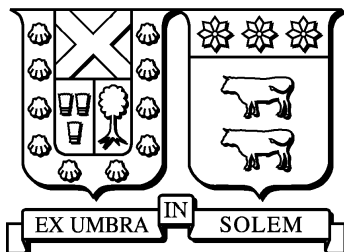


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

SANTIAGO – CHILE



“OPTIMIZACIÓN DE TRANSACCIONES EN
AGREGADORES DEX: UN ENFOQUE DE
PROGRAMACIÓN LINEAL PARA MAXIMIZAR
RETORNOS EN AMBIENTES
DESCENTRALIZADOS ”

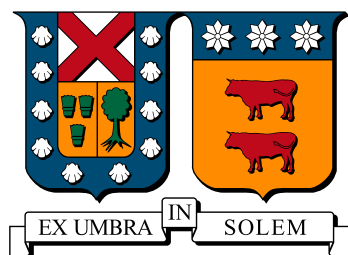
IGNACIO LIZANA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: CARLOS CASTRO

OCTUBRE 2023

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



**“OPTIMIZACIÓN DE TRANSACCIONES EN
AGREGADORES DEX: UN ENFOQUE DE
PROGRAMACIÓN LINEAL PARA MAXIMIZAR
RETORNOS EN AMBIENTES
DESCENTRALIZADOS ”**

IGNACIO LIZANA

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO**

PROFESOR GUÍA: CARLOS CASTRO

PROFESOR CORREFERENTE: PEDRO GAJARDO

OCTUBRE 2023

MATERIAL DE REFERENCIA, SU USO NO INVOLUCRA RESPONSABILIDAD DEL AUTOR O DE LA INSTITUCIÓN

Agradecimientos

Primero que todo, quiero extender mi profundo agradecimiento a mi madre, Angelina. Su presencia y apoyo incondicional, en especial durante los momentos más arduos, ha sido mi pilar. Mi hermano Eduardo no se quedó atrás; siempre estuvo atento a mi progreso y evolución, dándome su enfoque humanista.

También a mis amistades quienes han sido una parte fundamental en este camino:

Nicolás, mi compañero de batallas, con quien compartimos y nos apoyamos en tantos momentos de nuestra carrera, estando casi siempre en las mismas situaciones, incluida la elaboración de nuestras memorias.

Fernanda, quien desde las primeras etapas de este trabajo, estuvo con la mejor disposición a apoyarme y brindar consejos valiosos para el avance del documento.

Manfred, con su pasión por el mundo cripto, siempre mostró entusiasmo por aprender del documento, desde sus etapas tempranas.

Carlitos, por ser una inspiración constante a seguir adelante.

Benjamín que siempre me apoyó en los momentos de mayor exigencia.

Un reconocimiento especial a mi profesor guía, Carlos Castro, ya que con su predisposición y guía me ayudó a delinear y centrar el enfoque de mi trabajo. Finalmente, un reconocimiento especial a mi profesor correferente, Pedro Gajardo, al cual le agradezco por su cercanía, colaboración continua y feedback oportuno.

Resumen

Este trabajo se centra en la optimización de transacciones en DEX (intercambios descentralizados), especialmente mediante agregadores DEX. El reto principal radica en maximizar el número de tokens obtenidos a través de transacciones, teniendo en cuenta tanto los costos de gas como las tasas de conversión que pueden variar significativamente en el dinámico ecosistema de las finanzas descentralizadas.

A través de un modelo de programación lineal, se creó una solución que identifica rutas óptimas para transacciones. Esta solución toma en consideración múltiples factores, tales como el costo de la comisión de la transacción, la liquidez disponible, el potencial de deslizamiento y las variaciones en tasas de conversión. Adicionalmente, se realizó un análisis extenso de la estructura y comportamiento de las rutas de conversión que utilizan los agregadores, revelando la relación entre las comisiones, la liquidez, el enrutamiento y el deslizamiento.

Palabras clave: DEX, agregadores DEX, programación lineal, optimización, costos de gas, deslizamiento, tasas de conversión, liquidez, blockchain.

Abstract

This work focuses on the optimization of transactions in DEX (decentralized exchanges), especially through DEX aggregators. The primary challenge lies in maximizing the number of tokens obtained through transactions, considering both gas costs and conversion rates that can vary significantly in the dynamic ecosystem of decentralized finance.

Using a linear programming model, a solution was developed to identify optimal transaction paths. This solution takes into account multiple factors, such as the transaction fee cost, available liquidity, slippage potential, and variations in conversion rates. Additionally, an extensive analysis of the structure and behavior of conversion paths used by aggregators was conducted, revealing the relationship between fees, liquidity, routing, and slippage.

Keywords: DEX, DEX aggregators, linear programming, optimization, slippage, gas costs, conversion rates, liquidity, blockchain.

Índice de Contenidos

Agradecimientos	III
Resumen	IV
Abstract	V
Índice de Contenidos	VI
Lista de Tablas	VIII
Lista de Figuras	IX
Glosario	XI
Introducción	1
1. Definición del Problema	2
2. Estado del Arte	5
2.1. Proof-of-Work	5
2.2. Costos y problemáticas de Proof-of-Work	6
2.3. Proof-of-Stake	6
2.4. Gas y tarifas	7

2.5. Dex y pools de liquidez	8
2.6. Modelado por enrutamiento lineal	9
2.7. Agregadores Dex (Dex Aggregators)	11
2.8. Modelado por enrutamiento dividido	12
2.9. Funcionamiento del protocolo 0x	16
3. Propuesta	18
3.1. Metodología	24
4. Implementación	27
4.1. Datos Iniciales	27
4.2. Parámetros Aleatorios	29
4.3. Modelo multi-ruta	29
4.4. Modelo para Simulación por Rama	32
5. Resultados	35
5.1. Discusión	44
6. Conclusiones	46
7. Anexos	49
7.1. Matcha y su operativa <i>Gasless Trading</i>	49
7.2. Ataques Sandwich y CowSwap	53
7.3. Llenado parcial y dinámico	54
Bibliografía	57

Índice de tablas

5.1. Valores de B Promedio Obtenidos	37
5.2. Relación entre B promedio, Deslizamiento, Liquidez y Ruta	40

Índice de figuras

1.1. Árbol del problema	4
2.1. Ganancias mensuales totales de mineros durante un ciclo de mercado alcista.	6
2.2. Ruta lineal con Uniswap	9
2.3. Grafo de enrutamiento lineal.	10
2.4. Diagrama de funcionamiento de los Dex Aggregators.	12
2.5. Enrutamiento dividido con multi-saltos.	14
2.6. Ejemplo de enrutamiento dividido en la interfaz de 1inch.	15
2.7. Enrutamiento lineal con multi-saltos.	17
2.8. Enrutamiento dividido con saltos individuales.	17
3.1. Ejemplo de swap entre token A y B a través del pool A-B.	18
3.2. Ejemplo de representación gráfica del enrutamiento dividido y costos de gas. Los subíndices en los costos de gas $g_{i,j}$ indican a cuál transacción de intercambio corresponde cada costo.	21
3.3. Ejemplo de swap con modelo aplicado.	23
5.1. 10 pools disponibles.	35

5.2. B obtenidos por simulación multi-rama y por ramas individuales.	38
5.3. Relación entre deslizamiento y B promedio obtenido.	42
5.4. Relación entre deslizamiento y B promedio obtenido.	43
7.1. Cotización normal en Matcha	51
7.2. Cotización con Gasless Trade activado	52
7.3. Ejemplo de Sandwich attack en listado de transacciones del par HEX/USDC.	53
7.4. Modelo de swap con Partial y Dynamic Filling.	55

Glosario

Agregadores Dex: Plataformas o herramientas que exploran múltiples Dex en busca de las mejores tasas de cambio para una transacción específica. Estos agregadores optimizan las transacciones de los usuarios al proporcionar una visión comprensiva de las opciones disponibles y sugerir la ruta más rentable para el intercambio de criptomonedas.

Blockchain: Base de datos distribuida en nodos en una red global. La información está recopilada en grupos conocidos como bloques, estos contienen conjuntos de información. Los bloques tienen ciertas capacidades de almacenamiento y, cuando se llenan, se cierran y vinculan al bloque previamente llenado, formando una cadena de datos.

dApp: *Decentralized Application* (Aplicación Descentralizada) es un tipo de aplicación de software de código abierto distribuido. Este se ejecuta en una red blockchain en lugar de en una sola computadora.

Deslizamiento (*Slippage*): Es la diferencia entre el precio esperado de una transacción y el precio al cual la transacción se ejecuta finalmente. En el contexto de los Dex, el deslizamiento puede ocurrir debido a la volatilidad del mercado o cuando la cantidad de una operación es lo suficientemente grande como para afectar directamente el precio del activo. Los usuarios suelen establecer un porcentaje de deslizamiento aceptable para garantizar que sus operaciones se realicen dentro de márgenes de precio tolerables, pero también corriendo el riesgo de que la operación no se ejecute si el deslizamiento supera el umbral establecido.

Dex: *Decentralized Exchange* (Intercambio descentralizado) es la evolución tecnológica de un Exchange tradicional, se traslada todo el funcionamiento al Blockchain ejecutándose sobre Smart Contracts, con los cuales se maneja todo dentro de la plataforma de manera descentralizada.

Exchange: Espacio virtual en el que se realiza acciones de compra-venta de criptomonedas, el usuario (*trader*) puede obtener ganancias gracias a las variaciones de precio que se dan en el mismo.

Gas: Medida de la cantidad de trabajo necesario para ejecutar operaciones en la Blockchain, como hacer transacciones o ejecutar contratos inteligentes. En el contexto de Ethereum y otras blockchains, el gas se refiere al costo que un usuario incurre para realizar estas operaciones. Esencialmente, es una tarifa que los usuarios pagan a los mineros para que incluyan sus transacciones en la Blockchain.

Peer-to-peer: Red de comunicación descentralizada entre pares sin necesidad de un servidor central, cada parte o usuario puede tener la función de servidor o de cliente.

Pools de liquidez: Conjuntos de fondos (generalmente en forma de criptomonedas) bloqueados en un contrato inteligente. Estos fondos se utilizan para facilitar el comercio y proporcionar liquidez a los Dex. Los proveedores de liquidez depositan pares de tokens en estos pools a cambio de tarifas de transacción generadas por el trading que ocurre en el pool.

Programación Lineal: Método matemático utilizado para optimizar la asignación de recursos limitados entre diferentes actividades, buscando maximizar o minimizar una función objetivo lineal, sujeto a un conjunto de restricciones lineales. Esta técnica es ampliamente utilizada en operaciones de investigación para resolver problemas de decisión complejos.

Smart Contracts: Programas almacenados en un Blockchain que se ejecutan al cumplir predeterminadas condiciones.

Swap: Función de los Dex donde se facilita el intercambio instantáneo entre dos criptomonedas dentro del mismo Blockchain, sin la necesidad de cambiar a dinero tradicional, o hacia pares de intercambio determinados que ofrecen los Exchange tradicionales.

Introducción

Actualmente, el ecosistema financiero está experimentando una transformación sin precedentes con el surgimiento de las Finanzas Descentralizadas (DeFi). A diferencia de las plataformas que le preceden, estas permiten a los usuarios realizar transacciones y acceder a servicios financieros sin la intervención de intermediarios centralizados como las plataformas de intercambios tradicionales. Dentro de este ámbito, los intercambios descentralizados (DEX) han surgido como una alternativa prominente, otorgando mayor control, autonomía y seguridad a los usuarios sobre sus activos.

Para el caso de los DEX, el caso de estudio a analizar, plantea desafíos novedosos. Las tasas de conversión, las comisiones, deslizamiento y otros factores varían ampliamente entre las diferentes rutas de transacción y plataformas. En este contexto, la optimización de transacciones se convierte en una necesidad primordial para quienes buscan maximizar sus retornos en el universo DeFi.

Mediante un modelo de programación lineal, se busca encontrar rutas de transacción óptimas que consideren las complejidades y variables del mercado. Junto a esto, se analiza en detalle la relación entre liquidez, tasas de conversión y deslizamiento para proporcionar una visión completa del funcionamiento de los agregadores DEX.

El objetivo principal de esta investigación es modelar y comprender en profundidad el funcionamiento de los agregadores DEX y sus mecánicas subyacentes que les permiten utilizar múltiples rutas de transacción, optimizando así la rentabilidad de las operaciones. Adicionalmente, se pretende validar estas funciones en diversos escenarios de mercado utilizando variables aleatorias para probar si las transacciones son tanto efectivas como rentables.

Capítulo 1

Definición del Problema

Las blockchains funcionan como bases de datos distribuidas y descentralizadas, operando sobre un modelo *peer-to-peer* (P2P). En un sistema P2P, cada participante (denominado "nodo" o "peer") interactúa directamente con otros sin la necesidad de un intermediario central. Esta estructura favorece una distribución equitativa del poder y del control, donde las transacciones, representadas como datos, están organizadas y almacenadas en bloques. Estos bloques se conectan secuencialmente, formando una cadena que es compartida y consensuada entre todos los participantes de la red. Según el paper "Position Paper on Blockchain Technology: Smart Contract and Applications" [8], la naturaleza y la implementación de estas aplicaciones Blockchain se pueden desglosar en diferentes capas: la de consenso, que se encarga de garantizar que todos los participantes tengan una visión unificada y acordada de la cadena; de estructura de datos, que organiza cómo se registran y almacenan los datos; y de aplicación, que determina cómo se interpretan y utilizan esos datos.

Una de las innovaciones más prominentes en el mundo Blockchain es el concepto de Smart Contracts. Estos contratos inteligentes son programas autoejecutables con las instrucciones del contrato directamente escritas en líneas de código. Ethereum, creado en 2015, es uno de los sistemas Blockchain pioneros en ofrecer una plataforma completa para la creación y ejecución de Smart Contracts. Para mantener la integridad y seguridad de este sistema, los mineros juegan un papel crucial validando y registrando transacciones en el Blockchain. A cambio de este trabajo computacional, reciben comisiones, cuyos valores fluctúan en función

de la demanda y la congestión de la red [7].

En este ecosistema nace Uniswap, que tiene el honor de ser el primer Intercambio Descentralizado o Dex (por sus siglas en inglés). Los Dex son plataformas que permiten a los usuarios intercambiar criptomonedas directamente entre sí, sin la necesidad de intermediarios. Sin embargo, con la creciente popularidad de Uniswap, las comisiones y deslizamiento por intercambio comenzaron a experimentar grandes fluctuaciones.

Cuatro años después de la creación de Ethereum, la comunidad comenzó a darse cuenta de las ineficiencias y altas fluctuaciones en tarifas y deslizamiento en plataformas como Uniswap. Ante esto, llegó una propuesta creativa para reducir estos problemas del swap de monedas: los Dex Aggregators. 1inch se destacó como el primer agregador Dex [4]. Estas plataformas funcionan escaneando múltiples Dex en busca de las mejores tasas de intercambio para el usuario, optimizando la ruta de transacción y, en muchos casos, ahorrando en comisiones y bajando el deslizamiento.

Sin embargo, las comisiones no solo dependen de las tasas de intercambio, sino también de la congestión de la red y demanda de transacciones. Además del problema de las comisiones, el deslizamiento en los intercambios también representa una preocupación significativa para los usuarios el cual tiene que ver con la liquidez de los Dex. El deslizamiento se refiere a la diferencia entre el precio esperado de una operación y el precio al que la operación se ejecuta finalmente, y puede ser causado por la volatilidad del mercado o la falta de liquidez. Los usuarios, al estar expuestos a estas fluctuaciones, a menudo establecen un margen de tolerancia tanto para las comisiones como para el deslizamiento, garantizando que la transacción se realice dentro de un rango aceptable de costos.

Es crucial abordar las ineficiencias de los Dex y sus agregadores. A largo plazo, si no se abordan estos problemas, la adopción y usabilidad de estas plataformas podrían verse amenazadas. Además, los usuarios podrían alejarse debido a las altas tarifas o transacciones que fallan debido a la volatilidad de las comisiones y problemas de deslizamiento.

Ante este panorama, el objetivo principal de esta memoria es investigar profundamente la eficiencia de los Dex Aggregators, en particular en la red Ethereum, pero también en otros blockchains con capacidades de Smart Contracts. Mediante técnicas de programación lineal,

se busca exponer su funcionamiento interno para ver como se maximiza específicamente el retorno de una transacción.

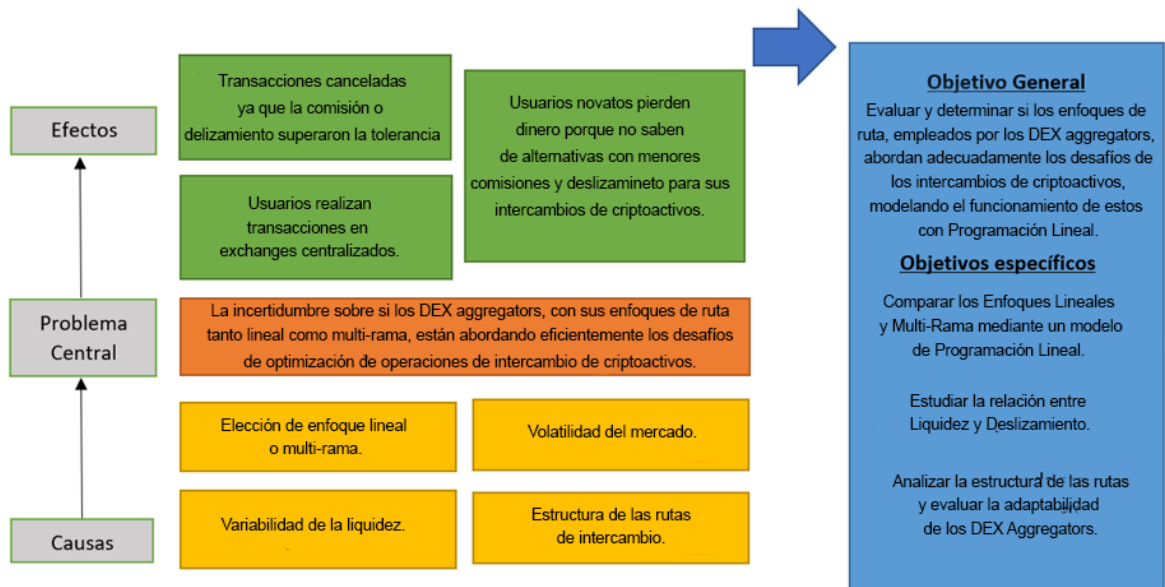


Figura 1.1: Árbol del problema

Fuente: Elaboración propia.

Capítulo 2

Estado del Arte

2.1. Proof-of-Work

Las aplicaciones Blockchain se pueden dividir en varias capas, principalmente: consenso, estructura de datos y la aplicación. La capa de consenso garantiza que distintos nodos de la red compartan la misma vista de la cadena de bloques actual, y determina quién está autorizado para agregar un nuevo bloque a la cadena de registro.

Uno de los principales algoritmos de consenso para crear y validar bloques es *Proof-of-Work*, implementado inicialmente en el Blockchain de Bitcoin y en la versión inicial de Ethereum. En este, para añadir un nuevo bloque a la cadena, se utilizan algoritmos de minería o funciones hash criptográficas que realizan múltiples intentos hasta encontrar la solución correcta por fuerza bruta. Este proceso, llevado a cabo por los mineros, garantiza el mantenimiento del mecanismo de consenso y la descentralización de la cadena de bloques. La dificultad para encontrar la solución se ajusta en función de la cantidad de mineros, con el objetivo de generar un bloque cada cierto tiempo, variando según la cadena de bloques.

2.2. Costos y problemáticas de Proof-of-Work

Según el artículo “Why Proof-of-Work Is Not Viable in the Long-Term” [9] el principal problema observado es que Proof-of-Work tiene costos de operación sustancialmente más altos debido a la depreciación de la tecnología; como el hardware mejora continuamente, los mineros se ven obligados a actualizar constantemente su equipamiento para mantenerse competitivos. Esto eleva la dificultad del problema criptográfico a resolver y también incrementa las comisiones por transacción en la Blockchain.

Los mineros priorizan validar las transacciones que conllevan una mayor comisión, por lo tanto, si al aumentar la dificultad de criptografía van mejorando el hardware usado, se aumentan el precio de todas las operaciones en la Blockchain.



Figura 2.1: Ganancias mensuales totales de mineros durante un ciclo de mercado alcista.

Fuente: cryptorank.io

2.3. Proof-of-Stake

Para reducir el requerimiento computacional y energético de Proof-of-Work, surge *Proof-of-Stake*. En este algoritmo, los usuarios bloquean en depósito (*staking*) tokens para recibir recompensas provenientes de las comisiones de cada transacción y operación. Los nodos que

han depositado mayor cantidad de monedas y por más tiempo son favorecidos para validar el siguiente bloque. A su vez, para evitar que grandes nodos dominen una Blockchain, la antigüedad de las monedas depositadas se restablece a cero una vez que un nodo ha forjado un bloque. Como recompensa, el nodo recibe las tarifas de transacción asociadas con las transacciones en el bloque y esta se divide para cada usuario de manera proporcional a la cantidad de monedas que depositó en el nodo [1].

2.4. Gas y tarifas

En la red Ethereum, se usa una unidad llamada "Gas" para medir el esfuerzo computacional requerido para ejecutar diversas operaciones. Las tarifas de Gas se pagan en ETH, la moneda nativa de Ethereum, y se expresan en unidades de gwei, donde 1 gwei equivale a 10^{-9} ETH. Gwei es una contracción de "giga-wei", donde 1 gwei es 10^9 wei, siendo el wei la unidad mínima de ETH, cuyo nombre proviene de Wei Dai, el creador de B-Money ¹.

Para realizar cualquier transacción en Ethereum, los usuarios deben tener ETH adicional en su billetera para pagar la tarifa de Gas. Por ejemplo, si un usuario desea transferir 1 ETH y sólo tiene 1 ETH en su billetera, no será capaz de cubrir la tarifa de Gas y la transacción no será posible.

Considérese un caso en el que un usuario quiere enviar 1 ETH a otro. Si se establece un límite de Gas de 21000 unidades para la transacción y el precio del Gas es de 200 gwei, la tarifa total en gwei sería el producto de las unidades de Gas y el precio del Gas. Esto se muestra en la ecuación a continuación:

$$\text{Unidades Gas límite} * \text{Precio del Gas} \leq \text{Tarifa total}$$

$$21.000[\text{Gas}] * 200[\text{gwei/Gas}] \leq 4.200.000[\text{gwei}]$$

$$4.200.000[\text{gwei}] = 0.0042[\text{ETH}]$$

¹Wei Dai propuso B-Money en 1998 como un sistema de efectivo electrónico anónimo y distribuido, precursor de las criptomonedas modernas [6]

Así, el remitente tendría 1.0042 ETH menos en su billetera, el destinatario recibiría 1 ETH, y el minero que validó la transacción recibiría 0.0042 ETH (la tarifa de Gas).

El límite de Gas es una estimación del trabajo total requerido para una transacción. Si muchas personas están usando la red, un usuario puede aumentar su límite de Gas para incentivar a los mineros a validar su transacción primero. Sin embargo, si el límite de Gas es demasiado bajo, la transacción puede no completarse, en cuyo caso el usuario pierde los ETH gastados en Gas. Por otro lado, si la transacción se completa antes de llegar al límite, el usuario recupera los ETH no gastados en Gas [?].

2.5. Dex y pools de liquidez

Las dApps más populares actualmente corresponden a los exchanges descentralizados (Dex). Los usuarios mantienen la custodia de sus claves privadas/billeteras y los intercambios se realizan interactuando directamente con smart contracts, los cuales registran cada intercambio directamente en la Blockchain.

Los Dex funcionan mediante protocolos autónomos llamados *Automated Market Makers* que utilizan smart contracts para definir el precio de los activos digitales y proveer liquidez. Estos smart contracts son denominados *pools de liquidez* (reservas de liquidez).

Las pools de liquidez son fundamentales para hacer posible el trading descentralizado. Los usuarios que proveen liquidez dejan en depósito (*staking*) un par de tokens en cantidades de valor equivalente en una reserva (pool), facilitando así las funciones de intercambio entre ellos en el Dex. Los proveedores de liquidez reciben comisiones a partir de los intercambios que se realizan en la pool que aportaron, de manera proporcional a su participación en la liquidez total.

Dado que no todos los pares de tokens cuentan con su propia pool de liquidez para realizar un intercambio directo, a veces el Dex debe realizar intercambios intermediarios para poder cambiar de un token a otro. Esto incrementa el costo del gas en la transacción total, ya que técnicamente se están realizando múltiples intercambios. Para abordar este problema y

optimizar el costo del gas, los Dex pueden buscar el camino óptimo dentro de sus pools de liquidez mediante un algoritmo de enrutamiento lineal.

2.6. Modelado por enrutamiento lineal

El enrutamiento lineal es el proceso de obtener un camino de intercambio óptimo, donde cada par a intercambiar pasa solo por una Liquidity Pool. Por ejemplo, si un usuario quiere intercambiar ETH por USDC, el camino óptimo que encontró el algoritmo puede ser cambiar ETH por USDT en Uniswap con su propio pool, y luego USDT por USDC en PancakeSwap con su propio pool. En total, el usuario pasaría por dos pools, que pueden estar (o no) en dos protocolos diferentes.



Figura 2.2: Ruta lineal con Uniswap

Fuente: app.uniswap.org

Este problema puede modelarse de manera abstracta con un grafo dirigido, utilizando la nomenclatura del artículo [3]. Cada intercambio o *swap* puede representarse como un

camino desde un nodo origen i a un nodo final f . El costo de cada arista está asociado al Gas estimado, tal como se muestra a continuación:

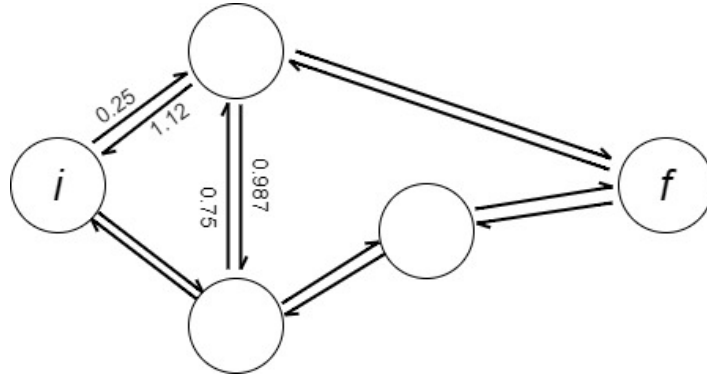


Figura 2.3: Grafo de enrutamiento lineal.

Fuente: Elaboración propia.

Entonces, el problema se puede resumir como: desde el token inicial i , encontrar un camino que minimice el costo hacia el token final f . Esto se asemeja a los problemas típicos del camino más corto, pero con la peculiaridad de que el costo de cada arista está asociado al camino que se encuentra por delante en la secuencia de nodos.

Por lo tanto, en el camino óptimo de nodos que se van agregando a la cola, cada nodo tiene un estado que necesita mantenerse en tiempo real, provocando que el estado de los nodos subsiguientes se mantengan con el estado de los predecesores. Por consiguiente, el peso mínimo final no se calcula sumando el costo de todas las aristas del camino, si no que calculando el costo de entrada del nodo final f .

Por último, el peso mínimo final no se calcula sumando el costo de todas las aristas del camino, sino calculando el costo de entrada del nodo final f . Cuando hay pocos nodos, es más intuitivo utilizar el algoritmo DFS (Búsqueda en Profundidad), que recorre todos los caminos y obtiene el precio final de $toToken$. Uniswap V2 utiliza este método. El modelo mencionado anteriormente no es adecuado para el algoritmo BFS (Búsqueda en Anchura), debido a su condición FIFO (*First In, First Out*).

El uso del algoritmo DFS puede garantizar la ruta óptima de i a f del grafo actual, con una complejidad $O(n^3)$, siendo n el número total de aristas.

2.7. Agregadores Dex (Dex Aggregators)

Los agregadores Dex son plataformas que permiten a los usuarios intercambiar criptomonedas utilizando diferentes Dex en una misma interfaz. En lugar de tener que visitar cada Dex individualmente para verificar las tarifas de Gas, los usuarios pueden dejar que el Dex Aggregator cotice automáticamente una ruta de intercambio entre diferentes pares y Dex. Esto se debe a que el intercambio entre cada par de monedas puede tener un costo de gas diferente, al igual que los Dex. Ante esto, el objetivo es llegar al token deseado con una tarifa total más baja que si se realizara el cambio de forma directa en un Dex individual.

Los agregadores Dex como 1inch y 0x Protocol, además de contar con sus propias pools de liquidez, también consideran las de otros Dex al momento de calcular una ruta para intercambiar un token por otro. Los Dex populares como Balancer y Uniswap V3 también dividirán una transacción del usuario en múltiples rutas, con un algoritmo de enrutamiento que se basa en sus propias pools de liquidez. En cambio, los agregadores Dex hacen un uso más amplio de la liquidez de la Blockchain. Esto proporciona a los usuarios la mejor cotización posible, ya que permite acceder tanto a sus propios pools como a los de otros Dex, aumentando la fuente de liquidez de la Blockchain en su totalidad. Asimismo, introducen el enrutamiento dividido, donde el token inicial se divide en las distintas pools creando ramas de intercambios.

A modo de ejemplo, un usuario desea realizar un swap de USDT por el token nativo de 1inch (1INCH), el aggregator de 1inch primero separa la cantidad inicial de USDT del usuario entre los pools de USDT-USDC de Uniswap y Balancer (mediante enrutamiento dividido), luego toma los USDC resultantes de ambos intercambios y los cambia por DAI en Balancer, ya que es el único Dex que cuenta con esta pool. Finalmente, se cambia DAI por 1INCH en el propio pool de liquidez de 1inch y se le transfieren a la billetera del usuario.

Los agregadores Dex han ido aumentando en popularidad a medida que el interés en las finanzas descentralizadas (DeFi) ha crecido.

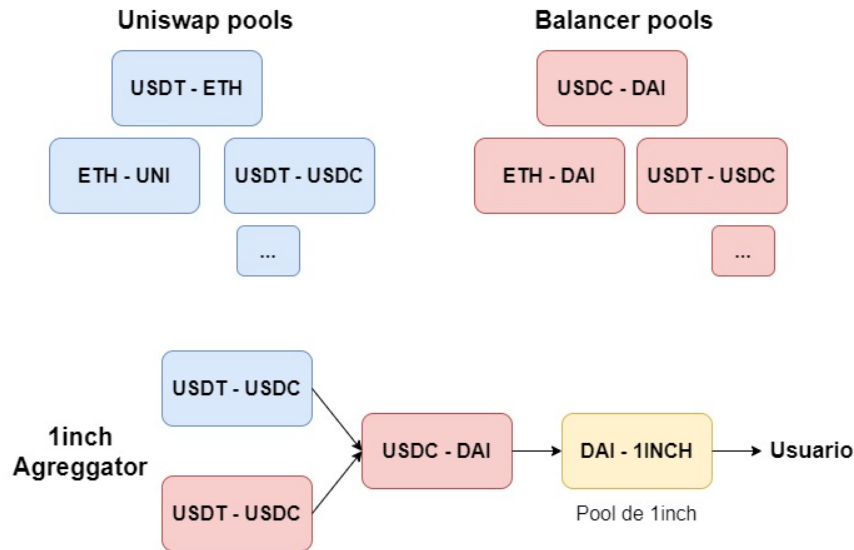


Figura 2.4: Diagrama de funcionamiento de los Dex Aggregators.

Fuente: Elaboración propia.

Según [2], “el primer Dex Aggregator se creó como parte de una competencia de hackathon en 2019. Los participantes en la competencia, sintieron que necesitaban la información agregada de numerosos Dex, pero esto aún no estaba disponible como servicio. Por consiguiente, Sergej Kunz y Anton Bukov, crearon el PMV de 1inch en el lapso de 18 horas durante el hackathon. Por supuesto, la versión inicial del aggregator no era completamente funcional y pulida, sin embargo, este fue un ejemplo perfecto de por qué los agregadores Dex eran un servicio necesario.”. Este acontecimiento se ve con más detalle en el artículo “What Are agregadores Dex? A Deep Dive by 1inch” [4].

2.8. Modelado por enrutamiento dividido

Cómo funciona 1inch

En la base de 1inch se encuentra el algoritmo conocido como *Pathfinder*, una API de detección y encargada de encontrar las mejores rutas para un swap utilizando un enrutamiento dividido. Este enfoque también reduce la probabilidad de transacciones fallidas.

Cuando un usuario realiza una solicitud de swap, Pathfinder primero divide la solicitud entre varias rutas principales (designadas como *mainRouteParts* en la API). Para minimizar la probabilidad de transacciones fallidas, 1inch implementa las funciones de *Dynamic Filling* (Llenado dinámico) y *Partial Filling* (Llenado parcial).

Dynamic Filling se encarga de detectar problemas técnicos o cambios en el libro de órdenes de los intercambios descentralizados (Dexs). Además, permite re-rutar automáticamente las órdenes a través de diferentes intercambios para garantizar una ejecución eficiente.

Si una rama falla debido a problemas técnicos o cambios en el libro de órdenes, Dynamic Filling detecta el problema y busca una nueva ruta para la orden. En caso de que la ruta fallida esté en un pool incompatible con otra rama, Dynamic Filling buscará una nueva ruta que incluya intercambios compatibles.

Por ejemplo, si una ruta fallida incluía un intercambio que solo admite ciertas criptomonedas, Dynamic Filling buscará una nueva ruta que incluya intercambios que admitan la misma criptomoneda. Esto garantiza que la orden se ejecute con éxito y se eviten problemas adicionales.

Por otro lado, si la rama fallida no puede ser redirigida a las ramas favorables, se devuelve al usuario la cantidad correspondiente de la ficha inicial de esa rama, y las demás ramas completan sus intercambios (Partial Filling). Esta última función debe estar habilitada en la interfaz, de lo contrario, se cancelarán todas las ramas y se devolverá al usuario la totalidad de la ficha inicial.

Una rama puede fallar por varias razones, incluyendo:

1. Problemas técnicos: Los Dexs pueden experimentar problemas técnicos, como interrupciones del servicio, problemas de conectividad o errores en el software que pueden afectar la capacidad de una ruta para ejecutar órdenes de manera satisfactoria.
2. Cambios en el libro de órdenes: Los Dexs suelen tener un libro de órdenes, donde se registran las compras y ventas de criptomonedas. Si hay un cambio en el libro de órdenes, como una gran cantidad de compra o venta, este puede afectar las tarifas y la capacidad de una ruta para ejecutar órdenes de manera satisfactoria.

3. Liquidación de una posición: Una ruta de intercambio puede fallar si un usuario ha liquidado una posición en una criptomoneda y no hay suficiente liquidez para cubrir esa posición, causando una caída en el precio y el cierre de la ruta.

Al calcular la ruta óptima, la cantidad total del token inicial i puede dividirse en diferentes rutas o pools, creando una red de flujo dividiendo la cantidad total del token en n partes.

Aunque Pathfinder no excluye nodos duplicados (diferentes caminos a través del mismo token), evita el problema de utilizar el mismo pool dos veces al dividirlos en sub-rutas. Si esto se aplica a un gráfico, las sub-rutas separadas por el mismo conjunto de arcos no deben seleccionarse repetidamente.

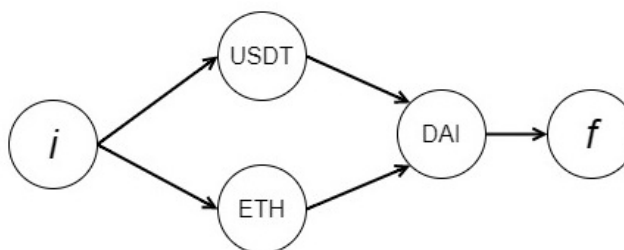


Figura 2.5: Enrutamiento dividido con multi-saltos.

Fuente: Elaboración propia.

De forma similar al caso anterior, el costo final mínimo depende directamente del costo de entrada del nodo final.

Debido a que el costo también depende de la ruta pasada, el algoritmo de camino aumentado no es aplicable para resolver el costo mínimo y flujo máximo. Otro problema es que este método limita el mismo pool solo para tomar una de las rutas secundarias, por lo que no se puede calcular la suma entre las rutas secundarias, además la ruta aumentada no puede simplemente invertir el flujo.

El enrutamiento dividido en linch tiene la siguiente estructura:

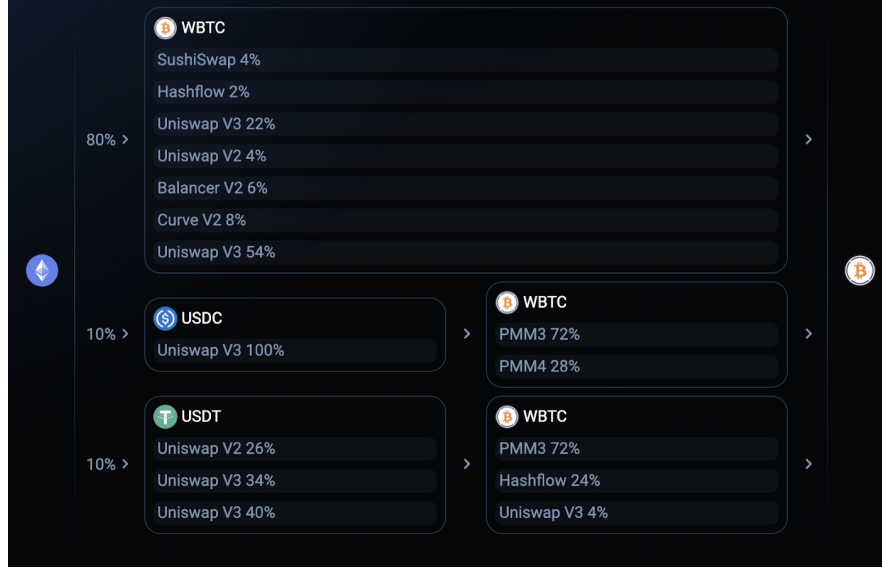


Figura 2.6: Ejemplo de enrutamiento dividido en la interfaz de 1inch.

Fuente: 1inch.io

El funcionamiento de los agregadores Dex puede modelarse de acuerdo con el artículo de referencia [3]. En este modelo, la cantidad total del token de origen, A_o , se divide en n caminos, representado por la siguiente fórmula:

$$A_o = \sum_{i=1}^n a_{oi}$$

De manera similar, la cantidad total del token final, A_f , que también se distribuye a través de n caminos, se representa de la siguiente manera:

$$A_f = \sum_{i=1}^n a_{fi}$$

En este proceso, se cuenta con un total de P pools que se distribuyen entre todas las ramas i :

$$P = \sum_{i=1}^n t_i - 1$$

Donde $t_i - 1$ representa la cantidad total de pools en cada rama i y t_i el total de tokens de cada rama.

Cada camino puede pasar por diferentes pools de diferentes exchanges/protocolos, y dentro de cada uno puede haber más intercambios. Cada uno de estos puede representarse con la función: $f(a)$ donde cada rama convierte una cantidad total de tokens iniciales a_{oi} en una cantidad final de tokens a_{fi} , representado por:

$$\{a_{fi}\}, i \in [1, \dots, n]$$

Usando el modelo mencionado, a_{fi} puede expresarse directamente como:

$$a_{fi} = f_{i1} (f_{i(t_i-2)} (\dots f_{i(t_i-1)} (a_{oi})))$$

En el artículo se menciona que .esta solución se puede interpretar como la búsqueda de rutas no repetitivas de costo mínimo”[3]. Esto significa que se buscan las rutas de intercambio que minimizan el costo (como el slippage y las tarifas), evitando el uso de la misma ruta más de una vez. Esto es especialmente útil en situaciones donde se dispone de múltiples opciones de intercambio, ya que permite obtener el mejor valor posible de la transacción.

Además, se menciona que este modelo .es una versión simplificada que toma en cuenta situaciones específicas, como cuando dos rutas de transacción diferentes convergen en el mismo nodo”[3]. En términos más concretos, este escenario puede ocurrir cuando distintas rutas de intercambio a través de varias pools y tokens acaban implicando la misma pool o token en algún punto.

2.9. Funcionamiento del protocolo 0x

El protocolo 0x aspira a proporcionar una plataforma para intercambiar todo tipo de tokens en la blockchain Ethereum. Además de esto, dispone de su propio Dex que utiliza los

algoritmos de enrutamiento mencionados anteriormente, pero con algunas modificaciones propias.

El Dex de 0x simplifica la búsqueda de cotizaciones en dos modelos separados, cada uno calculando su cotización óptima. Luego se selecciona la opción más favorable para el usuario.

El primer modelo corresponde a un enrutamiento lineal con múltiples "saltos", utilizando diferentes pools del protocolo 0x.

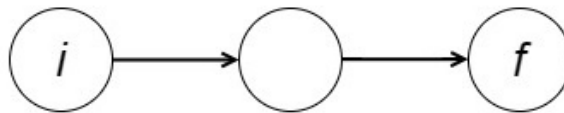


Figura 2.7: Enrutamiento lineal con multi-saltos.

Fuente: Elaboración propia.

El segundo modelo es un enrutamiento dividido que utiliza solo un Dex o protocolo por rama, donde cada uno se considera un salto individual.

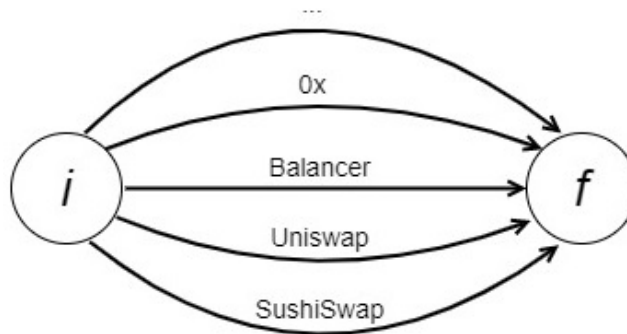


Figura 2.8: Enrutamiento dividido con saltos individuales.

Fuente: Elaboración propia.

En su funcionamiento, 0x invoca la función de enrutamiento de Dexs como Uniswap o Sushiswap para cada rama, tratando cada uno como un pool independiente. Esto ocurre a pesar de que, internamente, estos Dexs pueden realizar enrutamientos lineales con múltiples saltos intermedios para la operación de intercambio.

Capítulo 3

Propuesta

En esta sección, se propondrá un modelo basado en [3] para los agregadores Dex, con algunos cambios significativos para adaptarse a las condiciones específicas de este estudio. Estas modificaciones incluyen la consideración del costo de gas para cada intercambio intermediario y las tasas de cambio. Asimismo, el cambio de notación del token inicial A_o que será representado por A , y el token final A_f como B .

A continuación, se expone en la figura un swap directo donde no se involucran rutas intermedias ni múltiples saltos:

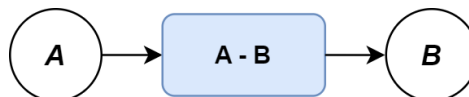


Figura 3.1: Ejemplo de swap entre token A y B a través del pool A-B.

Fuente: Elaboración propia.

En el diagrama simplificado:

- **Token A:** Es la criptomoneda que el usuario desea intercambiar.
- **Pool A-B:** Facilita el intercambio directo entre Token A y Token B. Este pool mantiene un balance de ambos tokens.

- **Token B:** Es la criptomoneda que el usuario desea obtener a cambio de Token A.

Este diagrama muestra un swap directo de un token inicial, denotado como Token A, a un token final, denotado como Token B, mediante un único pool de liquidez. No se involucran rutas intermedias ni múltiples saltos. Es importante destacar que la tasa de cambio entre los tokens A y B está determinada por la liquidez presente en el pool y la demanda del mercado en el momento del intercambio.

Considerando un intercambio, el usuario proporciona una cierta cantidad del token original, denotada por A. Este token se intercambia en el pool por una cantidad equivalente del token final, denotada por B. Esta transacción puede representarse a través de la siguiente función:

$$f(A) = B \quad (3.1)$$

Donde f es la función que representa un intercambio dentro del pool, determinando la cantidad de *tokens* B que el usuario recibe a cambio de sus *tokens* A. Esta función f se basa en la reserva actual de *tokens* A y B en el pool, así como en la tasa de cambio actual entre A y B.

Dado que en los agregadores Dex el token inicial A puede ser dividido en n ramas para optimizar el intercambio (y cada rama con uno o más pools), cada rama parte con un total de *tokens* iniciales a_{oi} , representado por:

$$A = \sum_{i=1}^n a_{oi} \quad (3.2)$$

Cada rama convierte una cantidad total de *tokens* iniciales a_{oi} en una cantidad final de *tokens* a_{fi} , representado por:

$$a_{fi}, \quad i \in [1, \dots, n] \quad j \in [1, \dots, t_i - 1] \quad (3.3)$$

Con este modelo, también se puede expresar a_{fi} como una serie de aplicaciones de la funciones de intercambio f_{ij} :

$$a_{fi} = f_{i1} (f_{i(t_i-2)} (\cdots f_{i(t_i-1)} (a_{oi}))) \quad (3.4)$$

Donde $t_i - 1$ representa la cantidad total de pools (o intercambios) en cada rama i y t_i el total de tipos de tokens distintos en cada rama.

De manera similar, B es la cantidad total del token que se obtiene al final del intercambio, que corresponde a la cantidad obtenida a través de todos los n caminos.

$$B = \sum_{i=1}^n a_{fi} \quad (3.5)$$

Además, el intercambio puede pasar por múltiples pools de liquidez en cada ruta. El total de los pools de todo el proceso (la suma de los pools entre todas las rutas) se denota por P :

$$P = \sum_{i=1}^n t_i - 1 \quad (3.6)$$

Como observación, se tiene a P como cota superior de n , ya que nunca habrá más ramas que pools totales en el proceso.

Aquí, f_{ij} representa la función de intercambio en el pool j en la ruta i . Esta función toma una cantidad de tokens a_{oi} y proporciona una cantidad de *tokens* a_{fi} , basándose en las condiciones del mercado y las reglas específicas del pool de liquidez.

Es crucial tener en cuenta que cada uno de estos intercambios incurre en un costo de gas, que denotamos como g_i .

El grafo que sigue, ilustra el proceso de dividir el token original A en 2 fracciones a_{oi} para ser enviadas a través de dos rutas ($n = 2$) y así obtener el token de destino B , donde la primera ruta ($i = 1$) es un intercambio directo y la segunda ($i = 2$) pasa por el token intermediario C , cada intercambio pasando por un pool, incurriendo en sus respectivos costos de gas $g_{i,j}$,

en donde i es la rama (ruta de intercambio) y j representa el intercambio específico en esa rama (de un token a otro). Por lo tanto, el subíndice i, j corresponde a la transacción de intercambio específica:

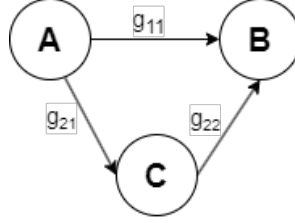


Figura 3.2: Ejemplo de representación gráfica del enrutamiento dividido y costos de gas. Los subíndices en los costos de gas $g_{i,j}$ indican a cuál transacción de intercambio corresponde cada costo.

Fuente: Elaboración propia.

Si para el intercambio j de la ruta i , se denota a o_{ij} como el token de origen y a d_{ij} como el token de destino. Deben cumplirse las siguientes condiciones:

- El token de origen para el primer intercambio en cualquier ruta es una fracción del token inicial de todo el proceso, es decir, $o_{i1} = a_{oi}$.
- El token de destino para el último intercambio en cualquier ruta es una fracción del token final de todo el proceso, es decir, $d_{i(t_i-1)} = a_{fi}$.
- La unidad de r_{ij} es la unidad de d_{ij}/o_{ij} . Esto implica que la tasa de conversión r_{ij} se expresa en términos de las unidades del token de destino dividido por las unidades del token de origen.

Por consiguiente, la cantidad de tokens B recibidos a través de la ruta i , denotada por a_{fi} , se calcula como:

$$a_{fi} = a_{oi} \cdot \prod_{j=1}^{t_i-1} r_{ij} \quad (3.7)$$

Aquí, r_{ij} es la tasa de cambio entre los tokens consecutivos en la ruta i . Esta tasa se aplica a la cantidad de tokens que se intercambian en cada paso de la ruta, y luego se resta el costo del gas total para esa ruta, el cual ya está en unidades de token final B.

Cada intercambio dentro de un pool de liquidez implica un costo de gas g_{ij} , medido en la unidad nativa de la red (por ejemplo ETH en la red Ethereum). Si se tienen t_i tokens en la rama i , habrá $t_i - 1$ intercambios en esa rama, y por lo tanto $t_i - 1$ costos de gas, considerando que cada pool de liquidez permite el intercambio de un token por otro y se suman los costos del gas entre todos los intercambios.

Para una ruta específica i , la suma de todos estos costos de gas se denota como G_i . Esta se calcula sumando los costos de gas $g_{i,j}$ para cada intercambio j en la ruta:

$$G_i = \sum_{j=1}^{t_i-1} g_{ij} \quad (3.8)$$

Cabe mencionar que estos costos de gas se deben mantener dentro de un límite predefinido γ , el cual fue establecido por el usuario al inicio, al momento de aceptar la transacción.

El costo total de gas para todo el proceso se calcula sumando los costos de gas G_i de cada ruta y debe satisfacer la restricción $G \leq \gamma$:

$$G = \sum_{i=1}^n G_i \quad (3.9)$$

$$G \leq \gamma \quad (3.10)$$

$$G = \sum_{i=1}^n \sum_{j=1}^{t_i-1} g_{i,j} \quad (3.11)$$

donde n es el número total de rutas.

En este modelo, el costo total de gas G para una transacción puede ser significativo, especialmente si hay muchas rutas y pools involucrados.

A modo de ejemplo, aplicando el modelo a un swap exitoso desde una cantidad A de ETH hacia una cantidad B de WBTC con enrutamiento dividido y multi-saltos:

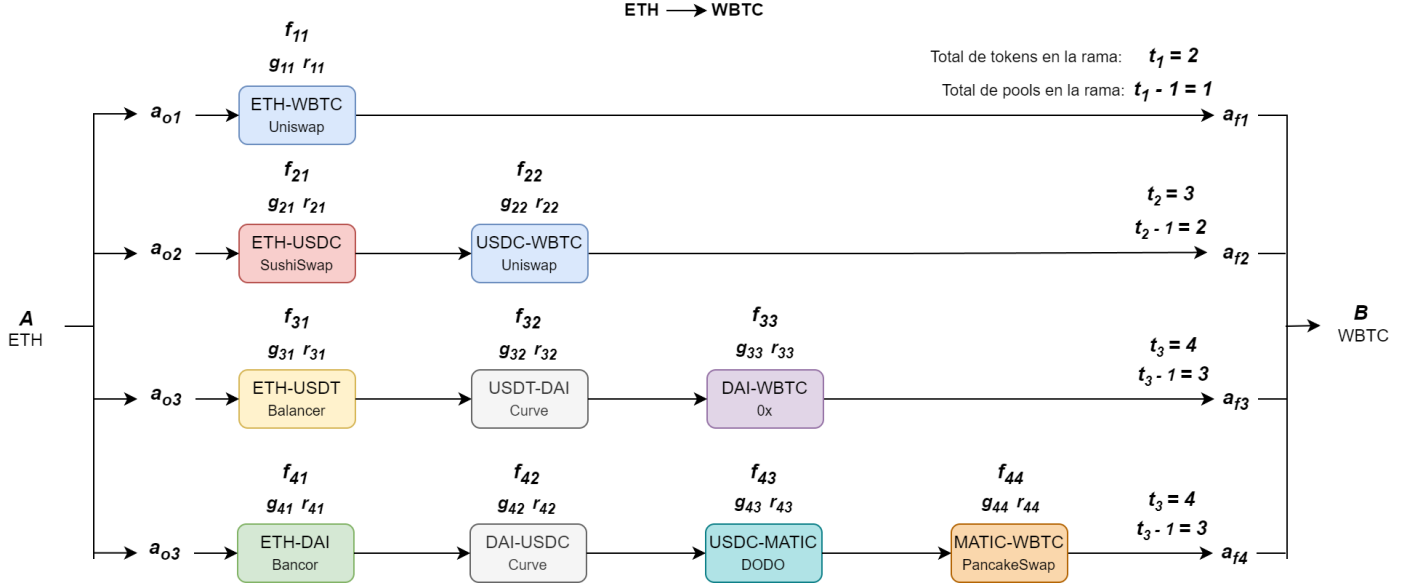


Figura 3.3: Ejemplo de swap con modelo aplicado.

Fuente: Elaboración propia.

Para encontrar la ruta óptima a través de todos los pools de liquidez disponibles, es necesario resolver el siguiente problema de optimización:

Función objetivo:

$$\text{Maximizar } B \quad (3.12)$$

Sujeto a:

$$A = \sum_{i=1}^n a_{oi} \quad (3.13)$$

$$B = \sum_{i=1}^n a_{fi} \quad (3.14)$$

$$G \leq \gamma \quad (3.15)$$

$$G = \sum_{i=1}^n G_i \quad (3.16)$$

$$G_i = \sum_{j=1}^{t_{i-1}} g_{ij} \quad (3.17)$$

$$a_{fi} = a_{oi} \cdot \prod_{j=1}^{t_{i-1}} r_{ij} \quad (3.18)$$

Las restricciones aseguran que se conserve el balance total de tokens en cada paso del intercambio, y que los costos de gas para cada transacción individual sean considerados. La función objetivo maximiza la cantidad de tokens B recibidos, menos el costo de gas total.

La ecuación 3.13 establece que la cantidad total de tokens A gastados, es igual a la suma de todos los tokens a_{oi} gastados en cada ruta i .

Por otro lado, en la ecuación 3.14, B se define como la suma de todos los tokens b_{fi} recibidos a través de cada ruta i . Esto significa que la cantidad total de tokens B que se obtienen al final del proceso de intercambio es igual a la suma de todos los tokens b_{fi} que se reciben a través de todas las rutas.

3.1. Metodología

1. Definición de los Datos Iniciales y Recolección de Datos:

- Se iniciará definiendo parámetros básicos del problema, como la cantidad de tokens iniciales (A), límite de gas (γ) y cantidad de ramas (n).
- Se recolectará y compilará toda la información necesaria sobre las tasas de 10 pools de DEX reales (véase la figura 7.4), así como su liquidez total. Estos datos

alimentarán las simulaciones y permitirán modelar el deslizamiento con mayor cercanía a un caso real.

- Se definirá ETH como el token de origen y WBTC como el token de destino para todas las simulaciones.

2. Generación de Datos Aleatorios y Configuración de Pools:

- Generación números aleatorios dentro de rangos realistas, simulando distintas configuraciones de tasas de conversión (dentro de un rango de $\pm 5\%$ basado en cotizaciones reales) al igual que para los costos de gas.¹.
- Se identificarán e instanciarán todas las rutas posibles considerando las combinaciones de los 10 pools.

3. Modelado Matemático e Identificación de Variables:

- Creación de un modelo matemático programado utilizando el lenguaje de programación *Python*, a través de su librería *Pyomo*.
- Se definirán sets, parámetros y variables que reflejen las particularidades y restricciones del problema, incluyendo una simulación del deslizamiento basado en la liquidez de los pools.

4. Definición de la Función Objetivo y Restricciones:

- La función objetivo se centra en maximizar la cantidad final de tokens después de todas las transacciones.
- Se incorporan las restricciones pertinentes al modelo, asegurando que se respeten límites como el gas, las características de liquidez y las restricciones de ruta basadas en los 10 pools.
- Se adaptan todas las restricciones matemáticas del modelo propuesto a código Python utilizando el paquete Pyomo.

5. Implementación de la Simulación:

¹Las tasas de conversión y los valores de gas en ETH para swaps fueron obtenidos el 20 de septiembre entre las 14:00 y 15:00 hrs.

- Se configura un bucle de simulación para ejecutar el modelo 1000 veces, utilizando diferentes conjuntos de datos aleatorios y considerando todas las rutas posibles de transacción entre los 10 pools.
- Para simular de manera más precisa el comportamiento interno de un agregador DEX, se incorpora la simulación del deslizamiento (slippage), ya que este es uno de los principales motivos de la existencia de algoritmos de enrutamiento dividido. Esta simulación se hará en función de la liquidez de los pools, para reflejar condiciones más cercanas al mercado real.
- En cada iteración, se compara la estrategia de distribución usando múltiples ramas con una estrategia de enviar todo el token A por cada una de las ramas.

6. Resolución del Modelo y Análisis:

- Utilización del *solver GLPK* para la resolución de problemas de programación lineal, con el fin de determinar la solución óptima en cada simulación.
- Compilación y análisis de los 1000 resultados de la simulación, donde se extraerán y presentarán los resultados obtenidos del modelo.

Capítulo 4

Implementación

El modelo es programado a través del lenguaje de programación *Python*, por medio de la librería de *Pyomo*. Este modelo tiene como objetivo maximizar el valor de los tokens obtenidos, tras una serie de transacciones en diferentes ramas o DEXs, teniendo en cuenta factores reales como los costos de gas, las tasas de conversión y el deslizamiento.

4.1. Datos Iniciales

- A : Cantidad de tokens iniciales que se desea intercambiar. Para esta simulación se utiliza un valor de 100.
- γ : Límite de gas dispuesto a gastarse para todas las transacciones. Se utiliza un valor de 0.05.
- n : Número total de ramas disponibles. Se utilizan 9 lo cual es el máximo de ramas posibles generadas con los 10 pools.
- m : Número máximo de transacciones que se pueden realizar en una rama. Se le asignó el largo de la rama más larga generada, correspondiente a 5.
- ETH_PRICE : Valor aproximado de ETH en USD para el cálculo del deslizamiento. Tiene un valor asignado de 1600.

- *MAX_SLIPPAGE*: Porcentaje de deslizamiento máximo por rama. Se le asignó el valor de 0.05 correspondiente a un 5 %.
- *start_token*: Nombre del token de origen para la configuración de rutas. Se le asignó el string "ETH".
- *end_token*: Nombre del token de destino para la configuración de rutas. Se le asignó el string "WBTC".
- *Pool*: Clase para instanciar las pools donde se tienen los atributos:
 - *token_origen*: String de nombre del token de origen que acepta el pool.
 - *token_destino*: String de nombre del token de destino que arroja el pool.
 - *nombre_dex*: String de nombre del Dex al que pertenece el pool.
 - *cotizacion_rango*: Tupla con floats que representan las tasas de conversión dentro de un rango de $\pm 5\%$ con respecto al valor intermediario.
 - *liquidez_total*: Representa la liquidez total del pool en USD. Es importante destacar que los datos de liquidez se obtuvieron directamente de *dexscreener*. Para asegurar que el efecto del deslizamiento fuera perceptible y proporcionara resultados claros y significativos en las simulaciones, se realizó una escala en la magnitud de estos datos por un factor de 10. Esto permite una visualización más efectiva de sus implicaciones en el deslizamiento.

```

1 pool1 = Pool("ETH", "WBTC", "Uniswap", (0.05447565, 0.0605285), 8200000)
2 pool2 = Pool("ETH", "USDC", "SushiSwap", (1549.5, 1712.6), 1797000)
3 pool3 = Pool("USDC", "WBTC", "Uniswap", (0.000034865, 0.000038535), 2300000)
4 pool4 = Pool("ETH", "USDT", "Balancer", (1540.3, 1702.4), 33700000)
5 pool5 = Pool("USDT", "DAI", "Curve", (0.950038, 1.050042), 2500000)
6 pool6 = Pool("DAI", "WBTC", "0x", (0.000035074, 0.000038766), 3560000)
7 pool7 = Pool("ETH", "DAI", "Bancor", (1534.35, 1695.8), 7300000)
8 pool8 = Pool("DAI", "USDC", "Curve", (0.94992305, 1.04991495), 5880000)
9 pool9 = Pool("USDC", "MATIC", "DODO", (1.8233749, 2.0153091), 4500000)
10 pool10 = Pool("MATIC", "WBTC", "PancakeSwap", (0.0000189335, 0.0000209265),
    6200000)

```

4.2. Parámetros Aleatorios

Se simulan aspectos del entorno de transacciones generando números aleatorios para:

- Costos de gas ($g_{ij,data}$) por transacción en cada rama dentro de un rango entre 0,0075 ETH y 0,015 ETH. Al momento de la recopilación de datos el precio de ETH era aproximadamente \$1600 USD, por lo que el precio del gas varía entre \$12 USD y \$24 USD.
- Tasas de conversión ($r_{ij,data}$) por transacción en cada pool dentro de un rango de $\pm 5\%$.

4.3. Modelo multi-ruta

El modelo general en Pyomo se estructura de la siguiente manera:

1. Conjuntos y Parámetros:

- `model.I`: Conjunto que representa las diferentes ramas o rutas.
- `model.J`: Conjunto que representa las diferentes transacciones en una ruta.
- `model.g_ij`: Parámetro que representa el costo de gas de una transacción específica (i, j).
- `model.r_ij`: Parámetro que indica la relación de intercambio entre tokens para una transacción específica (i, j).
- `model.L_i`: Parámetro que determina la cantidad de tokens a_oi a enviar por rama.

2. Variables:

- `model.a_o` y `model.a_f`: Variables que denotan los tokens al inicio y al final de una ruta, respectivamente.
- `model.G`: Variable que señala el costo total de gas para todas las transacciones.

3. Función Objetivo: Maximizar la cantidad de tokens al final de todas las rutas, teniendo en cuenta las tasas de intercambio.

4. Restricciones:

- `model.total_tokens`: La cantidad total de tokens al inicio en todas las rutas debe ser igual a A .
- `model.gas_used`: El gas total usado en todas las transacciones debe ser igual a `model.G`.
- `model.gas_limit`: El gas total usado no debe exceder un límite predefinido, γ .
- `model.final_tokens`: Define la relación entre los tokens al inicio y al final en cada ruta, considerando las tasas de intercambio.
- `model.a_o_limit`: El número de tokens al inicio de una ruta no debe exceder el máximo posible por ruta.

Manejo del Slippage

El *slippage* es una variación en el precio esperado de una transacción debido a la falta de liquidez. Para controlar su impacto, se sigue un proceso iterativo:

1. **Cálculo de slippage por transacción:** La función `calculate_slippage` tiene como objetivo calcular el *slippage* de una transacción dada una cantidad específica de activos y la liquidez total disponible en el exchange. Matemáticamente, la función se define como:

$$\text{calculate_slippage}(a, \text{liquidez_total}) = \frac{a \cdot \text{ETH_PRICE}}{\text{liquidez_total}} \quad (4.1)$$

donde:

- a es la cantidad de tokens que se quieren intercambiar.
- ETH_PRICE es el valor aproximado de ETH en USD. Tiene un valor asignado de 1600.
- $liquidez_total$ es la liquidez total disponible en el exchange para esa transacción.

Por lo tanto, el *slippage* se calcula como la proporción del valor de la liquidez que se quiere intercambiar (cantidad de tokens multiplicado por su precio) respecto a la liquidez total disponible. Un slippage alto indica una proporción significativa de los activos en comparación con la liquidez total, lo que puede conducir a variaciones de precio durante la ejecución de la transacción.

2. **Ajuste de Tasas de Cambio:** Para cada transacción j en cada ruta válida i , se calcula el slippage. Luego, se ajusta la tasa de cambio en r_{ij} según:

$$r_{ij,modified} = r_{ij} \cdot (1 - \text{slippage}(j))$$

Este proceso compone los efectos del slippage de todas las transacciones en una ruta.

3. **Identificación de Rutas con Slippage Excesivo:** Para cada ruta, se calcula el slippage compuesto usando la fórmula:

$$S_{ruta} = 1 - \left(\prod_{pool \in ruta} (1 - \text{slippage}(pool)) \right)$$

donde el slippage para cada transacción en la ruta se calcula con la función `calculate_slippage`. Las rutas cuyo deslizamiento supere el valor de `MAX_SLIPPAGE` se consideran inválidas.

4. **Redistribución de Tokens:** Las rutas inválidas se descartan y los tokens asignados a estas se suman y redistribuyen entre las rutas válidas de forma equitativa.
5. **Verificación de Condiciones de Salida:** Se verifica si todas las rutas válidas aún tienen un slippage dentro del rango aceptable. Si alguna ruta válida ahora excede el slippage máximo, el proceso se repite desde el paso 2.
6. **Reporte de Slippage:** Se imprime el slippage total para cada ruta válida.

4.4. Modelo para Simulación por Rama

La función `simulate_branch` simula la optimización para una rama o ruta específica. La estructura es similar al modelo general, pero está adaptada para considerar una sola rama a la vez.

1. Entrada:

- `index`: Indica el índice de la rama o ruta que se está simulando.

2. Conjuntos y Parámetros:

- `model_single_branch.J`: Conjunto que representa las transacciones en la ruta especificada por el índice j .
- `model_single_branch.gij` y `model_single_branch.rij`: Parámetros que representan el costo de gas y la relación de intercambio, respectivamente, para las transacciones en la ruta dada.
- `total_slippage`: Cálculo del deslizamiento total para la ruta como un porcentaje.

3. Variables: Similar al modelo general, pero adaptado para una sola rama.

4. Función Objetivo: Maximizar la cantidad de tokens al final de la rama considerando tasas de intercambio y deslizamiento.

5. Restricciones: Adaptaciones de las restricciones del modelo general para una sola rama, incluyendo el deslizamiento.

6. Solver: Se utiliza igualmente el solver GLPK para encontrar la solución óptima para la rama especificada.

7. Salida:

- La función retorna detalles clave de la solución, como la cantidad de tokens finales, tasa de conversión promedio, deslizamiento y gas usado.

Aplicación del Deslizamiento en la Tasa de Cambio

Para aplicar el deslizamiento compuesto en la tasa de cambio final en una rama específica, primero se calcula el deslizamiento total para la ruta dada:

$$\text{total_slippage} = \text{slippage_specific_route}(\text{all_routes}[\text{index}], A) \quad (4.2)$$

La función `slippage_specific_route` está diseñada para calcular el deslizamiento total a través de una ruta específica, utilizando la misma lógica que para el modelo multi-rama.

Dado:

- Una ruta específica, denotada como `route`, que es una secuencia de intercambios o pools.
- Una cantidad inicial de tokens a , el cual corresponde la cantidad A , ya que se envían todos los tokens por rama de manera individual.

El deslizamiento compuesto para la ruta, `slippage_for_route`, se inicializa en 1. Esto se hace para utilizarlo como multiplicador acumulativo a lo largo del proceso.

Para cada intercambio en la ruta:

$$\text{slippage} = \frac{a \cdot \text{ETH_PRICE}}{\text{pool.liquidez_total}} \quad (4.3)$$

Donde `exchange.liquidez_total` representa la liquidez total del intercambio actual.

El deslizamiento compuesto acumulado se actualiza multiplicando el deslizamiento acumulado anterior por $(1 - \text{slippage})$:

$$\text{slippage_for_route} \leftarrow \text{slippage_for_route} \cdot (1 - \text{slippage}) \quad (4.4)$$

Finalmente, después de recorrer todos los intercambios en la ruta, el deslizamiento compuesto para la ruta completa se obtiene como:

$$\text{slippage_for_route} = 1 - \text{slippage_for_route} \quad (4.5)$$

Una vez calculado el deslizamiento total, la tasa de intercambio compuesta ajustada por deslizamiento para la ruta es:

$$\text{adjusted_rate} = r_{ij_data}[\text{index}] \cdot (1 - \text{total_slippage}) \quad (4.6)$$

Se aplica el deslizamiento total directamente en la función objetivo como un factor que multiplica la tasa neto de la rama.

La función objetivo, teniendo en cuenta las tasas de intercambio y deslizamiento, que busca maximizar la cantidad de tokens al final de la rama, se define como:

$$\text{maximize } a_{oi} \cdot \prod_{j \in J} r_{ij}[j] \cdot (1 - \text{total_slippage}) \quad (4.7)$$

Donde:

- a_{oi} es el número inicial de tokens.
- $r_{ij}[j]$ es la tasa de intercambio para la transacción j .
- J es el conjunto de todas las transacciones en la ruta.

Capítulo 5

Resultados

A continuación, se expondrán todas las posibles ramas generadas con los 10 pools disponibles:

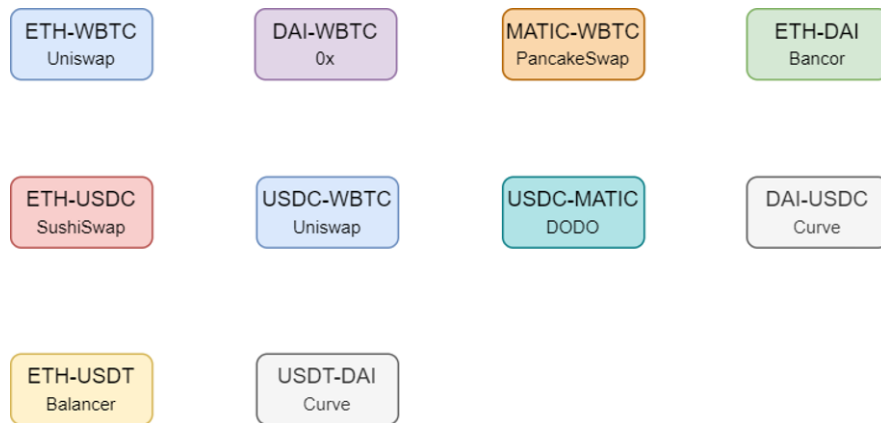


Figura 5.1: 10 pools disponibles.

Fuente: Elaboración propia

1. Ruta 1:

- ETH → WBTC en Uniswap con rango de tasas: (0.056882865, 0.062870535)

2. Ruta 2:

- ETH → USDC en SushiSwap con rango de tasas: (1549.5, 1712.6)

- USDC → WBTC en Uniswap con rango de tasas: $(3.4865 \cdot 10^{-5}, 3.8535 \cdot 10^{-5})$

3. Ruta 3:

- ETH → USDC en SushiSwap con rango de tasas: (1549.5, 1712.6)
- USDC → MATIC en DODO con rango de tasas: (1.8233749, 2.0153091)
- MATIC → WBTC en PancakeSwap con rango de tasas: $(1.89335 \cdot 10^{-5}, 2.09265 \cdot 10^{-5})$

4. Ruta 4:

- ETH → USDT en Balancer con rango de tasas: (1540.3, 1702.4)
- USDT → DAI en Curve con rango de tasas: (0.950038, 1.050042)
- DAI → WBTC en 0x con rango de tasas: $(3.5074 \cdot 10^{-5}, 3.8766 \cdot 10^{-5})$

5. Ruta 5:

- ETH → USDT en Balancer con rango de tasas: (1540.3, 1702.4)
- USDT → DAI en Curve con rango de tasas: (0.950038, 1.050042)
- DAI → USDC en Curve con rango de tasas: (0.94992305, 1.04991495)
- USDC → WBTC en Uniswap con rango de tasas: $(3.4865 \cdot 10^{-5}, 3.8535 \cdot 10^{-5})$

6. Ruta 6:

- ETH → USDT en Balancer con rango de tasas: (1540.3, 1702.4)
- USDT → DAI en Curve con rango de tasas: (0.950038, 1.050042)
- DAI → USDC en Curve con rango de tasas: (0.94992305, 1.04991495)
- USDC → MATIC en DODO con rango de tasas: (1.8233749, 2.0153091)
- MATIC → WBTC en PancakeSwap con rango de tasas: $(1.89335 \cdot 10^{-5}, 2.09265 \cdot 10^{-5})$

7. Ruta 7:

- ETH → DAI en Bancor con rango de tasas: (1534.35, 1695.8)

- DAI → WBTC en 0x con rango de tasas: $(3.5074 \cdot 10^{-5}, 3.8766 \cdot 10^{-5})$

8. Ruta 8:

- ETH → DAI en Bancor con rango de tasas: (1534.35, 1695.8)
- DAI → USDC en Curve con rango de tasas: (0.94992305, 1.04991495)
- USDC → WBTC en Uniswap con rango de tasas: $(3.4865 \cdot 10^{-5}, 3.8535 \cdot 10^{-5})$

9. Ruta 9:

- ETH → DAI en Bancor con rango de tasas: (1534.35, 1695.8)
- DAI → USDC en Curve con rango de tasas: (0.94992305, 1.04991495)
- USDC → MATIC en DODO con rango de tasas: (1.8233749, 2.0153091)
- MATIC → WBTC en PancakeSwap con rango de tasas: $(1.89335 \cdot 10^{-5}, 2.09265 \cdot 10^{-5})$

Rama	Valor Promedio
Multi-Rama	5.94
Rama 1	5,63
Rama 2	5,07
Rama 3	5,33
Rama 4	5,32
Rama 5	5,02
Rama 6	5,27
Rama 7	5,56
Rama 8	5,24
Rama 9	5,54

Tabla 5.1: Valores de B Promedio Obtenidos

En la implementación del modelo para el análisis de rendimiento en las ramas, se estudiaron 1000 instancias recopiladas de las simulaciones. Ante esto, se identifica una mejora

significativa utilizando el enfoque multi-rama, en comparación con el rendimiento de la mejor rama individual por instancia, favoreciendo el 80.5 % de los resultados.

En promedio, el rendimiento multi-rama ilustró una mejora del 1.97 % en comparación al mejor resultado obtenido por una rama individual para cada una de las simulaciones.

A continuación, el gráfico expuesto muestra la cantidad de B obtenida entre las 1000 simulaciones para todas las ramas. Además, se puede observar el eje x, que representa el número de simulaciones, mientras que el eje y, muestra la cantidad de B obtenida. Finalmente, cada línea de color corresponde a una rama en específico, incluyendo la simulación Multi-Rama.

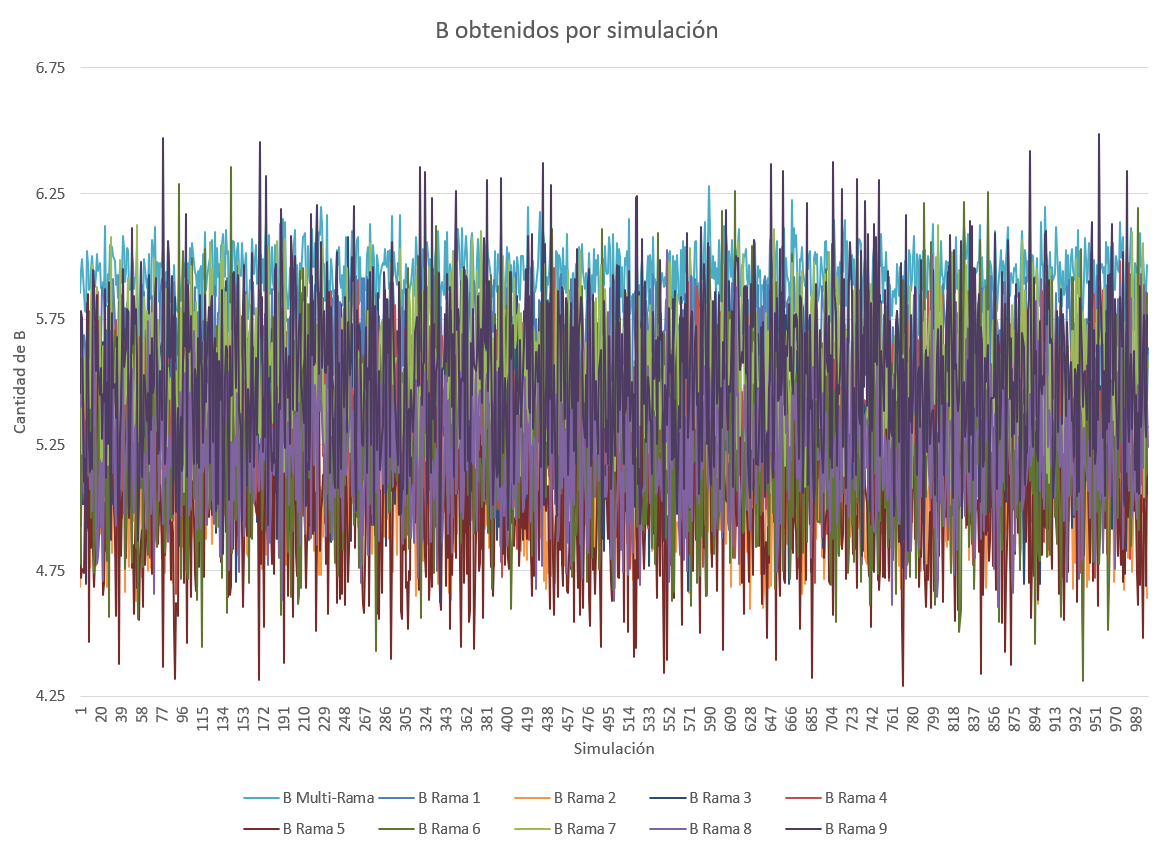


Figura 5.2: B obtenidos por simulación multi-rama y por ramas individuales.

Fuente: Elaboración propia

En el gráfico expuesto anteriormente, se destacan las presencias de picos prominentes a lo largo de todas las ramas. Estos picos representan valores altos y bajos de B que se obtuvieron

en ciertas simulaciones, además indica situaciones en las que, a pesar del deslizamiento, el enrutamiento lineal resulta ser más efectivo. Sumado a lo anterior, esto podría atribuirse a los extremos del rango de las tasas de conversión, de forma favorable y desfavorable (potenciado en el borde inferior por el deslizamiento).

No obstante, la naturaleza dinámica de los mercados de criptomonedas implica que no siempre es favorable optar por una estrategia multi-rama, es decir, en algunos escenarios puede ser más óptimo y eficiente enviar una transacción por medio de una ruta única.

Siguiendo la misma línea, estos escenarios, donde es más favorable la utilización de una sola rama, representan casos en los que los DEX aggregators podrían escoger un enrutamiento lineal. Esto puede ocurrir por diversas razones, tales como:

- Tasas de conversión excepcionalmente favorables en un exchange específico.
- Pools con liquidez extremadamente alta que minimizan el deslizamiento, haciendo innecesario dividir el monto a través de múltiples rutas.
- Costos de gas o tarifas asociadas con transacciones multi-rama que, en ciertas circunstancias, pueden superar los beneficios obtenidos de la distribución de liquidez.

Si bien, las tasas de conversión tienen selección aleatoria dentro de un rango determinado en las simulaciones, cabe la posibilidad de que, en algunas de las 195 simulaciones restantes, las condiciones pueden ser tales que una rama individual resultara ser más favorable que el enfoque multi-rama. No obstante, estas situaciones son excepcionales y no rige la norma.

La afirmación expuesta anteriormente es respaldada, ya que el enfoque multi-rama fue favorable en el resto de las 805 simulaciones, representando un amplio margen de eficiencia y optimización en la gran mayoría de los casos.

Análisis de Rutas de Intercambio

A continuación, la tabla manifiesta la cantidad de B promedio que se obtuvo. Está ordenado de mayor a menor, junto al deslizamiento y liquidez total sumada entre todos los pools de cada ruta específica.

B promedio	Deslizamiento	Liquidez	Ruta
5,63	1,95 %	8.200.000	Ruta 1
5,56	6,59 %	10.860.000	Ruta 7
5,54	10,60 %	23.880.000	Ruta 9
5,33	14,41 %	12.497.000	Ruta 3
5,32	11,03 %	39.760.000	Ruta 4
5,27	14,86 %	52.780.000	Ruta 6
5,24	11,47 %	15.480.000	Ruta 8
5,07	15,24 %	4.097.000	Ruta 2
5,02	15,68 %	44.380.000	Ruta 5

Tabla 5.2: Relación entre B promedio, Deslizamiento, Liquidez y Ruta

El análisis que se realizó, tomó en cuenta la estructura de las múltiples rutas de intercambio y consideró la liquidez disponible en cada pool, así como el deslizamiento asociado. Algunas de las observaciones alcanzadas por los resultados son:

1. La **Ruta 1** presenta el deslizamiento más bajo, del 1,95 %, al intercambiar ETH por WBTC directamente en Uniswap. Este bajo deslizamiento se atribuye a la liquidez sustancial disponible en este pool, siendo \$8.200.000.
2. Las rutas que involucran múltiples intercambios, como la **Ruta 5** y la **Ruta 6**, tienen deslizamientos más altos, 15,68 % y 14,86 % respectivamente. Estos valores indican que, aunque estas rutas pueden ofrecer una diversidad en el intercambio, a menudo conllevan mayores costos en términos de deslizamiento debido a la multiplicidad de transacciones y variaciones en la liquidez entre los pools.

3. Es notable que el pool en Balancer, que intercambia ETH y USDT, tiene una liquidez muy alta de \$33.700.000. Sin embargo, las rutas que utilizan este pool (Rutas 4, 5 y 6) no ofrecen necesariamente el deslizamiento más bajo. Esto sugiere que, aunque la liquidez es un factor crucial, no es el único determinante del deslizamiento.
4. La **Ruta 7** destaca por combinar un intercambio directo en Bancor (ETH a DAI) con una transacción subsiguiente en 0x (DAI a WBTC). A pesar de involucrar dos transacciones, su deslizamiento es relativamente bajo, del 6,59 %. Esto puede ser atribuido a la liquidez adecuada en ambos pools y a la eficiencia de las tasas ofrecidas.
5. Las rutas que implican intercambios en DODO y PancakeSwap, específicamente las que involucran a MATIC, tienden a tener deslizamientos más altos. Esto puede estar relacionado con la volatilidad y la demanda de MATIC debido a los datos obtenidos, así como con la liquidez específica de estos pools.

El siguiente gráfico ilustra la relación entre el porcentaje de deslizamiento y el B promedio obtenido por cada ruta:

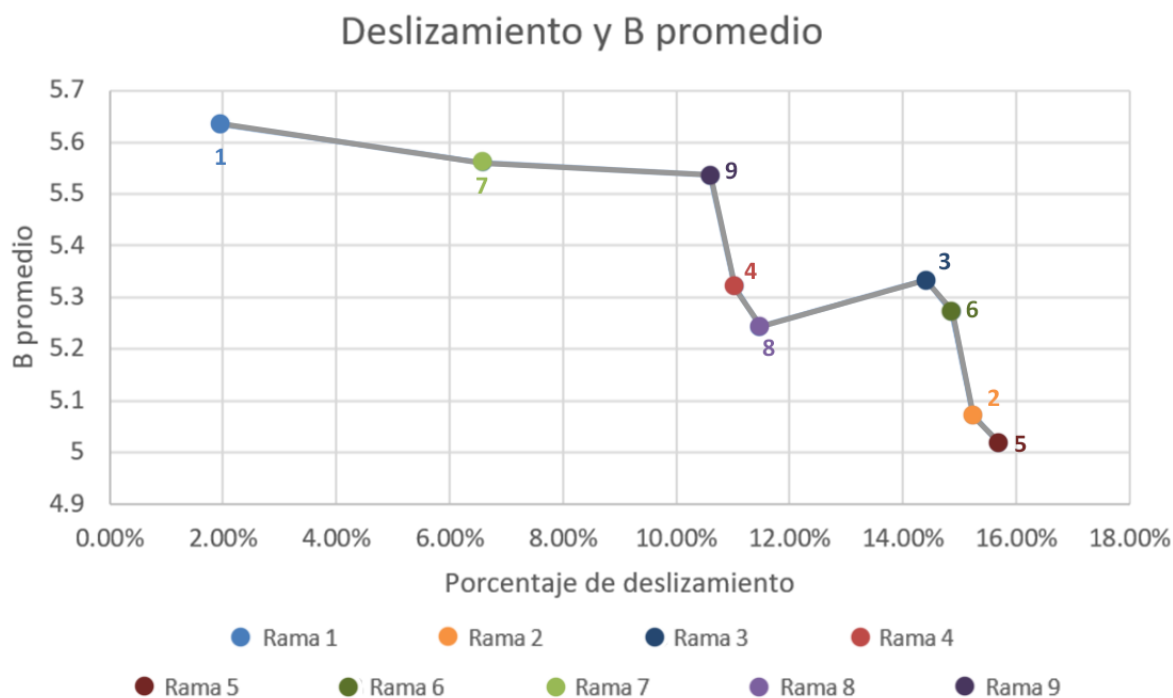


Figura 5.3: Relación entre deslizamiento y B promedio obtenido.

Fuente: Elaboración propia

A medida que el porcentaje de deslizamiento aumenta desde 0 % hasta aproximadamente el 8 %, la cantidad de B obtenido parece mantenerse constante alrededor de 5.6.

El comportamiento constante de B entre el 0 % y 8 % de deslizamiento sugiere que, hasta cierto punto, este no tiene un impacto significativo. No obstante, hay una evidente caída después del 8 %. Esto indica que después hay un umbral, el cual expone cómo el aumento en el porcentaje de deslizamiento tiene efecto negativo. Por otro lado, la recuperación de B, después del 14 %, puede sugerir que hay otros factores que no están reflejados en el gráfico, como lo es la tasa de conversión.

El siguiente gráfico ilustra la relación entre el porcentaje de deslizamiento compuesto por rama y la liquidez total sumada entre todos los pools de cada ruta específica:

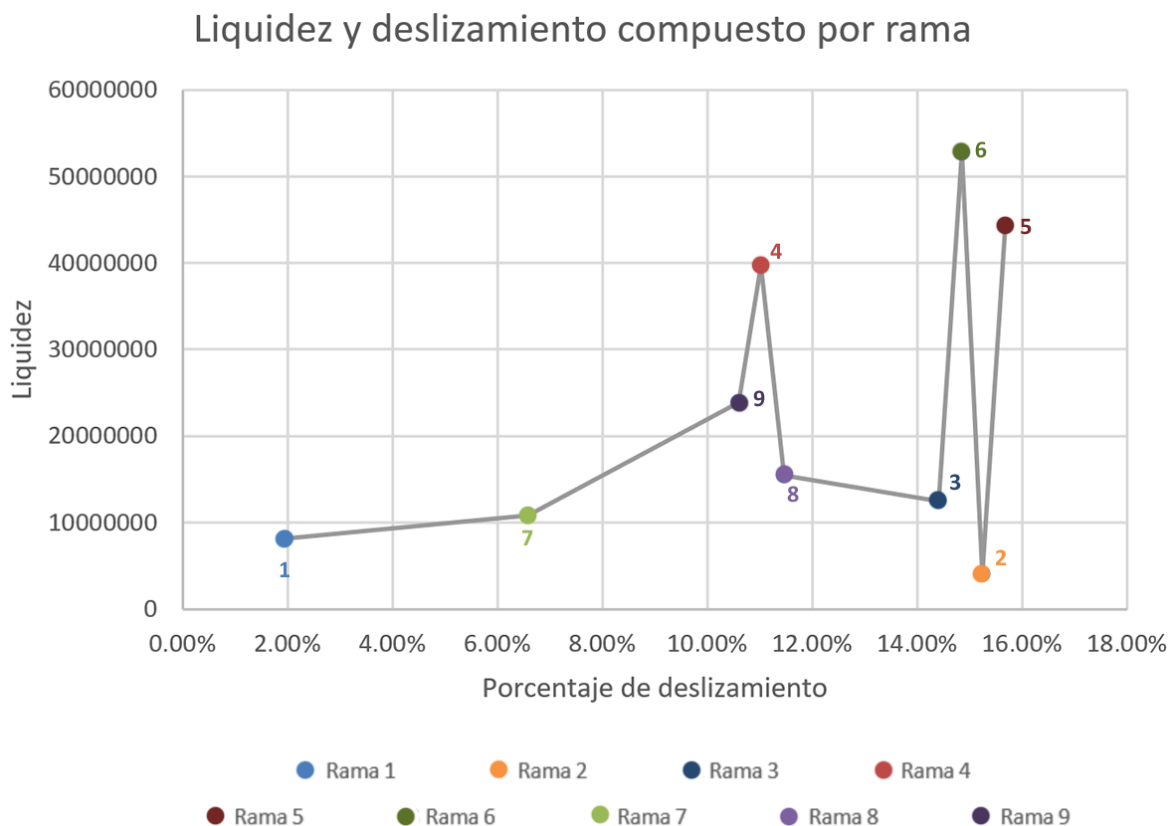


Figura 5.4: Relación entre deslizamiento y B promedio obtenido.

Fuente: Elaboración propia

Al comparar el gráfico de la figura 5.4 con la figura 5.3, se puede observar la compleja relación entre liquidez, deslizamiento y el valor promedio de B. Mientras que se podría esperar que una alta liquidez reduzca el deslizamiento, esto no siempre es el caso. Tomando como ejemplo la Ruta 6: a pesar de ser una de las con mayor liquidez, presenta un alto deslizamiento y un B promedio más bajo. Esto podría deberse a múltiples factores, incluidos la estructura de la ruta y las tasas de intercambio específicas en cada pool. Por otro lado, en el gráfico de deslizamiento y B promedio 5.3, se ve que las rutas con mayor deslizamiento tienden a tener un B promedio más bajo, lo cual es coherente con las observaciones anteriores. Las rutas 2, 3, 5 y 6, por ejemplo, tienen deslizamientos más altos y valores más bajos de B.

La Ruta 1, que tiene una liquidez relativamente baja, muestra un deslizamiento mínimo y un valor de B relativamente alto. Esto podría indicar un enrutamiento eficiente ya

que es la única ruta con un único pool directo entre A y B y una tasa de conversión favorable.

5.1. Discusión

La relación entre el deslizamiento, la liquidez y el valor de B es compleja y multifacética. Mientras que una alta liquidez puede minimizar el deslizamiento, otros factores, como la longitud de la ruta y las tasas de intercambio específicas en cada pool, también juegan un papel crucial.

En ciertas rutas, a pesar de tener múltiples nodos intermedios, la liquidez abundante afirma que las transacciones se completen con eficacia. Por otro lado, hay rutas más directas que, debido a la falta de liquidez, pueden resultar en deslizamientos significativos.

Siguiendo la misma línea, el deslizamiento, como se expuso en datos y gráficos, es una variable crítica en el ecosistema DeFi. Este no solo representa un costo adicional para el usuario en sí, sino que también puede ser un indicativo de problemas subyacentes en la ruta o mercado. Asimismo, las variaciones en el deslizamiento, sobre todo cuando son significativas, pueden ser indicadores de cambios drásticos en la oferta y demanda de un activo en específico.

Al recopilar toda la información disponible, se expone una relación intrínseca entre la estructura de las rutas, liquidez disponible y deslizamiento resultante. No obstante, situaciones donde el deslizamiento es mínimo, a pesar de condiciones adversas como baja liquidez, pueden indicar un enrutamiento eficiente o una favorable tasa de conversión durante esa instancia específica.

En otra arista, una alta liquidez no necesariamente garantiza un deslizamiento bajo si es que la ruta seleccionada no es la mejor opción. Ahora bien, es necesario considerar el enrutamiento y liquidez en conjunto, para así determinar la eficiencia de una transacción y sus costos asociados.

Finalmente, es evidente que, para optimizar las transacciones en una plataforma DeFi, no basta con observar individualmente la liquidez o enrutamiento, es decir, la consideración

conjunta es esencial.

Capítulo 6

Conclusiones

A través de este estudio exhaustivo y analizado sobre los DEX aggregators utilizando un enfoque multi-rama y lineal, se han proporcionado resultados considerables y reveladores sobre la eficiencia en la operación de intercambio de criptoactivos.

En el ámbito de la descentralización y comercio de criptomonedas, es de suma importancia maximizar el rendimiento y eficiencia, especialmente cuando se expone e ilustra la temática de los tokens o pools de baja liquidez.

A partir del análisis cuantitativo, se deja en evidencia que la implementación de un enfoque multi-rama en DEX aggregators, ofrece y brinda ventajas considerables, en comparación a los enfoques más tradicionales como, por ejemplo, el enrutamiento lineal.

Por otro lado, al emplear múltiples rutas o ramas para realizar estas operaciones de intercambio, se obtiene una diversificación que contribuye a mitigar los efectos del deslizamiento. Sumado a lo anterior, esta reducción en el deslizamiento y, en casos particulares, puede ser significativa, especialmente en pools con liquidez limitada.

Lo mencionado anteriormente, se emplea como una ventaja directa para el usuario que realiza estas acciones, ya que tiene la posibilidad de adquirir una cantidad superior de tokens de destino, en comparación a otros métodos más tradicionales o ya expuestos con anterioridad.

Aunque los enfoques multi-rama tienden a ofrecer mejores resultados en términos generales debido a su habilidad para capitalizar la distribución de liquidez entre varios exchanges, hay circunstancias en las cuales un enrutamiento lineal a través de una única rama puede resultar más beneficioso. Es esencial que los DEX aggregators continúen evolucionando y adaptándose para identificar y actuar según estas circunstancias a fin de garantizar siempre la mejor tasa posible para los usuarios.

Esta maximización del rendimiento es de suma importancia y brinda beneficios entendiendo el contexto actual de la industria de criptomonedas donde, las fluctuaciones y volatilidad, son factores que están presentes en todo momento.

Sumado a lo anterior, es de gran importancia la referencia de que esta optimización en los intercambios realizados no solo es un beneficio para los usuarios individuales en sí, sino que, al promover la eficiencia y reducir el deslizamiento ya empleado, se contribuye al fortalecimiento y estabilidad del ecosistema descentralizado en general.

Tras el profundo análisis realizado sobre los Dex Aggregator y los desafíos de liquidez y deslizamiento, se identificaron distintas áreas para trabajos futuros. En primer lugar, es de suma importancia expandir la investigación a otras redes blockchain que admitan Smart Contracts. Asimismo, se destaca la importancia de un estudio más detallado sobre cómo la liquidez y el tamaño del pool influyen en el deslizamiento para obtener percepciones más precisas.

Por otro lado, a medida que el mercado y la tecnología evolucionan, la optimización continua es vital. Se propone desarrollar algoritmos más robustos para la optimización de las operaciones, considerando posiblemente funciones multiobjetivo. En el aspecto del usuario, es importante el cuestionamiento de cómo se mejora la interfaz para educar y guiarlos adecuadamente sobre estos temas y, con estos esfuerzos combinados, se busca un ecosistema más eficiente y amigable para ellos.

A modo de cierre, es de carácter primordial destacar la necesidad de seguir la investigación, desarrollando tecnologías que faciliten y ayuden a optimizar las transacciones creadas

por usuarios en el espacio descentralizado. Por consiguiente, a medida que la industria continúa evolucionando, tal como en los últimos años o, incluso, meses, la demanda de soluciones más eficientes y rentables para la necesidad del usuario, será cada vez más grande y pertinente que antes.

Capítulo 7

Anexos

Los siguientes anexos proporcionan información adicional sobre el funcionamiento del gas y los agregadores Dex, pero sin relacionarse al modelo de optimización de este documento.

7.1. Matcha y su operativa *Gasless Trading*

Matcha es un agregador de Dex construido sobre el protocolo 0x, obteniendo liquidez de otros Dex como Uniswap, Kyber, Curve, etc. Su objetivo es proporcionar a los usuarios comunes la posibilidad de aprovechar la capacidad de agregación de liquidez de 0x.

Aunque Matcha no está limitado a la liquidez de la misma blockchain y puede acceder a fuentes de liquidez *off-chain* de 0x, es relevante mencionar que 0x es en sí mismo un protocolo de agregación de liquidez. Por tanto, Matcha actúa más como una interfaz para 0x.

En septiembre de 2021, Matcha introdujo la función *Gasless Trading*, que permite operar en Ethereum sin incurrir en tarifas de gas y sin *deslizamiento*.¹. Esto es posible gracias a una fuente de liquidez provista por un Automated Market Maker (AMM) dedicado, que opera en paralelo a los pools de liquidez tradicionales y proporciona una cotización de precio

¹Slippage corresponde al ajuste del límite de movimiento del precio cotizado en forma de porcentaje.

individualizada y sin slippage. Una vez que la transacción es firmada, los market makers envían la transacción para su verificación y cubren el costo del gas. Como viene de pools de liquidez dedicados, la opción de Gasless Trade solo está habilitada para el intercambio de pares de tokens específicos; específicamente entre tokens que tengan pares WETH² en las pool.

Para llevar a cabo lo anterior, se utiliza una *meta-transacción*, que permite interactuar con la blockchain desde cuentas o carteras sin ETH.

Las meta-transacciones posibilitan que los usuarios interactúen con una blockchain sin pagar una tarifa de transacción. Esto lleva a una experiencia de usuario más fluida ya que los usuarios no necesitan comprender el funcionamiento interno de la blockchain ni la dinámica del mercado de tarifas de transacción.

Las meta-transacciones son prácticamente idénticas a las transacciones regulares, pero con la adición de un contrato proxy, también conocido como *relayer* (repetidores). En las meta-transacciones, los usuarios utilizan su firma para enviar y autenticar transacciones, pero una vez hecho esto, la transacción es gestionada por el relayer que paga el gas y la envía a la dirección de destino. Esto se logra con la función *permit()* del contrato del token que permite el gasto por parte de otra entidad gracias a la firma del propietario, sin la necesidad de tener ETH en la billetera.

En las meta-transacciones se elabora una transacción similar a una transacción tradicional de Ethereum, pero se agrega información adicional:

- Datos: Información codificada que detalla una lista de acciones a realizar, incluido un hash del nombre de la función a llamar junto con argumentos para pasar a esa función.
- Recompensa: Saldrán del contrato inteligente del proxy intermediario. Se pueden usar ETH o tokens para pagar a los mineros.
- Requisitos: Para verificarse dentro del smart contract. Un ejemplo es una verificación cronológica: La transacción solo puede realizarse a fin de mes.

²WETH es un contenedor de ETH, ambos tienen un valor 1:1. Se utiliza para solucionar problemas de interoperabilidad de ETH ya que fue lanzada antes del estándar ERC-20 por lo que su uso en las DApps y ecosistemas es limitado. Al enviar ETH a un smart contract, se recibe la versión *wrapped* a cambio.

Después, en lugar de enviar la transacción directamente a un contrato inteligente, se envía a un contrato alojado en un proxy.

Finalmente, cuando el smart contract del proxy recibe la transacción enviada por un usuario, se analizarán las instrucciones, se pagará al tercero y se ejecutarán los comandos. Estas instrucciones podrían ser el envío de tokens, llamadas a funciones o cualquier cosa que pueda hacer una transacción normal de blockchain.

En una prueba de Matcha, al cotizar un intercambio entre las *stablecoins*³ USDT y USDC, se observó que al momento de la cotización, con 100 USDT se obtienen aproximadamente 99,88 USDC (debido al arbitraje del mercado), con una tarifa GAS equivalente a \$2,88 USD (la cual se paga aparte en ETH). Si se le resta el valor de la tarifa a lo que se recibe en USDC, resultan aproximadamente \$97 dólares.

The screenshot displays the Matcha trading interface. At the top, there are tabs for 'Market' and 'Limit', with a settings icon to the right. Below this, the 'You Pay' section shows 'USDT' with a dropdown arrow and a value of '100' in a light blue box. A double-headed arrow icon separates this from the 'You Receive' section, which shows 'USDC' with a dropdown arrow and a value of '99.879511' in a light blue box. Above the USDC value, it says '1 USDC = 1.001206 USDT' with an information icon. A large blue button labeled 'Review Order' is positioned below the 'You Receive' section. At the bottom, the 'Network Fee' is listed as '\$2.88', and there is a toggle switch for 'Enable Gasless Trade' which is currently turned off.

Figura 7.1: Cotización normal en Matcha

Fuente: matcha.xyz

³Son 'tokens' que están asociados al valor de una moneda 'fiat' (como el dólar o el euro), a bienes materiales como el oro o los inmuebles, o a otra criptomoneda. También hay 'stablecoins' que no están asociadas a ninguna otra moneda sino que están controladas mediante algoritmos para mantener un precio estable.

Al activar la opción Gasless Trade en Matcha, la necesidad de tener ETH adicional para las tarifas se elimina. Sin embargo, esto tiene un impacto en la cantidad de USDC recibida: en lugar de 99,88, la cifra se reduce a aproximadamente 97,55, con un descuento de \$2,33 USD. Este cambio se atribuye a que se está utilizando una pool de liquidez diferente a través del protocolo 0x, como lo indica el ratio distinto de 1 USDC a 1.025153 USDT.

The screenshot displays the Matcha trading interface. At the top, there are tabs for 'Market' and 'Limit', with a settings icon to the right. Below this, the 'You Pay' section shows 'USDT' with a dropdown arrow and the value '100'. A double-headed vertical arrow icon separates this from the 'You Receive' section. The 'You Receive' section shows 'USDC' with a dropdown arrow and the value '97.546323'. Above this value, it states '1 USDC = 1.025153 USDT' with an information icon. A blue 'Review Order' button is positioned below the 'You Receive' section. At the bottom, the 'Network Fee' is listed as '\$2.73' with 'Free!' in green text next to it. Below that, the 'Enable Gasless Trade' toggle is shown as a blue switch that is currently turned on.

Figura 7.2: Cotización con Gasless Trade activado

Fuente: matcha.xyz

Este descuento de \$2,33 USD es recibido por el smart contract del Automated Market Maker (AMM) y es equivalente al precio del gas, por lo que se distribuye entre los usuarios de la pool de liquidez. La discrepancia entre este monto descontado y el costo del gas cotizado en la primera instancia con Gasless Trade desactivado (\$2,88) se debe a una actualización en el precio del gas en gwei. Debido a que son instancias diferentes y la segunda no incluye slippage, las cifras presentadas difieren.

7.2. Ataques Sandwich y CowSwap

La naturaleza pública de los blockchain permite que todas las transacciones sean visibles en el mempool⁴. Algunos usuarios programan bots para aprovecharse de esta característica, buscando generar ganancias de las transacciones listadas públicamente.

Estos bots analizan la lista de transacciones en busca de órdenes de compra significativas para un token específico. Una vez identificada una "víctima", el bot coloca su orden de compra justo antes de la orden de la víctima, pagando un poco más de gas para asegurarse la prioridad. Esta acción incrementa el precio del token para la víctima y su slippage. Posteriormente, el bot genera una orden de venta coincidiendo con la orden de compra de la víctima para venderle los tokens a un precio más alto. Este tipo de maniobra se conoce como "Sandwich attack", dado que el bot sitúa dos órdenes rodeando a la de la víctima (Ver figura 7.3).

USD ▼	HEX ▼	USDC ▼		PRICE	TXN
449.99	9,928	450.00		\$0.04532	0xa20c
888.15	19,595	888.15		\$0.04532	0xc9d7
1,272.39	28,074	1,272		\$0.04532	0xd4f5
1,390.29	30,677	1,390		\$0.04531	0x1cd5
1,618.44	35,714	1,618		\$0.04531	0xa396
108,832.20	2,407,795	108,832	Bot	\$0.04519	0xdb4d
266,077.14	5,900,482	266,077	Víctima	\$0.04509	0x744a
108,832.20	2,392,754	108,832	Bot	\$0.04548	0xbff1f
660.51	14,399	660.51		\$0.04587	0xf2a1

Figura 7.3: Ejemplo de Sandwich attack en listado de transacciones del par HEX/USDC.

Fuente: Dexscreener.io

Una forma de prevenir los Sandwich attacks es reduciendo manualmente el porcentaje de slippage. Sin embargo, esto puede provocar que la transacción tarde más en completarse o incluso no se realice debido a la falta de liquidez.

⁴El mempool es un mecanismo de un blockchain que almacena información de transacciones no confirmadas, funcionando como una lista de espera para transacciones pendientes de agregarse a un bloque. Para Ethereum, se puede consultar el mempool, el precio del gas, tokens, contratos y toda la información relevante en tiempo real en etherscan.io.

CowSwap es un agregador Dex que aborda la problemática de los Sandwich attacks. Funciona como un agregador de agregadores de Dex y, aparte de competir por cotizar el gas más bajo, soluciona los Sandwich attacks a través de las órdenes agrupadas o "batch orders" de su protocolo CoW⁵. En este protocolo, las órdenes que coinciden en precio se agrupan y se envían al mempool del blockchain como una sola. De esta forma, un grupo de usuarios obtiene exactamente el mismo intercambio que confirmaron, al mismo precio, para el mismo par de tokens y en el mismo bloque, evitando así la posibilidad de un Sandwich attack.

7.3. Llenado parcial y dinámico

El siguiente modelo ilustra el proceso de enrutamiento dividido en la plataforma de intercambio descentralizado 1inch. Destaca la aplicación de estrategias de llenado parcial y llenado dinámico en un intercambio de tokens A y B. Al adaptarse a las condiciones del mercado y ajustar las rutas de intercambio en tiempo real, 1inch optimiza la eficiencia del intercambio y minimiza el deslizamiento del precio.

⁵CoW: Coincidence of Wants (Coincidencia de Deseos) ocurre cuando dos operaciones o más poseen un activo que la otra quiere. Sus órdenes pueden liquidarse directamente entre ellas, sin la necesidad de un market maker externo o un proveedor de liquidez. [5]

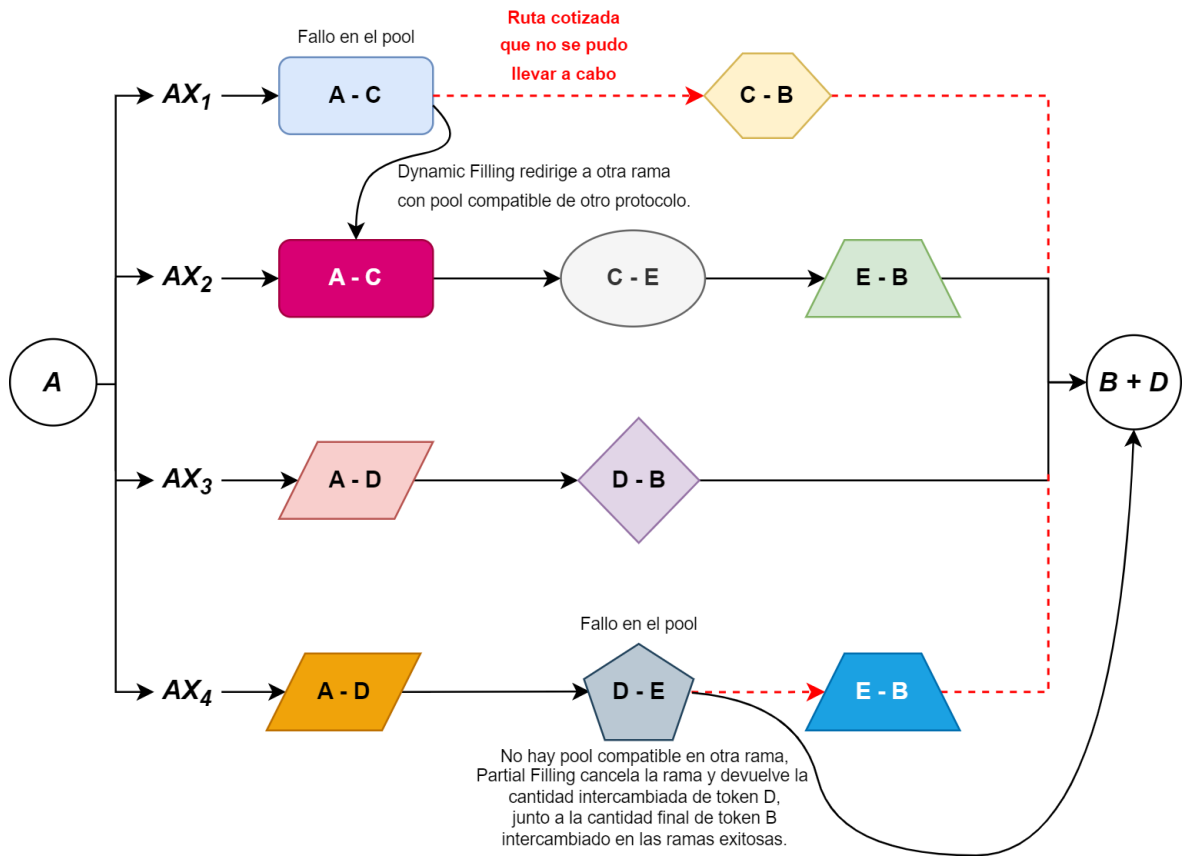


Figura 7.4: Modelo de swap con Partial y Dynamic Filling.

Fuente: Elaboración propia.

1. El modelo muestra el proceso de intercambio de un token A hacia un token B dentro de la plataforma 1inch utilizando enrutamiento dividido, Partial Filling y Dynamic Filling.
2. La cantidad inicial del activo A se separa en cantidades AX_y (formula $A = AX_1 + AX_2 + \dots + AX_n$), siendo X la cantidad por cada rama. Las líneas sólidas que conectan el nodo inicial A con las distintas formas (intercambios en diferentes pools de liquidez) representan las rutas de intercambio utilizadas en el proceso de enrutamiento dividido. Estas rutas permiten que las órdenes se ajusten y se envíen a través de múltiples pools de liquidez para maximizar la eficiencia y minimizar el deslizamiento.
3. Las formas dentro de cada rama representan los distintos pares de intercambios intermedios. La forma del contorno representa el mismo par de intercambio y el color

representa que se realiza en un pool de liquidez distinto. Los pools dan lugar a tokens intermedios que se utilizan en el proceso de intercambio, como un token C. En este caso, el token A se intercambia primero por el token C en un pool de liquidez, y luego el token C se intercambia por el token B en otro pool de liquidez (para la primera ruta cotizada). Dentro de las ramas, los pools puede ser identificadas como:

- a)* A - C (celeste claro)
 - b)* C - B (amarillo claro)
 - c)* A - C (magenta)
 - d)* C - E (gris)
 - e)* E - B (verde)
 - f)* A - D (rojo claro)
 - g)* D - B (morado claro)
 - h)* A - D (naranja)
 - i)* D - E (azul)
 - j)* E - B (celeste opaco)
4. La línea punteada representa la ruta originalmente cotizada para el usuario en el intercambio de tokens A y B. Sin embargo, esta ruta no se llevó a cabo, y en su lugar, se aplicaron estrategias de llenado parcial o llenado dinámico.
5. Las líneas solidas que se extienden desde donde se indica que hay un "Fallo en el pool", ilustran las rutas alternativas tomadas debido a la aplicación de llenado parcial o llenado dinámico. Estas rutas permiten que la orden se ajuste para maximizar la eficiencia, minimizar el deslizamiento y llevar a cabo el intercambio a pesar de errores.

Bibliografía

- [1] Binance Academy. ¿qué es la prueba de estaca (proof of stake)? 2018.
- [2] Coinmarketcap Alexandria. Dex aggregator. 2019.
- [3] DODO. Reveal the secrets of the aggregator — problem analysis and model building. 2022.
- [4] Sergej Kunz. What are dex aggregators? a deep dive by 1inch. *Alexandria*, 2021.
- [5] CoW Protocol. Coincidence of wants. 2021.
- [6] Nathan Reiff. B-money. 2021.
- [7] Rick Dudley Matthew Slipper Ian Norden Abdelhamid Bakhta Vitalik Buterin, Eric Conner. Fee market change for eth 1.0 chain. *EIP-1559*, 2019.
- [8] Xianmin Wang Joseph Liu Zuoxia Yu Jin Li Yongjun Zhao Sherman S. M. Chow Weizhi Meng, Jianfeng Wang. Position paper on blockchain technology: Smart contract and applications. *Lecture Notes in Computer Science*, 11058:474–483, 2018.
- [9] M. Zochowski. Why proof-of-work is not viable in the long-term,» 19 febrero 2019. [en línea]. 2019.