Global Order Routing on Exchange Networks

Vincent Danos, Hamza El Khalloufi, Julien Prat

Global order routing on exchange networks

Vincent Danos[1], Hamza El Khalloufi[1], and Julien Prat[2]

[1] CNRS, DI ENS, PSL-ENS, INRIA
[2] CNRS, CREST, École polytechnique

Abstract. We propose an abstract notion of networks of exchanges with an eye to modelling the global money market of DeFi (decentralised finance). We formalise routing and arbitrage on such networks as convex optimisation problems. We provide bounds with closed formulas in the specific case of Uniswap-like automated markets and a restricted form of cyclic arbitrage. We compute the associated lower bounds on actual data derived from the Ethereum blockchain.

1 Introduction

The global money market of DeFi (decentralised finance) allows traders to exchange assets represented by ERC20 tokens on the Ethereum blockchain. Each money market on a specific pair of tokens A/B can be seen as a 2-sided platform where liquidity providers transact with liquidity consumers. There are several implementations of such platforms. Some use the traditional form of the limit order book [1], but most use the so-called constant function automated market makers (eg Uniswap v2 [2]) which compute prices algorithmically as a function of their current reserves in A and B. The intense competition for liquidity and high clonability of the said platforms has given rise to a dense and complex network (a tiny subgraph of which is shown in Fig. 5). Prior academic work on such networks [3,4] focusses on local questions, with the exception of the recent Ref. [7] which looks at cyclic arbitrage (about which more later). Our contribution explores two global questions centred on liquidity consumers (also known as takers or traders).

The first question is *routing*: eg "Here I have a 100 ETH, how should I best convert them to DAIs". Unsurprisingly, direct routes may not always be best, and convex combination of routes may dominate any particular path. Fig. 1 gives an actual example of routing a 100 ETH in order to maximise the amount of DAIs obtained. One sees that the order is split among 7 distinct money markets.

The second question is *arbitrage* (aka price consistency): eg in a given state of the network "Is there any way I can chain operations leading to certain non-zero profit". We define in this paper a class of convex optimisation problems which encompasses both the global routing problem and the arbitrage problem. We show that problems in this class always have solutions. We also show that optimal arbitrage eliminates price inconsistencies. One can construe this result as saying that the global network 'learns' a consistent set of prices under optimal arbitrage.

For both questions we introduce tractable sub-problems. Specifically, we demonstrate that arbitraging along cycles, while sub-optimal in general can be efficiently tested and solved. This leads to computationally cheap lower bounds on profit. For Uniswap price functions, we derive a closed formula to compute the maximal extractible profit on any given cycle. Putting the two results together (cyclic arbitrage detection and explicit max profit), we look at actual Ethereum data and efficiently find lower bounds for several arbitrage opportunities. For routing we introduce the convex subproblem where one restricts to convex combinations of a given set of independent paths. It is possible to derive a simple algorithm for optimal convex combinations of Uniswap price functions, which is of independent interest and that we will publish elsewhere.

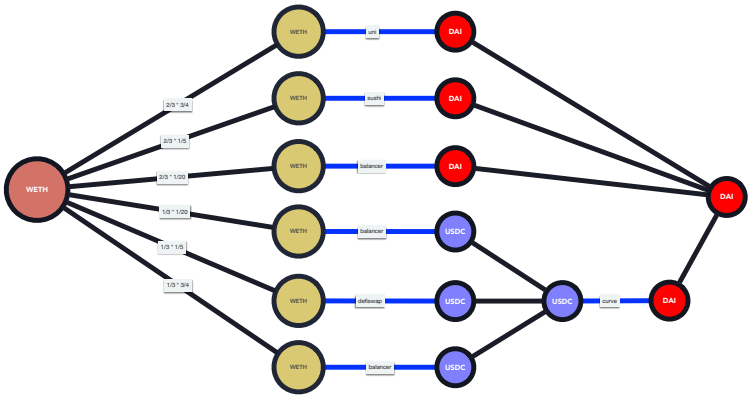


Fig. 1: A transfer plan using 7 distinct money markets (blue edges). Notice that a third of the original 100 ETH travels through an indirect path.

Acknowledgments: The authors wish to thank Saad Bouhoud, Ayman Elyahmidi, and Vincent Bernardoff for data-related discussions, and for sharing code to obtain the relevant Uniswap data from an Ethereum node. The authors also wish to thank Jérôme de Tychey for numerous interesting discussions on the matters of this paper, and Ye Wang and co-authors for sharing an early version of their paper on cyclic arbitrage [7].

2 Prices, Plans, Profits

Many price functions have been considered to define prices in algorithmic exchanges. We will use the specific Uniswap function (defined right below) to obtain closed form solutions for cyclic arbitrage. Up to that point all our development is generic and relies only on the following abstract definition.

Definition 1 *A* price function *is a map* $f : \mathbb{R}_+ \to \mathbb{R}_+$ *such that* $f(0) = 0$, f *is monotonically increasing, strictly concave, bounded, and continuous.*

We also sometimes suppose our price functions are differentiable. It will be clear when we do.

The requirements encapsulated in Def. 1 are natural for prices: $f(0) = 0$ means one gets nothing for nothing, increasing means one gets more for more, concave means returns decrease, and boundedness expresses the fact that liquidity reserves are finite.

Notice that a (half) order-book on an A/B pair -ie a price-sorted list of discrete offers (p_i, V_i) where V_i is an amount of Bs and p_i the unit price to pay in A to take that offer- also defines a price function. It just happens to be piecewise affine (hence is not everywhere differentiable). Boundedness holds whether liquidity is provided via an order book or a constant function market-maker.

A simple example of price function is the Uniswap one. For an A/B pair it reads:

$$f(x) := [B](\gamma x)/(\gamma x + [A]) \tag{1}$$

where $[A]$, $[B] > 0$ are the local reserves (or pools) in tokens A and B. The γ parameter is such that $0 \leq 1 - \gamma < 1$, and $1 - \gamma$ represents the fee extracted by the liquidity providers for every transaction. It is easy to see that $0 \leq f(x) < [B]$. Hence a Uniswap pair never pays out more

than the reserve however small.[3] This function satisfies all the requirements of a price function as defined above. The dual price function which specifies how many As one gets for a given amount of Bs is obtained simply by exchanging the roles of $[A]$, $[B]$ (not to be confused with f^{-1} which specifies how many As one needs to spend to obtain a given amount of Bs).

The reserves $[A]$, $[B]$ are modified by every transaction and therefore induce a modification of the price function -also known as the price impact of the transaction.[4]

The class of price functions is closed under: sum, composition, pre- and post-composition by positive scalar multiplication.

Definition 2 *An* exchange network *consists of:*
- *an undirected multi-graph $G = (V, E)$, without loops*
- *a fixed chosen orientation for each edge e in E*
- *a family of price functions $(f_e;\ e \in E)$*

We write $s(e)$, $t(e) \in V$ for the source and target of an edge e. Hence both s and t have type $E \to V$; and for $A \in V$, $s^{-1}(A)$ is the set of edges emanating from A, while $t^{-1}(A)$ is the set of edges pointing to A.

Given $\phi = e_1, \ldots, e_n$ a simple path in G, set:

$$f_\phi = f_{e_n} \circ \cdots \circ f_{e_1}$$

f_ϕ is a price function (by composition). The path has to be simple, else the first visit to an edge may change the reserves, and the second one will find an updated price function. In this paper, we only rarely consider specific update mechanisms, and almost always work with a given fixed state of the network of abstract price functions.

An example of such a structure is a snapshot of the Uniswap network. Nodes are ERC20s, edges are pairs with a chosen orientation. The level of reserves in each pair determines the price function f_e. In reality, the graph part also changes slowly as new nodes and pairs are added.

We now define our main object of interest:

Definition 3 *A* transfer plan τ *is an element of the standard cone $E \to \mathbb{R}_+$ (hence convex). The* support *of τ is the subset of E where τ is non-zero.*

For each e, $\tau(e) \geq 0$ specifies the amount of $s(e)$ injected in the price function f_e.

There are implicit restrictions in Def. 3 which are worth discussing. We do not consider sliced orders, ie repeated swaps on the same directed edge. It is known for constant function aMMs that slicing leads to sub-optimal plans [3]. Also, we do not consider backtracking, ie transfer plans where an underlying edge is used twice with opposite orientations. This is because each edge is directed.[5] Finally, we do not incorporate in our notion of transfer plan the very real possibility which DeFi agents have, namely to modify the reserves (eg by depositing or withdrawing liquidity in the various pools of aMMs) and therefore the price functions while trading. It seems unlikely that backtracking or liquidity modifications could improve transfer plans relative to the objectives given below, but it remains to be seen.

A plan τ translates directly in a concrete Ethereum transaction. Because of the existence of flash loans, at least for most liquid tokens, the order in which each elementary swaps are performed is irrelevant. We neglect in this model gas costs and flash loan fees.

[3] Actually one could redefine f as $f(x) = f(\infty)(xf'(0))/(xf'(0) + f(\infty))$ as $f(\infty) = [B]$ the total reserve in $[B]$, and $f'(0) = \gamma[B]/[A]$ the marginal price of A in B.

[4] One could call the Uniswap price function a 'price state machine' as it defines a price function for every state of its internal reserves.

[5] To express backtracking plans and allow sequences of swaps on e with alternating orientations, one could use the larger cone of finite sequences $E \to \sum_n \mathbb{R}_+^n$.

Given (G, f) we can now define the profit map Ψ which maps a transfer plan (an E-vector), to its resulting balance (a V-vector):

Definition 4 *The profit map $\Psi : \mathbb{R}_+^E \to \mathbb{R}^V$ derives from the data as follows:*

$$\begin{array}{ll} \Psi(\tau)(A) & := \Psi_+(\tau)(A) - \Psi_-(\tau)(A) \\ \Psi_+(\tau)(A) & := \sum_{e \in t^{-1}(A)} f_e(\tau(e)) \\ \Psi_-(\tau)(A) & := \sum_{e' \in s^{-1}(A)} \tau(e') \end{array}$$

The first component $\Psi_+(\tau)(A)$ is the amount of A returned by τ, while the second $\Psi_-(\tau)(A)$ is the amount of A invested by τ. The difference $\Psi(\tau)(A)$ is the balance change of A as a result of executing τ -ie the profit (which can be negative!).

We see that:
- $\Psi_+(\_)(A)$ is concave, non-decreasing, bounded, differentiable (if edges are); it is only increasing and strictly concave in those components τ_e such that $t(e) = A$.
- $\Psi_-(\_)(A)$ is a linear function $\mathbb{R}_+^E \to \mathbb{R}_+$;
- $\Psi(\_)(A)$ is a concave function bounded above by $\sum_{e \in t^{-1}(A)} f_e(\infty)$ (sum of A's liquidities available in the network) and $\Psi(0_E) = 0_V$.

Say a node A is:
- a *source* in τ if $\Psi_+(\tau)(A) = 0$, $\Psi_-(\tau)(A) > 0$,
- a *sink* if $\Psi_+(\tau)(A) > 0$, $\Psi_-(\tau)(A) = 0$,
- an *intermediate* if $\Psi_+(\tau)(A) = \Psi_-(\tau)(A) > 0$.

For differentiable price functions we can compute Ψ's Jacobian:

$$\Psi(\tau + h)(A) - \Psi(\tau)(A) = \sum_{e \in t^{-1}(A)} (h_e f'_e(\tau(e)) + o(h_e)) - \sum_{e' \in s^{-1}(A)} h_{e'}$$

So the Jacobian matrix of Ψ of dimension $V \times E$ is:

$$J\Psi(\tau)(A, e) = \begin{cases} f'_e(\tau(e)) & \text{if } e \in t^{-1}(A) \\ -1 & \text{if } e \in s^{-1}(A) \\ 0 & \text{else} \end{cases}$$

(Note that as G is loopless, no e is both in $s^{-1}(A)$ and $t^{-1}(A)$.)

As said, a transfer plan also has a side effect (here a monoid action) on the price functions which we denote by $f_e \mapsto^\tau \tau(e) \cdot f_e$; that is to say if f_e is the price function of edge e, and the amount $\tau(e)$ is injected in e, we write $\tau(e) \cdot f_e$ for the new price function. We also write more generally $\tau \cdot \Psi$ for the new profit function induced by the execution of plan τ. We leave this action implicit and just ask that it verifies the *no-slicing* property:

$$\Psi(\tau_1 + \tau_2) \geq \Psi(\tau_1) + \tau_1 \cdot \Psi(\tau_2)$$

This inequality expresses the fact that no profit can be made by simply slicing a plan in two parts. It is easy to see that Uniswap price functions satisfy no-slicing, and, therefore, so do profit functions derived from Uniswap price functions.

We can represent some of the plans defined above in a more diagrammatic way. Fig. 1 (retrieved from the 1inch price aggregator web site) gave an example with a unique source ETH and unique sink DAI and one intermediate node USDC (with zero balance). The plan τ invests at ETH and collects the returns at DAI: $\Psi_-(\tau)(\text{ETH}) = x$, $\Psi_+(\tau)(\text{ETH}) = 0$; $\Psi_-(\tau)(\text{DAI}) = 0$, $\Psi_+(\tau)(\text{DAI}) = y$; y can be computed as the composite function of x indicated by the diagram.

Diagrams are convenient ways to represent those plans that have only sources, sinks, and intermediates. Every diagram gives rise to a plan. Not every plan is a diagram, but with respect to some of the objectives functions presented in the next section, optimal ones will be.

3 Orders, routes, arbitrage

We define now a number of convex problems for transfer plans in standard form [5, §4.2.1]. In each case we ask whether the problem is feasible and has a bounded objective function. At the end of the section, we prove the existence of solutions for these problems (provided they are feasible). As profit functions are not strictly concave in all coordinates, the solutions may not be unique in general.

3.1 Smart order routing

Recall the routing question in the introduction: "what is the maximum amount of DAI I can get for 100ETH using all aMMs and dexes available?"

This question can be recast as the *forward routing problem* with $A \neq B$, $a \in \mathbb{R}_+$:

$$sor(a : A, B) = \begin{array}{ll} \max & \Psi(\tau)(B) \\ \text{with} & \Psi(\tau)(A) \geq -a \\ & \Psi(\tau)(C) \geq 0,\ C \neq A, B \end{array} \tag{2}$$

For a plan to satisfy the constraints (aka be feasible), it should not cost more than $a : A$ (but could invest more), and have non-negative balance for Cs which are neither A nor B. The zero plan 0_E satisfies the constraints, so the problem is feasible.

An optimal τ will have $\Psi(\tau)(A) = -a$, iff A and B are connected in G. One says the constraint is active in this case. Indeed, pick any path ϕ from A to B in G, if there is some unspent A in τ, ie $\Psi(\tau)(A) > -a$ one can push the remainder $a' > 0$ through ϕ to obtain $f_\phi(a') > 0$ additional Bs. Conversely, if there is no possible way to use some A to get some B, the amount of A spent is indifferent. Likewise, the C constraints will be active for optimum τ if connected to B.

We also see that for the routing problem to be sensible, there must be a compatibility with the orientation of G in the sense that directed paths must exist between inputs and outputs of the problem. Else the problem is degenerate.

As the user may have several types of tokens on hand, it makes senses to generalise the above allowing for multiple inputs (but still one output):

$$sor(a_1 : A_1, \ldots, a_n : A_n, B) = \begin{array}{ll} \max & \Psi(\tau)(B) \\ \text{with} & \Psi(\tau)(A_i) \geq -a_i \\ & \Psi(\tau)(C) \geq 0,\ C \neq A_i, B \end{array} \tag{3}$$

What about the inverse question "what is the minimum amount of ETH I need to spend to get 100DAI?". This question can also recast as the *backward routing problem* with $A \neq B$, with $b \in \mathbb{R}_+$:

$$sor(A, b : B) = \begin{array}{ll} \max & \Psi(\tau)(A) \\ \text{with} & \Psi(\tau)(B) \geq b \\ & \Psi(\tau)(C) \geq 0,\ C \neq A, B \end{array} \tag{4}$$

A feasible plan is one that pays $\geq b$ tokens of type B and has positive balance on all Cs. Differently from the forward problem, the backward feasibility set may be empty. Suppose the only edge in G joins A to B and $b > f_e(\infty)$. No matter how much money one injects on the A side, it will never obtain b. If the feasibility set is not empty, an optimum plan will strive to minimise the expense in A. Note that $\Psi(\tau)(A)$ will be negative at optimum, unless there is an arbitrage opportunity (which user could take, see below).

So far the objectives only concern one output. To generalise to multi-output plans, we can set a reference price $p \in \mathbb{R}_+^V$ and use it to scalarise the problem [5, §4.7.4].

For $a \in \mathbb{R}_+$, $\psi_0 \leq 0_V$, we define:

$$sor(p, a : A) = \begin{array}{ll} \max & \langle p, \Psi(\tau) \rangle \\ \text{with} & \Psi_-(\tau)(A) \leq a \\ & \Psi(\tau) \geq \psi_0 \end{array} \tag{5}$$

Here a feasible plan must invest no more than $a : A$, and respect an overall budget limit ψ_0. Hence there is always the zero plan.

3.2 Arbitrage

On to the price consistency problem:

$$arb(A) = \begin{array}{ll} \max & \Psi(\tau)(A) \\ \text{with} & \Psi(\tau) \geq 0 \end{array} \tag{6}$$

A feasible plan is any that results in a non-negative balance for all tokens. It is feasible as 0_E satisfies constraints. The optimal value will therefore also be non-negative. Of course the interesting question is qualitatively whether there is a non-zero solution, and quantitatively how to compute it. As above one can prove that any solution will activate the constraints $\Psi(\tau)(B) = 0$ for $B \neq A$.

As in the routing problem, one can scalarise the multi-output version of this problem by maximising $\langle p, \Psi(\tau) \rangle$ under the free-lunch constraint $\Psi(\tau) \geq 0_V$.

Assuming that Ψ satifies no-slicing:

Proposition 1 *The arbitrage problem (6) is idempotent.*

Proof: Let $\tau_1^\star$, $\tau_2^\star$ be two successive optimal plans. We compute:

$$\Psi(\tau_1^\star + \tau_2^\star) \geq_{nsl} \Psi(\tau_1^\star) + \tau_1^\star \cdot \Psi(\tau_2^\star) \geq 0$$

where recall $\tau \cdot \Psi$ is the profit function after the execution of τ. The sum $\tau_1^\star + \tau_2^\star$ represents the fused plan. The first inequality is the no slicing condition defined earlier (joining orders on the same edge is always better). By definition, both terms on the rhs of this inequality are positive, and, therefore, so is the lhs. In other words $\tau_1^\star + \tau_2^\star$ is feasible. It also follows that $\Psi(\tau_1^\star + \tau_2^\star) \geq \Psi(\tau_1^\star)$, and, therefore, by optimality of $\tau_1^\star$, it must be that $\tau_2^\star = 0$. □

In the differentiable case it is enough to look near the zero plan to detect non-zero arbitrage.

Proposition 2 *There is a non-zero solution to the arbitrage problem iff there exists $\epsilon \geq 0_E$, such that $J\Psi(0)(\epsilon) \geq 0$ with at least one coordinate $J\Psi(0)(\epsilon)(A) > 0$.*

Proof: The if part is clear. The only if part follows from the fact that Ψ is concave. To see this pick a τ (not necessarily optimal) such that $\Psi(\tau)(A) > 0$, and choose $t \in (0, 1)$:

$$\Psi(t\tau) = \Psi((1-t)0_E + t\tau) \geq (1-t)\Psi(0_E) + t\Psi(\tau) = t\Psi(\tau)$$

where we use the fact that Ψ is concave (in each argument) and $\Psi(0_E) = 0_V$.

By definition of the Jacobian:

$$\Psi(t\tau) = J\Psi(0)(t\tau) + o(\|t\|_1)$$

It follows that for $t > 0$ small enough, $J\Psi(0)(\epsilon)(A) > 0$, with $\epsilon = t\tau$. $\square$

The criterion implies that at least one $\epsilon_e > 0$. Keep in mind that the arbitrage may be very small (see lower bound examples later). The criterion says nothing about its magnitude; it merely gives a direction ϵ in the cone of plans in which to look for one.

Using the expression obtained earlier for $J(\Psi)$, we can rephrase the criterion as follows:

Corollary 1 *Problem (6) has a non-zero solution iff there is $\epsilon \in \mathbb{R}_+^E$ such that for all $A \in V$:*

$$\sum_{e \in t^{-1}(A)} f_e'(0)\epsilon_e \geq \sum_{e' \in s^{-1}(A)} \epsilon_{e'}$$

and for at least one A the inequality is strict.

For concrete price functions such as Uniswap's with derivatives at zero which are 0-homogenous in the reserves, a rescaling of these reserves by a positive coefficient leaves the criterion invariant.

3.3 Existence

Except for problem (5), problems considered so far have the following form:

$$\begin{array}{ll} \max & h(\Psi(\tau)) \\ \text{with} & \Psi(\tau) \geq \psi_0 \end{array} \tag{7}$$

with $h : \mathbb{R}^V \to \mathbb{R}$ a continuous function, $\psi_0 \leq 0_V$.

Proposition 3 *The feasible set $C = \{\tau \mid \Psi(\tau) \geq \psi_0\}$ of problem (7) is compact and non-empty in $\mathbb{R}_+^E$; therefore problem (7) has solutions.*

Proof:

First C is non-empty as 0_E is in C.

Second $C = \cap_A \Psi(\tau)(A)^{-1}[\psi_0(A), \infty)$ is closed in $\mathbb{R}_+^E$, as $\Psi(\tau)(A)$ is continuous.

Suppose C is not bounded. Pick a sequence $\tau_n \in C$ such that $\|\tau_n\|_\infty \geq n$, and $e_n \in E$ such that $\|\tau_n\|_\infty = \tau_n(e_n)$. As E is finite, there must a subsequence of e_n which is constant and equal to some e_0 with source A. Let τ_m' be the associated subsequence of plans. By construction $\tau_m'(e_0) \to \infty$.

For general reasons, we have:

$$\Psi(\tau)(A) \leq \sum_E f_e(\infty) - \tau(e_0)$$

hence $\Psi(\tau_m')(A) \to -\infty$ which contradicts the budget constraint $\Psi(\tau)(A) \geq \psi_0(A)$.

As $h \circ \Psi$ is continuous, the second point follows. $\square$

4 Lower bounds

The problems considered in the preceding section may have solutions, but the proof hardly tells us how to find them. In this section, we add new feasibility constraints and derive simpler and tractable subproblems which will give lower bounds to the original ones.

In the appendix we further specialise to Uniswap's price functions and obtain closed formulas.

4.1 Routing on independent paths

Let us return to the forward routing problem with source A and target B. Fix $(\phi_i;\ 0 \leq i < n)$ a family of independent paths in G from A to B with underlying edge and node sets $E' \subseteq E$, $V' \subseteq V$, and strictly concave price functions.

We restrict the forward A/B routing problem (2) by restricting plans to have their support included in E'. This subproblem is again convex, evidently. All nodes $C \in V' \setminus \{A, B\}$ have non-negative balance by definition, but we have noticed already that optimal solutions of the original problem satisfy $\Psi(\tau^\star)(C) = 0$ (Cs are intermediates). This leads us to an alternative and equivalent formulation of the subproblem:

$$sor(a : A, \phi_1, \ldots, \phi_n) := \begin{array}{ll} \max & \sum_i f_{\phi_i}(t_i a) \\ \text{with} & (t_i) \in \Delta_n \end{array} \tag{8}$$

with Δ_n the simplex of dimension $n-1$, where n is the number of support paths. The quantity t_i represents the fraction of the original budget a allocated to path ϕ_i.

Write $\hat{\Psi}_a(t) := \sum_i \phi_i(t_i a)$ for the new objective function.

For any convex combination $u + v = 1$, $u, v > 0$:

$$\begin{aligned} \hat{\Psi}_a(u(t_1, \ldots, t_n) + v(s_1, \ldots, s_n)) &= \textstyle\sum_i f_{\phi_i}((ut_i + vs_i)a) \\ &= \textstyle\sum_i f_{\phi_i}(ut_i a + vs_i a) \\ &\geq \textstyle\sum_i u f_{\phi_i}(t_i a) + \sum_i v f_{\phi_i}(s_i a) \\ &= u \textstyle\sum_i f_{\phi_i}(t_i a) + v \sum_i f_{\phi_i}(s_i a) \\ &= u\hat{\Psi}_a(t_1, \ldots, t_n) + v\hat{\Psi}_a(s_1, \ldots, s_n) \end{aligned}$$

hence $\hat{\Psi}_a$ is stricly concave, as the ϕ_is are.

We have proved that for any choice of a family of paths:

Proposition 4 *The restricted forward routing problem (8) has a unique solution $t^\star$; moreover, its optimum is a lower bound to that of the unrestricted forward routing problem (2).*

Not every plan can be expressed as a sum of independent paths. There seems to be a natural intermediate and possibly tractable subproblem, where one maximises over diagrams. This is a larger subproblem as is evident from the example Fig. 1 which is not a sum of independent paths (because of the last USDC/DAI leg). Extending this proposition to diagrams would improve the lower bound. However, it is unclear how to do this as diagrams (say with one source, and one sink) do not form a convex subset of the plans.

4.2 Arbitraging simple cycles

Let us return to the arbitrage problem (6) with source A. Similarly to the routing problem, we restrict the arbitrage one. Specifically, we ask for plans which are supported by a given simple cycle going through A. The restricted problem is still convex as are all subproblems based on restriction on the support. Also, it is clear that the original problem has solutions that are not supported by a cycle, so this approach will only provide lower bounds, in general.

Let ϕ be a directed cycle in G. One needs only to direct edges in the cycle all in one direction or the other. As constraints will be active for solutions of the subproblem, we have an alternative and equivalent formulation:

$$arb(a : A, \phi) := \begin{array}{ll} \max & f_\phi(a) - a \\ \text{with} & a \in \mathbb{R}_+ \end{array} \tag{9}$$

Note that $f_\phi(a) - a$ is the profit function associated to the unique plan τ with $\tau(e) = a$ for the only edge e in ϕ with source A, which induces a zero profit at every other node of the cycle.

The only optimisation variable is now a. As f_ϕ is a strictly concave price function:

Proposition 5 *The cyclic arb subproblem has a unique solution (possibly trivial).*

The arbitrage criterion simplifies to:

Proposition 6 *The cyclic arbitrage problem (9) has a non-zero solution iff:*

$$f'_\phi(0) = \prod_{e\in\phi} f'_e(0) > 1$$

Using the argument of the proof of Prop. 2, one can show that the set of as such that $f_\phi(a) \geq a$ is a compact interval of the form $[0, a_0]$. The solution is somewhere in between and is non-trivial iff $0 < a_0$.

5 Cyclic arbitrage: the Uniswap case

Consider again the Uniswap graph, where nodes are ERC20 tokens, edges are pairs of reserve pools with fees $0 \leq 1 - \gamma \ll 1$ (possibly different in each direction). As above the reserves of an A/B edge are written $[A]$, $[B]$ and we define the ratio $\rho_{AB} = [B]/[A]$ -so that the marginal price of A in B is $\gamma_{AB}\rho_{AB}$.

5.1 Closed formulas for arbitrage

To simplify notations we consider triangular cycles on tokens A, B and C.

We have a profitable triangular arbitrage if:

$$a_0 \xrightarrow{Swap} b \xrightarrow{Swap} c \xrightarrow{Swap} a_1, \text{ with } a_0 < a_1$$

We can specialise the arbitrage criterion of the main text as follows:

Lemma 1 ***(cyclic arbitrage):*** *A triangular cycle is arbitrage-free iff:*

$$\gamma_{AB}\gamma_{BC}\gamma_{CA} \leq \rho_{AB}\rho_{BC}\rho_{CA} \leq (\gamma_{AB}\gamma_{BC}\gamma_{CA})^{-1} \tag{10}$$

Although we have already proven this result, it is instructive to redo the proof in this special case, as composition of Uniswap price functions can be computed explicitely. Specifically, we have:

$$a_1 = \frac{\rho_{CA}}{\frac{\rho_{CB}\rho_{BA}}{\gamma_{CA}\gamma_{BC}\gamma_{AB}}\frac{1}{a_0} + \frac{\rho_{CB}\rho_{BA}}{\gamma_{CA}\gamma_{BC}[A]_{AB}} + \frac{\rho_{CB}}{\gamma_{CA}[B]_{BC}} + \frac{1}{[C]_{CA}}}$$

To have a non-zero arbitrage we need to have $a_0 < a_1$:

$$a_1 = \frac{\rho_{CA}}{\frac{\rho_{CB}\rho_{BA}}{\gamma_{CA}\gamma_{BC}\gamma_{AB}}\frac{1}{a_0} + \frac{\rho_{CB}\rho_{BA}}{\gamma_{CA}\gamma_{BC}[A]_{AB}} + \frac{\rho_{CB}}{\gamma_{CA}[B]_{BC}} + \frac{1}{[C]_{CA}}} > a_0$$

Equivalently:

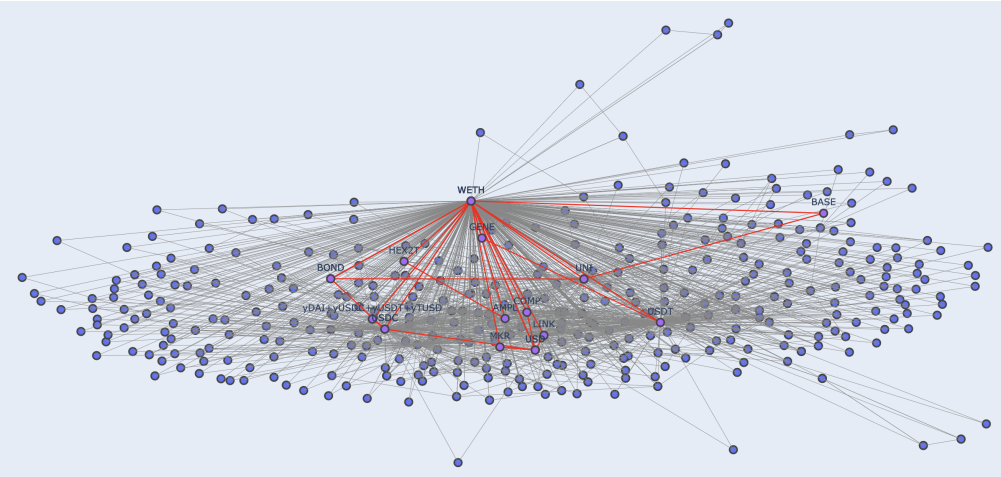
$$0 < a_0 < \frac{\gamma_{AB}\gamma_{BC}\gamma_{CA}\rho_{AB}\rho_{BC}\rho_{CA} - 1}{\frac{\gamma_{AB}}{[A]_{AB}} + \frac{\gamma_{BC}\gamma_{AB}\rho_{AB}}{[B]_{BC}} + \frac{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}}{[C]_{CA}}}$$

which can always be achieved by choosing a_0 small enough, provided $\gamma_{AB}\gamma_{BC}\gamma_{CA}\rho_{AB}\rho_{BC}\rho_{CA} > 1$, and the conclusion follows.□

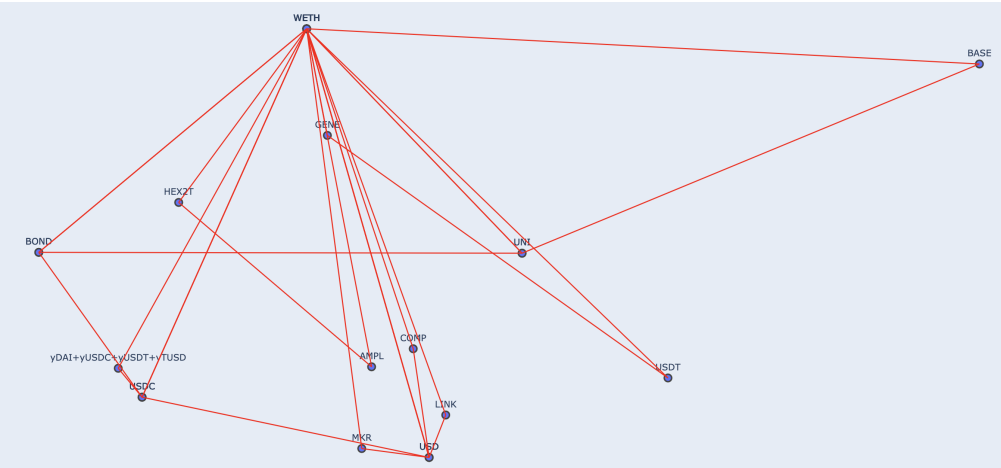
One sees that arbitrage can only exist in one orientation of the cycle.

In case of no fees, ($\gamma_{AB} = \gamma_{BC} = \gamma_{CA} = 1$), in order for the triangular cycle to be arbitrage-free, we need to have $\rho_{AB}\rho_{BC}\rho_{CA} = 1$, meaning that the product of marginal prices should be equal to 1. In the presence of fees the no-arbitrage zone is 'thicker' so to speak.

The explicit calculation shows that the arbitrage condition itself is homogeneous (invariant under a rescaling of the reserves). Next, we compute the max extractable profit and will se that the actual reserve sizes do matter.



(a) In red arbitrageable cycles among all existing cycles.



(b) Isolated arbitrageable cycles.

Fig. 2: Example of arbitrageable cycles in Uniswap.

Proposition 7 *In case of the existence of a triangular arbitrage, we obtain the following:*

– *The **optimal input** a_0^* that maximises the arbitrage profit is:*

$$\max_{a_0}(a_1 - a_0) = \max_{a_0}\left(\frac{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}\rho_{CA}a_0}{(\frac{\gamma_{AB}}{[A]_{AB}} + \frac{\gamma_{BC}\gamma_{AB}\rho_{AB}}{[B]_{BC}} + \frac{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}}{[C]_{CA}})a_0 + 1} - a_0\right)$$

$$a_0^* = \frac{\sqrt{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}\rho_{CA}} - 1}{\frac{\gamma_{AB}}{[A]_{AB}} + \frac{\gamma_{BC}\gamma_{AB}\rho_{AB}}{[B]_{BC}} + \frac{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}}{[C]_{CA}}} \tag{11}$$

– *The **optimal output** a_1^* that maximises the arbitrage profit obtained from this arbitrage operation is:*

$$a_1^* = \frac{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}\rho_{CA} - \sqrt{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}\rho_{CA}}}{\frac{\gamma_{AB}}{[A]_{AB}} + \frac{\gamma_{BC}\gamma_{AB}\rho_{AB}}{[B]_{BC}} + \frac{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}}{[C]_{CA}}} \tag{12}$$

– *The **maximum profit** obtained from this arbitrage operation is:*

$$Profit^* = a_1^* - a_0^* = \frac{(\sqrt{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}\rho_{CA}} - 1)^2}{\frac{\gamma_{AB}}{[A]_{AB}} + \frac{\gamma_{BC}\gamma_{AB}\rho_{AB}}{[B]_{BC}} + \frac{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}}{[C]_{CA}}} \tag{13}$$

The above results can be obtained by straightforward computations. There are few things worth observing. If one rescales each reserves by a coefficient $\lambda \geq 0$, the arbitrage profit is also multiplied by the same coefficient. In other words the max arbitrage profit is homogeneous of degree 1 in the size of the reserves.

Closed formulas for max profit for Uniswap price functions give explicit lower bounds on optimal values for the corresponding original problems. One would also think that lifting solutions of the subproblems may give good initialisers to the original ones.

If there is an arbitrage, the arbitrageur can choose to start from any origin of the cycle. One may wonder whether the relative variation of the arbitrageur's portfolio depends on this choice.

Proposition 8 *If the external prices of the tokens present in the cycle do not change before and after the execution, then the maximum profit is independent from the origin.*

Proof: We position ourselves from an arbitrageur perspective. We suppose as above that the cycle is a triangle to simplify the notations. We also suppose that the arbitrageur possesses in her/his portfolio a sufficient quantities of tokens A, B and C, greater than a_0^*, b_0^* and c_0^* respectively. Initially, we also assume the existence of an external liquid market where the arbitrageur can exchange its tokens against a reference token R (it can be euros or a stable coin for example).
We suppose the existence of arbitrage, let X be one of A, B, C, x_0^* the optimal input quantity that maximises the arbitrage profit and p_R^X the price of X in R.
We can express the value in R of the portfolio part containing X before and after the arbitrage execution (V_{0R}^X and V_{1R}^X respectively), as follows:

$$V_{R_0}^X = x_0^* p_{R_0}^X \text{ and } V_{R_1}^X = x_1^* p_{R_1}^X$$

The percentage variation in the value of the portfolio part containing X is given by:

$$\Delta = \frac{V_{1R}^X - V_{0R}^X}{V_{0R}^X} = \frac{x_1^*}{x_0^*}\pi_{p_R^X} - 1$$

where $\pi_{p_R^X} := p_{R_1}^X / p_{R_0}^X$ measures the change of price of X before and after the arbitrage.

From the preceding proposition we have:

$$\frac{x_1^*}{x_0^*} = \frac{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}\rho_{CA} - \sqrt{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}\rho_{CA}}}{\sqrt{\gamma_{CA}\gamma_{BC}\gamma_{AB}\rho_{AB}\rho_{BC}\rho_{CA}} - 1}$$

Hence we can derive explicitly the relative change in wealth of the optimal arbitrageur:

$$\Delta = \sqrt{\gamma_{AB}\gamma_{BC}\gamma_{CA}\rho_{AB}\rho_{BC}\rho_{CA}}\pi_{p_R^X} - 1$$

which is positive if $p_{R_1}^X \geq p_{R_0}^X$, and, if prices stay the same, is indeed independent of the choice of the origin X. □

5.2 Some empirical results

As of the 15th of December 2020, Uniswap contained 26139 pairs (WETH being connected to more than 12365 tokens) and more than 965 triangular cycles. We analyzed the data obtained from Etherum blockchain from block 11299400 to 11360599 (from 21/11/2020 to 30/11/2020) for the 200 most liquid pairs. We selected 11 triangular cycles that generated the maximum profits per block during this period on Uniswap:

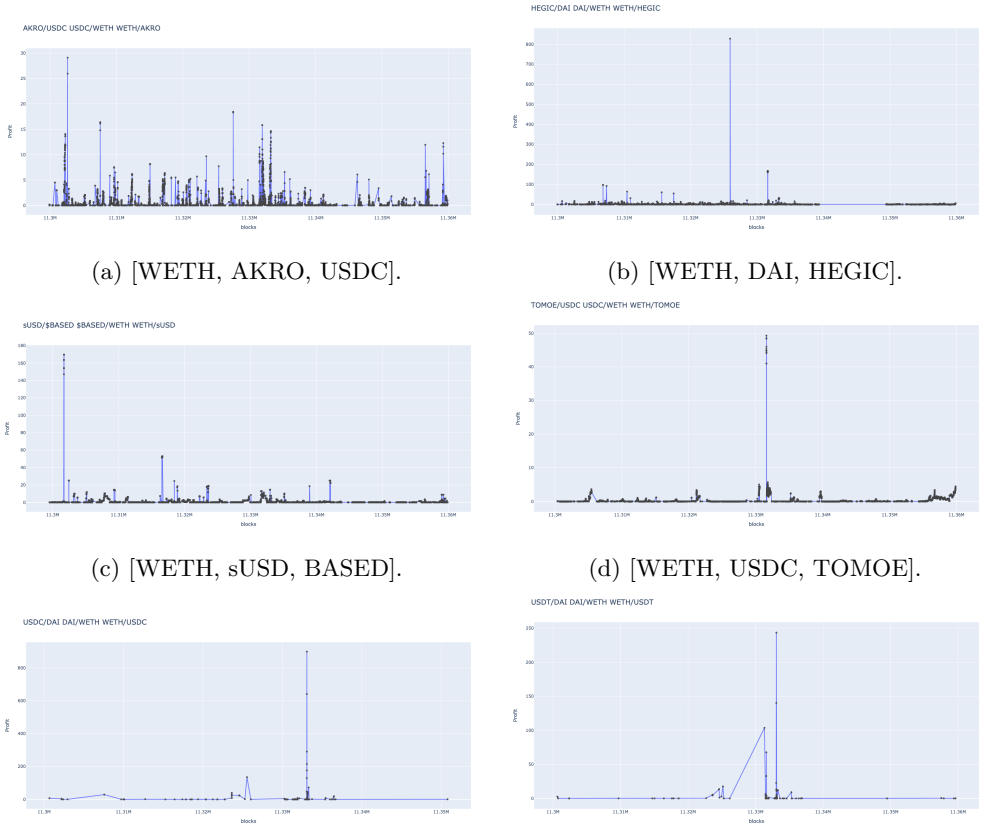
[WETH, AKRO, USDC]
[WETH, DAI, HEGIC]
[WETH, sUSD, BASED]
[WETH, USDC, TOMOE]
[WETH, DAI, USDC]
[WETH, DAI, USDT]
[USDT, USDC, TOMOE]
[WETH, USDT, TOMOE]
[WETH, USDT, USDC]
[WETH, WBTC, USDC]
[WETH, USDT, YFV]

For each block we look for arbitrageable triangular cycles. Once detected, we compute the maximum profit per cycle and per block during the whole period. We plot the maximum profit (measured in USD) per block for each of the 11 selected triangular cycles. One can see that some of the optimal arbitrage profits disappear instantly (ie have a one block life time). Others last longer. A key difference with Ref. [7] is coverage. Their data covers Uniswapv2 for a much longer larger period of time and does not look for prediction of cyclic arbitrage on restricted set of tokens, as we do, but for detection thereof. And indeed, their findings show larger actual profits than the potential ones which our small scale data experiment predicts.

6 Conclusions

We have represented a class of global routing problems on networks of money markets as convex optimisation problems on a suitable domain of transfer plans. We have shown that feasible problems in this class have solutions (maybe not unique). We have also built tractable subproblems which allow one to find efficiently lower bounds to the original problems.

Preliminary data analysis shows the presence of non-trivial cyclic arbitrage opportunities (see appendix), and *a fortiori* general ones. There is also evidence of substantial gains from non-trivial



(a) [WETH, AKRO, USDC].

(b) [WETH, DAI, HEGIC].

(c) [WETH, sUSD, BASED].

(d) [WETH, USDC, TOMOE].

(e) [WETH, DAI, USDC].

(f) [WETH, DAI, USDT].

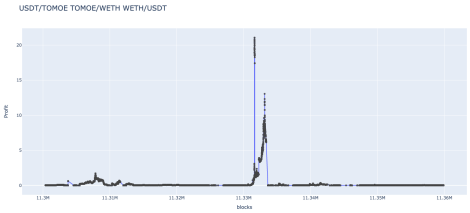
Fig. 3: Maximum profit (USD) per cycle and per block.

routing (Fig. 1). So there is undeniably a practical interest in solving exactly or approximately, and efficiently, these global problems. All the more so if liquidity continues to fragment in DeFi, increasing the complexity of optimal routing. Substantial liquidity migrations on DeFi's money markets have happened, and it is unclear if and when liquidity will aggregate.

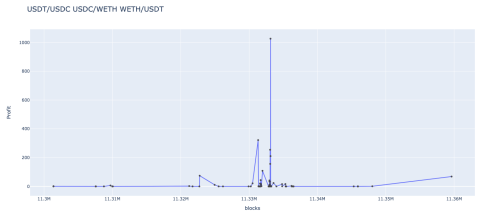
One limitation of our approach is that the problems before and after executing a given plan are related but here we do not exploit that information. So there is room for designing on-line versions of the above problems of which the amortised cost could be vastly improved - compared to resolving anew the problem at each update. A minimal way to exploit that relation would be to use a former optimum as initial data for a new gradient descent on the updated problem.

Also we have ignored gas fees as well as the uncertainty generated by the asynchrony inherent on blockchain-based smart contracts: the state of the world at the time the problem is solved, may be very different from the state at the time the corresponding instructions are executed. It would be interesting to include both aspects of this uncertainty (gas costs and asynchrony) in the problem for more robustness.

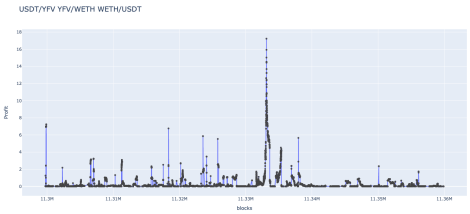
(a) [USDT, USDC, TOMOE].



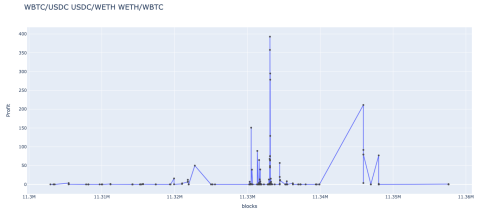
(b) [WETH, USDT, TOMOE].



(c) [WETH, USDT, USDC].



(d) [WETH, USDT, YFV].



(e) [WETH, WBTC, USDC].

Fig. 4: Maximum profit (USD) per cycle and per block.

References

1. Frédéric Abergel and Aymen Jedidi. A mathematical approach to order book modeling. *International Journal of Theoretical and Applied Finance*, 16(05):1350025, 2013.
2. Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core, March 2020.
3. Guillermo Angeris, Hsien-Tang Kao, Rei Chiang, Charlie Noyes, and Tarun Chitra. An analysis of uniswap markets. *Cryptoeconomic Systems Journal*, 2019.
4. Angeris, Guillermo and Chitra, Tarun. Improved price oracles: Constant function market makers. *arXiv preprint arXiv:2003.10001*, June 2020.
5. Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
6. Bas Lemmens and Roger Nussbaum. *Nonlinear Perron-Frobenius Theory*, volume 189. Cambridge University Press, 2012.
7. Ye et al. Wang. Cyclic arbitrage in decentralized exchange markets. Presented at the 1st workshop on Decentralized Finance (DeFi), 2021.

A The Uniswap graph

Fig. 5 proposes a view of the global Uniswap money market [2] restricted the top 64 ERC20 tokens and their 283 Uniswap pairs. It is worth stressing that the actual graph of interest for routing and arbitrage is far more complex as it includes many more automated market makers (sometimes with considerable liquidity) as well as other forms of decentralised exchanges.

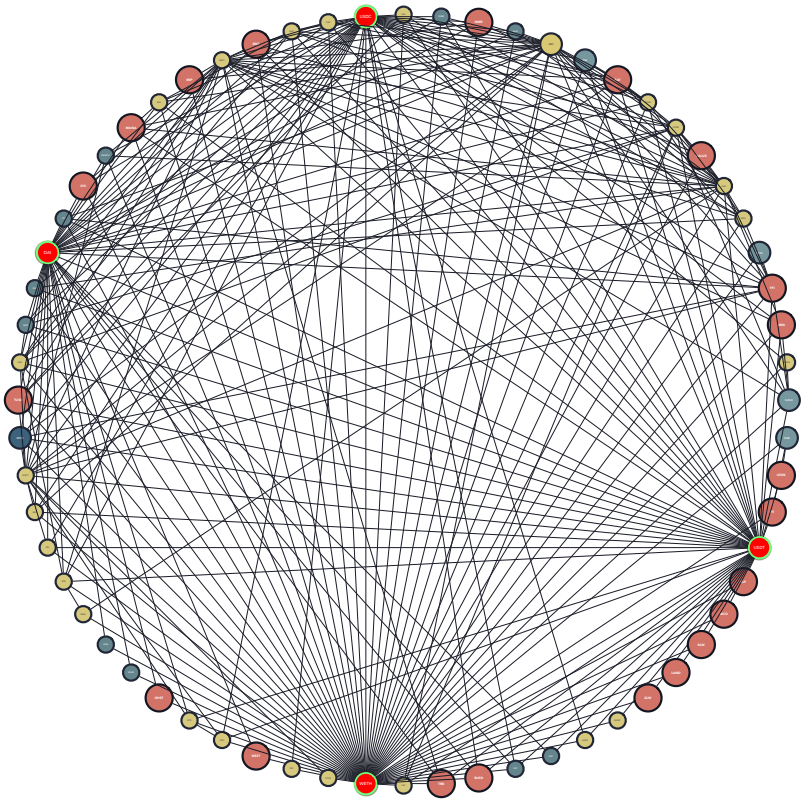


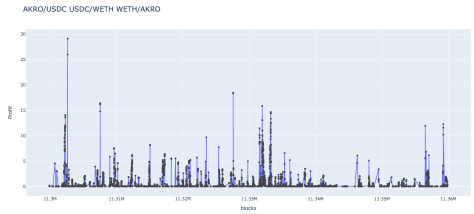
Fig. 5: The top 64 ERC20s (in market cap) and their pairs on Uniswapv2: there is an edge between two nodes if there exists a Uniswap pair between them (Nov 2020).

B Some empirical results

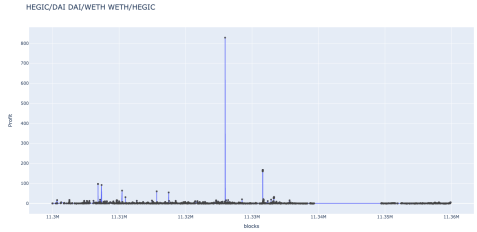
As of the 15th of December 2020, Uniswap contained 26139 pairs (WETH being connected to more than 12365 tokens) and more than 965 triangular cycles. We analyzed the data obtained from Etherum blockchain from block 11299400 to 11360599 (from 21/11/2020 to 30/11/2020) for the 200 most liquid pairs. We selected 11 triangular cycles that generated the maximum profits per block during this period on Uniswap:

[WETH, AKRO, USDC]
[WETH, DAI, HEGIC]
[WETH, sUSD, BASED]
[WETH, USDC, TOMOE]
[WETH, DAI, USDC]
[WETH, DAI, USDT]
[USDT, USDC, TOMOE]
[WETH, USDT, TOMOE]
[WETH, USDT, USDC]
[WETH, WBTC, USDC]
[WETH, USDT, YFV]

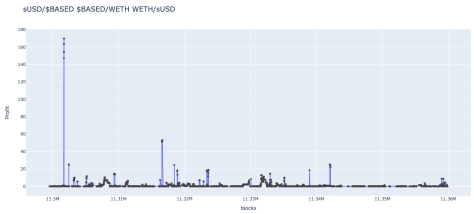
For each block we look for arbitrageable triangular cycles. Once detected, we compute the maximum profit per cycle and per block during the whole period. We plot in Figure 7 the maximum profit (measured in USD) per block for each of the 11 selected triangular cycles. One can see that some of the optimal arbitrage profits disappear instantly (ie have a one block life time). Others last longer.



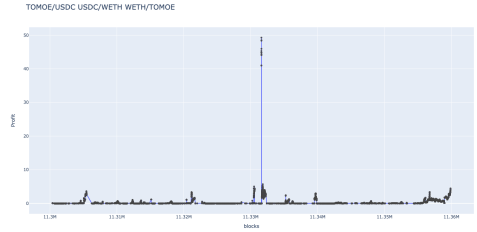
(a) [WETH, AKRO, USDC].



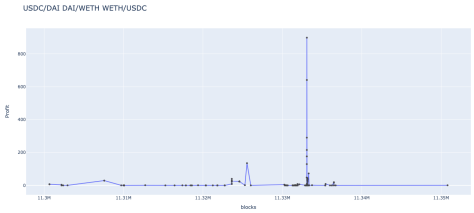
(b) [WETH, DAI, HEGIC].



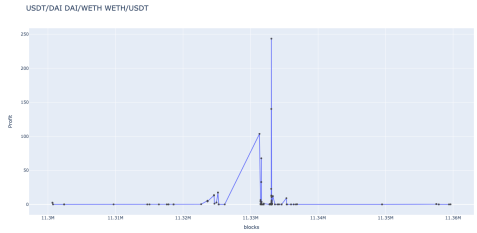
(c) [WETH, sUSD, BASED].



(d) [WETH, USDC, TOMOE].

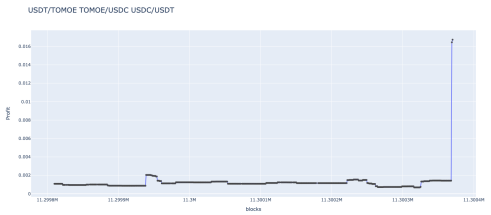


(e) [WETH, DAI, USDC].

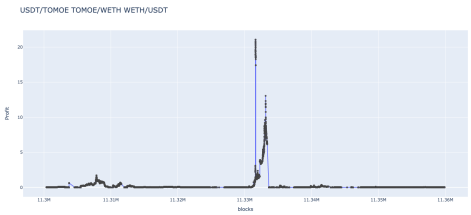


(f) [WETH, DAI, USDT].

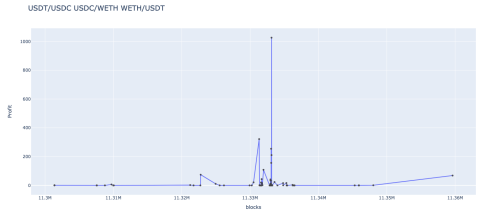
Fig. 6: Maximum profit (USD) per cycle and per block.



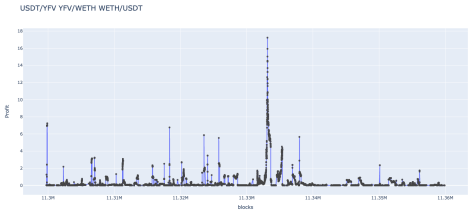
(a) [USDT, USDC, TOMOE].



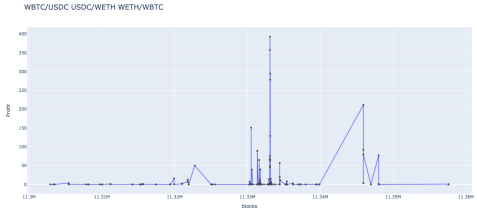
(b) [WETH, USDT, TOMOE].



(c) [WETH, USDT, USDC].



(d) [WETH, USDT, YFV].



(e) [WETH, WBTC, USDC].

Fig. 7: Maximum profit (USD) per cycle and per block.