The Optimization problem

In this section, we describe all the different components of the optimization problem that needs to be solved within each batch.

User orders

Suppose that there are{ 1 , 2 , . . . k } {1,2,...k} { 1 ,

2 ,

... k } tokens. From a high-level perspective, we can define a user order as anacceptance set S ⊂ R k S \subset \mathbb R^k S

⊂ R k specifying the trades a user is willing to accept (where negative entries of a vector represent tokens sold, while positive entries represent tokens bought). So, for example, ifk = 2 k=2 k

= 2 and[x − y] ∈ S \begin{bmatrix} x \-y \end{bmatrix}\in S [x − y]

∈ S then a user is happy to receivex units of token 1 in exchange fory units of token 2.

note This is from the user's perspective, and is therefore net of fees. We also assume that0 ∈ S 0 \in S 0

∈ S that is, when submitting an order a user accepts that the order may not be filled. Also, to each orderS S S we definesurplus_function U S : S → R U_S:S\rightarrow \mathbb R U S

: S

→ R , measuring "how good" a trade is from the point of view of the user who submitted orderS . By definitionU S (0) = 0 U_S(0)=0 U S (0)

= 0 .

Practically speaking, CoW Protocol allows only some types of orders, which we can think of as constraints on the setS that a user can submit. One such constraint is that only pairwise swaps are allowed, that is, all vectors inS S S have zeros ink − 2 k-2 k

− 2 dimensions. Furthermore, each order must fit within one of the categories we now discuss. To simplify notation, when discussing these categories, we assume thatk = 2 k=2 k

= 2 .

Limit Sell Orders

Alimit sell order specifies a maximum sell amount of a given tokenY

0, a buy tokenb , and a limit priceπ \pi π , that corresponds to the worst-case exchange rate that the user is willing to settle for. They can be fill-or-kill whenever the executed sell amount must be Y (or nothing). They can be partially fillable if the executed sell amount can be smaller or equal to Y. Formally, ifx denotes the (proposed) buy amount andy denotes the (proposed) sell amount of the order, a fill-or-kill limit sell order has the form

S

{ [x − y]

s . t .

y π ≤ x and y ∈ { 0 , Y } } , S=\left{\begin{bmatrix} x \-y \end{bmatrix}~~s.t. ~~\frac{y}{\pi}\leq x \hbox{ and } y\in{0,Y} \right}, S

= { [x − y]

s . t .

π

y

≤

x and y

∈

{ 0 ,

Y } }

,

and a partially-fillable sell order has the form

S

{ [x – y]

s . t .

y π ≤ x and y ∈ [0 , Y] } . S= \left { \begin{bmatrix} x \-y \end{bmatrix} ~~s.t. ~~\frac{y}{\pi} \leq x \hbox{ and } y \in [0,Y] \right }. S

= { [x – y]

s . t .

π

y

≤

x and y

∈

[0 ,

Y] }

.

In both cases, the surplus function is defined as

U (x , – y) = (x – y / π) p (b) U(x,-y)=(x-y / \pi)p(b) U (x ,

– y)

= (x

– y / π) p (b) ,

where(x – y / π) (x-y / \pi) (x

– y / π) is the additional amount of buy tokens received by the user relative to the case in which they trade at the limit price, andp (b) p(b) p (b) is the price of the buy token relative to a numéraire (in our case ETH) and is externally provided (i.e., by an oracle). The functionU (x , – y) U(x,-y) U (x ,

– y) is therefore expressed in units of the numéraire and is always non-negative.

A final observation is that orders can be valid over multiple batches. For a fill-or-kill, this means that an order that is not filled remains valid for a certain period (specified by the user). For a partially-fillable order, this also means that only a fraction of it may be executed in any given batch.

Limit Buy Orders

Alimit buy order is specified by a maximum buy amountX

> 0 and a limit priceπ \pi π corresponding to the worst-case exchange rate the user is willing to settle for. Withx denoting the buy amount andy denoting the sell amount of the order, fill-or-kill limit buy orders have the form

S

{ [x – y]

s . t .

y ≤ x · π and x ∈ { 0 , X } } S = \left{\begin{bmatrix} x \-y \end{bmatrix}~~s.t.~~ y \leq x \cdot \pi \hbox{ and } x \in{0, X} \right} S

= { [x – y]

s . t .

y

≤

x

·

π and x

∈

{ 0 ,

X } }

while partially-fillable limit buy orders have the form

S

{ [x – y]

s . t .

y ≤ x · π and x ∈ [0 , X] } . S = \left{\begin{bmatrix} x \-y \end{bmatrix}~~s.t.~~ y \leq x \cdot \pi \hbox{ and } x \in[0, X] \right}. S

= { [x – y]

s . t .

y

≤

x

·

π and x

∈

[0 ,

X] }

.

Again, the surplus function is defined as

U ({ x , – y }) = (x · π – y) p (s) U({x,-y})=(x \cdot \pi-y)p(s) U ({ x ,

– y })

= (x

· π

– y) p (s) ,

wherep (s) p(s) p (s) is the price of the sell token relative to a numéraire and is externally provided. Also here, orders can be executed over multiple batches.

Fees

Each user order has an associated fee paid to the protocol. At a high level, these fees can be represented by a function that, for a given orderS S S maps all possible trades to a positive vector of tokens, that isf S : S → R + k f_S:S \rightarrow \mathbb R^k_+ f S

: S

→ R + k withf S (0) = 0 f_S(0)=0 f S (0)

= 0 .

From the practical viewpoint, for market fill-or-kill orders, the fee is always in the sell token and is pre-specified: it is an estimate of the cost of executing an order and is explicitly shown to the user before the order is submitted. Instead, (long-standing) limit orders are "feeless" from the user's perspective: users are guaranteed a limit price without specifying how fees will be calculated. Solvers are the ones proposing a fee under the expectation that this fee should equal the cost of execution of this trade in isolation. The fee of limit orders is again denominated in the sell token.

Solution

Solvers propose solutions to the protocol, where a solution is a set of trades to execute. Formally, suppose there areI I I users andJ liquidity sources. A solution is a list of trades{ o 1 , o 2 , . . . o I , l 1 , l 2 , . . . , l J } {o_1, o_2, ...o_I, l_1, l_2, ..., l_J} { o 1 ,

o 2 ,

... o I ,

l 1 ,

l 2 ,

... ,

l J } one per user and one per liquidity source such that:

- Incentive compatibility and feasibility
- : the trades respect the user and liquidity sources, that is,o
- i
- ∈
- S
- i
- ∀
- i
- ≤
- I
- o_i\in S_i~~\forall i\leq I
- o
- i
-
- ∈
- S
- i
-
- ∀
- i
- ≤
- I
- andl
- j
- ∈
- L
- j
- ∀

- j
- ≤
- J
- l_j \in L_j~~\forall j\leq J
- l
- j
-
- ∈
- L
- j
-
- ∀
- j
- ≤
- J
- .
- Sufficient endowments
- : each user should have enough sell tokens in their wallet. Note that the protocol already performs this check at order creation. However, a user could move funds between order creation and execution or create multiple orders pledging the same sell amount multiple times. Hence, each solver should also check that users' endowments are sufficient to execute the proposed solution.
- Uniform clearing prices
- : all users must face the same prices. Importantly, this constraint is defined at the moment when the swap occurs. So, for example, suppose useri
- receivesx
- units of token 1 in exchange fory
- units of token 2 and that the protocol takes a fee in the sell tokenf
- 2
- f_2
- f
- 2
-
- . Definep
- 1
- ,
- 2
- =
- y
- −
- f
- 2
- x
- p_{1,2}=\frac{y-f_2}{x}
- p
- 1
- ,
- 2
-
- =
- x
- y
- −
- f
- 2
-
-
- as the price at which the swap occurs. Uniform clearing prices means thatp
- 1
- ,
- 2
- p_{1,2}
- p
- 1
- ,
- 2
-
- is the same for all users swapping token 1 and token 2. Furthermore, prices must be consistent, in the sense that for any three tokens 1, 2, and 3, ifp

- 1
- ,
- 2
- ,
- p
- 2
- ,
- 3
- ,
- p
- 1
- ,
- 3
- $p_{1,2},~ p_{2,3}, ~p_{1,3}$
- p
- 1
- ,
- 2
-
- ,
- p
- 2
- ,
- 3
-
- ,
- p
- 1
- ,
- 3
-
- are well-defined, then it must be thatp
- 1
- ,
- 2
- ⋅
- p
- 2
- ,
- 3
- =
- p
- 1
- ,
- 3
- $p_{1,2}\cdot p_{2,3}=p_{1,3}$
- p
- 1
- ,
- 2
-
- ⋅
- p
- 2
- ,
- 3
-
- =
- p
- 1
- ,
- 3
-
- . Note that this implies that prices can be expressed with respect to a common numéraire, giving rise to a uniform price clearing vectorp
- p
- p
- .

- Social consensus rules
- : These are a set of principles that solvers should follow, which were voted by CIPs.

caution At CoW DAO's discretion, systematic violation of these rules may lead to penalizing or slashing of the offending solver. From the protocol viewpoint, each solution that satisfies the above constraints has aquality given by the total surplus generated and the fees paid to the protocol:

∑ o U (o) + p · ∑ o f (o) \sum_o U(o)+p\cdot \sum_of(o) ∑ o

U (o)

+ p

· ∑ o

f (o) ,

wherep is a vector of externally-determined prices used to express all fees in terms of the common numéraire.

Finally, solvers compete for the right to settle a batch by participating in an auction, aiming to implement the solution with the highest quality at the lowest possible cost to the protocol. Thesolver that wins the auction is rewarded by the protocol.
Edit this page Previous Batch Auction mechanism Next Solver competition rules