Multi-token Batch Auctions with Uniform Clearing Prices

–

Features and Models

Tom Walther
tom@gnosis.pm

July 6, 2018

... abstract ... [1].

- write abstract
- change t for tokens to τ everywhere
- change o for orders to ω everywhere
- add additional inequalities for MIP1
- write down MIP3
- write something about min/max fluctuation
- computational results for MIP{2—3}
- choice of reference token example
- pictures of order-token-graph

[1] ... footnote ...

Contents

1 Introduction **3**

2 Problem statement **4**
- 2.1 Data . . . 4
- 2.2 Objectives . . . 5
- 2.3 Constraints . . . 5
- 2.4 Properties . . . 6

3 Models and Formulations **7**
- 3.1 Nonlinear programming model . . . 8
- 3.2 Mixed-integer linear programming model I . . . 10
- 3.3 Mixed-integer linear programming model II . . . 13
- 3.4 Mixed-integer linear programming model III . . . 16
- 3.5 Min-cost flow model . . . 16
- 3.6 Computational comparison . . . 17

4 Extensions **19**

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

$$p_{A|B} \cdot p_{B|C} = p_{A|C} \tag{1}$$

to be satisfied for the prices of all tokens A,B,C.

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, as well as present a mixed-integer programming solution approach.

2 Problem statement

2.1 Data

Let $\mathcal{T} := \{\tau_0 \dots \tau_{n-1}\}$ denote the set of the n tokens that we want to consider. The token τ_0 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 $\mathcal{I}^t := \{0 \dots n-1\}$ to denote the indices of our token set.

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

Let there be a set $\mathcal{O} = \{\omega_1 \dots \omega_N\}$ of N limit orders in the batch to be processed and let $\mathcal{I}^b, \mathcal{I}^s \subseteq \{1 \dots N\} =: \mathcal{I}^o$ denote the sets of indices of buy and sell orders, respectively. Every order must be of exactly one of the two types, so $\mathcal{I}^b \cap \mathcal{I}^s = \emptyset$ and $\mathcal{I}^b \cup \mathcal{I}^s = \{1 \dots N\}$.

Moreover, every buy order consists of a tuple $(\tau_{j_b}, \tau_{j_s}, \overline{x}, \pi)$ that is to be read as

> *"Buy (at most) $\overline{x}$ units of token τ_{j_b} for token τ_{j_s} if the rate $p_{j_b|j_s}$ is at most π".*

Analogously, every sell order is represented by a tuple $(\tau_{j_s}, \tau_{j_b}, \overline{y}, \pi)$ with the meaning

> *"Sell (at most) $\overline{y}$ units of token τ_{j_s} for token τ_{j_b} if the rate $p_{j_s|j_b}$ is at least π".*

The semantics of the two order types are somewhat similar in that there is one token to be exchanged against another, if a certain limit price is guaranteed. Such orders are hence referred to as *limit orders*. The difference between buy and sell orders is the following:

- In a buy order, a fixed (maximum) buy volume $\overline{x}$ is set, so the buyer indicates a preference to buy $\overline{x}$ units of token τ_{j_b}. The amount of units of token τ_{j_s} to be sold is flexible, as long as the limit price is respected (the smaller, the better).
- In a sell order, a fixed (maximum) sell volume $\overline{y}$ is set, so the seller indicates a preference to sell $\overline{y}$ units of token τ_{j_s}. The amount of units of token τ_{j_b} to be bought is flexible, as long as the limit price is respected (the higher, the better).

Nevertheless, the question remains whether it makes sense to keep the distinction between buy and sell orders. Consider a buy order $(\tau_{j_b}, \tau_{j_s}, \overline{x}, \pi)$, which reveals that the buyer would be ready to sell at most $\pi \cdot \overline{x}$ tokens τ_{j_s} in exchange for tokens τ_{j_b}. Then, from an economically rational point of view, the buyer should not object to receiving more than $\overline{x}$ tokens τ_{j_b} for a maximum of $\pi \cdot \overline{x}$ tokens τ_{j_s}, i.e., if the exchange rate $p_{j_b|j_s}$ is lower than π. Hence, the buy order could be translated into an equivalent sell order $(\tau_{j_s}, \tau_{j_b}, \pi \cdot \overline{x}, \pi)$. In this paper, however, we will continue to treat the different order types separately.

2.2 Objectives

Having collected a batch of buy and sell orders for our set of tokens, we ultimately want to compute exchange rates for all token pairs. Therefore, the following optimization criteria could be used:

- maximize the amount of trade that is enabled (w.r.t. some reference token)
- maximize *trader welfare*, e.g., defined as difference of paid price vs. limit price

2.3 Constraints

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

- *(buy limit price)*:
  for all buy orders $\omega_i \in \mathcal{O}$, $i \in \mathcal{I}^b$: the order can only be executed (fully or fractionally) if the exchange rate does not exceed the stated limit price, i.e., $p_{b|s} \leq \pi$.
- *(sell limit price)*:
  for all sell orders $\omega_i \in \mathcal{O}$, $i \in \mathcal{I}^s$: the order can only be executed (fully or fractionally) if the exchange rate is not lower than the stated limit price, i.e., $p_{s|b} \geq \pi$.
- *(token balance)*:
  for every token $\tau_j \in \mathcal{T}$: the amount of tokens τ_j that were bought must equal the amount of tokens τ_j that were sold.
- *(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 2.1. *(arbitrage freeness) ⇒ (price coherence).*

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

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

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

Inserting (i) into (ii) 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 $\tau_j \in \mathcal{T}$.

2.4 Properties

(Before presenting different modelling approaches and formulations for the batch auction problem, we first want to analyse/show some important properties.)

Here is a list of research questions that we would like to answer.

- In an optimal solution, is it guaranteed that orders can always be fully executed if their limit prices are strictly higher (buy order) or lower (sell order) than the exchange rate between the respective token pair? If true, we would only need to resort to partially executing orders if the execution price is equal to the limit price.
- Does the optimal solution depend on the choice of the reference token, or not?
- What is the impact of optimizing the trading volume vs. the traders' welfare?

3 Models and Formulations

In this section, we want to present and discuss several solution approaches for our batch auction problem, mainly in terms of mathematical optimization formulations. We will begin with a nonlinear programming formulation (NLP) and proceed with mixed-integer linear (MIP) and network flow formulations thereafter.

With the exchange rates between pairs of tokens being variables, modelling arbitrage-freeness constraints of type (1) intuitively leads to many multiplications being required. This may result in unnecessary limitations to the problem size that is tractable as well as numerical instability. However, we can do better by not considering all pairwise token rates explicitely but representing all token prices only with respect to a single reference token τ_0. Let $p_j := p_{j|0}$ denote the price of token τ_j expressed in units of token τ_0 (hence, $p_0 = 1$). Applying the arbitrage-freeness and price-coherence conditions directly, we can express the exchange rate between two tokens τ_j and τ_k as

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

Data

In the following, 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.

We introduce two data matrices

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

As for now, we are only considering orders of one token type against one other, so there must be exactly one entry equal to 1 per row (order) in both $\mathbf{T}^b$ and $\mathbf{T}^s$.

The maximum number of units of tokens to be bought in an order ω_i shall be denoted by $\overline{x}_i$ and stored, for all orders, in a vector $\overline{\mathbf{x}} \in \mathbb{R}^N_{\geq 0} \cup \{+\infty\}$. Generically, $\overline{x}_i$ takes a finite value if ω_i is a limit buy order (i.e., $i \in \mathcal{I}^b$). In case ω_i is a sell order (i.e., $i \in \mathcal{I}^s$), $\overline{x}_i$ can be set to infinity, since the seller does specify an upper limit on the tokens to be received.

Similarly, let $(\overline{y}_i) =: \overline{\mathbf{y}} \in \mathbb{R}^N_{\geq 0} \cup \{+\infty\}$ contain the maximum amounts of tokens to be sold in every order. For all limit sell orders ω_i, we have $\overline{y}_i < \infty$. On the other hand, if ω_i is a limit buy order, we can either set $\overline{y}_i = +\infty$ and rely on the implicit bound given via the limit price π_i (as defined below), or state it explicitely as $\overline{y}_i = \overline{x}_i \cdot \pi_i$.

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 2.1).

3.1 Nonlinear programming model

The first model that we developed involves nonlinear constraints through multiplication of token amounts and prices, but does not require binary variables.

Variables

We represent all token prices in a vector $\mathbf{p} \in \mathbb{R}^n_{\geq 0}$, i.e., p_j denotes the price of token τ_j w.r.t. the reference token τ_0. It makes sense to incorporate some explicit lower and upper bound for the price of every token τ_j (for instance, in order to avoid excessive fluctuations), so let us require $p_j \in [\underline{p}_j, \overline{p}_j]$. In our case, we use a maximum fluctuation parameter $\delta \geq 0$ to bound the deviation of the computed exchange rates from the previous ones. In particular, let p_j^{old} be the price of token τ_j found in the previous batch auction iteration. We then require $p_j \in [(\frac{1}{1+\delta})p_j^{\text{old}}, (1+\delta)\, p_j^{\text{old}}]$ for all tokens $\tau_j \neq \tau_0$. However, this only limits fluctuations on token pairs involving the reference token to the desired extent. In order to guarantee the same maximum fluctuation on all other token pairs (τ_j, τ_k), we further add the condition

$$p_{j|k} \in \left[\left(\frac{1}{1+\delta}\right) p_{j|k}^{\text{old}}, (1+\delta)\, p_{j|k}^{\text{old}}\right] \quad \Leftrightarrow \quad \frac{p_j}{p_k} \in \left[\left(\frac{1}{1+\delta}\right) \frac{p_j^{\text{old}}}{p_k^{\text{old}}}, (1+\delta)\, \frac{p_j^{\text{old}}}{p_k^{\text{old}}}\right]$$

$$\Leftrightarrow \quad \begin{cases} p_j \geq \left(\frac{1}{1+\delta}\right) \frac{p_j^{\text{old}}}{p_k^{\text{old}}}\, p_k \\ p_j \leq (1+\delta)\, \frac{p_j^{\text{old}}}{p_k^{\text{old}}}\, p_k \end{cases} .$$

As an example, $\delta = 1$ would set the bounds to half/double the previous exchange rates. If no previous price exists for some token τ_j, the price bounds $\underline{p}_j$ and $\overline{p}_j$ could be determined on the basis of the limit prices given in the orders that involve τ_j. In any case, we set $\underline{p}_0 = \overline{p}_0 = 1$ in order to fix the price of τ_0 to $p_0 = 1$. All price bounds shall be stored in vectors $\underline{\mathbf{p}}$ and $\overline{\mathbf{p}}$, respectively.

For the executed volumes of all orders, we define a vector $\mathbf{v} \in \mathbb{R}^N$, where $v_i \in \mathbf{v}$ contains the traded volume of order ω_i in terms of units of the reference token τ_0.

The number of tokens bought in an order ω_i shall be denoted by x_i, with $(x_i) =: \mathbf{x} \in \mathbb{R}^N_{\geq 0}$. Conversely, $(y_i) =: \mathbf{y} \in \mathbb{R}^N_{\geq 0}$ represents the amount of tokens sold per order.

Model

$$\text{maximize} \quad \sum_{i \in \mathcal{I}^o} v_i \tag{3a}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{I}^o} t^b_{i,j}\, x_i = \sum_{i \in \mathcal{I}^o} t^s_{i,j}\, y_i \qquad \forall\, j \in \mathcal{I}^t \tag{3b}$$

$$v_i = x_i \sum_{j \in \mathcal{I}^t} t^b_{i,j}\, p_j \qquad \forall\, i \in \mathcal{I}^o \tag{3c}$$

$$v_i = y_i \sum_{j \in \mathcal{I}^t} t^s_{i,j}\, p_j \qquad \forall\, i \in \mathcal{I}^o \tag{3d}$$

$$x_i \leq \overline{x}_i \qquad \forall\, i \in \mathcal{I}^b \tag{3e}$$

$$y_i \leq x_i\, \pi_i \qquad \forall\, i \in \mathcal{I}^b \tag{3f}$$

$$y_i \leq \overline{y}_i \qquad \forall\, i \in \mathcal{I}^s \tag{3g}$$

$$x_i \geq y_i\, \pi_i \qquad \forall\, i \in \mathcal{I}^s \tag{3h}$$

$$x_i, y_i, v_i \in \mathbb{R}_{\geq 0} \qquad \forall\, i \in \mathcal{I}^o \tag{3i}$$

The objective function (3a) maximizes the total volume in terms of units of the reference token τ_0 that is processed with all orders.

Constraint (3b) 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 (3c) and (3d) compute the buy and sell trade volume for every order w.r.t. 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.

Finally, the limits in terms of token amounts to be bought/sold in a limit order are incorporated into the model via the constraints (3e)–(3h).

One major weakness of the model is the fact that orders can be left unexecuted even if the

computed token prices satisfy the given limit price. We believe that this can only be cured with the introduction of binary variables that indicate whether prices allow for an order to be executed, or not.

3.2 Mixed-integer linear programming model I

As an alternative to the NLP model presented above, we are now going to propose a mixed-integer linear programming formulation.

Variables

In addition to the same price and volume variables as in the NLP model (3), $\mathbf{p} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^N$, respectively, the MIP model requires several other variables. First and foremost, let $\mathbf{z} \in \{0,1\}^N$ be a vector of binary variables, where $z_i \in \mathbf{z}$ indicates whether order ω_i may be (fully or partially) executed, or not. Precisely, we require $z = 1$ if and only if the prices of the tokens that are present in the order satisfy the respective limit price π_i, otherwise $z = 0$.

The feasible region for the execution volume v_i of an order ω_i depends on the value of z_i. In particular, v_i must be set to zero if $z_i = 0$ and can only be non-zero otherwise. In order to model this disjoint behaviour, we make use of a *disjunctive programming* formulation that needs the following auxiliary non-negative price variable vectors:

$$\mathbf{p}^{b,0}, \mathbf{p}^{b,1}, \mathbf{p}^{s,0}, \mathbf{p}^{s,1} \in \mathbb{R}^N_{\geq 0}.$$

Parameters

The model allows for setting a minimum and maximum fraction of execution for every order. For now, we will use global values $0 \leq \underline{r} \leq \overline{r} \leq 1$ and for all current purposes set $\overline{r} = 1$.

Model

$$\text{maximize} \quad \sum_{i \in \mathcal{I}^o} v_i \tag{4a}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{I}^o} t^b_{i,j}\, v_i = \sum_{i \in \mathcal{I}^o} t^s_{i,j}\, v_i \qquad \forall\, j \in \mathcal{I}^t \tag{4b}$$

$$\sum_{j \in \mathcal{I}^t} t^b_{i,j}\, \underline{p}_j (1 - z_i) \le p^{b,0}_i \le \sum_{j \in \mathcal{I}^t} t^b_{i,j}\, \overline{p}_j (1 - z_i) \qquad \forall\, i \in \mathcal{I}^o \tag{4c}$$

$$\sum_{j \in \mathcal{I}^t} t^s_{i,j}\, \underline{p}_j (1 - z_i) \le p^{s,0}_i \le \sum_{j \in \mathcal{I}^t} t^s_{i,j}\, \overline{p}_j (1 - z_i) \qquad \forall\, i \in \mathcal{I}^o \tag{4d}$$

$$\sum_{j \in \mathcal{I}^t} t^b_{i,j}\, \underline{p}_j z_i \le p^{b,1}_i \le \sum_{j \in \mathcal{I}^t} t^b_{i,j}\, \overline{p}_j z_i \qquad \forall\, i \in \mathcal{I}^o \tag{4e}$$

$$\sum_{j \in \mathcal{I}^t} t^s_{i,j}\, \underline{p}_j z_i \le p^{s,1}_i \le \sum_{j \in \mathcal{I}^t} t^s_{i,j}\, \overline{p}_j z_i \qquad \forall\, i \in \mathcal{I}^o \tag{4f}$$

$$\sum_{j \in \mathcal{I}^t} t^b_{i,j}\, p_j = p^{b,0}_i + p^{b,1}_i \qquad \forall\, i \in \mathcal{I}^o \tag{4g}$$

$$\sum_{j \in \mathcal{I}^t} t^s_{i,j}\, p_j = p^{s,0}_i + p^{s,1}_i \qquad \forall\, i \in \mathcal{I}^o \tag{4h}$$

$$p^{b,0}_i \le \pi_i\, p^{s,0}_i (1 - \varepsilon) \qquad \forall\, i \in \mathcal{I}^b \tag{4i}$$

$$p^{b,1}_i \ge \pi_i\, p^{s,1}_i \qquad \forall\, i \in \mathcal{I}^b \tag{4j}$$

$$v_i \ge \underline{r}\, \overline{x}_i\, p^{b,1}_i \qquad \forall\, i \in \mathcal{I}^b \tag{4k}$$

$$v_i \le \overline{r}\, \overline{x}_i\, p^{b,1}_i \qquad \forall\, i \in \mathcal{I}^b \tag{4l}$$

$$p^{s,0}_i \le \pi_i\, p^{b,0}_i (1 - \varepsilon) \qquad \forall\, i \in \mathcal{I}^s \tag{4m}$$

$$p^{s,1}_i \ge \pi_i\, p^{b,1}_i \qquad \forall\, i \in \mathcal{I}^s \tag{4n}$$

$$v_i \ge \underline{r}\, \overline{y}_i\, p^{s,1}_i \qquad \forall\, i \in \mathcal{I}^s \tag{4o}$$

$$v_i \le \overline{r}\, \overline{y}_i\, p^{s,1}_i \qquad \forall\, i \in \mathcal{I}^s \tag{4p}$$

$$z_i \in \{0, 1\} \qquad \forall\, i \in \mathcal{I}^o \tag{4q}$$

$$v_i,\, p^{b,0}_i,\, p^{b,1}_i,\, p^{s,0}_i,\, p^{s,1}_i \in \mathbb{R}_{\ge 0} \qquad \forall\, i \in \mathcal{I}^o \tag{4r}$$

$$p_j \in \mathbb{R}_{\ge 0} \qquad \forall\, j \in \mathcal{I}^t \tag{4s}$$

The objective function (4a) maximizes the total volume in terms of units of the reference token τ_0 that is processed with all orders.

Constraint (4b) secures that the total buy and sell volumes across all orders must be equal for every token. With uniform clearing prices, volume balance implies token balance.

The auxiliary variables in $\mathbf{p}^{b,0}$ and $\mathbf{p}^{s,0}$ refer to the prices of the buy- and sell-token of every order ω_i if that order is not executed ($z_i = 0$). In that case, their values must then lie within in the respective bounds provided by $\underline{\mathbf{p}}$ and $\overline{\mathbf{p}}$, and otherwise shall be set to zero. This requirement is ensured by the constraints (4c) and (4d). Note that the summations in the constraints are only needed to ensure that the right variable is selected from the respective variable vector.

Similarly as above, the constraints (4e) and (4f) control the auxiliary variables in $\mathbf{p}^{b,1}$ and $\mathbf{p}^{s,1}$ for the case that an order ω_i is executed ($z_i = 1$).

The relation between the auxiliary price variables and the actual token prices is established by the constraints (4g) and (4h).

For buy orders, the disjunctive behaviour is modelled by the constraints (4i)–(4l). The idea is as follows: If some order ω_i is not to be executed ($z_i = 0$), then the prices $p_i^{b,0}$ and $p_i^{s,0}$ must both lie within the bounds given by (4c) and (4d) as well as fulfill (4i) (i.e., not satisfy the limit price). We want to ensure that the token prices are at least a little margin off the stated limit price in that case, hence the multiplication with $(1 - \varepsilon)$. At the same time, $p_i^{b,1}$ and $p_i^{s,1}$ are set to zero by (4e) and (4f), thus trivially satisfying (4j). This then also implies the volume v_i to be set to zero by (4k) and (4l). Conversely, if ω_i is to be executed ($z_i = 1$), $p_i^{b,0}$ and $p_i^{s,0}$ are set to zero by (4c) and (4d), while $p_i^{b,1}$ and $p_i^{s,1}$ lie within the bounds provided by (4e) and (4f) as well as fulfill (4j) (i.e., satisfy the limit price). This finally requires the execution volume v_i to be within the specified fractions via the constraints (4l) and (4l).

The reasoning is analogous for sell orders and expressed by (4m)–(4p).

Computational results

The following Table 1 should give an indication of the solving times of the model w.r.t. different numbers of tokens and orders. Every entry in the table represents the mean solving time of a set of 10 instances. The buy/sell token quantities and limit prices of the orders have been randomly generated. In order to guarantee feasibility for all instances, we have selected a value of $\underline{r} = 0$, so the solver may choose to execute 0% of an order even if the token prices

		# orders per token pair			
# tokens	(# pairs)	4	8	20	40
2	(1)	0.05	0.06	0.08	0.10
3	(3)	0.07	0.12	0.31	0.89
4	(6)	0.13	0.24	0.74	3.09
5	(10)	0.25	0.69	4.22	84.36
6	(15)	0.40	2.04	77.32	684.40

Table 1: Solving times in [s] of the MIP formulation to global optimality.

comply with the limit price. We have modelled the problem using PYTHON/PYOMO and employed GUROBI 7.5.2 as a solver.

It can be seen that the running times increase sharply for 5 or more tokens and the many-order cases. One influencing factor might be the equal number of orders for all token pairs in our testset, which leads all token prices being maximally interconnected and which might not occur as such in practice. On the other hand, the more orders there are on some token pair, the denser the stated limit prices become, which might make it possible to aggregate orders as a means of preprocessing.

3.3 Mixed-integer linear programming model II

The previous MIP model (4) considers all orders independent from each other, i.e., the decision whether some order should be executed or not is not explicitely connected to the decision for other orders. The connection is only indirectly established through the propagation of feasible price ranges. In practice, however, there is more structure to the execution of orders, particularly within the same token pair. For instance, if some sell order with limit price π^* is enabled by the current choice of prices, all other sell orders with higher limit prices should be executed as well (and vice-versa for buy orders). This insight gives rise to an alternative MIP formulation that we will present in the following.

Data

We want to aggregate orders on every pair of tokens that is traded, so let the set of token index pairs be denoted by $\mathcal{I}^p := \{(j,k) \in \mathcal{I}^t \times \mathcal{I}^t, j \neq k\}$, with $(j,k) \in \mathcal{I}^p$ representing the pair of tokens τ_j and τ_k. Notice that we consider the token pairs to be ordered tuples, so $(j,k) \in \mathcal{I}^p$ and $(k,j) \in \mathcal{I}^p$ are treated separately.

For every token pair $(j,k) \in \mathcal{I}^p$, we collect all buy and sell orders $(\tau_j, \tau_k, \cdot, \cdot)$ that were

submitted for (j,k), and sort them in an increasing order by their limit prices. Assuming that there are m orders for token pair (j,k), we get

$$\underline{p}_{j|k} =: \pi_{j,k}^{(0)} < \pi_{j,k}^{(1)} < \pi_{j,k}^{(2)} < \ldots < \pi_{j,k}^{(m)} < \pi_{j,k}^{(m+1)} := \overline{p}_{j|k}, \tag{5}$$

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). Notice, moreover, that orders of the same type (buy/sell) with the same limit price are aggregated into one single order beforehand, such that the strict inequalities always hold. The above ordering (5) defines regions $r_l := [\pi^{(l)}, \pi^{(l+1)}]$, $l \in \{0 \ldots m\}$, for the exchange rate $p_{j|k}$. Let $\mathcal{I}^r_{j,k} := \{0 \ldots m\}$ denote the set of 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 $\tilde{x}^b_{j,k,l}$ denote to the cumulated amount of tokens τ_j that could be bought across all buy orders of token pair (j,k), if the exchange rate $p_{j|k}$ were in r_l. Similarly, $\tilde{y}^s_{j,k,l}$ shall refer to the amount of tokens τ_j to be sold across all sell orders under the same condition. Let $\tilde{\mathbf{x}} := (\tilde{x}_{j,k,l})$ and $\tilde{\mathbf{y}} := (\tilde{y}_{j,k,l})$. From here on, we are only working with this aggregated order representation.

Variables

We will use the same price variables $\mathbf{p} \in \mathbb{R}^n_{\geq 0}$ as previously. In addition, we will use variables $(v^{\text{abs}}_{j,k}) =: \mathbf{v}^{\text{abs}} \in \mathbb{R}^{|\mathcal{I}^p|}_{\geq 0}$ and $(v^{\text{net}}_{j,k}) =: \mathbf{v}^{\text{net}} \in \mathbb{R}^{|\mathcal{I}^p|}_{\geq 0}$ to represent the total absolute and net volume (in terms of units of token τ_0) that is traded on each token pair. Furthermore, for every token pair (j,k) and every exchange rate interval r_l, $l \in \mathcal{I}^r_{j,k}$, we introduce a binary variable $z_{j,k,l} \in \{0,1\}$ that indicates whether the exchange rate $p_{j|k} = \frac{p_j}{p_k}$ lies within the interval r_l or not. Let $\mathbf{z} := (z_{j,k,l})$. In order for the disjunctive MIP formulation to work, we again need a set of auxiliary (disaggregated) variables. Let $\mathbf{v}^b := (v^b_{j,k,l})$ and $\mathbf{v}^s := (v^s_{j,k,l})$ represent the volumes that are traded in buy and sell orders, respectively, for every token pair and every exchange rate interval. Moreover, $\mathbf{p}^{(1)} := (p^{(1)}_{j,k,l})$ and $\mathbf{p}^{(2)} := (p^{(2)}_{j,k,l})$ shall denote the disaggregated price variables for the two respective tokens of every token pair and all price intervals.

Parameters

This model allows for the same parameters $\underline{r}$ and $\overline{r}$ as the previous MIP model (4).

Model

$$\text{maximize} \quad \sum_{(j,k)\in\mathcal{I}^p} v^{\text{abs}}_{j,k} \tag{6a}$$

$$\text{subject to} \quad \sum_{\substack{k\in\mathcal{I}^t\\ k\neq j}} v^{\text{net}}_{k,j} = \sum_{\substack{k\in\mathcal{I}^t\\ k\neq j}} v^{\text{net}}_{j,k} \qquad \forall\, j\in\mathcal{I}^t \tag{6b}$$

$$\sum_{l\in\mathcal{I}^r_{j,k}} z_{j,k,l} = 1 \qquad \forall\,(j,k)\in\mathcal{I}^p \tag{6c}$$

$$v^{\text{abs}}_{j,k} = \sum_{l\in\mathcal{I}^r_{j,k}} \left(v^b_{j,k,l} + v^s_{j,k,l}\right) \qquad \forall\,(j,k)\in\mathcal{I}^p \tag{6d}$$

$$v^{\text{net}}_{j,k} = \sum_{l\in\mathcal{I}^r_{j,k}} \left(v^b_{j,k,l} - v^s_{j,k,l}\right) \qquad \forall\,(j,k)\in\mathcal{I}^p \tag{6e}$$

$$p_j = \sum_{l\in\mathcal{I}^r_{j,k}} p^{(1)}_{j,k,l} \qquad \forall\,(j,k)\in\mathcal{I}^p \tag{6f}$$

$$p_k = \sum_{l\in\mathcal{I}^r_{j,k}} p^{(2)}_{j,k,l} \qquad \forall\,(j,k)\in\mathcal{I}^p \tag{6g}$$

$$p^{(1)}_{j,k,l} \geq p^{(2)}_{j,k,l}\,\pi^{(l)}_{j,k} \qquad \forall\,(j,k)\in\mathcal{I}^p, l\in\mathcal{I}^r_{j,k} \tag{6h}$$

$$p^{(1)}_{j,k,l} \leq p^{(2)}_{j,k,l}\,\pi^{(l+1)}_{j,k} \qquad \forall\,(j,k)\in\mathcal{I}^p, l\in\mathcal{I}^r_{j,k} \tag{6i}$$

$$p^{(2)}_{j,k,l} \geq \underline{p}_k\, z_{j,k,l} \qquad \forall\,(j,k)\in\mathcal{I}^p, l\in\mathcal{I}^r_{j,k} \tag{6j}$$

$$p^{(2)}_{j,k,l} \leq \overline{p}_k\, z_{j,k,l} \qquad \forall\,(j,k)\in\mathcal{I}^p, l\in\mathcal{I}^r_{j,k} \tag{6k}$$

$$v^b_{j,k,l} \geq \underline{r}\,\tilde{x}^b_{j,k,l}\,p^{(1)}_{j,k,l} \qquad \forall\,(j,k)\in\mathcal{I}^p, l\in\mathcal{I}^r_{j,k} \tag{6l}$$

$$v^b_{j,k,l} \leq \overline{r}\,\tilde{x}^b_{j,k,l}\,p^{(1)}_{j,k,l} \qquad \forall\,(j,k)\in\mathcal{I}^p, l\in\mathcal{I}^r_{j,k} \tag{6m}$$

$$v^s_{j,k,l} \geq \underline{r}\,\tilde{y}^s_{j,k,l}\,p^{(1)}_{j,k,l} \qquad \forall\,(j,k)\in\mathcal{I}^p, l\in\mathcal{I}^r_{j,k} \tag{6n}$$

$$v^s_{j,k,l} \leq \overline{r}\,\tilde{y}^s_{j,k,l}\,p^{(1)}_{j,k,l} \qquad \forall\,(j,k)\in\mathcal{I}^p, l\in\mathcal{I}^r_{j,k} \tag{6o}$$

$$p_j \in \mathbb{R}_{\geq 0} \qquad \forall\, j\in\mathcal{I}^t \tag{6p}$$

$$v^{\text{abs}}_{j,k}, v^{\text{net}}_{j,k} \in \mathbb{R}_{\geq 0} \qquad \forall\,(j,k)\in\mathcal{I}^p \tag{6q}$$

$$z_{j,k,l} \in \{0,1\} \qquad \forall\,(j,k)\in\mathcal{I}^p, l\in\mathcal{I}^r_{j,k} \tag{6r}$$

$$p^{(1)}_{j,k,l}, p^{(2)}_{j,k,l}, v^b_{j,k,l}, v^s_{j,k,l} \in \mathbb{R}_{\geq 0} \qquad \forall\,(j,k)\in\mathcal{I}^p, l\in\mathcal{I}^r_{j,k} \tag{6s}$$

The objective function (6a) maximizes the sum of the trading volumes over all trading pairs and is supposed to yield the same value as the objective (4a) of the previous MIP model.

The first constraint (6b) 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.

All other constraints are induced by the disjunctive formulation that our model is based on. For every token pair, the solver may choose exactly one price range, which is indicated by the constraint (6c). The purpose of the following constraints (6d)– (6g) is the aggregation of the disaggregated variables, both for token prices and trading volumes. This builds the connection between the auxiliary variables and the actual price and volume variables. A main feature of disjunctive programming is that the auxiliary variables belonging to some part of the disjunction are set to zero if that part is not selected to be active, and that they take some meaningful value otherwise. This is expressed by the constraints (6h)–(6o). Thereby, the main dependency between the binary and the auxiliary variables is given in (6j) and (6k), where auxiliary prices on some disjunction part are either set to zero or are restricted to be within (non-negative) bounds. If they are set to zero, it is implied that all auxiliary variables for the same part are also set to zero by the other constraints. In the other case, the respective auxiliary variables are restricted to fulfill the semantics of the model, which are similar to the previous MIP model (4).

3.4 Mixed-integer linear programming model III

In 3.3, we consider aggregated orders on directed token pairs, so the pair (τ_i, τ_j) exists as well as (τ_j, τ_i). However, we can also aggregate further and only consider undirected token pairs $\{\tau_i, \tau_j\}$.

TODO!

3.5 Min-cost flow model

???

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

$\Rightarrow \underline{p} = 0.5, \overline{p} = 2.0, \underline{r} = 0.2$

TODO!

The results of our computational experiments as in Table 2, Table 3 and Table 4 show that runtimes of the MIP formulation sharply increase both with the number of assets and the number of orders that are being considered. Several instances with more than 20 assets and more than 200 orders could not even be solved to optimality within the timelimit. 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

		# assets			
# orders		5	10	20	50
100	∅ runtime	0.32	0.32	0.90	5.54
	# timeouts	0	0	0	0
	– ∅ gap	-	-	-	-
200	∅ runtime	1.31	2.49	26.25	571.10
	# timeouts	0	0	0	12
	– ∅ gap	-	-	-	23.46%
500	∅ runtime	15.67	82.47	522.72	.
	# timeouts	0	0	11	-
	– ∅ gap	-	-	21.59%	-

Table 2: Computational results for MIP model I (4).

# orders		# assets			
		5	10	20	50
100	∅ runtime	.	.	.	.
	# timeouts	.	.	.	.
	– ∅ gap	.	.	.	.
200	∅ runtime	.	.	.	.
	# timeouts	.	.	.	.
	– ∅ gap	.	.	.	.
500	∅ runtime	.	.	.	.
	# timeouts	.	.	.	.
	– ∅ gap	.	.	.	.

Table 3: Computational results for MIP model II (6).

# orders		# assets			
		5	10	20	50
100	∅ runtime	.	.	.	.
	# timeouts	.	.	.	.
	– ∅ gap	.	.	.	.
200	∅ runtime	.	.	.	.
	# timeouts	.	.	.	.
	– ∅ gap	.	.	.	.
500	∅ runtime	.	.	.	.
	# timeouts	.	.	.	.
	– ∅ gap	.	.	.	.

Table 4: Computational results for MIP model III (**??**).

4 Extensions

The problem can possibly be extended in various directions:

- 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.