SenSwap: A Meshing Protocol for Automated Market Making

Tu Phan
Descartes Network
tuphan@decartes.network

***Abstract*—Based on the idea of constant product market makers, this paper proposes an advanced protocol that follows a quadratic constant function, however, reduces pool complexity to $\mathcal{O}(n)$.**

Theoretically, the protocol will simulate a virtual anchor that plays a role of bridge to pair any tokens and forms a complete market. The system is designed platform-free. This means the system possibly works on any blockchains supporting smart contract. Although there are many such blockchains, Solana is chosen to implement the system because of low fee and short confirmation time.

With regard to security, the protocol has a quadratic slippage function which can saturate to infinity. This key prevents attackers drain off the pools. Similarly to Uniswap, SenSwap also suffers the impermanent loss that requires liquidity providers must monitor the pool and withdraw liquidity accordingly to minimize the loss.

In conclusion, the protocol allows users deposit into a single pool, and/or pair any tokens to trade.

***Index Terms*—blockchain, decentralized finance, automated market makers**

I. Introduction

In recent years, we are witnessing a great migration of applications from centralized finance (CeFi) to decentralized finance (DeFi). This development is mostly based on the beauty of blockchain that has introduced several attractive properties such as decentralization, transparency, immutability, democracy, and many others. Uniswap, Compound, and Synthetix are typical financial services in the decentralized economy. They respectively implement a decentralized exchange, a decentralized lending & borrowing, and a decentralized derivative, which are three most well-known applications in DeFi [1]. Within scope of this paper, we will discuss about decentralized exchanges.

There are two most popular categories in exchange implementation. One is order books, and the other is automated market makers (AMM). Comparing order-book-based exchanges and AMM-based exchanges, an order-book-based exchange usually maintains a database of orders in both sides bid, and ask to match them, while a AMM-based exchange is supported by function market makers; the function can calculate amount in stock pools and generate corresponding transactions to user's orders. Obviously, AMM-based exchanges can significantly reduce the time of orders matching, increase market liquidity, and open a chance for DeFi applications.

In AMM-based exchanges, assets in pools are deposited by liquidity providers (LP). The reward LPs will get back is transaction fee. Each transaction travels through the system will burden a factional amount of fee. The fee is will redistributed to LPs in proportion. As a compulsory rule in many common application like Uniswap[1], Curve[2], or Balancer[3], LPs must create a pool of two at least. It means LPs have to deposit a pair of tokens with a predefined rate into the pool to create a complete market for these tokens. Therefore, it needs $\frac{n(n-1)}{2}$ pools to create a fully complete $n - tokens$ market, which represents $\mathcal{O}(n^2)$ complexity [2].

There exist some rough solutions to alleviate the problem of no direct connection between two tokens. StableSwap, in Curve, proposed a solution in which a pool can includes more than two tokens, then reduced complexity to $\mathcal{O}(\frac{n^2}{i^2})$, where $i \in [1..n]$ (see II). Nevertheless, all tokens must be involved in the initial state; it cannot be added more afterwards. Another approach is routing order. A routing can find a third shared token and create a bridge to connect two original tokens. This solution is very simple and straight forward to implement, however, very expensive due to multiple transaction fees.

SenSwap is inspired by the idea of routing and uses bridges in the abstract model. Each pool now is a single-token pool. Next, the protocol is trying to imagine an anchor, a virtual stable token, on each pool. Based on the anchor, the protocol computes the value of reserve and the price of token, which are two important indicators for construct the key formulae in SenSwap. Concretely, the value of reserve is identical to the quantity of anchor. The price of token is equal to the quantity of anchor divided by the reserve.

Theoretically, the protocol must satisfy two main criteria. First, when a LP deposit, the price of token in the pool must be constant over all the other pools. Second, when a transaction is executed, the new state must lie on the constant product function. With appearance of the anchor, if reserve changes in the first criterion, the quantity of anchor must change accordingly. In the second criterion, to keep the new state staying on the curve, the quantity of anchor must be unchanged whatever the reserve varies. In practice, SenSwap will make use of quick block confirmation time based on Proof-of-History on Solana to alleviate the front-run attack,

[1]https://app.uniswap.org/
[2]https://curve.fi/
[3]https://balancer.exchange/

and low fee to attract liquidity [3].

II. Related work

The concept of AMM has been studied for many years in academic literature [4]. Instead of human decision makers, some algorithmic makers are created to self operate in electronic markets. To commence, it's relevant to have a look at bonding curves, simple mechanisms used to define price of a token based on supply. The curve can be a linear function, a quadratic function, or even a cube function. In details, giving p is the current price and S is the current supply,

$$p \propto S \quad \text{(A linear bonding curve)} \tag{1}$$

$$p \propto S^2 \quad \text{(A quadratic bonding curve)} \tag{2}$$

$$p \propto \frac{1}{3}S^3 \quad \text{(A cube bonding curve)} \tag{3}$$

Even though bonding curve can serve a price function over supply, it's not observed as a "complete" AMM because bonding curve doesn't examine the relationship between two or more stocks, particularly tokens. Actually, the most famous AMM is the logarithmic market scoring rules (LMSR) [5]. Original applications of LMSR is in prediction market. One of largest tests of LMSR is Gates Hillman Prediction Market. The test sets a context to predict the opening day of a new computer science buildings, the Gates and Hillman Centers, at Carnegie Mellon University [6]. Then it evolves to DeFi, Augur[4], Gnosis[5] for example, which utilized Liquidity-Sensitive LMSR (LS-LMSR) [7], LMSR Primer respectively. In general, the cost function C in the LMSR family is represented as following,

$$C(\overrightarrow{q}) = b \ln \sum_i e^{q_i/b}, \tag{4}$$

where $\overrightarrow{q}$ is the payout vector and $b > 0$ is the liquidity parameter. When b is small, the price will quickly response to the trading, and vice versa. Particularly, in LS-LMSR,

$$b(\overrightarrow{q}) = \alpha \sum_i q_i, \tag{5}$$

where $\alpha > 0$ is a constant.

Beside the variants of LMSR, a Bayesian market maker has introduced a number of advantages over LMSR in binary markets. It offers lower expected loss compared and quickly adapt a market shock [8].

All of the aforementioned approaches are more or less being applied by a number of DeFi platforms over years. However, Uniswap coming was such a game changer. It really created a significant break though for DeFi. Since the moment, not only the number of applications but also the number of users in DeFi has exploded drastically. The main idea behind seems very simple, but it has strongly proved functionality, possibility in both experimental [9] and practical. Uniswap's algorithm, called the constant product function which is a class of constant function market makers (CFMM), formulates the quoting price via the pair of token reserves. Giving token A, and B with corresponding reserves R_A, R_B, the algorithm will maintain the following equation,

$$R_A \times R_B = k, \tag{6}$$

where $k > 0$ is a constant defined at the initial state [2]. In other words, let's call α is a "changing rate" of R_A after a trading (sell A, buy B for example). Then $R'_A = \frac{1}{\alpha}R_A$ induces $R'_B = \alpha R_B$, where $0 < \alpha < 1$, to maintain the constant product,

$$R_A \times R_B = R'_A \times R'_B = k. \tag{7}$$

At timestamp t, the marginal price $p^{(t)} = R_B^{(t)}/R_A^{(t)}$. Therefore, $p^{(0)} = R_B^{(0)}/R_A^{(0)}$ that means the first liquidity provider must deposit both tokens, A and B, with reserves that satisfies the reference market price at the initial state. After the pool's setup, subsequent liquidity providers must follow p by depositing both tokens accordingly as well. It's a problem when users usually have only one side.

A variant of the constant product function is the constant mean function that has introduced by Balancer. Actually, the function is a weighted geometric mean constant product,

$$\prod_{i=1}^{n} R_i^{\omega_i} = k, \tag{8}$$

where $\omega_i \in \mathbb{R}_+^n$ $(i = 1, ..., n)$ and $n >= 2$ is the number of tokens included in the pool [10]. The constant mean function allows a pool can add n tokens over at most 2 tokens like Uniswap. Additionally, the constant mean function is the constant product function when $n = 2$ and $\omega_i = 1, \forall i$.

Consequently, there exists a slippage rate s in both cases because of the constant product.

Definition 1. *For p is the current marginal price, p' is the next marginal price, then a slippage rate,*

$$s = \frac{p'}{p}. \tag{9}$$

Applying to the constant product function in Uniswap and the constant mean function in Balancer,

$$s_{\text{UNISWAP}} = \frac{R'_B/R'_A}{R_B/R_A} = \alpha^2 \tag{10}$$

$$s_{\text{BALANCER}} = \frac{R'^{\omega_B}_B/R'^{\omega_A}_A}{R^{\omega_B}_B/R^{\omega_A}_A} = \alpha^{2\omega_A/\omega_B}. \tag{11}$$

With high slippage rates, these market makers are slightly unsuitable for some relatively stable-priced markets like a pool of two stable tokens. The market needs a less

[4]https://augur.net/

[5]https://gnosis.io/

sensitive algorithm. Due to the demand, Curve has introduced Stableswap, a hybrid CFMM, which employs both a constant sum function (CSF) and a constant product function (CPF). This function shows a great property when it can regulate the slippage rate. It acts like a constant sum at the center of the curve, and shift towards to a constant product at two boundaries [11]. Intuitively, Stableswap connected CSF and CPF via a "leverage" parameter χ,

$$\chi CSF + CPF = k. \tag{12}$$

When $\chi = 0$, the function is obviously identical to a CPF. But when $\chi \to \infty$, the CSF term will dominate the whole function and it becomes a CSF.

In this paper, the problem of protocol complexity will be examined and resolved by two main contributions,

- Construct a $\mathcal{O}(n)$-complexity swap protocol.
- Evaluate the protocol's security by the simulation-based proof.
- Develop a swap on Solana.

III. Core criteria

Because this protocol is developing the constant product market maker, Uniswap will be standard for examination. This section focuses on the aspects of liquidity provision, and exchange which are important to learn main criteria for the meshing protocol.

A. Liquidity Provision

In Uniswap, a pool contains two tokens, A and B for example. At a timestamp t with the marginal price $p^{(t)}$, a user, who wishes to become a LP, must deposit r_A and $r_B = p^{(t)} r_A$ to the pool. The price before and after deposit is constant; And so withdrawal is, where r_A is negative.

Remark 1. *The marginal price will be constant with any liquidity provision, r_A and $r_B = \frac{R_B}{R_A} r_A$,*

$$\frac{R_B}{R_A} = \frac{R_B + r_B}{R_A + r_A}. \tag{13}$$

Furthermore, it's clear that reserve of both tokens will be changed after deposit or withdraw. With every deposit, LPs will receive an amount of share, called liquidity provider token (LPT). Let's assume that $1\ LPT = 1\ USD$, then total value of the pool, in US Dollar, is equal to the sum of LPTs.

B. Exchange

With regard to exchange, the price is not a constant anymore. Instead, it will be changed by following the CPF. Giving a transaction that has changing rate $\alpha = R'_A/R_A$, the price curve is a quadratic function, $p' = \alpha^2 p$ which can be rewritten as $\frac{R'_B}{R'_A} = \alpha^2 \frac{R_B}{R_A}$. We also know that the values of pool A and pool B is equal to each other. Then if R_A change by $1/\alpha$, R_B will change by α,

$$\frac{R'_B}{R'_A} = \frac{\alpha R_B}{\frac{1}{\alpha} R_A}. \tag{14}$$

Let LPT_A and LPT_B be the values of pool A and B respectively, then $LPT_A = LPT_B$. We transform both sides of equation 14 with p_* notating the price of token $*$ over USD. Firstly,

$$\frac{R'_B}{R'_A} = \frac{R'_B}{LPT'_B} \frac{LPT'_A}{R'_A} = \frac{p'_A}{p'_B}. \tag{15}$$

Secondly,

$$\frac{\alpha R_B}{\frac{1}{\alpha} R_A} = \frac{\alpha R_B}{LPT_B} \frac{LPT_A}{\frac{1}{\alpha} R_A} = \frac{\alpha p_A}{\frac{1}{\alpha} p_B}. \tag{16}$$

By connecting these transformation, the obtained equation will be equivalent to equation 14. If market demand of A loses α, then market demand of B gains α in terms of price, $0 < \alpha < 1$. Let's p_A and p_B be price of A and B over USD,

$$\frac{p'_A}{p'_B} = \frac{\alpha p_A}{\frac{1}{\alpha} p_B}. \tag{17}$$

The price change can be explained by equation 16 when the LPT is unchanged, but reserve is changed by α.

Remark 2. *The value of pool is constant with any exchanges,*

$$LPT_* = p_* R_* = constant. \tag{18}$$

Here the formula introduces a conservation-of-value dilemma. Verbally, the pool A has increase number of reserves and the pool B has lose some reserves accordingly. These likely vary the pool value, however, the remark 2 shows the opposite. The reason is that the exchange, without fee, just change the pool equilibrium at t to another pool equilibrium at $t+1$. The pool equilibrium only explains the market demand via the pool price. It won't change the pool value or even trader.

Remark 3. *Without exchange fee, the values of r_A and r_B are equal to each other.*

$$|R'_A - R_A| = |R_B - R'_B|, \tag{19}$$

where the pair of vertical bars is value of r_* by slipped price.

In general, we know that $R'_A = \frac{1}{\alpha} R_A$, $p'_A = \alpha p_A$ (see equation 14, 17). Therefore,

Proposition 1. *Let a_A denote acceleration [6] of slipped price of token A over USD. We have:*

[6]Because we are examining a trade sell A and buy B, then the price p_A will go down and acceleration is negative obviously.

$$a_A = \frac{p'_A - p_A}{R'_A - R_A} = -\alpha \frac{p_A}{R_A}. \tag{20}$$

Corollary 1. *By the proposition above, we can transform $|r_A|$ in terms of a_A,*

$$|r_A| = p_A(R'_A - R_A) + \frac{1}{2}a_A(R'_A - R_A)^2 \tag{21}$$
$$= \frac{1-\alpha}{\alpha}p_A R_A + \frac{1}{2}(-\alpha\frac{p_A}{R_A})(\frac{1-\alpha}{\alpha})^2 R_A^2 \tag{22}$$
$$= \frac{1-\alpha}{\alpha}p_A R_A - \frac{(1-\alpha)^2}{2\alpha}p_A R_A \tag{23}$$
$$= \frac{1-\alpha^2}{2\alpha}p_A R_A \tag{24}$$

IV. A SINGLE-TOKEN POOL

Recall that an anchor, LPT, plays a role of a stable token. Because of US dollar is widely accepted as a common unit for stable token in cryptocurrency markets, it's reasonable for us to define one LPT has extract value of one US dollar.

$$1\ LPT = 1\ USD$$

A pool in SenSwap has no longer contained a pair of tokens. Instead, it has only one token. The anchor, LPT, will be paired with the primary token in the pool to realize remark 1 and 2.

Liquidity Provision. With respect to remark 1, the marginal price must be unchanged before and after any liquidity provision. When a LP deposits r_A, he/she will receive back an amount of lpt_A,

$$lpt_A = p_A r_A, \tag{25}$$

where $p_A = LPT_A/R_A$ is the current price of A, LPT_A is the pool value and R_A is the pool reserve.

With withdrawal, the price is a constant too. When a LP burns, he/she will be able to withdraw a amount of token A,

$$r_A = \frac{lpt_A}{p_A}. \tag{26}$$

(It makes sense because he/she will withdraw a proportional quantity to other LPs, $r_A = \frac{lpt_A}{p_A} = lpt_A \frac{R_A}{LPT_A} = R_A \frac{lpt_A}{LPT_A}$.)

Exchange. Following remark 2, the pool value must be unchanged with any exchanges, $LPT'_A = LPT_A$. On the other hand, R_A will be changed by $0 < \alpha < 1$, $R'_A = \frac{1}{\alpha}R_A$. Therefore,

$$p'_A = \frac{LPT_A}{R'_A} = \frac{LPT_A}{\frac{1}{\alpha}R_A} = \alpha p_A \tag{27}$$

In case of symmetric pool like Uniswap, the corresponding side, token B, is quite simple to compute when $r_B = R_B - R'_B = (1-\alpha)R_B$. Unfortunately, SenSwap is asymmetric AMM as design. For convenience, let β denote the changing rate in pool B according to α. From remark 3 and corollary 1, we have:

$$\frac{1-\alpha^2}{2\alpha}p_A R_A = \frac{1-\beta^2}{2\beta}p_B R_B. \tag{28}$$

Because $p = LPT_B/LPT_A$, we can shorten the equation to:

$$\frac{1-\alpha^2}{2\alpha}\frac{1}{p} = \frac{1-\beta^2}{2\beta}. \tag{29}$$

Or,

$$\beta^2 + \frac{1-\alpha^2}{\alpha p}\beta - 1 = 0. \tag{30}$$

By solving this quadratic equation, we can find $0 < \beta < 1$ and compute $R'_B = \beta R_B$.

Price Oracle. Arbitrage is the key that SenSwap utilizes to tackle the problem of price. Arbitrage could be considered as actions that instantaneously buy and sell securities, currency, or commodities in two or more different markets in order to take advantage of asset prices. People do arbitrage can be called arbitrageurs (Arbs). In the system, arbitrageurs buy tokens in a low-price market then sell it in another high-price market to make profit from the differential amount. Because there are always a differing price between SenSwap and reference markets, arbitrageurs have motivation to make risk-free profit from markets. It's clear that these actions will help the on-chain prices simultaneously updated to the same price in the reference markets.

V. SENSWAP ANALYSIS

A. Slippage rate

The slippage rates of p_A and p_B are $s_A = 1/\alpha$ and $s_B = \beta$ respectively. Then the ratio s_B/s_A is the slippage rate of the quoting price,

$$s = \frac{s_B}{s_A} = \alpha\beta. \tag{31}$$

By the quadratic formula, we also have:

$$\beta = \frac{\alpha^2 - 1 + \sqrt{(1-\alpha^2)^2 + 4\alpha^2 p^2}}{2\alpha p}. \tag{32}$$

Then the slippage rate of the quoting price is only dependent on α and p.

Proposition 2. *For a pool with a constant $p = LPT_B/LPT_A$ in case of exchanges,*

$$s = \frac{\alpha^2 - 1 + \sqrt{(1-\alpha^2)^2 + 4\alpha^2 p^2}}{2p}. \tag{33}$$

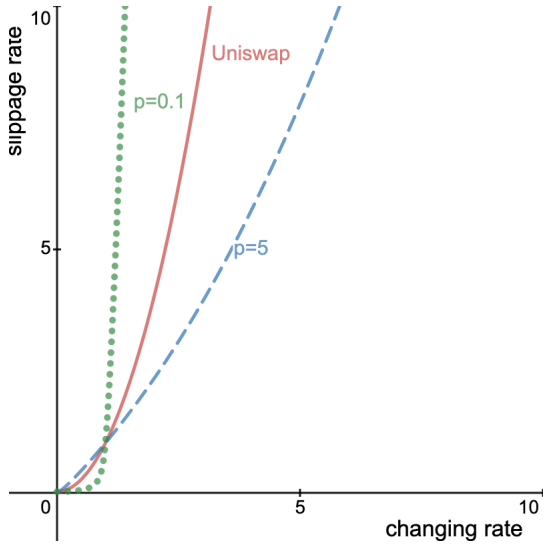

Fig. 1. The continous line is the quadratic slippage curve of Uniswap which is forced in all pools. Depending on the quoting price p, the slippage rates in SenSwap can be very varied. The dotted line in SenSwap's slippage curve in case $p = 0.1$, and the dashed line is for $p = 5$.

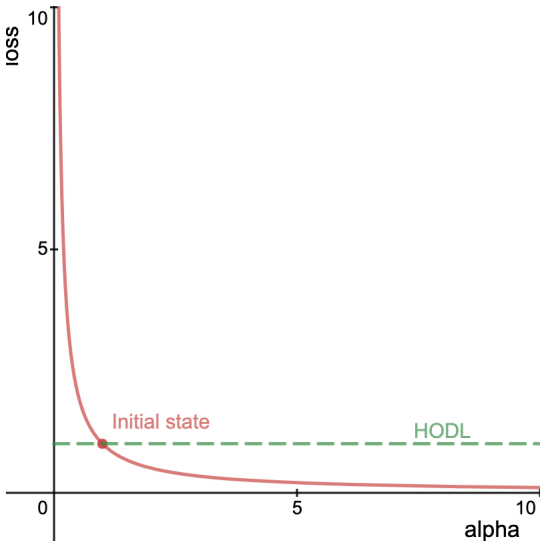

Fig. 2. If the market of A is in selling fashion, becoming a LP gains more profit than HODL. On the contrary, if the market is in buying fashion, the LP must suffer the loss.

B. Impermanent loss

The impermanent loss represents the possible loss of LPs compared to HODL. The loss is due to the divergence in price between deposit and withdrawal. Let's say a LP initializes a pool A to pair with existing B and forms market states $\{R_A, R_B, p_A, p_B\}$. After a time t, the market states is $\{R'_A, R'_B, p'_A, p'_B\}$. Reusing $\alpha = p'/p$ as the changing rate, however, now $\alpha \in \mathbb{R}_+$, the impermanent loss would be shown as:

$$loss = \frac{p'_A R'_A}{p'_A R_A} = \frac{p'_A R'_A}{\alpha p_A R_A} = \frac{1}{\alpha}, \tag{34}$$

$$\tag{35}$$

because $LPT_A = p_A R_A = p'_A R'_A$.

Proposition 3. *Giving the changing rate $\alpha \in \mathbb{R}_+$, then the impermanent loss:*

$$loss = \frac{1}{\alpha}. \tag{36}$$

VI. Conclusions

The paper has proposed a meshing protocol using the single-side AMM. It significantly reduces the pool complexity to $\mathcal{O}(n)$. Now LPs can do deposit and withdrawal by a single token, users can pair any tokens to swap without the restriction of existence. This key can be an technical element that attracts flows of liquidity, the most critical thing for an AMM application's success.

References

[1] D. Lau, D. Lau, and S. Teh, *How to DeFi*. Independently Published, 2020. [Online]. Available: https://books.google.nl/books?id=g6WazQEACAAJ

[2] H. Adams, "Uniswap," *URl: https://docs. uniswap. io/. 13One possibility is that the parties could treat the option itself as the underlying asset, and constantly update their payment channel based on the option's current computed value*, 2019.

[3] A. Yakovenko, "Solana: A new architecture for a high performance blockchain," 2018.

[4] A. M. Othman, "Automated market making: Theory and practice," 2012.

[5] R. Hanson, "Logarithmic markets coring rules for modular combinatorial information aggregation," *The Journal of Prediction Markets*, vol. 1, no. 1, pp. 3–15, 2007.

[6] A. Othman and T. Sandholm, "Automated market-making in the large: the gates hillman prediction market," in *Proceedings of the 11th ACM conference on Electronic commerce*, 2010, pp. 367–376.

[7] A. Othman, D. M. Pennock, D. M. Reeves, and T. Sandholm, "A practical liquidity-sensitive automated market maker," *ACM Transactions on Economics and Computation (TEAC)*, vol. 1, no. 3, pp. 1–25, 2013.

[8] A. Brahma, M. Chakraborty, S. Das, A. Lavoie, and M. Magdon-Ismail, "A bayesian market maker," in *Proceedings of the 13th ACM Conference on Electronic Commerce*, 2012, pp. 215–232.

[9] G. Angeris, H.-T. Kao, R. Chiang, C. Noyes, and T. Chitra, "An analysis of uniswap markets," *Cryptoeconomic Systems Journal*, 2019.

[10] F. Martinelli and N. Mushegian, "A non-custodial portfolio manager, liquidity provider, and price sensor," *URl: https://balancer. finance/whitepaper*, 2019.

[11] M. Egorov, "Stableswap-efficient mechanism for stablecoin liquidity," *URl: https://medium. com/yield-protocol/introducing-ydai-43a727b96fc7*, 2019.