Proportioning Multipath Trades with Maximum Gain to Destination and Yield to Destination

A. Carreira and P. Bhardwaj

Context

This project was funded by the Uniswap Grants program. Source code is available here: `https://github.com/valve-finance/uniswap_routing`.

The original scope of work was to identify areas of improvement for the existing Uniswap V2 on-chain router. An offline tool sourcing data from the Graph Protocol Uniswap V2 SubGraph [1] was constructed to create and update an undirected graph for analyzing the network of Uniswap pairs. From this and discussions with Uniswap Labs, work scope expanded to an online service that would provide routes between arbitrarily specified tokens, comparing them to the results of the on-chain router. The project went further adding various routing features, including multipath trades, configurable report generation and visualization capabilities. Much of the extended scope was completed in very short order. The mulitpath route method presented herein was quickly constructed as a placeholder for later efforts related to the extended scope work.

Alternate methods exist which are more sophisticated and appropriate for production systems. Uniswap has since upgraded their routing system and it is recommended that readers familiarize themselves with that effort [2].

More background on this project can be found in [3] and [4].

1 Introduction

This document describes a method for proportioning multipath trades. The method was created as part of a larger project exploring alternatives to the Uniswap V2 on-chain router. The alternative sought to provide improved solutions to the on-chain router by incorporating continuously updated data driven routes, free from the on-chain router's restrictions, in addition to providing complete multipath routes to reduce slippage.

2 Background

Uniswap V2 is a Constant Product Automated Market Maker (CPAMM) facilitating the trading of assets on the Ethereum blockchain. A Uniswap V2 pair stores pooled reserves of two assets, providing liquidity for those two assets and ensuring that reserves do not decrease by maintaining an invariant given by the constant product formula. [5]

2.1 Swaps

A swap can be defined as exchanging one asset of a pair for the other as illustrated in Figure 1.

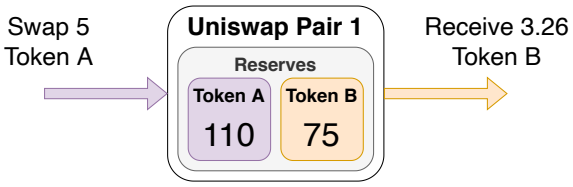

Figure 1: A swap of 5 Token A for 3.26 Token B. This is also a "1-hop" trade.

Note that this entire discussion ignores fees charged by a pool or pair for the sake of simplicity. For a more detailed example including fees, consult [6]

2.1.1 Swap Price

The price of assets in a pair are measured against the reserves of the other asset in the pair. For example consider a pair containing reserves of 100 Token A and 1000 Token B. The price of Token A is given by Equation 1, the price or simple exchange equation:

$$Price_A = \frac{Reserve_B}{Reserve_A} = \frac{1000}{100} = 10 \;\; Token\, B / Token\, A \tag{1}$$

However, this price is different than the effective price a swap user will receive because it ignores the constant product equation relating the reserves of the pair to an invariant. The invariant or constant product, k, is given in Equation 2:

$$k = Reserve_A \cdot Reserve_B \tag{2}$$

When the amount being swapped is much less than the reserve amount, then Equation 1 is reasonably accurate, however as the amount being swapped approaches and exceeds the reserve amount, slippage affects the actual price received in a swap.

2.1.2 Swap Slippage

The actual amount of Token B received, $Received_B$, in a swap of Token A, $Swapped_A$, is given by the swap equation, Equation 3:

$$Received_B = Reserve_B - \frac{k}{Reserve_A + Swapped_A} \tag{3}$$

The price equation can then be used with the swap equation to compute the amount of slippage, measured in tokens, for a trade. Equation 4 computes the *Slippage* measured in Token B of selling $Swapped_A$ Token A to a pair. The values $Expected_B$ and $Received_B$ are computed from Equations 1 and 3, respectively, as shown in the substitutions below:

$$\begin{aligned} Slippage &= Expected_B - Received_B \\ &= (Swapped_A \cdot Price_A) - \left(Reserve_B - \frac{k}{Reserve_A + Swapped_A}\right) \\ &= \left(Swapped_A \cdot \frac{Reserve_B}{Reserve_A}\right) - \left(Reserve_B - \frac{k}{Reserve_A + Swapped_A}\right) \\ &= \frac{(Swapped_A)^2 \cdot Reserve_B}{(Reserve_A + Swapped_A) \cdot Reserve_A} \end{aligned} \tag{4}$$

2.2 Trades

A trade can be defined as one or more swaps facilitating an exchange of one asset for another asset as shown in Figure 2; the number of swaps comprising a trade are sometimes referred to as "hops".

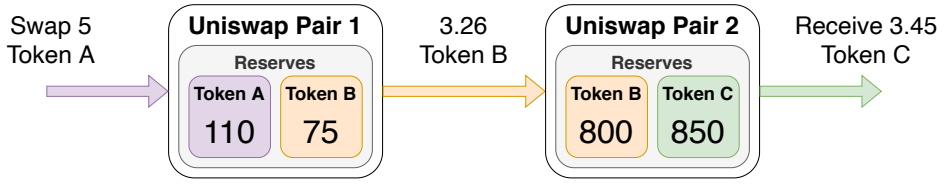

Figure 2: A trade of 5 Token A for 3.45 Token C. This "2-hop" trade is comprised of 2 swaps, one from Token A to Token B and another from Token B to Token C.

A trade is a cascade of swaps and Equations 1 and 3, for computing the price and amount received, respectively, can be applied sequentially to a series of swaps in a trade

to determine the overall price and amount received. For the trade illustrated in Figure 2, the overall trade price is given by applying Equation 5. The resulting value can then be used to calculate the ideal amount expected to be received using Equation 6.

$$\begin{aligned} Price_C &= Price_{A_{Pair1}} \cdot Price_{B_{Pair2}} \\ &= \frac{Reserve_{B_{Pair1}}}{Reserve_{A_{Pair1}}} \cdot \frac{Reserve_{C_{Pair2}}}{Reserve_{B_{Pair2}}} \\ &= \cdot \frac{75}{110} \cdot \frac{850}{800} \\ &= 0.72... \ \ Token\, C/Token\, A \end{aligned} \tag{5}$$

$$\begin{aligned} Expected_C &= Price_C \cdot Swapped_A \\ &= 0.72 \cdot 5 \\ &= 3.62... \ Token\, C \end{aligned} \tag{6}$$

The actual amount received due to the invariant in both pairs can be calculated as shown in Equation 7:

$$\begin{aligned} Received_C &= Reserve_{C_{Pair2}} - \frac{k_{Pair2}}{Reserve_{B_{Pair2}} + Received_{B_{Pair1}}} \\ &= Reserve_{C_{Pair2}} - \frac{k_{Pair2}}{Reserve_{B_{Pair2}} + \left(Reserve_{B_{Pair1}} - \frac{k_{Pair1}}{Reserve_{B_{Pair1}} + Swapped_A}\right)} \\ &= 850 - \frac{800 \cdot 850}{800 + \left(75 - \frac{110 \cdot 75}{110+5}\right)} \\ &= 850 - \frac{344000}{800 + (3.26...)} \\ &= 3.45... \end{aligned} \tag{7}$$

Using the results from applying Equations 6 and 7 to compute $Expected_C$ and $Received_C$, respectively, the overall Slippage can be computed by applying Equation 4 below to calculate the amount of slippage in Token C:

$$
\begin{aligned}
Slippage &= Expected_C - Received_C \\
&= 3.62 - 3.45 \\
&= 0.17... \ Token\, C
\end{aligned}
$$

The same methods of computing expected, received and slippage values can be applied to trades consisting of any number of pairs cascaded sequentially.

2.3 Multipath Trades

A multipath trade is one that takes two or more different paths from asset A to asset B as shown in Figure 3. The paths can be as simple as a single swap or increase in complexity to multiple hop trades.

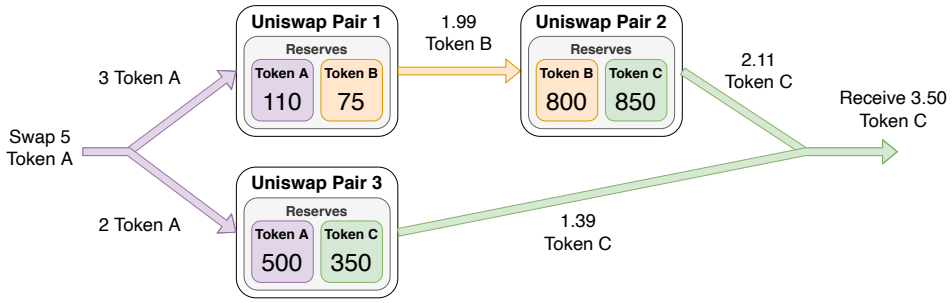

Figure 3: A trade of 5 Token A for 3.45 Token C. This "2-hop" trade is comprised of 2 swaps, one from Token A to Token B and another from Token B to Token C.

The calculations for the expected, received and slippage amounts can be applied to each individual path in a multipath trade and then combined using superposition to express the overall expected, received and slippage amounts for a multipath trade. This is visualized in Figure 3 with the values annotated on each segment of the individual trades and combined into overall values at the start and end of the multipath trade (5 Token A and 3.50 Token C, respectively).

3 Considerations for Multipath Trades

The 3.45 Token C received in the trade depicted in Figure 2 is less than the 3.50 Token C received in the multipath trade of Figure 3, despite both trades starting by swapping 5 Token A. This is because splitting the trade into two paths reduces the loss due to slippage

by increasing the amount of liquidity available for the trade, demonstrating the motivation and advantage of multipath trades.

To construct a multipath trade, a number of important considerations must be made, including operating constraints. The scope of this project was the exploration of data-driven routing solutions that were not required to be computed on-chain. This eliminates constraints such as the number of multiroute paths considered etc.–constraints that might force design decisions for a router or routing contract implemented on chain like [2]. The reason to constrain an on-chain router is that the gas used to read and write data to and from the blockchain translates into additional costs for the users of the router.

3.1 Gas

However, gas is a consideration within the context of this project because each path in a multipath trade imposes additional gas use. A simplistic way to model the gas used is to assign a proportional amount of gas used to each multipath trade path based on the number of hops in the path. For example, in Figure 1, gas used could be 1 in the simplistic model, while in Figure 2, gas use could be 2 in the simplistic model. The numbers arise from the number of hops. Applying this model to the multipath trade of Figure 3, the gas use for the overall trade would be 3. Future work in this project would have commenced with the simplistic model described here, evolving to a more accurate model based on real-world gas use figures and predicition that would be scaled according to trade size.

Gas is not considered in the discussion below.

3.2 The Proportioning Optimization Problem

How much Token A should be sent to Uniswap Pair 1 versus Uniswap Pair 3 in Figure 3 to get the most Token C?

This question necessitates the formulation of an optimization problem to produce the best solution. The optimization problem depends on the slippage of each individual pair in the multipath trade, the current price offered by each pair for the swap, and the amount of gas used to perform each path of the multipath route. Ignoring gas, Equation 3 and its application in Equation 7 can be used to provide a general equation for calculating the amount received in an N-path multipath trade in Equation 8.

$$\begin{aligned} Received_{Total} = & Recieved_{Path1}(SwapAmount_{Path1}) + \\ & Received_{Path2}(SwapAmount_{Path2}) + ... \\ & Received_{PathN}(SwapAmount_{PathN}) \end{aligned} \tag{8}$$

Optimizing individual values of each path's *SwapAmount*, also referred to as proportioning the swap, will yield the best returns for a specific total swap amount.

3.2.1 Two Path Example

Solving this problem for the multipath trade of Figure 3 is trivial because it is only two paths. Equation 8, the total amount received for a swap can be written with each path's swap amount expressed in terms of the variable x for the example in Figure 3:

$$\begin{aligned} Received_{Total} = Recieved_{Path1}(x) + \\ Received_{Path2}(SwapAmount - x) \end{aligned} \tag{9}$$
$$Where: 0 \leq x \leq SwapAmount, x \in \mathbb{R}$$

Equation 9 can then be plotted with the Token C received on the vertical axis commensurate to amount of Token A passed through path 1, "Pair 1 - Pair 2", as variable x, the horizontal axis. This is shown in Figure 4. The total swap amount is 5 Token A. The benefit of the multipath trade shown is slight because the swap size is small relative to the pair reserves. Still, the convex curvature is visible and the point highlighted at $x = 3$ shows the larger 3.50 Token A received over the 3.46 for the single path trade in Figure 2.

The benefit of a multipath trade becomes more apparent as the trade amount approaches the liquidity of Token A available. Consider the same pairs of the previous example, but with a total multipath trade size of 500 Token A. The different amounts of Token C received from varying the proportion across the two paths is shown in Figure 5.

Optimal proportioning for the two path scenario can be found using standard techniques for identifying the maximum of a two dimensional continuous function within an interval.

3.2.2 More Than Two Paths

The complexity of finding an optimal solution increases when more paths are introduced to the trade. Furthermore the number of paths that should be considered in a trade also needs to be optimized against diminishing returns due to increased gas use. Equation 10 shows the amount received by proportioning the *SwapAmount* among three paths using the variables α, β, and χ.

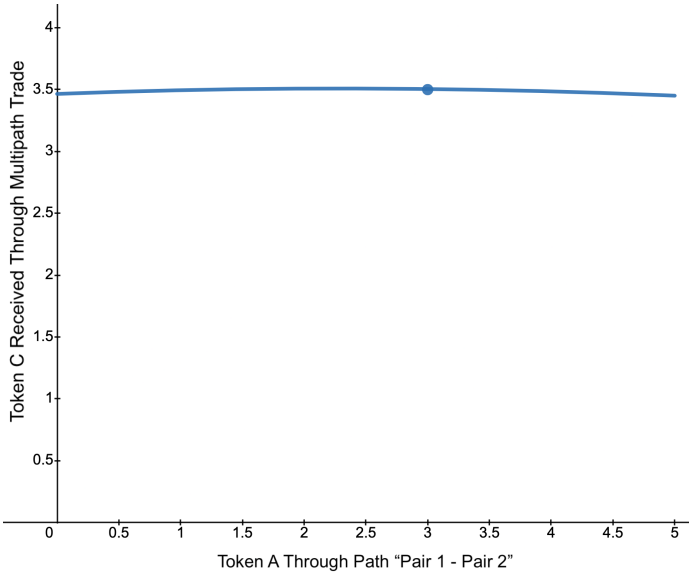

Figure 4: Token C received for a swap of 5 Token A in the multipath trade depicted in Figure 3 when the proportioning is varied across both paths.

$$
\begin{aligned}
Received_{Total} = & \, Recieved_{Path1}(\alpha \cdot SwapAmount) + \\
& Received_{Path2}(\beta \cdot SwapAmount) + \\
& Received_{Path3}(\chi \cdot SwapAmount) \qquad (10) \\
Where : & \, \alpha + \beta + \chi = 1 \\
& 0 \leq \alpha \leq 1, \alpha \in \mathbb{R} \\
& 0 \leq \beta \leq 1, \beta \in \mathbb{R} \\
& 0 \leq \chi \leq 1, \chi \in \mathbb{R}
\end{aligned}
$$

4 An Initial Multipath Trade Solution

As mentioned in the section on context, a placeholder solution for the optimization problem introduced above was used due to schedule constraints. The placeholder facilitated construction of visualization, data collection, and processing infrastructure–while demonstrably improving trade results over single path trades in many situations.

This section describes the placeholder solution as used in the offline router. The solution would benefit from a more appropriate optimization algorithm for this family of problems from the existing literature; a task planned for future phases of the project.

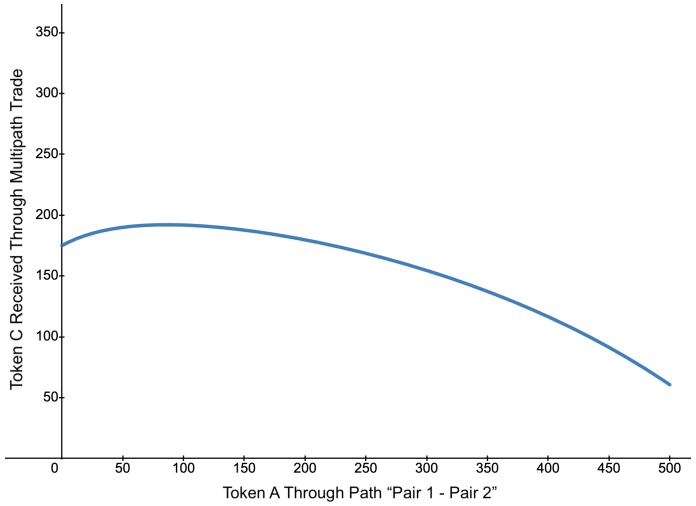

Figure 5: Token C received for a total swap of 500 Token A in the multipath trade depicted in Figure 3 when the proportioning is varied across both paths.

4.1 Generating Routes

A routing service with a continuously updated undirected graph modeling the Uniswap V2 network of pairs was used to construct a collection of possible routes between a specified token source and destination. The Uniswap V2 network model was updated using data from the Graph Protocol subgraph [1] in this project; a production system would have to source live data from the Ethereum network memory pool to provide the most accurate results.

For each route returned by the router, the network data and Uniswap V2 SDK was used to compute the slippage (also referred to as "impact") and number of output tokens resulting for every pair in a route for the overall trade amount. The calculated impact and output tokens were then used to annotate each segment between swaps of a route with calculated Maximum Gain To Destination and Yield To Destination values, as discussed in the following sections.

4.1.1 Maximum Gain To Destination (MGTD)

The slippage for every pair calculated during route generation is a worst case figure because it uses the desired trade input amount instead of a smaller proportioned amount. This is useful to understand where slippage will impact the trade most severely if it is to be proportioned amongst multiple routes; it is also useful for filtering out routes that would be undesirable paths in a multipath trade.

In a trade with multiple swaps cascaded, it is not possible to directly apply the slippage or impact of individual pairs conveniently to evaluate the the overall effect the route will have on the trade input amount. Taking an analogy from electrical engineering or signal processing, this effect can be framed as "gain", analogous to the effect of an amplifier on a signal. To determine the gain of a multiple swap trade, the impact or slippage of each swap can be thought of as loss and when converted to a relative value between zero and unity is easily converted to gain. Equation 11 shows how to compute relative slippage for a swap, Equation 12 the gain, and Equation 13 shows how to calculate the MGTD of an N swap trade:

$$Slippage_{Relative} = \frac{Expected - Received}{Expected} \tag{11}$$

$$Gain = 1 - Slippage_{Relative} \tag{12}$$

$$MGTD = \prod_{i=1}^{N} Gain_i \tag{13}$$

4.1.2 Yield To Destination (YTD)

MGTD proved inadequate as it did not consider price discrepancies between different pairs comprising the routes found in route generation. Specifically some routes were determined to provide poor exchange rates because of the state of their reserves. To identify and filter these routes out, another metric, YTD, was introduced. YTD is the ratio of tokens output from a route to the ratio of tokens input to a route at a particular segment of the route, as calculated as in Equation 14:

$$YTD = \frac{Tokens_{in}}{Tokens_{out}} \tag{14}$$

4.2 Modelling Multipath Routes

Generated routes, after annotation with MGTD and YTD, are used to construct a tree structure used in proportioning the multipath trade. Figure 6 shows the multipath trade of Figure 3 annotated with MGTD and YTD values in the tree structure for a trade amount of 5 Token A:

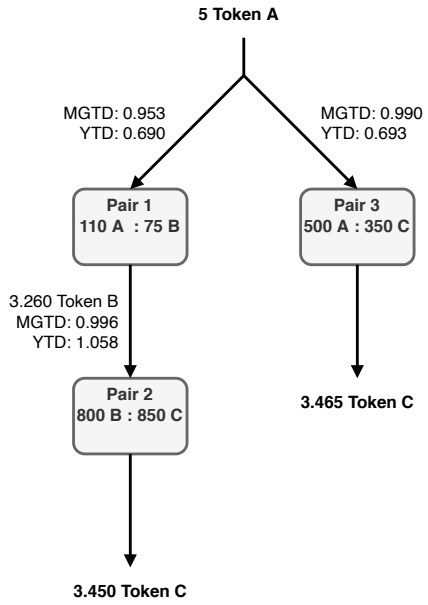

Figure 6: MGTD and YTD Annotated Trade Tree for Multipath Trade of 5 token A.

4.2.1 Pruning Multipath Routes

Pruning of routes occurs at multiple phases in the route generation system. Prior to constructing the tree structure, routes are sorted by MGTD, where a specified number of routes with the highest MGTD are retained. Other filter criteria can be specified in the routing request, including minimum MGTD and maximum number of hops per route.

During multipath route creation, if high MGTD routes exceeding 0.98 are detected, all others are pruned because slippage for high MGTD routes is negligible, eclipsing potential multipath benefits (especially when gas use is considered).

As previously mentioned, YTD is also used to prune routes with poor pricing based on reserve state. The specific criteria used for this type of pruning is any routes where normalized YTD was lower than MGTD (normalized YTD being the route's YTD divided by the maximum YTD found amongst all routes). Such routes typically offered poor pricing, negating any benefit of including them as a path within a multipath route.

Routes containing the same pair as other routes (duplicated pairs) were analyzed and pruned if their slippage exceeded 2%. The analysis would choose the route to retain based on highest MGTD. The reason for this is that proportioning a trade across multiple routes where the same pair was used twice would require modeling the slippage resulting from each proportion to ensure accuracy in the quoted results. For simplicity and schedule

considerations this advanced modeling was omitted, prioritizing quoting accuracy.

4.2.2 Proportioning

To proportion each branch of the multipath trade tree of Figure 6 a breadth first traversal is applied to calculate the proportion for each descending branch based on its MGTD value. The proportion of each branch of the current node in the traversal is computed using Equation 15; i and n are indices of the N branches of the current visited node of the traversal. (Pairs in the tree are nodes as is the root, which is the amount sourced).

$$Proportion_n = \frac{(MGTD_n)^4}{\sum_{i=1}^{N}(MGTD_i)^4} \tag{15}$$

Applying the proportioning equation to the nodes of Figure 6 and annotating the proportions and resulting values onto the tree is illustrated in Figure 7. The combined path results in 3.509 Token C from a swap of 5 Token A, exceeding both single path routes.

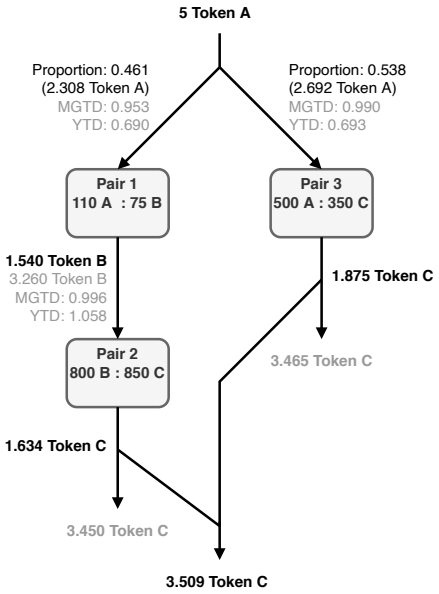

Figure 7: MGTD Proportioned Trade Tree for Multipath Trade of 5 token A, showing combined path output of 3.509 Token C, exceeding both single paths. Gray token values represent single path amounts, black token values represent proportioned multipath amounts.

4.2.3 Example Result

Figure 8 below shows a multipath trade generated with the proportioning algorithm described herein for a trade of $7.5M USDT for wPE. At this particular block in the history of the Uniswap V2 network, the trade netted over 16% improvement over the single path online Uniswap route, yielding an additional $785,332.32 USD of wPE. The proportioning percentage and MGTD for each path are annotated on the diagram along with the amount of each token and its $USD equivalent in parentheses.

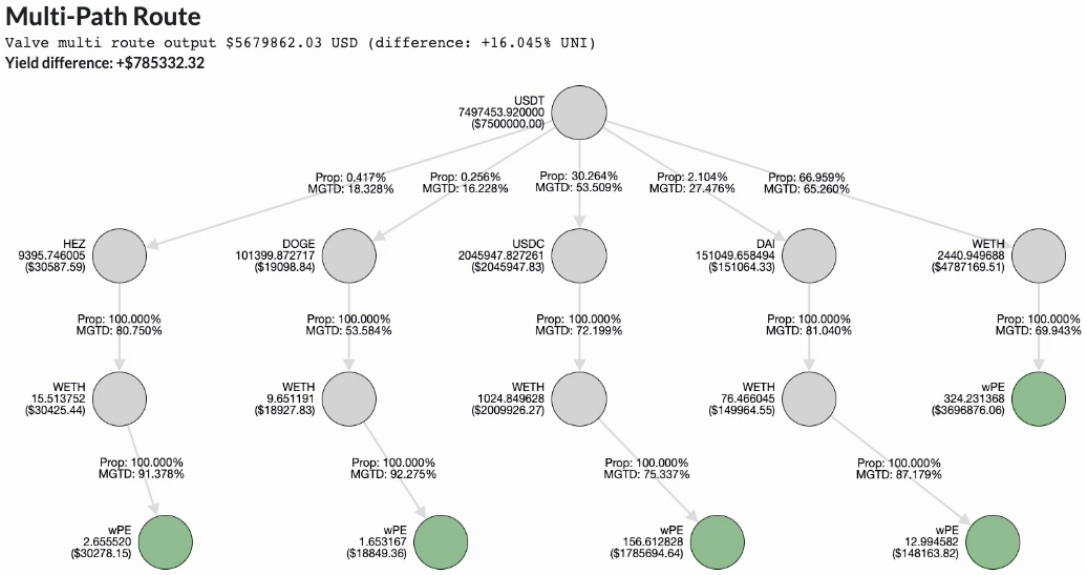

Figure 8: Multipath $7.5M USDT to wPE trade generated with MGTD proportioning, showing over 16% improvement over Uniswap V2 single path route. Green leaf nodes show proportioned path amounts, the sum of which are the total trade result (indicated upper left, next to "multi route output").

5 Conclusion

The MGTD proportioning algorithm allowed creation of infrastructure for analysis and prototyping of routing algorithms for trades in the Uniswap V2 network of pairs. A considerable amount of future work offered improved trade potentials, however with the advent of the Uniswap V3 protocol and resulting migration of liquidity, cross-protocol routes offered the largest potential advantage. Especially given the advantages offered by Uniswap V3's concentrated liquidity for large trades. Fortunately for customers, the Uniswap team has implemented cross-protocol routing in [1].

Thank You

Thank you to Uniswap Grants Program for funding this work.

Thank you to William Pote of Uniswap Labs for suggestions and inspiration that led to much of the work in this area.

References

[1] U. Labs, "Uniswap v2." https://thegraph.com/hosted-service/subgraph/uniswap/uniswap-v2, February 2020. Accessed on 2022-05-07.

[2] U. Team, "Auto router v2." https://uniswap.org/blog/auto-router-v2, December 2021. Accessed on 2022-05-07.

[3] A. Carreira and P. Bhardwaj, "Building blocks for dex router construction & analysis." https://medium.com/@ValveFinance/building-blocks-for-dex-router-construction-analysis-acc03b9f15d8, August 2021. Accessed on 2022-05-07.

[4] A. Carreira and P. Bhardwaj, "Addressing uniswap v2 routing inefficiency." https://medium.com/@ValveFinance/addressing-uniswap-v2-routing-inefficiency-fa29e80ce66a, September 2021. Accessed on 2022-05-07.

[5] H. Adams, N. Zinsmeister, and D. Robinson, "Uniswap v2 core." https://uniswap.org/whitepaper.pdf, March 2020. Accessed on 2022-05-07.

[6] H. Adams, "Uniswap whitepaper." https://hackmd.io/@HaydenAdams/HJ9jLsfTz#DEX-inside-a-Whitepaper, 2018. Accessed on 2022-05-07.