UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRUEBAS DE CONOCIMIENTO CERO PARA PROBLEMAS DE INCIDENCIA

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIAS

PRESENTA
ING. JORGE ALBERTO SOLANO GÁLVEZ

TUTOR
DR. JOSÉ DAVID FLORES PEÑALOZA
Facultad de Ciencias, UNAM

CIUDAD UNIVERSITARIA, CDMX, JUNIO 2022

A Sairi.

Agradecimientos

Investigación realizada gracias al Programa UNAM-PAPIIT IN120520.

A mi esposa por su apoyo, compresión y amor.

A mis padres por su amor y apoyo.

A mis suegros por su motivación.

A mi tutor por su paciencia y guía.

A mi sínodo por sus comentarios y aportaciones al trabajo.

Índice general

Agradecimientos II

I Pruebas de conocimiento cero 1

1 Introducción 2
§1.1 Problemas de incidencia . . . 3
§1.2 Justificación del trabajo . . . 4
§1.3 Trabajo relacionado . . . 5
§1.4 Motivación . . . 5

2 Sistema de pruebas de conocimiento cero 8
§2.1 Sistema de pruebas interactivo de conocimiento cero . . . 9
§2.1.1 Protocolo interactivo . . . 9
§2.1.2 Sistema de pruebas interactivo . . . 9
§2.1.3 Sistema de pruebas de conocimiento cero . . . 10
§2.1.4 Propiedades de un sistema de pruebas interactivos de conocimiento cero . . . 12
§2.2 Esquema de compromiso . . . 13
§2.2.1 Propiedades de un esquema de compromiso . . . 13

3 Sistema de pruebas de conocimiento cero para cualquier problema en NP 16
§3.1 Sistema de pruebas de conocimiento cero para gráfica 3-coloreable . . . 16
§3.1.1 Protocolo de conocimiento cero de gráfica 3 colorable . . . 17
§3.1.2 Análisis de complejidad del protocolo de gráfica 3 colorable . . . 21

II Resultados originales 22

4 3SAT 23
§4.1 Definición del problema . . . 23
§4.2 Complejidad . . . 24
§4.3 Prueba de conocimiento cero para 3SAT . . . 24
§4.3.1 Protocolo de conocimiento cero para 3SAT . . . 26
§4.3.2 Demostración del protocolo propuesto para 3SAT . . . 29
§4.3.3 Análisis de complejidad del protocolo de 3SAT . . . 35

5 Cobertura de conjunto **36**
§5.1 Definición del problema . . . 36
§5.2 Complejidad . . . 36
§5.3 Prueba de conocimiento cero para cobertura de conjunto . . . 37
§5.3.1 Protocolo de conocimiento cero para cobertura de conjunto . . . 38
§5.3.2 Demostración del protocolo propuesto para cobertura de conjunto . . . 40
§5.3.3 Análisis de complejidad del protocolo de cobertura del conjunto . . . 45

6 Cobertura de puntos **47**
§6.1 Definición del problema . . . 47
§6.2 Complejidad . . . 47
§6.3 Prueba de conocimiento cero para el problema de cobertura de puntos . . . 48
§6.3.1 Protocolo de conocimiento cero para cobertura de puntos . . . 49
§6.3.2 Demostración del protocolo propuesto para cobertura de puntos . . . 50
§6.3.3 Análisis de la complejidad del protocolo de cobertura de puntos . . . 55

7 Marco de trabajo generalizado **57**
§7.1 Pasos del marco de trabajo . . . 57
§7.2 Caso de estudio para el marco de trabajo propuesto . . . 58
§7.3 21 problemas de Karp . . . 61

8 Conclusiones y trabajos futuros **75**

A Clases de complejidad **77**
§A.1 Clases DTIME y P . . . 77
§A.2 Clase NP . . . 77
§A.3 Clase NP-Completo . . . 78
§A.4 Clase PP . . . 78
§A.5 Clase IP . . . 78

B Probabilidad **80**
§B.1 Variables aleatorias . . . 80
§B.2 Función de probabilidad . . . 80
§B.3 Distribuciones de probabilidad . . . 81
§B.3.1 Distribución de Bernoulli . . . 81
§B.3.2 Distribución geométrica . . . 81
§B.3.3 Valor esperado . . . 82
§B.3.4 Distribución uniforme discreta . . . 83

C Teoría de gráficas **84**
§C.1 Gráfica . . . 84
§C.2 Gráfica completa . . . 84
§C.3 Gráfica isomorfa . . . 84
§C.4 Gráfica bipartita . . . 84

D Máquinas de Turing **85**

§D.1 Máquina de Turing probabilística de tiempo polinomial 85
§D.2 Máquina de Turing no uniforme probabilística de tiempo polinomial 85
§D.3 Máquina de Turing interactiva 85

Parte I

Pruebas de conocimiento cero

Capítulo 1

Introducción

Una prueba de conocimiento cero describe un protocolo de comunicación interactivo que tiene como fin demostrar que existe la solución de un problema sin dar la solución en sí. Un ejemplo de este tipo de pruebas es la cueva de Ali Babá:

Imaginemos una cueva con dos entradas (digamos izquierda y derecha) de acceso a sendos caminos que terminan en una puerta que los separa, la cual siempre está cerrada y para abrirla es necesario decir las palabras mágicas (figura 1.1).

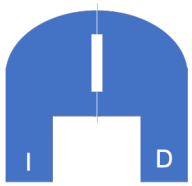

Figura 1.1: La cueva de Ali Babá

Alice quiere demostrarle a Bob que conoce las palabras mágicas para abrir la puerta, pero no le quiere decir cuáles son éstas. Bob no cree que Alice conozca las palabras puesto que no se las quiere revelar.

Para que Alice pueda demostrarle a Bob que conoce las palabras mágicas sin revelárselas siguen los siguientes pasos (figura 1.2):

1. Alice y Bob llegan a las entradas de la cueva.
2. Bob se aleja de la cueva para no ver la entrada que toma Alice. Alice elige una entrada y llega a la puerta por ese camino.
3. Bob regresa a la cueva, elige una entrada y le pide a Alice (gritándole) que salga por ella.

Si Alice conoce las palabras mágicas siempre va a poder regresar por la entrada que le pida Bob. Si Alice no conoce las palabras mágicas, tiene un 50 % de probabilidad de salir por la entrada que le pida Bob (esto es, de engañar a Bob). Para reducir la probabilidad de ser engañado, Bob repite los pasos varias veces.

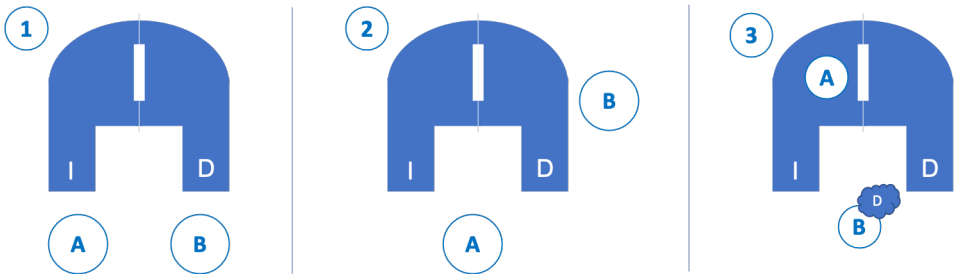

Figura 1.2: Protocolo de la cueva de Ali Babá

El presente trabajo tiene como objetivo generar protocolos de conocimiento cero *ad hoc* para problemas de incidencia. Si bien se sabe que existe una prueba general de conocimiento cero para cualquier problema NP, se desea generar protocolos a la medida para problemas de incidencia que sean intuitivos en su construcción y fáciles de implementar.

> **Hipótesis**: *Es posible generar protocolos de conocimiento cero* ad hoc *para problemas de incidencia concretos, que sean fáciles de entender en su construcción y factibles de implementar.*

El escrito está dividido en dos partes. En la primera parte se definirán los conceptos básicos requeridos para comprender el uso, la interacción y la generación de pruebas de conocimiento cero: En el capítulo 2 se definirán los conceptos que se requieren para crear un sistema de pruebas interactivo de conocimiento cero y demostrar que el protocolo generado cumple las propiedades requeridas por los sistemas de conocimiento cero. En el capítulo 3 se presentará un sistema de pruebas de conocimiento cero para cualquier problema en NP. En la segunda parte se generarán pruebas de conocimiento cero *ad hoc* para diversos problemas de incidencia siguiendo el marco de trabajo propuesto: En el capítulo 4 se generará el protocolo *ad hoc* con todo detalle para el problema de *3SAT*. En el capítulo 5 se generará el protocolo *ad hoc* con todo detalle para el problema de *Cobertura de conjunto*. En el capítulo 6 se generará el protocolo *ad hoc* con todo detalle para el problema de *Cobertura de puntos*. En el capítulo 7 se describirá el marco de trabajo propuesto que permite crear protocolos de conocimiento cero a la medida para problemas de inicidencia; después, se detallará el uso del marco de trabajo propuesto utilizando el problema de *Ciclo hamiltoniano no dirigido*; al final, se esbozará la aplicación del marco de trabajo para diseñar protocolos ad-hoc de algunos de los 21 problemas de Karp utilizando el marco de trabajo propuesto y se realizará un análisis de complejidad de cada protocolo generado (este análisis de complejidad se realizará utilizando Máquinas de Acceso Aleatorio y no Máquinas de Turing); para los problemas a los que no se les pudo aplicar el marco de trabajo se presentará una discusión razonada de las razones por las cuales no se pudo aplicar. En el capítulo 8 se expondrán las conclusiones del trabajo y el trabajo futuro.

1.1. Problemas de incidencia

El término *problemas de inicidencia* hace referencia a un problema que se puede modelar como una gráfica; este término viene del concepto de vértices adyacentes en teoría de grafos: dos vértices

son adyacentes si están conectados por una arista, cada uno de estos vértices es incidente a dicha arista.

Para utilizar el marco de trabajo propuesto y, con ello, generar un protocolo de conocimiento cero primero se debe modelar el problema como una gráfica bipartita, para ello los elementos se deben transformar a nodos y las relaciones a aristas. Con este enfoque se puede tener una granularidad de información a nivel de aristas y vértices individuales para descubrir (mostrar) siempre la misma cantidad de información.

Una vez que un problema ha sido transformado a una gráfica, entonces se puede intentar aplicar el marco de trabajo propuesto en el presente trabajo para generar el protocolo de conocimiento cero *ad hoc*.

1.2. Justificación del trabajo

En la literatura revisada se muestra que existe un sistema de pruebas de conocimiento cero para todo problema en NP como se demuestra en [6]. Este teorema es existencial y parte de la premisa de poder reducir el lenguaje que describe el problema original a una instancia del problema de Gráfica 3-coloreable, para después poder aplicar el protocolo propuesto en 3.1.1.

Por otra parte, en la práctica existen diversas aplicaciones de pruebas de conocimiento cero (tanto interactivas como no interactivas) como lo son:

- *Blockchain*: La criptomoneda *Zcash* permite verificar la validez de sus transacciones bajo las reglas de consenso de la red mediante el uso de pruebas zk-SNARK (*Zero-Knowledge Succinct Non-Interactive Argument of Knowledge*) [26].
- Finanzas: La institución financiera ING lanzó una prueba de rango de conocimiento cero (ZKRP, por sus siglas en inglés), la cual permite probar que un entero se encuentra dentro de cierto intervalo [11].
- Voto electrónico: En un sistema de elecciones universalmente verificable la correctitud de las pruebas de conocimiento cero permiten verificar que un voto ha sido emitido y contabilizado [9].
- Aprendizaje autónomo: Las pruebas de conocimiento cero en un modelo de aprendizaje autónomo permiten convencer a otros que un modelo calcula una predicción o que alcanza una gran precisión en un conjunto de datos sin fugar información sobre el modelo [21].
- Control de armas: Se intenta demostrar que dos o más cabezas de guerra nuclear tienen recuentos idénticos de transmisión y emisión de neutrones bajo la irradiación de neutrones de alta energía [5].

Sin embargo, en la investigación realizada no encontramos un ensayo que muestre una forma de realizar protocolos de conocimiento cero a la medida para problemas específicos. En este trabajo propondremos un marco de trabajo que muestre cómo diseñar pruebas de conocimiento cero interactivas para problemas específicos. Tener pruebas de conocimiento cero originales es importante por dos razones:

1. Matemáticamente es importante generar protocolos *ad hoc* y la literatura no muestra como hacerlos. Intrínsecamente, todos los protocolos generados en el trabajo son resultados matemáticos originales.

2. En la práctica, se pueden generar e implementar pruebas de conocimiento cero para aplicaciones *ad hoc* como la que se muestra en [5].

1.3. Trabajo relacionado

Las pruebas de conocimiento cero fueron introducidas por primera vez por Shafi Goldwasser, Silvio Micali y Charles Rackoff en 1985 [7] como un procedimiento de demostración de teoremas (un método nuevo y eficiente capaz de comunicar una prueba) e interactivo (el receptor de la demostración debe hacer preguntas y recibir respuestas del probador para verificar de forma eficiente que una declaración es correcta), el cual debería cumplir con las siguientes propiedades:

1. Es posible demostrar un teorema verdadero

2. Es imposible demostrar un teorema falso

3. La comunicación de la demostración debe ser eficiente: sin importar cuánto se tarde el probador en el proceso de prueba, el cómputo requerido por el verificador debe ser fácil.

En los artículos *The Knowledge Complexity of Interactive Proof- System* publicados en 1985 [7] y en 1989 [8] se presentan las pruebas de conocimiento cero para los problemas de *Residuo cuadrático* y *Residuo no cuadrático*. En el artículo *Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems* [6] se presentan las pruebas de conocimiento cero para los problemas de *Gráfica isomorfa* y *Gráfica no isomorfa*. Además, también en [6] se demuestra que todos los lenguajes en NP tienen un sistema de pruebas de conocimiento cero, como se muestra en el capítulo 3.

Sin embargo, en la investigación realizada no se encontraron otros trabajos cuyo propósito sea enseñar a diseñar protocolos *ad hoc*, menos aún trabajos que de forma integrada presenten un marco de trabajo para diseñar sistemas de pruebas de conocimiento *ad hoc*. Creémos que es importante que un lector interesado pueda entender y diseñar un protocolo de conocimiento cero *ad hoc* y, por ello, creemos que el trabajo aquí presentado es importante e inédito.

1.4. Motivación

Se desea crear un marco de trabajo que permita generar pruebas de conocimiento cero para problemas específicos haciéndo énfasis en el modelado del problema y el análisis necesario para generar el protocolo de conocimiento cero y que este cumpla con las propiedades de completitud, solidez y conocimiento cero, debido a que en la investigación realizada no se encontró algo similar.

El marco de trabajo propuesto se aplicó sobre problemas NP-Duros que se pueden modelar como problemas de incidencia, esto es, a la instancia original se le asoció una gráfica (usualmente bipartita) para generar las pruebas de conocimiento cero respectivas.

Se desarrollaron 14 pruebas de conocimiento cero *ad hoc* para 14 problemas *NP-Completos* aplicando el marco de trabajo propuesto en 7.1, con lo cual se pudo comprobar que los pasos propuestos son eficaces (permitieron generar una prueba de conocimiento cero) y replicables (se pudo repetir el mismo enfoque para cada problema seleccionado y se obtuvo el resultado esperado).

Para introducir al lector en el desarrollo de pruebas de conocimiento cero utilizando el marco de trabajo propuesto se desarrollaron los sistemas de pruebas de conocimiento cero con todo detalle para los problemas de *3SAT* 4.3.1, *Cobertura de conjunto* 5.3.1 y *Cobertura de puntos* 6.3.1:

- El problema de $3SAT$ se eligió por la importancia del teorema de Cook-Levin. El también conocido como teorema de Cook establece que los problemas de satisfacción booleana de una fórmula en Forma Normal Conjuntiva (SAT) y una fórmula en 3 Forma Normal Conjuntiva ($3SAT$) son NP-Completos. El problema $3SAT$ es un lenguaje raíz para una gran cantidad de reducciones de problemas NP-Completos y para los cuales, con el sistema de pruebas generado en 4.3.1 se tiene una prueba de conocimiento cero directa. Para construir la gráfica bipartita se modelaron el conjunto de cláusulas y el conjunto de literales como nodos y la incidencia de la i-ésima literal con la j-ésima cláusula como aristas.

- El problema de *Cobertura de conjunto* se eligió por ser un problema de cobertura, que es el enfoque que sigue marco de trabajo propuesto, ya que si bien el marco de trabajo es para problemas de incidencia, las ideas se originan en el problema de cobertura. Para construir la gráfica bipartita se utilizaron el conjunto de elementos y el conjunto de subconjuntos (como nodos) y la incidencia del i-ésimo elemento en el j-ésimo subconjunto (como aristas).

- El problema de *Cobertura de puntos* es especial porque es un problema en un dominio geométrico que permitió explotar propiedades extras del dominio y que nos permite demostrar que utilizando propiedades específicas del dominio, podemos construir protocolos aún más eficientes (toda instancia de este problema es también instancia de Cobertura de conjunto, pero el protocolo a la medida para este problema es más eficiente que el genérico de Cobertura de conjunto). Para construir la gráfica bipartita se utilizaron el conjunto de puntos y el conjunto de líneas (como nodos) y la incidencia de la k-ésima línea en el i-ésimo punto (como aristas).

Debido a que el marco de trabajo presenta similitudes en el análisis del problema y el diseño del protocolo, creemos que no es necesario explicar los detalles repetitivos una y otra vez, por lo que, una vez desarrollados en extenso los protocolos *3SAT*, *Cobertura de conjunto* y *Cobertura de puntos*, se procedió a detallar el marco de trabajo propuesto, describiendo los pasos a seguir para generar las pruebas de conocimiento cero. Para ejemplificar la aplicación del marco de trabajo se utilizó como caso de estudio el problema de *Ciclo hamiltoniano no dirigido* 7.2:

- El problema de *Ciclo hamiltoniano no dirigido* se seleccionó porque existe un protocolo de conocimiento cero bien conocido para este problema que es de una sola ronda que no funciona en nuestro marco propuesto y para el cual es importante mostrar cómo se puede abordar de otra manera utilizando nuestro marco de trabajo.

Al final del trabajo, se revisaron las reducciónes realizadas por Richard Karp en [17], debido a que estos problemas son emblemáticos como problemas NP-Completos, resultan adecuados para

explorar el alcance y las limitaciones del marco de trabajo. De ahí se analizó a cuales de los 21 problemas de Karp se les podía aplicar el marco de trabajo. Además, para cada problema seleccionado se generó un esbozo del protocolo de conocimiento cero y se realizó un análisis de complejidad del protocolo generado. Para los problemas de Karp a los cuales no se pudo aplicar el marco de trabajo, se presenta una discusión razonada del porqué.

Capítulo 2

Sistema de pruebas de conocimiento cero

Un *sistema de pruebas de conocimiento cero* permite verificar la solución de un problema sin dar a conocer, explícitamente, la solución. La prueba de conocimiento cero se lleva a cabo a través de un protocolo interactivo.

Un *protocolo interactivo* consiste en un protocolo de comunicación entre un par de *máquinas de Turing interactivas* (A y B) que intercambian mensajes, donde A es el probador que posee la solución del problema y B es el verificador que quiere conocer la solución del problema. A tiene poder de cómputo no acotado mientras que B tiene poder de cómputo polinomialmente acotado. El problema que se quiere resolver es muy difícil (problemas que se presume son más que polinomiales) y por ello B no puede resolverlo por sí mismo.

A y B desconfían mutuamente uno de otro, por lo que para verificar que A sí tiene la solución del problema seguirán un protocolo de comunicación de varias rondas a base de retos elegidos por B de forma aleatoria.

Un *sistema de pruebas de conocimiento cero* (*zero knowledge proof*) consiste en un *protocolo interactivo* de varias rondas donde A le demuestra a B que posee la solución a un problema, cumpliendo con las propiedades de completitud (*completeness*), solidez (*soundness*) y conocimiento cero (*zero knowledge*). Para cumplir con estas propiedades se debe garantizar que, si A tiene la solución del problema, siempre será capaz de responder el reto (completitud) y la información que se envía no se podrá ligar entre rondas para obtener la solución original del problema (conocimiento cero); y, si A no tiene la solución del problema, B se podrá dar cuenta con alta probabilidad (solidez).

En cada ronda del protocolo interactivo A se debe comprometer con la información necesaria para que B pueda verificar que sí tiene la solución del problema. Para que B no pueda ligar la información entre rondas, A la cambia en cada ronda, usando transformaciones aleatorias. Además, la información que se envía en cada ronda A la protege (oculta) a través de un *esquema de compromiso*, lo que garantiza que el contenido original no podrá ser deducido por el receptor B, ni podrá ser modificado por el emisor A.

A continuación, se definirán qué es un sistema de pruebas interactivo de conocimiento cero, sus propiedades y el concepto de esquema de compromiso de manera formal.

2.1. Sistema de pruebas interactivo de conocimiento cero

Un sistema de pruebas interactivo es un modelo computacional en el que dos máquinas (A y B) intercambian mensajes. La máquina A es capaz de resolver problemas NP A.2 o más difíciles A.3. La máquina B tiene capacidad de cómputo polinomial A.1.

2.1.1. Protocolo interactivo

Un protocolo interactivo está formado por un par de Máquinas de Turing Interactivas D.3 (digamos A y B), que comparten la cinta de entrada, y la cinta de comunicación de solo escritura de B es la cinta de comunicación de solo lectura de A y viceversa.

Las máquinas van tomando turnos activos en cada etapa del protocolo, una después de la otra. Cualquier máquina puede terminar el protocolo. A no está limitado computacionalmente mientras que B está limitado polinomialmente por la longitud de la cadena de entrada.

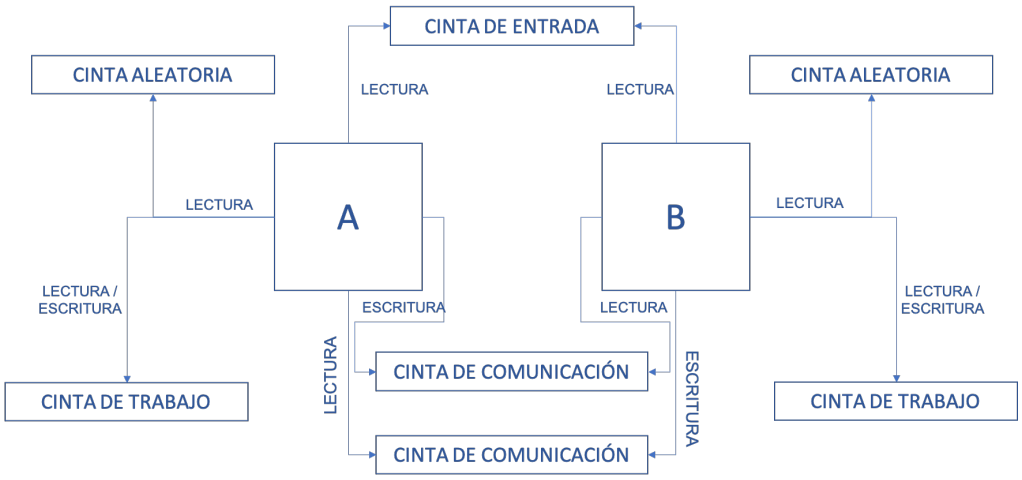

Figura 2.1: Protocolo interactivo propuesto por Goldwasser, Micali y Rackoff [7]

2.1.2. Sistema de pruebas interactivo

Sea $L \subseteq \{0,1\}^*$ un lenguaje y $\langle A, B\rangle$ un protocolo interactivo, se dice que L tiene un *sistema de pruebas interactivo* si A (el probador) tiene poder de cómputo no acotado, B (el verificador) es una máquina de Turing probabilista de tiempo polinomial D.1 y las siguientes propiedades se satisfacen [24]:

- Completitud: Para cualquier $x \in L$, existe un testigo de la solución $y \in \{0,1\}^*$ que hace que B se detenga y acepta con probabilidad 1.
- Solidez: Para cualquier probador A' y cualquier $x \notin L$, B acepta con probabilidad a lo más $\frac{1}{2}$. Nota: La definición usual pide una probabilidad de a lo más una función despreciable $\epsilon(|x|)$, pero por simplicidad en este trabajo nosotros usaremos $\frac{1}{2}$. Esto es suficiente, puesto que un número lineal de repeticiones del protocolo, amplifica la probabilidad de rechazo exponencialmente $\frac{1}{2}$.

Las probabilidades de las propiedades de completitud y solidez se alcanzan después de k iteraciones del protocolo a partir de la entrada x. La probabilidad de cada propiedad está en función de las elecciones aleatorias que haga B.

2.1.3. Sistema de pruebas de conocimiento cero

Un sistema de pruebas interactivo $\langle A, B\rangle$ es de conocimiento cero si $\forall$ verificador $B^* \in MT\,PTP$ existe un simulador $S^* \in MT\,PTP$ tal que $\forall\ x \in L$ la distribución de la salida del $S^*(x)$ es indistinguible de la distribución de la salida del $\langle A, B\rangle(x)$.

Sea $\langle A, B\rangle$ un sistema de pruebas interactivo para algún lenguaje $L \in NP$ y sea S^* una máquina de Turing probabilista de tiempo polinomial para el lenguaje L, para toda $x \in L$, sus distribuciones de salidas son computacionalmente indistinguibles, lo que se denota como:

$$\{\langle A, B\rangle(x)\} \approx_c \{S^*(x)\}$$

Entonces, para cualquier $x \in L$, no es posible distinguir la distribución de la salida del simulador S^* con la entrada x y la distribución de la salida del protocolo interactivo $\langle A, B\rangle$ con la entrada x.

La propiedad de conocimiento cero garantiza que cualquier probador A que ejecute el protocolo no proporcionará más información que el hecho de que $x \in L$, incluso si interactúa con un verificador tramposo B'. La existencia del simulador S^* que simule a un verificador B^* (ambas $MT\,PTP$) permite verificar que el protocolo termina en tiempo polinomial y es de conocimiento cero.

La distribución de probabilidad de la salida hace referencia a los bits de la cinta de comunicación al ejecutar el protocolo para una $x \in L$. La propiedad de conocimiento cero garantiza que, para cualquier cadena $x \in L$, no es posible diferenciar cuándo la distribución de probabilidad de la salida es realizada por el protocolo interactivo $\langle A, B\rangle$ o cuándo es realizada por el simulador S^*.

Existen 3 tipos de conocimiento cero [14]:

- **Conocimiento cero perfecto**: Para toda $x \in L$ y para todo verificador B^*, la distribución de la salida del simulador $\{S^*(x)\}$ y la distribución de la salida del sistema de pruebas interactivo $\{\langle A, B^*\rangle(x)\}$ son exactamente iguales.

- **Conocimiento cero estadístico**: Para toda $x \in L$ y para todo verificador B^*, la distribución de la salida del simulador $\{S^*(x)\}$ y la distribución de la salida del sistema de pruebas interactivo $\{\langle A, B^*\rangle(x)\}$ son estadísticamente cercanas.

  Dadas dos variables aleatorias X y Y B.1, sus distribuciones de probabilidad B.3 son estadísticamente cercanas si y sólo si

  $$\forall c\ \exists N \text{ tal que } \forall n > N$$

  $$\sum_{\alpha \in \{0,1\}^n} |Pr[X = \alpha] - Pr[Y = \alpha]| < \frac{1}{n^c}$$

  donde N está dada por la longitud de la cadena de salida. Se requieren un número más que polinomial de muestras para diferenciar las distribuciones $\{X\}$ y $\{Y\}$ para toda $x \in L$.

- **Conocimiento cero computacional**: Para toda $x \in L$ y para todo verificador B^*, la distribución de la salida del simulador $\{S^*(x)\}$ y la distribución de la salida del sistema de pruebas interactivo $\{\langle A, B^*\rangle(x)\}$ son computacionalmente indistinguibles por una máquina de Turing no uniforme probabilista de tiempo polinomial D.2.

  Dadas dos variables aleatorias X y Y B.1, sus distribuciones de probabilidad B.3 son computacionalmente indistinguibles en tiempo polinomial si y sólo si

$$\forall c, \forall A \in MT\,PTP, \forall x \in L, \forall r_1 \in \{0,1\}^* \text{ y } \forall r_2 \in \{0,1\}^*$$
$$\exists N \text{ tal que } \forall n > N$$
$$|Pr_X[A(x, r_1) = 1] - Pr_Y[A(x, r_2) = 1]| < \tfrac{1}{n^c}$$

  Es decir, la diferencia de las distribuciones $\{X\}$ y $\{Y\}$ para toda $x \in L$ es tan pequeña que una máquina de Turing no uniforme probabilista de tiempo polinomial no las puede distinguir.

En el presente trabajo se trabajará con conocimiento cero computacionalmente indistinguible, lo cual se denotará como $\approx_c$.

Lema híbrido

La indistinguibilidad computacional es transitiva, esto es, si $\{X\} \approx_c \{Y\}$ y $\{Y\} \approx_c \{Z\}$ entonces $\{X\} \approx_c \{Z\}$.

Lema 2.1.1. *Sean $X^1, X^2, \ldots, X^m$ una secuencia de distribuciones de probabilidad. Suponiendo que un distinguidor D puede distinguir X^1 y X^m con una probabilidad $> \epsilon$, entonces existe alguna $i \in [1, \ldots, m-1]$ tal que D distingue X^i y X^{i+1} con probabilidad $\geq \frac{\epsilon}{m}$.*

Demostración. Para la prueba se hace uso de la desigualdad del triángulo que dice que $|a + b| \leq |a| + |b|$. Generalizando se tiene que

$$|\sum_{i=1}^{k} x_i| \leq \sum_{i=1}^{k} |x_i|$$

Supongamos que D distingue X_1 y X_m con probabilidad $> \epsilon$, entonces:

$$|Pr[t \leftarrow X^1 : D(t) = 1] - Pr[t \leftarrow X^m : D(t) = 1]| > \epsilon \tag{2.1}$$

Sea $g_i = Pr[t \leftarrow X^i : D(t) = 1]$. Entonces, la ecuación 2.1 se puede reescribir como $|g_1 - g_m|$, quedando como:

$$\epsilon > |g_1 - g_m|$$
$$= |g_1 - g_2 + g_2 - g_3 + \cdots + g_{m-1} - g_m|$$
$$= |\sum_{i=1}^{m-1} g_i - g_{i+1}|$$
$$\leq \sum_{i=1}^{m-1} |g_i - g_{i+1}| \qquad \text{(Por la desigualdad del triángulo)}$$

Entonces, la suma de los $m-1$ valores absolutos deben exceder el valor de ϵ. Esto quiere decir que existe una i tal que:

$$\epsilon > |g_i - g_{i+1}| \geq \frac{\epsilon}{m-1} \geq \frac{\epsilon}{m}$$

(ya que, de otro modo, la sumatoria sería menor que ϵ). $|g_i - g_{i+1}|$ es exactamente la probabilidad que D distinga X^i y X^{i+1}. Por lo tanto, D distingue X^i y X^{i+1} con probabilidad $\geq \frac{\epsilon}{m}$, como se requiere. □

Función despreciable

Una función $\epsilon(x)$ es despreciable si para cada constante c existe una x_0 tal que para todas las $x > x_0$, $\epsilon(x) \leq \frac{1}{x^c}$. Intuitivamente, una función despreciable es asintóticamente menor que el inverso de cualquier polinomio fijo.

Por el contrario, una función $t(n)$ no es despreciable si existe una constante c tal que para una infinidad de puntos $\{x_0, x_1, \ldots\}$, $t(x_i) > \frac{1}{x_i^c}$. Intuitivamente, una función no es despreciable si es mayor que el inverso de algún polinomio fijo para una infinidad de puntos $\{x_0, x_1, \ldots\}$.

2.1.4. Propiedades de un sistema de pruebas interactivos de conocimiento cero

Un sistema de pruebas interactivo de conocimiento cero para un lenguaje L está constituido por un par de máquinas de Turing interactivas $\langle A, B\rangle$, con B de tiempo polinomial, que cumple con las propiedades [6]:

- **Completitud (*completeness*):** Dada cualquier $x \in L$, el verificador B acepta después de interactuar con el probador A y corroborar los hechos, con probabilidad 1.
- **Solidez (*soundness*):** Dada cualquier $x \notin L$ y para cualquier probador A^*, el verificador B rechaza al interactuar con A^*, con probabilidad $\geq 1/2$.
- **Conocimiento cero (*zero knowledge*):** Para cualquier verificador B^* existe un simulador S^*, que es una máquina de Turing probabilística de tiempo polinomial esperado, tal que para toda $x \in L$ las distribuciones de salida del simulador S^* y del sistema $\langle A, B\rangle$ con x como entrada son polinomialmente indistinguibles, lo cual se denota como:

  $$\{\langle A, B\rangle(x)\} \approx_c \{S^*(x)\}$$

  Intuitivamente, la existencia del simulador S^* implica que dada una $x \in L$ el verificador B^* al interactuar con A no puede aprender nada extra que no pudiera calcular por él mismo en tiempo polinomial.

  Observemos que el comportamiento de S^* no está definido cuando $x \notin L$. Esto es crucial porque S^* no puede, en principio, identificar si una x dada está o no en L. Intuitivamente, al ejecutarse S^* con una cadena x lo que B aprenderá será (estadísticamente) lo mismo, independientemente de si x está o no en L.

2.2. Esquema de compromiso

Durante el proceso de comunicación que se lleva a cabo en un sistema de pruebas interactivo de conocimiento cero, el probador A se debe comprometer con la solución del problema. Esta solución debe permanecer oculta hasta que el verificador B lance un reto y solicite una comprobación. Para evitar que B obtenga más información de la solución, la información inicial que A transmite a B la oculta utilizando esquemas de compromiso. Esto le permitirá más adelante a A revelar fragmentos individuales de la información que envió anteriormente y a B le permitirá tener un alto nivel de confianza de que la información que A le revela es la misma que originalmente le envió de forma oculta.

Un esquema de compromiso o *commitment scheme* es una primitiva criptográfica que permite comprometer un valor mientras se mantiene oculto otro (la información). Una característica importante en un sistema de pruebas de conocimiento cero es que este valor no puede ser modificado una vez comprometido.

El esquema de compromiso dentro del espacio de mensajes M está compuesto por tres operaciones elementales: *SETUP*, *COMMIT* y *DECOMMIT* [20].

- La operación *SETUP* genera la llave de compromiso $ck = SETUP(\{0,1\}^k)$ con valores aleatorios elegidos con probabilidad uniforme $\in_R \{0,1\}$ de tamaño k, con $k \in \mathbb{N}$.
- La operación *COMMIT* genera el compromiso $c = COMMIT_{ck}(m)$ para el mensaje m con la llave ck.
- La operación *DECOMMIT* a partir de un mensaje m' y una llave ck', genera el compromiso $c' = DECOMMIT_{ck'}(m')$.

Supongamos que A quiere comprometer la información m para enviársela a B utilizando la llave ck, (con $ck = SETUP(\{0,1\}^k)$). A genera el compromiso $c = COMMIT_{ck}(m)$ y envía c (se compromete con c) a B. Después, A necesita descubrir la información por lo que le envía m y ck a B, quien realiza $c' = DECOMMIT_{ck}(m)$. B admite el mensaje original comprometido si y sólo si $c' = c$.

Suponiendo que los esquemas de compromiso existen, los protocolos propuestos en el presente trabajo se pueden implementar, independientemente del esquema de compromiso utilizado.

2.2.1. Propiedades de un esquema de compromiso

Los esquemas de compromiso constan de dos fases: la *fase de compromiso* donde se envía un mensaje en una caja cerrada y la *fase de revelación* donde se abre la caja y se revela el mensaje.

A continuación, se definirá de manera formal un esquema de compromiso de mensajes simples, es decir, la fase de compromiso y la fase de revelación consisten de un solo mensaje.

Esquema de compromiso de un solo mensaje

Una máquina de tiempo polinomial M es llamada esquema de compromiso si existe un polinomio $l()$ que conserva las siguiente dos propiedades:

- Propiedad de ocultamiento (*hiding*): El compromiso $c = COMMIT_{ck}(m)$ no revela información del mensaje m.

  Dado cualquier distinguidor no uniforme probabilista de tiempo polinomial D existe una función despreciable $\epsilon()$ tal que para cada $n \in \mathbb{N}$ y para cualquier mensaje $m_0, m_1 \in \{0,1\}^n$, D distingue las siguientes distribuciones con probabilidad a lo más $\epsilon(n)$:

  - $\{COMMIT_{ck}(m_1)$, con $ck = SETUP(\{0,1\})^{l(n)}\}$
  - $\{COMMIT_{ck}(m_2)$, con $ck = SETUP(\{0,1\})^{l(n)}\}$

- Propiedad vinculante (*binding*): Es computacionalmente difícil generar una colisión entre valores compromiso, esto es, dados dos mensajes m y m' con $m \neq m'$, el emisor A no es capaz de abrir el compromiso $c = COMMIT_{ck}(m)$ dando m' y ck', con $ck' \neq ck$.

  Para cualquier $n \in \mathbb{N}$ y para cualquier mensaje $m_0, m_1 \in \{0,1\}^n$ y cualquier llave $ck_0, ck_1 \in \{0,1\}^{l(n)}$ se cumple que:

$$COMMIT_{ck_0}(m_0) \neq COMMIT_{ck_1}(m_1)$$

Las propiedades anteriores definen los esquemas de compromiso para un solo mensaje. Sin embargo, durante la ejecución del protocolo se deben comprometer múltiples mensajes.

Esquema de compromiso de múltiples mensajes

Un esquema de compromiso *SETUP*, *COMMIT* y *DECOMMIT* se considera seguro para múltiples mensajes si para cualquier distinguidor no uniforme probabilista de tiempo polinomial D y para cualquier función de tiempo polinomial $p(n)$, existe una función despreciable $\epsilon()$ tal que para cualquier $n \in \mathbb{N}$ y cualquier $m_1, m_2, \ldots, m_{q(n)}, m'_1, m'_2, \ldots, m'_{p(n)} \in \{0,1\}^n$, D puede distinguir las siguientes distribuciones con probabilidad a lo más $\epsilon(n)$:

- $\{COMMIT_{ck}(m_1), COMMIT_{ck}(m_2), \ldots, COMMIT_{ck}(m_{p(n)})\}$
- $\{COMMIT_{ck}(m'_1), COMMIT_{ck}(m'_2), \ldots, COMMIT_{ck}(m'_{p(n)})\}$

con $ck = SETUP(\{0,1\})^{l(n)}$ para cada operación *COMMIT* [24].

De la misma manera en la que se define una función segura para *múltiples mensajes*, es posible definir la noción de *seguridad de valores múltiples* en compromisos, por lo que cualquier esquema de compromiso es seguro para valores múltiples.

En la práctica, una forma de llevar a cabo esquemas de compromiso es utilizando funciones *hash* criptográficas, bajo el supuesto de que éstas son funciones de un solo sentido.

Por lo tanto, cuando A se compromete con un mensaje m (*COMMIT*) lo que hará es obtener y enviar el *hash* criptográfico del mensaje m con un *nonce*[1] ck (*SETUP*), es decir:

$$c = hash(m, ck)$$

[1]El término *nonce* es el acrónimo (en inglés) de número de un solo uso.

Cuando A deba abrir un valor ($DECOMMIT$), entonces deberá mandar el mensaje original m y el *nonce* ck, para que el verificador B pueda obtener el *hash* de m y ck, es decir:

$$c' = hash(m, ck)$$

Para que B pueda validar que el mensaje ($hash$) comprometido proviene del mensaje original deberá comparar los valores c y c', si $c = c'$ acepta, de otro modo rechaza.

Los esquemas de compromiso que utilizarán en los protocolos propuestos en este trabajo harán uso implícito de funciones *hash* criptográficas.

Capítulo 3

Sistema de pruebas de conocimiento cero para cualquier problema en NP

Todo lenguaje en NP tiene un sistema de pruebas de conocimiento cero si las funciones unidireccionales existen[1]. La demostración sigue los siguientes pasos [24]:

1. Se tiene un sistema de pruebas interactivo de conocimiento cero $\langle A, B\rangle$ para un lenguaje $L \in$ NP-Completo. El lenguaje L considerado es **gráfica 3-coloreable** (G3C)[2].

2. Para cualquier lenguaje $L \in NP$ y cualquier cadena $x \in L$, A y B reducen x a una instancia x' de $G3C$ (en tiempo polinomial) y después ejecutan el protocolo dado en 1 con la instancia x'.

Dado el lenguaje $L \in NP$, se reduce L a $G3C$ mediante una función de reducción f de tiempo polinomial. Se puede considerar a f como un conjunto de reducciones, primero una reducción de L a $3SAT$ (utilizando la prueba de Cook [22]) y después una reducción de $3SAT$ a $G3C$ (como se muestra en [10]). Entonces, $x \in L$ si y sólo si $f(x) \in G3C$.

El sistema de pruebas interactivo $\langle A, B\rangle$ tiene como entrada común a $x \in L$. Tanto A como B realizan la reducción $G = f(x)$. El probador A utiliza un sistema de pruebas interactivo para demostrar que G es 3-coloreable. El verificador B puede aceptar o rechazar según el resultado.

3.1. Sistema de pruebas de conocimiento cero para gráfica 3-coloreable

A continuación, se presenta un sistema de pruebas de conocimiento cero para gráfica 3-coloreable propuesto en [24]. El sistema de pruebas interactivo $\langle A, B\rangle$ tiene como entrada común $G = (V, E)$ con $n = |V|$ y $m = |E|$. El probador A tiene como testigo de la solución a $w = c_0, c_1, \cdots, c_{n-1}$ que es una 3-coloración válida para la gráfica G ($c_i \in \{1, 2, 3\}$ es el color del vértice v_i).

[1] Una función $f : \{0,1\}^* \to \{0,1\}^*$ es unidireccional o *one way* si f se puede computar en tiempo polinomial, pero un algoritmo F de tiempo polinomial aleatorio que intente computar la pseudo inversa de la función f tendría éxito con una probabilidad insignificante.

[2] G3C es el conjunto de gráficas cuyos vértices pueden ser coloreados por 3 colores 1, 2, 3 de tal forma que 2 vértices conectados no tengan el mismo color.

3.1.1. Protocolo de conocimiento cero de gráfica 3 colorable

El siguiente protocolo se repite $|E|$ veces.

1. **Probador** A: Elige una permutación aleatoria π sobre los colores $\{1, 2, 3\}$ y la aplica sobre los vértices de la gráfica a partir del testigo de la solución w. Para cada vértice $i \in [0, n-1]$ utiliza un esquema de compromiso para comprometerse con los colores de cada vértice $c'_i = COMMIT_{ck_i}(\pi(c_i))$, con la llave $ck_i = SETUP(\{0,1\}^k)$ de k bits.

2. **Verificador** B: elige una arista aleatoria $e_{(i,j)} \in E$ y se la envía a A.

3. **Probador** A: abre los colores c'_i y c'_j de los vértices comprometidos v'_i y v'_j y se los envía a B.

4. **Verificador** B: Si $c'_i \neq c'_j$ continúa con la siguiente ronda. Si $c'_i = c'_j$ rechaza y termina. Si después de las $|E|$ rondas B no rechazó, entonces acepta.

Para que un protocolo sea un sistema de pruebas de conocimiento cero debe cumplir con las propiedades de completitud, solidez y conocimiento cero.

Teorema 3.1.1. *El protocolo 3.1.1 es un sistema de pruebas interactivo de conocimiento cero para gráfica 3-coloreable.*

Demostración

- **Completitud ($x \in L$):** Si G es 3-coloreable y tanto el probador A como el verificador B siguen el protocolo, entonces el probador A se va a poder comprometer con una 3-coloración válida y, sin importar la arista $e \in E$ que elija el verificador B, $c'_i \neq c'_j$, por lo que B acepta con probabilidad 1.

- **Solidez ($x \notin L$):** Si G no es 3-coloreable, entonces, en la 3-coloración comprometida, existe por lo menos una arista que tiene dos vértices con el mismo color. Si el verificador B sigue el protocolo, un probador tramposo A' será atrapado en la mentira con probabilidad, al menos, $\frac{1}{|E|}$. Dado que el protocolo se repite $|E|$ veces, la probabilidad de que B acepte es de

$$(1 - \tfrac{1}{|E|})^{|E|} \leq \tfrac{1}{e} < \tfrac{1}{2}$$

  Lo cual cumple con la probabilidad que se busca[3].

- **Conocimiento cero ($\{\langle A, B\rangle(x)\} \approx_c \{S^*(x)\}$, $x \in L$):** Para toda $x \in L$, la distribución de salida del sistema de pruebas interactivo $\langle A, B\rangle$ es computacionalmente indistinguible a la distribución de salida del simulador S^* ejecutando a B^*. Para probar esta propiedad se va a hacer uso del simulador descrito en 2.1.4.

[3] En [3] se muestra que el

$$\lim_{x\to\infty}(1 - \frac{1}{x})^x = \frac{1}{e}$$

Sea una máquina de Turing interactiva B^* y sea S^* una máquina de Turing de tiempo esperado polinomial. S^* ejecuta a B^* varias veces, en cada ocasión S^* elige las cadenas que va a poner en las cintas de lectura de B^* y lee las cintas de salida que B^* genera. A continuación, se muestra la descripción del simulador S^*.

Los siguientes pasos se ejecutan $|E|$ veces por S^*:

1. Elige una arista aleatoria $e_{(i',j')} \in E$ y elige colores aleatorios $c_{i'}, c_{j'} \in \{1,2,3\}$ para los vértices $v_{i'}, v_{j'} \in V$ con $c_{i'} \neq c_{j'}$. El resto de los vértices los pinta con un color nulo $c_k = 0$, $v_k \in \{V\} \setminus \{v_{i'}, v_{j'}\}$.
2. Realiza los compromisos $c_{x'} = COMMIT(c_x, ck_x)$ con $1 \leq x \leq n$ y escribe estos compromisos en la cinta de comunicación de solo lectura de B^*.
3. Simula a B^*. Los bits de la cinta aleatoria de B^* los va llenando S^* sobre demanda al momento de la simulación.
4. Lee la arista $e_{(i,j)}$ de la cinta de comunicación de solo escritura de B^*.
5. Si $e_{(i,j)} = e_{(i',j')}$ descubre los compromisos $c_{i'}$ y $c_{j'}$ y los escribe en la cinta de comunicación de solo lectura de B^*. En otro caso, repite esta ronda desde el paso 1 restableciendo el estado de B^* al inicio de la ronda actual.
6. Simula a B^*.

Ahora hay que probar que el simulador se ejecuta en tiempo polinomial esperado y que para todo B^* y para toda $x \in L$ la distribución de salida del simulador S^* es computacionalmente indistinguible de la distribución de salida del sistema de pruebas interactivo $\langle A, B^* \rangle$.

Es importante aclarar que el simulador S^* es una máquina interactiva extendida que tiene el poder de modificar todas las cintas de B^* y de reestablecer el estado de B^* al inicio de la ronda actual. Es decir, cada vez que repite una ronda S^* puede reestablecer el estado de las cintas de B^*, para que la ronda que está anulando no tenga efecto en la ronda siguiente.

Proposición 3.1.1.1. *El simulador S^* termina en tiempo polinomial esperado.*

Demostración. El tiempo de ejecución del algoritmo del simulador está en función de la probabilidad de que B elija la misma arista $e_{(i,j)}$ que elige el probador A, esta probabilidad es $\frac{1}{|E|}$. Por B.3.3, el número esperado de intentos para completar una iteración es $|E|$. Debido a que el algoritmo se repite $|E|$ veces, el número esperado de reintentos es $|E|^2$. Dado que el número máximo de aristas en una gráfica es $E = \frac{n*(n-1)}{2}$, el tiempo esperado de ejecución del algoritmo es $O(n^4)(n+T_{B^*}(n))$, donde $T_{B^*}(n)$ es el tiempo de ejecución de B^* que tiene como entrada una gráfica de n vértices, lo que por hipótesis es de tiempo polinomial esperado; por lo tanto, el tiempo de ejecución total del simulador es de tiempo polinomial esperado. □

Por otra parte, hay que probar que la distribución de salida del simulador S^* y la distribución de salida del sistema de pruebas interactivo $\langle A, B^* \rangle$ para toda $x \in L$ son computacionalmente indistinguibles. Para realizar este análisis se van a generar una serie de simuladores que muestren la transición desde un simulador que se comporta como el probador A hasta un simulador que se comporta como S^*, analizando entre cada simulador contiguo su transcripción de la cinta de salida para ver si se pueden distinguir o no (computacionalmente).

1. Sea S_1 un simulador con el testigo de la solución w. S_1 ejecuta el siguiente algoritmo A_1:
   a) **Utiliza el testigo de la solución** w**.** Se compromete con los colores c_k con $1 \leq k \leq n$. Simula a B^*.
   b) **Recibe** una arista $e_{i,j} \in E$.
   c) **Abre** c_i y c_j. Simula a B^*. Como $c_i \neq c_j$ entonces B^* continúa.

   La transcripción de la conversación de S_1 es τ_1. La transcripción de la conversación de $\langle A, B\rangle$ es τ_0. S_1 actúa como un probador honesto A, por lo tanto, la distribución de la transcripción $\{\tau_0\}$ es computacionalmente indistinguible a la distribución de la transcripción $\{\tau_1\}$, es decir, $\{\tau_0\} \approx_c \{\tau_1\}$.

2. Sea S_2 un simulador con el testigo de la solución w. S_2 ejecuta el siguiente algoritmo A_2:
   a) **Utiliza el testigo de la solución** w **y elige una arista** $e_{i',j'} \in E$**.** Se compromete con los colores c_k con $1 \leq k \leq |V|$. Simula a B^*.
   b) **Recibe** una arista $e_{i,j} \in E$.
   c) Si $e_{i,j} = e_{i',j'}$, **abre** c_i y c_j; simula a B^*; como $c_i \neq c_j$ entonces B^* continúa. **Si** $e_{i,j} \neq e_{i',j'}$**, restablece las cintas de** B^* **al estado inicial de esta ronda y regresa al paso** *a*)**.**

   La transcripción de la conversación de S_2 es τ_2.

3. Sea S_3 un simulador con el testigo de la solución w. S_3 ejecuta el siguiente algoritmo A_3:
   a) Utiliza el testigo de la solución w y elige una arista $e_{i',j'} \in E$. **Se compromete con los colores** $c_{i'}, c_{j'} \in w$ **de los vértices** $v_{i'}, v_{j'}$ **y con los colores** $c_k = 0$**, para todos los vértices** V **excepto** $v_{i'}, v_{j'}$**.**
   b) **Recibe** una arista $e_{i,j} \in E$.
   c) Si $e_{i,j} = e_{i',j'}$, **abre** c_i y c_j; simula a B^*; como $c_i \neq c_j$ entonces B^* continúa. Si $e_{i,j} \neq e_{i',j'}$, restablece las cintas de B^* al estado inicial de esta ronda y regresa al paso *a*).

   La transcripción de la conversación de S_3 es τ_3.

4. Sea $S_4 = S^*$ un simulador que ejecuta el siguiente algoritmo A_4:
   a) **Elige una arista** $e_{i',j'} \in E$**. Elige de forma aleatoria los colores** $c_{i'}, c_{j'} \in_R \{1, 2, 3\}$ **de los vértices** $v_{i'}, v_{j'}$**, con** $c_{i'} \neq c_{j'}$**. Se compromete con los colores** $c_{i'}, c_{j'}$ **y con los colores** $c_k = 0$**, para todos los vértices** V **excepto** $v_{i'}, v_{j'}$**.**
   b) **Recibe** una arista $e_{i,j} \in E$.
   c) Si $e_{i,j} = e_{i',j'}$, **abre** c_i y c_j; simula a B^*; como $c_i \neq c_j$ entonces B^* continúa. Si $e_{i,j} \neq e_{i',j'}$, restablece las cintas de B^* al estado inicial de esta ronda y regresa al paso *a*).

   La transcripción de la conversación de S_4 es τ_4.

Las distribuciones $\{\tau_1\}$ y $\{\tau_2\}$ son computacionalmente indistinguibles ya que la única diferencia entre los algoritmos A_1 y A_2 es que A_2 espera que la arista que recibe el simulador sea la misma que eligió.

La distribuciones $\{\tau_2\}$ y $\{\tau_3\}$ difieren en los esquemas de compromiso de los colores c_k, mientras que en A_2 se envían a partir del testigo de la solución w, en A_3 se envían dos colores $c_{i'}, c_{j'}$ de los vértices $v_{i'}, v_{j'}$ de la arista elegida de forma aleatoria $e_{i',j'}$ a partir del testigo de la solución w y el resto de los colores nulos.

Proposición 3.1.1.2. *Para cualquier distinguidor D no uniforme PTP existe una función despreciable $\epsilon()$ tal que para todo $x \in L$ suficientemente grande, D distingue las distribuciones $\{\tau_2\}$ y $\{\tau_3\}$ con probabilidad a lo más $\epsilon(|x|)$.*

Demostración. Supongamos por contradicción que existe un distinguidor no uniforme D, una constante c y un número infinito de cadenas $x \in L$ tales que D puede distinguir las distribuciones $\{\tau_2(x)\}$ y $\{\tau_3(x)\}$ con probabilidad mayor que $\frac{1}{|x|^c}$.

Sean S' un simulador que actúa como S_2 y S'' un simulador que actúa como S_3. Consideremos la secuencia de simuladores híbridos $H_0, H_1, \ldots, H_{|E|}$, el simulador H_k ejecuta las primeras k rondas como el simulador S'' y el resto de las rondas como el simulador S'. Nótese que la distribución de la salida de $\{H_0(x)\}$ es idéntica a la distribución de la salida de $\{S'(x)\}$ y la distribución de la salida de $\{H_{|E|}(x)\}$ es idéntica a la distribución de la salida de $\{S''(x)\}$. Entonces, por el lema híbrido (lema 2.1.1), existe algún $k \in [0, |E|)$ para la cual D distingue las siguientes distribuciones:

- $\{H_k(x)\}$
- $\{H_{k+1}(x)\}$

con probabilidad mayor a $\frac{1}{|x|^c \cdot |E|}$. Como $|x| > |E|$, entonces esta probabilidad es mayor que $\frac{1}{|x|^{c+1}}$. Sin embargo, la única diferencia entre los simuladores H_k y H_{k+1} es que en la ronda $(k+1)$-ésima, el simulador H_{k+1} manda los valores comprometidos a partir del testigo de la solución w y en la ronda k-ésima el simulador H_k manda dos vértices con colores válidos y el resto de vértices con valores nulos. Entonces, D puede distinguir los valores comprometidos con probabilidad mayor a $\frac{1}{|x|^{c+1}}$.

Sin embargo, esto contradice la propiedad de ocultamiento de valores múltiples de los esquemas de compromiso 2.2.1, la cual asegura que existe alguna función despreciable $\epsilon_1()$ que acota por arriba la probabilidad de distinguir dos distribuciones que provienen de un número polinomial de valores ocultos mediante esquemas de compromiso. □

Las distribuciones $\{\tau_3\}$ y $\{\tau_4\}$ son computacionalmente indistinguibles ya que solo difieren en la elección del color de los vértices $c_{i'}$ y $c_{j'}$ (mientras que A_3 los obtiene de w, A_4 los selecciona de forma aleatoria entre los colores válidos).

Por el lema híbrido 2.1.3 se sabe que la indistinguibilidad computacional es transitiva, por lo que se puede afirmar que las distribuciones $\{\tau_0\} \approx_c \{\tau_4\}$ y, por lo tanto, $\{\langle A, B\rangle(x)\} \approx_c \{S^*(x)\}$. ■.

3.1.2. Análisis de complejidad del protocolo de gráfica 3 colorable

Ahora, se realizará el análisis de complejidad de la comunicación durante la ejecución del protocolo de *gráfica 3-coloreable*.

El sistema de pruebas $\langle A, B\rangle$ de $G3C$ tiene como entrada común la gráfica $G = (V, E)$ con $n = |V|$ y $m = |E|$. A tiene como testigo de la solución a $w = c_0, c_1, \ldots, c_{n-1}$ que es una 3-coloración válida.

El protocolo se repite $|E|$ veces.

1. **Probador** A: Elige una permutación aleatoria π sobre los colores $\{1, 2, 3\}$ y la aplica sobre todos vértices de la gráfica a partir del testigo de la solución w. Para cada vértice $i \in [0, n-1]$ envía un esquema de compromiso de $c'_i = COMMIT_{ck_i}(\pi(c_i))$, con la llave $ck_i = SETUP(0, 1^K)$ de k bits.

   En este paso se envían los n vértices con la coloración comprometida. Crear la coloración y los esquemas de compromiso toma un tiempo $O(n)$. Los vértices comprometidos ocupan espacio $O(n)$.

2. **Verificador** B: elige una arista aleatoria $e_{(i,j)} \in E$ y se la envía a A.

   La elección de la arista $e_{i,j}$ toma un tiempo $O(1)$. La arista ocupa espacio $O(1)$.

3. **Probador** A: abre los colores c'_i y c'_j de los vértices comprometidos v'_i y v'_j y se los envía a B.

   Para abrir los colores c'_i y c'_j se ocupa un tiempo $O(1)$. La apertura de los colores c'_i y c'_j ocupan espacio $O(1)$.

4. **Verificador** B: Si $c'_i \neq c'_j$ acepta y continúa, en otro caso rechaza y termina.

   Validar si $c'_i \neq c'_j$ toma tiempo $O(1)$. El espacio para hacer la validación es $O(1)$.

Debido a que el protocolo se repite $|E|$ veces, el tiempo total de ejecución del protocolo es $O(n \cdot |E|)$ y el espacio utilizado en la comunicación es $O(n \cdot |E|)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(n \cdot |E|)$$

Es importante comentar que el protocolo de *gráfica 3-coloreable* será utilizado como protocolo de referencia en el presente trabajo debido a que es la reducción que maneja la literatura para demostrar que cualquier $L \in NP$ tiene un sistema de pruebas de conocimiento cero. Sin embargo, la referencia o comparación entre diversos protocolos no es directa, ya que no siempre se involucran los mismos elementos y no se puede hablar de una comparación equitativa, pero sí se puede tener una comparación relativa en términos de eficiencia en la comunicación.

Parte II

Resultados originales

Capítulo 4

3SAT

En este capítulo se generará un protocolo de conocimiento cero *ad hoc* para el problema de 3SAT.

Una cláusula es la disyunción ($\vee$) de una o más literales. La fórmula clausal es la conjunción ($\wedge$) de una o más cláusulas. Una fórmula ϕ es satisfacible si existe una asignación de valores a sus variables que hagan que ϕ sea verdadera. El problema de SAT consiste en, dada un fórmula booleana ϕ, decidir si ϕ es satisfacible o no. $3SAT$ es una restricción de SAT donde cada cláusula tiene exactamente 3 literales.

4.1. Definición del problema

El problema de satisfacción booleana (SAT) consiste en determinar si existe una interpretación que satisfaga una fórmula booleana dada. Si las variables de una fórmula booleana pueden ser reemplazadas (consistentemente) por valores *TRUE* o *FALSE* tal que la fórmula se evalúe como *TRUE*, entonces se dice que la fórmula es satisfacible.

Una literal puede ser una variable (llamada literal positiva) o la negación de la variable (llamada literal negativa). Una cláusula es una disyunción de una o más literales. Una fórmula está en Forma Normal Conjuntiva (FNC) si es una conjunción de cláusulas (o una sola cláusula):

$$\bigwedge_i \left(\bigvee_j v_{i_j} \right)$$

donde v_{i_j} es una variable positiva u_k o negativa $\neg u_k$. Los términos v_{i_j} son las literales y las expresiones $v_{i_1} \vee v_{i_2} \vee \cdots \vee v_{i_l}$ son las cláusulas. Una fórmula $k\,FNC$ es una fórmula FNC en las que todas sus cláusulas tiene a lo más k literales [22].

El problema 3 satisfacción (3SAT), igual que el problema de SAT, consiste en determinar la satisfabilidad de una fórmula en Forma Normal Conjuntiva donde cada cláusula consta de exactamente 3 literales.

4.2. Complejidad

En [22] se demuestra que SAT es NP-duro. Ahí mismo se demuestra que $3SAT$ es NP-Completo a través de una reducción de $SAT \leq_p 3SAT$.

4.3. Prueba de conocimiento cero para 3SAT

Supongamos que el probador A quiere demostrarle al verificador B que la fórmula $\phi \in 3SAT$ es satisfacible con conocimiento cero. La entrada común del protocolo es ϕ que está formada por $2n$ literales $l_i \in L$ (las n variables más sus respectivas negaciones) y por m cláusulas $c_i \in C$. A posee como testigo de la solución a $V = v_0, \ldots, v_{n-1}$ con las asignaciones a las literales que hacen que ϕ se satisfaga.

La idea detrás del protocolo es trabajar con una gráfica bipartita asociada a cada una de las instancias del problema. Por un lado se tiene el conjunto L de literales y por el otro lado se tiene el conjunto C de cláusulas, donde cada cláusula $c_i \in C$ es adyacente a 3 literales $(l_{i,0}, l_{i,1}, l_{i,2}) \in L$ que hacen que ésta se satisfaga. La asignación a las variables se guarda en V, que es el testigo de la solución.

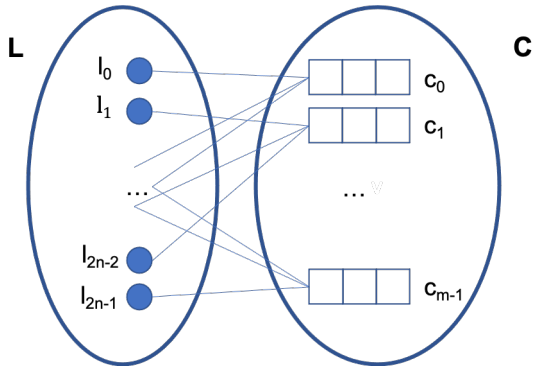

Figura 4.1: Representación de la fórmula $3SAT$ como gráfica bipartita.

La satisfabilidad de la fórmula y la estructura de la gráfica bipartita que se va a verificar están conectadas de la siguiente manera: existe una asignación de valores a las variables que hace que la fórmula se satisfaga si y sólo si por cada par de vértices en la gráfica asociados a una variable (esto es, sus literales), se puede seleccionar exactamente uno de estos dos vértices, de tal forma que para todo vértice de la gráfica asociado a una cláusula, ese vértice es adyacente al menos a uno de los vértices seleccionados.

Demostración. Sean:

- A: Existe una asignación de valores a las variables que hace que la fórmula se satisfaga.
- B: Por cada par de vértices en la gráfica asociados a una variable (esto es, sus literales), se puede seleccionar exactamente uno de estos dos vértices, de tal forma que para todo vértice de la gráfica asociado a una cláusula, ese vértice es adyacente al menos a uno de los vértices seleccionados.

$A \to B$

Consideremos como vértices seleccionados en la gráfica bipartita a los n vértices asociados a las literales que corresponden a la asignación (es decir, si a x_i se asigna a 1 (verdadero), entonces se selecciona el vértice asociado a la literal $2i$, y si se asigna a 0 (falso), entonces se selecciona el vértice asociado a la literal $2i + 1$. Puesto que toda cláusula contiene al menos una variable que se evalua a verdadero en esta asignación, entonces el respectivo vértice asociado a una cláusula en la gráfica bipartita es adyacente al vértice de la literal de esa variable que se asigna a verdadero y que fue seleccionado, por lo que el resultado se sigue.

$B \to A$

A partir de una selección de vértices del lado de las literales con la propiedad de tener exactamente un vértice por cada par asociado a una variable y tal que todo vértice del lado de cláusulas es adyacente a al menos un vértice seleccionado, consideremos la asignación de valores a las variables de la fórmula que consiste en asignar a cada variable el valor de verdad asociado al vértice correspondiente que fue seleccionado. Entonces toda cláusula de la fórmula, bajo esta asignación, tiene al menos una variable que se evaluó a verdadero: la que corresponde a algún vértice seleccionado adyacente al vértice de cláusula correspondiente. □

La representación de la fórmula se hará utilizando una matriz M de tamaño $m \times 3$ para representar las m cláusulas, donde $M[i][j]$ contendrá la j-ésima literal de la i-ésima cláusula. La representación de las literales se hará en un arreglo L de tamaño $2n$, donde la literal positiva de la i-ésima variable estará en la posición $2i$ y la literal negativa de la i-ésima variable estará en la posición $2i + 1$. El arreglo V de tamaño n representa al testigo de la solución.

Antes de iniciar el protocolo, el probador y el verificador generan el arreglo de aristas G a partir del conjunto de cláusulas C y del conjunto de literales L (agregando las aristas $e_{i,j}$). El probador también genera un vector de índices I de las aristas que dan solución a cada cláusula. En el vector I se guardan los índices de G que contienen a la arista e_i que hace que la cláusula c_j se satisfaga. La lista de índices I es parte de un pre-procesamiento que tiene A para poder responder los retos de forma más eficiente.

Ejemplo: Sea la fórmula $\phi = (x_0 \vee \bar{x_1} \vee x_2) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_2}) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_0}) \wedge (x_1 \vee \bar{x_3} \vee x_4)$ y el testigo de la solución $V = \{x_0 = 1, x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 0\}$. En la figura 4.2 se puede observar el conjunto de literales L, el conjunto de cláusulas C (representadas como una matriz M de tamaño $m \times 3$), el testigo de la solución V y las transformaciones después de aplicar la permutación π y el intercambio aleatorio $flip$ a las variables de la fórmula. En la figura 4.3 se puede observar la permutación π_M sobre la matriz M, los índices de pre-procesamiento I y la gráfica asociada G.

El protocolo consta de varias rondas. En cada ronda el probador A reescribe (lógicamente) la fórmula por una equivalente, renombrando literales y cláusulas. Además, intercambia mediante permutaciones aleatorias las variables por sus negaciones (decisiones binarias aleatorias). Al final, oculta cada una de las aristas de la gráfica G y cada una de las asignaciones de las variables del vector solución V. El probador envía el contenido comprometido de las $3m$ aristas de G y las n asignaciones V. El verificador B responde con un reto binario. En el primer escenario, el verificador solicita que se compruebe que la fórmula original es equivalente a la fórmula comprometida (enviada como la lista de aristas G). En el segundo escenario, el verificador elige una cláusula c_i y solicita que se muestre que alguna literal l_j se satisface con la asignación a una variable $v_k \in V$ del testigo comprometido.

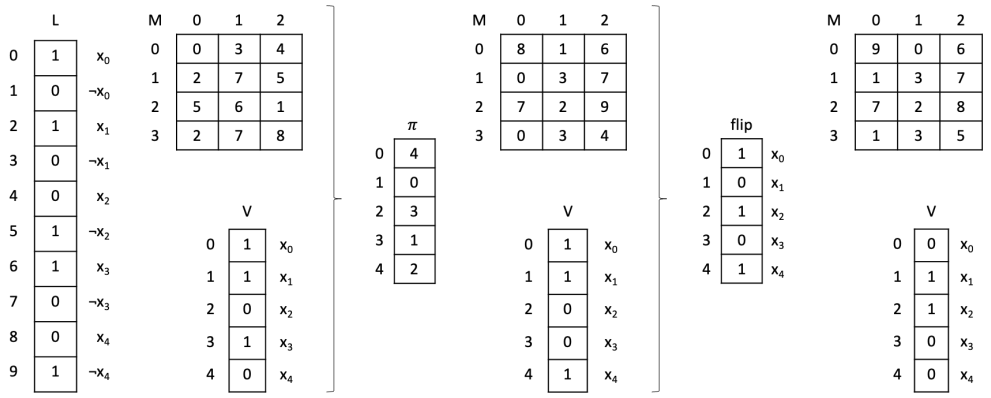

Figura 4.2: Ejemplo de la permutación π y el intercambio aleatorio $flip$ de una fórmula $3SAT$.

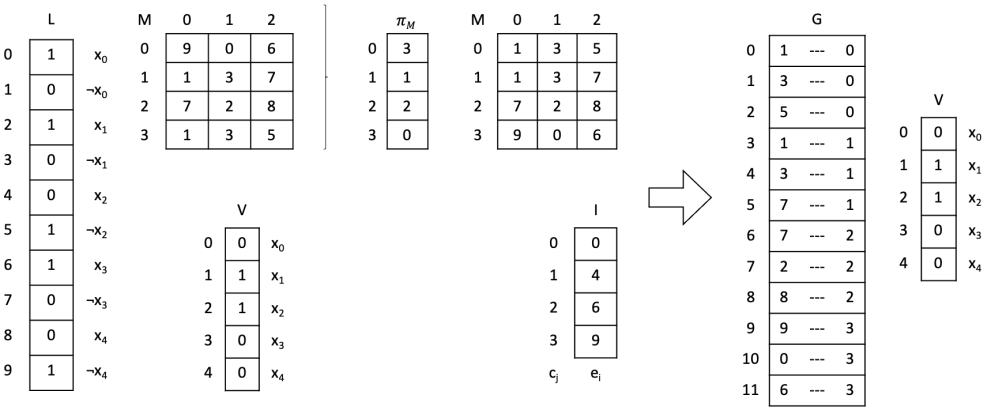

Figura 4.3: Ejemplo de la permutación π_M y de la representación de una fórmula $3SAT$ como gráfica bipartita.

4.3.1. Protocolo de conocimiento cero para 3SAT

A y B tienen como entrada la fórmula ϕ.

Recordemos que el valor de la literal positiva de la variable i-ésima se guarda con el índice $k = 2 * i$ y el valor de la literal negativa de la variable i-ésima se guarda con el índice $k = 2 * i + 1$, de tal modo que la operación $floor(k/2)$ permite obtener el índice de la variable.

Durante el protocolo, en cada ronda, A va a hacer uso de 3 permutaciones: una permutación π para renombrar las n variables de la fórmula; una permutación π_{ren_M} sobre los renglones de la matriz M para revolver las m cláusulas; y una permutación $flip$ para intercambiar el signo de las variables (con decisiones binarias aleatorias)

Los siguientes cuatro pasos se realizan $2m$ veces, cada vez utilizando variables aleatorias independientes.

1. El probador A inicia el protocolo y realiza lo siguiente en cada ronda:

   a) Renombra las variables de la fórmula:

   1: **while** $0 \leq i < m$ **do**
   2: $\quad pos = random.randint(0, m)$

```
    while pos in π do
        pos = random.randint(0, m)
    end while
    π[i] = pos
end while
for 0 ≤ i < m do
    for 0 ≤ j < 3 do
        // Obtiene el índice de la variable
        v_idx = floor(M[i][j]/2)
        // Obtiene el signo de la literal
        signo_{v_idx} = M[i][j] %2
        // Aplica la permutación al índice y le agrega el signo
        M1[i][j] = 2 * π[v_idx] + signo_{v_idx}
    end for
end for
for 0 ≤ i < n do
    // Aplica la permutación al índice conservando su asignación
    V1[π[i]] = V[i]
end for
```

b) Intercambia variables por sus negaciones, con decisiones binarias aleatorias:

```
for 0 ≤ i < n do
    flip[i] = random.randint(0, 1)
end for
for 0 ≤ i < m do
    for 0 ≤ j < 3 do
        // Obtiene el índice de la variable
        v_idx = floor(M1[i][j]/2)
        // Obtiene el signo de la variable
        signo_{v_idx} = M1[i][j] %2
        // Intercambia el signo de la variable con decisiones aleatorias
        signo_{v_idx} = (signo_{v_idx} + flip[v_idx]) %2
        // Almacena el índice con el nuevo signo calculado
        M2[i][j] = 2 * v_idx + signo_{v_idx}
    end for
end for
for 0 ≤ i < n do
    V2[i] = (V1[i] + flip[i]) %2
end for
```

c) Permuta las cláusulas de la fórmula:

```
while 0 ≤ i < m do
    pos = random.randint(0, m)
    while pos in π_{ren_M} do
        pos = random.randint(0, m)
    end while
```

6: $\pi_{ren_M}[i] = pos$
7: **end while**
8: **for** $0 \leq i < m$ **do**
9: $M3[\pi_{ren_M}[i]] = M2[i]$
10: **end for**

d) Genera la gráfica G:

1: $k = 0$
2: **for** $0 \leq i < m$ **do**
3: **for** $0 \leq j < 3$ **do**
4: $G[k] = (M3[i][j], i)$
5: $k = k + 1$
6: **end for**
7: **end for**

e) Oculta los valores[1] de G y $V2$:

1: $x = 256$
2: **for** $0 \leq i < 3 * m$ **do**
3: $ck_G[i] = SETUP(\{0,1\}^x)$
4: $G1[i] = COMMIT(G[i], ck_G[i])$
5: **end for**
6: **for** $0 \leq k < n$ **do**
7: $ck_V[k] = SETUP(\{0,1\}^x)$
8: $V3[k] = COMMIT(V2[k], ck_V[k])$
9: **end for**

A envía $G1$ y $V3$.

2. El verificador B responde al probador A con un reto aleatorio $bit \in_R \{0,1\}$ y un índice $0 \leq index < m$ de las cláusulas, tal que:

   a) Si $bit = 0$, B le solicita los cambios realizados para validar que la fórmula comprometida es equivalente a la fórmula original.

   b) Si $bit \neq 0$, B solicita que se demuestre que la cláusula c_{index} se satisface.

3. El probador A recibe el par $bit, index$:

   a) Si $bit = 0$, A envía el renombre de variables π, el intercambio de signo $\{flip\}$, la permutación π_{ren_M} sobre los renglones de la matriz M y las $3m$ llaves ck_G para generar $G1$.

   b) Si $bit \neq 0$, A envía el índice i y la llave $ck_G[i]$ para abrir la arista $e_i \in G1$ y el índice k y la llave $ck_V[k]$ para abrir el valor la variable $v_k \in V3$; si $index < 0$ o $index \geq m$ termina.

4. B verifica

[1] Para ocultar los valores se utiliza un esquema de compromiso.

a) Si $bit = 0$, B genera $G1'$ a partir de la fórmula ϕ original (representada como una matriz M) y si es igual a la gráfica $G1$ comprometida acepta y continua, en caso contrario rechaza y termina.

b) Si $bit \neq 0$, B realiza lo siguiente:

- Abre la asignación de la variable $v_k = DECOMMIT(V4[k], ck_V[k])$.
- Abre la arista $e_i = DECOMMIT(G1[i], ck_G[i])$.
- A partir de la arista $e_i = (l, c)$, valida que la cláusula $c = index$, que la literal l corresponde a la variable k-ésima (o es $2k$ o es $2k+1$) y que la variable l haga que la cláusula se satisfaga con la asignación v_k.

Si sí acepta y continúa, si no rechaza y termina.

Si el verificador B completa $2m$ rondas de los pasos descritos, entonces acepta.

4.3.2. Demostración del protocolo propuesto para 3SAT

El sistema de pruebas de conocimiento cero debe cumplir con las propiedades de completitud (*completeness*), solidez (*soundness*) y conocimiento cero (*zero knowledge*). A continuación, se analizarán estas tres propiedades para el protocolo 4.3.1.

Teorema 4.3.1. *El protocolo 4.3.1 constituye un sistema de pruebas interactivo de conocimiento cero para el problema de 3SAT.*

Demostración

- **Completitud:** $\forall\ x \in L$ y un probador honesto A, el verificador B acepta con probabilidad de 1.

  Si A es un probador honesto quiere decir que tiene la solución del problema y que sigue el protocolo, por lo tanto, va a poder responder correctamente cualquiera de los dos retos que un verificador B le solicite con una probabilidad de 1.

- **Solidez:** $\forall\ x \notin L$ y cualquier probador A^*, B acepta con probabilidad menor o igual a $\frac{1}{2}$.

  En cada iteración el verificador B envía el par $bit, index$. La probabilidad de que A^* supere el reto de B está en función de bit y de la probabilidad de que A^* cambie o no la fórmula original por una de la cual sí tenga solución en la ronda.

  En la tabla 4.1 se pueden ver las probabilidades de ser engañado en el protocolo. El bit de reto se elige con probabilidad uniforme, por ello la probabilidad de alguno de los dos posibles valores es $\frac{1}{2}$ (primer factor). La probabilidad de que A^* no cambie la fórmula original es p_i y la probabilidad de que sí la cambie es $1 - p_i$ (segundo factor). El tercer factor se refiere a la probabilidad de ser engañado y está en función del respectivo evento, para lo cual se tienen que analizar 4 casos:

  - Si $bit = 0$ y A^* cambia la fórmula original por alguna de la cual sí tenga solución, entonces la probabilidad de que A^* supere el reto es de 0, ya que se van a solicitar las transformaciones para comprobar que la fórmula no fue cambiada, pero como sí fue cambiada B rechazará la ejecución.

- Si $bit = 0$ y A^* no cambia la fórmula original, entonces la probabilidad de que A^* supere el reto es 1, ya que las transformaciones que envíe A^* permitirán obtener la fórmula comprometida a partir de la fórmula original.
- Si $bit \neq 0$ y A^* cambia la fórmula original por alguna de la cual sí tenga solución, entonces la probabilidad de que A^* supere el reto es de 1, ya que se va a solicitar que se verifique que la cláusula i-ésima se satisfaga y, como la fórmula fue cambiada, sí tiene la solución al reto.
- Si $bit \neq 0$ y A^* no cambia la fórmula original, entonces la probabilidad de que A^* supere el reto es, a lo más, $\frac{m-1}{m}$, ya que, por lo menos, una de las cláusulas no se va a satisfacer.

	Cambie $1-p_i$	No cambie p_i
$bit = 0$	$= \frac{1}{2} * (1-p_i) * 0$	$= \frac{1}{2} * p_i * 1$
$bit = 1$	$= \frac{1}{2} * (1-p_i) * 1$	$\leq \frac{1}{2} * p_i * (\frac{m-1}{m})$

Tabla 4.1: Probabilidades de ser engañado en el protocolo de 3SAT.

Entonces, la suma de probabilidades de la tabla 4.1 es, a lo más, $\frac{1}{2}(1+p_i(\frac{m-1}{m}))$. La probabilidad se maximiza cuando $p_i = 1$, entonces la probabilidad de ser engañado es $\leq \frac{2m-1}{2m} = 1-\frac{1}{2m}$ en cada ronda. Como el protocolo se repite $2m$ veces, la probabilidad de ser engañado en todas las rondas es a lo más $(1-\frac{1}{2m})^{2m}$. En [3] se muestra que

$$\lim_{x\to\infty}\left(1-\frac{1}{x}\right)^{x} = \frac{1}{e} \tag{4.1}$$

Además, como la función es creciente, se implica que para todo valor de m, el valor de la función es menor que $\frac{1}{2}$.

- **Conocimiento cero:** De forma intuitiva, el protocolo 4.3.1 es de conocimiento cero debido a que la única información que recibe el verificador B en cada ronda que se solicita la solución del problema (cuando $bit \neq 0$) es una arista y la asignación $\{0,1\}$ de la variable que corresponde al primer vértice de esa arista. Observemos que, debido a que las permutaciones que renombran las cláusulas, que renombran las variables y que cambian el signo de las variables en las fórmulas son todas ellas seleccionadas con respectivas probabilidades uniformes, entonces cualquier pareja de vértices (i,j) que pertenecen a la literal i-ésima y a la cláusula j-ésima en los conjuntos originales después de la permutación tienen exactamente la misma probabilidad de renombrarse como cualquier (i',j'), es decir, en cada ronda se abre una estructura y esa estructura puede ser cualquiera de las estructuras que hay ahí, todas con la misma probabilidad. Por ello, es importante que el probador A utilice permutaciones aleatorias independientes en cada ronda, para que tanto las cláusulas como las variables no estén correlacionadas entre una y otra ronda. Por tanto, intuitivamente se deduce que los valores abiertos en cada ronda no revelan ninguna información sobre la asignación de los valores de las variables que satisfacen la fórmula ϕ y además no se pueden combinar con valores abiertos en rondas anteriores para deducir más información.

  Ahora analizaremos la propiedad de conocimiento cero de manera formal.

Existe un simulador S^* tal que $\forall$ verificador B^* y $\forall\ x \in L$, S^* es una Máquina de Turing de tiempo polinomial esperado que utiliza a B^* para simular al verificador, de tal forma que la distribución de la salida del simulador S^* y la distribución de la salida del sistema de pruebas interactivo $\langle A, B^* \rangle$ son computacionalmente indistinguibles.

Recordemos que el simulador S^* utiliza a B^* como una subrutina, tiene el poder de modificar el estado de las cintas de B^* para ejecutar la n-ésima ronda.

El simulador S^* ejecuta los siguientes pasos $2m$ veces:

1. Elige la cláusula i con $c_i \in_R C$ y una literal $l_j \in c_i$ y hace que la asignación $l_j = v_k$ satisfaga la cláusula c_i. Genera el conjunto solución V con la asignación de la variable v_k que satisface la cláusula c_i y con $n-1$ asignaciones 0.
2. Realiza los mismos pasos que el probador A para generar $G1$ y $V3$. Escribe $G1$ y $V3$ en la cinta de comunicación de B^*.
3. Simula a B^*.
4. Lee el par $bit \in_R \{0,1\}$ e $index \in_R [0, m-1]$ generado por B^*.
   a) Si $bit = 0$, S^* pone en la cinta de comunicación de solo lectura de B^* el renombre de variables π, el intercambio de signo $\{flip\}$, la permutación π_{ren_M} sobre los renglones de la matriz M y las $3m$ llaves ck_i de los esquemas de compromiso para generar $G1$.
   b) Si $bit \neq 0$, se tienen dos opciones:
      1) Si $index = i$ (el mejor caso), S^* pone en la cinta de comunicación de solo lectura de B^* y en su cinta de salida el índice i y la llave $ck_G[i]$ para abrir la arista $e_i \in G1$ y el índice k y la llave $ck_V[k]$ para abrir el valor de la variable $v_k \in V3$.
      2) Si $index \neq i$ (el peor caso), S^* restablece las cintas de B^* a su estado inicial y regresa al punto 1.
5. Simula a B^*.

Ahora hay que probar que el simulador se ejecuta en tiempo polinomial esperado y que para todo B^* y para toda $x \in L$, la distribución de salida del simulador S^* es computacionalmente indistinguible de la distribución de salida del sistema de pruebas interactivo $\langle A, B^* \rangle$.

Proposición 4.3.2.1. *El simulador S^* termina en tiempo polinomial esperado.*

Demostración. El número de reintentos de una ronda del protocolo está en función del *bit* y del índice *index* que selecciona B^*. Cuando $bit = 0$, S^* responde al reto en el primer intento. Cuando $bit \neq 0$, la probabilidad de que $index = i$ es de $\frac{1}{m}$, entonces el número de intentos esperado para que $index = i$ en cada iteración es m (B.3.3). El número total de intentos después de $2m$ iteraciones es $O(m^2)$.

El tiempo esperado de ejecución del algoritmo es $O(|x|^2 * (p(|x|) + q(|x|) + q(|x|)^2))$, donde $|x|$ es la longitud de la entrada. El factor $|x|^2$ corresponde al número de iteraciones (parciales o completas), mientras que los términos del segundo factor son respectivamente los pasos del algoritmo en una iteración (parcial o completa) con la entrada x que corresponderían al probador ($p(|x|)$), al verificador ($q(|x|)$) y a la restauración de los valores de las cintas del

simulador ($q(|x|)^2$); por hipótesis sabemos que B^* se ejecuta en tiempo polinomial esperado. Además, claramente, $p(|x|)$ es polinomial, por lo tanto, el tiempo de ejecución total del simulador es de tiempo polinomial esperado. □

Por otro lado, la indistinguibilidad computacional está en función de la distribución del contenido de las cintas de S^* y de la distribución del contenido de las cintas del sistema de pruebas interactivo $\langle A, B^* \rangle$. Para realizar el análisis de las distribuciones se van a generar una serie de simuladores que muestren las transiciones desde el probador honesto A hasta el simulador S^*, comparando en cada transcripción contigua las distribuciones de salida.

1. Sea S_1 un simulador honesto que posee el testigo correcto de la solución w. S_1 ejecuta el siguiente algoritmo A_1:
   a) **Utiliza el testigo de la solución** w. Crea y se compromete con $G1$ y $V3$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*
      1) Si $bit = 0$, **envía** el renombre de variables π, el intercambio de signo $\{flip\}$, la permutación π_{ren_M} y las $3m$ llaves ck_G. Simula a B^*. B^* acepta.
      2) Si $bit \neq 0$, **envía** el índice i, la llave $ck_G[i]$, el índice k y la llave $ck_V[k]$. Simula a B^*, quien abre la asignación $v_k = DECOMMIT(V3[k], ck_V[k])$ y abre la arista $e_i = DECOMMIT(G1[i], ck_G[i])$ con $e_i = (l_j, c_{index})$, a partir de la literal l_j se obtiene la variable v_{l_j} y se le asigna el valor v_k (si $l_j \,\%2 = 0$ $v_{l_j} = v_k$, si no $v_{l_j} = \neg v_k$); como la asignación hace que la cláusula c_{index} se satisfaga, entonces B^* acepta.

   Sea τ_1 la transcripción de la conversación de S_1. Sea τ_0 la transcripción de la conversación de $\langle A, B^* \rangle$. S_1 actúa como el sistema de pruebas $\langle A, B^* \rangle$, por lo tanto, $\{\tau_0\} \approx_c \{\tau_1\}$.
2. Sea S_2 un simulador con el testigo correcto de la solución w. S_2 ejecuta el siguiente algoritmo A_2:
   a) **Elige una cláusula** i **con** $c_i \in_R C$ **y una literal** $l_j \in c_i$ **(sea la variable de esa literal** v_{l_j}**) que satisfaga la cláusula** c_i **a partir del testigo de la solución** w. Crea y se compromete con $G1$ y $V3$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*
      1) Si $bit = 0$, **envía** el renombre de variables π, el intercambio de signo $\{flip\}$, la permutación π_{ren_M} y las $3m$ llaves ck_G. Simula a B^*. B^* acepta.
      2) Si $bit \neq 0$ verifica:
         - Si $index = i$, **envía** el índice i, la llave $ck_G[i]$, el índice k y la llave $ck_V[k]$. Simula a B^*, quien abre la asignación $v_k = DECOMMIT(V3[k], ck_V[k])$ y abre la arista $e_i = DECOMMIT(G1[i], ck_G[i])$ con $e_i = (l_j, c_{index})$, a partir de la literal l_j se obtiene la variable v_{l_j} y se le asigna el valor v_k (si $l_j \,\%2 = 0$ $v_{l_j} = v_k$, si no $v_{l_j} = \neg v_k$); como la asignación hace que la cláusula c_{index} se satisfaga, entonces B^* acepta.

- **Si** $index \neq i$**, restablece las cintas de** B^* **al estado inicial de esta ronda y regresa al paso a).**

Sea τ_2 la transcripción de la conversación de S_2.

3. Sea S_3 un simulador con el testigo correcto de la solución w. S_3 ejecuta el siguiente algoritmo A_3:
   a) Elige una cláusula i con $c_i \in_R C$ y una literal $l_j \in c_i$ (sea la variable de esa literal v_{l_j}) que satisfaga la cláusula c_i a partir del testigo de la solución w. **Crea el vector solución** V **con la asignación de la variable** $v_k = v_{l_j}$ **(si** $l_j \,\%2 = 0$**) o** $v_k = \neg v_{l_j}$ **(si** $l_j \,\%2 = 1$**) que satisface la cláusula** c_i **y** $n-1$ **asignaciones a** 0. Crea y se compromete con $G1$ y $V3$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*
      1) Si $bit = 0$, **envía** el renombre de variables π, el intercambio de signo $\{flip\}$, la permutación π_{ren_M} y las $3m$ llaves ck_G. Simula a B^*. B^* acepta.
      2) Si $bit \neq 0$ verifica:
         - Si $index = i$, **envía** el índice i, la llave $ck_G[i]$, el índice k y la llave $ck_V[k]$. Simula a B^*, quien abre la asignación $v_k = DECOMMIT(V3[k], ck_V[k])$ y abre la arista $e_i = DECOMMIT(G1[i], ck_G[i])$ con $e_i = (l_j, c_{index})$, a partir de la literal l_j se obtiene la variable v_{l_j} y se le asigna el valor v_k (si $l_j \,\%2 = 0$ $v_{l_j} = v_k$, si no $v_{l_j} = \neg v_k$); como la asignación hace que la cláusula c_{index} se satisfaga, entonces B^* acepta.
         - Si $index \neq i$, restablece las cintas de B^* a su estado inicial y regresa al paso a).

   Sea τ_3 la transcripción de la conversación de S_3.

4. Sea $S_4 = S^*$ un simulador que ejecuta el algoritmo A_4:
   a) Elige una cláusula i con $c_i \in_R C$ y una literal $l_j \in c_i$ (sea la variable de esa literal v_k), **le asigna un valor a** v_k **que satisfaga la cláusula** c_i. Crea el vector solución V con la asignación de la variable v_k que hace que la literal l_j satisfaga la cláusula c_i y $n-1$ asignaciones a 0. Crea y se compromete con $M4$ y $V4$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*
      1) Si $bit = 0$, **envía** el renombre de variables π, el intercambio de signo $\{flip\}$, la permutación π_{ren_M} y las $3m$ llaves ck_G. Simula a B^*. B^* acepta.
      2) Si $bit \neq 0$ verifica:
         - Si $index = i$, **envía** el índice i, la llave $ck_G[i]$, el índice k y la llave $ck_V[k]$. Simula a B^*, quien abre la asignación $v_k = DECOMMIT(V3[k], ck_V[k])$ y abre la arista $e_i = DECOMMIT(G1[i], ck_G[i])$ con $e_i = (l_j, c_{index})$, a partir de la literal l_j se obtiene la variable v_{l_j} y se le asigna el valor v_k (si $l_j \,\%2 = 0$ $v_{l_j} = v_k$, si no $v_{l_j} = \neg v_k$); como la asignación hace que la cláusula c_{index} se satisfaga, entonces B^* acepta.
         - Si $index \neq i$, restablece las cintas de B^* a su estado inicial y regresa al paso a).

Sea τ_4 la transcripción de la conversación de S_4.

Las distribuciones $\{\tau_1\}$ y $\{\tau_2\}$ son computacionalmente indistinguibles ya que la única diferencia entre los simuladores S_1 y S_2 es que cuando $bit \neq 0$ A_2 repite el proceso hasta que el índice de la cláusula que recibe ($index$) sea el mismo que él eligió (i).

Las distribuciones $\{\tau_2\}$ y $\{\tau_3\}$ difieren en el conjunto solución V, mientras que S_2 lo envía a partir del testigo de la solución w, S_3 lo genera con la asignación v_k que satisface la cláusula c_i y $n-1$ asignaciones 0.

Proposición 4.3.2.2. *Para cualquier distinguidor D no uniforme PTP existe una función despreciable $\epsilon()$ tal que para todo $x \in L$ suficientemente grande, D distingue las distribuciones $\{\tau_2\}$ y $\{\tau_3\}$ con probabilidad a lo más $\epsilon(|x|)$.*

Demostración. Supongamos por contradicción que existe un distinguidor no uniforme D, una constante c y un número infinito de cadenas $x \in L$ tales que D puede distinguir las distribuciones $\{\tau_2(x)\}$ y $\{\tau_3(x)\}$ con probabilidad mayor que $\frac{1}{|x|^c}$.

Sean S' un simulador que actúa como S_2 y S'' un simulador que actúa como S_3. Consideremos la secuencia de simuladores híbridos $H_0, H_1, \ldots, H_{|2m|}$, el simulador H_k ejecuta las primeras k rondas como el simulador S'' y el resto de las rondas como el simulador S'. Nótese que la distribución de la salida de $\{H_0(x)\}$ es idéntica a la distribución de la salida de $\{S'(x)\}$ y la distribución de la salida de $\{H_{|2m|}(x)\}$ es idéntica a la distribución de la salida de $\{S''(x)\}$. Entonces, por el lema híbrido (lema 2.1.1), existe alguna ronda k para la cual D distingue las siguientes distribuciones:

- $\{H_k(x)\}$
- $\{H_{k+1}(x)\}$

con probabilidad mayor a $\frac{1}{|x|^c \cdot |2m|}$. Como $|x| > |3m|$, entonces esta probabilidad es mayor que $\frac{1}{|x|^{c+1}}$. Sin embargo, la única diferencia entre los simuladores H_k y H_{k+1} es que en la ronda $(k+1)$-ésima, el simulador H_{k+1} manda los valores comprometidos a partir del testigo de la solución w y en la ronda k-ésima el simulador H_k crea el vector solución con la asignación correcta de la variable v_k que satisface la literal l_j que pertenece a la cláusula c_i que seleccionó y $n-1$ asignaciones a 0. Entonces, D puede distinguir los valores comprometidos con probabilidad mayor a $\frac{1}{|x|^{c+1}}$.

Sin embargo, esto contradice la propiedad de ocultamiento de valores múltiples de los esquemas de compromiso 2.2.1, la cual asegura que existe alguna función despreciable $\epsilon_1()$ que acota por arriba la probabilidad de distinguir dos distribuciones que provienen de un número polinomial de valores ocultos mediante esquemas de compromiso. □

Las distribuciones $\{\tau_3\}$ y $\{\tau_4\}$ son computacionalmente indistinguibles ya que la única diferencia entre los algoritmos es la selección de la cláusula i (mientras que S_3 obtiene la cláusula y el valor de la literal a partir del testigo de la solución w, S_4 selecciona al momento una cláusula de la fórmula de entrada, selecciona una literal de esa cláusula y le asigna un valor que la satisfaga).

Por el lema híbrido 2.1.3 se sabe que la indistinguibilidad computacional es transitiva, por ello se puede afirmar que para todo B^* y para toda $x \in L$ las distribuciones $\{\tau_0\} \approx_c \{\tau_4\}$ y, por lo tanto, $\{\langle A, B^* \rangle\} \approx_c \{S^*\}$. ■.

4.3.3. Análisis de complejidad del protocolo de 3SAT

El sistema de pruebas $\langle A, B\rangle$ tiene como entrada común la fórmula que está formada por las literales L y las cláusulas C.

El protocolo se repite $2m$ veces.

1. **Probador** A: Reescribe la fórmula por una equivalente, para ello renombra las variables, intercambia las variables por sus negaciones (con decisiones binarias aleatorias), permuta las cláusulas y oculta los valores. Al final se compromete con la lista de aristas $G1$ y el vector de asignaciones válidas $V3$.

   El tiempo para renombrar las variables en el vector testigo V y en la matriz es $O(m)+O(n)$, el tiempo del intercambio de variables por sus negaciones es $O(m)+O(n)$, el tiempo de la permutación de las cláusulas es $O(m)$, el tiempo para generar la gráfica es $O(m)$ y el tiempo para ocultar los valores es $O(m)+O(n)$, por lo tanto, el tiempo de este paso es $O(m)+O(n)$. El espacio utilizado en la comunicación es $O(m)+O(n)$.

2. **Verificador** B: Responde al probador A con un reto aleatorio $bit \in_R \{0,1\}$ y un índice $0 \leq index < m$ de las cláusulas.

   El tiempo para elegir bit e $index$ es $O(1)$. El espacio requerido para enviar los valores es $O(1)$.

3. **Probador** A: Recibe el par $bit, index$ y envía π, $\{flip\}$, π_{ren_M} y las $m \times 3$ llaves ck_G, si $bit = 0$; o envía i, $ck_G[i]$, k y $ck_V[k]$, si $bit \neq 0$.

   El tiempo que requiere A para enviar lo solicitado a B es $O(m)+O(n)$. El espacio requerido para contestar al reto es $O(m)+O(n)$.

4. **Verificador** B: Verifica que la lista de aristas $G4$ comprometida se generó a partir de la fórmula original; o verifica que la asignación de la variable que descubre A satisface la cláusula que B solicitó.

   El tiempo para verificar el reto es $O(m)+O(n)$. El espacio que se requiere para verificar el reto es $O(m)+O(n)$.

Debido a que el protocolo se repite $2m$ veces, el tiempo de ejecución del protocolo es $O(m^2)+O(n*m)$ y el espacio utilizado en la comunicación es $O(m^2)+O(n*m)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(m^2)+O(n \cdot m)$$

Sin embargo, la expresión anterior se puede simplificar. Como cada variable ocurre en al menos una cláusula y cada cláusula tiene 3 variables, entonces $n \leq 3 \cdot m$. Por lo tanto, la complejidad espacial y temporal del protocolo es:

$$O(m^2).$$

Capítulo 5

Cobertura de conjunto

5.1. Definición del problema

En este capítulo se generará un protocolo de conocimiento cero *ad hoc* para el problema de Cobertura de conjunto (*set cover*).

Dados el conjunto de elementos $U = \{u_0, \ldots, u_{n-1}\}$, el conjunto de subconjuntos $S = \{s_0, \ldots, s_{m-1}\}$ y un entero positivo k el problema de *cobertura de conjunto* consiste en decidir si existen $\leq k$ subconjuntos de S que cubran todos los elementos de U [17].

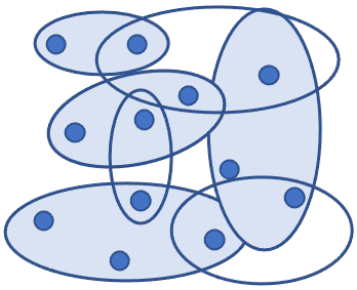

Figura 5.1: Selección de 4 conjuntos de S que cubren los 12 puntos de U.

5.2. Complejidad

En 1972 Richard Karp dio una lista de 21 problemas NP-Completos, dando sus reducciones a partir del problema SAT del teorema de Cook. El problema de Cobertura de conjunto es parte de estos 21 problemas. Con la reducción dada por Karp se demuestra que Cobertura de conjunto $\in$ NP-Completo [17].

5.3. Prueba de conocimiento cero para cobertura de conjunto

Supongamos que el probador A quiere demostrarle al verificador B que existen k subconjuntos de $S = \{s_0, \ldots, s_{m-1}\}$ que cubren los n elementos de $U = \{u_0, \ldots, u_{n-1}\}$ con conocimiento cero. La entrada común del protocolo interactivo la forman U, S y k. A posee como testigo de la solución a $C = \{c_0, \ldots, c_{k-1}\}$ cuyos elementos son los k índices de los subconjuntos de S que cubren todos los elementos de U. Es importante comentar que el número de subconjuntos de S que cubren todos los elementos de U pueden ser $< k$, en ese caso el vector C se puede rellenar con entradas nulas o repetidas para volverse de tamaño k.

La idea detrás del protocolo es trabajar no en el sistema de conjuntos directamente, sino con la gráfica bipartita asociada de forma natural a cada instancia, donde las dos clases de vértices son los elementos del universo U y los elementos de la familia de subconjuntos S; un vértice u_i es adyacente a otro s_j si $u_i \in s_j$. En concreto, trabajaremos con la lista de aristas E de la gráfica bipartita (E es un arreglo de $|E|$ renglones y 2 columnas, la primera columna contiene el índice i que representa al elemento $u_i \in U$ y la segunda columna contiene el índice j que representa al subconjunto $s_j \in S$, con $u_i \in s_j$) para poder establecer los mecanismos de transmisión de información con conocimiento cero que conforman al protocolo.

La instancia de Cobertura de conjunto con su gráfica asociada están conectadas de la siguiente manera: En la instancia de Cobertura de conjunto existe un conjunto de k elementos de S cuya unión es U si y sólo si en la gráfica bipartita asociada existe un conjunto de k vértices en la parte asociada a S de tal forma que todo elemento de U es adyacente a al menos uno de esos k vértices.

Antes de iniciar el protocolo, el probador y el verificador generan E a partir de los conjuntos U y S (construyendo las aristas $e_{i,j}$). El probador genera el vector I de tamaño $n \times 2$, el cual contiene en la primera columna los índices de las aristas de E que cubren todos los elementos de U y en la segunda columna los índices del conjunto solución C que contiene los k subconjuntos de S. De tal forma que cuando se solicite un elemento u_i, u_i será el índice del arreglo I, donde el valor $I[u_i][0] = k$ se refiere al índice k del arreglo E y el valor $I[u_i][1] = l$ se refiere al índice l del vector solución C, con:

- $E[k][0] = u_k$
- $E[k][1] = s_k$
- $C[l] = s_l$

Tal que $u_k = u_i$ y $s_k = s_l$. La lista de índices I es parte de un preprocesamiento que tiene A para poder responder los retos de forma eficiente.

Ejemplo: Sean $U = \{0, \cdots, 11\}$, $S = \{s_0 = \{0, 1\}, s_1 = \{2, 3, 4\}, s_2 = \{3, 6\}, s_3 = \{5, 6, 7, 8\}, s_4 = \{9, 10, 11\}, s_5 = \{1, 4, 9\}, s_6 = \{8, 11\}\}$, $k = 4$ y el testigo de la solución $C = \{s_0, s_1, s_3, s_4\}$; el conjunto U, el conjunto S, el testigo de la solución C, la gráfica E asociada a los conjuntos U y S y los índices del preprocesamiento I se pueden observar en la figura 5.2.

El protocolo consiste de varias rondas. En cada una de ellas, el probador A genera una gráfica bipartita isomorfa a la original (renombrando los elementos y los subconjuntos), al final, oculta los valores con un esquema de compromiso independiente por cada arista de E y cada índice de C.

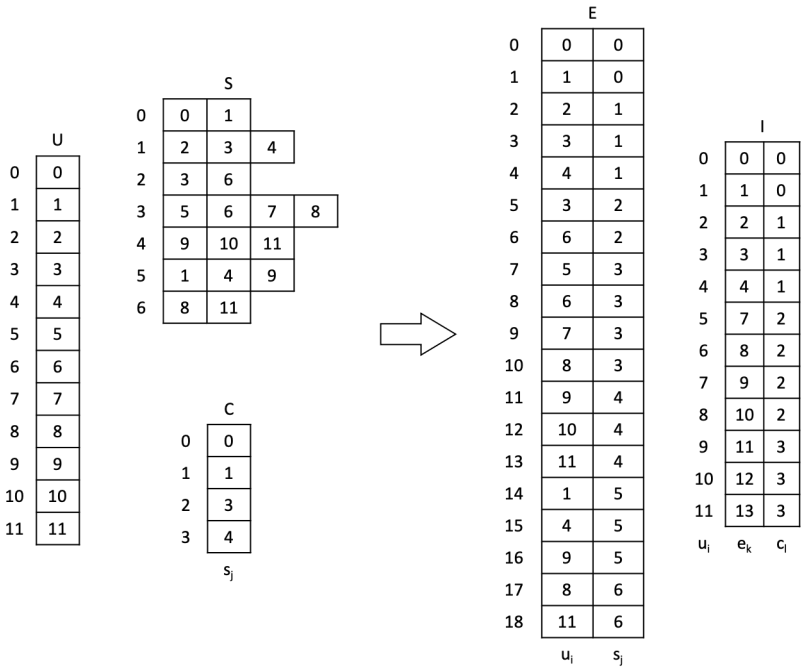

Figura 5.2: Representación de la gráfica bipartita E y del vector I a partir del conjunto U, el conjunto S y del testigo C.

El probador se compromete con los conjuntos E y C renombrados. El verificador B responde con un reto binario. En el primer escenario, se pide que A compruebe que la gráfica comprometida es isomorfa a la gráfica original; con esto el verificador comprueba que los conjuntos de entrada no fueron modificados. En el segundo escenario, el verificador B elige un índice $index$ de los elementos de U comprometidos y el probador A debe abrir un valor de C (digamos j) y una arista $a \in E$, tal que $a = (index, j)$; con eso se comprueba que C contiene como una de sus k entradas al conjunto j que cubre al elemento $index$.

5.3.1. Protocolo de conocimiento cero para cobertura de conjunto

A y B tienen como entrada común el conjunto U, el conjunto S y el entero positivo k. A tiene al vector C que es el conjunto solución de, a lo más, k índices de S que cubren los elementos de U. A partir de U y S se genera la lista de aristas E. A utiliza a E y al conjunto solución C para generar la lista de índices solución I.

Los siguientes cuatro pasos se realizan $2n$ veces, cada vez utilizando variables aleatorias independientes:

1. El probador A realiza lo siguiente:

   a) Renombra los elementos U y los subconjuntos S (aplicando una permutación aleatoria π_{ren_U} y π_{ren_S}, respectivamente):

   1: **for** $0 \leq i < n$ **do**
   2: $\quad U1[\pi_{ren_U}[i]] = U[i]$

3: **end for**
4: **for** $0 \leq i < m$ **do**
5: $S1[\pi_{ren_S}[i]] = S[i]$
6: **end for**
7: **for** $0 \leq i < len(E)$ **do**
8: $E0[i] = (\pi_{ren_U}[E[i][0]], \pi_{ren_S}[E[i][1]])$
9: **end for**
10: **for** $0 \leq i < k$ **do**
11: $C0[i] = \pi_{ren_S}[C[i]]$
12: **end for**

b) Revuelve las aristas $E0$ (aplicando una permutación aleatoria π_{ren_E}):

1: **for** $0 \leq i < len(E)$ **do**
2: $E1[\pi_{ren_E}[i]] = E0[i]$
3: **end for**

c) Revuelve los valores $C0$ (aplicando una permutación aleatoria π_{ren_C}):

1: **for** $0 \leq i < k$ **do**
2: $C1[\pi_{ren_C}[i]] = C0[i]$
3: **end for**

d) Oculta los valores[1] de $E1$ y $C1$:

1: $x = 256$
2: **for** $0 \leq i < len(E1)$ **do**
3: $ck_E[i] = SETUP(\{0,1\}^x)$
4: $E2[i] = COMMIT(E1[i], ck_E[i])$
5: **end for**
6: **for** $0 \leq i < len(C1)$ **do**
7: $ck_C[i] = SETUP(\{0,1\}^x)$
8: $C2[i] = COMMIT(C1[i], ck_C[i])$
9: **end for**

A envía $E2$ y $C2$.

2. El verificador B responde al probador A con un reto aleatorio $bit \in_R \{0,1\}$ y un valor $0 \leq index < n$ (que hace referencia a un elemento de U bajo el esquema de renombrado comprometido), tal que:

a) Si $bit = 0$, B le solicita las permutaciones π_{ren_U}, π_{ren_S} y π_{ren_E}, y las llaves ck_E del esquema de compromiso. Lo anterior para verificar que $E2$ es una gráfica bipartita isomorfa a la original.

b) Si $bit \neq 0$[2], B le solicita que abra un valor de $c_x \in C2$ y una arista $a_y \in E2$ tal que $c_x = j$ y $a_y = (index, j)$. Lo anterior para verificar el conjunto $C2$ comprometido tiene como uno de sus k valores al conjunto j que cubre al elemento $index$.

[1]Para ocultar los valores se utiliza un esquema de compromiso.

[2]Los casos cuando $bit = 1$ y cuando el bit es inválido se tratan aquí, debido a que un verificador tramposo B' que manda algo que no es un bit no podrá obtener información de la solución.

3. El probador A recibe el par $bit, index$:

   a) Si $bit = 0$, A envía las permutaciones π_{ren_U}, π_{ren_S} y π_{ren_E}, y las llaves ck_E del esquema de compromiso.

   b) Si $bit \neq 0$, A envía el índice x, la llave $ck_C[x]$, el índice y y la llave $ck_E[y]$ para descubrir el valor $c[x] = j$ y la arista $a[y] = (index, j)$. Si $index < 0$ o $index \geq n$, entonces termina.

4. B verifica:

   a) Si $bit = 0$, verifica que $E2$ es isomorfa a la gráfica bipartita original. Si sí acepta y continúa, si no, rechaza y termina.

   b) Si $bit \neq 0$, realiza lo siguiente:

   - Descubre el índice $c_x = DECOMMIT(C2[x], ck_C[x])$ con $c_x = s_x$.
   - Descubre la arista $a_y = DECOMMIT(E2[y], ck_E[y])$, con $a_y = (u_y, s_y)$.
   - Valida que $u_y = index$, que $s_x = s_y$ y que $0 \leq s_x < n$.

   Si sí acepta y continúa, si no, rechaza y termina.

Si el verificador B completa $2n$ rondas de los pasos descritos, entonces acepta.

5.3.2. Demostración del protocolo propuesto para cobertura de conjunto

El sistema de pruebas de conocimiento cero debe cumplir con las propiedades de completitud (completeness), solidez (soundness) y conocimiento cero (zero knowledge). A continuación, se analizarán estas propiedades para el protocolo 5.3.1.

Teorema 5.3.1. *El protocolo 5.3.1 constituye un sistema de pruebas interactivo de conocimiento cero para el problema de* cobertura de conjunto.

Demostración

- **Completitud:** $\forall\ x \in L$ y un probador honesto A, el verificador B acepta con probabilidad 1.

  Si A es un probador honesto quiere decir que tiene la solución del problema y que sigue el protocolo, por lo tanto, va a poder responder correctamente cualquiera de los dos retos que un verificador B le solicite con una probabilidad de 1.

- **Solidez:** $\forall\ x \notin L$ y cualquier probador A^*, B acepta con probabilidad menor o igual a $\frac{1}{2}$.

  En cada iteración el verificador B envía el par $bit, index$. La probabilidad de que A^* supere el reto de B está en función de bit y de la probabilidad de que en la iteración i A^* cambie (o no) los conjuntos originales por unos de los que sí conozca su solución.

  En la tabla 5.1 se pueden observar las probabilidades de ser engañado en el protocolo. El bit de reto se elige con probabilidad uniforme, por ello la probabilidad de alguno de los dos posibles valores es $\frac{1}{2}$ (primer factor). La probabilidad de que A^* no cambie los conjuntos

originales es p_i y la probabilidad de que sí los cambie es $1 - p_i$ (segundo factor). El tercer factor se refiere a la probabilidad de ser engañado y está en función del respectivo evento, para lo cual se tienen que analizar 4 casos:

- Si $bit = 0$ y A^* cambia los conjuntos originales por algunos de los cuales sí tenga solución, entonces la probabilidad de que A^* supere el reto es de 0, ya que se van a solicitar las transformaciones para comprobar que los conjuntos originales no fueron cambiados, pero como sí fueron cambiados B rechazará la ejecución.
- Si $bit = 0$ y A^* no cambia los conjuntos originales, entonces la probabilidad de que A^* supere el reto es 1, ya que las transformaciones que envíe A^* permitirán obtener $E2$ a partir de los conjuntos originales.
- Si $bit \neq 0$ y A^* cambia los conjuntos originales por algunos de los cuales sí tenga solución, entonces la probabilidad de que A^* supere el reto es de 1, ya que se va a solicitar que se abra el valor $c_x \in C2$ con $c_x = j$ y que se abra la arista $a_y \in E2$ con $a_y = (index, j)$ y como los conjuntos fueron cambiados, sí se tiene la solución al reto.
- Si $bit \neq 0$ y A^* no cambia los conjuntos originales, entonces la probabilidad de que A^* supere el reto es, a lo más, $\frac{n-1}{n}$, ya que, por lo menos, uno de los elementos renombrados de U no va a estar cubierto por los k subconjuntos comprometidos en $C2$.

	Sí cambia $1 - p$	No cambia p
$bit = 0$	$= \frac{1}{2} * (1 - p_i) * 0$	$= \frac{1}{2} * p_i * 1$
$bit = 1$	$= \frac{1}{2} * (1 - p_i) * 1$	$\leq \frac{1}{2} * p_i * (\frac{n-1}{n})$

Tabla 5.1: Probabilidades de ser engañado en el protocolo de Cobertura de conjunto.

Entonces, la suma de probabilidades de la tabla 5.1 es $\leq \frac{1}{2}(1 + p_i(\frac{n-1}{n}))$. La probabilidad se maximiza cuando $p_i = 1$, entonces la probabilidad de ser engañado es $\leq \frac{2n-1}{2n} = 1 - \frac{1}{2n}$ en cada ronda. Como el protocolo se repite $2n$ veces, la probabilidad de ser engañado en todas las rondas es a lo más $(1 - \frac{1}{2n})^{2n}$. Por lo tanto, como se puede ver en 4.1, la probabilidad de ser engañado es $\leq \frac{1}{e} < \frac{1}{2}$.

- **Conocimiento cero:** De forma intuitiva, el protocolo 5.3.1 es de conocimiento cero debido a que la única información que recibe el verificador B en cada ronda que se solicita la solución del problema (cuando $bit \neq 0$) es una arista $a \in E2$ y un índice $c_i \in C2$ que corresponde con un vértice de la arista a. Observemos que, debido a que las permutaciones que renombran los elementos y los subconjuntos son todas ellas seleccionadas con respectivas probabilidades uniformes, entonces cualquier pareja de vértices (i, j) que pertenecen al i-ésimo elemento y al j-ésimo subconjunto en los conjuntos originales, después de la permutación tienen exactamente la misma probabilidad de renombrarse como cualquier (i', j'), es decir, en cada ronda se abre una estructura y esa estructura puede ser cualquiera de las que hay ahí, todas con la misma probabilidad. Por ello, es importante que el probador A utilice permutaciones aleatorias independientes en cada ronda, para que tanto los elementos como los subconjuntos no estén correlacionados entre una y otra ronda. Por tanto, se deduce que los valores abiertos

en cada ronda no revelan ninguna información sobre la selección de los k subconjuntos de S que cubren a los elementos de U y además no se pueden combinar con valores abiertos en rondas anteriores para deducir más información.

Ahora analizaremos la propiedad de conocimiento cero de manera formal.

Existe un simulador S^* tal que $\forall$ verificador B^* y $\forall\ x \in L$, S^* es una Máquina de Turing de tiempo polinomial esperado que utiliza a B^* para simular al verificador, de tal forma que la distribución de la salida del simulador S^* y la distribución de la salida del sistema de pruebas interactivo $\langle A, B^* \rangle$ son computacionalmente indistinguibles.

Recordemos que el simulador S^* utiliza a B^* como una subrutina, tiene el poder de modificar el estado de las cintas de B^* para ejecutar la n-ésima ronda.

El simulador S^* ejecuta los siguientes pasos $2n$ veces:

1. Elige un índice ind de las aristas de E (con $0 \leq ind < len(E)$), tal que $a_{ind} = (r, j) \in E$, y crea el vector solución C con el valor j y $k-1$ valores 0.
2. Realiza los mismos pasos que el probador A para generar $E2$ y $C2$. Escribe $E2$ y $C2$ en la cinta de comunicación de B^*.
3. Simula a B^*.
4. Lee el par $bit \in_R \{0, 1\}$ e $index \in_R [0, n-1]$ generado por B^*.
   a) Si $bit = 0$, S^* pone en la cinta de comunicación de B^* las permutaciones π_{ren_U}, π_{ren_S} y π_{ren_E} y las llaves ck_E de los esquemas de compromiso.
   b) Si $bit \neq 0$, se tienen dos opciones:
      1) Si $index = r$ (el mejor caso), S^* pone en la cinta de comunicación y en su cinta de salida el índice x, la llave $ck_C[x]$, el índice ind y la llave $ck_E[ind]$ para descubrir el valor $c[x] = j$ y la arista $a[y] = (index, j)$.
      2) Si $index \neq r$ (el peor caso), S^* restablece las cintas de B^* a su estado inicial y regresa al punto 1.
5. Simula a B^*.

Ahora hay que probar que el simulador se ejecuta en tiempo polinomial esperado y que para todo B^* y para toda $x \in L$ la distribución de salida del simulador S^* es computacionalmente indistinguible de la distribución de salida del sistema de pruebas interactivo $\langle A, B^* \rangle$.

Proposición 5.3.2.1. *El simulador S^* termina en tiempo polinomial esperado.*

Demostración. El número de reintentos en cada ronda del algoritmo está en función del índice $index$ que selecciona B^*. Cuando $bit \neq 0$, la probabilidad de que $index = r$ es de $\frac{1}{n}$ con $n = |U|$, el número esperado de reintentos para que $index = r$ en cada iteración es n (B.3.3). El número total de reintentos después de $2n$ iteraciones es $O(n^2)$.

El tiempo esperado de ejecución del algoritmo es $O(|x|^2 * (p(|x|) + q(|x|) + q(|x|)^2))$, donde $|x|$ es la longitud de la entrada. El factor $|x|^2$ corresponde al número de rondas (parciales o completas), mientras que los términos del segundo factor son respectivamente los pasos del algoritmo en una ronda (parcial o completa) con la entrada x que corresponderían al

probador ($p(|x|)$), al verificador ($q(|x|)$) y a la restauración de los valores de las cintas del simulador ($q(|x|)^2$); por hipótesis sabemos que B^* se ejecuta en tiempo polinomial esperado. Además, claramente, $p(|x|)$ es polinomial, por lo tanto, el tiempo de ejecución total del simulador es de tiempo polinomial esperado. □

Por otro lado, la indistinguibilidad computacional está en función de la distribución del contenido de las cintas de S^* y de la distribución del contenido de las cintas del sistema de pruebas interactivo $\langle A, B^* \rangle$. Para realizar el análisis de las distribuciones se van a generar una serie de 4 simuladores que muestren las transiciones desde el probador honesto A hasta el simulador S^*, comparando en cada transcripción contigua las distribuciones de salida.

1. Sea S_1 un simulador honesto que posee el testigo correcto de la solución w. S_1 ejecuta el siguiente algoritmo A_1:
   a) **Utiliza el testigo de la solución** w. Crea y se compromete con $E2$ y $C2$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*:
   - Si $bit = 0$, **envía** las permutaciones π_{ren_U}, π_{ren_S} y π_{ren_E}, y las llaves ck_E del esquema de compromiso de $E2$. Simula a B^*. B^* acepta.
   - Si $bit \neq 0$, **envía** el índice x, la llave $ck_C[x]$, el índice y y la llave $ck_E[y]$. Simula a B^*, quien abre el valor $c_x = DECOMMIT(C2[x], ck_C[x])$ con $c_x = s_x$ y la arista $a_x = DECOMMIT(E2[y], ck_E[y])$ con $a_y = (u_y, s_y)$. Como $u_y = index$, $s_x = s_y$ y $0 \leq s_x < n$, B^* acepta.

   Sea τ_1 la transcripción de la conversación de S_1. Sea τ_0 la transcripción de la conversación de $\langle A, B^* \rangle$. S_1 actúa como el sistema de pruebas interactivo $\langle A, B^* \rangle$, por lo tanto, $\{\tau_0\} \approx_c \{\tau_1\}$.
2. Sea S_2 un simulador con el testigo correcto de la solución w. S_2 ejecuta el siguiente algoritmo A_2:
   a) **Elige una arista** $a_y \in E$ **y un valor** $c_x \in C$ **a partir del testigo de la solución** w**, tal que** $c_x = j$ **y** $a_y = (r, j)$. Crea y se compromete con $E2$ y $C2$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*:
   - Si $bit = 0$, **envía** las permutaciones π_{ren_U}, π_{ren_S} y π_{ren_E}, y las llaves ck_E del esquema de compromiso de $E2$. Simula a B^*. B^* acepta.
   - Si $bit \neq 0$ verifica:
     - Si $index = r$, **envía** el índice x, la llave $ck_C[x]$, el índice y y la llave $ck_E[y]$. Simula a B^*, quien abre el valor $c_x = DECOMMIT(C2[x], ck_C[x])$ con $c_x = j$ y la arista $a_y = DECOMMIT(E2[y], ck_E[y])$ con $a_y = (r, j)$. Como $r = index$, $j = j$ y $0 \leq j < n$, B^* acepta.
     - Si $index \neq r$, restablece las cintas de B^* a su estado inicial y regresa al paso a).

   Sea τ_2 la transcripción de la conversación de S_2.

3. Sea S_3 un simulador con el testigo correcto de la solución w. S_3 ejecuta el siguiente algoritmo A_3:
   a) Elige una arista $a_y \in E$ y un valor $c_x \in C$ a partir del testigo de la solución w, tal que $c_x = j$ y $a_y = (r, j)$. **Crea el vector solución** C **con el valor** j **y** $k-1$ **valores** 0. Crea y se compromete con $E2$ y $C2$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*:
   - Si $bit = 0$, **envía** las permutaciones π_{ren_U}, π_{ren_S} y π_{ren_E}, y las llaves ck_E del esquema de compromiso de $E2$. Simula a B^*. B^* acepta.
   - Si $bit \neq 0$ verifica:
     - Si $index = r$, **envía** el índice x, la llave $ck_C[x]$, el índice y y la llave $ck_E[y]$. Simula a B^*, quien abre el valor $c_x = DECOMMIT(C2[x], ck_C[x])$ con $c_x = j$ y la arista $a_y = DECOMMIT(E2[y], ck_E[y])$ con $a_y = (r, j)$. Como $r = index$, $j = j$ y $0 \leq j < n$, B^* acepta.
     - Si $index \neq r$, restablece las cintas de B^* a su estado inicial y regresa al paso a).

   Sea τ_3 la transcripción de la conversación de S_3.
4. Sea $S_4 = S^*$ un simulador que ejecuta el algoritmo A_4:
   a) **Elige una arista** a_{ind} **de** E **con** $a_{ind} = (r, j)$. Crea el vector solución C con el valor j y $k-1$ valores 0. Crea y se compromete con $E2$ y $C2$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*:
   - Si $bit = 0$, **envía** las permutaciones π_{ren_U}, π_{ren_S} y π_{ren_E}, y las llaves ck_E del esquema de compromiso de $E2$. Simula a B^*. B^* acepta.
   - Si $bit \neq 0$ verifica:
     - Si $index = r$, **envía** el índice x, la llave $ck_C[x]$, el índice ind y la llave $ck_E[ind]$. Simula a B^*, quien abre el valor $c_x = DECOMMIT(C2[x], ck_C[x])$ con $c_x = j$ y la arista $a_{ind} = DECOMMIT(E2[ind], ck_E[ind])$ con $a_{ind} = (r, j)$. Como $r = index$, $j = j$ y $0 \leq j < n$, B^* acepta.
     - Si $index \neq r$, restablece las cintas de B^* a su estado inicial y regresa al paso a).

   Sea τ_4 la transcripción de la conversación de S_4.

   La distribuciones $\{\tau_1\}$ y $\{\tau_2\}$ son computacionalmente indistinguibles ya que la única diferencia entre los simuladores S_1 y S_2 es que, cuando $bit \neq 0$ S_2 espera que el índice del elemento que recibe ($index$) sea el mismo que eligió (r).

   La distribuciones $\{\tau_2\}$ y $\{\tau_3\}$ difieren en el conjunto C, mientras que en S_2 se envía a partir del testigo de la solución w, en S_3 se genera un conjunto solución con el valor j de la arista $a_y = (r, j)$ y $k-1$ índices 0.

   Proposición 5.3.2.2. *Para cualquier distinguidor D no uniforme PTP existe una función despreciable $\epsilon()$ tal que para toda $x \in L$ suficientemente grande, D distingue las distribuciones $\{\tau_2\}$ y $\{\tau_3\}$ con probabilidad a lo más $\epsilon(|x|)$.*

Demostración. Supongamos por contradicción que existe un distinguidor no uniforme D, una constante c y un número infinito de cadenas $x \in L$ tales que D puede distinguir las distribuciones $\{\tau_2(x)\}$ y $\{\tau_3(x)\}$ con probabilidad mayor que $\frac{1}{|x|^c}$.

Sean S' un simulador que actúa como S_2 y S'' un simulador que actúa como S_3. Consideremos la secuencia de simuladores híbridos $H_0, H_1, \ldots, H_{|2n|}$, el simulador H_k ejecuta las primeras k rondas como el simulador S'' y el resto de las rondas como el simulador S'. Nótese que la distribución de la salida de $\{H_0(x)\}$ es idéntica a la distribución de la salida de $\{S'(x)\}$ y la distribución de la salida de $\{H_{|2n|}(x)\}$ es idéntica a la distribución de la salida de $\{S''(x)\}$. Entonces, por el lema híbrido (lema 2.1.1), existe alguna ronda k para la cual D distingue las siguientes distribuciones:

- $\{H_k(x)\}$
- $\{H_{k+1}(x)\}$

con probabilidad mayor a $\frac{1}{|x|^c \cdot |2n|}$. Como $|x| > |2n|$, entonces esta probabilidad es mayor que $\frac{1}{|x|^{c+1}}$. Sin embargo, la única diferencia entre los simuladores H_k y H_{k+1} es que en la ronda $(k+1)$-ésima, el simulador H_{k+1} manda los valores comprometidos a partir del testigo de la solución w y en la ronda k-ésima el simulador H_k crea el vector solución con el valor j y $k-1$ valores 0. Entonces, D puede distinguir los valores comprometidos con probabilidad mayor a $\frac{1}{|x|^{c+1}}$.

Sin embargo, esto contradice la propiedad de ocultamiento de valores múltiples de los esquemas de compromiso 2.2.1, la cual asegura que existe alguna función despreciable $\epsilon_1()$ que acota por arriba la probabilidad de distinguir dos distribuciones que provienen de un número polinomial de valores ocultos mediante esquemas de compromiso. □

La distribuciones $\{\tau_3\}$ y $\{\tau_4\}$ son computacionalmente indistinguibles ya que solo difieren en la elección del índice *ind* (mientras que S_3 la obtiene a partir del testigo de la solución w, S_4 lo elige al momento a partir de E).

Por el lema híbrido 2.1.3 se sabe que la indistinguibilidad computacional es transitiva, por ello se puede afirmar que para todo B^* y para toda $x \in L$ las distribuciones $\{\tau_0\} \approx_c \{\tau_4\}$ y, por lo tanto, $\{\langle A, B^* \rangle\} \approx_c \{S^*\}$. ■.

5.3.3. Análisis de complejidad del protocolo de cobertura del conjunto

El sistema de pruebas $\langle A, B \rangle$ tiene como entrada común al conjunto U, al conjunto S y al entero k.

El protocolo se repite $2n$ veces.

1. **Probador** A: Aplica una permutación aleatoria π_U al conjunto U, una permutación π_S al conjunto S y aplica las permutaciones π_U y π_S a los valores de E y de I. Después, aplica una permutación aleatoria π_E a E y una permutación aleatoria π_C a C. Al final, oculta los valores de E y C y se compromete con los conjuntos $E2$ y $C2$.

   El tiempo para realizar las permutaciones de U y S son, respectivamente, $O(n)$ y $O(m)$, el tiempo para cambiar los valores de C y E son, respectivamente, $O(k)$ y $O(|E|)$; el tiempo

de las permutaciones de C y E son, respectivamente, $O(k)$ y $O(|E|)$; los esquemas de compromiso toman un tiempo de $O(|E|) + O(k)$; por lo tanto, el tiempo de este paso es $O(|E|)$. El espacio utilizado en la comunicación es $O(|E|)$.

2. **Verificador** B: Elige un $bit \in_R \{0, 1\}$ y un elemento $index$ de los elementos de $0 \leq index < n$ comprometidos.

   El tiempo para elegir bit e $index$ es $O(1)$. El espacio requerido para guardar los valores es $O(1)$.

3. **Probador** A: Recibe el par $bit, index$ y envía las permutaciones π_{ren_U}, π_{ren_S}, π_{ren_E} y las llaves c_E del esquema de compromiso, si $bit = 0$; o envía el índice x, la llave $ck_C[x]$, el índice y y la llave $ck_E[y]$, si $bit \neq 0$.

   El tiempo que requiere A para enviar lo solicitado a B es $O(|E|)$. El espacio requerido para contestar al reto es $O(|E|)$.

4. **Verificador** B: Verifica que los conjuntos comprometidos son isomorfos a los originales con las transformaciones proporcionadas, si $bit = 0$; o abre la arista $a_y \in E2$ con $a_y = (u_y, s_y)$ y el índice $c_x \in C2$ con $c_x = s_x$ y verifica que $u_y = index$, $s_x = s_y$ y $0 \leq s_x < n$.

   El tiempo para verificar el reto es $O(|E|)$. El espacio que se requiere para verificar el reto es $O(|E|)$.

Debido a que el protocolo se repite $2n$ veces, el tiempo de ejecución del protocolo es $O(n*|E|)$ y el espacio utilizado en la comunicación es $O(n*|E|)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(n \cdot |E|).$$

Capítulo 6

Cobertura de puntos

6.1. Definición del problema

Dados n puntos $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$ en el plano, determinar si existen k líneas rectas $l_0, \cdots, l_{k-1}$ tal que todo (x_i, y_i) esté contenido en al menos una línea l_j [25].

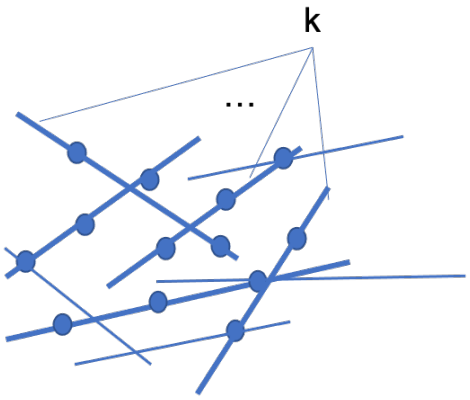

Figura 6.1: Cobertura de n puntos con k líneas.

Claramente, el problema de *Cobertura de puntos* es un caso particular del problema *Cobertura de conjunto* por lo que podríamos aplicar el mismo protocolo. Sin embargo, explotando la geometría de la instancia del problema de Cobertura de puntos, podemos representar la gráfica bipartita (puntos, líneas) de forma implícita, y así obtener un protocolo *ad hoc* más eficiente.

6.2. Complejidad

El problema de *Cobertura de puntos* (CP) se define de la siguiente manera: Dado un conjunto de puntos $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$, con x_i, y_i racionales para $i = 0, \ldots, n-1$ y un entero k, decidir si existe una colección de líneas rectas $l_0, \cdots, l_{k-1}$, tal que todo (x_i, y_i) esté contenido en al menos una l_j.

En [25] se demuestra que el problema de Cobertura de puntos es *NP-duro* a través de una reducción $3SAT \leq_p CP$.

6.3. Prueba de conocimiento cero para el problema de cobertura de puntos

Supongamos que el probador A quiere demostrarle al verificador B que k líneas cubren los n puntos del conjunto P con conocimiento cero. La entrada común para el protocolo interactivo son P y k. A posee el conjunto solución K. Se denota como $n = |P|$.

Para representar los n puntos se hará uso de una matriz P de tamaño $n \times 2$ (la primera columna del i-ésimo renglón contendrá la abscisa del punto i y la segunda columna del i-ésimo renglón contendrá la ordenada del punto i). La representación del vector solución será a través de un arreglo K de tamaño k el cual contendrá en cada elemento la ecuación de una recta en la forma $ax + b$.

Antes de iniciar el protocolo el probador A genera el vector de índices I. Cada índice del vector I corresponde con el punto p_j del conjunto P y cada valor del vector I (es decir, $I[p_j]$) contiene un índice k_i del vector K, de tal forma que el valor $K[k_i]$ contiene la recta l_i que cubre al punto p_j. La lista de índices I es parte de un preprocesamiento que tiene A para responder los retos de forma eficiente.

Ejemplo: Sean $P = \{(1,1),(2,1.5),(3,2),(3,1),(4,1),(3.5,2),(4,3),(2,3)\}$ y $k = 3$. El conjunto de puntos P, el testigo de la solución K y el vector de índices I se pueden observar en la figura 6.2.

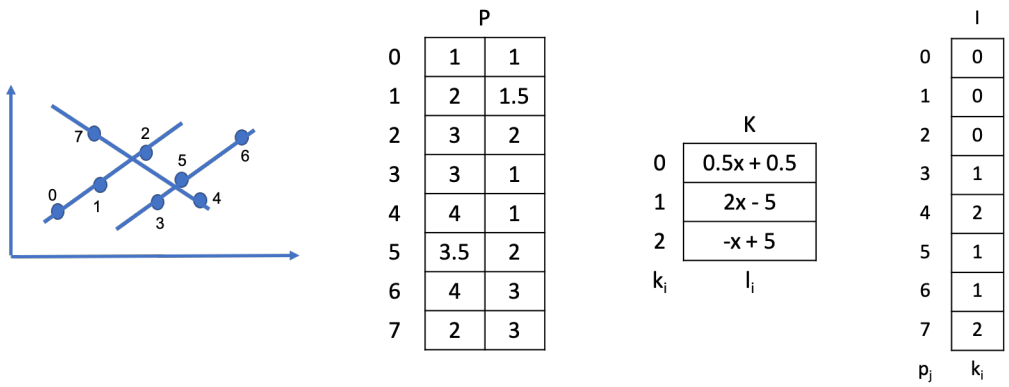

Figura 6.2: Representación de los conjuntos P, K e I.

El protocolo consta de varias rondas. En cada ronda el probador A inicia el protocolo, primero realiza una transformación aleatoria ϕ del plano (generando $P1$), después realiza una permutación aleatoria π_P de los puntos transformados y una permutación aleatoria π_K sobre el conjunto solución (generando $P2$, $K1$ y $K2$), y oculta cada valor de los conjuntos[1] (generando $P3$ y $K3$); al final envía a B los valores comprometidos $P3$ y $K3$.

El verificador B responde con un reto binario *bit* y un elemento *index* del conjunto $P3$. En el primer escenario, B solicita que se compruebe que el conjunto original P es geométricamente equivalente al conjunto comprometido $P3$. En el segundo escenario, B solicita que se compruebe que para algún índice del conjunto solución $s_x \in K3$ contiene la ecuación de la recta l_y que pasa por el punto $p_{index} \in P3$.

[1]Para ocultar los valores se utiliza un esquema de compromiso.

6.3.1. Protocolo de conocimiento cero para cobertura de puntos

A y B tienen como entrada común el conjunto de puntos P y el entero positivo k.

Durante el protocolo A va a hacer uso de una transformación y tres permutaciones: una transformación[2] aleatoria ϕ al conjunto P, una permutación π_P sobre P para revolver los puntos, y una permutación π_K sobre K para revolver las ecuaciones del testigo de la solución.

Los siguientes cuatro pasos se realizan $2n$ veces, cada vez utilizando variables aleatorias independientes:

1. El probador A inicia el protocolo y realiza lo siguiente:

   - Al conjunto de puntos P le aplica una transformación ϕ, una permutación π_P y un esquema de compromiso:

     for $0 \leq i < n$ **do**
     $P1[i] = \phi(P[i])$
     end for
     for $0 \leq i < n$ **do**
     $P2[\pi_P[i]] = P1[i]$
     end for
     $x = 256$
     for $0 \leq i < n$ **do**
     $ck_P[i] = SETUP(\{0,1\}^x)$
     $P3[i] = COMMIT(P2[i], ck_P[i])$
     end for

   - Al conjunto solución K le aplica una transformación ϕ, una permutación π_K y un esquema de compromiso:

     for $0 \leq i < k$ **do**
     $K1[i] = \phi(K[i])$
     end for
     for $0 \leq i < k$ **do**
     $K2[\pi_K[i]] = K1[i]$
     end for
     $x = 256$
     for $0 \leq i < k$ **do**
     $ck_K[l] = SETUP(\{0,1\}^x)$
     $K3[l] = COMMIT(K2[l], ck_K[l])$
     end for

   A envía $P3$ y $K3$.

2. El verificador B responde al probador A con un reto aleatorio $bit \in_R \{0,1\}$ y un índice $index$ con $0 \leq index < n$, tal que:

   a) Si $bit = 0$, B le solicita los cambios realizados al conjunto P para verificar que el conjunto de puntos $P3$ comprometido es geométricamente equivalente al conjunto original P.

[2]La transformación incluye rotación, traslación y multiplicación por un escalar (homotecia).

b) Si $bit \neq 0$, B le solicita que muestre el elemento $s_x \in K3$ contiene la ecuación de una línea l_x que pasa por el punto $p_{index} \in P3$.

3. El probador A recibe el par $bit, index$:

   a) Si $bit = 0$, A envía la transformación aleatoria ϕ, la permutación π_P sobre los elementos del vector de puntos P y las llaves ck_P para generar $P3$ a partir de P.

   b) Si $bit \neq 0$, A envía el índice x y la llave $ck_K[x]$ para abrir un valor $s_x \in K3$, y la llave $ck_K[index]$ para abrir el punto $p_{index} \in P3$; si $0 \leq p_{index} < n$. Si $index < 0$ o $index \geq n$, entonces termina.

4. B verifica:

   a) Si $bit = 0$, B genera $P3'$ a partir de P y si es igual al conjunto $P3$ comprometido acepta y continúa, si no, rechaza y termina.

   b) Si $bit \neq 0$, B realiza lo siguiente:

   - Abre el valor $s_x = DECOMMIT(K3[x], ck_K[x])$, con $s_x = l_y$.
   - Abre el punto $p_{index} = DECOMMIT(P3[index], ck_P[index])$, con $p_{index} = (p_x, p_y)$
   - Verifica que la recta l_y es incidente en el punto $p_{index} = (p_x, p_y)$.

   Si sí acepta y continúa, si no, rechaza y termina.

Si el verificador completa $2n$ iteraciones de los pasos descritos, entonces acepta.

6.3.2. Demostración del protocolo propuesto para cobertura de puntos

El sistema de pruebas de conocimiento cero debe cumplir con las propiedades de completitud (completeness), solidez (soundness) y conocimiento cero (zero knowledge). A continuación, se analizarán estas propiedades para el protocolo 6.3.1.

Teorema 6.3.1. *El protocolo 6.3.1 constituye un sistema de pruebas interactivo de conocimiento cero para el problema de cubrir un conjunto de n puntos con k líneas.*

Demostración

- **Completitud:** $\forall\ x \in L$ y un probador honesto A, el verificador B acepta con probabilidad de 1.

  Si A es un probador honesto quiere decir que tiene la solución del problema y que sigue el protocolo, por lo tanto, va a poder responder correctamente cualquiera de los dos retos que un verificador B le solicite con una probabilidad de 1.

- **Solidez:** $\forall\ x \notin L$ y cualquier probador A^*, B acepta con probabilidad menor o igual a $\frac{1}{2}$.

  En cada iteración el verificador B envía el par $bit, index$. La probabilidad de que A^* engañe a B está en función de bit y de la probabilidad de que A^* cambie o no los conjuntos originales por uno del cual sí conozca su solución.

  En la tabla 6.1 se pueden observar las probabilidades de ser engañado en el protocolo. El bit de reto se elige con probabilidad uniforme, por ello la probabilidad de los dos posibles

valores es $\frac{1}{2}$ (primer factor). La probabilidad de que A^* no cambie los conjuntos originales es p_i y la probabilidad de que sí los cambie es $1 - p_i$ (segundo factor). El tercer factor se refiere a la probabilidad de ser engañado y está en función del respectivo evento, para lo cual se tienen que analizar 4 casos:

- Si $bit = 0$ y A^* cambia el conjunto original por alguno del que sí tenga solución, entonces la probabilidad de que A supere el reto es de 0, ya que se van a solicitar las transformaciones para comprobar que el conjunto original no fue cambiado, pero como sí fue cambiado B rechazará la ejecución.
- Si $bit = 0$ y A^* no cambia el conjunto original, entonces la probabilidad de que A supere el reto es 1, ya que las transformaciones que envíe A^* permitirán obtener el conjunto comprometido a partir del conjunto original.
- Si $bit \neq 0$ y A^* cambia el conjunto original por alguno del que sí tenga solución, entonces la probabilidad de que A supere el reto es de 1, ya que se va a solicitar que se verifique que el punto $p_{index} \in P_3$ esté cubierto por alguna línea l_y, por lo que tiene que abrir el valor $s_x \in K_3$ con $s_x = l_y$, y, como el conjunto fue cambiado, sí se tiene la solución al reto.
- Si $bit \neq 0$ y A^* no cambia el conjunto original, entonces la probabilidad de que A supere el reto es, a lo más, $\frac{n-1}{n}$, ya que, por lo menos, uno de los puntos de P_3 no va a estar cubierto por alguna línea del conjunto solución K_3.

	Sí cambia $1 - p_i$	No cambia p_i
$bit = 0$	$= \frac{1}{2} * (1 - p_i) * 0$	$= \frac{1}{2} * p_i * 1$
$bit = 1$	$= \frac{1}{2} * (1 - p_i) * 1$	$\leq \frac{1}{2} * p_i * (\frac{n-1}{n})$

Tabla 6.1: Probabilidades de ser engañado en el protocolo de Cobertura de puntos.

Entonces, la suma de probabilidades de la tabla 6.1 es $\leq \frac{1}{2}(1 + p_i(\frac{n-1}{n}))$. La probabilidad se maximiza cuando $p_i = 1$, entonces la probabilidad de ser engañado es $\leq \frac{2n-1}{2n} = 1 - \frac{1}{2n}$ en cada ronda. Como el protocolo se repite $2n$ veces, la probabilidad de ser engañado en todas las rondas es a lo más $(1 - \frac{1}{2n})^{2n}$. Por lo tanto, como se puede ver en 4.1, la probabilidad de ser engañado es $\leq \frac{1}{e} < \frac{1}{2}$.

- **Conocimiento cero:** De forma intuitiva, el protocolo 6.3.1 es de conocimiento cero debido a que la única información que recibe el verificador B en cada ronda que se solicita la solución del problema (cuando $bit \neq 0$) es la ecuación de una recta de la forma $ax + b$ y un punto de la forma (x, y). Debido a que las permutaciones que renombran los puntos y los valores del conjunto solución son todas ellas seleccionadas con respectivas probabilidades uniformes, entonces para cualquier valor i_1 y cualquier punto p_{index} en el conjunto original, después de la permutación tienen exactamente la misma probabilidad de renombrarse como i'_1 y p'_{index}, es decir, en cada ronda se abre una estructura y esa estructura puede ser cualquiera de la que hay ahí, todas con la misma probabilidad. Por ello, es importante que el probador A utilice permutaciones aleatorias independientes en cada ronda, para que los puntos, los

índices solución y las rectas no estén correlacionados entre una y otra ronda. Por tanto, se deduce que los valores abiertos en cada ronda no revelan ninguna información sobre las rectas que dan solución al problema y además no se pueden combinar con valores abiertos en rondas anteriores para deducir más información.

Ahora analizaremos la propiedad de conocimiento cero de manera formal.

Existe un simulador S^* tal que $\forall$ verificador B^* y $\forall\ x \in L$, S^* es una máquina de Turing de tiempo polinomial esperado que utiliza a B^* para simular al verificador, de tal forma que la distribución de la salida del simulador S^* y la distribución de la salida del sistema de pruebas interactivo $\langle A, B^* \rangle$ son computacionalmente indistinguibles.

Recordemos que el simulador S^* utiliza a B^* como una subrutina, tiene el poder de modificar el estado de las cintas de B^* para ejecutar la n-ésima ronda.

El simulador S^* ejecuta los siguientes pasos $2n$ veces:

1. Elige una recta l_y que cubra algún punto $p_r \in P$ y genera el conjunto solución K con la ecuación de la recta $k_s = l_y$ y $k - 1$ valores 0.
2. Realiza los mismos pasos que el probador A para generar $P3$ y $K3$. Escribe $P3$ y $K3$ en la cinta de comunicación de B^*.
3. Simula a B^*.
4. Lee el par $bit \in_R \{0, 1\}$ e $index \in_R [0, n-1]$ generado por B^*.
   a) Si $bit = 0$, S^* pone en la cinta de comunicación de B^* la transformación aleatoria ϕ, la permutación π_P sobre los elementos de P y las llaves ck_P para generar $P3$.
   b) Si $bit \neq 0$ se tienen dos opciones
      1) Si $index = r$ (el mejor caso) pone en la cinta de comunicación de solo lectura de B^* y en su cinta de salida el índice s y la llave $ck_K[s]$ para abrir el valor $K_3[s]$ y la llave ck_{index} para abrir el punto $P3[index]$.
      2) Si $index \neq r$ (el peor caso) restablece las cintas de B^* a su estado inicial y regresa al punto 1.
5. Simula a B^*.

Ahora hay que probar que el simulador se ejecuta en tiempo polinomial esperado y que para todo B^* y para toda $x \in L$ la distribución de salida del simulador S^* es computacionalmente indistinguible de la distribución de salida del sistema de pruebas interactivo $\langle A, B^* \rangle$.

Proposición 6.3.2.1. *El simulador S^* termina en tiempo polinomial esperado.*

Demostración. El número de intentos esperado de esta ronda está en función del *bit* y del índice *index* que selecciona B^*. Cuando $bit = 0$, S^* responde con probabilidad de 1. Cuando $bit \neq 0$, la probabilidad de que $index = r$ es de $\frac{1}{n}$ con $n = |P|$, por tanto, el número de intentos esperado para que $index = r$ en cada iteración es n (B.3.3). El número total de reintentos después de $2n$ iteraciones es $O(n^2)$.

El tiempo esperado de ejecución del algoritmo es $O(|x|^2 * (p(|x|) + q(|x|) + q(|x|)^2))$, donde $|x|$ es la longitud de la entrada. El factor $|x|^2$ corresponde al número de iteraciones (parciales

o completas), mientras que los términos del segundo factor son respectivamente los pasos del algoritmo en una iteración (parcial o completa) con la entrada x que corresponderían al probador ($p(|x|)$), al verificador ($q(|x|)$) y a la restauración de los valores de las cintas del simulador ($q(|x|)^2$); por hipótesis sabemos que B^* se ejecuta en tiempo polinomial esperado. Además, claramente, $p(|x|)$ es polinomial, por lo tanto, el tiempo de ejecución total del simulador es de tiempo polinomial esperado. □

Por otro lado, la indistinguibilidad computacional está en función de la distribución del contenido de las cintas de S^* y de la distribución del contenido de las cintas del sistema de pruebas interactivo $\langle A, B^* \rangle$. Para realizar el análisis de las distribuciones se van a generar una serie de simuladores que muestren las transiciones desde el probador honesto A hasta el simulador S^*, comparando en cada transcripción contigua las distribuciones de salida.

1. Sea S_1 un simulador honesto con el testigo correcto de la solución w. S_1 ejecuta el siguiente algoritmo A_1:
   a) **Utiliza el testigo de la solución** w. Crea y se compromete con $P3$ y $K3$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*:
      1) Si $bit = 0$ **envía** la transformación aleatoria ϕ, la permutación π_P y las llaves ck_P. Simula a B^*. B^* acepta.
      2) Si $bit \neq 0$ **envía** el índice x, la llave $ck_K[x]$ y la llave $ck_P[index]$. Simula a B^*. Como el índice $s_x = DECOMMIT(K3[x], ck_K[x])$ contiene la ecuación de la recta l_y y l_y es incidente en el punto $p_{index} = DECOMMIT(P3[index], ck_P[index])$, B^* acepta.

   Sea τ_1 la transcripción de la conversación de S_1. Sea τ_0 la transcripción de la conversación de $\langle A, B^* \rangle$. S_1 actúa como el sistema de pruebas $\langle A, B^* \rangle$, por lo tanto, $\{\tau_0\} \approx_c \{\tau_1\}$.
2. Sea S_2 un simulador con el testigo correcto de la solución w. S_2 ejecuta el siguiente algoritmo A_2:
   a) **Elige una recta** l_y **que cubra algún punto** $p_r \in P$ **con** $s_x = l_y \in K$ **a partir del testigo de la solución** w. Crea y se compromete con $P3$ y $K3$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*:
      1) Si $bit = 0$ **envía** la transformación aleatoria ϕ, la permutación π_P y las llaves ck_P. Simula a B^*. B^* acepta.
      2) Si $bit = 0$ verifica:
         - Si $index = r$ **envía** el índice x, la llave $ck_K[x]$ y la llave $ck_P[index]$. Simula a B^*. Como el índice $s_x = DECOMMIT(K3[x], ck_K[x])$ contiene la ecuación de la recta l_y y la recta l_y es incidente en el punto $p_{index} = DECOMMIT(P3[index], ck_P[index])$, B^* acepta.
         - **Si** $index \neq r$**, restablece las cintas de** B^* **a su estado inicial y regresa al paso** *a*).

Sea τ_2 la transcripción de la conversación S_2.

3. Sea S_3 un simulador con el testigo correcto de la solución w. S_3 ejecuta el siguiente algoritmo A_3.
   a) Elige una recta l_y que pasa por algún punto $p_r \in P$ con $s_x = l_y \in K$ a partir del testigo de la solución w. **Crea el conjunto solución K con la ecuación de la recta l_y en el índice x y $k-1$ índices** 0. Crea y se compromete con $P3$ y $K3$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*:
      1) Si $bit = 0$ **envía** la transformación aleatoria ϕ, la permutación π_P y las llaves ck_P. Simula a B^*. B^* acepta.
      2) Si $bit \neq 0$ verifica:
         - Si $index = r$ **envía** el índice x, la llave $ck_K[x]$ y la llave $ck_P[index]$. Simula a B^*. Como el índice $s_x = DECOMMIT(K3[x], ck_K[x])$ contiene a la ecuación de la recta $s_x = l_y$ y la recta l_y es incidente en el punto $p_{index} = DECOMMIT(P3[index], ck_P[index])$, B^* acepta.
         - **Si $index \neq r$, restablece las cintas de B^* a su estado inicial y regresa al paso** *a*).

   Sea τ_3 la transcripción de la conversación S_3.

4. Sea $S_4 = S^*$ un simulador que ejecuta el algoritmo A_4:
   a) **Elige una recta l_y que pasa por algún punto** $p_r \in P$. Crea el conjunto solución K con la ecuación de la recta l_y en el índice x (con $0 \leq x < k$) y $k-1$ índices 0. Crea y se compromete con $P3$ y $K3$. Simula a B^*.
   b) **Recibe** *bit* e *index*.
   c) Verifica el *bit*:
      1) Si $bit = 0$ **envía** la transformación aleatoria ϕ, la permutación π_P y las llaves ck_P. Simula a B^*. B^* acepta.
      2) Si $bit \neq 0$ verifica:
         - Si $index = r$ **envía** el índice x, la llave $ck_K[x]$ y la llave $ck_P[index]$. Simula a B^*. Como el índice $s_x = DECOMMIT(K3[x], ck_K[x])$ contiene la ecuación de la recta $s_x = l_y$ y la recta l_y es incidente en el punto $p_{index} = DECOMMIT(P3[index], ck_P[index])$, B^* acepta.
         - **Si $index \neq r$, restablece las cintas de B^* a su estado inicial y regresa al paso** *a*).

   Sea τ_4 la transcripción de la conversación S_4.

Las distribuciones $\{\tau_1\}$ y $\{\tau_2\}$ son computacionalmente indistinguibles ya que la única diferencia entre los simuladores S_1 y S_2 es que cuando $bit \neq 0$, S_2 repite el algoritmo hasta que el punto que solicita B^* (p_{index}) sea el mismo que él eligió (p_r).

Las distribuciones $\{\tau_2\}$ y $\{\tau_3\}$ difieren en el conjunto solución K, mientras que S_2 lo envía a partir del testigo de la solución w, S_3 lo genera con la ecuación de la recta l_y en el índice x y $k-1$ índices 0.

Proposición 6.3.2.2. *Para cualquier distinguidor D no uniforme PTP existe una función despreciable $\epsilon()$ tal que para todo $x \in L$ suficientemente grande, D distingue las distribuciones $\{\tau_2\}$ y $\{\tau_3\}$ con probabilidad a lo más $\epsilon(|x|)$.*

Demostración. Supongamos por contradicción que existe un distinguidor no uniforme D, una constante c y un número infinito de cadenas $x \in L$ tales que D puede distinguir las distribuciones $\{\tau_2(x)\}$ y $\{\tau_3(x)\}$ con probabilidad mayor que $\frac{1}{|x|^c}$.

Sean S' un simulador que actúa como S_2 y S'' un simulador que actúa como S_3. Consideremos la secuencia de simuladores híbridos $H_0, H_1, \ldots, H_{2n}$, el simulador H_k ejecuta las primeras k rondas como el simulador S'' y el resto de las rondas como el simulador S'. Nótese que la distribución de la salida de $\{H_0(x)\}$ es idéntica a la distribución de la salida de $\{S'(x)\}$ y la distribución de la salida de $\{H_{|2n|}(x)\}$ es idéntica a la distribución de la salida de $\{S''(x)\}$. Entonces, por el lema híbrido (lema 2.1.1), existe alguna k para la cual D distingue las siguientes distribuciones:

- $\{H_k(x)\}$
- $\{H_{k+1}(x)\}$

con probabilidad mayor a $\frac{1}{|x|^c \cdot 2n}$. Como $|x| > |2n|$, entonces esta probabilidad es mayor que $\frac{1}{|x|^{c+1}}$. Sin embargo, la única diferencia entre los simuladores H_k y H_{k+1} es que en la ronda $(k+1)$-ésima, el simulador H_{k+1} manda los valores comprometidos a partir del testigo de la solución w y en la ronda k-ésima el simulador H_k crea el conjunto solución con la ecuación de la recta l_y en el índice x y $k-1$ índices 0. Entonces, D puede distinguir los valores comprometidos con probabilidad mayor a $\frac{1}{|x|^{c+1}}$.

Sin embargo, esto contradice la propiedad de ocultamiento de valores múltiples de los esquemas de compromiso 2.2.1, la cual asegura que existe alguna función despreciable $\epsilon_1()$ que acota por arriba la probabilidad de distinguir dos distribuciones que provienen de un número polinomial de valores ocultos mediante esquemas de compromiso. □

Las distribuciones $\{\tau_3\}$ y $\{\tau_4\}$ son computacionalmente indistinguibles ya que solo difieren en la elección de la recta l_y y el punto $p_r \in P$ (mientras que S_3 los obtiene a partir del testigo de la solución w, S_4 elije al momento una recta l_y que pase por un punto $p_r \in P$).

Por el lema híbrido 2.1.3 se sabe que la indistinguibilidad computacional es transitiva, por ello se puede afirmar que para todo B^* y para toda $x \in L$ las distribuciones $\{\tau_0\} \approx_c \{\tau_4\}$ y, por lo tanto, $\{\langle A, B^* \rangle(x)\} \approx_c \{S^*(x)\}$. ■.

6.3.3. Análisis de la complejidad del protocolo de cobertura de puntos

El sistema de pruebas $\langle A, B \rangle$ tiene como entrada común el conjunto de puntos P y el entero positivo k.

El protocolo se repite $2n$ veces.

1. **Probador** A: Aplica una transformación (que incluye traslación, rotación y multiplicación por un escalar) aleatoria ϕ al plano. Después aplica una permutación aleatoria π_P y π_K a

los conjuntos P y K. A cada elemento permutado le aplica un esquema de compromiso y se compromete con los conjuntos $P3$ y $K3$.

Se envían una matriz de $n \times 2$ con los esquemas de compromiso de los puntos y un vector con los esquemas de compromiso de los índices del conjunto solución. El tiempo para crear las transformaciones, las permutaciones y los esquemas de compromiso es de $O(n) + O(k)$. El espacio requerido para enviar los vectores es $O(n) + O(k)$.

2. **Verificador** B: Elige un $bit \in_R \{0, 1\}$ y un índice $index$ con $0 \leq index < n$ y se los envía a A.

   El tiempo para elegir bit e $index$ es $O(1)$. El espacio requerido para enviar los valores es $O(1)$.

3. **Probador** A: Recibe el par $bit, index$ y envía ϕ, π_P y las llaves ck_P de los esquemas de compromiso del conjunto $P3$ si $bit = 0$; o envía el índice x, la llave $ck_K[x]$ para abrir un valor $s_x \in K3$, y la llave $ck_[index]$ para abrir el punto $p_{index} \in P3$.

   El tiempo que requiere A para enviar lo solicitado a B es $O(n)$. El espacio requerido para contestar al reto es $O(n)$.

4. **Verificador** B: verifica que $P3$ se generó a partir del conjunto original P, si $bit = 0$; o abre el valor $s_x = l_y$ y el punto $p_{index} = (p_x, p_y)$ y verifica que l_y es incidente en p_{index}, si $bit \neq 0$.

   El tiempo para verificar el reto es $O(n)$. El espacio que se requiere para verificar el reto es $O(n)$.

Debido a que el protocolo se repite $2n$ veces, el tiempo de ejecución del protocolo es $O(n^2) + O(n * k)$ y el espacio utilizado en la comunicación es $O(n^2) + O(n * k)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(n^2) + O(n \cdot k)$$

Sin embargo, la expresión anterior se puede simplificar. Debido a que el número de puntos k es, a lo más, n, entonces, $k \leq n$. Por lo tanto, la complejidad espacial y temporal del protocolo es:

$$O(n^2).$$

Capítulo 7

Marco de trabajo generalizado

A partir de la redacción de los protocolos de conocimiento cero propuestos se pueden observar algunas coincidencias: el enfoque para abordar los problemas y el análisis para cumplir con las propiedades de completitud, solidez y conocimiento cero. Derivado de estas coincidencias encontradas se generó un marco de trabajo generalizado para abordar y generar protocolos de conocimiento cero para problemas de incidencia.

7.1. Pasos del marco de trabajo

El marco de trabajo propuesto para generar un protocolo de conocimiento cero se puede aplicar a problemas de incidencia, para ello se deben seguir los siguientes pasos:

1. **Modelar el problema como una gráfica**. Los problemas de incidencia o de cobertura se pueden modelar en forma de gráfica, donde los elementos o conjuntos serán los nodos y las incidencias o coberturas serán las aristas. El testigo de la solución debe modificarse también para demostrar la solución en la gráfica generada.

2. **Analizar los retos**. El objetivo de la prueba es demostrar que se tiene la solución de un problema sin dar información de la solución. Para ello se debe ofuscar la solución, transformando la entrada original, comprometiendo la entrada transformada y trabajando con la entrada transformada (la granularidad del ofuscamiento y de los esquemas de compromiso se propone a nivel de vértice y arista). El número de retos está en función del problema que se quiere abordar. Si el número de retos es r, entonces se selecciona cada uno de ellos con probabilidad uniforme $\frac{1}{r}$.

3. **Proponer el protocolo**. El protocolo debe garantizar que el probador A pueda demostrarle al verificador B que tiene la solución del problema sin dar información de la solución. El protocolo propuesto debe abordar el problema modelado como gráfica y debe satisfacer los retos identificados.

   Utilizando este marco de trabajo ningún protocolo puede ser de una sola ronda, ya que eso implicaría que el simulador debe tener un certificado completo de la solución para cada problema, sin embargo, el simulador genera un certificado parcial en cada ejecución porque es una máquina de Turing PTP.

7.2. Caso de estudio para el marco de trabajo propuesto

A continuación, se aplicará el marco de trabajo propuesto 7.1 para el problema de Ciclo hamiltoniano no dirigido (*Undirected hamiltonian cycle*). Un ciclo hamiltoniano no dirigido es un ciclo que visita cada nodo de la gráfica no dirigida una sola vez y donde el primero y el último nodo son el mismo.

Como entrada se tiene la gráfica $G = \{V, E\}$, donde $n = |V|$ y $m = |E|$.

Es importante aclarar que para este problema hay un protocolo de conocimiento cero bien conocido, basado en dos retos, el cual es de una sola ronda. El protocolo sigue los siguientes pasos:

1. El probador A crea una gráfica H, isomorfa a la original, y se compromete con ella.
2. El verificador B lanza un reto binario.
   - En el primer escenario B pide que se compruebe que H es isomorfa a G.
   - En el segundo escenario B pide que se muestre que H tiene un ciclo hamiltoniano.

Sin embargo, esta solución no se puede aplicar en el marco de trabajo propuesto, debido al argumento del simulador S^*. El simulador S^* es una máquina de Turing PTP que debe calcular el certificado parcial de la solución en tiempo de ejecución, sin embargo, su poder computacional está limitado polinomialmente, por lo que no podría responder el segundo reto del verificador B (a menos que $P = NP$). Por lo anterior y como ya se había mencionado, los protocolos que se generen siguiendo el marco de trabajo propuesto 7.1 no pueden ser de una sola ronda.

1. **Modelar el problema como una gráfica**. En este caso, la entrada del problema es en sí una gráfica G, la cual se representa por su lista de aristas E. A tiene como testigo de la solución al arreglo de vértices $C = \{c_0, \ldots, c_{n-1}\}$ que corresponde al ciclo hamiltoniano, donde dos elementos consecutivos (circularmente) son adyacentes. La información ofuscada y comprometida en cada ronda será la lista de aristas E y el vector solución C.
2. **Analizar los retos**. Para este problema se tienen 2 retos posibles. El *bit* de reto se elige con probabilidad uniforme. El análisis de los retos es el siguiente:

   a) Verificar que la gráfica original no fue cambiada.

   b) Verificar (parcialmente) que se tiene la solución del problema.

3. **Proponer el protocolo**.

   El protocolo consiste de $2n$ rondas. En cada ronda se realiza lo siguiente:

   El probador A renombra los vértices V (aplica una permutación π_V), renombra y revuelve las aristas E (aplica π_V y una permutación π_E) y renombra y recorre circularmente los vértices de C (aplica π_V y un recorrido $shift_C$); al final se compromete con la lista de aristas E' y con el arreglo C'.

   El verificador B responde con un reto. En el primer escenario, B solicita las permutaciones π_V, π_E y las llaves ck_{E_i}, lo anterior para validar que la gráfica comprometida es isomorfa a

la gráfica original. En el segundo escenario, B elige un nodo del espacio de índices comprometidos, digamos v_i, y A debe abrir el ciclo comprometido C' y las aristas $e_{a,i}, e_{i,b} \in E'$, lo anterior para validar que el nodo seleccionado está en el ciclo comprometido C' y tiene el mismo sucesor y el mismo antecesor en la gráfica comprometida E' y en el ciclo C'.

A y B tienen como entrada común la gráfica G. Los siguientes 4 pasos se realizan $2n$ veces:

a) El probador A realiza lo siguiente:

1) Genera una gráfica isomorfa a E, para ello aplica una permutación π_V a los nodos de E y de C, y recorre circularmente los nodos de C:

1: **for** $0 \leq i < m$ **do**
2: $\quad E1[i] = \pi_V[E[i][0]], \pi_V[E[i][1]]$
3: **end for**
4: **for** $0 \leq i < n$ **do**
5: $\quad C1[i] = \pi_V[C[i]]$
6: **end for**
7: **for** $0 \leq i < n$ **do**
8: $\quad C2[(i + shift_C)\,\%n] = C1[i]$
9: **end for**

2) Revuelve las aristas de E, aplicando una permutación π_E, respectivamente:

1: **for** $0 \leq i < m$ **do**
2: $\quad E2[\pi_E[i]] = E1[i]$
3: **end for**

3) Oculta los valores de $E2$ y de $C2$:

1: $x = 256$
2: **for** $0 \leq i < m$ **do**
3: $\quad ck_E[i] = SETUP(\{0,1\}^x)$
4: $\quad E2[i] = COMMIT(E1[i], ck_E[i])$
5: **end for**
6: **for** $0 \leq i < n$ **do**
7: $\quad ck_C[i] = SETUP(\{0,1\}^x)$
8: $\quad C2[i] = COMMIT(C1[i], ck_C[i])$
9: **end for**

A envía $E2$ y $C2$ al verificador B.

b) El verificador B responde al probador A con un reto aleatorio $bit \in_R \{0,1\}$ y con un vértice aleatorio $v_i \in_R V$, con los siguientes casos:

1) Si $bit = 0$, B solicita las permutaciones realizadas para obtener $E2$, con esto valida que la gráfica original no fue cambiada.
2) Si $bit \neq 0$, B solicita que se abra el arreglo $C2$ (con esto valida que existe la solución del problema) y solicita que se demuestre que el nodo v_i de $E2$ corresponde con un nodo de $C2$ (con esto valida que existe un ciclo hamiltoniano y que el ciclo contiene al vértice v_i en la gráfica comprometida).

c) El probador A recibe el par bit, v_i y realiza lo siguiente:

1) Si $bit = 0$, envía π_V, π_E y las llaves ck_E.
2) Si $bit \neq 0$, envía las llaves ck_C, el índice e_1, la llave $ck_E[e_1]$, el índice e_2 y la llave $ck_E[e_2]$.

d) El verificador B realiza los siguiente:

1) Si $bit = 0$, verifica que la gráfica comprometida $E2$ sea isomorfa a la gráfica original.
2) Si $bit \neq 0$, verifica que los nodos de $C2$ forman un ciclo hamiltoniano y que el vértice v_i es incidente a las aristas $e_1 = (a, i)$ y $e_2 = (i, b)$ tanto en $C2$ como en $E2$.

Si sí, acepta y continúa, si no, rechaza y termina.

Una vez propuesto el protocolo, hay que demostrar que el sistema de pruebas es de conocimiento cero, para ello debe cumplir con las propiedades de completitud (*completeness*), solidez (*soundness*) y conocimiento cero (*zero knowledge*). El análisis de estas propiedades es similar a los realizados en el presente trabajo, por lo cuál solo se comentará en prosa, confirmando así que el marco de trabajo propuesto es replicable a problemas de incidencia.

La propiedad de completitud supone que la cadena $x \in L$ y que el probador A es honesto, por tanto, tiene la solución del problema. Entonces, el probador A es capaz de responder a cualquiera de los dos retos con probabilidad 1.

La propiedad de solidez supone que la cadena $x \notin L$ y que, por lo mismo, no existe solución del problema, entonces A^* intentará engañar al verificador. En este caso, existen dos posibilidades: que A^* no cambie la gráfica original (con probabilidad p_i) o que A^* cambie la gráfica original por alguna de la cual sí tenga solución (con probabilidad $1 - p_i$). Para ambos casos se debe analizar la probabilidad de éxito cuando B elije cada uno de los retos con probabilidad uniforme. Por lo tanto, la probabilidad de engaño del protocolo después de $2n$ rondas es $\leq (1 - \frac{1}{2n})^{2n}$, lo que por [3] se sabe que es $\leq \frac{1}{e}$.

La propiedad de conocimiento cero supone que la cadena $x \in L$ y que se tiene un simulador S^* que puede ejecutar a un verificador B^*, ambos de tiempo polinomial esperado. Por tanto, para cumplir con conocimiento cero se debe proponer un algoritmo que pueda ejecutar S^* (es decir, que sea de tiempo polinomial esperado) cuya salida sea computacionalmente indistinguible de la del sistema de pruebas $\langle A, B^* \rangle$.

El simulador no tiene un testigo de la solución, pero sí debe simular que lo tiene, para ello crea un arreglo C con todos los nodos V y elige uno de los nodos (digamos n_k) que contenga dos aristas incidentes $e_{a,k}$, $e_{k,b}$, de tal manera que S^* va a esperar hasta que B^* elija el mismo nodo n_k para mostrar los nodos del arreglo C y las aristas $e_{a,k}$, $e_{k,b}$ incidentes en n_k.

El tiempo esperado para que la respuesta que eligieron S^* y B^* coincida es polinomial en el tamaño de V, por lo tanto, el algoritmo termina en tiempo polinomial. Para demostrar que las distrubuciones de S^* y de $\langle A, B^* \rangle$ son computacionalmente indistinguibles se deben proponer un conjunto de simuladores, siendo el primer simulador idéntico al sistema de pruebas $\langle A, B^* \rangle$ y el último simulador idéntico a S^*.

a) El primer simulador va a hacer uso del testigo de la solución w y se va a comprometer con la solución como lo haría $\langle A, B^* \rangle$.

b) El segundo simulador va a hacer uso de la solución w, va a seleccionar un nodo n_i y va a descubrir la solución en cada ronda hasta que B^* seleccione el mismo nodo n_i.

c) El tercer simulador va a hacer uso de la solución w, va a seleccionar un nodo n_i, va a generar un arreglo solución C con n_i, su antecesor y su sucesor y el resto de valores nulos y va a descubrir la solución en cada ronda hasta que B^* seleccione el mismo nodo n_i.

d) El cuarto simulador va a seleccionar un nodo n_i, va a generar un arreglo solución C con n_i, su antecesor y su sucesor y el resto de valores nulos y va a descubrir la solución en cada ronda hasta que B^* seleccione el mismo nodo n_i.

Al final, si los simuladores contiguos son computacionalmente indistinguibles, utilizando el lema híbrido se puede aseverar que el primero y el último son computacionalmente indistinguibles.

El protocolo generado tiene una complejidad temporal y espacial de $O(|E|) + O(n)$ por ronda. Debido a que el protocolo se repite $2n$ veces, el tiempo total de ejecución del protocolo es $O(n \cdot |E|) + O(n^2)$ y el espacio utilizado en la comunicación es $O(n \cdot |E|) + O(n^2)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(n \cdot |E|)$$

En conclusión y como se puede observar, siguiendo los pasos descritos en el marco de trabajo 7.1 se puede generar un protocolo de conocimiento cero *ad hoc* para diversos problemas de cobertura.

7.3. 21 problemas de Karp

Los 21 problemas de Karp son un conjunto de problemas computacionales NP-Completos de diferentes tipos (gráficas, enteros, arreglos de enteros, conjuntos, fórmulas booleanas, etc.). En el artículo [17] se muestra, a partir del teorema de Cook-Levin, un conjunto de reducciones desde el problema de SAT hacia problemas de combinatoria y de teoría de gráficas.

En la figura 7.1 se muestran los 21 problemas que analizó Richard Karp y las direcciones de las reducciones que siguió de forma descendente. En la misma figura 7.1, en azul, se encuentran los problemas a los que, según nuestro análisis, sí se les puede aplicar el marco de trabajo propuesto.

A continuación, se propondrá en prosa el protocolo *ad hoc* para cada problema en azul de la figura 7.1 siguiendo el marco de trabajo propuesto. Es importante resaltar que la mayoría de los protocolos propuestos en el presente trabajo son parte de los 21 problemas de Karp, por lo que se generó el protocolo *ad hoc* con todo detalle para estos problemas y así se mencionará en el siguiente análisis.

En los siguientes protocolos propuestos en prosa se omite la verificación de las propiedades de completitud, solidez y conocimiento cero, sin embargo, un lector interesado puede de forma relativamente simple adaptar la verificación de los capítulos anteriores a estos problemas.

Dentro de los protocolos que se propondrán a continuación hay dos constantes al momento de tratar la información: se ofusca y se compromete. Para ofuscar la información se hace uso de transformaciones a través de permutaciones aleatorias π. Para comprometer la información se hace uso de un esquema de compromiso para cada elemento de información utilizando llaves aleatorias ck.

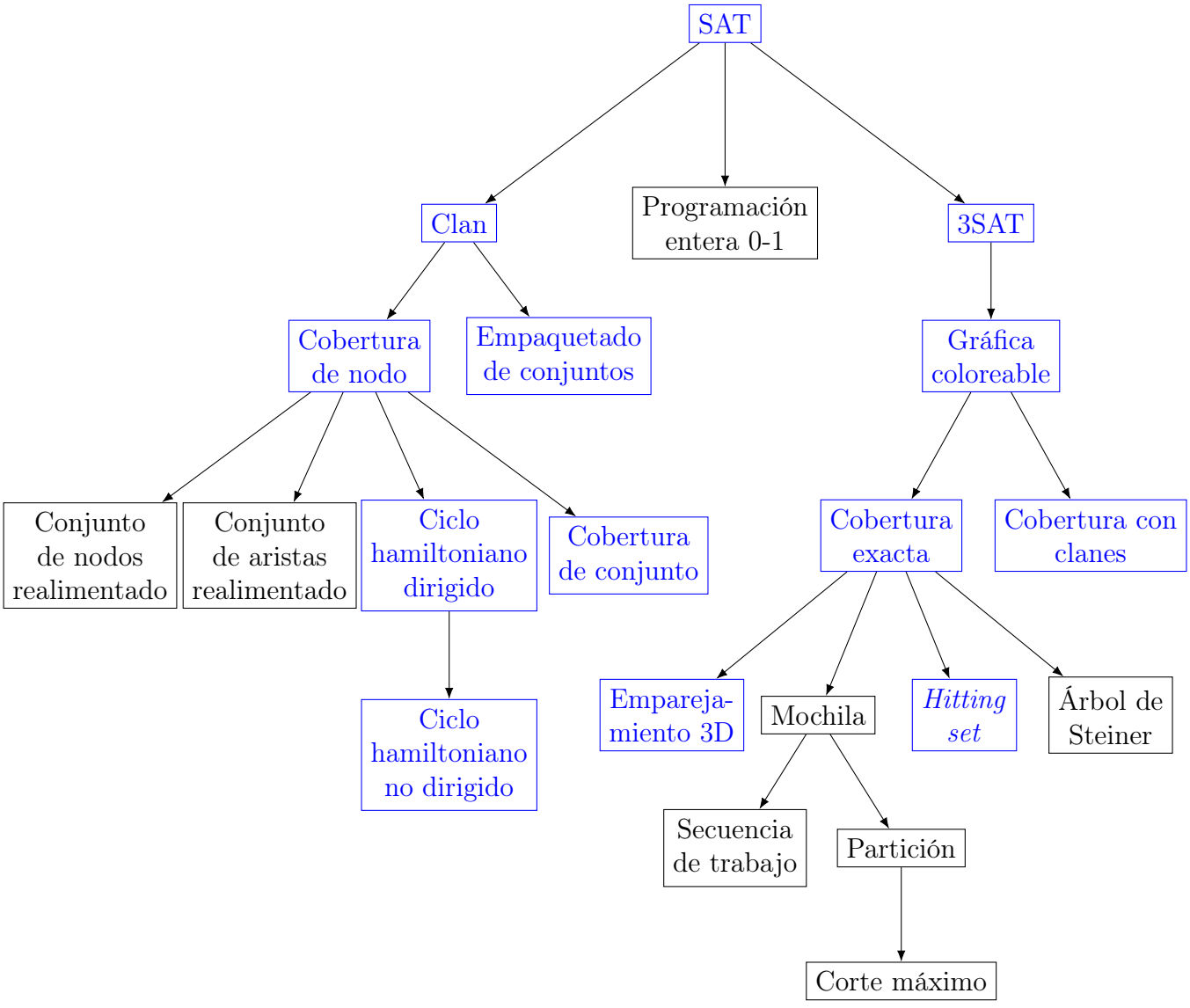

Figura 7.1: Diagrama de los 21 problemas analizados por Karp

1. **SAT**

   - **Problema**. Dada una fórmula booleana ϕ formada por conjunciones de m cláusulas $\{c_0 \wedge \ldots \wedge c_{m-1}\}$ donde cada cláusula c_i está formada por disyunciones de $2n$ literales $U = \{u_0, \bar{u_0}, \ldots, u_{n-1}, \bar{u_{n-1}}\}$, ¿existe una asignación que satisface a ϕ?
   - **Aplicación del marco de trabajo**.
     La idea de este protocolo es la misma que la desarrollada en 3SAT 4.3.1.

     a) Sean el conjunto $U = \{u_0, \bar{u_0}, \ldots, u_{n-1}, \bar{u_{n-1}}\}$ de literales y el conjunto $C = \{c_0, \ldots, c_{m-1}\}$ de cláusulas. A genera la lista de aristas E entre U y C con $a_{i,j} \in E \iff u_i \in c_j$. A tiene al arreglo S como testigo de la solución, el cual contiene las asignaciones a las variables que hacen que ϕ se satisfaga. La información ofuscada y comprometida en cada ronda será la lista de aristas E y el testigo S.

     b) Los retos posibles para este problema son dos:
     - Verificar que los conjuntos originales no fueron cambiados.
     - Verificar (parcialmente) que se tiene la solución del problema.

     c) El protocolo consiste de $2m$ rondas. En cada ronda el sistema de pruebas interactivo realiza lo siguiente:
     El probador A renombra las literales (aplica una permutación π_U), intercambia de forma aleatoria una variable por su negación (aplica un intercambio $flip_i$), renombra y revuelve las cláusulas (aplica π_U y una permutación π_C) y renombra, intercambia y revuelve las asignaciones S (aplica la permutación π_U, el intercambio $flip_i$ y una permutación π_S), generando así la lista de aristas isomorfa E' y el testigo isomorfo S'; al final se compromete con la lista de aristas E' y con el arreglo S'.
     El verificador B responde con un reto. En el primer escenario, B solicita las permutaciones π_U, π_C y las llaves ck_{E_i} del esquema de compromiso, esto para validar que el conjunto comprometido es equivalente al original. En el segundo escenario, B elije una cláusula comprometida c_j y solicita que se abra la arista $e_{i,j}$ (incidente en v_i y en c_j) y que se muestre la asignación $s_i \in S'$ en el vector solución tal que $v_i = s_i$ satisfaga la cláusula.

     Cada ronda del protocolo generado tiene una complejidad temporal y espacial de $O(|E|)$. Debido a que el protocolo se repite $2m$ veces, el tiempo total de ejecución del protocolo es $O(m \cdot |E|)$ y el espacio utilizado en la comunicación es $O(m \cdot |E|)$. Entonces, la complejidad espacial y temporal del protocolo es:

     $$O(m \cdot |E|).$$

2. **Clan**

   - **Problema**. Dada una gráfica G y un entero positivo k, ¿G tiene un conjunto de k nodos mutuamente adyacentes (una subgráfica completa)?
   - **Aplicación del marco de trabajo**.

     a) Sean una gráfica $G = (V, E)$ y un entero positivo k, G se representa como una lista de aristas E. A tiene como testigo de la solución al arreglo C de tamaño k, el cual contiene los nodos que forman el clique. La información ofuscada y comprometida en cada ronda será la lista de aristas E y el testigo C.

b) Los retos posibles para este problema son dos:
- Verificar que la gráfica original no fue cambiada.
- Verificar (parcialmente) que se tiene la solución del problema. Para validar (parcialmente) que los k vértices del testigo C son adyacentes entre sí en cada ronda, el verificador seleccionará uno de esos vértices y el probador debe demostrarle que los otros $k-1$ vértices del testigo son adyacentes a él. Claramente, la probabilidad de fallar al reto si es que no existe la solución es, al menos, $\frac{1}{k}$.

c) El protocolo consiste de $2k$ rondas. En cada ronda el sistema de pruebas interactivo realiza lo siguiente:
El probador A renombra V (aplica una permutación π_V), renombra y revuelve las aristas de E (aplica π_V y una permutación π_E), generando así E' y renombra y revuelve las entradas de C (aplicando π_V y una permutación π_C), generando así C'; al final se compromete con la lista de aristas E' y el arreglo C'.
El verificador B responde con un reto. En el primer escenario, B solicita las permutaciones π_V y π_E y las llaves ck_{E_i} del esquema de compromiso, lo anterior para validar que la gráfica comprometida es isomorfa a la gráfica original. En el segundo escenario, B elige un índice i de C' y solicita que se abra el arreglo C' y $k-1$ aristas de E' que conectan todos los vértices de C' distintos de c_i con c_i, lo anterior para validar (parcialmente) que el probador A posee la solución al problema y la solución corresponde a la gráfica comprometida.

Cada ronda del protocolo generado tiene una complejidad temporal y espacial de $O(|E|) + O(k)$. Debido a que el protocolo se repite $2k$ veces, el tiempo total de ejecución del protocolo es $O(k \cdot |E|) + O(k^2)$ y el espacio utilizado en la comunicación es $O(k \cdot |E|) + O(k^2)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(k \cdot |E|) + O(k^2).$$

3. **Empaquetado de conjuntos**

- **Problema**: Dada una familia de conjuntos $\{S_j\}$ y un entero positivo k, ¿existen k conjuntos de $\{S_j\}$ que sean mutuamente disjuntos?
- **Aplicación del marco de trabajo**. Sean una familia de conjuntos S, un conjunto de elementos U y un entero positivo k, el problema se representa como una gráfica de compatibilidad (tal que sus vértices son los elementos de S y dos de ellos son adyacentes si y sólo si son disjuntos) la cual está representada por su lista de aristas H. De tal manera que todo clique en H corresponde con un empaquetado en la instancia original. A tiene como testigo de la solución al arreglo C de vértices de H. Con esta representación A y B ejecutan el protocolo de Clan sobre la gráfica H y el entero k.

  El protocolo generado tiene la misma complejidad que el protocolo de Clan. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(k \cdot |E|) + O(k^2).$$

4. **Cobertura por nodo**

- **Problema**. Dada una gráfica $G = (V, E)$ y un entero positivo k, ¿existe un conjunto $R \subseteq V$ de tamaño a lo más k ($|R| \leq k$) tal que cada arista $e_i \in E$ sea incidente en un vértice de R?
- **Aplicación del marco de trabajo**.
  a) Sean una gráfica $G = (V, E)$ y un entero positivo k, G se representa por su lista de aristas E. A tiene como testigo de la solución al arreglo R que contiene k vértices de V. La información ofuscada y comprometida en cada ronda será la lista de aristas E y el testigo R.
  b) Los retos posibles para este problema son dos:
    - Verificar que la gráfica original no fue cambiada.
    - Verificar (parcialmente) que se tiene la solución del problema. Para validar (parcialmente) que toda arista $e_i \in E$ es incidente a alguno de los k vértices del testigo R, el verificador seleccionará una arista de E y el probador debe demostrarle que alguno de los nodos de la arista está en el conjunto solución R. Claramente, la probabilidad de fallar al reto si es que no existe la solución es, al menos, $\frac{1}{|E|}$.

  c) El protocolo consiste de $2|E|$ rondas. En cada ronda el sistema de pruebas interactivo realiza lo siguiente:
  El probador A renombra los vértices V (aplica una permutación π_V), renombra y revuelve las aristas E (aplica π_V y una permutación π_E) y renombra y revuelve los vértices de R (aplica π_V y una permutación π_R); al final se compromete con la lista de aristas E' y con el arreglo R'.
  El verificador B responde con un reto. En el primer escenario, B solicita las permutaciones π_V y π_E y las llaves ck_{E_i} del esquema de compromiso, lo anterior para validar que la gráfica comprometida es isomorfa a la original. En el segundo escenario, B elige una arista, digamos e_x, del conjunto de aristas comprometidas y A debe abrir la arista $e_x = (i, j)$ y un nodo $n_y \in R'$ de los nodos comprometidos, tal que $n_y = i$ o $n_y = j$, lo anterior para validar que la arista es incidente a uno de los nodos solución comprometidos.

  Cada ronda del protocolo generado tiene una complejidad temporal y espacial de $O(|E|)$. Debido a que el protocolo se repite $2|E|$ veces, el tiempo total de ejecución del protocolo es $O(|E|^2)$ y el espacio utilizado en la comunicación es $O(|E|^2)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(|E|^2).$$

5. **Ciclo hamiltoniano dirigido**

- **Problema**. Dada una digráfica $D = (V, E)$, ¿existe un ciclo dirigido en D que contenga cada nodo de V una sola vez?
- **Aplicación del marco de trabajo**.
  La idea de este protocolo es la misma que en Ciclo hamiltoniano no dirigido 7.2.
  a) Sea $D = (V, E)$ una digráfica, D se representa por su lista de flechas E. A tiene como testigo de la solución al arreglo de vértices C que corresponde al ciclo Hamiltoniano

dirigido (dos elementos consecutivos circularmente son adyacentes). La información ofuscada y comprometida en cada ronda será la lista de flechas E y el vector solución C.

b) Los retos posibles para el problema son dos:
- Verificar que la digráfica original no fue cambiada.
- Verificar (parcialmente) que se tiene la solución del problema. Para validar (parcialmente) la solución, el verificador seleccionará un vértice de C y el probador debe demostrarle que los valores del arreglo C son un ciclo (no hay vértices repetidos) y que el vértice seleccionado junto con su predecesor y sucesor están en la gráfica E comprometida (para ello abre dos flechas de E). Claramente, la probabilidad de fallar al reto si es que no existe la solución es, al menos, $\frac{1}{|V|}$.

c) El protocolo consiste de $2n$ rondas (con $n = |V|$). En cada ronda el sistema de pruebas interactivo realiza lo siguiente:
El probador A renombra los vértices V (aplica una permutación π_V), renombra y revuelve las flechas E (aplica π_V y una permutación π_E) y renombra y recorre los vértices de C (aplica π_V y un recorrido $shift_C$); al final se compromete con la lista de flechas E' y el arreglo C'.
El verificador B responde con un reto. En el primer escenario, B solicita las permutaciones π_V, π_E y las llaves ck_{E_i}, lo anterior para validar que la digráfica comprometida es isomorfa a la original. En el segundo escenario, B elige un nodo del espacio de índices comprometidos, digamos n_i, y A debe abrir el ciclo comprometido C' y las flechas $e_{a,i}, e_{i,b} \in E'$, lo anterior para validar que el nodo seleccionado está en el ciclo comprometido y tiene el mismo sucesor y el mismo antecesor que en la digráfica comprometida E' y en el ciclo C'.

Cada ronda del protocolo generado tiene una complejidad temporal y espacial de $O(|E|) + O(n)$. Debido a que el protocolo se repite $2n$ veces, el tiempo total de ejecución del protocolo es $O(n \cdot |E|) + O(n^2)$ y el espacio utilizado en la comunicación es $O(n \cdot |E|) + O(n^2)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(n \cdot |E|)$$

6. **Gráfica coloreable**

- **Problema**: Dada una gráfica $G = (V, E)$ y un entero positivo k, ¿existe una k-coloración $\phi : N \to Z_k$, tal que para cualquier par de nodos adyacentes u y v el color $\phi(u) \neq \phi(v)$.
- **Aplicación del marco de trabajo**.
La idea de este protocolo es la misma que en gráfica 3-coloreable 3.1.1.

a) Sean $G = (V, E)$ y un entero positivo k. A tiene como testigo de la solución el arreglo C con la k-coloración de V. La información ofuscada y comprometida en cada ronda será el testigo C.

b) Los retos posibles para el problema son dos:
- Verificar que la gráfica original no fue cambiada.

- Verificar (parcialmente) que se tiene la solución del problema. Para validar (parcialmente) la solución, el verificador seleccionará una arista (dos vértices adyacentes) y el probador debe demostrarle que la coloración asignada a cada uno de esos dos vértices es distinta y válida (colores distintos entre $[1, k]$). Claramente, la probabilidad de fallar al reto si es que no existe la solución es, al menos, $\frac{1}{|E|}$.

c) El protocolo consiste de $2|E|$ rondas. En cada ronda el sistema de pruebas interactivo realiza lo siguiente:
El probador A revuelve los colores de C (aplica una permutación π_C) y se compromete con el arreglo de colores C'.
El verificador B elige una arista, digamos e_i, y A debe abrir la arista $e_i = (a, b)$ y la coloración c_a y c_b asignada, lo anterior para validar que los nodos en los que es incidente la arista seleccionada tienen colores distintos entre $[1, k]$.

Cada ronda del protocolo generado tiene una complejidad temporal y espacial de $O(n)$. Debido a que el protocolo se repite $2|E|$ veces, el tiempo total de ejecución del protocolo es $O(n \cdot |E|)$ y el espacio utilizado en la comunicación es $O(n \cdot |E|)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(n \cdot |E|)$$

7. **Cobertura con clanes**

- **Problema**: Dada una gráfica $G = (V, E)$ y un entero positivo k, ¿existen a lo más k cliques cuya unión cubra todos los vértices de V?
- **Aplicación del marco de trabajo**.

a) Sean $G = (V, E)$ y un entero positivo k, con $n = |V|$ y $m = |E|$. A posee como testigo de la solución a un arreglo de arreglos M (M está formado por un arreglo de tamaño k y cada elemento del arreglo tiene a su vez un arreglo de tamaño $c \leq n$ que contiene los nodos que forman el clan). Sin embargo, si se compromete así la solución, puede haber fuga de información, ya que el tamaño de los clanes en la solución es variable y dar a conocer su tamaño daría pistas sobre la identidad de los vértices que conforman esos clanes.
Para evitar esta fuga de información se transformará la gráfica G en una gráfica $H = (V', E')$, agregando un clan F de tamaño $n - 1$ en el que cada nodo del clan F se une con cada nodo de la gráfica G de tal forma que hay $z \leq k$ clanes en G cuya unión es $V(G)$ si y sólo si hay z clanes en H de tamaño n que cubren los vértices $V(G)$. Los vértices de H son entonces $V(G) \cup V(F)$ y las aristas son $E(G) \cup E(F) \cup \{e_{u,v} | u \in V(G), v \in V(F)\}$. Todos los clanes de M deben ser de tamaño n, por lo que se agregan los nodos necesarios del clan F, generando así la matriz C (C es una matriz de tamaño $k \times n$).
H se representa por su lista de aristas E. A posee como testigo de la solución a la matriz C con, a lo más, k cliques, todos de tamaño n. La información ofuscada y comprometida en cada ronda será la lista de aristas E, los vértices $O = V(G)$ y el testigo C.

b) Los retos posibles para el problema son dos:
- Verificar que la gráfica original no fue cambiada.
- Verificar (parcialmente) que se tiene la solución del problema. La verificación parcial consiste en comprobar que algún vértice seleccionado por el verificador es adyacente a todos los vértices del supuesto clan que lo cubre. Claramente, la probabilidad de fallar el reto si es que no existe la solución es, al menos, $\frac{1}{n}$.

c) El protocolo consiste de $2n$ rondas. En cada ronda el sistema de pruebas interactivo realiza lo siguiente:
El probador A renombra los vértices $V(H)$ (aplica una permutación π_V), renombra y revuelve los vértices O (aplica π_V y una permutación π_O), renombra y revuelve las aristas E (aplica π_V y una permutación π_E) y renombra y revuelve los vértices de C (aplica π_V y una permutación π_C); al final se compromete con la lista de aristas E', el arreglo de vértices O' y la matriz C'.
El verificador B responde con un reto. En el primer escenario, B solicita las permutaciones π_V, π_O, π_E y las llaves ck_{O_i} y ck_{E_i}, lo anterior para validar que la gráfica comprometida E' se generó a partir de G y F y que O contiene los nodos originales $V(G)$. En el segundo escenario, B elige uno de los n nodos de O', digamos o_i, y A debe abrir los vértices del vector $c_j \in C'$ con $o_i \in c_j$ y $n-1$ aristas de E' que conectan todos los vértices de $c_x \in c_i$ con o_i, lo anterior para validar que todos los vértices del supuesto clan son adyacentes a o_i.

Cada ronda del protocolo generado tiene una complejidad temporal y espacial de $O(|E|)$. Debido a que el protocolo se repite $2n$ veces, el tiempo total de ejecución del protocolo es $O(n \cdot |E|)$ y el espacio utilizado en la comunicación es $O(n \cdot |E|)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(n \cdot |E|)$$

8. ***Hitting set***[1]

- **Problema**: Dada una familia de subconjuntos $\{S_i\}$ del conjunto de elementos $U = \{e_0, e_1, \ldots, e_{n-1}\}$ y un entero positivo k, ¿existe un conjunto W de tamaño a lo más k tal que para cada i, $|W \bigcap S_i| = 1$?
- **Aplicación del marco de trabajo**.

a) Sea S una familia de subconjuntos s_j, con $j = 0, 1, \ldots, m-1$ y sea U un conjunto de elementos e_i, con $i = 0, 1, \ldots, n-1$, se genera una gráfica bipartita completa entre S y U, donde cada arista debe tener el formato (i, j, c) que indique que e_i y s_j tienen una arista de color c con $c = 0$ si $e_i \notin s_j$ y $c = 1$ si $e_i \in s_j$; $c = 0$ denota una arista roja y $c = 1$ denota una arista azul, es decir:
- Se agrega una arista azul entre e_i y s_j si y sólo si $e_i \in s_j$.
- Se agrega una arista roja entre e_i y s_j si y sólo si $e_i \notin s_j$.

La gráfica bipartita se representa como una lista de aristas E. A tiene como testigo de la solución al arreglo W de elementos de U. La información ofuscada y comprometida en cada ronda será la gráfica E y el testigo W.

[1]Este término se dejó en inglés porque pensamos que brinda una mejor idea que su traducción literal.

b) Los retos posibles para el problema son dos:
- Verificar que los conjuntos originales no fueron cambiados.
- Verificar (parcialmente) que se tiene la solución del problema. La verificación parcial consiste en comprobar que algún subconjunto seleccionado por el verificador es adyacente a un solo elemento de W. Claramente, la probabilidad de fallar el reto si es que no existe la solución es, al menos, $\frac{1}{m}$.

c) El protocolo consiste de $2m$ rondas. En cada ronda el sistema de pruebas interactivo realiza lo siguiente:
El probador A renombra los subconjuntos de S (aplica una permutación π_S), renombra los elementos (aplica una permutación π_U), renombra y revuelve los elementos de E (aplica π_S, π_U y una permutación π_E) y renombra y revuelve los elementos de W (aplica π_U y una permutación π_W); al final se compromete con la lista de aristas E' y el arreglo W'.
El verificador B responde con un reto. En el primer escenario, B solicita las permutaciones π_S, π_U, π_E y las llaves ck_{E_i}, lo anterior para validar que los conjuntos comprometidos son equivalentes a los originales. En el segundo escenario, B elige un subconjunto de S, digamos s_i, y A debe abrir todos los elementos de W' y $|W'|$ aristas de G que van desde s_i a todos los elementos de W' y donde una arista es azul y $|W'| - 1$ son aristas rojas, lo anterior para validar que solo un elemento es compartido entre s_i y W'.

Se puede observar que en este protocolo hay fuga de información: se puede saber la longitud de la solución W. Sin embargo, se puede ocultar la longitud de la solución utlizando vértices de Steiner, es decir, se puede extender el conjunto U con $k-1$ nodos artificiales (conectados con aristas rojas a los elementos de S) y, si $|W| < k$, agregar $k - |W|$ nodos de Steiner al vector W y trabajar con estos conjuntos extendidos en lugar de los originales, siguiendo el protocolo propuesto.

Cada ronda del protocolo generado tiene una complejidad temporal y espacial de $O(n \cdot m)$. Debido a que el protocolo se repite $2m$ veces, el tiempo total de ejecución del protocolo es $O(n \cdot m^2)$ y el espacio utilizado en la comunicación es $O(n \cdot m^2)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(n \cdot m^2)$$

9. **Cobertura exacta**

- **Problema**: Dada una familia de subconjuntos $\{S_i\}$ del conjunto de elementos $U = \{e_0, e_1, \ldots, e_{n-1}\}$, ¿existe una subfamilia $\{T_h\} \subseteq \{S_j\}$ tal que cualesquiera dos elementos de T son disjuntos par a par y $\bigcup T_h = U$?
- **Aplicación del marco de trabajo**: Sea S una familia de subconjuntos s_j, con $j = 0, 1, \ldots, m-1$ y U un conjunto de elementos $\{e_0, e_1, \ldots, e_{n-1}\}$, se genera una gráfica bipartita completa entre S y U de la siguiente manera:
  - Se agrega una arista azul entre e_i y s_j si y sólo si $e_i \in s_j$.
  - Se agrega una arista roja entre e_i y s_j si y sólo si $e_i \notin s_j$.

La gráfica bipartita se representa como una lista de aristas E. A tiene como testigo de la solución al arreglo T. La información ofuscada y comprometida en cada ronda será la gráfica E y el testigo T. Con esta representación de gráfica bipartita es fácil ver que Cobertura exacta es equivalente a *Hitting set*: en el caso de *Hitting set*, hay que probar la existencia de un subconjunto W de vértices de U, tal que para todo vértice v de S haya exactamente una arista azul (y el resto rojas) conectando a v con los vértices de W; en el caso de Cobertura exacta, hay que probar la existencia de un subconjunto T de vértices de S, de tal forma que para todo vértice v de U, haya exactamente una arista azul (y el resto rojas) conectando a v con los vértices de T. Por lo tanto, el protocolo es el mismo, simplemente invirtiendo los papeles de los conjuntos de vértices S y U.

Cada ronda del protocolo generado tiene una complejidad temporal y espacial de $O(n \cdot m)$. Debido a que el protocolo se repite $2n$ veces, el tiempo total de ejecución del protocolo es $O(n^2 \cdot m)$ y el espacio utilizado en la comunicación es $O(n^2 \cdot m)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(n^2 \cdot m)$$

10. **Emparejamiento 3D**

- **Problema**: Dados los conjuntos finitos X, Y y Z con $|X| = |Y| = |Z|$ y el conjunto $U \subseteq X \times Y \times Z$, el problema consiste en determinar si existe $W \subseteq U$ tal que $|W| = |X|$ y para cualquier par de tripletas $(x_1, y_1, z_1) \in W$ y $(x_2, y_2, z_2) \in W$ se tiene que $x_1 \neq x_2$ y $y_1 \neq y_2$ y $z_1 \neq z_2$.
- **Aplicación del marco de trabajo**: Dados los conjuntos finitos X, Y y Z y el conjunto $U \subseteq X \times Y \times Z$, se genera una gráfica bipartita G entre los conjuntos X, Y y Z y las tripletas de U (las tripletas tienen la forma $t_l = (x_i, y_j, z_k)$ con $x_i \in X$, $y_j \in Y$ y $z_k \in Z$) tal que existe una arista entre los vértices (x_i, t_l), (y_j, t_l) y (z_k, t_l), generando así una lista de aristas E de tamaño $3|U|$. A tiene como testigo de la solución al arreglo W, el cual contiene los índices de U que dan solución al problema.

  Por la construcción antes mencionada, una tripleta t contiene a un elemento e de $X \cup Y \cup Z$ si y sólo si el vértice de G asociado a t es adyacente al vértice de G asociado a e; por tanto, se puede afirmar lo siguiente:

  > El conjunto de n tripletas de W es un emparejamiento perfecto si y sólo si los n vértices asociados a esas tripletas son una cubierta exacta de los vértices asociados a $X \cup Y \cup Z$ en G.

  Con esta afirmación se puede proponer un protocolo *ad hoc* para *Emparejamiento 3D* utilizando el protocolo propuesto para el problema de *Cobertura exacta*. Sin embargo, también podríamos utlilizar un protocolo más simple, eliminando la propiedad *exacta* en la cobertura, con la siguiente observación:

  > En la gráfica bipartita G, todo conjunto de vértices de tamaño n del lado U que cubren todos los vértices del lado de $X \cup Y \cup Z$, los cubren de forma exacta.

  Supongamos, por contradicción, que la observación anterior no se cumple. Eso quiere decir que existe una cubierta (no exacta) de tamaño n, sea esa cubierta $C =$

$\{v_0, v_1, \ldots, v_{n-1}\}$. Entonces, existe al menos un vértice del lado de $X \cup Y \cup Z$ que es cubierto por, al menos, dos vértices de la cubierta. Lo que implica que el número de aristas incidentes a los n vértices de la cubierta C es, al menos, $3n + 1$. Pero esto no es posible ya que, por construcción, el grado de cada uno de los n vértices de C es exactamente 3.

Por lo anterior, un protocolo *ad-hoc* de conocimiento cero para el problema de *Emparejamiento 3D* es el siguiente: tanto el probador como el verificador calculan la gráfica bipartita asociada G, y después ejecutan los mismos pasos que en el protocolo de *Cobertura de conjunto*, sobre esta gráfica de trabajo G.

Cada ronda del protocolo generado tiene una complejidad temporal esperada y espacial de $O(|E|)$. Debido a que el protocolo se repite $2n$ veces, el tiempo total de ejecución del protocolo es $O(n \cdot |E|)$ y el espacio utilizado en la comunicación es $O(n \cdot |E|)$. Entonces, la complejidad espacial y temporal del protocolo es:

$$O(n \cdot |E|)$$

Como se mencionó previamente, la mayoría de los protocolos propuestos en este trabajo corresponden a buena parte de los 21 problemas de Karp, por lo tanto, el protocolo *ad hoc* para ellos ya se generó siguiendo el marco de trabajo propuesto. A continuación, se nombran los 3 protocolos desarrollados que sirven para tratar parte de los 21 problemas de Karp.

11. **Cobertura de conjunto**

    - **Problema**. Dada una familia finita de conjuntos finitos $\{S_j\}$ y un entero positivo k, ¿existe una subfamila $\{T_i\} \subseteq \{S_j\}$ de tamaño a lo más k ($|\{T_i\}| \leq k$) cuya unión sea ($\bigcup T_i = \bigcup S_j$)?
    - **Aplicación del marco de trabajo**. El protocolo *ad hoc* se generó en el presente trabajo en 5.3.1.

12. **Ciclo hamiltoniano no dirigido**

    - **Problema**. Dada una gráfica $G = (V, E)$, ¿existe un ciclo en G que contenga cada nodo de V una sola vez?
    - **Aplicación del marco de trabajo**. El protocolo *ad hoc* se generó en el presente trabajo en 7.2.

13. **3SAT**

    - **Problema**. Dado un conjunto de cláusulas $c_0, \ldots, c_{m-1}$, donde cada cláusula consiste de 3 literales del conjunto de literales $\{u_0, \ldots, u_{n-1}\} \cup \{\bar{u_0}, \ldots, \bar{u_{n-1}}\}$, ¿se puede satisfacer el conjunto $c_0, \ldots, c_{m-1}$?
    - **Aplicación del marco de trabajo**. El protocolo *ad hoc* se generó en el presente trabajo en 4.3.1.

Consideramos que el marco de trabajo propuesto no se puede aplicar a problemas que involucren pesos, debido a que para modelar pesos en las gráficas, pensamos que sería necesario o bien etiquetar con pesos a los vértices o a las aristas (y al momento de abrirlos como muestra parcial de la solución, esas etiquetas permitirían conocer sus identidades originales), o bien repetir un vértice o una arista tantas veces en la gráfica como el valor de su peso (y eso haría que la gráfica de trabajo tuviera un tamaño exponencial en el tamaño de la instancia original). Por esta razón, pensamos que, de los 21 problemas de Karp, a los siguientes 6 no se les puede aplicar el marco de trabajo:

1. **Programación entera 0-1**: Dada una matriz C de enteros y un vector d de enteros, ¿existe un vector x de $0-1$ tal que $Cx = d$?.
2. **Mochila**: Dado $(a_0, a_1, \ldots, a_{r-1}, b) \in Z^{n+1}$, ¿existe la solución a $\sum a_j x_j = b$?
3. **Árbol de Steiner**: Dada una gráfica $G = \{V, E\}$, un conjunto $R \subseteq V$, una función $w : E \to Z$ y un entero positivo k, ¿existe una subárbol en G de peso $\leq k$ que contenga los nodos de R?
4. **Secuencia de trabajo**: Dada una secuencia de n longitudes enteras de trabajos $L = [l_0, \ldots, l_{n-1}]$, una secuencia de n *deadlines* enteros $D = [d_0, \ldots, d_{n-1}]$, una secuencia de n penalizaciones enteras $P = [p_0, \ldots, p_{n-1}]$ (cada trabajo tiene asignada una penalización si no se termina en tiempo a lo más su *deadline*) y un entero k, ¿existe una permutación de los n trabajos, tal que, comenzando desde el instante $t = 0$ y procesando secuencialmente los trabajos en el orden de esa permutación, la suma de penalizaciones de los trabajos que no se terminaron en tiempo a lo más su *deadline*, es a lo más k?
5. **Partición**: Dado $(c_0, c_1, \ldots, c_{s-1}) \in Z^s$, ¿existe un conjunto $I \subseteq \{0, 1, \ldots, s-1\}$ tal que $\sum_{h \in I} c_h = \sum_{h \notin I} c_h$?
6. **Corte máximo**: Dada una gráfica $G = (V, E)$, una función $w : E \to Z$ y un entero positivo k, ¿existe un conjunto $S \subseteq V$ tal que $\sum_{\{u,v\} \in E, u \in S, v \notin S} w(\{u, v\}) \geq k$?

Además, existen un par de problemas que aparentemente requieren comprometer los ciclos de una gráfica pero los ciclos en una gráfica son exponenciales y estos se requieren para probar la solución del problema y, por lo tanto, no se puede tener una solución polinomial. Estos problemas son:

7. **Conjunto de nodos realimentado** Dada una digráfica $D = (V, E)$ y un entero positivo k, ¿existe un subconjunto $R \subseteq V$ de tamaño a lo más k tal que cada ciclo (dirigido) de D contiene un nodo en R?
8. **Conjunto de aristas realimentado** Dada una digráfica $D = (V, E)$ y un entero positivo k, ¿existe un subconjunto $S \subseteq E$ de tamaño a lo más k tal que cada ciclo (dirigido) de D contiene una arista en S?

En resumen, el marco de trabajo propuesto se pudo aplicar a 13 de los 21 problemas de Karp, para demostrarlo se generó el protocolo *ad hoc* para los problemas de SAT, Clan, Empaquetado de conjuntos, Cobertura por nodo, Ciclo hamiltoniano dirigido, Gráfica coloreable, Cobertura con clanes, *Hitting set*, Cobertura exacta, Emparejamiento 3D, Cobertura de conjunto, Ciclo hamiltoniano no dirigido y 3SAT. Además, también se aplicó el marco de trabajo al problema de Cobertura de puntos (como se muestra en el capítulo 6.3.1).

En la tabla 7.1 se muestra de forma resumida el análisis de complejidad (temporal y espacial) de los protocolos creados en el presente trabajo a partir del marco de trabajo propuesto, junto con el protocolo de *gráfica 3-coloreable*.

No.	**Problema**	**Complejidad temporal y espacial**
0	Gráfica 3-coloreable	$O(n \cdot \|E\|)$
1	SAT	$O(m \cdot \|E\|)$
2	Clan	$O(k \cdot \|E\|) + O(k^2)$
3	Empaquetado de conjuntos	$O(k \cdot \|E\|) + O(k^2)$
4	Cobertura por nodo	$O(\|E\|^2)$
5	Ciclo hamiltoniano dirigido	$O(n \cdot \|E\|)$
6	Gráfica coloreable	$O(n \cdot \|E\|)$
7	Cobertura con clanes	$O(n \cdot \|E\|)$
8	*Hitting set*	$O(n \cdot m^2)$
9	Cobertura exacta	$O(n^2 \cdot m)$
10	Emparejamiento 3D	$O(n \cdot \|E\|)$
11	Cobertura de conjunto	$O(n \cdot \|E\|)$
12	Ciclo hamiltoniano no dirigido	$O(n \cdot \|E\|)$
13	3SAT	$O(m^2)$
14	Cobertura de puntos	$O(n^2)$

Tabla 7.1: Tabla de complejidades de los protocolos analizados.

La tabla 7.1 muestra una eficiencia competitiva entre los protocolos generados en el presente trabajo en comparación con el protocolo de referencia *Gráfica 3-coloreable*, lo cual es un resultado alentador pensando en que estos protocolos se pueden ocupar de manera directa para las instancias originales sin necesidad de transformarlos a una instancia de *Gráfica 3-coloreable*.

Las pruebas de conocimiento cero obtenidas en este capítulo sustentan la motivación del trabajo. Por un lado, se identificaron y modelaron problemas de incidencia y por otro lado se pudo aplicar el marco de trabajo propuesto y generar las pruebas de conocimiento cero *ad hoc* para cada problema seleccionado.

Durante el diseño de las pruebas de conocimiento cero para los problemas seleccionados de los 21 problemas de Karp se encontraron los siguientes tipos:

- Problemas inéditos. Problemas para los cuales se generó la prueba de conocimiento cero

desde el inicio. Estos problemas fueron *Clan*, *Cobertura por nodo*, *Cobertura con clanes*, *Hitting set*, *Emparejamiento 3D*, *Cobertura de conjunto*, *Ciclo Hamiltoniano no dirigido* y *3SAT*

- Problemas similares. Problemas que se abordaron de forma similiar a otra prueba, pero sin necesidad de transformar la instancia original. Estos problemas fueron *SAT* similar a *3SAT*, *Ciclo Hamiltoniano dirigido* similar a *Ciclo Hamiltoniano no dirigido* y *Gráfica coloreable* similar a *Gráfica 3-coloreable*.

- Problemas transformados. Problemas cuya instancia original se tuvo que transformar a una instancia de otro problema. Estos problemas fueron *Empaquetado de conjuntos* a *Clan* y *Cobertura exacta* a *Hitting set*.

El análisis realizado para cada uno de los problemas permitió generar 8 pruebas de conocimiento cero inéditas, 3 pruebas similares y 2 transformaciones de instancias. Con las pruebas de conocimiento cero realizadas se pudo profundizar en el entendimiento y correcta aplicación del marco de trabajo propuesto.

Capítulo 8

Conclusiones y trabajos futuros

El objetivo general del trabajo era generar protocolos de conocimiento cero a la medida para problemas de incidencia partiendo de la siguiente hipótesis:

> **Hipótesis**: *Es posible generar protocolos de conocimiento cero* ad hoc *para problemas de incidencia concretos, que sean fáciles de entender en su construcción y factibles de implementar.*

Para comprobar la hipótesis se analizaron diversos problemas NP-Completos en busca de problemas de incidencia (problemas que se pudieran modelar como una gráfica), donde los elementos se modelaron como nodos y las relaciones como aristas incidentes en los nodos. Se intentó aplicar el marco de trabajo a la colección más conocida de problemas NP-Completos: los 21 de Karp. Para 14 se tuvo éxito y se generaron sus protocolos de incidencia concretos utilizando el marco de trabajo propuesto en 7.1 y para el resto de los problemas se presentó una discusión razonada de las razones por las cuales no se pudo aplicar el marco de trabajo, lo que ayudará a un lector a comprender si el marco de trabajo podrá ser o no aplicado a algún problema específico al que en un futuro le quiera diseñar un protocolo de conocimiento cero a la medida.

De manera general, la forma de construir un protocolo que no sea a la medida, es decir, utilizando el teorema existencial de Micali *et al.* [6], implica una reducción del lenguaje $L \in NP$ al lenguaje de SAT lo cual es altamente no trivial porque es justo la demostración del teorema de Cook-Levin. Sin embargo, el marco de trabajo propuesto permite generar protocolos de conocimiento cero para problemas de incidencia de manera relativamente intuitiva, esto es, una vez modelado el problema como una gráfica, si se siguen los pasos descritos en el marco de trabajo es factible generar e implementar un protocolo de conocimiento cero de manera relativamente fácil (además, se pueden tomar como referencia los protocolos generados con todo detalle en el presente trabajo).

Por lo tanto, el *marco de trabajo* propuesto permitió generar 14 pruebas de conocimiento cero *ad hoc* para 14 problemas de incidencia *NP-Completos*:

- Para los problemas de *3SAT*, *Cobertura de conjunto* y *Cobertura de puntos* se generó su protocolo de conocimiento cero con todo detalle, describiendo el protocolo de comunicación y demostrando las propiedades de completitud, solidez y conocimiento cero.
- Para el problema de *Ciclo hamiltoniano no dirigido* se generó el protocolo de conocimiento cero con un detalle general para ejemplificar el marco de trabajo propuesto, describiendo

el protocolo de comunicación y dando un esbozo de la demostración de las propiedades de completitud, solidez y conocimiento cero.

- Para los problemas de *SAT*, *Clan*, *Empaquetado de conjuntos*, *Cobertura por nodo*, *Ciclo hamiltoniano dirigido*, *Gráfica coloreable*, *Cobertura con clanes*, *Hitting set*, *Cobertura exacta* y *Emparejamiento 3D* se generó un esbozo del protocolo de conocimiento cero, describiendo en prosa el protocolo de comunicación.

Además, comparando la complejidad del protocolo de referencia *gráfica 3-coloreable* con la complejidad de los 14 protocolos realizados bajo el marco de trabajo propuesto (tabla 7.1) podemos afirmar que se generaron protocolos *ad hoc* cuya eficiencia es competitiva.

En la práctica tener protocolos *ad hoc* entendibles, implementables y eficientes es importante porque entonces estos nuevos protocolos desarrollados podrían utilizarse para problemas cotidianos sin disminuir la eficiencia con respecto, por ejemplo, al protocolo de referencia G3C. Además, el protocolo *ad hoc* de *3SAT* se puede ocupar para probar directamente que todo lenguaje $L \in NP$ tiene una prueba de conocimiento cero sin recurrir a la reducción de $3SAT$ a $G3C$.

Queda como trabajo futuro generar el protocolo de conocimiento cero con todo detalle para los problemas para los que solo se generó un esbozo del mismo. También, se podría ahondar en la generación de protocolos de conocimiento cero para problemas de incidencia fuera de los comprendidos entre los 21 problemas de Karp. Además, se podrían encontrar aplicaciones prácticas para los protocolos generados en el presente trabajo o para otros problemas NP cuyo protocolo de conocimiento cero se genere a partir del marco de trabajo propuesto.

Apéndice A

Clases de complejidad

Una clase de complejidad está definida por aquel conjunto de funciones que pueden ser computadas con ciertos recursos computacionales. Un sistema de pruebas interactivo permite la interacción entre dos máquinas de Turing, el probador A y el verificador B, estas máquinas de Turing pueden tener distintas capacidades computacionales (polinomiales, probabilísticas de tiempo esperado polinomial, con cómputo infinito, etc.). A continuación, se mencionan las clases de complejidad más importantes utilizadas en los protocolos generados.

Las siguientes definiciones de clases de complejidad fueron extraídas de [22].

A.1. Clases DTIME y P

La clase $DTIME$ está constituida por el conjunto de funciones booleanas (funciones que tienen como salida un bit) que son computables en un tiempo $c \cdot T(n)$, con $c > 0$ y $T : \mathbb{N} \to \mathbb{N}$.

La escencia de la clase P está en capturar la noción de los problemas de decisión que son factibles de resolver, es decir, aquellos problemas que se pueden resolver en tiempo eficiente.

Una definición aproximada de la clase de problemas de decisión que son resolubles en tiempo eficiente es:

$$P = \bigcup_{c \geq 1} \mathrm{DTIME}(n^c)$$

Entonces, la clase P está compuesta por todas aquellas funciones que pueden ser computadas en un tiempo $c \cdot n^k$, con $c, k > 0$.

A.2. Clase NP

La clase NP está formada por todos aquellos problemas cuya solución es verificable de forma eficiente, a partir de un certificado de la solución u de tamaño polinomial.

Sea una cadena $x \in L$ y un certificado de la solución u de tamaño polinomial $u \in \{0,1\}^{p(|x|)}$. $L \in NP$ si y sólo si una máquina de Turing de tiempo polinomial M acepta con x y u como entradas, es decir, $M(x, u) = 1$.

A.3. Clase NP-Completo

La clase NP-Completo está formada por todos aquellos problemas que son tan difíciles como cualquier problema en NP.

Para poder demostrar que un lenguaje $L \in NP$ es muy difícil (L no puede decidirse en tiempo polinomial, suponiendo que $P \neq NP$), se debe realizar una reducción en tiempo polinomial de un lenguaje que esté en NP-Completo hacia el lenguaje L. En [22] se demuestra que el problema de satisfacción proposicional (SAT por sus siglas en inglés) $\in$ NP-Completo y a partir de éste se han realizado varias reducciones a otros lenguajes. La reducción se denota como $\leq_p$, entonces, $SAT \leq_p L$ quiere decir que SAT se puede reducir en tiempo polinomial a L.

Se dice que el lenguaje B es **NP-duro** si $A \leq_p B$ para cualquier lenguaje $A \in NP$. Se dice que el lenguaje B es **NP-completo** si B es **NP-duro** y $B \in NP$.

A.4. Clase PP

La clase de complejidad PP está constituida por los problemas de decisión que pueden ser resueltos por una máquina de Turing probabilista en tiempo polinomial con una probabilidad de error $\leq \frac{1}{2}$ para todas las instancias.

A.5. Clase IP

La clase de complejidad IP está constituida por los sistemas de pruebas interactivos $\langle A, B\rangle$, donde el verificador B es una máquina de Turing probabilista de tiempo polinomial.

Para empezar, consideremos un *sistema de pruebas interactivo determinista* con las funciones $f, g : \{0,1\}^* \to \{0,1\}^*$. Las funciones deterministas f y g van a estar interactuando entre sí un número determinado de veces (dependiendo del protocolo que estén ejecutando), al número de interacciones que se susciten entre f y g se le llama ronda. Además, en cada interacción i las funciones reciben la cadena x y las secuencias de cadenas $a_1, \cdots, a_{i-1} \in \{0,1\}^*$ anteriores.

La interacción de la k-ésima ronda entre f y g con la entrada $x \in \{0,1\}^*$, denotado como $\langle f, g\rangle(x)$, están formadas por la secuencia de las cadenas $a_1, \cdots, a_{2k} \in \{0,1\}^*$, las cuales se definen como sigue:

$$\begin{aligned} a_1 &= f(x) \\ a_2 &= g(x, a_1) \\ &\cdots \\ a_{2i+1} &= f(x, a_1, \cdots, a_{2i}) \\ a_{2i+2} &= g(x, a_1, \cdots, a_{2i+1}) \end{aligned} \tag{A.1}$$

Entonces, la cadena a_{2i+1} que genera f está en función de la cadena x y la secuencia de las cadenas $a_1, \cdots, a_{2i} \in \{0,1\}^*$; la cadena a_{2i+2} que genera g está en función de la cadena x y la secuencia de las cadenas $a_1, \cdots, a_{2i+1} \in \{0,1\}^*$.

La salida de f (denotada como $out_f\langle f, g\rangle(x)$) al finalizar la interacción se define como $f(x, a_1, \cdots, a_k)$. La salida de g (denotada como $out_g\langle f, g\rangle(x)$) al finalizar la interacción se define como $g(x, a_1, \cdots, a_k)$.

Para aprovechar todo el potencial de la interacción se requiere que el verificador B sea de tiempo polinomial probabilístico. Para ello, hay que extender la noción de *sistema de pruebas interactivo* hacia funciones probabilísticas (solo para el verificador).

Para modelar la interacción entre f y g con f probabilista se agrega una entrada adicional r de m bits a la función f en A.1. La variable r es una cadena aleatoria que se elige con probabilidad uniforme y de tamaño m, lo que se denota como $r \in_R \{0,1\}^m$. La interacción de f queda como:

$$\begin{aligned} a_1 &= f(x, r) \\ a_3 &= f(x, r, a_1, a_2) \\ &\cdots \\ a_{2i+1} &= f(x, r, a_1, \cdots, a_{2i}) \end{aligned} \tag{A.2}$$

Con la cadena aleatoria $r \in \{0,1\}^m$, el contenido de las cintas de comunicación del sistema de pruebas interactivo $\langle f, g\rangle(x)$ es ahora una variable aleatoria. Del mismo modo, la salida $out_f\langle f, g\rangle(x)$ es también una variable aleatoria.

Se dice que un lenguaje $L \in IP$ si, dado un sistema de pruebas interactivo $\langle A, B\rangle$, existe una máquina de Turing B tal que dadas las entradas $x, r, a_1, \cdots, a_i$, B se ejecuta en tiempo polinomial en $|x|$ y:

- Si $x \in L$, existe un probador A tal que $Pr[out_B\langle A, B\rangle(x) = 1] \geq 2/3$.
- Si $x \notin L$, para cualquier probador A, se tiene que $Pr[out_B\langle A, B\rangle(x) = 1] \leq 1/3$.

Apéndice B

Probabilidad

Un sistema de pruebas interactivo debe cumplir con las propiedades de completitud y solidez con cierta probabilidad. Además, para cumplir con conocimiento cero computacional las distribuciones de salida deben cumplir con *indistinguibilidad computacional* con cierta probabilidad.

La Probabilidad es una rama de las matemáticas que define de forma numérica la probabilidad de que ocurra un evento o en particular de que una proposición sea verdadera. La probabilidad de un evento se encuentra entre los valores 0 y 1, donde 0 indica la imposibilidad de un evento y 1 la certeza de que ocurra ese evento.

B.1. Variables aleatorias

Un espacio de probabilidad finito es un conjunto finito $\Omega = \{x_1, \ldots, x_N\}$ junto con un conjunto de números $Pr_1, \ldots, Pr_N \in [0, 1]$ Un elemento aleatorio se selecciona de este espacio eligiendo un x_i con probabilidad Pr_i [22].

Una *variable aleatoria* es un mapeo desde un espacio de probabilidad a $\mathbb{R}$ [22]. Es decir, si Ω, como se describió anteriormente, es el conjunto de todas los posibles resultados de n lanzamientos de una moneda justa, entonces podemos denotar como X el número de monedas que dieron cara.

Los tipos de variables aleatorias dependen del conjunto de valores de donde éstas se toman, las cuales se clasifican en dos grandes grupos: discretas y continuas.

Una variable aleatoria discreta es aquella que puede tomar un número finito de valores o bien un número infinito de valores pero numerable. Una variable aleatoria continua es aquella que puede tomar todos los valores dentro de un conjunto infinito no numerable de valores. En el presente trabajo se hace uso únicamente del concepto de variable aleatoria discreta.

B.2. Función de probabilidad

Una variable aleatoria tiene asociada una distribución de probabilidad que establece los valores que puede tomar la variable aleatoria y las probabilidades con las que toma estos valores.

Una función de probabilidad discreta $f_X(x_i)$, con $x_i \in \mathbb{R}$ y $f_X(x_i) \in [0, 1]$, se define como:

$$f_X(x_i) = Pr_i[X = x_i]$$

donde Pr representa la probabilidad de que la variable aleatoria X tome el valor x_i.

La función de probabilidad de una variable aleatoria discreta cumple con las siguientes propiedades [12]:

- La probabilidad de que x pueda tomar un valor específico es $f_X(x_i)$, esto es:

$$Pr[X = x] = f_X(x_i)$$

- $f_X(x_i)$ es un valor no negativo para todo valor real x_i.
- La suma de probabilidades $f_X(x_i)$ de todos los posibles valores de x_i es 1:

$$\sum_{\forall x_i} f_X(x_i) = 1$$

Además, la probabilidad de que la variable aleatoria esté en un intervalo se obtiene a partir de la suma de los valores de la función de probabilidad en los puntos del intervalo que pertenecen al dominio de la función:

$$Pr[a \leq X \leq b] = \sum_{x_i=a}^{b} f_X(x_i)$$

B.3. Distribuciones de probabilidad

B.3.1. Distribución de Bernoulli

Un ensayo de Bernoulli es un experimento que solo puede tener dos posibles resultados, éxito con probabilidad p o fracaso con probabilidad $q = 1 - p$. Sea X una variable aleatoria discreta que representa la probabilidad de tener *éxito* y se realiza un experimento con dos posibles resultados (éxito o fracaso), entonces X tiene una distribución de Bernoulli de parámetro p, con $0 \leq p \leq 1$:

$$f_X(x) = p^x \ q^{1-x} \ ; \ x = 0, 1$$

La variable x puede tomar el valor de éxito (1) o fracaso (0).

B.3.2. Distribución geométrica

Sea X una variable aleatoria discreta que representa el número de intentos (ensayos de Bernoulli) que se deben realizar para que ocurra el primer éxito. Entonces X tiene una distribución geométrica con parámetro p (probabilidad de éxito de la variable de Bernoulli):

$$f_X(x_i) = q^{x_i - 1} \ p \ ; \ x_i = 1, 2, \ldots$$

Una variable aleatoria de Bernoulli tiene las siguientes características:

- El resultado del ensayo puede ser éxito o fracaso.
- Los resultados de los ensayos son independientes.
- Cada ensayo tiene la misma probabilidad de éxito p.

Además, los ensayos no tienen un orden específico de ocurrencia de éxito, es decir, el éxito puede ocurrir en el primer intento o en el intento n (con n no acotado).

B.3.3. Valor esperado

El valor esperado de una variable aleatoria discreta X está definido como

$$E(X) = \sum_{\forall x_i} x_i \Pr[X = x_i] \tag{B.1}$$

El valor esperado de una variable aleatoria X con distribución geométrica con probabilidad de éxito p es el número esperado de intentos hasta obtener el primer éxito, y está dado por [2]:

$$E(X) = \frac{1}{p}$$

Demostración. El valor esperado de cualquier variable aleatoria discreta X está dado por B.1, esto es:

$$E(X) = 1 * \Pr[X = 1] + 2 * \Pr[X = 2] + 3 * \Pr[X = 3] + \cdots \qquad (a)$$

$$E(X) = p + 2p(1-p) + 3p(1-p)^2 + \cdots \qquad (b)$$

Si se multiplica ambos lados de la ecuación por $(1-p)$ se tiene:

$$(1-p)E(X) = p(1-p) + 2p(1-p)^2 + \cdots \qquad (c)$$

Si se resta (b) - (c) se tiene:

$$E(X) - (1-p)E(X) = p + p(1-p) + p(1-p)^2 + \cdots \qquad (d)$$

Desarrollando el lado izquierdo de la ecuación se tiene:

$$E(X) - E(X) + p * E(x) = p + p(1-p) + p(1-p)^2 + \cdots \qquad (e)$$

Si se divide ambos lados de la ecuación entre p se tiene:

$$E(x) = 1 + (1-p) + (1-p)^2 + \cdots \qquad (f)$$

Esta serie geométrica es igual a:

$$E(x) = \frac{1}{1-(1-p)} = \frac{1}{p} \qquad (g)$$

□

Un corolario de este resultado es el siguiente: Si la probabilidad de que una variable aleatoria discreta X tome un cierto valor v es $\frac{1}{m}$

$$p = Pr[X = v] = \frac{1}{m}$$

entonces, sea Y el número esperado de evaluaciones que hay que hacer de X para que $X = v$ por primera vez, el valor esperado es:

$$E(Y) = \frac{1}{1/m} = m$$

B.3.4. Distribución uniforme discreta

En esta distribución la variable aleatoria asume cada uno de sus valores con la misma probabilidad. Sea X una variable aleatoria que toma los valores $x_1, x_2, \ldots, x_n$ con igual probabilidad

$$f_X(x_i) = Pr[X = x_i] = \frac{1}{n},\ x_i = x_1, x_2, \ldots, x_n$$

Apéndice C

Teoría de gráficas

Para modelar los protocolos se ocupó la teoría de gráficas. En los protocolos propuestos se modeló la instancia del problema original como una gráfica, esta transformación permitió tratar la solución del problema como incidencia entre nodo y arista. Además, tener esa granularidad (a nivel de nodo y arista) permitió, al momento de mostrar la solución parcial, abrir siempre la misma cantidad de información en cada interacción.

C.1. Gráfica

Una gráfica G consiste en un conjunto de vértices no vacío V y un conjunto de aristas E que une dos vértices $v_i, v_j \in V$ [4]. Por lo anterior, una gráfica se define como $G = (V, E)$.

C.2. Gráfica completa

Una gráfica G con n vértices es completa si existe una arista entre cada par de vértices $v_i, v_j \in V$. El número total de aristas en una gráfica completa es $|E| = \frac{n(n-1)}{2}$ [4].

C.3. Gráfica isomorfa

Sean $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ dos gráficas, se dice que G_1 es isomorfa a G_2 si existe una función biyectiva $f : V_1 \to V_2$, de tal manera que para cualquier par de vértices $u, v \in V_1$:

$$\{u, v\} \in E_1 \iff \{f(u), f(v)\} \in E_2$$

La función f es el isomorfismo entre G_1 y G_2 [4]. De manera intuitiva se puede decir que dos gráficas son isomorfas si son las mismas después de renombrar sus vértices.

C.4. Gráfica bipartita

Una gráfica bipartita es una gráfica que particiona sus vértices en dos conjuntos L y R de tal manera que cada arista es incidente en un vértice de L y en un vértice de R [4].

Apéndice D

Máquinas de Turing

D.1. Máquina de Turing probabilística de tiempo polinomial

Una máquina de Turing Probabilística de tiempo polinomial ($MT\,PTP$) es una máquina de Turing que tiene una cinta extra con valores aleatorios, cada bit de esta cinta aleatoria es elegido con probabilidad uniforme y de forma independiente. El tiempo de ejecución de una $MT\,PTP$ está en función de la cinta aleatoria particular elegida en la ejecución.

Una $MT\,PTP$ A se ejecuta en un tiempo $T(n)$ si para cualquier cadena $x \in \{0,1\}^*$ y para cualquier cinta aleatoria r, $A(x,r)$ se detiene en a lo más $T(|x|)$ pasos. A se ejecuta en tiempo polinomial si existe una constante c tal que A se ejecuta en tiempo $T(|x|) = n^c$ [24].

D.2. Máquina de Turing no uniforme probabilística de tiempo polinomial

Una máquina de Turing no uniforme probabilística de tiempo polinomial A es una secuencia infinita de máquinas de Turing probabilístas de tiempo polinomial $\{A_1, A_2, \ldots\}$. Dependiendo de la longitud de la entrada x se ejecuta la máquina $A_{|x|}$, es decir, si $|x| = n$ se ejecuta la máquina A_n [13]. Además, A debe cumplir con:

- Existe una función polinomial $p()$ tal que para cada entrada n, la descripción de la máquina A_n está limitada por $p(n)$, es decir, $|A_n| \leq p(n)$.
- Existe una función polinomial $q()$ tal que para cada entrada n, el tiempo de ejecución de la máquina A_n en la longitud de la entrada $|n|$ está limitada por $q(n)$, es decir, A_n se ejecuta en un tiempo $\leq q(n)$

D.3. Máquina de Turing interactiva

Una máquina de Turing interactiva (MTI) es una máquina de Turing que tiene una cinta de entrada de solo lectura, una cinta de trabajo de lectura y escritura, una cinta aleatoria de solo lectura, una cinta de comunicación de solo lectura y una cinta de comunicación de solo escritura.

La cinta aleatoria contiene una secuencia infinita de bits elegidos con una distribución uniforme, la cual se lee de izquierda a derecha. La cinta de entrada contiene los valores iniciales, comunes para la interacción. La cinta de comunicación de solo escritura permite enviar mensajes y la cinta de comunicación de solo lectura permite recibir mensajes. En la cinta de salida se guarda lo mismo que en la cinta de comunicación de solo escritura (es decir, la información que se envía), en ella (la cinta de salida) se puede ver la distribución de la interacción.

Bibliografía

[1] Cornell University CS 482. Introduction to Algorithms: Proving NP-Completeness. Cornell University, (2006). Consultada el 02-05-2021 de http://www.cs.cornell.edu/courses/cs482/2007sp/NPComplete.pdf.

[2] Jeff. A. Proof of expected value of geometric random variable. Khan Academy, (2017). Consultada el 15-04-2021 de https://www.khanacademy.org/math/ap-statistics/random-variables-ap/geometric-random-variable/v/proof-of-expected-value-of-geometric-random-variable.

[3] WolframAlpha computational intelligence. Límite. Wolfram|Alpha. Consultada el 17-05-2021 de https://www.wolframalpha.com/input/?i=lim+%281-1%2Fx%29%5Ex+as+x-%3Einf.

[4] Mathematics for Computer Science. Graph Theory. MIT Open Course Ware, (2010). Consultada el 17-05-2021 de https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-042j-mathematics-for-computer-science-fall-2010/readings/MIT6_042JF10_chap05.pdf.

[5] Goldston R. Glaser A., Barak B. A zero-knowledge protocol for nuclear warhead verification. *Nature*, page 497–502, (1972).

[6] Wigderson A. Goldreich O., Micali S. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *ACM*, 38(3):692–729, (1991).

[7] Rackoff C. Goldwasser S., Micali S. The knowledge complexity of interactive proof-systems. *ACM*, pages 291–304, (1985).

[8] Rackoff C. Goldwasser S., Micali S. The knowledge complexity of interactive proof systems. *SIAM*, 18(1):186–208, (1989).

[9] Lipmaa H. Secure electronic voting protocols, (2005). Consultado el 21-04-2022 de https://kodu.ut.ee/∼lipmaa/papers/voting4hb.pdf.

[10] Mouatadid L. Introduction to Complexity Theory: 3-Colouring is NP-complete. Budapest University of Technology and Economics, (2014). Consultada el 01-07-2021 de http://cs.bme.hu/thalg/3sat-to-3col.pdf.

[11] Koens T. Morais E., van Wijk C. Zero knowledge set membership, (2018). Consultado el 21-04-2022 de https://www.ingwb.com/binaries/content/assets/insights/themes/distributed-ledger-technology/zero-knowledge-set-membership_whitepaper.pdf.

[12] Information Technology Laboratory | NIST. 1.3.6. Probability Distributions. SUNY Polytechnic Institute. Consultada el 17-05-2021 de `https://www.itl.nist.gov/div898/handbook/eda/section3/eda36.htm`.

[13] Lin R. Lecture 2: Computational Security – Introduction to Cryptography. University of California, Santa Barbara, (2014). Consultada el 30-12-2021 de `https://sites.cs.ucsb.edu/~rachel.lin/courses/14f290G/Lec2.pdf`.

[14] Ostrovsky R. Foundations of Cryptography: Lecture 9. UCLA Samueli School of Engineering, (2005). Consultada el 05-04-2021 de `http://web.cs.ucla.edu/~rafail/TEACHING/WINTER-2005/L9/L9.pdf`.

[15] Pass R. Lecture 3: One-Way Functions. Cornell University, (2009). Consultada el 23-12-2021 de `https://www.cs.cornell.edu/courses/cs6830/2009fa/scribes/lecture3-owf.pdf`.

[16] Luis Rincón. *Estadística descriptiva.* Las prensas de ciencias, 1 edition, 2017.

[17] Karp R.M. Reducibility among combinatorial problems. *Plenum Press*, pages 85–103, (1972).

[18] Chawla S. Greedy Approximations: Set Cover and Min Makespan. University of Wisconsin-Madison, (2006). Consultada el 24-05-2021 de `http://pages.cs.wisc.edu/~shuchi/courses/880-S07/scribe-notes/lecture03.pdf`.

[19] Saarland University. Zero Knowledge Proofs. Max Planck Institute for Informatics, (2014). Consultada el 25-01-2021 de `http://resources.mpi-inf.mpg.de/departments/d1/teaching/ss14/gitcs/notes6.pdf`.

[20] Dodis Y. Lecture 14: Introduction to Cryptography. Computer Science Department at New York University, (2012). Consultada el 01-07-2020 de `https://cs.nyu.edu/courses/fall08/G22.3210-001/lect/lecture14.pdf`.

[21] Zhang Y. Zero-knowledge proofs for machine learning, (2020). Consultado el 21-04-2022 de `https://dl.acm.org/doi/10.1145/3411501.3418608`.

[22] Arora S. y Barak B. *Computational Complexity: A Modern Approach.* Princeton University, 1 edition, 2007.

[23] Goldwasser S. y Bellare M. *Lecture Notes on Cryptography.* Cambridge, 1 edition, 2008.

[24] Pass R. y Shelat A. *A Course in Cryptography.* 3 edition, 2010.

[25] Megiddo N. y Tamir A. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, (1982).

[26] Zcash. What are zk-SNARKs?, (2021). Consultado el 21-04-2022 de `https://z.cash/technology/zksnarks/`.