Multi-token Batch Auctions with Uniform Clearing Prices

–

Features and Models

Tom Walther
tom.walther@gnosis.io

March 4, 2021

This document describes the problem of multi-token batch auctions with uniform clearing prices as a price-finding mechanism proposed for a decentralized token trading platform. Moreover, it contains solution approaches based on combinatorial optimization formulations, as well as some computational results.

Contents

1 Introduction

In continuous-time token exchange mechanisms, orders are typically collected in order books of two tokens that are traded against each other. A trade happens whenever a buy order of one token is matched by a sell order of the other, i.e., if there exists an exchange rate that satisfies the limit prices stated in the respective orders. In a setting of multiple tradeable tokens, one separate order book is required for every token pair combination. This may significantly limit liquidity for less frequently traded token pairs and lead to large bid-ask spreads and low trading volumes.

In our approach, we want to collect orders for a set of multiple tokens in a single joint order book and compute exchange prices for all token pairs simultaneously at the end of discrete time intervals (*multi-token batch auction*). Trades between the same token pairs are then all executed at the same exchange rate (*uniform clearing price*). Moreover, this mechanism enables so-called *ring trades*, where orders are matched along cycles of tokens. In order to exclude arbitrage opportunities, we require prices to be consistent along such cycles, i.e., we want the constraint

$$p_{j|k} \cdot p_{k|l} = p_{j|l} \tag{1}$$

to be satisfied for the exchange rates $p_{j|k}, p_{j|l}, p_{k|l}$ between all tokens τ_j, τ_k, τ_l.

The concept of frequent batch auctions aiming at reducing arbitrage and improving liquidity has been investigated extensively in [4]. Moreover, the advantages of uniform-price clearing have been discussed in [5]. In this document, we want to describe the problem of determining uniform clearing prices for all pairs of tokens involved in a multi-token batch auction process, and present a mixed-integer programming (MIP) solution approach.

2 Problem statement

In this section, we want to give a high-level description of the problem that we are investigating, thereby introducing some general notation and giving an overview of the constraints and properties that we will focus on.

2.1 Model components

Tokens & exchange rates Let $\mathcal{T} := \{\tau_1 \dots \tau_n\}$ denote the set of the n tokens that we want to consider. Additionally, we introduce an artificial token τ_0 that shall be referred to as *reference token*, which will become important in our modelling approaches later on in Section 3. For convenience of notation, we use $\mathcal{I}^t := \{1 \dots n\}$ to denote the indices of our token set.

The pairwise exchange rates between two tokens τ_j and τ_k will be denoted by $p_{j|k}$, meaning the price of one unit of τ_j measured in units of τ_k. As an example, if $p_{j|k} = 10$, one would need to pay an amount of 10 units of τ_k in order to purchase one unit of τ_j.

Orders Let there be a set $\mathcal{O} = \{\omega_1 \dots \omega_N\}$ of N orders in the batch to be processed and let $\mathcal{I}^o := \{1 \dots N\}$ denote its set of indices. Every order is specified as a tuple $\omega = (j, k, \overline{x}, \overline{y}, \pi)$ whose elements are to be understood as follows:

$j \in \mathcal{I}^t$	...	τ_j is the token to be bought,
$k \in \mathcal{I}^t$	...	τ_k is the token to be sold,
$\overline{x} \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$	...	maximum amount of τ_j to be bought,
$\overline{y} \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$	...	maximum amount of τ_k to be sold,
$\pi \in \mathbb{R}_{> 0} \cup \{+\infty\}$	...	limit exchange rate for order execution, i.e., $p_{j\|k} \leq \pi$.

Using this universal notation, we can express a variety of different order types:

1. *Limit buy orders* as $(j, k, \overline{x}, +\infty, \pi)$ with $\overline{x} < +\infty$ and $\pi < +\infty$:

   "Buy (at most) $\overline{x}$ units of token τ_j for τ_k if the exchange rate $p_{j|k}$ is at most π".

   The market participant sets a fixed maximum buy amount $\overline{x}$ and a limit exchange rate π, whereas the amount of tokens τ_k to be spent is implicitely bounded by $\overline{x} \cdot \pi$.

2. *Limit sell orders* as $(j, k, +\infty, \overline{y}, \pi)$ with $\overline{y} < +\infty$ and $\pi < +\infty$:

   "Sell (at most) $\overline{y}$ units of token τ_k for τ_j if the exchange rate $p_{k|j}$ is at least $1/\pi$".

The market participant sets a fixed maximum sell amount $\overline{y}$ and a limit exchange rate π, but does not have a limit on the amount of tokens τ_j to be obtained. The higher the exchange rate $p_{k|j}$, the more tokens τ_j the trader receives.

3. *Double-sided limit orders* as $(j, k, \overline{x}, \overline{y}, \pi)$ with $\overline{x} < +\infty$, $\overline{y} < +\infty$ and $\pi < +\infty$:

   "Spend (at most) $\overline{y}$ units of token τ_k to buy (at most) $\overline{x}$ units of token τ_j if the exchange rate $p_{j|k}$ is at most π".

   The market participant sets both a fixed maximum buy amount $\overline{x}$ and sell amount $\overline{y}$ as well as a limit price π. In contrast to a standard limit sell order, the trader prefers spending less tokens τ_k over receiving more tokens τ_j in case the exchange rate $p_{j|k}$ is significantly more favorable as anticipated.

4. *Market buy orders* as $(j, k, \overline{x}, +\infty, +\infty)$ with $\overline{x} < +\infty$:

   "Buy (at most) $\overline{x}$ units of token τ_j for τ_k at market rate.

   The market participant only sets a fixed maximum buy amount $\overline{x}$ and is ready to buy tokens τ_j at any market rate.

5. *Market sell orders* as $(j, k, +\infty, \overline{y}, +\infty)$ with $\overline{y} < +\infty$:

   "Sell (at most) $\overline{y}$ units of token τ_k for τ_j at market rate.

   The market participant only sets a fixed maximum sell amount $\overline{y}$ and is ready to sell tokens τ_k at any market rate.

6. *Double-sided market orders* as $(j, k, \overline{x}, \overline{y}, +\infty)$ with $\overline{x} < +\infty$ and $\overline{y} < +\infty$:

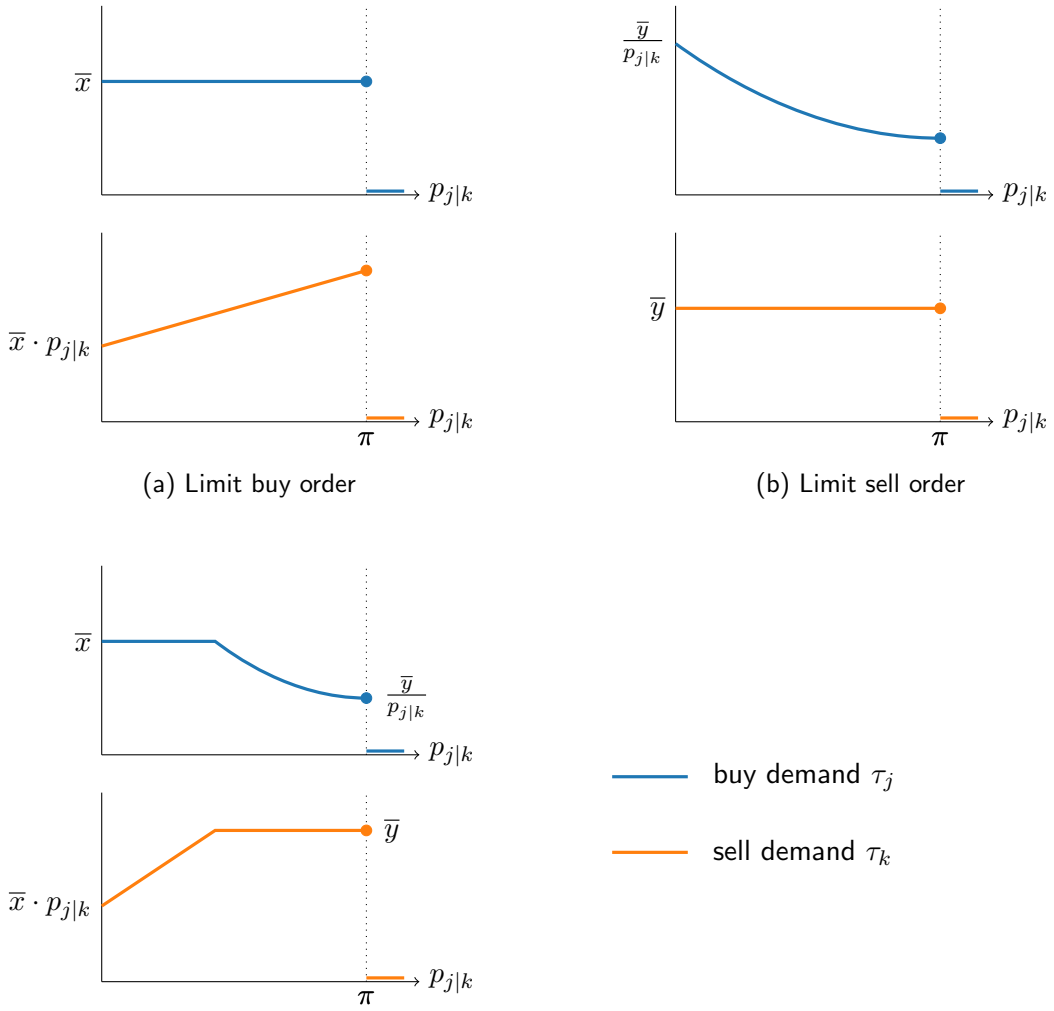
   "Spend (at most) $\overline{y}$ units of token τ_k to buy (at most) $\overline{x}$ units of token τ_j regardless of the exchange rate $p_{j|k}$.

   The market participant sets both a fixed maximum buy amount $\overline{x}$ and sell amount $\overline{y}$, but no limit price π. Hence, the trader expresses a willingness to a buy up to a certain amount of τ_j independent of the price per unit, but limits the total amount of τ_k that needs to be paid.

Figure 1 illustrates the characteristic buy- and sell-profiles of the three different limit order types depending on the exchange rate $p_{j|k}$.

Please note that orders with $j = k$, i.e., identical buy- and sell-token, imply trading with oneself. While such orders similarly fit into all of the following theory and models as any other orders, it might make sense to explicitely require $j \neq k$ from a semantic perspective.

Observation 2.1. *Assuming that the trader has constant utility for an arbitrary amount of buy-tokens τ_j, the following argument can be used to convert a limit buy order to a limit sell*



(a) Limit buy order

(b) Limit sell order

(c) Double-sided limit order

Figure 1: Buy- and sell-profiles for limit orders (types 1/2/3). The graphs show the transacted amounts of tokens with respect to the exchange rate $p_{j|k}$ in the case that the order is fully executed. The maximum buy-amount is computed as $\min\{\overline{x}, \overline{y}/p_{j|k}\}$, whereas the maximum sell-amount equals $\min\{\overline{y}, \overline{x} \cdot p_{j|k}\}$. The graphs look similar for market orders (types 4/5/6), with the only difference being the absence of a limit price π.

order: Consider a limit buy order $(j, k, \overline{x}, +\infty, \pi)$, which reveals that the trader would be ready to spend at most $\pi \cdot \overline{x}$ tokens τ_k in exchange for a maximum of tokens τ_j. Then, the trader would not object to receiving more than $\overline{x}$ tokens τ_j for a maximum of $\pi \cdot \overline{x}$ tokens τ_k, i.e., in case the exchange rate $p_{j|k}$ is strictly lower than π. Hence, the trader would only benefit from submitting a sell order $(j, k, +\infty, \pi \cdot \overline{x}, \pi)$.

2.2 Variables

The goal of our approach is to determine the exchange rates between all token pairs that allow for the best possible matching of orders (according to one of the metrics suggested in Section 2.4). Hence, both the exchange rates as well as the traded token amounts per order represent the generic set of variables for our problem.

For every order $\omega = (j, k, \overline{x}, \overline{y}, \pi) \in \mathcal{O}$, we introduce two variables $x \in \mathbb{R}_{\geq 0}$ and $y \in \mathbb{R}_{\geq 0}$ that denote the amount of tokens τ_j bought and the amount of tokens τ_k sold, respectively. Obviously, these need to satisfy the specified order amounts, i.e., $x \leq \overline{x}$ and $y \leq \overline{y}$.

Working directly with the exchange rates $p_{j|k}$ as variables for all token pairs (τ_j, τ_k) leads to nonlinear dependencies in the constraints that we are aiming to satisfy. Hence, for every token $\tau_j \in \mathcal{T}$, we introduce a new variable $p_j := p_{j|0}$ denoting the price of τ_j expressed in units of the reference token τ_0. The price p_0 of the reference token itself can be fixed to some constant in order to remove the excess degree of freedom. Without loss of generality, we will use $p_0 = 1$ throughout this paper. The relationship between the exchange rate and the price variables will be derived in Section 2.3.

2.3 Constraints

The solution that we are trying to find needs to satisfy several requirements that can be stated on a high level as follows:

- *(maximum buy and sell amount)*:
  for every order $\omega = (j, k, \overline{x}, \overline{y}, \pi) \in \mathcal{O}$: the amount of tokens bought and sold must not exceed the specified limits, i.e., $x \leq \overline{x}$ and $y \leq \overline{y}$.
- *(limit price)*:
  for every order $\omega = (j, k, \overline{x}, \overline{y}, \pi) \in \mathcal{O}$: the order can only be executed (fully or fractionally) if the exchange rate satisfies the limit price, i.e., $p_{j|k} \leq \pi$.
- *(uniform clearing price)*:
  for every order $\omega = (j, k, \overline{x}, \overline{y}, \pi) \in \mathcal{O}$: the order is executed using the exchange rate $p_{j|k}$ which is uniquely determined for each token pair (τ_j, τ_k), i.e., , $y = x \cdot p_{j|k}$.

- *(token balance)*:
  for every token $\tau \in \mathcal{T}$: the amount of tokens τ that were bought must equal the amount of tokens τ that were sold across all orders.
- *(price coherence)*:
  for all token pairs (τ_j, τ_k): $p_{j|k} \cdot p_{k|j} = 1$.
- *(arbitrage freeness)*:
  for all token triples (τ_j, τ_k, τ_l): $p_{j|k} \cdot p_{k|l} = p_{j|l}$.

Lemma 2.2. *(arbitrage freeness)* $\Rightarrow$ *(price coherence)*.

Proof. Consider three tokens $\{\tau_j, \tau_k, \tau_l\} \subset \mathcal{T}$. We apply the arbitrage-freeness condition twice:

$$\text{(i)}\ p_{j|k} \cdot p_{k|l} = p_{j|l}$$
$$\text{(ii)}\ p_{j|l} \cdot p_{l|k} = p_{j|k}$$

Inserting (i) into (ii) and assuming prices to be strictly positive yields

$$p_{j|k} \cdot p_{k|l} \cdot p_{l|k} = p_{j|k} \qquad \Leftrightarrow \qquad p_{k|l} \cdot p_{l|k} = 1$$

□

Notice that the above constraints also imply $p_{j|j} = 1$ for every token $\tau_j \in \mathcal{T}$.

(Relative) Exchange rates vs. (absolute) token prices Moreover, the arbitrage-freeness and price-coherence conditions help establish a relation between the exchange rate variable $p_{j|k}$ for token pair (τ_j, τ_k) and the price variables p_j and p_k for the individual tokens (as defined in Section 2.2):

$$p_{j|k} = p_{j|0} \cdot p_{0|k} = \frac{p_{j|0}}{p_{k|0}} = \frac{p_j}{p_k}. \tag{2}$$

Consequently, working with token price variables rather than pairwise exchange rates automatically guarantees arbitrage-freeness across all tokens. Moreover, the uniform clearing price condition can be rewritten as $\frac{p_j}{p_k} \leq \pi$, while the uniform clearing price constraint becomes $x \cdot p_j = y \cdot p_k$. Both conditions together yield an equivalent expression for the limit price: $x \cdot \pi \geq y$.

Reference token We now have some freedom to decide what precisely the artificial reference token should represent. For example, we could select one particular $\tau_j \in \mathcal{T}$ to be the *numeraire*, i.e.,

$$p_0 = p_j. \tag{3a}$$

Then, all trading volume would be computed with respect to τ_j, making this the principal token in the batch auction. Alternatively and more generally, we could also identify τ_0 with a weighted sum of all participating tokens in $\mathcal{T}$, i.e.,

$$p_0 = \sum_{j \in \mathcal{I}^t} \gamma_j \cdot p_j, \tag{3b}$$

with weights $\gamma_j > 0$. This way, it would be possible to give similar importance to all tokens, e.g., by using weights $\gamma_j = 1/(n \cdot p_j^{\text{old}})$, where p_j^{old} denotes the price of token τ_j found in the previous batch auction iteration.

Observation 2.3. *The choice of representation of the reference token has an influence on the solution. This can be seen by considering a single token pair (τ_j, τ_k) with only two buy orders $\omega_1 = (j, k, 1.0, +\infty, 2.0)$ and $\omega_2 = (k, j, 1.5, +\infty, 1.0)$. Both orders can be matched if $p_{j|k} \in [1, 2]$. However, if we choose token τ_j to be the reference token, a maximum of $1.0\,\tau_j$ can be transacted for prices $p_{j|k} \in [1, 1.5]$, whereas the maximum trading volume of $1.5\,\tau_k$ can be achieved when $p_{j|k} \in [1.5, 2]$.*

2.4 Objectives

As mentioned previously, for a given batch of orders $\mathcal{O}$ on the set of tokens $\mathcal{T}$, it is our goal to determine the exchange rates $p_{j|k}$ for all token pairs (τ_j, τ_k) together with an order matching such that a certain optimization criterion is maximized. We are considering the following three objective functions:

Trading volume The most obvious optimization criterion would be to target a solution that enables as much trade as possible. Due to their different prices, we can not simply maximize for the total number of tokens traded, but need a common scale as a measure of value. Hence, for every order $\omega = (j, k, \overline{x}, \overline{y}, \pi) \in \mathcal{O}$, a new variable $v \in \mathbb{R}_{\geq 0}$ representing the *trading volume* is introduced, defined as

$$v := x \cdot p_j = y \cdot p_k. \tag{4}$$

In words, the trading volume measures the value of all transactions in terms of units of the reference token τ_0. The objective function comprises the sum of the trading volumes of all orders.

Social surplus/welfare As a downside, the trading volume does not take the limit prices of orders into account. For orders on the same token pair, traders might expect the ones with the best limit prices to be filled with preference over those whose limit prices are very close to the determined exchange rate. A measure that captures the valuation of the order executed by the trader is called *social surplus* or *social welfare*, denoted by a variable $w \in \mathbb{R}_{\geq 0}$ per order. It uses the amount x of tokens τ_j bought, determines how many tokens τ_k the trader would have been ready to spend (via the limit price π), and values this amount with the token price p_k:

$$w := x \cdot \pi \cdot p_k. \tag{5}$$

Again, the objective function is the sum of all w.

Trader's utility Finally, combining trading volume and valuation, another measure called *trader's utility* could be used as objective. We will denote it by $u \in \mathbb{R}_{\geq 0}$ and it can be considered in two variants:

- the difference between the amount that the trader would have been ready to spend for an amount of x buy-tokens τ_j and the amount y that is actually spent (weighted by the price p_k of the sell-token τ_k):

$$u_A := (x \cdot \pi - y) \cdot p_k \tag{6a}$$

- the difference between the amount x of tokens τ_j bought and the amount that the trader would have accepted as a minimum for spending y tokens τ_k (weighted by p_j):

$$u_B := (x - y/\pi) \cdot p_j \tag{6b}$$

Notice that definition (6a) yields $u = w - v$.

The uniform clearing price condition ($x \cdot p_j = y \cdot p_k$) allows to rewrite formulation (6a) as $u_A = x \cdot (\pi \cdot p_k - p_j)$, making it intuitively more suitable for buy orders. Analogously, definition (6b) becomes $u^B = y \cdot (p_k - \frac{p_j}{\pi})$ and will primarily be used for sell orders.

By definition, both utility functions evaluate to zero if $p_{j|k} = \pi$, i.e., whenever the exchange rate between the tokens τ_j and τ_k equals the limit price of the order. This means that any market participant would be indifferent to trading at limit price or not trading at all. If this seems undesirable and trading opportunities should always be favoured over no trade, the utility functions can slightly be adjusted to return positive values for all exchange rates satisfying the limit price:

$$u_A^{(\alpha)} := (x \cdot \pi - \frac{y}{1+\alpha}) \cdot p_k = x \cdot (\pi \cdot p_k - \frac{p_j}{1+\alpha})), \tag{6a'}$$

$$u_B^{(\alpha)} := ((1+\alpha) \cdot x - \frac{y}{\pi}) \cdot p_j = y \cdot ((1+\alpha) \cdot p_k - \frac{p_j}{\pi}), \tag{6b'}$$

for some parameter $\alpha > 0$.

Disregarded utility Trader's utility as objective criterion assigns some value to all orders that are executed. However, it does not account for orders that were *not* (fully) executed even though their limit price was satisfied, i.e., because their was no matching counterpart. In some cases, the resulting prices seem somewhat arbitrary. For example, if two orders of different size on one token pair can be matched directly according to their limit prices, it is unclear what the optimal exchange rate will be. It could either be equal to one of the two limit prices or lie somewhere inbetween. In this case, our intuition is that the larger of the two orders should determine the optimal exchange rate. For this purpose, we introduce the concept of *disregarded utility*, that essentially represents the difference between the maximum utility that a trader could get at given prices, and the actual utility gained by the proposed order execution.

We define the maximum utility as

$$u_A^{\max} := \max\{\overline{x} \cdot (\pi \cdot p_k - p_j);\, 0\} \qquad \text{if } \overline{x} < +\infty, \tag{8a}$$

$$u_B^{\max} := \max\{\overline{y} \cdot (p_k - \frac{p_j}{\pi});\, 0\} \qquad \text{if } \overline{y} < +\infty. \tag{8b}$$

Disregarded utility can then be written as $u_{\{A|B\}}^{\max} - u_{\{A|B\}}$.

Notice that the α-adjustment as described in the case of pure trader's utility can straightforwardly be incorporated into the disregarded utility formulas as well.

It is an open question if there always exists a solution with zero disregarded utility.

2.5 Example

A typical instance of our batch auction problem could look like shown in Figure 2. It is assumed that some token pairs will encounter high trading demand whereas other will only rarely be traded on.

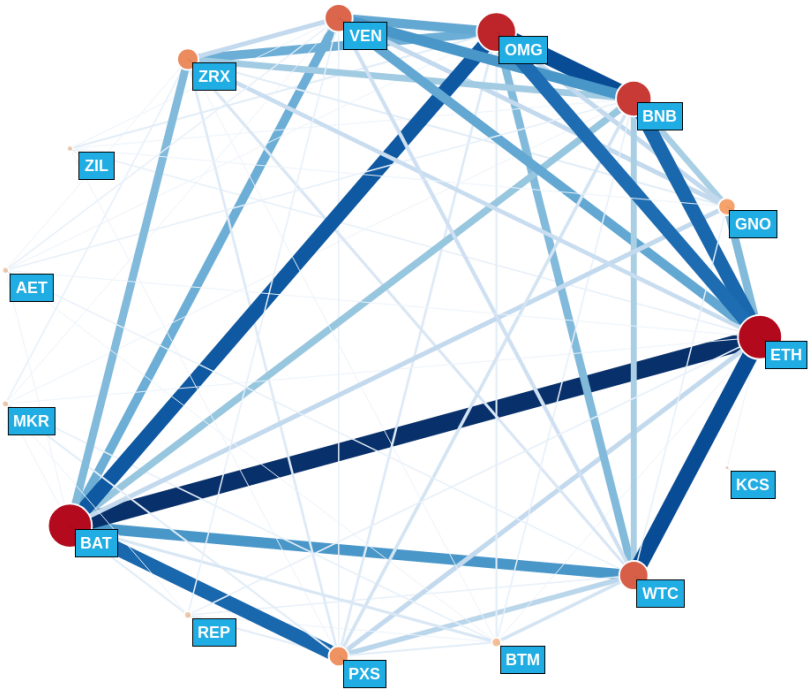


Figure 2: Token-order-graph representing a multi-token batch auction instance. Nodes correspond to tokens (with their size indicating the trading demand), the thickness of an edge represents the number of orders placed on the respective token pair.

3 Models and Formulations

In this section, we want to present and discuss several solution approaches for our batch auction problem in terms of mathematical optimization formulations.

Data At first, we will express all information given in the orders in terms of data matrices and vectors, aiming at being able to write down a complete optimization model. Recall that every order $\omega \in \mathcal{O}$ is defined by a tuple $(j, k, \overline{x}, \overline{y}, \pi)$ specifying the buy- and sell-token, the maximum amount of tokens to be bought and sold, as well as the limit price.

We introduce two data matrices

$$\begin{aligned} \mathbf{T}^b \in \{0,1\}^{N \times n} \quad &\text{with} \quad \mathbf{T}^b \ni t^b_{i,j} = 1 \Leftrightarrow \text{token } \tau_j \text{ to be bought in order } \omega_i \\ \mathbf{T}^s \in \{0,1\}^{N \times n} \quad &\text{with} \quad \mathbf{T}^s \ni t^s_{i,j} = 1 \Leftrightarrow \text{token } \tau_j \text{ to be sold in order } \omega_i \end{aligned}$$

For now, we are only considering orders of one token against one other, so there must be exactly one entry equal to 1 per row (order) in both $\mathbf{T}^b$ and $\mathbf{T}^s$. An extension to so-called *basket orders* where multiple tokens are traded against multiple others is discussed as an extension Section 4.3.

Let $(\overline{x}_i) =: \overline{\mathbf{x}} \in \mathbb{R}^N_{\geq 0}$ and $(\overline{y}_i) =: \overline{\mathbf{y}} \in \mathbb{R}^N_{\geq 0}$ contain the maximum amounts of tokens to be bought and sold, respectively, in every order.

Finally, the limit prices of all orders shall be stored as vector $(\pi_i) =: \boldsymbol{\pi} \in \mathbb{R}^N_{\geq 0}$, where $\pi_i \in \boldsymbol{\pi}$ refers to the exchange rate between the respective buy- and sell-tokens at which order ω_i may be executed (according to the definition in Section 2).

Variables Let $(p_j) =: \mathbf{p} \in \mathbb{R}^n_{>0}$ be a vector containing all token prices to be determined. For the orders, we use the following notation:

$$\begin{aligned} (x_i) =: \mathbf{x} \in \mathbb{R}^N_{\geq 0} \quad &\ldots \text{ number of tokens bought} \\ (y_i) =: \mathbf{y} \in \mathbb{R}^N_{\geq 0} \quad &\ldots \text{ number of tokens sold} \\ (v_i) =: \mathbf{v} \in \mathbb{R}^N_{\geq 0} \quad &\ldots \text{ trading volume} \\ (w_i) =: \mathbf{w} \in \mathbb{R}^N_{\geq 0} \quad &\ldots \text{ social surplus} \\ (u_i) =: \mathbf{u} \in \mathbb{R}^N_{\geq 0} \quad &\ldots \text{ trader's utility} \end{aligned}$$

Constraints We need to distinguish two different behaviours for every order: Either the exchange rate on the respective token pair satisfies its limit price (and thus the order may be

executed), or it does not (thus the order needs to remain untouched). Optionally, we could incorporate a minimum fraction of execution, denoted by a parameter $r^{\min} \in [0, 1]$, for the case that an order may be executed. This parameter may be set globally for all orders, or for every order individually. In the following, however, we assume that there is no minimum order execution fraction imposed, i.e., $r^{\min} = 0$. We will discuss the impact of $r^{\min} > 0$ in Section 4.

In general, for an order given as $\omega = (j, k, \overline{x}, \overline{y}, \pi)$, this yields the following two sets of constraints:

$$\mathfrak{C}^1 := \left\{ \begin{array}{rcl} x & \leq & \overline{x} \\ y & \leq & \overline{y} \\ x \cdot p_j & = & y \cdot p_k \\ \frac{p_j}{p_k} & \leq & \pi \end{array} \right\} \quad \text{and} \quad \mathfrak{C}^0 := \left\{ \begin{array}{rcl} x & = & 0 \\ y & = & 0 \\ \frac{p_j}{p_k} & > & \pi \end{array} \right\} \tag{9}$$

Objective As introduced in Section 2.4, we consider four different objectives:

$$\begin{aligned} &\max \sum_{i \in \mathcal{I}^o} v_i && \ldots \text{ total trading volume} \\ &\max \sum_{i \in \mathcal{I}^o} w_i && \ldots \text{ total social surplus} \\ &\max \sum_{i \in \mathcal{I}^o} u_i && \ldots \text{ total traders' utility} \\ &\min \sum_{i \in \mathcal{I}^o} (u_i^{\max} - u_i) && \ldots \text{ disregarded utility} \end{aligned}$$

Optional conditions In some situations, it might make sense to bound the deviation of the new exchange rates $p_{j|k}$ from the previously computed exchange rates $p_{j|k}^{\text{old}}$. Therefore, let p_j^{old} denote the price of token τ_j found in the previous batch auction iteration. Using a maximum fluctuation parameter $\delta > 0$, we impose the conditions

$$\begin{aligned} p_{j|k} \in \left[\left(\frac{1}{1+\delta} \right) p_{j|k}^{\text{old}}, (1+\delta)\, p_{j|k}^{\text{old}} \right] \quad &\Leftrightarrow \quad \frac{p_j}{p_k} \in \left[\left(\frac{1}{1+\delta} \right) \frac{p_j^{\text{old}}}{p_k^{\text{old}}}, (1+\delta)\, \frac{p_j^{\text{old}}}{p_k^{\text{old}}} \right] \\ &\Leftrightarrow \quad \begin{cases} p_j \geq \left(\frac{1}{1+\delta} \right) \frac{p_j^{\text{old}}}{p_k^{\text{old}}}\, p_k \\ p_j \leq (1+\delta)\, \frac{p_j^{\text{old}}}{p_k^{\text{old}}}\, p_k \end{cases} . \end{aligned} \tag{10}$$

As an example, $\delta = 1$ would set the bounds to half/twice the previous exchange rates. Note that, if no previous price exists for some token τ_j, the exchange rate bounds for token pairs

involving τ_j could be determined on the basis of the limit prices given in the respective orders, or simply be set to $[0, \infty]$. In theory, the model permits a different fluctuation parameter on every token pair, which allows to capture real-world expections, e.g., , related to the liquidity that is available on each token pair.

Moreover, it is helpful to incorporate some explicit lower and upper bound for the price of every token τ_j, so let us require $p_j \in [\underline{p}_j, \overline{p}_j]$.

Lemma 3.1. *For the choice of weights $\gamma_j = 1/(n \cdot p_j^{\text{old}})$ in (3b) and a given maximum fluctuation parameter δ, we can impose bounds*

$$p_j \in \left[\left(\frac{1}{1+\delta}\right) p_j^{\text{old}}, (1+\delta)\, p_j^{\text{old}}\right] \forall\, j \in \mathcal{I}^t. \tag{11}$$

Proof. We will prove that the upper bound holds. Proving the lower bound works analogous. Assume $p_j > (1+\delta)\, p_j^{\text{old}}$. Then, (10) yields

$$(1+\delta)\, \frac{p_j^{\text{old}}}{p_k^{\text{old}}}\, p_k > (1+\delta)\, p_j^{\text{old}} \qquad \Leftrightarrow \qquad p_k > p_k^{\text{old}} \qquad \forall k \neq j.$$

Inserting into (3b) leads to a contradiction:

$$p_0 = 1 = \frac{1}{n \cdot p_j^{\text{old}}}\, p_j + \sum_{j \neq k \in \mathcal{I}^t} \frac{1}{n \cdot p_k^{\text{old}}} \cdot p_k \; > \; \frac{1+\delta}{n} + \sum_{j \neq k \in \mathcal{I}^t} \frac{1}{n} > 1.$$

□

All price bounds shall be stored in vectors $\underline{\mathbf{p}}$ and $\overline{\mathbf{p}}$, respectively.

3.1 Nonlinear programming model

When looking at the sets of constraints (9), the most intuitive model seems to be a nonlinear program. As we will see, it is possible to avoid binary variables. For simplicity, we will state this (and all the following formulations) using total trading volume as objective function. The other objectives can be used by inserting and calculating the respective variables.

$$\text{maximize} \quad \sum_{i \in \mathcal{I}^o} v_i \tag{12a}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{I}^o} t^b_{i,j}\, x_i = \sum_{i \in \mathcal{I}^o} t^s_{i,j}\, y_i \qquad \forall\, j \in \mathcal{I}^t \tag{12b}$$

$$v_i = x_i \sum_{j \in \mathcal{I}^t} t^b_{i,j}\, p_j \qquad \forall\, i \in \mathcal{I}^o \tag{12c}$$

$$v_i = y_i \sum_{j \in \mathcal{I}^t} t^s_{i,j}\, p_j \qquad \forall\, i \in \mathcal{I}^o \tag{12d}$$

$$x_i \leq \overline{x}_i \qquad \forall\, i \in \mathcal{I}^o \tag{12e}$$

$$y_i \leq \overline{y}_i \qquad \forall\, i \in \mathcal{I}^o \tag{12f}$$

$$y_i \leq x_i\, \pi_i \qquad \forall\, i \in \mathcal{I}^o \tag{12g}$$

$$\sum_{j \in \mathcal{I}^t} \gamma_j \cdot p_j = 1 \tag{12h}$$

$$p_j \in [\underline{p_j}, \overline{p_j}] \qquad \forall\, j \in \mathcal{I}^t \tag{12i}$$

$$x_i, y_i, v_i \in \mathbb{R}_{\geq 0} \qquad \forall\, i \in \mathcal{I}^o \tag{12j}$$

The objective function (12a) maximizes the total volume in terms of units of the reference token τ_0 that is processed with all orders.

Constraint (12b) ensures that the total numbers of tokens bought and sold are equal for every token across all orders. The summations in this constraint are only responsible for selecting the correct tokens that are traded in the orders.

The constraints (12c) and (12d) compute the buy and sell trade volume for every order with respect to the reference token, and make sure these two are equal. This guarantees that the token prices are chosen such that they are consistent with the traded amounts of tokens. If the traded token amounts x_i and y_i are zero for some order ω_i, i.e., ω_i is not executed at

all, the corresponding trade volume v_i will be zero as well. However, this comes at the price of introducing nonlinearity (and even nonconvexity) into the model.

The maximum token amounts to be bought and sold as specified in every order are incorporated into the model via the constraints (12e) and (12f). Constraint (12g) ensures that the limit price is satisfied, if the order is to be executed, or that both x_i and y_i are set to zero otherwise. Please note that if either of $\overline{x}$, $\overline{y}$ or π_i equals $+\infty$, the corresponding constraint is trivially satisfied. Altogether, the constraints (12c)–(12g) capture both cases of (9) simultaneously.

Finally, as described earlier, the constraint (12h) specifies the representation of the reference token.

3.2 Mixed-integer linear programming model I

Since nonlinearity and nonconvexity usually pose strong restrictions on the tractable model size, we want to proceed with proposing a mixed-integer linear programming (MIP) formulation as an alternative to the NLP model presented above. Thereby, the challenge is to find an equivalent linear formulation for (9).

The key idea is to avoid explicitely computing the values x and y for every order $\omega = (j, k, \overline{x}, \overline{y}, \pi)$, but only working with the trading volume v as defined in (4) instead. With uniform clearing prices, x and y can be computed unambiguously from the value of v later on. The order constraints (9) can be expressed as follows:

$$\tilde{\mathfrak{C}}^1 := \left\{ \begin{array}{rcl} v & \leq & \overline{x} \cdot p_j \\ v & \leq & \overline{y} \cdot p_k \\ \frac{p_j}{p_k} & \leq & \pi \end{array} \right\} \quad \text{and} \quad \tilde{\mathfrak{C}}^0 := \left\{ \begin{array}{rcl} v & = & 0 \\ \frac{p_j}{p_k} & > & \pi \end{array} \right\} \tag{13}$$

Please notice that this step is not possible for the social surplus w as in (5) and the trader's utility u as in (6). Hence, we have not found a linear formulation for that maximize these two optimization criteria.

In order to reformulate (13) in terms of a mixed-integer linear program, we first introduce a binary variable $z \in \{0, 1\}$ for every order and associate its states to the constraint sets as $\{z = 1\} \Leftrightarrow \tilde{\mathfrak{C}}^1$ and $\{z = 0\} \Leftrightarrow \tilde{\mathfrak{C}}^0$. For modelling these dependencies, there exist two major strategies that we will apply in the following: the *big-M* approach as, e.g., mentioned in [3], as well as *disjunctive programming*, introduced by Balas [2].

Big-M reformulation In the big-M approach, the constraints for both $\tilde{\mathfrak{C}}^0$ and $\tilde{\mathfrak{C}}^1$ are modified so that they are equivalent to the original ones if the binary variable takes the respective value, and are rendered redundant by large-enough constants in the opposite case. As for our constraint systems, this translates into:

$$p_j \leq \pi\, p_k + (1 - z)(\overline{p}_j - \pi \underline{p}_k) \tag{14a}$$

$$p_j \geq (\pi + \varepsilon)\, p_k + z\,(\underline{p}_j - (\pi + \varepsilon)\, \overline{p}_k) \qquad \text{if } \pi < +\infty \tag{14b}$$

$$v \leq \overline{x}\, p_j \tag{14c}$$

$$v \leq \overline{x}\, \overline{p}_j\, z \qquad \text{if } \overline{x} < +\infty \tag{14d}$$

$$v \leq \overline{y}\, p_k \tag{14e}$$

$$v \leq \overline{y}\, \overline{p}_k\, z \qquad \text{if } \overline{y} < +\infty \tag{14f}$$

It can easily be verified that substituting $z = 1$ and $z = 0$ yields the desired constraints (13). Notice that the strict inequality $\frac{p_j}{p_k} > \pi$ in $\tilde{\mathfrak{C}}^0$ is approximated by an inequality $\frac{p_j}{p_k} \geq \pi + \varepsilon$, with some very small $\varepsilon > 0$.

Disjunctive programming reformulation The advantage of the big-M reformulation is its simplicity and compactness. However, its relaxation is not as tight as it can be, i.e., it does not describe the convex hull of the feasible regions of $\tilde{\mathfrak{C}}^0$ and $\tilde{\mathfrak{C}}^1$. This property can be secured in the disjunctive programming approach through the addition of additional auxiliary variables.

For every order $\omega = (j, k, \overline{x}, \overline{y}, \pi)$, we introduce two pairs of auxiliary nonnegative price variables: $p^{b,0}, p^{b,1}$ – referring to the price p_j of the buy-token, as well as $p^{s,0}, p^{s,1}$ – referring to the price p_k of the sell-token. Then, a disjunctive programming formulation can be given as follows:

$$(1 - z)\,\underline{p}_j \leq p^{b,0} \leq (1 - z)\,\overline{p}_j \tag{15a}$$
$$(1 - z)\,\underline{p}_k \leq p^{s,0} \leq (1 - z)\,\overline{p}_k \tag{15b}$$
$$z\,\underline{p}_j \leq p^{b,1} \leq z\,\overline{p}_j \tag{15c}$$
$$z\,\underline{p}_k \leq p^{s,1} \leq z\,\overline{p}_k \tag{15d}$$
$$p_j = p^{b,0} + p^{b,1} \tag{15e}$$
$$p_k = p^{s,0} + p^{s,1} \tag{15f}$$
$$p^{b,1} \leq \pi\, p^{s,1} \tag{15g}$$
$$p^{b,0} \geq (\pi + \varepsilon)\, p^{s,0} \tag{15h}$$
$$v \leq \overline{x}\, p^{b,1} \tag{15i}$$
$$v \leq \overline{y}\, p^{s,1} \tag{15j}$$

The general idea is that both $p^{b,1}$ and $p^{s,1}$ shall take the true value of p_j and p_k within the respective bounds if $z = 1$ (i.e., the order can be executed), and be set to zero otherwise; and vice-versa for $p^{s,0}$ and $p^{b,0}$. This is modelled through constraints (15a)–(15d). The aggregation of the auxiliary price variables to the actual token prices is expressed in constraints (15e) and (15f). All the other constraints (15g)–(15j) represent the original model constraints from (13), now expressed in terms of the auxiliary variables.

Model Using either of the two reformulations of (13) for all orders as a substitution for (16c), the full MIP model can be stated as:

$$\text{maximize} \quad \sum_{i \in \mathcal{I}^o} v_i \tag{16a}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{I}^o} t^b_{i,j}\, v_i = \sum_{i \in \mathcal{I}^o} t^s_{i,j}\, v_i \qquad \forall\, j \in \mathcal{I}^t \tag{16b}$$

$$\tilde{\mathfrak{C}}^1_i \vee \tilde{\mathfrak{C}}^0_i \qquad \forall\, i \in \mathcal{I}^o \tag{16c}$$

$$\sum_{j \in \mathcal{I}^t} \gamma_j \cdot p_j = 1 \tag{16d}$$

$$p_j \in [\underline{p_j}, \overline{p_j}\,] \qquad \forall\, j \in \mathcal{I}^t \tag{16e}$$

$$v_i \in \mathbb{R}_{\geq 0} \qquad \forall\, i \in \mathcal{I}^o \tag{16f}$$

Additional inequalities The model does not take into account that there exist dependencies between orders on the same token pair as well as across adjacent token pairs. In the following, we will derive valid inequalities that capture some of these dependencies.

First, whenever an order with a certain limit price π is executed (fully or partially) at some determined market price, all orders with the same buy- and sell-token and a with a better (i.e., higher) limit price must also be executed. Conversely, if some order can not be executed, all orders with worse (i.e., lower) limit prices may not be executed as well. In particular, let $\omega_{i_1} = (j, k, \cdot, \cdot, \pi_{i_1})$ and $\omega_{i_2} = (j, k, \cdot, \cdot, \pi_{i_2})$ be two orders buying token τ_j for τ_k with $\pi_{i_1} \leq \pi_{i_2}$. Then, we can relate the corresponding binary variables as $z_{i_1} \leq z_{i_2}$.

This idea induces a natural ordering of the orders on every token pair by their limit price. Let there be $N_{j,k}$ orders $(\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_{N_{j,k}}})$ on token pair (τ_j, τ_k), i.e., orders that buy token τ_j for τ_k. Assuming that these orders are sorted increasingly by their limit prices

$$\pi_{i_1} \leq \pi_{i_2} \leq \ldots \leq \pi_{i_{N_{j,k}}},$$

this yields the chain of inequalities

$$z_{i_1} \leq z_{i_2} \leq \ldots \leq z_{i_{N_{j,k}}}. \tag{17}$$

Hence, there are only $N_{j,k} - 1$ non-redundant such inequalities per token pair and thus always less than N overall. Please note that if two orders ω_{i_1} and ω_{i_2} are given with equal limit prices $\pi_{i_1} = \pi_{i_2}$, the equality $z_{i_1} = z_{i_2}$ needs to hold.

Other types of valid inequalities can be derived when considering two orders going in the opposite direction, i.e., $\omega_{i_1} = (j, k, \cdot, \cdot, \pi_{i_1})$ buying token τ_j for τ_k and $\omega_{i_2} = (k, j, \cdot, \cdot, \pi_{i_2})$

vice-versa. Then, if $\pi_{i_1} \cdot \pi_{i_2} < 1$, we can conclude that ω_{i_1} and ω_{i_2} can not both be executed at same time, because their feasible price ranges are disjoint. Hence, we can impose

$$z_{i_1} + z_{i_2} \leq 1.$$

Conversely, $\pi_{i_1} \cdot \pi_{i_2} \geq 1$ would imply that every exchange rate between τ_j and τ_k allows executing at least one of the two orders, which yields

$$z_{i_1} + z_{i_2} \geq 1.$$

These two types of inequalities can also be generalized to chains of orders along cycles of tokens. Let $(\tau_{j_1}, \tau_{j_2}, \ldots, \tau_{j_m}, \tau_{j_1})$ be such a token cycle of length m, and let there be a set of orders $(\omega_{i_1} = (j_1, j_2, \cdot, \cdot, \pi_{i_1}), \omega_{i_2} = (j_2, j_3, \cdot, \cdot, \pi_{i_2}), \ldots, \omega_{i_m} = (j_m, j_1, \cdot, \cdot, \pi_{i_m}))$. Now, if $\prod_{l=1}^{m} \pi_{i_l} < 1$, our arbitrage-freeness requirement again forbids that all orders can be executed at the same time. Hence, we can write

$$\sum_{l=1}^{m} z_{i_l} \leq m - 1. \tag{18}$$

In the other case where $\prod_{l=1}^{m} \pi_{i_l} \geq 1$, we know that at least one of the orders needs to be allowed being executed, which yields

$$\sum_{l=1}^{m} z_{i_l} \geq 1. \tag{19}$$

In both cases (18) and (19), we can use the inequalities (17) in order to detect redundancies. However, even the numbers of non-redundant inequalities of these types grow significantly with the length of the token cycles that are being considered. For our practical purposes, we have incorporated inequalities for cycles of length $m = \{2, 3\}$ and observed remarkable performance gains.

Describe how exactly non-redundant inequalities are determined!?

Computational results!

3.3 Mixed-integer linear programming model II

In the previous MIP model (16), we have modelled the constraints of every order separately, i.e., we have used one binary disjunctive formulation per order that controls its behaviour in case it may or may not be executed. On top of that, we have used the equations (17) to indicate the dependencies between orders on the same token pair. However, we can also encode these dependencies deeper into the model by considering order volumes within certain price ranges on a token pair in a somewhat aggregated fashion. This approach gives rise to an alternative MIP formulation that we will present in the following.

As before, let $\mathcal{I}^p := \{(j,k) \in \mathcal{I}^t \times \mathcal{I}^t, j < k\}$ denote the index set of (undirected) pairs of tokens. We want to aggregate orders on every token pair, so we collect all limit buy and sell orders on (j,k), i.e., all $\omega = (j,k,\cdot,\cdot,\pi) \in \mathcal{O}$ and $\omega = (j,k,\cdot,\cdot,\pi) \in \mathcal{O}$, and sort them in an increasing order of their limit prices. Assuming that there are m orders for token pair (j,k), we get

$$\underline{p}_{j|k} =: \pi^{(0)}_{j,k} < \pi^{(1)}_{j,k} < \pi^{(2)}_{j,k} < \ldots < \pi^{(m)}_{j,k} < \pi^{(m+1)}_{j,k} := \overline{p}_{j|k}, \tag{20}$$

where $\underline{p}_{j|k}$ and $\overline{p}_{j|k}$ are explicit but somewhat arbitrary bounds on the exchange rate (e.g., half and double the previous rate). The above ordering (20) defines regions $r_l := [\pi^{(l)}, \pi^{(l+1)}]$, $l \in \{0 \ldots m\}$, for the exchange rate $p_{j|k}$. Notice that, without loss of generality and for simplicity of notation, we assume the limit prices of all orders to be pairwise different, so that the strict inequalities in (20) hold. If some limit prices were equal, we would only end up with less exchange rate regions. Let $\mathcal{I}^r_{j,k} := \{0 \ldots m\}$ denote the set of all such regions for every token pair (j,k), i.e., $l \in \mathcal{I}^r_{j,k}$ refers to the interval r_l of that token pair.

Now, let $\tilde{x}^{(1)}_{j,k,l}$ denote to the cumulated amount of tokens τ_j that could be bought across all limit buy orders $\omega = (j,k,\overline{x},\cdot,\cdot)$ (with $\overline{x} < +\infty$) if the exchange rate $p_{j|k}$ were in r_l. Conversely, let $\tilde{x}^{(2)}_{j,k,l}$ denote the cumulated number of available tokens τ_k in all limit buy orders $\omega = (k,j,\overline{x},\cdot,\cdot)$. Both shall be stored in vectors $\tilde{\mathbf{x}}^{(1)} := (\tilde{x}^{(1)}_{j,k,l})$ and $\tilde{\mathbf{x}}^{(2)} := (\tilde{x}_{j,k,l})^{(2)}$. Analogously, we aggregate available sell tokens for limit sell orders $\omega = (j,k,\cdot,\overline{y},\cdot)$ and $\omega = (k,j,\cdot,\overline{y},\cdot)$, and store them in $\tilde{\mathbf{y}}^{(1)} := (\tilde{y}^{(1)}_{j,k,l})$ and $\tilde{\mathbf{y}}^{(2)} := (\tilde{y}^{(2)}_{j,k,l})$, respectively. From here on, we are only working with this aggregated order representation.

In terms of variables, we will use the same price variables $\mathbf{p} \in \mathbb{R}^n_{\geq 0}$ as previously. In addition, we will use nonnegative variables $\mathbf{v}^A := (v^A_{j,k,l})$ and $\mathbf{v}^B := (v^B_{j,k,l})$ to represent trading volumes for every price interval r_l on every token pair (j,k). Moreover, we represent the total absolute and net trading volume on each token pair by $(v^{\text{abs}}_{j,k}) =: \mathbf{v}^{\text{abs}} \in \mathbb{R}^{|\mathcal{I}^p|}_{\geq 0}$ and $(v^{\text{net}}_{j,k}) =: \mathbf{v}^{\text{net}} \in \mathbb{R}^{|\mathcal{I}^p|}_{\geq 0}$.

If the exchange rate on a token pair (j,k) falls into region r_l, the following set of constraints needs to hold:

$$\tilde{\mathfrak{C}}_{j,k}^{(l)} := \left\{ \begin{array}{rcl} v_{j,k,l}^{A} & \geq & 0 \\ v_{j,k,l}^{A} & \leq & \tilde{x}_{j,k,l}^{(1)}\, p_j + \tilde{y}_{j,k,l}^{(2)}\, p_k \\ v_{j,k,l}^{B} & \geq & 0 \\ v_{j,k,l}^{B} & \leq & \tilde{y}_{j,k,l}^{(1)}\, p_j + \tilde{x}_{j,k,l}^{(2)}\, p_k \\ p_j & \geq & \pi_{j,k}^{(l)}\, p_k \\ p_j & \leq & \pi_{j,k}^{(l+1)} p_k \end{array} \right\} \tag{21}$$

On a token pair (j,k) with m regions r_l, the disjunction $\bigvee_{l\in\{0\ldots m\}} \tilde{\mathfrak{C}}_{j,k}^{(l)}$ needs to be satisfied.

Model Similarly as described in Section 3.2 for the first MIP model, (21) can be reformulated using either a big-M or a disjunctive programming approach. Without going into detail about this, the overall model reads as in (22).

The objective function (22a) maximizes the sum of the trading volumes over all trading pairs and is supposed to yield the same value as the objective (16a) of the previous MIP model.

The first constraint (22b) secures, again, the token balance for every token by requiring the net traded volumes to be balanced over all token pairs in which the respective token is involved.

$$\text{maximize} \quad \sum_{(j,k)\in\mathcal{I}^p} v_{j,k}^{\text{abs}} \tag{22a}$$

$$\text{subject to} \quad \sum_{\substack{k\in\mathcal{I}^t \\ k\neq j}} v_{k,j}^{\text{net}} = \sum_{\substack{k\in\mathcal{I}^t \\ k\neq j}} v_{j,k}^{\text{net}} \qquad \forall\, j \in \mathcal{I}^t \tag{22b}$$

$$v_{j,k}^{\text{abs}} = \sum_{l\in\tilde{\mathcal{I}}_{j,k}^r} \left(v_{j,k,l}^A + v_{j,k,l}^B\right) \qquad \forall\, (j,k) \in \mathcal{I}^p \tag{22c}$$

$$v_{j,k}^{\text{net}} = \sum_{l\in\tilde{\mathcal{I}}_{j,k}^r} \left(v_{j,k,l}^A - v_{j,k,l}^B\right) \qquad \forall\, (j,k) \in \mathcal{I}^p \tag{22d}$$

$$\bigvee_{l\in\{0\ldots m\}} \tilde{\mathfrak{C}}_{j,k}^{(l)} \qquad \forall\, (j,k) \in \mathcal{I}^p \tag{22e}$$

$$\sum_{j\in\mathcal{I}^t} \gamma_j \cdot p_j = 1 \tag{22f}$$

$$p_j \in [\,\underline{p_j}, \overline{p_j}\,] \qquad \forall\, j \in \mathcal{I}^t \tag{22g}$$

$$v_{j,k}^{\text{abs}}, v_{j,k}^{\text{net}} \in \mathbb{R}_{\geq 0} \qquad \forall\, (j,k) \in \mathcal{I}^p \tag{22h}$$

$$v_{j,k,l}^A, v_{j,k,l}^B \in \mathbb{R}_{\geq 0} \qquad \forall\, (j,k) \in \mathcal{I}^p, l \in \tilde{\mathcal{I}}_{j,k}^r \tag{22i}$$

3.4 Computational comparison

In order to investigate the performance of our MIP models, we have conducted computational experiments for different numbers of tokens ($n \in \{5, 10, 20, 50\}$) and orders ($N \in \{100, 200, 500\}$). For every combination of n and N, we have generated 20 random instances, whereby the randomness reflects our expectation of somewhat realistic situations. In particular, we expect not all tokens to be equally important in terms of trading volume and, hence, to have varying numbers of orders on different token pairs. We used Gurobi 8.0.0 as MIP solver on an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz machine with 16Gb RAM and using 4 threads, and a timelimit set to 1800 seconds for every instance.

The value of the maximum fluctuation parameter has been set to $\delta = 0.1$, meaning that prices are allowed to change by at most 10% from the (randomly set) previous prices. Moreover, we used $r^{\min} = 0$, hence orders can be left untouched even if their limit price complies with the computed exchange rate. This guarantees that all our test instances are feasible.

The (geometric) means of the runtimes have been computed only with respect to the instances that could be solved to optimality before the timelimit. Conversely, the average optimality gap does not take solved instances into account.

In general, the results of our computational experiments as in Table 1 and Table 2 show that runtimes of the two MIP formulations sharply increase both with the number of tokens and the number of orders that are being considered. While all instances with no more than 200 orders could be solved relatively fast by all models, instances with 500 orders are much harder to solve, with some (or even most) of the instances running into our set timelimit. The MIP model I (16) generally performs better on our testsets. Interestingly, for this model the big-M reformulation seems to be superior, whereas for the MIP model II, running times are lower using the Disjunctive Programming reformulation.

Without further model improvements, instances with 50 tokens and 500 orders already seem to impose a limit to the tractable problem size. While this limit may be pushed towards somewhat bigger instances by making use of additional valid inequalities such as (17)–(19), it appears unlikely that the tractable problem size can be increased by some orders of magnitude. If larger problems are to be considered, we can think of the following heuristics/approximations:

- Aggregate orders with similar limit prices on every asset pair
- Optimize over subsets of assets separately and fix prices in overall problem

# orders		# assets 5	10	20	50
100	∅ runtime	0.15	0.20	0.24	0.50
	# timeouts	0	0	0	0
	– ∅ gap	-	-	-	-
200	∅ runtime	0.42	1.78	2.26	6.67
	# timeouts	0	0	0	0
	– ∅ gap	-	-	-	-
500	∅ runtime	3.46	158.66	190.70	896.63
	# timeouts	0	0	0	7
	– ∅ gap	-	-	-	4.23%

(a) Big-M reformulation (33).

# orders		# assets 5	10	20	50
100	∅ runtime	0.25	0.38	0.48	1.09
	# timeouts	0	0	0	0
	– ∅ gap	-	-	-	-
200	∅ runtime	0.92	3.34	4.08	12.28
	# timeouts	0	0	0	0
	– ∅ gap	-	-	-	-
500	∅ runtime	6.96	184.60	229.98	1398.84
	# timeouts	0	1	2	17
	– ∅ gap	-	3.25%	2.27%	4.85%

(b) Disjunctive programming reformulation (15).

Table 1: Computational results for MIP model I (16).

# orders		# assets 5	10	20	50
100	∅ runtime	0.44	0.65	0.54	0.57
	# timeouts	0	0	0	0
	– ∅ gap	-	-	-	-
200	∅ runtime	4.14	18.40	14.27	20.18
	# timeouts	0	0	0	0
	– ∅ gap	-	-	-	-
500	∅ runtime	647.98	646.57	857.77	-
	# timeouts	0	19	19	20
	– ∅ gap	-	8.19%	6.85%	8.12%

(a) Big-M reformulation.

# orders		# assets 5	10	20	50
100	∅ runtime	0.34	0.85	1.12	1.14
	# timeouts	0	0	0	0
	– ∅ gap	-	-	-	-
200	∅ runtime	1.18	8.81	10.64	25.50
	# timeouts	0	0	0	0
	– ∅ gap	-	-	-	-
500	∅ runtime	24.70	288.50	315.62	-
	# timeouts	0	3	6	20
	– ∅ gap	-	2.31%	2.66%	5.39%

(b) Disjunctive programming reformulation.

Table 2: Computational results for MIP model II (22).

4 Extensions

4.1 Accounts and balances

...

4.2 Fee model

While the proposed trading mechanism guarantees fair prices and preserves value for all market participants, it does not yet consider trading fees to be paid towards the market operator. In most continuous orderbook-based exchanges, a trading fee is simply levied as a percentage of the trade that is executed. The approach described in the following adopts this common practice. Moreover, it allows for fees to be paid out in a designated *fee token* only and thus avoids for the fees to be denominated in every possible token that is present in the batch iteration.

Let $\varphi \in [0, 1]$ be the fee percentage for every trade and $\tau^{(\varphi)}$ the fee token. For a single order $\omega = (j, k, \overline{x}, \overline{y}, \pi)$, we had defined the constraint set (9) in Section 3 describing the required restrictions depending on whether ω is executed or not. One of our goals is to satisfy the limit price of an order even after the deduction of fees, i.e., the constraint $y \leq \pi \cdot x$ still needs to hold. We interpret the fee as a deduction from the buy amount x (as opposed to a surcharge on top of the sell amount y), so the uniform clearing price condition changes to $x = y \cdot \frac{p_k}{p_j} \cdot (1 - \varphi)$.

Altogether, this yields the following modified constraint sets:

$$\mathfrak{C}^1_{(\varphi)} := \left\{ \begin{array}{rcl} x & \leq & \overline{x} \\ y & \leq & \overline{y} \\ x \cdot p_j & = & y \cdot p_k \cdot (1 - \varphi) \\ \frac{p_j}{p_k} & \leq & \pi \cdot (1 - \varphi) \end{array} \right\} \quad \text{and} \quad \mathfrak{C}^0_{(\varphi)} := \left\{ \begin{array}{rcl} x & = & 0 \\ y & = & 0 \\ \frac{p_j}{p_k} & > & \pi \cdot (1 - \varphi) \end{array} \right\} \tag{23}$$

The token balance constraint (12b) as in the nonlinear programming formulation remains similar for all tokens except the fee token $\tau^{(\varphi)}$, for which it is relaxed to

$$\sum_{i \in \mathcal{I}^o} t^b_{i,j}\, x_i \leq \sum_{i \in \mathcal{I}^o} t^s_{i,j}\, y_i \qquad\qquad \tau_j = \tau^{(\varphi)}. \tag{12b'}$$

This allows for more tokens $\tau^{(\varphi)}$ to be sold than bought across all executed orders. The difference represents the total fee amount that is paid towards the operator.

In the mixed-integer model I, the constraint set (13) needs to be modified as follows in order to accomodate the fee model:

$$\tilde{\mathfrak{C}}^1_{(\varphi)} := \left\{ \begin{array}{rcl} (1-\varphi)\cdot v & \leq & \overline{x}\cdot p_j \\ v & \leq & \overline{y}\cdot p_k \\ \frac{p_j}{p_k} & \leq & \pi\cdot(1-\varphi) \end{array} \right\} \quad \text{and} \quad \tilde{\mathfrak{C}}^0_{(\varphi)} := \left\{ \begin{array}{rcl} v & = & 0 \\ \frac{p_j}{p_k} & > & \pi\cdot(1-\varphi) \end{array} \right\} \tag{24}$$

For this model, we had shown that token balance is equivalent to volume balance for every token. Considering fees, the respective constraint (16b) changes to

$$\begin{aligned} \sum_{i\in\mathcal{I}^o} t^b_{i,j}\, v_i \cdot (1-\varphi) &= \sum_{i\in\mathcal{I}^o} t^s_{i,j}\, v_i && \forall\, \tau_j \in \mathcal{T}\setminus\tau^{(\varphi)} \\ \sum_{i\in\mathcal{I}^o} t^b_{i,j}\, v_i \cdot (1-\varphi) &\leq \sum_{i\in\mathcal{I}^o} t^s_{i,j}\, v_i && \tau_j = \tau^{(\varphi)}. \end{aligned} \tag{16b'}$$

Please note that it is an easy extension to consider multiple fee tokens instead of only one.

Example 4.1. *Let us consider a set of three tokens $\mathcal{T} = \{\tau_1, \tau_2, \tau_3\}$ with $\tau^{(\phi)} = \tau_1$, and limit sell orders $\mathcal{O} = \{\omega_1, \omega_2, \omega_3\}$ with*

$$\begin{aligned} \omega_1 &= (3, 1, +\infty, 10, 1/19.8) \\ \omega_2 &= (2, 3, +\infty, 200, 200/0.99) \\ \omega_3 &= (1, 2, +\infty, 1, 1/9.9) \end{aligned}$$

a) $\varphi = 0$: without a fee, the prices $p_1 = 20$, $p_2 = 1$, and $p_3 = 200$ allow for the following execution of the orders:

$$\begin{aligned} (x_1, y_1) &= (200, 10), \\ (x_2, y_2) &= (1, 200), \\ (x_3, y_3) &= (10, 1). \end{aligned}$$

b) $\varphi = 0.01$: with a 1% fee, the same prices as above lead to

$$\begin{aligned} (x_1', y_1') &= (198, 10), \\ (x_2', y_2') &= (0.9801, 198), \\ (x_3', y_3') &= (9.70299, 0.9801). \end{aligned}$$

Hence, a total amount of $(10 - 9.70299) = 0.29701$ tokens τ_1 is spent as fees and the individual exchange rate that the traders receive for any order is slightly lower as in a).

As a short note on the impact of fees on the trader's utility, it can be stated that the definitions (6a) and (6b) remain unaffected. However, the reformulations via the uniform clearing price condition change to

$$u_A = x \cdot (\pi \cdot p_k - \frac{p_j}{1-\varphi})),$$

$$u_B = y \cdot ((1-\varphi) \cdot p_k - \frac{p_j}{\pi}).$$

These can be combined with the α-adjustment as in (6a') and (6b') to always favor trade execution over no trade in the case that $\frac{p_j}{p_k} = \pi \cdot (1-\varphi)$. For example, with $\alpha^* = \frac{\varphi}{2 \cdot (1-\varphi)}$, we obtain

$$u_A^{(\alpha^*)} = x \cdot (\pi \cdot p_k - \frac{p_j}{1-\frac{\varphi}{2}})),$$

$$u_B^{(\alpha^*)} = y \cdot ((1-\frac{\varphi}{2}) \cdot p_k - \frac{p_j}{\pi}).$$

The same can be applied to disregarded utility as objective criterion by modifying the equations (8a) and (8b) accordingly.

4.3 Basket orders

In some cases, traders might not only want to exchange one token against one other token, but instead trade collections of tokens directly against each other. As a typical example, a trader could want to buy a certain amount of different tokens for a maximum total price, without being interested in the particular price of every individual token. We refer to this order type as *basket order*.

For being able to use basket orders in our models, we first need to generalize the order definition. A basket of tokens shall be represented by a vector $\boldsymbol{\beta} \in \mathbb{R}^n_{\geq 0}$, where $\beta_j \in \boldsymbol{\beta}$ specifies the amount of token τ_j in the basket. In other words, a basket of tokens can be understood as a list of tuples $(\beta_j, \tau_j)_{i=j}^n$.

Similar as in Section 2, a basket order shall be defined as a tuple $\omega = (\boldsymbol{\beta}^b, \boldsymbol{\beta}^s, \overline{x}, \overline{y}, \pi)$, with the meaning of the elements being as follows:

$\boldsymbol{\beta}^b \in \mathbb{R}^n_{\geq 0}$	...	$(\beta^b_j, \tau_j)_{j=1}^n$ is the basket of tokens to be bought,
$\boldsymbol{\beta}^s \in \mathbb{R}^n_{\geq 0}$	...	$(\beta^s_j, \tau_j)_{j=1}^n$ is the basket of tokens to be sold,
$\overline{x} \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$	...	maximum units of basket $(\beta^b_j, \tau_j)_{j=1}^n$ to be bought,
$\overline{y} \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$	...	maximum units of basket $(\beta^s_j, \tau_j)_{j=1}^n$ to be sold,
$\pi \in \mathbb{R}_{>0} \cup \{+\infty\}$	...	limit exchange rate for order execution, i.e., $p_{\boldsymbol{\beta}^b \mid \boldsymbol{\beta}^s} \leq \pi$.

With the same arguments as in Section 3, the exchange rate $p_{\boldsymbol{\beta}^b \mid \boldsymbol{\beta}^s}$ between the two baskets of tokens can be written as the quotient of two prices with respect to the abstract reference token τ_0, i.e.,

$$p_{\boldsymbol{\beta}^b \mid \boldsymbol{\beta}^s} = p_{\boldsymbol{\beta}^b \mid 0} \cdot p_{0 \mid \boldsymbol{\beta}^s} = \frac{p_{\boldsymbol{\beta}^b \mid 0}}{p_{\boldsymbol{\beta}^s \mid 0}} =: \frac{p_{\boldsymbol{\beta}^b}}{p_{\boldsymbol{\beta}^s}}. \tag{2'}$$

Hence, $p_{\boldsymbol{\beta}^b}$ and $p_{\boldsymbol{\beta}^s}$ denote the price of one unit of the buy- or sell-basket, respectively, measured in units of τ_0.

Moreover, the price $p_{\boldsymbol{\beta}}$ of a basket of tokens simply equals the sum of the individual token prices p_j, $j = 1 \dots n$, weighted by the elements in $\boldsymbol{\beta}$, i.e.,

$$p_{\boldsymbol{\beta}} = \sum_{j=1}^{n} \beta_j \, p_j. \tag{27}$$

With this and using the same variables $x \geq 0$ and $y \geq 0$, we can rephrase the generic order

constraints (9) to

$$\mathfrak{C}^1_\beta := \left\{ \begin{array}{rcl} x & \leq & \overline{x} \\ y & \leq & \overline{y} \\ x \cdot p_{\boldsymbol{\beta^b}} & = & y \cdot p_{\boldsymbol{\beta^s}} \\ \frac{p_{\boldsymbol{\beta^b}}}{p_{\boldsymbol{\beta^s}}} & \leq & \pi \end{array} \right\} \quad \text{and} \quad \mathfrak{C}^0_\beta := \left\{ \begin{array}{rcl} x & = & 0 \\ y & = & 0 \\ \frac{p_{\boldsymbol{\beta^b}}}{p_{\boldsymbol{\beta^s}}} & > & \pi \end{array} \right\}. \tag{9'}$$

In the context of an entire batch auction, we store the token weights $\boldsymbol{\beta^b}$ and $\boldsymbol{\beta^s}$ of every in order in matrices $\mathbf{B}^b := (\boldsymbol{\beta}^{\boldsymbol{b}}_i) = (\beta^b_{i,j}) \in \mathbb{R}^{N\times n}_{\geq 0}$ and $\mathbf{B}^s := (\boldsymbol{\beta}^{\boldsymbol{s}}_i) = (\beta^s_{i,j}) \in \mathbb{R}^{N\times n}_{\geq 0}$.

This now allows us to treat basket orders in the nonlinear model (12) as any other orders, with only redefining the two data matrices $\mathbf{T}^b := \mathbf{B}^b$ and $\mathbf{T}^s := \mathbf{B}^s$.

Unfortunately, we currently do not see a straighforward way to use basket orders in combination with the mixed-integer model I (16), which is based on volume variables $v \geq 0$. The trading volume for a single order would need to be defined as

$$v = x \cdot p_{\boldsymbol{\beta}^b} = x \cdot \sum_{j=1}^{n} \beta^b_j \, p_j \qquad \text{and} \qquad v = y \cdot p_{\boldsymbol{\beta}^s} = x \cdot \sum_{j=1}^{n} \beta^s_j \, p_j \tag{??'}$$

Then, the constraint sets (13) translate to

$$\tilde{\mathfrak{C}}^1_\beta := \left\{ \begin{array}{rcl} v & \leq & \overline{x} \cdot p_{\boldsymbol{\beta^b}} \\ v & \leq & \overline{y} \cdot p_{\boldsymbol{\beta^s}} \\ \frac{p_{\boldsymbol{\beta^b}}}{p_{\boldsymbol{\beta^s}}} & \leq & \pi \end{array} \right\} \quad \text{and} \quad \tilde{\mathfrak{C}}^0_\beta := \left\{ \begin{array}{rcl} v & = & 0 \\ \frac{p_{\boldsymbol{\beta^b}}}{p_{\boldsymbol{\beta^s}}} & > & \pi \end{array} \right\}. \tag{28}$$

The problem arises, however, ...

4.4 Functional order amounts

In Section 2.1, we have defined an order as a tuple $\omega = (j, k, \overline{x}, \overline{y}, \pi)$, where the maximum buy and sell amounts $\overline{x}, \overline{y} \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$ are fixed given constants. However, there might be situations in which the trader is willing to buy or sell less or more units of some token depending on how favorable the token prices are. In this section, as an example, we will investigate a linear dependency between the maximum sell amount of an order and the exchange rate $p_{j|k} = p_j/p_k$ between the buy-token τ_j and the sell-token τ_k:

$$\overline{y}(p_j, p_k) := a \cdot \frac{p_j}{p_k} + b, \tag{29}$$

where $a, b \in \mathbb{R}$ are constants. In order to be meaningful, we should require $a \leq 0$, so that fewer tokens are sold when the buy-token τ_j becomes more expensive, and $a \cdot \pi + b \geq 0$, so that a non-negative amount of tokens τ_k can be sold when the limit price is satisfied. Notice that a constant maximum sell amount is a special case with $a = 0$ and $b = \overline{y}$.

Using a volume variable $v = y \cdot p_k$ as in our MIP model (16), we can reformulate the above as

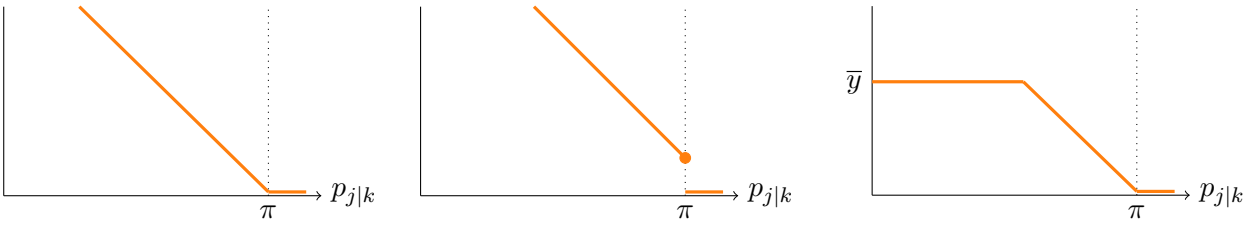
$$\overline{v}(p_j, p_k) := \overline{y}(p_j, p_k) \cdot p_k = a \cdot p_j + b \cdot p_k, \tag{30}$$

which describes a hyperplane in the space (p_j, p_k, v) of token prices and trading volume. As a result, the constraint set for a limit sell order changes to

$$\tilde{\mathfrak{C}}^1 := \left\{ \begin{array}{rcl} v & \leq & a \cdot p_j + b \cdot p_k \\ \frac{p_j}{p_k} & \leq & \pi \end{array} \right\} \quad \text{and} \quad \tilde{\mathfrak{C}}^0 := \left\{ \begin{array}{rcl} v & = & 0 \\ \frac{p_j}{p_k} & > & \pi \end{array} \right\} \tag{31}$$

Notice that we can even add a constant global maximum sell amount $\overline{y}$ to $\tilde{\mathfrak{C}}^1$ without complicating the MIP model.

With all this, a variety of new sell-profiles of a limit sell order can be expressed, e.g., :



(a) $a \cdot \pi + b = 0$ (b) $a \cdot \pi + b > 0$ (c) additional $\overline{y}$

Figure 3: Limit sell orders with functional maximum sell amounts.

4.5 Enforcing zero disregarded utility

From a perspective of mathematical economics, our batch auction exchange approach closely relates the so-called Arrow-Debreu Model of a competitive economy [1]. General equilibrium theory has shown that this model type always allows for a competitive equilibrium (or Walrasian Equilibrium as first introduced in [6]), given by a vector of prices and an allocation of goods for every agent, that is characterized as follows:

1. Each agent's allocation maximizes their personal utility function given the prices.

2. The market clears, i.e., for every good that is traded, the total allocation equals the aggregate endowment over all agents.

In Section 2.4, we have introduced the concept of disregarded utility as a potential objective function to be minimized. Applying the equilibrium theory to our context, we can even go one step further: a competitive equilibrium translates to a solution that does not contain any disregarded utility. In particular, this means that every order needs to be fully matched as soon as the computed prices for the tokens specified strictly satisfy the order's limit price. Only at exactly the limit price, i.e., at the point of indifference, an order may be partially settled. In this section, we want to investigate how we can enforce disregarded to be zero for every order in our MIP model (16), i.e., incorporate this requirement via linear constraints and additional variables. For simplicity, we will restrict our considerations to limit sell orders.

In general, requiring disregarded utility to be zero extends the order constraints (13) to a disjunction of three sets

$$\tilde{\mathfrak{C}}^0 := \left\{ \begin{array}{rcl} v & = & 0 \\ \frac{p_j}{p_k} & > & \pi \end{array} \right\} \vee \tilde{\mathfrak{C}}^{1,a} := \left\{ \begin{array}{rcl} v & \leq & \overline{y} \cdot p_k \\ \frac{p_j}{p_k} & = & \pi \end{array} \right\} \vee \tilde{\mathfrak{C}}^{1,b} := \left\{ \begin{array}{rcl} v & = & \overline{y} \cdot p_k \\ \frac{p_j}{p_k} & < & \pi \end{array} \right\}, \tag{32}$$

where $\tilde{\mathfrak{C}}^{1,a}$ and $\tilde{\mathfrak{C}}^{1,b}$ refer to the cases that prices are equal to or strictly satisfy the limit price of the order, respectively.

In order to reformulate (32) in terms of a mixed-integer linear program, we now need two binary variabels $z_1, z_2 \in \{0, 1\}$ for every order and associate their states to the constraint sets as $\{z_1 = z_2 = 0\} \Leftrightarrow \tilde{\mathfrak{C}}^0$, $\{z_1 = 1, z_2 = 0\} \Leftrightarrow \tilde{\mathfrak{C}}^{1,a}$, and $\{z_1 = z_2 = 1\} \Leftrightarrow \tilde{\mathfrak{C}}^{1,b}$. The

constraints can be modelled via a similar big-M approach as presented in (33):

$$p_j \geq \pi\, p_k + (\underline{p}_j - \pi\, \overline{p}_k)\, z_1 \tag{33a}$$

$$p_j \geq \pi\, p_k + (\underline{p}_j - \pi\, \overline{p}_k)\, z_2 \tag{33b}$$

$$p_j \leq \pi\, p_k + (\overline{p}_j - \pi\, \underline{p}_k)\, (1 - z_1) \tag{33c}$$

$$v \leq \overline{y}\, p_k \tag{33d}$$

$$v \leq \overline{y}\, \overline{p}_k\, z_1 \tag{33e}$$

$$v \geq \overline{y}\, (p_k - \overline{p}_k\, (1 - z_2)) \tag{33f}$$

$$z_2 \leq z_1 \tag{33g}$$

4.6 Incorporating constant-product liquidity pools

In addition to orders submitted by users, we want to introduce the concept of automated market makers (AMMs) to our exchange model. AMMs are passive market participants, standing ready to exchange tokens while acting on a predefined set of rules. By doing so, they provide liquidity to the market and enable trade even when there are no coinciding user orders available.

Uniswaps A very simple yet effective type of AMM is based on a constant-product formula on a pair of two tokens, used by the popular UNISWAP protocol.

Let there be a set $\mathcal{U} = \{\nu_1 \dots \nu_M\}$ of M Uniswap pools and let $\mathcal{I}^u := \{1 \dots M\}$ denote its set of indices. Every pool is specified as a tuple $\nu = (j, k, b_1, b_2, \Phi)$ whose elements mean the following:

$j \in \mathcal{I}^t$	...	τ_j is the first pool token
$k \in \mathcal{I}^t$	...	τ_k is the second pool token
$b_1 \in \mathbb{R}_{>0}$	...	current pool balance of τ_j
$b_2 \in \mathbb{R}_{>0}$	...	current pool balance of τ_k
$\Phi \in [0, 1]$	...	liquidity provider fee percentage

We refer to the ratio of the pool balances b_1/b_2 as the *marginal exchange rate* of the pool. The rule that specifies the behaviour of the pool simply keeps the product of the pool balances at a constant value κ on every trade (hence, κ is also referred to as the *invariant*):

$$b_1 \cdot b_2 = \kappa \tag{34}$$

This formula has the desirable property that larger trades (relative to reserves) will be executed at worse exchange rates than smaller ones. Moreover, liquidity pools may charge a relative fee Φ, taken from the token amount that is sold to the pool, which is added to the reserves. Thus, every usage of the pool actually increases the value of κ.

Let $x_1 \in \mathbb{R}$ and $x_2 \in \mathbb{R}$ denote the balance updates (i.e., trading amounts) of the pool tokens τ_j and τ_k, respectively. A positive update refers to the pool buying an amount of tokens, a negative update to selling. Pools can never sell all of their inventory, thus $x_1 \geq -b_1$ and $x_2 \geq -b_2$. The ratio of the balance updates x_1/x_2 is referred to as *effective exchange rate*. Ignoring fees for a start, (x_1, x_2) needs to follow the constant-product rule: $(b_1 + x_1) \cdot (b_2 + x_2) = \kappa$. This immediately implies $x_1 \cdot x_2 \leq 0$, i.e., one of the tokens always needs to be bought when the other one is sold. With fees, the feasible behaviour of

a Uniswap pool can be described by the union of the following constraint sets:

$$\mathfrak{C}^0 := \left\{ \begin{array}{c} x_1 = 0 \\ x_2 = 0 \end{array} \right\} \qquad \ldots \quad \text{no trade} \tag{35a}$$

$$\mathfrak{C}^1 := \left\{ \begin{array}{c} x_1 > 0 \\ -b_2 < x_2 < 0 \\ (b_1 + (1-\Phi)\, x_1) \cdot (b_2 + x_2) = \kappa \end{array} \right\} \qquad \ldots \quad \text{buy } \tau_j\text{, sell } \tau_k \tag{35b}$$

$$\mathfrak{C}^2 := \left\{ \begin{array}{c} -b_1 < x_1 < 0 \\ x_2 > 0 \\ (b_1 + x_1) \cdot (b_2 + (1-\Phi)\, x_2) = \kappa \end{array} \right\} \qquad \ldots \quad \text{sell } \tau_j\text{, buy } \tau_k \tag{35c}$$

This disjunction as well as the last equations in both $\mathfrak{C}^1$ and $\mathfrak{C}^2$ are nonlinear. Our goal in the following will be to derive a MIP formulation that we can use in combination with the model for user (limit) orders.

First of all, we introduce two binary variables $z_1, z_2 \in [0, 1]$ for controlling which of the constraint sets is active:

$$\begin{aligned} (z_1 = z_2 = 0) &\;\Rightarrow\; \mathfrak{C}^0 \\ (z_1 = 1) &\;\Rightarrow\; \mathfrak{C}^1 \\ (z_2 = 1) &\;\Rightarrow\; \mathfrak{C}^2 \\ z_1 + z_2 &\leq 1 \end{aligned}$$

Furthermore, we will work with volume variables instead of token amounts directly. For a Uniswap pool $\nu = (j, k, b_1, b_2, \Phi)$, let $v_1^b, v_1^s, v_2^b, v_2^s \in \mathbb{R}_{\geq 0}$ denote the buy- and sell-volumes of token τ_j and τ_k, respectively. With $x_i^+ = \max\{x_i, 0\}$ and $x_i^- = -\min\{x_i, 0\}$ for $i \in \{1, 2\}$, these are defined as $v_1^b := x_1^+ p_j$, $v_1^s := x_1^- p_j$, $v_2^b := x_2^+ p_k$, $v_2^s := x_2^- p_k$.

Similar to regular orders, we also want the balance updates of Uniswap pools to follow our uniform clearing prices, i.e., $x_1 \cdot p_j = -x_2 \cdot p_k$. This implies $v_1^b = v_2^s$ and $v_1^s = v_2^b$.

With all this, we are now going to transform the nonlinear constant-product equation in $\mathfrak{C}^1$

into a linear equivalent (analogous for $\mathfrak{C}^2$):

$$
\begin{aligned}
& & \kappa &= (b_1 + (1-\Phi)x_1) \cdot (b_2 + x_2) \\
&\Leftrightarrow & \kappa &= \underbrace{b_1 b_2}_{\kappa} + x_2 b_1 + (1-\Phi) x_1 b_2 + (1-\Phi) x_1 x_2 \\
&\Leftrightarrow & 0 &= x_2 b_1 + (1-\Phi) x_1 b_2 + (1-\Phi) x_1 x_2 \\
\left[x_1 = -x2\frac{p_k}{p_j}\right] &\Leftrightarrow & 0 &= x_2 b_1 - (1-\Phi) x_2 b_2 \frac{p_k}{p_j} - (1-\Phi) x_2^2 \frac{p_k}{p_j} \\
[\cdot p_j] &\Leftrightarrow & 0 &= x_2 b_1 p_j - (1-\Phi) x_2 b_2 p_k - (1-\Phi) x_2^2 p_k \\
[: x_2 \neq 0] &\Leftrightarrow & 0 &= b_1 p_j - (1-\Phi) b_2 p_k - (1-\Phi) \underbrace{x_2 p_k}_{-v_2^s} \\
&\Leftrightarrow & (1-\Phi) v_2^s &= (1-\Phi) b_2 p_k - b_1 p_j
\end{aligned}
$$

This can now be used to rewrite the constraint sets (35):

$$
\tilde{\mathfrak{C}}^0 := \left\{ \begin{array}{c} v_1^b = v_1^s = 0 \\ v_2^b = v_2^s = 0 \end{array} \right\} \qquad \ldots \text{ no trade} \qquad \text{(35a')}
$$

$$
\tilde{\mathfrak{C}}^1 := \left\{ \begin{array}{c} v_1^b = v_2^s > 0 \\ v_1^s = v_2^b = 0 \\ (1-\Phi) v_2^s = (1-\Phi) b_2 p_k - b_1 p_j \end{array} \right\} \qquad \ldots \text{ buy } \tau_j, \text{ sell } \tau_k \qquad \text{(35b')}
$$

$$
\tilde{\mathfrak{C}}^2 := \left\{ \begin{array}{c} v_1^b = v_2^s = 0 \\ v_1^s = v_2^b > 0 \\ (1-\Phi) v_1^s = (1-\Phi) b_1 p_j - b_2 p_k \end{array} \right\} \qquad \ldots \text{ sell } \tau_j, \text{ buy } \tau_k \qquad \text{(35c')}
$$

In order to handle the disjunction $\tilde{\mathfrak{C}}^0 \cup \tilde{\mathfrak{C}}^1 \cup \tilde{\mathfrak{C}}^2$ in a MIP model, we can again resort to standard big-M modelling techniques as in previous sections.

As mentioned, the current model for Uniswap pools ensures that the balance updates comply with the computed uniform clearing prices whenever the pool is used. However, interacting with a pool is not mandatory, i.e., it does not have to be used even when the computed exchange rate on the respective token pair deviates from the ratio of the pool balances. This behaviour can be controlled by adding another constraint to the model. Let $\Psi \in [0, 1]$ define a margin around the current marginal exchange rate of the pool in which it does not need to be used and outside of which it becomes mandatory. This can be translated into the

following implications (which in turn can be expressed using big-M constraints):

$$\left(\frac{p_j}{p_k} < \frac{b_2}{b_1} \cdot (1-\Psi)\right) \;\Rightarrow\; (z_1 = 1) \quad \Leftrightarrow \quad (z_1 = 0) \;\Rightarrow\; (b_1 p_j \geq b_2 p_k (1-\Psi)) \tag{37a}$$

$$\left(\frac{p_k}{p_j} < \frac{b_1}{b_2} \cdot (1-\Psi)\right) \;\Rightarrow\; (z_2 = 1) \quad \Leftrightarrow \quad (z_2 = 0) \;\Rightarrow\; (b_2 p_k \geq b_1 p_j (1-\Psi)) \tag{37b}$$

Balancer The idea of constant-product liquidity pools can be generalized to $m \geq 2$ tokens. Let $\mathbf{j} := (j_1, \ldots, j_m)$ refer to the tokens $(\tau_{j_1}, \ldots, \tau_{j_m})$, and let $\mathbf{b} := (b_1, \ldots, b_m)$ denote their respective pool balances. Moreover, we introduce a weight vector $\boldsymbol{\alpha} := (\alpha_1, \ldots, \alpha_m)$ with $\sum_{i=1}^{m} \alpha_i = 1$. Then, the constant-product formula governing the behaviour of the pool becomes

$$\prod_{i=1}^{m} b_i^{\alpha_i} = \kappa \tag{38}$$

4.7 Further ideas

The problem can possibly be extended in various directions:

- Add accounts/balances to the model.
- Minimum execution fraction: May lead to infeasibility.
- Basket orders: Buy/sell a set of tokens for a set of other tokens at some limit price.
- Automated market makers: Instead of signaling discrete demand at a specific price with one order, those would allow to express continues demand function over a price range.
- if it becomes a problem to find a valid solution *at all*, the optimization problem can be broadened to allow violations of the current constraints but measure the violation and include this to the objective function.

References

[1] Kenneth J. Arrow and Gerard Debreu. Existence of an equilibrium for a competitive economy. *Econometrica*, 22(3):265–290, 1954.

[2] Egon Balas. Disjunctive Programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

[3] Pierre Bonami, Andrea Lodi, Andrea Tramontani, and Sven Wiese. On mathematical programming with indicator constraints. *Mathematical Programming*, 151(1):191–223, 2015.

[4] Eric Budish, Peter Cramton, and John Shim. The High-Frequency Trading Arms Race: Frequent Batch Auctions as a Market Design Response. *The Quarterly Journal of Economics*, 130(4):1547–1621, 2015.

[5] Richard Engelbrecht-Wiggans and Charles M. Kahn. Multi-Unit Auctions with Uniform Prices. *Economic Theory*, 12(2):227–258, 1998.

[6] Léon Walras. *Éléments d'économie politique pure, ou, Théorie de la richesse sociale*. F. Rouge, 1896.