FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

DEPARTAMENTO DE TECNOLOGÍA INFORMÁTICA

Trabajo Practico Obligatorio- 2C- 2024

Profesor/es:

Maria Fernanda Cuadrado Estrebou

Grupo:

- Mendez, Jhoneybis - Legajo 1164567.
- Pinelli Bernard, Milton Ignacio – Legajo 124002
- Benitez, Merlene Nicole - Legajo
- Alvarez Russo, Matias – Legajo

Buenos Aires, 15 de Noviembre de 2024.-

Tabla de Contenidos

Introducción

Las *BlockChain* son *"*un mecanismo avanzado de bases de datos que permite compartir información transparente dentro de la red de una empresa. Una base de datos de cadena de bloques almacena los datos en bloques que se vinculan entre sí en una cadena." Para ello las Blockchaing aseguran la integridad de las transacciones mediante reglas criptográficas y consensos entre nodos.

En este trabajo practico, intentaremos desarrollar un algoritmo basado en técnicas de Backtracking para construir una BlockChaing válida que cumpla con un conjunto de reglas y restricciones especificas.

Descripción del Problema

Se busca resolver la generación de todas las posibles combinaciones de blockchains validas que se puedan formar con un conjunto de transacciones dadas, que contaran con atributos como tamaño, valor definido en satoshis, dependencias(relación con otra transacción) y el requerimiento de firmas.

Restricciones planteadas:

- Tamaño máximo de 1MB por bloque.
- Un máximo de 3 Transacciones por bloque.
- La suma de valor de un bloque no puede superar los 100 satoshis.
- La prueba de trabajo requiere que la suma de valores sea divisible por 10.

Estrategia de Resolución

Utilizaremos la técnica de BackTracking para explorar todas las combinaciones posibles de transacciones distribuidas en bloques.

Para esto, ordenaremos inicialmente las transacciones por prioridad según los criterios específicos que determinan su relevancia:

1. Valor: se priorizan las transacciones con mayor valor para maximizar la eficiencia del bloque.
2. Dependencia: Las transacciones que cuenten con dependencia se deberán considerar antes que las que no tienen.
3. Tamaño: Se tendrá en cuenta el limite definido de 1KB para no superar este limite con las transacciones incluidas.
4. Firmas Múltiples: ante transacciones con complejidad adicional se las considerara con una menor prioridad.

Definiremos un algoritmo que genere un bloque en construcción, verificando que cumpla con las restricciones de tamaño, valor y prueba de trabajo. Hallada una combinación valida de transacciones, esta se almacenará en una lista de soluciones.

Por nivel: Cada nivel del árbol de decisiones representa la incorporación de una nueva transacción al bloque actual

Opciones a evaluar por nivel: Desde el bloque actual, se evalúan todas las transacciones que aun no han sido agregadas al bloque.

Poda: Si alguna transacción viola alguna de las restricciones definidas en la descripción del problema. Representadas con los métodos cumpleRestriccion() y esValida()

El proceso de Backtraking se dará, ante la presencia de una transacción que no cumpla con las restricciones, haciendo un retroceso y pasando a la prueba siguiente de combinaciones posibles. Repetiremos este proceso recursivamente hasta que se hayan explorado todas las combinaciones posibles.

Pseudocódigo del Algoritmo de Resolución del Problema

Algoritmo BlockChainTPO

Entrada:

Transacciones: vector<transacciones: tamaño, valor, dependencia, firmas>
BLOQUEMAXIMO: entero -> 1 KB
VALORBMAXIMO: entero -> 100 satoshis
TRANSACCIONMAX: entero: entero -> 3 transacciones

Salida:
Soluciones: vector <>

```
contruirBlockchain( Transacciones, BLOQUEMAXIMO,
VALORMAXIMO,TRANSACCIONMAX)
        Soluciones=[vector ]
        ordenarTransaccionesPorPrioridad(Transacciones) <- transaccionesOrdenadas
        backTraking(Soluciones, transaccionesOrdenadas, [vector ])
        devuelve Soluciones.


backTracking (Soluciones, transaccionesOrdenadas, BlockchainActual)
        si esValida (BlockchainActual)
                Soluciones ← BlockchainActual
        Fin si
        TransaccionesRestantes<-TransaccionesOrdenadas
        Para Transacciones en TransaccionesRestantes
                Si cumpleRestriccion(Transaccion, BlockchainActual)
                        BlockChainActual<- transacción
                        TransaccionesRestantes – Transaccion.
                        backTracking (Soluciones, TransaccionesRestantes,
                BlockchainActual)
                Fin si
        Fin para
```

```
cumpleRestriccion(Transaccion,BlockchainActual)
        BloqueActual <- BlockChainActual[-1]
        Si ((BloqueActual.tamaño + Transaccion.tamaño) > BLOQUEMAXIMO)
                Devuelve FALSO
        Fin si
        Si ((BloqueActual.sumaValores+ Transaccion.valor) > VALORBMAXIMO)
                Devuelve FALSO
        Fin si
        Si ((BloqueActual.numeroTransacciones + 1) > TRANSACCIONMAX)
                Devuelve FALSO
        Fin si
        Si ((BloqueActual.sumaValores+ Transaccion.valor)%10 =! 0)
                Devuelve FALSO
        Fin si
        SINO
                Devuelve TRUE
        Fin sino

esValida (blockchain)
        para bloque en blockchain
                Si bloque. sumaValores > VALORBMAXIMO
                        Devuelve FALSO
                Fin si
                Si Bloque. numeroTransacciones > TRANSACCIONMAX
                        Devuelve FALSO
                Fin si
                Si bloque. Tamaño > BLOQUEMAXIMO
                        Devuelve FALSO
                Fin si
                Si bloque.sumaValores %10 =! 0
                        Devuelve FALSO
                Fin si
                Sino
                        Devuelve VERDADERO
                Fin sino
```

Análisis de Complejidad Temporal

Complejidad Temporal Teórica

El algoritmo planteado tiene una complejidad temporal que depende del numero de combinaciones posibles de transacciones y bloques. En el escenario pesimista la complejidad es de ***O(2^n)*** *siendo n el numero de transacciones , la verificación de validez de cada blockchain es O(m) donde m es el numeor de bloques*

Complejidad Temporal Práctica

******2da entrega******

Conclusiones

***** 2da entrega*****

Correcciones :

- *Incorporación del comportamiento del algoritmo:*

Algoritmo sin Poda	*Algoritmo con Poda*
Se exploraran todas las combinaciones posibles de transacciones y bloque. Recorriendo exhaustivamente cada nodo del árbol de decisiones e ir verificando si cumple con las restricciones o no. Esto garantiza que se encontraran todas las soluciones posibles, pero puede resultar ineficiente debido a la exploración de combinaciones que no son validas desde el inicio	*Se debe incorporar la verificación de las restricciones desde el inicio, y así descartar toda combinación que no cumpla con las restricciones, evitando así explorar sus ramas descenientes. De este modo reducimos significativamente el numero de evaluaciones pero requiere de una mayor lógica en su implementacion para determinar la poda*

De esta forma si incorporamos la poda, al encontrarse un bloque que supere el 1MB, es descartado o bien si la suma de los valores del bloque no es divisible por 10 tampoco se continua con su exploración en sus ramas.

- *Incorporacion de la evaluación temporal en el pseudocodigo*

Entrada:

```
contruirBlockchain( Transacciones, BLOQUEMAXIMO, VALORMAXIMO,TRANSACCIONMAX)
        Soluciones=[vector ]
        ordenarTransaccionesPorPrioridad(Transacciones) <- transaccionesOrdenadas
        backTraking(Soluciones, transaccionesOrdenadas, [vector ])
        devuelve Soluciones.
```

El ordenamiento realizado en ordenarTransaccionesPorPrioridad tiene una complejidad de O(n log n), donde n es el numero de transacciones.

```
backTracking (Soluciones, transaccionesOrdenadas, BlockchainActual)
        si esValida (BlockchainActual)
                Soluciones <- BlockchainActual
        Fin si
        TransaccionesRestantes<-TransaccionesOrdenadas
        Para Transacciones en TransaccionesRestantes ->(en el peor de los casos es O(n))
                Si cumpleRestriccion(Transaccion, BlockchainActual)
                        BlockChainActual<- transacción
                        TransaccionesRestantes – Transaccion.
                        backTracking (Soluciones, TransaccionesRestantes, BlockchainActual)
                        Fin si
        Fin para
```

El método backtracking realizado tiene una complejidad de O(2^n), donde n es el numero de transacciones.

```
cumpleRestriccion(Transaccion,BlockchainActual)
        BloqueActual <- BlockChainActual[-1]
        Si ((BloqueActual.tamaño + Transaccion.tamaño) > BLOQUEMAXIMO)
                Devuelve FALSO
        Fin si
        Si ((BloqueActual.sumaValores+ Transaccion.valor) > VALORBMAXIMO)
                Devuelve FALSO
        Fin si
        Si ((BloqueActual.numeroTransacciones + 1) > TRANSACCIONMAX)
                Devuelve FALSO
        Fin si
        Si ((BloqueActual.sumaValores+ Transaccion.valor)%10 =! 0)
                Devuelve FALSO
        Fin si
        SINO
                Devuelve TRUE
        Fin sino
```

El método cumpleRestriccion realizado tiene una complejidad lineal, ya que únicamente realiza comparaciones simples al igual que operaciones aritméticas simples.

```
esValida (blockchain)
        para bloque en blockchain
                Si bloque. sumaValores > VALORBMAXIMO
                        Devuelve FALSE
                Fin si
                Si Bloque. numeroTransacciones > TRANSACCIONMAX
                        Devuelve FALSE
                Fin si
                Si bloque. Tamaño > BLOQUEMAXIMO
                        Devuelve FALSE
                Fin si
                Si bloque.sumaValores %10 =! 0
                        Devuelve FALSE
                Fin si
                Sino
                        Devuelve TRUE
                Fin sino
```

El método esValida realizado tiene una complejidad O(m) , donde m es el numero de bloques en la blockchain actual

SEGUNDA ENTREGA

Para la segunda entrega, en función a las correcciones proporcionadas se reescribió la estrategia, teniendo como resultante:

Estrategia:
Definiremos un algoritmo que genere un bloque en construcción, verificando que cumpla con las restricciones de tamaño, valor y prueba de trabajo. Hallada una combinación valida de transacciones, esta se almacenará en una lista de soluciones.

Por Nivel: Cada nivel del árbol de decisiones representa una transacción potencial para añadir a un bloque actual.

Opciones a evaluar por nivel: Para cada bloque, revisamos las transacciones restantes (no agregadas previamente) que cumplen:

- Restricciones de tamaño.
- Condiciones de prueba de trabajo y valor.

- Dependencias satisfechas (si una transacción depende de T1, T1 debe estar en un bloque anterior).
- Se han alcanzado las firmas necesarias.

Poda:

- Si al añadir una transacción, se excede el tamaño o el valor, se ignora la ruta de esa rama.
- Si al agregar una transacción, se genera un ciclo invalido en dependencias o firmas, se retrocede.
- Cuando se seleccionan transacciones y no pueden contribuir a un bloque válido (por ejemplo: no pueden alcanzar divisibilidad por 10 en la suma de valores), la rama se poda

El proceso de Backtraking se dará, ante la presencia de una transacción que no cumpla con las restricciones, haciendo un retroceso y pasando a la prueba siguiente de combinaciones posibles. Repetiremos este proceso recursivamente hasta que se hayan explorado todas las combinaciones posibles o hasta que se asignen todas las transacciones a los bloques, una vez procesadas todas estas transacciones y si resultan en un conjunto de bloques validos, la solución es almacenada.

Complejidad temporal

a: Existen un un número potencial de llamadas recursivas, en el peor de los casos, cada transacción podría resultar en una nueva rama para cada bloque, sin embargo esta limitado por el numero maximo de bloques que entra como parametro.

b: Entre cada llamada recursiva, el tamaño del problema se reduce potencialmente en 1, ya que la transacción es o bien seleccionada y removida o dejada sin utilizar en esta iteración. Esto implica que b = 1.

k: Las operaciones de validacion podrian considerarse de complejidad constante = 1, los métodos de verificar si las dependencias están resueltas y de copiar el vector de la cadena de BlockChain se pueden considerar complejidad lineal = n. Podemos definir a k > 1 sin embargo son operaciones menos influyentes en el calculo final de la complejidad temporal, porque el coste del algoritmo se ve gobernado por los llamados recursivos de las combinaciones de las transacciones. Ademas en el caso especifico a > 1 no se considera k

Podemos concluir que: O (a ^ ([m ^ n] / b)) Donde ‘m’ es el numero maximo de bloques permitidos que limita la creacion de niveles y de profundidad de las ramificaciones, ‘n’ es el numero de transacciones.

sabiendo que b = 1 entonces

O (a ^ [m ^ n])

Complejidad Temporal Practica

Prueba para 8 transacciones, maximo de 4 transacciones por bloque, bloque de 1024kb y un valor maximo de 100.

Prueba1:

Max Bloques: 4 | Tiempo de ejecución: 1369167 ns | Soluciones: 3

Max Bloques: 6 | Tiempo de ejecución: 671167 ns | Soluciones: 38

Max Bloques: 9 | Tiempo de ejecución: 377875 ns | Soluciones: 48

Max Bloques: 11 | Tiempo de ejecución: 709125 ns | Soluciones: 48

Max Bloques: 20 | Tiempo de ejecución: 310667 ns | Soluciones: 48

Prueba2:

Max Bloques: 4 | Tiempo de ejecución: 1185042 ns | Soluciones: 3

Max Bloques: 6 | Tiempo de ejecución: 691750 ns | Soluciones: 38

Max Bloques: 9 | Tiempo de ejecución: 435000 ns | Soluciones: 48

Max Bloques: 11 | Tiempo de ejecución: 447833 ns | Soluciones: 48

Max Bloques: 20 | Tiempo de ejecución: 278875 ns | Soluciones: 48

Prueba3:

Max Bloques: 4 | Tiempo de ejecución: 3066083 ns | Soluciones: 3

Max Bloques: 6 | Tiempo de ejecución: 805333 ns | Soluciones: 38

Max Bloques: 9 | Tiempo de ejecución: 448167 ns | Soluciones: 48

Max Bloques: 11 | Tiempo de ejecución: 481958 ns | Soluciones: 48

Max Bloques: 20 | Tiempo de ejecución: 318833 ns | Soluciones: 48

Prueba4

Max Bloques: 4 | Tiempo de ejecución: 2147417 ns | Soluciones: 3

Max Bloques: 6 | Tiempo de ejecución: 663250 ns | Soluciones: 38

Max Bloques: 9 | Tiempo de ejecución: 401167 ns | Soluciones: 48

Max Bloques: 11 | Tiempo de ejecución: 315583 ns | Soluciones: 48

Max Bloques: 20 | Tiempo de ejecución: 385459 ns | Soluciones: 48

Analisis:

Los tiempos no crecen linealmente, lo cual tiene sentido ya que la complejidad temporal no es lineal y no depende de los parametros de entrada. En estas pruebas el numero de soluciones aumenta hasta un limite para luego estabilizarse, es decir, la mayoria de soluciones se alcanzan antes de alcanzar el limite de max bloques. Al tener una tecnica de poda eficiente, mas las restricciones definidas en la consigna y ademas la limitacion de la cantidad de bloques maximos a utilizar, hace que se generen soluciones rapidas reduciendo el numero de evaluaciones adicionales.

Tambien podemos notar que al comenzar con un número limitado de bloques disponibles, hay menos opciones para explorar en cómo distribuir las transacciones en los bloques. Esto puede prolongar el tiempo de ejecución, debido a que las restricciones siempre se deben aplicar, esto significa que hay que buscar soluciones válidas en un espacio de búsqueda más reducido.

Sin embargo, a medida que se incrementa el número máximo de bloques permitidos, el algoritmo dispone de más espacio para acomodar las transacciones. Este aumento en la capacidad de acomodar las transacciones, facilita la búsqueda de configuraciones válidas más rápidamente, lo que puede observarse como una disminución en los tiempos de ejecución.

Al contrario de lo que hace esperar la complejidad teorica en su peor caso, los tiempos de ejecución no escalan de la misma manera en como se plantean, esto se debe a la efectividad de las podas, las restricciones y la limitacion de bloques, lo cual disminuyen significativamente el numero de ramificaciones evaluadas.

A medida que el algoritmo aplica la poda y explora configuraciones válidas, la cantidad de evaluaciones disminuye considerablemente, lo que conduce a tiempos más manejables.

Grafico Prueba 1

Max Bloques	Tiempo ms
4	1,36
6	0,67
9	0,37
11	0,7
20	0,31

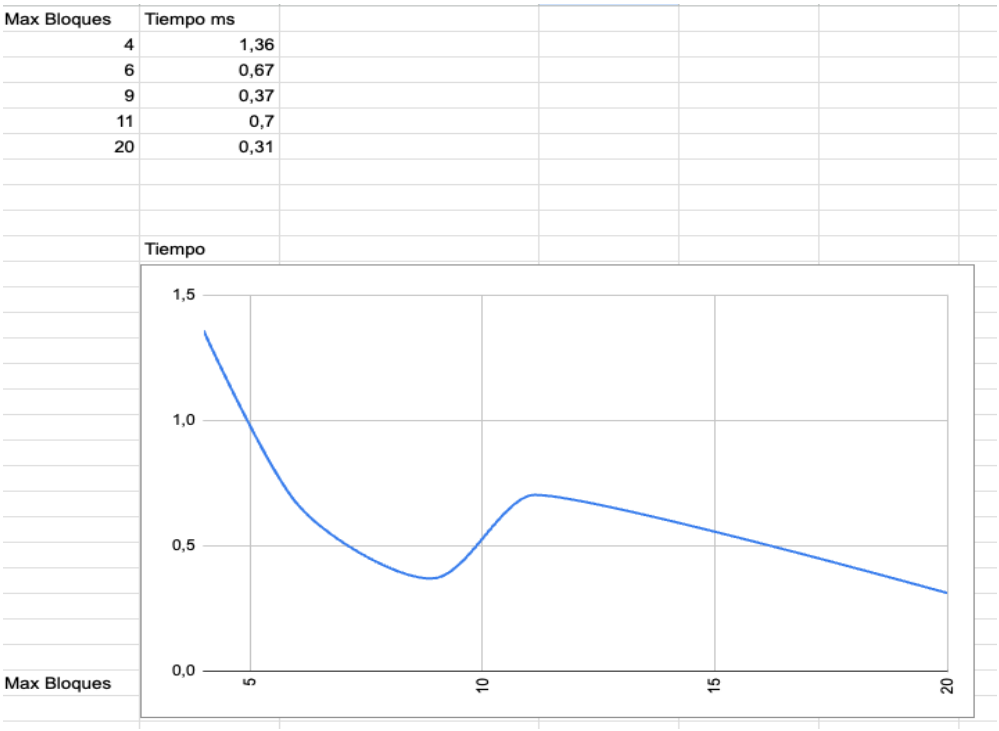


Grafico Prueba 2

Max Bloques	Tiempo ms
4	1,1
6	0,69
9	0,44
11	0,44
20	0,27

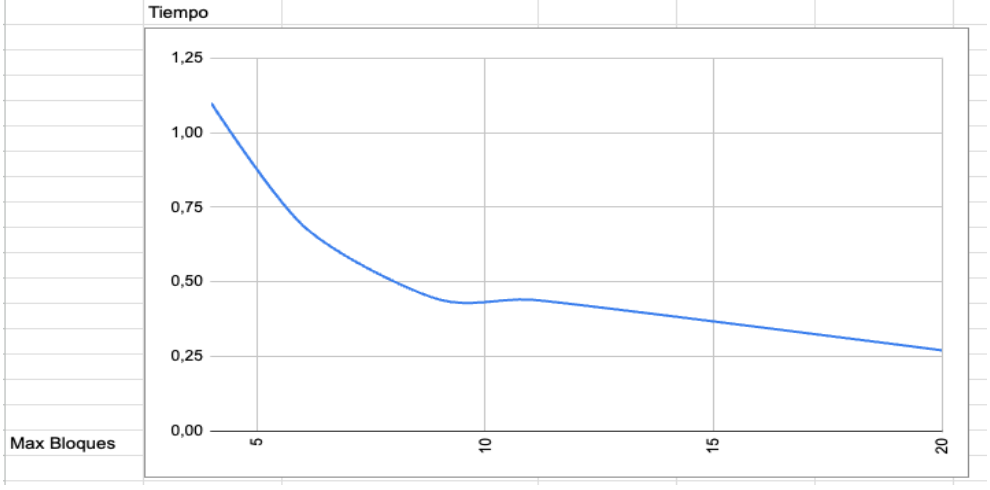


Grafico Prueba 3

Max Bloques	Tiempo ms
4	3,06
6	0,8
9	0,45
11	0,48
20	0,31

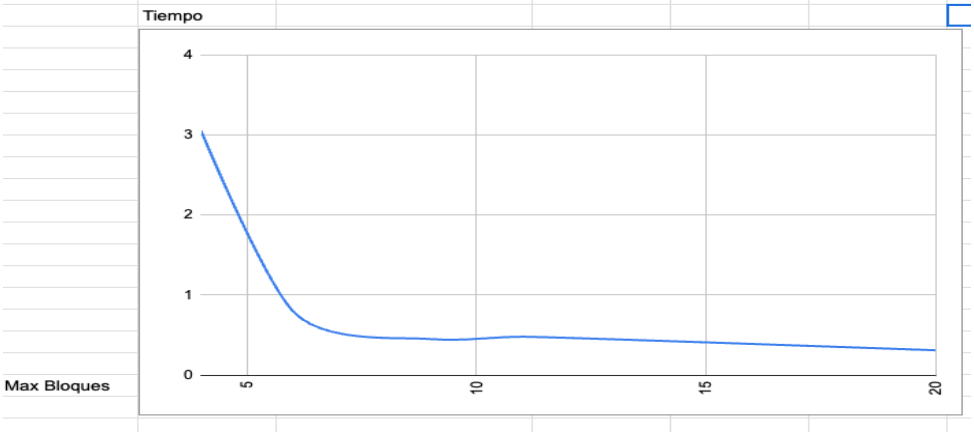
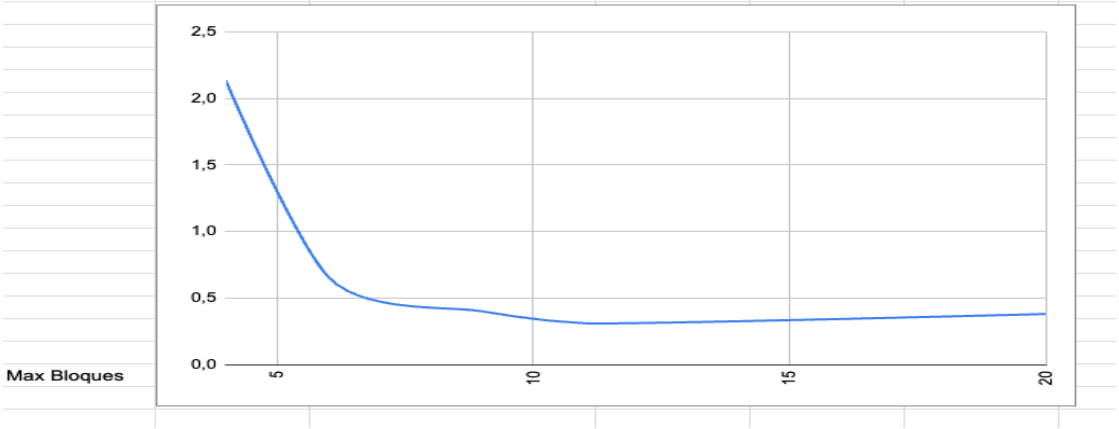


Grafico prueba 4

Max Bloques	Tiempo ms
4	2,14
6	0,66
9	0,4
11	0,31
20	0,38



Conclusiones

Luego de analizar e implementar el primer pseudocódigo, pudimos observar que el algoritmo inicialmente desarrollado no era eficiente, ademas de esto, tampoco consideramos restricciones adicionales, como el número máximo de bloques permitidos. De esta manera se decidió modificar el algoritmo, manteniendo la misma estrategia palnteada anteriormente. Este nuevo algoritmo puede conceptualizarse como una variación del problema de suma de subconjuntos, adaptando lo necesario para que el diseño pueda considerar y cumplir todas las consignas del ejercicio.

El algoritmo evalúa entre los distintos candidatos, en este caso, las transacciones, para determinar cuáles son válidas para integrarse en un nuevo conjunto (bloque). Durante este proceso, se asegura que se cumpla la prueba de trabajo requerida antes de decidir si debe incluir la transaccion al bloque.

Se incorporo un proceso de poda eficiente, esta tecnica elimina de forma prematura los caminos que no pueden satisfacer las restricciones, lo cual reduce de manera significativa la cantidad de combinaciones a evaluar. A pesar de que la complejidad sigue siendo exponencial este se puede considerar que se hace de manera controlada al limitar la cantidad de bloques maximos, asegurando la distribucion de transacciones de una manera eficaz en el espacio de busqueda.

Esto ultimo se puede comprobar con las pruebas de complejidad temporal practica, a medida que aumenta el maximo de bloques, la eficiencia para encontrar las soluciones incrementa, lo que reafirma la capacidad del algoritmo de ajustarse y maximizar su rendimiento.

En cuanto a la sinergia entre las dos complejidades, teorica y practica, podemos concluir que la teoría prepara para el peor escenario, establece un limite superior de lo que el algoritmo podria requerir en cuanto a tiempos y esfuerzo de calculo, sin embargo la complejidad practica revela que con su ejecución controlada, eficiente y restringida, muchas de las rutas no llegan a evaluarse. Esto resulta en un comportamiento que, aunque sigue siendo gobernado por una complejidad de tipo exponencial en teoría, se lleva a cabo de manera controlada y eficiente en la práctica.

Bibliografía

- https://aws.amazon.com/es/what-is/blockchain/#:~:text=La%20tecnolog%C3%ADa%20de%20cadena%20de%20bloques%20es%20un%20mecanismo%20avanzado,entre%20s%C3%AD%20en%20una%20cadena.