Cross-Domain Coincidence of Wants and Constant Function Market Maker Optimal Routing

Bruno Mazorra Dzmitry Lahoda Sydney Sweck Omar Zaki

January 30, 2024

1 Abstract

This paper introduces a method for solving cross-domain intents. We present a novel algorithm for determining the optimal pathway across multiple blockchains. Our approach introduces intents over asset spaces through utility functions and frames the multi-domain optimal routing as a mixed integer convex optimization problem. The methodology is underscored by modeling relay transaction fees and incorporating the Arrow-Debreu model for frequent batch auctions in multiple domains.

2 Introduction

In this paper, we introduce an approach for optimally solving cross-blockchain intents. In other words, we provide an algorithm for determining the optimal pathway for executing partially-defined cryptocurrency transactions across multiple blockchains.
This paper makes the following contributions:

- Introducing intents over the space of assets by using utility functions.
- Reducing the multi-domain optimal routing to a mixed integer convex optimization problem. To do so, we modelled relay transaction fees as constant sum market makers (also known as stable pools) using the framework of [Ang+22].
- Introducing the Arrow-Debreu model to define frequent batch auctions for multiple domains.

3 Background

3.1 Cross-Blockchain Interoperability

A *blockchain* (or simply a *chain*) is an immutable *ledger* maintained by a peer-to-peer network of nodes. Each node stores the state of the chain and participates in a consensus process which generates new *blocks* holding the new state of the chain. Each block is bound through a cryptographic hash to the preceding one, forming a chain. While Bitcoin (the first ever blockchain) supports transferring tokens between accounts only [Nak08], later innovations in the Ethereum blockchain offered a method for storing arbitrary data and executing arbitrary code, turning the blockchain into its own execution environment [But14]. With time, more blockchains with similar features were introduced such as Solana, NEAR, and TRON [NEA20; Yak18; TRO18].
Thus, cross-blockchain interoperability (also referred to by cross-domain or simply cross-chain interoperability) has became a popular area of research [Ou+22; MK+23] aiming to address the

lack of coordination between the myriad blockchains in existence. Cross-chain interoperability is important, as it enables new and potentially valuable use cases to users. These opportunities encompass a range of areas including cross-chain governance as well as cross-domain decentralized finance (DeFi) processes like yield aggregation, lending, borrowing, and other cryptocurrency transactions [ZLQ23].

3.2 Intents and Solvers

The present paper explores a means of facilitating cross-chain intent settlement by solvers. With cross-chain interoperability and its importance detailed, we now define intents, solvers, and their utility:

Definition 1. An *intent* [GYB22] is an expression of what a user wants to achieve whenever they interact with a blockchain protocol, for instance "transfer X asset from blockchain A to blockchain B" or "trade X asset for Y asset". Practically, an intent is an off-chain signed message that encodes which state transitions a user wants to achieve. Unlike transactions, intents are partial. Thus, one can think of intents as parts of transactions that require other direct or indirect parts as complements in order to form a final balanced transaction that satisfies all of a user's constraints.

Let $\mathcal{S}$ be all the set of valid states of a public ledger, and let $\mathcal{O}$ be the set of all valid state transitions. Then, an intent can be defined through a utility function $U : \mathcal{O} \to \mathbb{R} \cup \{-\infty\}$ (a mathematical function that describes an individual's preference for goods or services by assigning a level of utility or satisfaction to each bundle of goods or services.).

Definition 2. A *solver* [Cowb] can be defined as an entity that competes to determine an optimal solution (in the form of a transaction execution pathway) for a user's intent.

In this context, an *altruistic solver* is an agent that "tries" to find a path $\gamma^\star$ such that $\gamma^\star \in \text{argmax}_\gamma U(\gamma)$. However, in general, solvers are non-altruistic and have their own utility function over the states of the ledger.
A *rational solver* is an agent that has preferences represented by a utility function u over the space of states. A rational solver finds a feasible transition state $\gamma^\star \in \text{argmax}_\gamma \ \ u(\gamma)$. The fundamental difference between a rational and an altruistic solver is which utility function it is trying to maximize.
However, in general, state transitions are not deterministic and depend on the simultaneous actions of other agents. For example, if two incompatible transactions are sent to a private relayer (working with a first price auction type of mechanism, meaning the bidder with the highest bid pays its bid and receives the asset) [Rei+23], then just the transaction with the highest bid is included in the chain. In this scenario, agents have priority over the number of competitors, their preferences, and their actions.

4 Methods

4.1 Cross-Chain Intent Settlement Framework

In the present paper, we assume the existence of a cross-chain intent settlement framework with the following players/components:

1. Cross-Chain Bridge: facilitates execution of cross-chain intents/transactions
2. Users: end-users who submit intents to the framework
3. Solvers: defined in 3.2; compete to determine optimal execution pathways for user intents and subsequently execute upon these pathways

4. Intents/Problems: defined in 3.2; inputted by users to describe their desired outcome from the framework
5. Solutions: an execution pathway for users' intents; these can comprise any combination of a solver's own liquidity, Constant Function Market Makers (CFMMs), and order matching along the Coincidence of Wants (CoWs) principle [Cowa].

Along this cross-chain intent settlement framework, the algorithms described in this paper offer a solution for providing solvers with a means of optimizing their proposed solutions for users.

4.2 Optimal Routing

We will use the notation used in [Ang+22]. In the paper, the authors write the following:

Optimal routing

Our optimal routing problem (2) includes the trading costs built into CFMMs, via the parameters ψ. 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 $q_i \in \mathbb{R}^+$ denote the fixed cost of executing a trade with CFMM i, denominated in some numeraire. We pay this whenever we trade, *i.e.*, $\Delta_i \neq 0$. 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 $\Delta_{i,\max} \in \mathbb{R}^+$. We can then express the problem of maximizing the utility minus the fixed transaction cost as:

$$\begin{aligned}
\text{maximize} \quad & U(\Psi) - q^T\eta && (1)\\
\text{subject to} \quad & \Psi = \sum_{i=1}^{m} A_i(\Lambda_i - \Delta) && (2)\\
& \psi_i(R_i + \gamma_i\Delta_i - \Lambda_i) \geq \psi_i(R_i), \quad i = 1,\ldots,m && (3)\\
& 0 \leq \Delta_i \leq \eta_i\Delta_{i,\max}, \quad \Delta_i \geq 0, \quad i = 1,\ldots,m && (4)\\
& \eta \in \{0,1\}^m && (5)
\end{aligned}$$

where the variables are $\Psi, \Delta_i, \Delta, i = 1,\ldots,m$, and $\eta \in \{0,1\}^m$.
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 and take $\eta = 1$ when $\eta \geq t$ and $\eta = 0$ otherwise. 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 $q\eta$).
An alternative is a simple randomized method. We interpret η_i as probabilities and generate a η_i 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.

We can extend the idea to take into account cross-domain Inter-Blockchain Communication (IBC) Protocol transfers. The IBC Protocol is a trustless cross-blockchain interoperability protocol that has been previously described by [Goe20].

To our knowledge, relays charge fees in two different ways: with a fixed cost or with a percentage. Also, relays have token rate limits, i.e. during a time frame, there is a limit R_i of tokens that can be transferred from one chain to another one.
In this context, we can think of IBC transfers from chain i to chain j as a constant sum market maker (with invariant function $\varphi(x, y) = x+(1-\gamma_i)y$ with reserves (R_i, R_j), where R_1 is the rate limiter of tokens that can be transferred from the second chain to the first one, and R_2 being its analog in the other direction). If the IBC transfer has a constant cost, then we associate a non-zero fee η_{ij} and CFMM fee $\gamma_i = 0$. If the IBC transfer has a percentage cost, then we associate a non-zero fee $\eta_{ij} = 0$ and CFMM fee $\gamma_i > 0$. If the IBC transfer is non-symmetric, then one can model it by considering two different constant sum market makers with one of the reserves being zero.
Therefore, when we have an intent of an utility function U, the cross-domain optimal routing problem can be expressed as the following optimization problem:

$$\begin{aligned}
\text{maximize} \quad & U(\Psi) - q^T\eta && (6)\\
\text{subject to} \quad & \Psi = \sum_{i=1}^{m} A_i(\Lambda_i - \Delta_i) && (7)\\
& \psi_i(R_i + \gamma_i\Delta_i - \Lambda_i) \geq \psi_i(R_i), \quad i = 1, \ldots, m && (8)\\
& 0 \leq \Delta_i \leq \eta_i\Delta_{i,\max}, \quad \Delta_i \geq 0, \quad i = 1, \ldots, m && (9)\\
& \eta \in \{0, 1\}^m && (10)\\
& && (11)
\end{aligned}$$

where ψ_i represents all the CFMM and IBC transfers. The problem is a mixed integer convex optimization problem. Now we can apply the approximate solution method proposed before by [Ang+22]. However, sometimes the η do not represent how much we are using the pool. For example if $\Delta_{i,\max}$ is a large number, then η_i can be very small, under-representing the importance of using that pool for optimal routing. In other words, all η should be more or less on the same scale in order to choose which pools to use afterwards. To do so, we will substitute the constraints $0 \leq \Delta_i \leq \eta_i\Delta_{i,\max}, \quad \Delta_i \geq 0, \quad i = 1, \ldots, m$ with $0 \leq p_i \cdot \Delta_i \leq \eta_i \cdot p \cdot \Delta_{in}$, where p_i is the vector of prices of the assets of the pool, p is the price vector of the assets that the users have initially, and Δ_{in} is the basket of assets that the users has and is willing to trade. One can compute p_i using an oracle or using $\nabla\varphi_i(R_i)$.
An example of function U can be $U(z_1, ..., z_n) = \begin{cases} z_k, \text{ if } z + t \geq 0, \\ -\infty, \text{ otherwise.} \end{cases}$ With this function, we are solving the optimal routing problem when the user wants to liquidate a basket of assets $t = (t_1, ..., t_n)$ for a maximum amount of token k. In particular, if $t = (0, .., t_i, ..., 0)$, this changes all tokens t_i for maximum amount of tokens k.

4.3 Cross-Domain Frequent Batch Auction

Frequent Batch Auctions (FBAs) can be defined as uniform-price sealed-bid double auctions conducted at frequent but discrete time intervals [BCS14]. In the present paper, we apply the FBA method to cross-blockchain intents and transactions.
This Cross-Domain FBA system aims to address the complexities arising from trading assets that reside on disparate blockchain networks. Built upon the theoretical framework of the Arrow-Debreu market model [AD83], this system seeks to find the market equilibrium point where supply and demand intersect across multiple domains. The computational challenge of solving for this equilibrium, an NP-hard problem, is mitigated through the introduction of competitive "solvers". These agents are designed to maximize social welfare while considering the costs associated with asset bridging and interactions with CFMMs, a class of automated market-makers deployed via smart contracts in decentralized finance [Ram+23].

The Arrow-Debreu Market Model: The Arrow-Debreu market model serves as the foundational theory for the Cross-Domain FBA system. In this model, agents with heterogeneous utility functions interact within a market to reach an equilibrium point. The model assumes a set of goods $\mathbf{T} = \{T_1, T_2, \ldots, T_n\}$, a set of agents $A = \{1, 2, \ldots, m\}$, and a utility function $U_i : \mathbb{R}^n \to \mathbb{R}$ for each agent a_i concerning the goods. The objective is to find a set of prices for the goods that equates supply and demand, thereby achieving market equilibrium.
Utility Function is $U_i(x_1, x_2, \ldots, x_n)$. This quantifies the satisfaction or utility that agent a_i derives from consuming a bundle of goods $(x_1, x_2, \ldots, x_n)$. The utility function is assumed to be continuous, strictly increasing, and concave.
Maximality Condition: Each agent aims to maximize their utility subject to their budget constraint. Mathematically, this can be expressed as $\max_{x_1,x_2,\ldots,x_n} U_i(x_1, x_2, \ldots, x_n)$. This is subject to $\sum_{j=1}^{n} p_j x_{ij} \leq B_i$ where p_j is the price of good g_j and B_i is the budget of agent i. The budget $B_i = \sum_{i=j}^{n} p_j y_j$, where y_{ij} is the number of T_j assets that the agent i has (or commits) in their intent expression.
Equilibrium Price: A price vector $(p_1, p_2, \ldots, p_n)$ is said to be an equilibrium price if, given these prices, every agent maximizes their utility (satisfying the maximality condition) and the market clears, i.e. supply equals demand for each good. Formally, market clearing is expressed as:

$$\sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m} y_{ij} \quad \forall j$$

where x_{ij} is the quantity of good g_j consumed by agent a_i, and y_{ij} is the initial endowment of good g_j for agent a_i.
Computational Complexity: Finding the market equilibrium point according to the Arrow-Debreu model is an NP-hard problem, meaning this computational complexity grows exponentially with the increase in the number of agents and assets involved. Therefore, traditional optimization techniques are often infeasible for real-time or near-real-time calculations required in a decentralized trading environment.
Role of Solvers: To avoid the computational bottleneck, the Cross-Domain FBA employs "solvers", which are specialized agents that attempt to find the market equilibrium price by maximizing the overall welfare function. These solvers compete against each other to find the most efficient set of prices at which the batch of orders can be settled.
Access to Private Liquidity and CFMMs: Solvers are granted access to private liquidity pools and can engage with Constant Function Market Makers to improve the welfare outcome. They are also responsible for calculating the costs associated with bridging assets between different blockchains or domains. This ensures that the final batch of transactions is not only balanced but also maximizes the utility for all participating agents, given the feasibility constraints.

4.3.1 Warm up: One domain, two assets, no constraints

A set of users that have $x_1^i, ..., x_n^i$ T_1 tokens want to buy T_2. Another set of users have $y_1^i, ..., y_m^i$ tokens and want to sell these tokens. First, let us assume that these are market orders (e.g. trade at all prices). There is a $\mathcal{C}$ CFMM with reserves $R = (R_1, R_2)$. Let x_i^o be the amount of tokens that the ith user obtains and let $x^o = \sum_{i=1}^{n} x_i^o$. Let this same statement hold true for y. If we want all orders to settle at the same market price, we must hold the following feasible condition:

$$\frac{y^i}{x^o} = \frac{y^o}{x^i}$$

Since the orders are executed in a CFMM, the feasibility conditions for the CFMM are

$$C(R + (x^i - x^o, y^i - y^o)) = C(R)$$

In the case that the CFMM is a constant product market maker (e.g. is based on the function where quantity of asset X times quantity of asset Y is always equal to a constant product), this is equivalent to:

$$R_1 R_2 = (R_1 + x^i - x^o)(R_2 + y^i - y^o)$$

Putting this all together, we want to find x^o, y^o such that

$$R_1 R_2 = (R_1 + x^i - x^o)(R_2 + y^i - y^o) \tag{12}$$
$$y^i x^i = y^o x^o \tag{13}$$

If we solve the system of equations, we obtain that

$$x^o = \frac{R_1 + x^i}{R_2 + y^i} y^i \tag{14}$$
$$y^o = y^i x^i / x^o. \tag{15}$$

Then, we are able to give buyer j with x_j^i the amount $\frac{x_j^i}{\sum_{j=1}^n x_j^i} x^o$. The analog of this is thus able to occur for the sellers in this scenario.

4.3.2 2 Assets with constraints

Now, let us assume that we have a set of buy orders $\{(x_i, p_i^b)\}$ and sell orders $\{(y_j, p_j^s)\}$. The price denotes the worst price the users are willing to accept for trading. First, let us assume that we do not have a CFMM.
Let $p_{min} = \min\{\min\{p_i^b, p_j^s\} \text{ for all } i, j\}$, $p_{max} = \max\{\max\{p_i^b, p_j^s\} \text{ for all } i, j\}$, and $\mathcal{P} = [p_{min}, p_{max}]$. Our objective is to find $p^\star \in \text{argmax}_{p \in \mathcal{P}} \min\{\sum_{i=1}^n x_i \mathbb{1}_{p_i^b \geq p}, \sum_{j=1}^m y_j \mathbb{1}_{p_i^s \leq p}\}$. That is, we want to find a price $p^\star$ that maximizes the volume of assets exchanged. The minimum in the equation means that we permit partial fills.

Note: $\mathbb{1}_A = \begin{cases} 1, \text{ if } x \in A, \\ 0, \text{ otherwise.} \end{cases}$

Matching algorithm

1. Take the buy orders $\{(x_i, p_i^b)\}$ and sell orders $\{(y_j, p_j^s)\}$.
2. Denote p_k the prices p_i^b, p_j^s for $k = 1, ..., n+m$ and order them in descending order, i.e. $p_1 \geq p_2 \geq ... \geq p_k$.
3. For all prices p_l, compute $V(p_l) := \min\{\sum_{i=1}^n x_i \mathbb{1}_{p_i^b \geq p_l}, \sum_{j=1}^m p_l y_j \mathbb{1}_{p_i^s \leq p_l}\}$.
4. Take $p^\star \in \text{argmax}_{p_l} V(p_l)$.

With the above-described Matching Algorithm, we compute the price where the orders are settled. However, some orders will be partially filled. The question is therefore: which orders do we fill first? In the following, we propose an algorithm that sorts orders from large to small and fills them in this sequence. In this new algorithm, we add the following to the previous algorithm:

Trade resolution

5. Suppose that the min in $V(p^\star)$ is realized by the sells. Then, we fill all the sell orders with the assets provided by the buy orders. The buy orders are filled by descending order, filling the last one partially. The analog for this is true if the min is realized by the buys.

Observe that if there are many prices that realize the maximum, we have the freedom to choose other prices. For example, we could choose the one that is closest to the prices of the orders executed. Now, let us assume that there is a CFMM with reserves $R = (R_1, R_2)$ and trading function φ. The current price quoted by the CFMM is given by $p_c = \frac{\frac{\partial \varphi(R)}{\partial x_1}}{\frac{\partial \varphi(R)}{\partial x_2}}$. We can then implement the following algorithm:

Matching algorithm with CFMM

1. Take the buy orders $\{(x_i, p_i^b)\}$ and sell orders $\{(y_j, p_j^s)\}$.
2. First, compute $p^\star$ and $V(p^\star)$.
3. If $p^\star > p_c$, then put an amount x_s obtaining y_s (and so with price $p_o = x_s/y_s$ in the CFMM) such that we run the matching algorithm including the order of selling y_s at price p_o. Find p_o such that $p_o = p^\star$. The analog is done for this step if $p^\star < p_c$.

Note: $y_s = R_2 - \frac{R_1 R_2}{x_s + R_1}$.
Observe that we can change the function to maximize V to obtain other types of "global welfare functions". For example, we could also add that, if two solutions have the same volume, then we choose the one with a price closer to an oracle price. However, is unlikely that two solutions have exactly the same price, therefore this would not reflect the preference of the designer to have a "fair" price. Therefore, we could encapsulate this by maximizing a mix of both functions $W(p^\star) = \alpha V(p^\star) - (1-\alpha)|p_{oracle} - p^\star|$. Changing to this, we would punish solutions that are too far from a referred price.
This algorithm can be extended when there is more than one CFMM in the same chain doing optimal routing. The code for 2-assets-matching is in (work in progress): `https://github.com/BrunoMazorra/2-assets-matching`.

4.3.3 General mechanism for 2 assets

Intents that solely express swaps can be written as utility function $U : \mathbb{R}_+^l \to \mathbb{R}$, where l is the number of tokens. Here, we will assume that $l = 2$. So, if the current vector price is $p \in \mathbb{R}_+^l$, then the maximization problem becomes:

$$\begin{aligned} \max_{\Delta'} \quad & U(\Delta') \\ \text{s.t.} \quad & p \cdot \Delta = p \cdot \Delta'. \end{aligned}$$

That is, if a user has a basket of tokens Δ and the current price is p, then the solution of the previous optimization problem is the best execution of user utility under the current price p. For a set of intents $\{(U_i, \Delta_i)\}_{i=1,\dots,n}$, we denote the aggregated demand by $\zeta(p) = \sum_{i=1}^n \Delta_i'$.
Recall Δ_i and Δ_i' are vectors of two assets where the first component denotes the number of assets of the first token and the second component denotes the number of assets of the second token.
When we have a set of intents, our objective is to (i) match supply and demand by finding a price p such that $\zeta(p) = 0$, and (ii) maximize the traded volume (that is, maximize $p \cdot [\sum_{i=1}^n \Delta_i']$).
In the following, we explain the general mechanism for a system where solvers implement the aforementioned algorithms to create solution pathways to user intentions. We do not describe the bridging process itself, as this is outside of the scope of the present document. In the following mechanism, the orders are expressed as intents in terms of the price p and are not necessarily limit orders.

Solver inputs to the mechanism

A solution consists of the following:

1. A subset of orders $\mathcal{O}' \subseteq \mathcal{O}$.
2. An execution price p (a total fee is optional, bounded by 0.5%, that is subtracted pro-rata to each transaction).
3. A (set) private order $\mathcal{P}_a$ and/or a path of a CFMM with an input amount and a minimum output amount $\mathcal{P}_p$. The path execution also commits the expected assets obtained.
4. Some collateral C (this collateral can be settled before the commitment to the solution).
5. A set of IBC token transfers $\mathcal{I} = \{\mathcal{I}_1, ..., \mathcal{I}_k\}$.

In summary, an order consists of tuple $\{(\mathcal{O}', p, fee, \mathcal{P}_a, \mathcal{P}_p, \mathcal{I})\}$. The fee serves to incentivize users.

Mechanism

1. For some time interval $[t_0, t_1]$, users submit their orders $\mathcal{O}$.
2. Then, for time interval $[t_1, t_2]$, solvers submit their solutions $\mathcal{S}_1, ..., \mathcal{S}_m$.
3. Solutions are ordered by the committed performance (and the feasibility conditions are checked).
4. The best order is executed (optional: the orders can be executed with a small fee commission, which will incentivize the solver to execute orders that do not have inherent profits to pay the IBC transfer) and the score achieved is executed: S_r^1.
5. The collateral of the solver is changed: $C \leftarrow \max\{0, S_2 - S_r^1\}$.

Solver algorithm with CFMM

1. Take the set of orders $\mathcal{O}$ and $p_{\min} = (1-\alpha)p_c$ and $p_{\max} = (1+\alpha)p_c$.
2. For each price p, we compute the aggregated demand $z(p) = \zeta(p) - \Delta_T$.
3. We compute the zeros ζ; let $p^\star_{min}$ and $p^\star_{\max}$ be the minimum and maximum of the set of zeros.
4. If $p^\star_{min}$ and $p^\star_{\max}$ are in $[p_{\min}, p_{\max}]$, then we use the matching algorithm for the price that maximizes volume.
5. Otherwise, we find an order in the CFMM that buys one asset in the CFMM and sells it in this mechanism in such a way that a) has some profits, and b) matches the aggregated supply and demand.

Therefore, each solver input is a subset of orders $\mathcal{O}' \subseteq \mathcal{O}$, an execution price $\tilde{p}$, and a private order (that consists of a route of trades with an initial set of assets or a trade with no limit price). A solution is feasible if it meets the following classifications:

- Incentive compatible: All solutions hold the optimization constraint. That is, for each

order (U_i, Δ_i) the amount obtained Δ_i' through the execution is a solution to the optimization problem associated with the order (or close enough by some parameter τ).

- Feasible: The route is feasible, and the price of execution holds the bounds.

The solutions are scored by the volume, i.e. $p \cdot [\sum_{i=1}^{n} \Delta_i']$, holding the constraints previously mentioned.
Collateral Formula: Order solutions by the committed solution $S_1 \geq S_2 \geq ....S_l$. Let S_1^r be the realized volume execution. Then, the minimum collateral needed by the first player is $\max\{0, S_2 - S_1^r\}$. However, S_1^r can not be observed in the scoring rule. Thus, we need a lower bound in terms of the private liquidity provided and the expected volatility in the CFMM provided in the paths. In the prior-free setting, $S_1^r \geq S_1^{pl}$, where S_1^{pl} consists of the best solution that the player can provide with the worst CFMM execution. The worst CFMM execution is provided as parameter in the solution.
The minimum collateral needed is:

$$C_{min} = V_2 - S_1^{pl}$$

4.3.4 Cost minimization of IBC transfers for 2-assets

In the computation of volume maximization, the solver is agnostic to where those assets are held. Therefore, an algorithm is needed to minimize the cost of IBC transfers. Let us assume that we have a set of chains $C_1, ..., C_m$ and a set of orders $O_1, ..., O_n$. Each order O_i offers some supply token in one chain j and demands the other token on some chain k. We will assume that the cost of IBC transfers from chain C_i to chain C_j is a $c_{ij} > 0$, and we will assume that the triangle inequality for costs is as follows:

$$c_{ij} \leq c_{ik} + c_{kj} \quad \forall i, j, k.$$

The optimization problem can then be described as follows:
Variables:

- x_{ij}^A: Amount of Asset A transferred from domain i to domain j.
- x_{ij}^B: Amount of Asset B transferred from domain i to domain j.
- y_{ij}: Binary variable that is 1 if there is any transfer (either Asset A or Asset B or both) from i to j, and 0 otherwise.
- p: Market clearing prices for Asset A and Asset B.
- c_{ij}: Cost of transferring any asset from domain i to domain j.
- s_i^A, s_i^B: Supply of Asset A and Asset B at domain i. I.e. this is the total number of assets A and B supplied in that chain.
- d_i^A, d_i^B: Demand for Asset A and Asset B at domain i. I.e. this is the total number of assets A and B demanded in that chain.

$$\text{Minimize} \sum_{i,j} c_{ij} y_{ij} \tag{16}$$

1. Supply constraints for each asset at each domain i:

$$\sum_j x_{ij}^A \leq s_i^A \quad \forall i$$

$$\sum_j x_{ij}^B \leq s_i^B \quad \forall i$$

2. Demand constraints for each asset at each domain j:

$$\sum_i x_{ij}^A \geq d_j^A \quad \forall j$$

$$\sum_i x_{ij}^B \geq d_j^B \quad \forall j$$

3. Binary constraint for y_{ij}:

$$y_{ij} \in \{0, 1\} \quad \forall i, j$$

4. Linking x_{ij}^A, x_{ij}^B and y_{ij}:

$$y_{ij} = \max\{\mathbb{1}_{x_{ij}^A > 0}, \mathbb{1}_{x_{ij}^B > 0}\}$$

In the max we are assuming that we can do both IBC transfers in one transaction. Otherwise, this would be a sum, i.e., $y_{ij} = \mathbb{1}_{x_{ij}^A > 0} + \mathbb{1}_{x_{ij}^B > 0}$.

5. Non-negativity:

$$x_{ij}^A \geq 0 \quad \forall i, j$$

$$x_{ij}^B \geq 0 \quad \forall i, j$$

We can solve this problem using Mixed-Integer Linear Programming (MILP).

5 Limitations & Future Areas of Research

This study, while contributing to the field of cross-domain frequent batch auctions and optimal routing in blockchain networks, acknowledges several areas that require further exploration and improvement:

1. **Generalization of Frequent Batch Auctions to Multiple Assets:** The current approach focuses on two-asset FBAs. Extending this to include multiple assets is essential for having a more efficient exchange and increase the overall welfare.

2. **Simulations for Optimal Routing:** Further research through extensive simulations is necessary to evaluate the efficiency and scalability of the proposed routing algorithm. Such simulations could provide invaluable insights for future enhancements.

3. **Cross-Domain Atomicity:** Achieving atomicity in cross-domain transactions is challenging yet vital for ensuring the reliability of cross-domain optimal routing solutions. Future research should investigate mechanisms to attain this atomicity alongside cross-domain auctions.

These directions for future research are critical for advancing our understanding and capabilities in facilitating efficient, scalable, and secure cross-domain transactions in blockchain networks.

6 Results and Conclusion

The results of this study demonstrate the effective optimization of cross-blockchain transactions through the proposed algorithm. It showcases enhanced efficiency and interoperability across blockchain networks. The application of the Arrow-Debreu model further contributes to the understanding and efficiency of multi-domain transactions. This research paves the way for more streamlined and effective approaches in the domain of cross-domain transactions.

References

[AD83] Kenneth J. Arrow and Gerard Debreu. "Existence of an equilibrium for a competitive economy". In: *Mathematical Economics: Twenty Papers of Gerard Debreu*. Econometric Society Monographs. Cambridge University Press, 1983, 68–97. DOI: 10.1017/CCOL052123736X.005.

[Nak08] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[BCS14] Eric Budish, Peter Cramton, and John Shim. "Implementation Details for Frequent Batch Auctions: Slowing Down Markets to the Blink of an Eye". In: *American Economic Review* 104.5 (2014), pp. 418–24. DOI: 10.1257/aer.104.5.418. URL: https://www.aeaweb.org/articles?id=10.1257/aer.104.5.418.

[But14] Vitalik Buterin. *A next-generation smart contract and decentralized application platform*. 2014. URL: https://ethereum.org/en/whitepaper/.

[TRO18] TRON DAO. *TRON: Advanced Decentralized Blockchain Platform*. Whitepaper Version 2.0. 2018. URL: https://tron.network/static/doc/white_paper_v_2_0.pdf.

[Yak18] Anatoly Yakovenko. *Solana: A new architecture for a high performance blockchain*. Tech. rep. The Solana Foundation, 2018. URL: https://solana.com/solana-whitepaper.pdf.

[Goe20] Christopher Goes. *The Interblockchain Communication Protocol: An Overview*. 2020. arXiv: 2006.15918 [cs.DC].

[NEA20] NEAR Foundation. *The NEAR White Paper*. 2020. URL: https://near.org/papers/the-official-near-white-paper.

[Ang+22] Guillermo Angeris et al. *Optimal Routing for Constant Function Market Makers*. 2022. arXiv: 2204.05238 [math.OC].

[GYB22] Christopher Goes, Awa Sun Yin, and Adrian Brink. *Anoma Whitepaper*. 2022. URL: https://github.com/anoma/whitepaper/blob/main/whitepaper.pdf.

[Ou+22] Wei Ou et al. "An overview on cross-chain: Mechanism, platforms, challenges and advances". In: *Computer Networks* 218 (2022), p. 109378. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2022.109378. URL: https://www.sciencedirect.com/science/article/pii/S1389128622004121.

[MK+23] Kashif Mehboob Khan et al. "Empirical Investigation on Blockchain Interoperability". In: *VFAST Transactions on Software Engineering* 11 (Mar. 2023), pp. 25–36. DOI: https://doi.org/10.21015/vtse.v10i4.1315.

[Ram+23] Geoffrey Ramseyer et al. *Augmenting Batch Exchanges with Constant Function Market Makers*. 2023. arXiv: 2210.04929 [cs.GT].

[Rei+23] Daniël Reijsbergen et al. *PIEChain – A Practical Blockchain Interoperability Framework*. 2023. arXiv: 2306.09735 [cs.CR].

[ZLQ23] Jincheng Zheng, David Kuo Chen Lee, and Dejun Quian. "An In-Depth Guide to Cross-Chain Protocols Under Multi-Chain World". In: *World Scientific Annual Review of Fintech* (2023). URL: https://ssrn.com/abstract=4476061.

[Cowa] *Coincidence of Wants*. Accessed: Nov. 6, 2023. URL: https://docs.cow.fi/overview/coincidence-of-wants.

[Cowb] *Introduction: Cow Protocol*. Accessed: Nov. 6, 2023. URL: https://docs.cow.fi/solvers/solvers.