HipoSwap Protocol - 低滑点高流动性去中心化代币兑换协议

L. Han, G. S, J. W, X. Z, C. L, S. L, T. Wei

research@hipo.com

August 27, 2020

摘要

HipoSwap协议是一个自动做市商AMM（Automated Market Maker）交换协议。在CPMM (Constant Product Market Maker) 的基础上，引入两个单币种市价Maker池，为Maker提供市价挂单和赚取手续费的途径，同时大幅增加系统流动性，有效降低Taker的交易价格滑点。

行业背景

区块链产业发展迅猛，每年有数以千计的区块链资产被发行出来，并且这一趋势正在迅速上升，远没有达到顶峰。可以预见，未来市面上将存在数千万乃至更多的流通区块链资产（按每个公司发布一个区块链资产估算）。区块链资产本身是以去中心化的方式存储在区块链上的，用户有很强的兑换或交易（Swap or Exchange）需求。目前的区块链资产交易需求主要由中心化交易平台满足。用户需要将区块链上的去中心化资产，交由中心化交易平台代为管理，然后在中心化交易平台上采用挂单竞价或集合竞价的方式进行交易。

主要存在问题为:

1）用户失去了对区块链资产的实际控制权，一旦中心化交易平台发生问题，用户的资产安全将受到威胁。

2）中心化交易平台资源有限，只能处理部分热门资产的兑换。比如全球支持区块链资产最多的中心化交易平台之一的Gate.io，目前仅支持了300多个币种，500多个交易对。远未达到未来处理数千万种资产的处理能力。

去中心化的兑换交易平台，可以同时解决以上两个问题。去中心化兑换交易平台的主要问题是:

1）因为每一笔交易逻辑需要在每一个区块链节点上得以验证和存储，因此处理和存储逻辑需要极其简单，否则将产生很高的交易成本。因此传统的中心化交易平台的挂单和撮合逻辑并不能很好的适应去中心化交易。有的项目会采用链外挂单撮合（Off-chain orderbook and matching），链上成交清算的方式来降低链上执行成本问题，但是存在去中心化程度差和链外撮合容易作弊的信任危机。

2）去中心化资产的安全存储问题。去中心化的链上资产由用户个人持有和保管，私钥保存在用户手上，普通用户不一定具备中心化平台的专业管理经验和基础设施，很容易成为黑客盗取的目标。并

且用户私钥很容易出现损毁而无法恢复资产的情况。需要采用多签硬件设备或类似GateChain提出的链上保险账号等方式，降低去中心化资产安全保管门槛，让普通大众可以做到安全保管自己的资产。

行业前沿研究和最新方案介绍

2017年由 Alan Lu提出一种适合在以太坊网络上实现的去中心化交易协议[1]，在以太坊智能合约中采用乘积不变方程 X*Y=k，可以为X和Y资产兑换实现自动做市商。Uniswap项目实现了这一方法，并在2020年DeFi热潮中大获成功[2]。乘积不变方程还可以推广到更为一般的形式，比如Balancer [3] 依照$V = \prod_t B_t^{W_t}$ 价值函数（Value Function）为多种资产按任意权重配置并可根据市场价格变化自动调整资产数量保持资产价值比例，同时可以为两两资产提供兑换服务。是一个比Uniswap更加一般话的实现。

在使用常数乘积自动做市商方式CPMM 的去中心化实现中，存在两个缺点：

1）无常损失(Impermanent Loss)。当价格发生偏移时，不管是上涨还是下跌，都会给流动性提供者（Liquidity Provider LP）带来无常损失(impermanent loss)，这种损失只有价格回到LP抵押时的价格原点时才能得以消除，这会打击流动性提供者LP的积极性。因此在Uniswap中收取Taker 0.3%的高额手续费来奖励LP，以弥补他们的无常损失。但是这将会影响Taker的积极性。

2）高滑点成本(High Price Slippage Cost)。采用X*Y=k这样的乘积常数曲线模型，当单笔交易金额较大时，将产生很高的价格滑点 (price slippage)，给Taker带来很高的成本。要消除这一滑点，可以采用X+Y=k这样的直线模型，但是这将造成市场价格不随市场需求变化。curve.fi 针对稳定币的兑换特点，提出一种曲线和直线模型相结合的新模型[4]，可以有效降低稳定币兑换的滑点。但这一模型并不适合任意币种的兑换。

2020年，Bancor去中心化兑换协议在V2版本中提出一种利用预言机获得中心化交易市场价格，并动态更新LP池权重来消除无常损失的方法[5, 6]。并且对LP池采用20倍放大来降低交易价格滑点。但是这一方法只适合已经在中心化平台有成熟交易市场，并且已经有预言机支持的交易对，对于更多的新币种和小币种市场，预言机很难迅速和全面的支持，这一方法就不适应了。因此这大大限制了此方法的使用范围。

提出新的HipoSwap协议

我们提出一种新的低滑点，高流动性去中心化兑换协议 - HipoSwap。HipoSwap协议是一个自动做市商AMM交换协议。在CPMM的基础上，引入两个单币种市价Maker池，大幅增加系统流动性，并可以有效降低Taker的交易价格滑点。

HipoSwap协议中包括跟Uniswap或Balancer类似的一个常数乘积CPMM 或 CFMM（Constant Function Market Maker) 交易模型所需要的按资产比例关联的Liquidity Pool 包含X和Y资产，由Liquidity Provider （LP）用户同比提供流动性。同时引入两个新的互相对立的Maker池（Maker Pool X和Maker Pool Y），分别由X Maker 和 Y Maker 独立提供流动性。

HipoSwap协议采用杠杆的方式，将Taker的小部分需求（如10%）使用CPMM的Liquidity Pool满足，大部分（如90%）由X和Y Maker Pools满足，当X，Y Maker Pool不足的时候，只能由Liquidity Pool满足。

每当Taker从Liquidity Pool进行兑换的时候（Taker的部分需求，比如10%），会因为需求的改变造成价格的波动。不同方向的Taker操作会竞争形成一个合理的价格。

每当Taker从X，Y Maker Pool进行兑换的时候（Taker的部分需求，比如90%），价格以Liquidity Pool的当前价格为准，不会造成滑点扩大。当Taker从Y兑换为X的时候，X Maker Pool的X数量（以Xm表示）会减少，Y数量（以Ymr表示）会增多，同时会积累X Maker Pool的手续费奖励池Yfee。同样的，当Taker从X兑换为Y的时候，Y Maker Pool的Y数量（以Ym表示）会减少，X数量（以Xmr表示）会增多，同时会积累Y Maker Pool的手续费奖励池Xfee。

X Maker 把X存入 X Maker Pool之后，随时可以从Ymr池子里面提取要兑换的Y币种，兑换价格以Ymr历史积累的平均兑换价格price_xy_avg为准。为了鼓励X Maker更久的把X币种存入X Maker Pool我们设计了一套与存入时间相关的手续费制度：如果存入时间小于池子平均Maker存币时间，则需要缴纳手续费，手续费加入到Yfee；如果大于池子平均Maker存币时间，可以获得手续费，手续费取自Yfee。

Y Maker把Y存入Y Maker Pool之后，随时可以从Xmr池子里面提取要兑换的X币种，价格以Xmr历史积累的平均价格price_yx_avg为准，手续费也跟存入时间有关系，存入或者取自Xfee，方式同上。

Maker Pool需要按一定原则不定期更换。Maker Pool里面的资产会逐渐兑换为另一个资产，随着时间的推移，有可能出现Maker Pool里面已经兑换完成的资产价值不如场外合算的情况。比如Maker Pool里面已经完成兑换的BTC平均价格为11000美金，但是其他市场里面BTC价格大涨已经到了12000美金，那么当前的Maker Pool就不会再有人存BTC进来兑换，并且已经存储进来的BTC会被提空，造成Maker Pool一直处于枯竭状态，直到价格再次跟外面持平或反转才能得到缓解，如果价格一直没有平衡，枯竭问题就一直得不到解决，这给Maker Pool的流动性带来了很强的不确定性。因此我们提出一个达到一定条件就换Maker Pool的操作来解决这一问题。为了避免频发切换或者因为某些特殊情况下被误触发切换，可以设置一个最低持续时间，比如一个Maker Pool至少持续使用30分钟（或对应的平均区块数）后才允许触发切换。

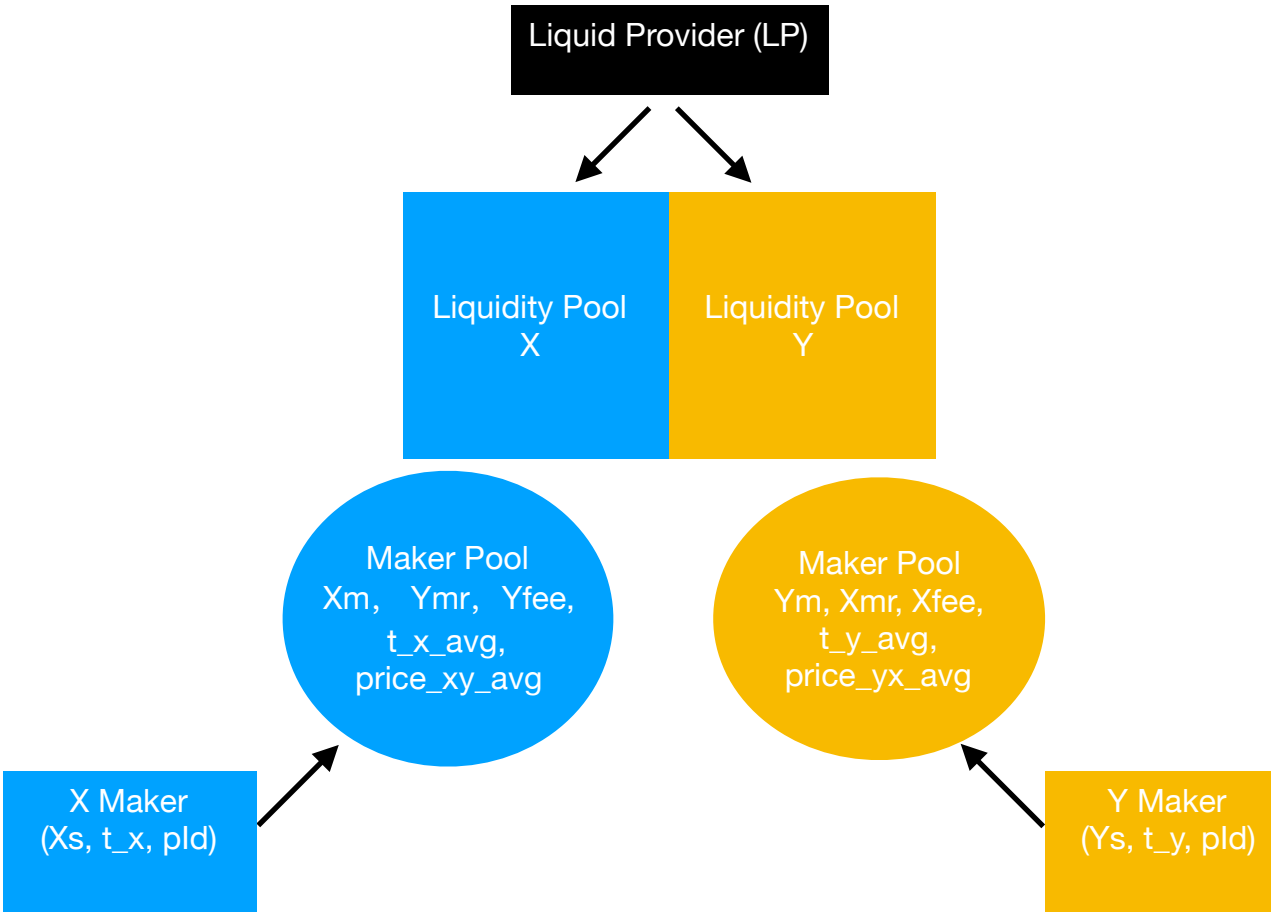
当当前Maker Pool大部分已经完成兑换（比如90%）时，下一笔用户的存入操作的时候，将切换到新Maker Pool。比如X Maker Pool里面里面未完成兑换的X价值与已经兑换完成的Ymr价值低于1:9时，就认为需要换新的X Maker Pool了。新存入的X将自动进入到新的X Maker Pool，并重新开始统计平均兑换价格。之前在老的X Maker Pool存入X的用户，只能从老的Maker Pool里面获取X或Y。这一操作可以不定期为一部分前期Maker用户完成结算，而不会因为市场价格波动影响到后面Maker用户提供流动性的热情。

整个过程需要注意，智能合约操作中减少storage的操作，也可以Taker操作的时候通过随机数判断从Liquidity Pool里面兑换还是从Maker Pool里面兑换，可以减少增提Taker的手续费损耗，不过将给兑换滑点带来不确定性。

本协议的优势

1）有效降低兑换滑点：如果杠杆率是n，那么可以让滑点降低为Uniswap模式的1/n。

2）提供更多的需求，比如计划一段时间内卖出一种资产获得另一种资产的用户可以来提供流动性，而不单纯是让同比抵押的用户提供流动性。



有四个流动性池子Pools：

Liquidity Pool (X-Y)： 保持市值比例存放LP存入的Staking资产

Maker Pool X：包含X Maker存入的X资产，总量Xm，还有兑换回来的Y资产Ymr，收到的Y币种Maker fee Yfee

Maker Pool Y：包含Y Maker存入的Y资产，总量Ym，还有兑换回来的X资产Xmr，收到的X币种Maker fee Xfee

市场参与者分为四种：

Liquidity Provider (LP)： 锁仓获取利息到用户。通过抵押的方式等价值比例的存入两种代币（或多种）到Liquidity Pool，来为CPMM提供流动性支持和价格发现。可以收到部分Taker的收益，但是需要承担无常损失 （Impermanent Loss）。

X Maker：要卖出X买入Y的用户。将X注入到Maker Pool X，经过一段时间后，X将会被系统自动按市价兑换成Y。这个兑换过程需要一段时间才能完成，按照现在Uniswap的流动性估算，大概需要12小时。市价是按照Taker发起交易时的CPMM系统的价格。

当X Maker向Maker Pool X存入X的时候，在合约账户里面记录（Xs, t_x)两个值，Xs指的是X sent，就是下单多少X，t_x是最后一次存入X的时间，用于计算提取的手续费。用户可以任何时候提取合约账户中的Xs或已经兑换完成的Y到自己的钱包地址。

Y Maker：要卖出Y买入X的用户。将Y注入到Maker Pool Y，经过一段时间后，Y将被系统自动按市价兑换成X。这个兑换过程需要一段时间完成。

当Y Maker向Maker Pool Y存入Y的时候，在合约账户里面记录（Ys, t_y)两个值，Ys指的是Y sent，就是下单多少Y。t_y是最后一次存入Y的时间。

Taker：Taker主动发起兑换请求

以下是各种操作的定义（这里以Uniswap x*y=k模型为例，也可以用更一般的乘积不变模型，比如在Balancer中使用的模型）：

Liquid Provider 增加流动性 addLiquidityS(Δx) ，Liquidity Pool 中的X，Y要同比增加:

$$X' = X * (1 + \alpha)$$

$$Y' = Y * (1 + \alpha)$$

$$\alpha = \frac{\Delta x}{X}$$

Liquid Provider 删除流动性 removeLiquidity(Δx)，Liquidity Pool中的X，Y要同比减少:

$$X' = X * (1 - \alpha)$$

$$Y' = Y * (1 - \alpha)$$

$$\alpha = \frac{\Delta x}{X}$$

X Maker增加流动性 XMakerDeposit(Δx), Maker Pool X中的Xm要增加，同时修改用户的debt 和存入时间（Xs, t_x)，如果之前已经有Xs存入还未取出，那么之前的存入时间会被覆盖，手续费奖励会受到影响。注意因为有多个X Maker Pool的存在，所以存入的时候需要获得最新的Pool Id，如果判断用户当前Pool Id不是最新的Pool Id，那么需要先按默认比例为用户完成老的Pool的清算（调用 X Maker 提取流动性操作），然后再进行存入新Pool的操作。

$$p_{Id} = Current\ X\ Pool\ Id$$
$$Xm'(p_{Id}) = Xm(p_{Id}) + \Delta x$$
$$Xs' = Xs + \Delta x$$
$$t_x' = current\ time$$

换Pool：在X Maker存入Δx操作之前，首先需要判断是否需要换Pool。发现X Maker Pool里面里面未完成兑换的X (Xm) 价值与已经兑换完成的Ymr价值低于1:9时，就认为需要换新的X Maker Pool了，为了避免频发切换或者因为某些特殊情况下被误触发切换，可以设置一个最低持续时间，比如一个Maker Pool至少持续使用30分钟（或对应的平均区块数）后才允许触发切换。

$$if(Xm's\ value < 1/9\ Ymr's\ value) : Current\ X\ Pool\ Id\ += 1$$

X Maker提取流动性 XMakerWithDraw(Δx, Δy), 不需要同比减少，可以只提取一个资产，也可以同时提取一部分X，和一部分Y资产。Maker Pool X中 Xm和Ymr 都应该减少，用户debt在合约中的记录（Xs）需要减少。Xs最低减为0 。提取X资产，没有触发兑换，不收取手续费也不奖励手续费。从Ymr提取Y资产，价格以Ymr历史积累的平均兑换价格price_xy_avg为准，需要支付的手续费跟存入时间有关系，如果存入时间小于池子平均Maker存币时间，需要缴纳手续费，手续费加入到Yfee，如果大与池子平均Maker存币时间，可以获得手续费，手续费取自Yfee。注意需要根据X Maker账号记录的对应的Pool Id去提取流动性，不一定是最新的Maker Pool。
计算总计需要减记的用户debt:

$$\Delta x' = price\_xy\_avg * \Delta y + \Delta x$$

下面具体减记和提取操作:

$$Xm'(p_{Id}) = Xm(p_{Id}) - \Delta x'$$

$$Xs' = Xs - \Delta x'; only\ if\ \Delta x' <= Xs$$

$$Ymr'(p_{Id}) = Ymr(p_{Id}) - \Delta y$$

$$Yr' = Yr - \Delta y; only\ if\ \Delta y <= Yr$$

如果不指定，Δx, Δy比例，默认可以根据X Maker Pool里面已经完成的兑换比例来提取：

$$\sigma = \frac{Xm}{(Xm + price\_xy\_avg * Ymr)}$$

(这里可能需要考虑兑换前计算还是兑换后计算！！！！后面需要仔细评估）

计算总计需要减记的用户debt:

$$\Delta x = \sigma Xs$$

$$\Delta y = (1-\sigma)Xs * price\_xy\_avg$$

$$\Delta x' = Xs$$

下面的具体减记和提取操作就一样了:

$$Xm'(p_{Id}) = Xm(p_{Id}) - \Delta x'$$

$$Xs' = Xs - \Delta x'; only\ if\ \Delta x' <= Xs$$

$$Ymr'(p_{Id}) = Ymr(p_{Id}) - \Delta y$$

$$Yr' = Yr - \Delta y; only\ if\ \Delta y <= Yr$$

Y Maker增加流动性： YMakerDeposit(Δy), Maker Pool Y中的Ym要增加，同时修改用户的debt 和存入时间（Ys, t_y)，如果之前已经有Ys存入还未取出，那么之前的存入时间会被覆盖，手续费奖励会受到影响。注意因为有多个Y Maker Pool的存在，所以存入的时候需要获得最新的Pool Id，如果判断用户当前Pool Id不是最新的Pool Id，那么需要先按默认比例为用户完成老的Pool的清算（调用 Y Maker 提取流动性操作），然后再进行存入新Pool的操作，注意Current X Pool Id和Current Y Pool Id不一定一致，不能用同一个变量代替。

$$p_{Id} = Current\ Y\ Pool\ Id$$

$$Ym'(p_{Id}) = Ym(p_{Id}) + \Delta y$$

$$Ys' = Ys + \Delta y$$

$$t_y' = current\ time$$

换Pool：在Y Maker存入Δy操作之前，首先需要判断是否需要换Pool。发现Y Maker Pool里面里面未完成兑换的Y价值与已经兑换完成的Xmr价值低于1:9时，就认为需要换新的Y Maker Pool了，为了避免频发切换或者因为某些特殊情况下被误触发切换，可以设置一个最低持续时间，比如一个Maker Pool至少持续使用30分钟（或对应的平均区块数）后才允许触发切换。

$$if(Ym's\ value < 1/9\ Xmr's\ value) : Current\ Y\ Pool\ Id\ += 1$$

Y Maker提取流动性： YMakerWithdraw(Δy, Δx), 不需要同比减少，可以只提取一个资产，也可以同时提取一部分Y，和一部分X资产。Maker Pool Y中 Ym和Xmr 都应该减少，用户debt在合约中的记录（Ys）需要减少。Ys，Xr最低减为0 。提取Y资产，没有触发兑换，不收取手续费也不奖励手续费。从Xmr提取X资产，价格以Xmr历史积累的平均兑换价格price_yx_avg为准，需要支付的手续费跟存入时间有关系，如果存入时间小于池子平均Maker存币时间，需要缴纳手续费，手续费加入到Xfee，

如果大与池子平均Maker存币时间，可以获得手续费，手续费取自Xfee。注意需要根据Y Maker账号记录的对应的Pool Id去提取流动性，不一定是最新的Maker Pool。

计算总计需要减记的用户debt:

$$\Delta y' = price\_yx\_avg * \Delta x + \Delta y$$

下面具体减记和提取操作:

$$Ym'(p_{Id}) = Ym(p_{Id}) - \Delta y'$$

$$Ys' = Ys - \Delta y'; only\ if\ \Delta y' <= Ys$$

$$Xmr'(p_{Id}) = Xmr(p_{Id}) - \Delta x$$

$$Xr' = Xr - \Delta x; only\ if\ \Delta x <= Xr$$

如果不指定，Δx, Δy比例，默认可以根据Y Maker Pool里面已经完成的兑换比例来提取:

$$\sigma = \frac{Ym}{(Ym + price\_yx\_avg * Xmr)}$$

(这里可能需要考虑兑换前计算还是兑换后计算！！！！后面需要仔细评估）

计算总计需要减记的用户debt:

$$\Delta y = \sigma Ys$$

$$\Delta x = (1 - \sigma) Ys * price\_yx\_avg$$

$$\Delta y' = Ys$$

下面的具体减记和提取操作就一样了:

$$Ym'(p_{Id}) = Ym(p_{Id}) - \Delta y'$$

$$Ys' = Ys - \Delta y'; only\ if\ \Delta y' <= Ys$$

$$Xmr'(p_{Id}) = Xmr(p_{Id}) - \Delta x$$

$$Xr' = Xr - \Delta x; only\ if\ \Delta x <= Xr$$

Market Taker 提取流动性

杠杆倍数=n，那么理论上从Liquidity Pool里面提取的量的比例:

$$r_s = \frac{1}{n}$$

从Maker Pool里面提取的量的比例:

$$r_m = 1 - \frac{1}{n}$$

$$\gamma = 1 - \rho$$

ρ是Taker 手续费

Taker输入X的量=Δx，最终兑换获得Y的量=Δy

当前Liquidity Pool的价格为：p=x/y

首先从Maker Pool Y里面获得Y，获得的量为(注意这里考虑了手续费)：

$$\Delta y_1 = Min(\Delta x / p * r_m, Ym) * \gamma$$

完成兑换的X为：

$$\Delta x_1 = p * \Delta y_1 / \gamma$$

剩余需要兑换的X为:

$$\Delta x_2 = \Delta x - \Delta x_1$$

剩余部分全部从Liquidity Pool兑换(跟Uniswap一致)：

$$\Delta y_2 = \frac{\alpha\gamma}{1+\alpha\gamma} y$$

$$\alpha = \frac{\Delta x_2}{X}$$

更新Y Maker Pool的数量：

$$Ym' = Ym - \Delta y_1 / \gamma$$

$$Xmr' = Xmr + \Delta x_1$$

$$price\_yx\_avg' = \frac{price\_yx\_avg * Xmr + \Delta y_1}{Xmr + \Delta x_1}$$

更新Liquidity Pool的数量：

$$Y' = Y - \Delta y_2 / \gamma$$

$$X' = X + \Delta x_2$$

Taker输入Y的量=Δy，最终兑换获得X的量=Δx

当前Liquidity Pool的价格为：p=x/y

首先从Maker Pool X里面获得X，获得的量为(注意这里考虑了手续费)：

$$\Delta x_1 = Min(\Delta y * p * r_m, Xm) * \gamma$$

完成兑换的Y为:

$$\Delta y_1 = \Delta x_1 / p / \gamma$$

剩余需要兑换的X为:

$$\Delta y_2 = \Delta y - \Delta y_1$$

剩余部分全部从Liquidity Pool兑换(跟Uniswap一致):

$$\Delta x_2 = \frac{\beta}{1-\beta}\frac{1}{\gamma}x$$

$$\beta = \frac{\Delta y_2}{Y}$$

更新X Maker Pool的数量：

$$Xm' = Xm - \Delta x_1 / \gamma$$

$$Ymr' = Ymr + \Delta y_1$$

$$price\_xy\_avg' = \frac{price\_xy\_avg * Ymr + \Delta x_1}{Ymr + \Delta y_1}$$

更新Liquidity Pool的数量:

$$X' = X - \Delta x_2 / \gamma$$

$$Y' = Y + \Delta y_2$$

增加一个单独的触发换Maker Pool X的操作接口

这个接口并不是必须的，但当没有Maker主动出发换Maker Pool操作的时候，可以由任意好心人触发。收到这个请求时，需要检查确认X Maker Pool里面里面未完成兑换的X价值与已经兑换完成的Ymr价值低于1:9时，就认为需要换新的X Maker Pool，否则不执行操作:

$$if(Xm's\ value < 1/9\ Ymr's\ value) : Current\ X\ Pool\ Id\ +=1$$

增加一个单独的触发换Maker Pool Y的操作接口

这个接口并不是必须的，但当没有Maker主动出发换Maker Pool操作的时候，可以由任意好心人触发。收到这个请求时，需要检查确认Y Maker Pool里面里面未完成兑换的Y价值与已经兑换完成的Xmr价值低于1:9时，就认为需要换新的Y Maker Pool，否则不执行操作:

$$if(Ym's\ value < 1/9\ Xmr's\ value) : Current\ Y\ Pool\ Id\ +=1$$

总结

本协议综合了跟Uniswap一样的CPMM (Constant product market makers) 协议，可以在不依赖预言机喂价的情况下，自动发现市场价格。但是Uniswap协议没有Maker入口，除了流动性提供者LP之外，其他交易用户都必须成为Taker。对于大交易量交易需求的用户，Taker操作的滑点太大。本协议为Maker用户提供了市价下单的通道，对于交易资金量比较大，不着急立即成交的用户，可以通过Maker接口为市场提供流动性，同时给自己有效降低滑点，并且可以赚取Taker手续费。在不考虑市场涨跌的情况下，可以有效降低Maker用户的成交成本。同时因为Maker的存在，让Taker不止能通过CPMM的LP池（Liquidity Pool）获得流动性，还可以直接跟Maker Pool成交，有效降低滑点。在同等CPMM LP池（Liquidity Pool）流动性，并且本协议的Maker Pool充足的情况下，如果设置杠杆率为10，那么Taker的滑点可以降低到之前单纯与CPMM LP池成交的情况的1/10。为了补偿LP的无常损失，Uniswap收取Taker 0.3%的高额手续费造成交易用户的高交易成本。在本协议中， 可以设置不同的费率来整体优化Taker的成本。比如每一笔从Liquidity Pool成交的部分收取0.35%的手续费奖励LP用户以提高他们的积极性，每一笔从Maker Pool成交的部分收取0.15%的低手续费来降低Taker的成本。因为预期Taker大部分金额是从Maker Pool成交的，因此Taker的整体成本仍然会有显著的下降。理想情况下假设90%的金额通过Maker Pool成交，那么Taker的评价手续费可以降低到0.17%。

本协议里面包含的Liquidity Pool的运行机制跟Uniswap，Balancer等协议一致，当执行Taker操作的时候，也可以通过msg调用Uniswap/Balancer的合约完成兑换和获得价格，然后再进行Maker Pool操作，这样可以充分利用Uniswap LP池的流动性。

为激励Maker和Liquidity Provider为交易系统提供更多的流动性，可以通过发布治理代币(Governance Token)的方式为系统提供流动性挖矿奖励（Liquidity Mining Rewards），挖掘出来的治理代币可以参与未来系统升级提案的投票，获得系统手续费的奖励等。

本协议可以部署在不同的区块链上，Ethereum, Tron和EOS网络上可以使用智能合约实现。GateChain上可以使用原生代码实现。在GateChain上面的实现可以结合GateChain的普通账户（Regular Account）和保险账户（Vault Account）的特性，将常用交易资金存放在普通账户中，将不常用的大额资金存放在保险账户中，平衡用户的交易需求和安全性要求。

如果本协议要在转账交易吞吐量低(Low network throughput)，转账交易成本很高(high network transaction costs) 的区块链网络上部署，比如Ethereum，可以选择使用二层协议（Layer 2 protocols）的方式加快转账速度和降低转账成本。

References：

[1] 2017, Alen Lu, Building a Decentralized Exchange in Ethereum. https://blog.gnosis.pm/building-a-decentralized-exchange-in-ethereum-eea4e7452d6e
[2] 2020, Uniswap v2 Core. https://uniswap.org/whitepaper.pdf

[3] 2019, A non-custodial portfolio manager, liquidity provider, and price sensor. https://balancer.finance/whitepaper/
[4] 2019, StableSwap - efficient mechanism for Stablecoin liquidity, https://www.curve.fi/stableswap-paper.pdf
[5] 2020, Bancor V2 Launches. https://blog.bancor.network/bancor-v2-launches-b1fec492eeb2
[6] 2020, Breaking Down Bancor V2 Dynamic Automated Market Makers. https://blog.bancor.network/breaking-down-bancor-v2-dynamic-automated-market-makers-4e90c0f9a04