots, m$ are the Lagrange multipliers. (We derive these conditions in appendix A.)

The first condition (3) has a very simple interpretation: ν is the vector of marginal utilities of the tokens. The second set of conditions (4) has a simple interpretation that is very similar to the one given in [4, §5.1] for a single CFMM. The term $\nabla \varphi_i(R_i + \gamma_i \Delta_i - \Lambda_i)$, which we will write as $P_i \in \mathbf{R}_+^{n_i}$, can be interpreted as the unscaled prices of the tokens that CFMM i trades [4, §2.5], in the local indexing. Plugging (3) into (4),we get

$$\gamma_i \lambda_i P_i \leq A_i^T \nabla U(\Psi) \leq \lambda_i P_i, \quad i = 1, \dots, m. \tag{5}$$

The middle term is the vector of marginal utilities of the tokens that CFMM i trades, in the local indexing. These marginal utilities must lie between a multiple of the discounted prices, scaled, and the same prices, adjusted by γ_i.

We can also recognize the conditions (4) as those for the problem of finding an optimal trade for CFMM i alone, with the linear utility $U_i(z) = \pi_i^T z$, where $\pi_i = A_i^T \nabla U(\Psi)$. (See [4, §5.2].) This is very appealing: it states that the optimal trades for the network of CFMMs are also optimal for each CFMM separately, when they use a linear utility, with prices equal to the marginal utility of the overall trade.

Non-differentiable utility. When U is not differentiable, the optimality conditions are the same, but in (3) we substitute a supergradient $g \in -\partial(-U)(\Psi)$ for the gradient $\nabla U(\Psi)$, where ∂ denotes the subdifferential. When U is not differentiable at Ψ, there are multiple such g's, which we can consider to be multiple marginal utilities. The optimality condition is that (5) hold, with $\nabla U(\Psi)$ replaced with g, for any $g \in -\partial(-U)(\Psi)$.

No-trade condition. From the optimality conditions we can derive conditions under which the optimal trades are zero, *i.e.*, we should not trade. We will assume that $U(0) > -\infty$, *i.e.*, $\Psi = 0$ is feasible; if this is not the case, then evidently $\Psi = 0$ is not optimal. The zero trade

$$\Psi = 0, \qquad \Delta_i = \Lambda_i = 0, \quad i = 1, \dots, m,$$

is feasible for (5), so the optimality condition is

$$\gamma_i \lambda_i P_i \le A_i^T \nabla U(0) \le \lambda_i P_i, \quad i = 1, \dots, m, \tag{6}$$

where $P_i = \nabla \varphi_i(R_i)$ is the unscaled price of CFMM i at the current reserves R_i. This condition is a generalization of the no-trade condition given in [4, §5.1] for one CFMM, to the case where there are multiple CFMMs. When U is not differentiable, we replace $\nabla U(0)$ with a supergradient $g \in -\partial(-U)(0)$.

4 EXAMPLES

Linear utility. Consider the linear utility $U(z) = \pi^T z$, with $\pi > 0$. In this case the optimal routing problem is separable across the CFMMs, since the objective is a sum of functions of $\Lambda_i - \Delta_i$, and constraints are also only on (Λ_i, Δ_i). It follows that we can solve the optimal routing problem (2) by solving m single CFMM problems independently, using linear utilities with prices given by π.

Liquidating a basket of tokens. Suppose we start with an initial holding (basket) of tokens $h^{\text{init}} \in \mathbf{R}_+^n$ and wish to convert them all to token k. We use the utility function

$$U(z) = \begin{cases} z_k & h^{\text{init}} + z \ge 0 \\ -\infty & \text{otherwise.} \end{cases}$$

(This utility is nondecreasing but not increasing.)

A special case of this problem is converting one token into another, *i.e.*, when

$$h^{\text{init}} = t e_j,$$

where $t \ge 0$ and $e_j \in \mathbf{R}^n$ is the jth unit vector. In this special case, we can write the optimal value of the optimization problem, which we will call $u(t)$, as a function of t. It is not hard to show that u is nonnegative and increasing. This function is also concave as it is the partial maximization of a concave function (over all variables except t). We show an example instance of this problem, along with an associated function u, in §4.

Purchasing a basket of tokens. This is the opposite of liquidating a basket of tokens. Here too we start with initial token holdings $h^{\text{init}} \in \mathbf{R}_+^n$, and end up with the holdings $h^{\text{init}} + \Psi$. Let $h^{\text{des}} \in \mathbf{R}_+^n$ be our target basket; we wish to end up with the largest possible multiple of this basket. Let $\mathcal{K} \subseteq \{1, \dots, n\}$ denote the set of indices for which $h_i^{\text{des}} > 0$, *i.e.*, the indexes associated with tokens in the desired basket. We seek to maximize the value of α for which $h^{\text{init}} + \psi \ge \alpha h^{\text{des}}$. To do this we use the utility function

$$U(z) = \begin{cases} \min_{i \in \mathcal{K}} (h_i^{\text{init}} + \Psi_i)/h_i^{\text{des}} & h^{\text{init}} + z \ge 0 \\ -\infty & \text{otherwise.} \end{cases}$$

CFMM	**Trading function** φ_i	**Fee parameter** γ_i	**Reserves** R_i
1	Geometric mean, $w = (3, 2, 1)$	0.98	(3, .2, 1)
2	Product	0.99	(10, 1)
3	Product	0.96	(1, 10)
4	Product	0.97	(20, 50)
5	Sum	0.99	(10, 10)

Table 1. CFMM attributes.

Arbitrage detection. An arbitrage is a collection of valid CFMM trades with $\Psi \geq 0$ and $\Psi \neq 0$, *i.e.*, a set of trades for the CFMMs in which we tender no tokens, but receive a positive amount of at least one token. The optimal routing problem can be used to find an arbitrage, or certify that no arbitrage exists.

Consider any U that is increasing, with domain $\{\Psi \mid U(\Psi) > -\infty\} = \mathbf{R}^n_+$. Evidently there is an arbitrage if and only if there is a nonzero solution of the routing problem, which is the same as $U(\Psi^\star) > U(0)$, where $\Psi^\star$ is optimal. So by solving this optimal routing problem, we can find an arbitrage, if one exists.

No-arbitrage condition. Using the version of (6) for nondifferentiable U we can derive conditions under which there is no arbitrage. We consider the specific utility function

$$U(\Psi) = \begin{cases} \mathbf{1}^T\Psi & \Psi \geq 0 \\ -\infty & \text{otherwise,} \end{cases}$$

where $\mathbf{1}$ denotes the vector with all entries one. This utility is the total number of tokens received, when they are nonnegative. Its supergradient at 0 is

$$-\partial(-U)(0) = [1, \infty)^n = \{g \mid g \geq \mathbf{1}\}.$$

The condition (6) becomes: There exists $g \geq \mathbf{1}$, and $\lambda_i \geq 0$, for which

$$\gamma_i \lambda_i P_i \leq A_i^T g \leq \lambda_i P_i, \quad i = 1, \ldots, m.$$

By absorbing a scaling of g into the λ_i, we can say that $g > 0$ is enough.

This makes sense: it states that there is no arbitrage if we can assign a set of positive prices (given by g) to the tokens, for which no CFMM would trade. In the (unrealistic) case when $\gamma_i = 1$, *i.e.*, there is no trading cost, the no arbitrage condition is that there exists a global set of prices for the tokens, g, consistent with the local prices of tokens given by $\lambda_i P_i$.

5 NUMERICAL EXAMPLE

The Python code for the numerical example we present here is available at

`https://github.com/angeris/cfmm-routing-code.`

The optimization problems are formulated and solved using CVXPY [3]. A listing of the core of the code is given in appendix B.

Network. We consider the network of 5 CFMMs and 3 tokens shown in figure 1. The trading functions, Fee parameters, and reserves are listed in table 1.

Problem and utility. We wish to trade an amount $t \geq 0$ of token 1 for the maximum possible amount of token 3. This is a special case of the problem of liquidating a basket of tokens, as described in §4, with initial holdings $h^{\text{init}} = te_1$. The utility function is $U(z) = z_3$, provided $z + h^{\text{init}} \geq 0$, and

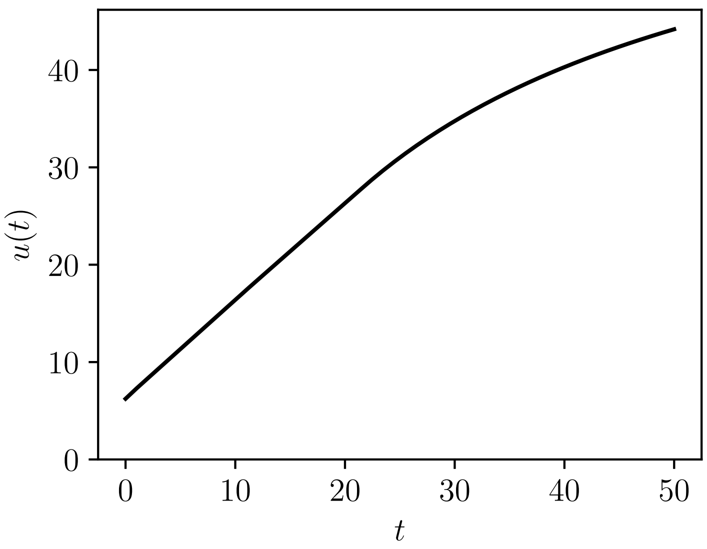


Fig. 2. Plot of $u(t)$, the maximum amount of token 3 we obtain when we tender the amount t of token 1.

$U(z) = -\infty$ otherwise. We let $u(t)$ denote the maximum amount of token 3 we can obtain from the network when we tender token 1 in the amount t.

Results. We solve the optimal routing problem for many values of t, from $t = 0$ to $t = 50$. The amount of token 3 we obtain in shown in figure 2. We see that $u(0) > 0$, which means there is an arbitrage in this network; indeed, there is a set of trades that requires giving zero net tokens to the network, but we receive an amount around 7 of token 3.

The associated optimal trades are shown in figure 3. We can see many interesting phenomena here. At $t = 0$ we see the arbitrage trades, indeed, the arbitrage trades that yield the largest amount of token 3. Several asset flows reverse sign as t varies. For example, for $t < 11$, we receive token 1 from CFMM 1, whereas for $t > 11$, we tender token 1 to CFMM 1. We can also see that the sparsity pattern of the optimal trades changes with t.

We illustrate the changing signs and sparsity of the optimal trades in figure 4, which shows the optimal trades for $t = 0$, $t = 20$, and $t = 50$. We plot whether each token is tendered to or received from each CFMM using color coded edges. A red edge connecting a token to a CFMM means that the CFMM is receiving this token, while a blue edge denotes that the CFMM is tendering this token. A dashed edge denotes that the CFMM neither tenders nor receives this token.

Even in this very simple example, the optimal trades are not obvious and involve trading with and among multiple CFMMs. For larger networks, the optimal trades are even less obvious.

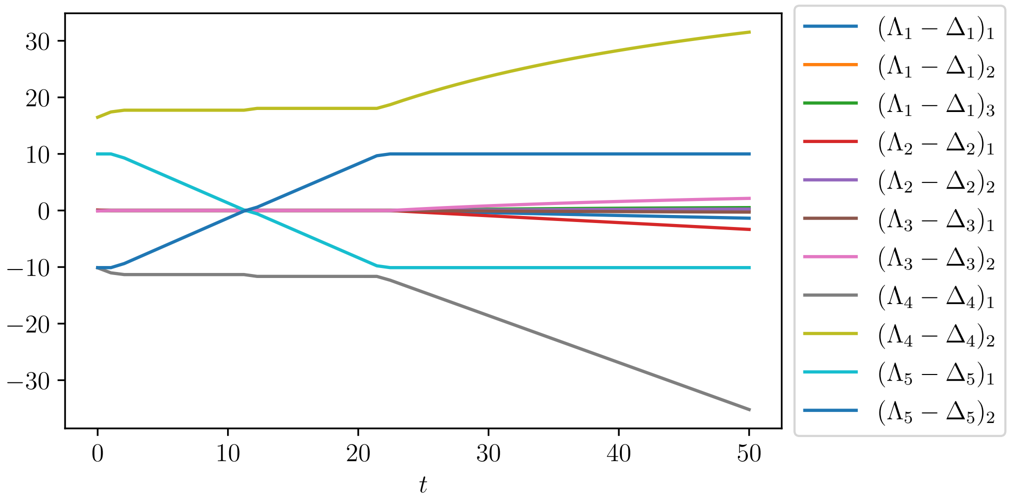
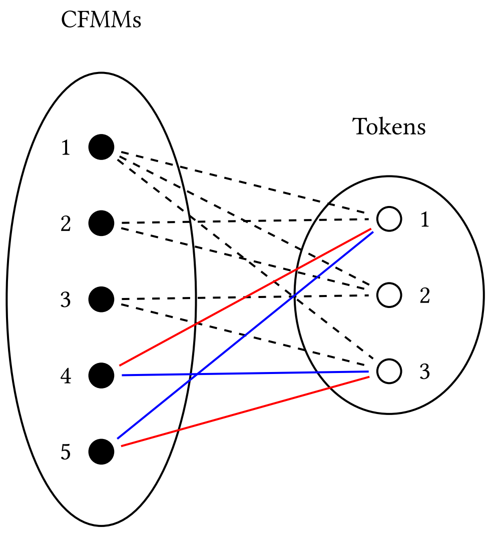
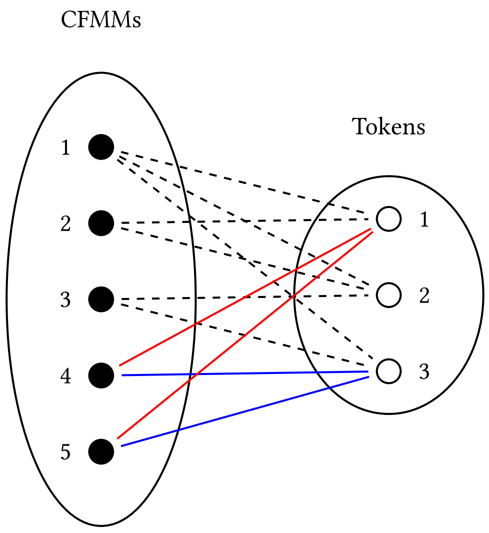


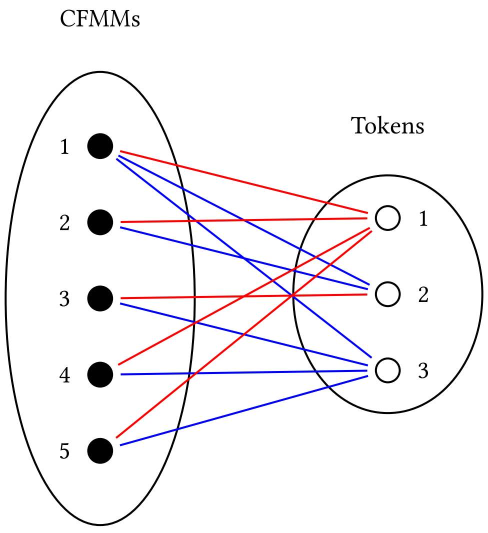
Fig. 3. Optimal trades versus t, the amount of token 1 tendered.



(a) $t = 0$



(b) $t = 20$



(c) $t = 50$

Fig. 4. Optimal trades as t varies. A red edge means the CFMM is receiving tokens; a blue edge means the CFMM is tendering tokens.

6 FIXED TRANSACTION COSTS

Our optimal routing problem (2) includes the trading costs built into CFMMs, via the parameters γ_i. But it does not include the small fixed cost associated with any trade. In this section we explore how these fixed transaction costs can be incorporated into the optimal routing problem.

We let $q_i \in \mathbf{R}_+$ denote the fixed cost of executing a trade with CFMM i, denominated in some numeraire. We pay this whenever we trade, *i.e.*, $\Lambda_i - \Delta_i \neq 0$. We introduce a new set of Boolean variables into the problem, $\eta \in \{0,1\}^m$, with $\eta_i = 1$ if a nonzero trade is made with CFMM i, and $\eta_i = 0$ otherwise, so the total fixed transaction cost is $q^T\eta$. We assume that there is a known maximum size of a tendered basket with CFMM i, which we will denote $\Delta_i^{\max} \in \mathbf{R}_+^{n_i}$. We can then express the problem of maximizing the utility minus the fixed transaction cost as

$$\begin{array}{ll}
\text{maximize} & U(\Psi) - q^T\eta \\
\text{subject to} & \Psi = \sum_{i=1}^m A_i(\Lambda_i - \Delta_i) \\
& \varphi_i(R_i + \gamma_i\Delta_i - \Lambda_i) \geq \varphi_i(R_i), \quad i = 1,\dots,m \\
& 0 \leq \Delta_i \leq \eta_i\Delta_i^{\max}, \quad \Lambda_i \geq 0, \quad i = 1,\dots,m, \\
& \eta \in \{0,1\}^m,
\end{array} \tag{7}$$

where the variables are Ψ, Λ_i, Δ_i, $i = 1,\dots,m$, and $\eta \in \{0,1\}^m$.

The optimal routing problem with fixed costs (7) is a mixed-integer convex program (MICP). It can be solved exactly, possibly with great computational effort, using global optimization methods, with MICP solvers such as Mosek [6] or Gurobi [13]. When m is small (say, under ten or so), it can be practical to solve it by brute force, by solving the convex problem we obtain for each of the 2^m feasible values of η.

Approximate solution methods. Many approximate methods have the speed of convex optimization, and (often) produce good approximate solutions. For example we can solve the relaxation of (7) obtained by replacing the constraints on η to $\eta \in [0,1]^m$ (which gives a convex optimization problem). After that we set a threshold $t \in (0,1)$ for the relaxed optimal values η^{rel}, and take $\eta_i = 1$ when $\eta^{\text{rel}} \geq t$ and and $\eta_i = 0$ when $\eta^{\text{rel}} < t$. We fix these values of η and then solve the resulting convex problem. This could be done for a modest number of values of t; we take the solution found with the largest objective (including the fixed costs $q^T\eta$).

An alternative is a simple randomized method. We interpret η_i^{rel} as probabilities, and generate $\eta \in \{0,1\}$ randomly using these probabilities. We then solve the convex problem associated with this choice of η. We can repeat this procedure a modest number of times, and pick the feasible point with the highest payoff.

7 CONCLUSION AND FUTURE WORK

In this paper, we have shown a simple, but very general, formulation of the routing problem, which includes the arbitrage problem as a special case. The fact that the routing problem, without fixed transaction costs, is convex leads to a number of important consequences. For example, it is easy to certify that there is no arbitrage in a given network using the optimality conditions of the original problem. It is additionally computationally easy to route orders across a number of exchanges, or to find an arbitrage that maximizes some utility, that simple heuristics are likely to miss.

Future work could involve finding better heuristics in the case where fixed transaction costs are included in the objective. Additional important work would be to find a reasonable framework for order execution that incorporates the mechanics of different chains.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their comments, many of which we have incorporated in this paper.

REFERENCES

[1] 2021. DEX Aggregator Volume. https://www.theblockcrypto.com/data/decentralized-finance/dex-non-custodial/dex-aggregator-trade-volume.

[2] Aave. 2020. Flash Loans: Pushing the Limits of DeFi. https://aave.com/flash-loans/.

[3] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. 2018. A Rewriting System for Convex Optimization Problems. *Journal of Control and Decision* 5, 1 (2018), 42–60.

[4] Guillermo Angeris, Akshay Agrawal, Alex Evans, Tarun Chitra, and Stephen Boyd. 2021. Constant Function Market Makers: Multi-Asset Trades via Convex Optimization. *arXiv:2107.12484 [math, q-fin]* (July 2021). arXiv:math, q-fin/2107.12484

[5] Guillermo Angeris and Tarun Chitra. 2020. Improved Price Oracles: Constant Function Market Makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. ACM, New York NY USA, 80–91. https://doi.org/10.1145/3419614.3423251

[6] MOSEK ApS. 2019. MOSEK Optimizer API for Python 9.1.5. https://docs.mosek.com/9.1/pythonapi/index.html.

[7] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization.* Cambridge University Press, Cambridge, UK ; New York.

[8] Vincent Danos, Hamza El Khalloufi, and Julien Prat. 2021. Global order routing on exchange networks. In *International Conference on Financial Cryptography and Data Security*. Springer, 207–226.

[9] Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research* 17, 1 (2016), 2909–2913.

[10] Alexander Domahidi, Eric Chu, and Stephen Boyd. 2013. ECOS: An SOCP Solver for Embedded Systems. In *2013 European Control Conference (ECC)*. IEEE, Zurich, 3071–3076. https://doi.org/10.23919/ECC.2013.6669541

[11] Iain Dunning, Joey Huchette, and Miles Lubin. 2017. JuMP: A Modeling Language for Mathematical Optimization. *SIAM Rev.* 59, 2 (Jan. 2017), 295–320. https://doi.org/10.1137/15M1020575

[12] Daniel Engel and Maurice Herlihy. 2021. Composing Networks of Automated Market Makers. *CoRR* abs/2106.00083 (2021). arXiv:2106.00083 https://arxiv.org/abs/2106.00083

[13] Gurobi Optimization, LLC. 2021. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[14] Uniswap Labs. 2021. Uniswap Router02. https://docs.uniswap.org/protocol/V2/reference/smart-contracts/router-02. (2021).

[15] Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. 2016. Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding. *Journal of Optimization Theory and Applications* 169, 3 (June 2016), 1042–1068. http://stanford.edu/~boyd/papers/scs.html

[16] Yi Zhang, Xiaohong Chen, and Daejun Park. 2018. Formal Specification of Constant Product ($xy = k$) Market Maker Model and Implementation. (2018).

A DERIVATION OF OPTIMALITY CONDITIONS

To derive condition (4), we introduce the Lagrangian of the convex optimal routing problem (2),

$$\mathcal{L}(\Psi, \Delta, \Lambda, \nu, \lambda, \eta, \xi) = -U(\Psi) + \nu^T \left(\Psi - \sum_{i=1}^m A_i(\Lambda_i - \Delta_i)\right)$$
$$+ \sum_{i=1}^m \lambda_i(\varphi_i(R_i) - \varphi_i(R_i + \gamma_i\Delta_i - \Lambda_i)) - \sum_{i=1}^m (\eta_i^T \Delta_i + \xi_i^T \Lambda_i).$$

Here, $\nu \in \mathbf{R}^n$, $\lambda \in \mathbf{R}_+^m$, and $\eta_i, \xi_i \in \mathbf{R}_+^{n_i}$, $i = 1, \dots, m$. These are the dual variables or Lagrange multipliers for the equality constraint, the relaxed inequality constraints, and nonnegativity constraints for the tendered and received baskets of assets for each CFMM, respectively.

The optimality conditions are (primal) feasibility and the dual conditions:

$$\nabla_\Psi \mathcal{L} = 0, \quad \nabla_\Delta \mathcal{L} = 0, \quad \nabla_\Lambda \mathcal{L} = 0.$$

The first of these is

$$-\nabla U(\Psi) + \nu = 0,$$

while the latter two are

$$-A_i^T \nu + \lambda_i \nabla\varphi_i(R_i + \gamma_i\Delta_i - \Lambda_i) - \xi_i = 0, \qquad A_i^T \nu - \gamma_i\lambda_i \nabla\varphi_i(R_i + \gamma_i\Delta_i - \Lambda_i) - \eta_i = 0.$$

These can be written as

$$-A_i^T \nu + \lambda_i \nabla\varphi_i(R_i + \gamma_i\Delta_i - \Lambda_i) \geq 0, \qquad A_i^T \nu - \gamma_i\lambda_i \nabla\varphi_i(R_i + \gamma_i\Delta_i - \Lambda_i) \geq 0,$$

which simplifies to (4).

B EXAMPLE CODE

```
import numpy as np
import cxvpy as cp

# We assume the following is already defined:
# - `A`, list of matrices A_i
# - `reserves`, list of vectors R_i
# - `fees`, list of fees for each CFMM i
# - `num_tokens`, list of n_i
# - `t`, amount of asset 1 provided to the network

current_assets = np.array([t, 0, 0])

# Build variables
deltas = [cp.Variable(n_i, nonneg=True) for n_i in num_tokens]
lambdas = [cp.Variable(n_i, nonneg=True) for n_i in num_tokens]
psi = cp.sum(A_i @ (L - D) for A_i, D, L in zip(A, deltas, lambdas))

# Objective is to trade t of asset 1 for a maximum amount of asset 3
obj = cp.Maximize(psi[2])

# Reserves after trade
new_reserves = [R + gamma_i*D - L \
    for R, gamma_i, D, L in zip(reserves, fees, deltas, lambdas)]

# Trading function constraints
cons = [
    # Balancer pool with weights 3, 2, 1
    cp.geo_mean(new_reserves[0], p=np.array([3, 2, 1])) \
        >= cp.geo_mean(reserves[0], p=np.array([3, 2, 1])),

    # Uniswap v2 pools
    cp.geo_mean(new_reserves[1]) >= cp.geo_mean(reserves[1]),
    cp.geo_mean(new_reserves[2]) >= cp.geo_mean(reserves[2]),
    cp.geo_mean(new_reserves[3]) >= cp.geo_mean(reserves[3]),

    # Constant sum pool
    cp.sum(new_reserves[4]) >= cp.sum(reserves[4]),
    new_reserves[4] >= 0,

    # Allow all assets at hand to be traded
    psi + current_assets >= 0
]

# Set up and solve problem
prob = cp.Problem(obj, cons)
prob.solve()

print(f"amount of asset 3 received: {psi[2].value}")
```