Notes

Miscellaneous notes about **arbi**.

Uniswap V2's optimal input amount

We consider two Uniswap V2-like pairs A and B both relative to the same two tokens. Let X_A and Y_A the reserves of the two tokens on the pair A and Y_A and Y_B the reserves on the pair B and assume that we want to perform 2 chained this way.

$$\ldots \overset{y^*}{\rightarrow} A \overset{x_{\text{out}}}{\rightarrow} B \overset{y_{\text{out}}}{\rightarrow} \ldots$$

with y^* the optimum amount to swap in order to maximize the gain function $G(y) = y_{\text{out}} - y^*$

Let $0 \leq f \leq 1$ be the fee (.03 by deault on Uniswap V2), we know[1] that the optimum is one of the roots of the following second-grade equation:

$$k^2 y^2 + 2kY_A X_B y + (Y_A X_B)^2 - (1-f)^2 X_A Y_B Y_A X_B = 0$$

where

$$k = (1-f)X_B + (1-f)^2 X_A$$

In the Uniswap V2 implementation we have that $1 - f = \frac{\varphi}{1000}$ (with $\varphi = 997$). Then we can rewrite:

$$k^2 y^2 + 2kY_A X_B y + (Y_A X_B)^2 - \left(\frac{\varphi}{1000}\right)^2 X_A Y_B Y_A X_B = 0$$

and

$$k = \frac{\varphi}{1000} X_B + \frac{\varphi^2}{1000^2} X_A$$

Let a, b and c be the three second-grade equation coefficients.

$$a = k^2$$

$$b = 2kY_A X_B$$

$$c = (Y_A X_B)^2 - \left(\frac{\varphi}{1000}\right)^2 X_A Y_B Y_A X_B$$

Since b is even we can find the roots with

$$y_i = \frac{-\frac{b}{2} \pm \sqrt{\frac{b^2 - 4ac}{4}}}{a}$$

Replacing our values:

$$\frac{-kY_A X_B y \pm \sqrt{k^2 (Y_A X_B)^2 \left((Y_A X_B)^2 - \frac{\varphi^2}{1000^2} X_A Y_B X_B Y_A\right)}}{k^2}$$

$$= -\frac{Y_A X_B}{k} \pm \frac{1}{k^2} \sqrt{k^2 \left((Y_A X_B)^2\right) - (Y_A X_B)^2 + \frac{\varphi^2}{1000^2} X_A Y_B X_B Y_A}$$

[1]https://www.youtube.com/watch?v=9EKksG-fF1k

$$= -\frac{Y_A X_B}{k} \pm \frac{1}{k}\sqrt{\frac{\varphi^2 X_B Y_B X_B Y_A}{1000^2}}$$

Which, since the square root is positive, can be positive only considering $+$. In conclusion we get the following formula for the optimal amount of token Y:

$$y^* = \frac{1}{k}\left(\sqrt{\frac{\varphi^2 X_A Y_B X_B Y_A}{1000^2}} - Y_A X_B\right)$$

Solidity implementation details

- Integer square roots can be effectively and cheaply computed using the Babylonian method[2]
- The square root can lead to overflow, in that case it can be convenient splitting it into something like

$$\sqrt{\varphi \times X_A \div 1000 \times Y_B}\sqrt{\varphi \times X_B \div 1000 \times Y_A}$$

[2]https://ethereum.stackexchange.com/a/97540/66173