Universidad Nacional Autónoma de México

Facultad de Ciencias

Reducciones polinomiales en la Teoría de Gráficas.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
MATEMÁTICO

PRESENTA:
Clara Valentina Gallardo Gutiérrez

DIRECTOR DE TESIS:
Dr. César Hernández Cruz

Ciudad Universitaria, Cd. Mx., 2016

Hoja de datos del jurado

1. Datos del alumno
   Gallardo
   Gutiérrez
   Clara Valentina
   55 4755 7241
   Universidad Nacional Autónoma de México
   Facultad de Ciencias
   Matemáticas
   307643885

2. Datos del tutor
   Dr.
   César
   Hernández
   Cruz

3. Datos del sinodal 1
   M. en C.
   Rafael
   Rojas
   Barbachano

4. Datos del sinodal 2
   Dra.
   Rita Esther
   Zuazua
   Vega

5. Datos del sinodal 3
   Mat.
   Jesús
   Alva
   Samos

6. Datos del sinodal 4
   Dr.
   Carlos
   Torres
   Alcaráz

7. Datos del trabajo escrito
   Relaciones polinomiales en la teoría de gráficas
   93 p
   2016

Reducciones polinomiales en la Teoría de Gráficas

Clara Valentina Gallardo Gutiérrez

2016

Índice general

Introducción

En 1931, Kurt Gödel publicó su famoso artículo *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme* (Sobre las proposiciones formalmente indecidibles en Principia Mathematica y sistemas relacionados), donde demuestra, a grandes rasgos, que dentro de todo sistema axiomático consistente, es decir que no tiene contradicciones, y que sea lo bastante fuerte para definir el concepto de números naturales, se puede construir una afirmación que ni se puede demostrar, ni se puede refutar dentro de ese sistema. Dando así, respuesta a las preguntas que David Hilbert se planteó en 1928: ¿Son completas las matemáticas?, ¿son consistentes? ¿son decidibles, en el sentido de que exista un método definido que se pueda aplicar a cualquier aseveración matemática y que determine si dicha aseveración es cierta?. Y fue así, como el articulo de Gödel, abrió nuevos intereses en la formalización de la Matemática y determinar qué métodos son válidos en la resolución de problemas. En 1936, el matemático Alan Turing publicó otro articulo famoso, *On computable numbers, with an application to the Entscheidungsproblem* (Los números computables, con una aplicación al Entscheidungsproblem), donde propuso que la tercera pregunta de Hilbert podía resolverse con la ayuda de una máquina, proponiendo así un concepto abstracto de lo que es una máquina (máquinas de Turing) y demostrar así la existencia de problemas que no son decidibles.

Este nuevo concepto de máquina fue la base para desarrollar la Teoría de la Computación y se empezaron a preguntar qué es todo lo que puede realizar una máquina y después, cuánto tiempo tardaría una máquina en resolver algún problema, lo que dio inicio al estudio de la Teoría de la Complejidad

Computacional, donde se compara la dificultad de los distintos métodos para la resolución de un problema.

En este trabajo veremos a grandes rasgos lo que es una máquina de Turing y algunas de sus diferentes variantes, las cuales nos darán una clasificación para los problemas decidibles; la clase P, de problemas de decisión, comprende aquellos que se pueden resolver en un tiempo polinomial por una máquina de Turing determinista, mientras que los problemas NP son aquellos que se resuelven en un tiempo polinomial por una máquina de Turing no determinista. Esto para poder definir la clase de problemas NP-completos.

En la parte de preliminares de lógica, veremos cuando una expresión es satisfacible, para así dar entrada al problema de decisión de satisfacibilidad booleana, SAT, que establece que dada una expresión booleana, podemos preguntarnos si ésta es satisfacible. En 1971, Stephen Cook, demostró que SAT es NP-completo. El teorema de Cook fue la primera prueba de existencia de problemas NP-completos, esta prueba la veremos en el Capítulo 2.

Para probar que otro problema NP es NP-completo es suficiente con mostrar que este se puede reducir en tiempo, a lo más, polinomial, desde un problema que ya se sabe que es NP-completo. En éste trabajo demostraremos que existe una reducción en tiempo polinomial de cualquier instancia de SAT desde una instancia equivalente de 3-SAT.

Actualmente, se sabe que en la Teoría de las Gráficas existen muchos problemas NP-completos, entre ellos saber si una gráfica es 3-colorable. En esta tesis, probaremos que 3-coloreabilidad es un problema NP-completo, y se establece una reducción en tiempo polinomial de 3-coloración desde SAT.

La intención de este trabajo es hacer reducciones en tiempo lineal de diversos problemas de la teoría de gráficas, las cuales veremos en el Capítulo 4, desde el problema de 3-coloración. Muchas de las reducciones están basadas en el artículo desarrollado por Creignou, *The class of problems that are linearly equivalent to Satisfiability or a uniform method for proving NP- completeness*, [2], (La clase de problemas que son linealmente equivalentes a Satisfacibilidad o un método uniforme para probar NP-completez) de las cuales algunas fueron mejoradas para garantizar que la reducción se hiciera en tiempo lineal. También se da un ejemplo de una reducción en tiempo polinomial no lineal.

A lo largo de este trabajo, demostraremos la importancia del concepto de NP-completo en los problemas decidibles de la Teoría de Gráficas.

Capítulo 1

Preliminares

1.1. Lógica

En esta sección daremos una breve introducción a la lógica matemática para así introducir la noción de satisfacibilidad para fórmulas bien formadas. Veremos que toda formula bien formada es lógicamente equivalente a otra escrita en la forma normal conjuntiva. Esto nos será útil para la demostración del Teorema de Cook que veremos más adelante. Debido a que este no es el tema central de este trabajo, no seremos tan rigurosos como lo exige el estudio de la lógica matemática.

En las matemáticas siempre hemos utilizado símbolos particulares para expresar resultados y conjeturas. El lenguaje de la matemática utiliza parte del lenguaje utilizado comúnmente, en nuestro caso el español, y agrega símbolos particulares como $\int$ ó $\in$. Estos símbolos tienen "reglas gramaticales" precisas para que sus expresiones tengan sentido.

En el intento de formalizar la matemática, surgió el concepto de lenguaje formal, el cual está formado por un conjunto de símbolos. En el presente caso de la lógica de proposiciones, algunos de los símbolos son:

$$(,), \rightarrow, \neg, A_1, A_2$$

que se combinan entre sí para hacer expresiones bien formadas mediante reglas "gramaticales". Estas expresiones tendrán traducciones admisibles entre el español y el lenguaje formal. Los símbolos A_1, A_2, ... pueden ser traducciones de sentencias declarativas del español que pueden ser verdaderas o falsas.

Veamos formalmente, cuales serán los símbolos que utilizaremos en el lenguaje de la lógica de proposiciones:

Símbolo	Nombre	Observación
(	Paréntesis izquierdo	Símbolo de puntuación
)	Paréntesis derecho	Símbolo de puntuación
$\neg$	Símbolo de negación	Español: no
$\wedge$	Símbolo de conjunción	Español: y
$\vee$	Símbolo de disyunción	Español: o (inclusivo)
$\rightarrow$	Símbolo de condicional	Español: si... entonces
$\leftrightarrow$	Símbolo de bicondicional	Español: si y sólo si
A_1	Primer símbolo de proposición	
A_2	Segundo símbolo de proposición	
...		
A_n	n-ésimo símbolo de proposición	
...		

Una de nuestras primeras "reglas gramaticales" será que ningún símbolo es una sucesión finita de otros símbolos. El propósito de este supuesto es para asegurar que las secuencias finitas de símbolos son descomponibles de manera única. A los símbolos $\neg$,$\wedge$,$\vee$,$\rightarrow$ y $\leftrightarrow$ los llamaremos **conectivos**. Los conectivos y los símbolos de puntuación son los símbolos lógicos. Los símbolos para las proposiciones son símbolos no-lógicos. Su traducción no es fija, se prestan a ser interpretados de varias maneras.

Hagamos un análisis más detallado de como funcionan los conectivos. Sean α, β, γ expresiones "bien escritas" del lenguaje formal, entonces:

1. **Conjunción**

   Aquiles se casó y tuvo hijos.

   Esta oración está formada por dos expresiones, $\alpha = \{$Aquiles se casó$\}$ y $\beta = \{$tuvo hijos$\}$, unidas por el conectivo "Y". La primera observación que debemos hacer con respecto a este conectivo, es que nosotros consideraremos el caso en el que "Y" es conmutativo, es decir, la expresión anterior y la siguiente, tienen el mismo significado:

Tuvo hijos y Aquiles se casó.

El símbolo que usaremos para este conectivo será: $\wedge$. Al afirmar que ocurre α como β, estamos afirmando que ocurre $(\alpha \wedge \beta)$. E igual al contrario, si afirmamos que ocurre $(\alpha \wedge \beta)$, estamos afirmando que tanto ocurre α como β. A los elementos α y β se les llama conyuntos.

2. **Disyunción**

   Aquiles cumple años o el Sr. T. formalizó el esquema de respuesta ϵ_0.

   Esta oración está formada por $\alpha = \{$Aquiles cumple años$\}$ y
   $\beta = \{$el Sr. T. formalizó el esquema de respuesta $\epsilon_0\}$, expresiones unidas por el conectivo "O". En el español, este conectivo tiene dos usos, el inclusivo y el exclusivo, aquel que permite que ocurran ambas cosas al mismo tiempo y el que permite que ocurra solo una, respectivamente. Nosotros formalizaremos el inclusivo y de esta forma, el exclusivo quedará en función de este.

   El símbolo que usaremos para este conectivo será: $\vee$. Al afirmar que ocurre al menos uno de α o β, estamos afirmando que ocurre $(\alpha \vee \beta)$. Y viceversa, si afirmamos que ocurre $(\alpha \vee \beta)$, decimos que ocurre α u ocurre β u ocurren ambos. A los elementos α y β se les llama disyuntos.

3. **Negación**

   No es la primera vez que al Sr. Perezoso le presentan un Bicíclope.

   El conectivo que estamos usando en esta oración es "NO", aplicada a la expresión α, donde
   $\alpha = \{$es la primera vez que al Sr. Perezoso le presentan un Bicíclope$\}$. Este es el único conectivo singular. En el español sigue al sujeto o niega un verbo.

   El símbolo que usaremos para este conectivo será: $\neg$. Al afirmar que ocurre $(\neg\alpha)$ estamos afirmando que no ocurre α. Y viceversa, si no ocurre α, afirmamos que $(\neg\alpha)$.

4. **Condicional**

   Si la tortuga gana entonces Aquiles se enfurece.

En esta oración, el conectivo es de la forma: "SI... ENTONCES...", aplicado a $\alpha = \{\text{la tortuga gana}\}$ y $\beta = \{\text{Aquiles se enfuerece}\}$. Si P entonces Q, estamos diciendo es que cada vez que ocurra P forzosamente ocurre Q, y que en caso de que no ocurra P, puede o no ocurrir Q.

El símbolo que usaremos para este conectivo será: $\rightarrow$. Los distintos significados o maneras de interpretar $\alpha \rightarrow \beta$ son:

a) α es una condición suficiente para β.

b) β si α.

5. **Bicondicional**

El Sr. T. gana la carrera si y sólo si Aquiles le da ventaja al Sr. T.

En esta oración, el conectivo es de la forma:"... SI Y SÓLO SI...", aplicado a $\alpha = \{\text{El Sr. T. gana la carrera}\}$ y $\beta = \{\text{Aquiles le da ventaja}\}$. Este conectivo no suele ser utilizado como tal en el español, sin embargo en Matemáticas es de uso común.

El símbolo que usaremos para este conectivo será: $\leftrightarrow$. Los distintos significados o maneras de interpretar $\alpha \leftrightarrow \beta$ son:

a) α es una condición necesaria y suficiente para β.

b) β si y sólo si α.

Una **expresión** es una secuencia finita de símbolos. Podemos especificar una expresión mediante la concatenación de los nombres de los símbolos; por lo tanto $(\neg A_1)$ es la secuencia $(, \neg, A_1,)$. Esta notación se extiende: Si α y β son secuencias de símbolos, entonces $\alpha\beta$ es la secuencia que consiste, primero, de los símbolos en la secuencia de α seguido de los símbolos en la secuencia de β. Por ejemplo, si α y β están dados por:

$$\alpha = (\neg A_1)\ ,\ \beta = A_2$$

entonces $(\alpha \vee \beta)$ es la expresión $((\neg A_1) \vee A_2)$.

Definimos como **fórmulas bien formadas** (FBS) a las expresiones "gramaticalmente correctas"; las expresiones sin sentido deben ser excluidas. La definición tendrá las siguientes consecuencias:

1. Todo símbolo de proposición es una fórmula bien formada. Les llamaremos fórmulas atómicas.
2. Si p y q son fórmulas bien formadas, entonces $\neg(p)$, $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$ y $(p \leftrightarrow q)$ son fórmulas bien formadas.
3. Una expresión no es una fórmula bien formada a menos de que sea de la forma descrita en los dos incisos anteriores.

Una fórmula bien formada es una expresión que puede ser construida por los símbolos de predicados aplicando un número finito de veces las fórmulas de construcción definidas por:

- $\mathcal{E}_{\neg}(\alpha) = (\neg\alpha)$.
- $\mathcal{E}_{\wedge}(\alpha, \beta) = (\alpha \wedge \beta)$.
- $\mathcal{E}_{\vee}(\alpha, \beta) = (\alpha \vee \beta)$.
- $\mathcal{E}_{\rightarrow}(\alpha, \beta) = (\alpha \rightarrow \beta)$.
- $\mathcal{E}_{\leftrightarrow}(\alpha, \beta) = (\alpha \leftrightarrow \beta)$.

Definimos una *sucesión de construcción* como una sucesión finita de expresiones $\langle \epsilon_1, ..., \epsilon_n \rangle$ tal que, para cada $i \leq n$ tenemos que se cumple uno de los siguientes casos:

- ϵ_i es un símbolo de proposición.
- $\epsilon_i = \mathcal{E}_{\neg}(\epsilon_j)$ para algún $j < i$.
- $\epsilon_i = \mathcal{E}_{\square}(\epsilon_j, \epsilon_k)$ para algún $j < i$, $k < i$.

Donde $\square$ es un símbolo de conectivos $\wedge$,$\vee$,$\rightarrow$ o $\leftrightarrow$. Así, las fórmulas bien formadas pueden definirse como la expresión α tal que alguna sucesión de construcción termina con α.

Este tipo de construcciones nos induce un principio de inducción. En lo siguiente, diremos que un conjunto S es cerrado bajo una función f n-aria si y sólo si siempre que $x_1, x_2, ..., x_n \in S$ entonces $f(x_1, ..., x_n) \in S$. No formalizaremos el hecho de que el conjunto S descrito a continuación, es un conjunto que acepta un principio de inducción puesto que esto no es estrictamente necesario para este trabajo.

Teorema 1.1. Principio de Inducción. Si S es un conjunto de fórmulas bien formadas que contiene a todos los símbolos de proposición y es cerrado bajo los conectivos, entonces S es el conjunto de todas las fórmulas bien formadas.

Demostración: Sea, α una fórmula bien formada. Si α es un símbolo de proposición, entonces $\alpha \in S$.

Supongamos que α es el último elemento de una secuencia de construcción $\langle \epsilon_1, ..., \epsilon_n \rangle$. Supongamos, como hipótesis inductiva, que $\epsilon_j \in S$ para toda $j < i$. Demostraremos que $\epsilon_i \in S$. Si ϵ_i es un símbolo de proposición, entonces $\epsilon_i \in S$ puesto que S contiene a todos los símbolos de proposiciones. Si $\epsilon_i = \mathcal{E}_{\neg}(\epsilon_j)$ para algún $j < i$, entonces $\epsilon_i \in S$ puesto que S es cerrado bajo $\neg$. Si $\epsilon_i = \mathcal{E}_{\square}(\epsilon_j, \epsilon_k)$ para algún $j < i$, $k < i$, donde $\square$ es un conectivo $\wedge$,$\vee$,$\rightarrow$ o $\leftrightarrow$ entonces $\epsilon_i \in S$ puesto que S es cerrado bajo $\wedge$,$\vee$,$\rightarrow$ y $\leftrightarrow$.

■

Una fórmula puede ser o verdadera o falsa, dependiendo de la verdad o falsedad de las fórmulas más simples que son sus fórmulas atómicas. Por ejemplo, la verdad o falsedad de $(p \vee q)$ depende de los valores de verdad de p y q. Así, podemos determinar la verdad o falsedad de una fórmula recurriendo a los valores de verdad dados por alguna interpretación de sus átomos. Fijemos el conjunto $\{F, V\}$ de *valores de verdad*, que está formado por los elementos:

F, llamado Falso,
V, llamado Verdadero.

La *asignación de verdad* v de un conjunto S de símbolos de predicado es la función:

$$v : S \rightarrow \{F, V\}.$$

Sea $\overline{S}$ el conjunto de fórmulas bien formadas que pueden ser construidas a partir de S un conjunto de símbolos de proposiciones y conectivos. $\overline{v}$ es la extensión de v, tal que:

$$\overline{v} : \overline{S} \rightarrow \{F, V\},$$

que asigna el correcto valor de verdad a cada fórmula bien formada en $\overline{S}$. Se deben cumplir las siguientes condiciones:

1. Para cualquier $A \in S$, $\overline{v}(A) = v(A)$

   Para $\alpha, \beta \in \overline{S}$:

2. $\overline{v}((\neg\alpha)) = \begin{cases} V & \text{si } \overline{v}(\alpha) = F, \\ F & \text{en otro caso.} \end{cases}$

3. $\overline{v}((\alpha \wedge \beta)) = \begin{cases} V & \text{si } \overline{v}(\alpha) = V \text{ y } \overline{v}(\beta) = V, \\ F & \text{en otro caso.} \end{cases}$

4. $\overline{v}((\alpha \vee \beta)) = \begin{cases} V & \text{si } \overline{v}(\alpha) = V \text{ o } \overline{v}(\beta) = V \text{ (o ambos)}, \\ F & \text{en otro caso.} \end{cases}$

5. $\overline{v}((\alpha \to \beta)) = \begin{cases} F & \text{si } \overline{v}(\alpha) = V \text{ y } \overline{v}(\beta) = F, \\ V & \text{en otro caso.} \end{cases}$

6. $\overline{v}((\alpha \leftrightarrow \beta)) = \begin{cases} V & \text{si } \overline{v}(\alpha) = \overline{v}(\beta), \\ F & \text{en otro caso.} \end{cases}$

Diremos que una asignación de verdad v **satisface** φ si y sólo si $\overline{v}(\varphi) = V$ donde el dominio de v contiene a todos los símbolos de proposiciones de φ.

Ahora consideremos un conjunto Σ de fórmulas bien formadas y τ otra fórmula bien formada. Diremos que Σ **implica tautologicamente** τ ($\Sigma \models \tau$) si y sólo si toda asignación de verdad de los símbolos de predicado en Σ y τ que satisface a cada miembro de Σ también satisface a τ. Cuando $\Sigma = \emptyset$ diremos que τ es una **tautología**.

Si Σ es el unitario $\{\sigma\}$, entonces escribiremos $\sigma \models \tau$ en lugar de $\{\sigma\} \models \tau$. Si se cumple que $\sigma \models \tau$ y $\tau \models \sigma$, diremos que σ y τ son tautológicamente equivalentes.

Mostraremos un procedimiento, ya conocido, para decidir si dadas las fórmulas bien formadas $\sigma_1, ..., \sigma_n$ y τ, se cumple, o no, que:

$$\{\sigma_1, ..., \sigma_n\} \models \tau.$$

En particular, si $n = 0$, decidir cuando una fórmula bien formada es una tautología. Por ejemplo, mostraremos que:

$$(\neg(A \vee B)) \models ((\neg A) \wedge (\neg B)).$$

Para esto, consideraremos todas las posibles asignaciones de verdad para $\{A, B\}$. En este caso se tienen 4 posibles combinaciones, que podemos observar en la siguiente tabla:

A	B
V	V
V	F
F	V
F	F

Esta tabla se puede expandir incluyendo $(\neg(A \vee B))$ y $((\neg A) \wedge (\neg B))$. Para cada fórmula, agregamos los valores de verdad debajo del conectivo usado, según las definiciones que vimos con anterioridad. Entonces, la tabla se vería de la siguiente forma:

A	B	$(\neg(A \vee B)$	$((\neg A) \wedge (\neg B))$
V	V	**F** V V V	F V **F** F V
V	F	**F** V V F	F V **F** V F
F	V	**F** F V V	V F **F** F V
F	F	**V** F F F	V F **V** V F

En la tabla, no es necesario escribir las primeras dos columnas. Podemos observar que las asignaciones de verdad que satisfacen a $(\neg(A \vee B))$ también satisface a $((\neg A) \wedge (\neg B))$, que en este caso la última asignación de valores de verdad que se representa en la tabla, es la que los satisface. Más aún, podemos observar que se cumple $(\neg(A \vee B)) \models ((\neg A) \wedge (\neg B))$ y también se cumple $((\neg A) \wedge (\neg B)) \models (\neg(A \vee B))$.

Observemos que este método, para ver que valores de verdad satisfacen una fórmula bien formada, aplicada a una fórmula con n símbolos de predicado, requiere una tabla con 2^n filas, así, si n aumenta, entonces 2^n crece de manera exponencial.

En lo que sigue, sea φ una fórmula bien formada con n letras proposicionales, diremos que f es una función de verdad de n argumentos, con dominio $\{Img(\overline{v})\}$, donde $\overline{v}$ es la extensión de una asignación de verdad y la imagen

de f es el conjunto $\{F, V\}$. Para esto, numeraremos las letras proposicionales de φ. Por ejemplo, $(A \to B)$ genera la función de verdad:

x_1	x_2	$f(x_1, x_2)$
V	V	V
F	V	V
V	F	F
F	F	V

Toda fórmula bien formada que contenga a n letras proposicionales, genera una función de verdad con n argumentos y con $img(f) = \{V, F\}$.

Así, toda función de verdad, es la tabla de verdad de alguna fórmula. Observemos que dos fórmulas que son tautológicamente equivalentes generan la misma función.

Proposición 1.2. Toda función esta generada por una función de verdad que está formada únicamente por los conectivos $\neg$, $\vee$ y $\wedge$.

Prueba: Sea $f(x_1, ..., x_n)$ una función de verdad. Podemos representar a f con una tabla de verdad con 2^n renglones, donde cada renglón representa alguna asignación de verdad de las variables $x_1, ..., x_n$, seguida del correspondiente valor de $f(x_1, ..., x_n)$. Si $1 \leq i \leq 2^n$, sea C_i la conjunción $U_1^i \wedge U_2^i \wedge ... \wedge U_n^i$, donde U_j^i es A_j si en el i-ésimo renglón de la tabla de verdad, x_j toma el valor de verdad V, y U_j^i es $\neg A_j$ si x_j toma el valor de verdad F en ese renglón. Sea D la disyunción de todos estos C_i tal que f toma el valor de verdad V para el i-ésimo renglón de la tabla de verdad. Si no existen dichos renglones, entonces f toma el valor de verdad F; y D es $A_1 \vee \neg A_1$ lo cual satisface el teorema. Notemos que D está formado únicamente por los conectivos $\wedge$, $\vee$ y $\neg$. Para ver que f es la función de verdad de D, dada una asignación de verdad a las letras proposicionales $A_1, ..., A_n$ y asumamos que la correspondiente asignación de las variables $x_1, ..., x_n$ es el renglón k de la tabla de verdad de f. Entonces C_k toma el valor de verdad T para esta asignación y cualquier otra C_i tiene el valor de verdad F. Si el valor de f es V para el renglón k, entonces C_k, es un disyunto de D. Así D toma el valor de verdad V para esta asignación. Si f toma el valor F para el renglón k, entonces C_k no es un disyunto de D y todos los disyuntos de toman el valor de verdad F para esta asignación. Entonces, D también tiene el valor F. Así, D genera la función de verdad f.

□

Así, toda fórmula es lógicamente equivalente a una que tiene las mismas letras proposicionales, cuyos únicos conectivos son la negación, la disyunción y la conjunción, y tal que las negaciones sólo afectan a las letras proposicionales. Llamaremos a estas expresiones, expresiones booleanas.

Con estos preliminares, podemos ahora plantear el siguiente problema:

- **Dada una expresión booleana, ¿es satisfacible?**

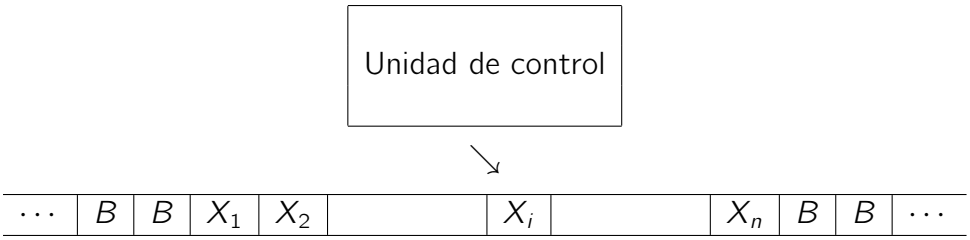
Este problema lo representaremos con las abreviación, SAT. El planteamiento de este problema será esencial en lo que sigue de este trabajo.

1.2. Máquinas de Turing

Como mencionamos en un principio, fue en 1936, que el matemático Alan Turing, en su artículo *On computable numbers, with an application to the Entscheidungsproblem*, definió, por primera vez, a las máquinas de Turing.

Las máquinas de Turing han servido como un modelo de la Teoría de la Computación, la idea básica es estudiar los procesos algorítmicos con este modelo. En este trabajo en particular nos enfocaremos en las máquinas de Turing no deterministas que describiremos más adelante, para adentrarnos en el campo de la Teoría de la Complejidad Computacional.

En la siguiente figura podemos visualizar una máquina de Turing, que consta de una *unidad de control*, que se puede encontrar en cualquiera de un conjunto de estados. Hay una *cinta* infinita dividida en cuadrados o *casillas* y cada casilla puede contener un símbolo de entre un número finito de símbolos.



Inicialmente, la *entrada*, que es una cadena de símbolos de longitud finita elegidos del *alfabeto de entrada*, se coloca en la cinta. Las restantes casillas de la cinta almacenan un símbolo especial llamado *espacio en blanco*, el cual

es un *símbolo de cinta*, pero no un símbolo de entrada. Pueden existir otros símbolos a parte de los símbolos de entrada y del espacio en blanco. Existe una *cabeza de la cinta* que siempre está situada en una de las casillas de la cinta. Inicialmente, la cabeza de la cinta está situada en la casillla más a la izquierda que contiene al primer símbolo de la entrada. Un *movimiento* de la máquina de Turing es una función del estado de la unidad de control y el símbolo de cinta al que señala la cabeza. En un movimiento, la máquina de Turing:

1. Cambiará de estado. El siguiente estado puede ser opcionalmente el mismo que el estado actual.

2. Escribirá un símbolo de cinta en la casilla que señala la cabeza. Este símbolo de cinta reemplaza a cualquier símbolo que estuviera anteriormente en dicha casilla, sin embargo, el símbolo escrito puede ser el mismo que el que ya se encontraba allí.

3. Moverá la cabeza de la cinta hacia la izquierda o hacia la derecha. No se permite que la cabeza quede estacionaria.

Formalmente una **máquina de Turing** (MT) es una séptupla:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

donde:

Q Es el conjunto finito de *estados* de la unidad de control.

Σ Es el conjunto finito de *símbolos de entrada*.

Γ Es el conjunto de *símbolos de cinta*; Σ es un subconjunto de Γ.

δ Es la *función de transición*. Los argumentos de $\delta(q, X)$ son un estado q y un símbolo de cinta X. El valor de $\delta(q, X)$, si está definido, es (p, Y, D), donde:

1. p es el siguiente estado de Q.

2. Y es el símbolo de Γ, que se escribe en la casilla que señala la cabeza y que sustituye a cualquier símbolo que se encontrara en ella.

3. D es una *dirección* y puede ser L o R, que nos indica la dirección en la que se mueve la cabeza, izquierda o derecha respectivamente.

q_0 Es el *estado inicial*, un elemento de Q, en el que inicialmente se encuentra la unidad de control.

B Es el simbolo *espacio en blanco*. Este símbolo pertenece a Γ pero no a Σ, y no es un símbolo de entrada.

F Es el conjunto de los estados *finales* o de *aceptación*, un subconjunto de Q.

Utilizaremos la cadena $X_1X_2...X_{i-1}qX_iX_{i+1}...X_n$ para describir las configuraciones de la máquina de Turing, donde q es el estado de la máquina de Turing, la cabeza de la cinta está señalando al símbolo i-ésimo empezando por la izquierda y $X_1X_2...X_n$ es parte de la cinta comprendida entre los símbolos distintos de los espacios en blanco más a la izquierda y más a la derecha. Como excepción, si la cabeza está a la izquierda del símbolo más a la izquierda que no es un espacio en blanco o a la derecha del símbolo más a la derecha que no es un espacio en blanco, entonces prefijo o un sufijo de $X_1X_2...X_n$ serán espacios en blanco e i será igual a 1 o a n, respectivamente.

Describiremos los movimientos de una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ utilizando la notación $\vdash_M$. cuando sea claro que hacemos referencia a una máquina de Turing, sólo usaremos $\vdash$ para indicar los movimientos. Cuando se hable de más de un movimiento, utilizaremos la notación $\vdash^*$.

Sea $\delta(q, X_i) = (p, Y, L)$; es decir, el siguiente movimiento a la izquierda. Entonces:

$$X_1X_2...X_{i-1}qX_iX_{i+1}...X_n \vdash_M X_1X_2...X_{i-2}pX_{i-1}YX_{i+1}...X_n$$

Donde la máquina cambia al estado p y la cabeza señala la casilla $i-1$. A continuación haremos algunas observaciones importantes:

- Si $i = 1$, entonces M se mueve al espacio en blanco que se encuentra a la izquierda de X_1:

$$qX_1X_2...X_n \vdash_M pBYX_2...X_n$$

- Si $i = n$ y $Y = B$, entonces el símbolo B escrito sobre X_n se añade a la secuencia infinita de los espacios en blanco que hay después de la cadena de entrada y no aparecerá en la siguiente configuración. Por tanto,

$$X_1X_2...X_{n-1}qX_n \vdash_M X_1X_2...X_{n-2}pX_{n-1}$$

Ahora, sea $\delta(q, X_i) = (p, Y, R)$, el siguiente movimiento a la derecha. Entonces:

$$X_1X_2...X_{i-1}qX_iX_{i+1}...X_n \vdash_M X_1X_2...X_{i-1}YpX_{i+1}...X_n$$

En este caso, el movimiento refleja el hecho de que la cabeza se ha movido a la casilla $i + 1$ y nuevamente tenemos dos observaciones importantes:

- Si $i = n$, entonces en la casilla $i+1$ almacena un espacio en blanco, por lo que dicha casilla no formaba parte de la configuración anterior. Así:

$$X_1X_2...X_{n-1}qX_n \vdash_M X_1X_2...X_{n-1}YpB$$

- Si $i = 1$ y $Y = B$, entonces el símbolo B escrito sobre X_1 se añade a la secuencia infinita de los espacios en blanco anteriores a la cadena de entrada y no aparecerá en la siguiente configuración:

$$qX_1X_2...X_n \vdash_M pX_2...X_n$$

Para seguir con la descripción de las máquinas de Turing, debemos definir ciertos conceptos con respecto a las entradas que puede aceptar una máquina de Turing. Diremos que un **alfabeto**, Σ, es un conjunto de símbolos finito y no vacío, por ejemplo:

1. $\Sigma = \{0, 1\}$, el alfabeto binario.
2. $\Sigma = \{a, b, c, ..., z\}$, el conjunto de todas las letras minúsculas.

Una **cadena de caracteres** (o **palabra**) , es una secuencia finita de símbolos seleccionados por algún alfabeto. Por ejemplo 01001110 es una cadena del alfabeto binario $\Sigma = \{0, 1\}$. La **cadena vacía**, ε, es aquella que no tiene símbolos y puede construirse desde cualquier alfabeto. La **longitud de una cadena**, es el número de posiciones ocupadas por símbolos dentro de la cadena, si w es una cadena, la longitud de w será denotada por $|w|$.

Definimos Σ^k como el conjunto de las cadenas de longitud k, tales que sus elementos pertenecen al alfabeto Σ. Al conjunto de todas las cadenas de un alfabeto Σ lo denotaremos por Σ^*. Finalmente, si Σ es un alfabeto y $L \subseteq \Sigma^*$, entonces L es un **lenguaje** de Σ. Por ejemplo:

- El lenguaje de todas las cadenas que constan de n ceros seguidos de n unos, para cualquier $0 \leq n$: $\{\varepsilon, 01, 0011, 000111\}$.
- Σ^* es un lenguaje para cualquier alfabeto Σ.

Así, siendo $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ una máquina de Turing, entonces $L(M)$ es el conjunto de cadenas w de Σ^* tales que $q_0 w \vdash^* \alpha p \beta$ para algún estado p de F y cualesquiera cadenas α y β.

Existe otro concepto de aceptación que normalmente se emplea con las maquinas de Turing, aceptación por parada. Decimos que una máquina de Turing se para si entra en un estado q, señalando a un símbolo de entrada X y $\delta(q, X)$ no está definida. Si existe un algoritmo que permite resolver un determinado problema, entonces decimos que el problema es decidible, así las máquinas de Turing que siempre se paran, representan un papel importante en la teoría de la decidibilidad.

Veamos ahora algunas extensiones de las máquinas de Turing, las cuales serán significativas en este trabajo.

La **máquina de Turing de varias cintas,** tiene una unidad de control y un número finito de cintas. cada cinta está dividida en casillas y cada casilla puede contener cualquier símbolo del alfabeto de cinta finito. Al igual que en las máquinas de Turing de una sola cinta, los símbolos de cinta incluyen al espacio en blanco y también esta formada de un subconjunto de símbolos de entrada, al cual no pertenece el espacio en blanco. El conjunto de estados incluye a un estado inicial y a varios estados de aceptación. Inicialmente:

1. Los símbolos de entrada son colocados en la primera cinta.

2. Todas las demás cintas contienen espacios en blanco.
3. La unidad de control se encuentra en el estado inicial.
4. La cabeza de la primera cinta apunta al extremo izquierdo de la entrada.
5. Las cabezas del resto de las cintas apuntan a una casilla arbitraria, pues todas las casillas están completamente en blanco.

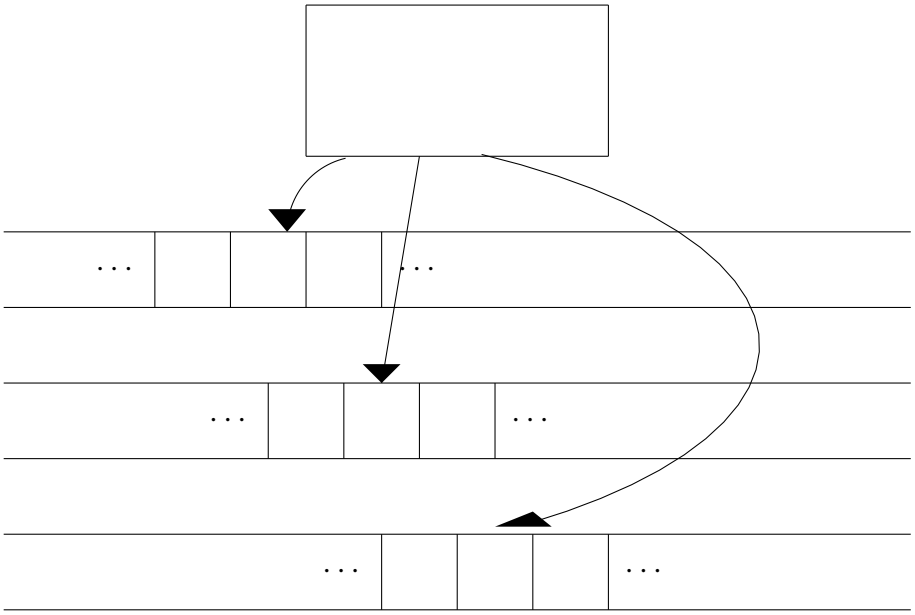


Figura 1.1: Máquina de Turing de varias cintas.

Un movimiento de la máquina de Turing de varias cintas depende del estado y del símbolo señalado por cada una de las cabezas de cada una de sus cintas. En un movimiento de la máquina de Turing de varias cintas sucede lo siguiente:

1. La unidad de control entra en un nuevo estado, puede ser el mismo en el que se encontraba anteriormente.
2. En cada cinta se escribe un nuevo símbolo de cinta en la casilla señalada por la cabeza, estos pueden ser los mismos.

3. Cada una de las cabezas de las cintas realizan un movimiento hacia la izquierda o hacia la derecha o estacionario. Las cabezas se mueven de manera independiente.

Teorema 1.3. Todo lenguaje aceptado por una máquina de Turing de varias cintas, puede ser aceptado por una máquina de Turing de una sola cinta.

Para la demostración de este teorema es conveniente pensar que la cinta de una máquina de Turing se compone de varias pistas, cada pista puede almacenar un símbolo, donde el alfabeto de la cinta consta de tuplas, con una componente para cada pista. Esta no es una extensión de la máquina de Turing, sólo se trata de ver de manera útil los símbolos de la cinta.

Demostración: Supongamos que L es un lenguaje aceptado por una máquina de Turing de k cintas. Simularemos a M con una máquina de Turing N de una sola cinta, cuya cinta consta de $2k$ pistas. La mitad de estas pistas almacenan a las cintas de M, y la otra mitad almacena, cada una de ellas, un único marcador que indica donde se encuentra actualmente la cabeza de la cinta correspondiente de M.

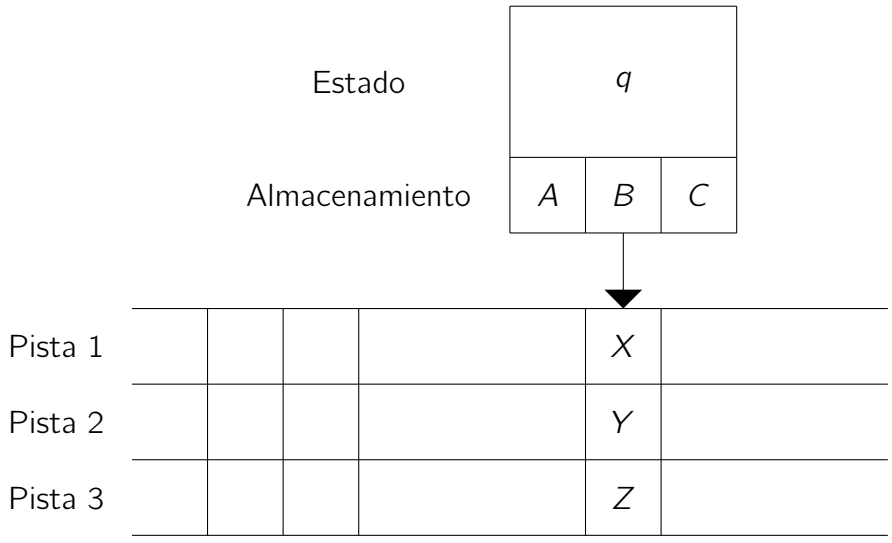


Figura 1.2: Una máquina de Turing con almacenamiento de la unidad de control y pistas múltiples.

Para simular un movimiento de M, la cabeza de N tiene que acceder a los k marcadores de cada cabeza. Después de acceder a cada marcador de cabeza y almacenar al símbolo que señalan en una componente de su unidad de control, N sabe cuáles son los símbolos señalados por cada una de las cabezas de M. N también conoce el estado de M, el cual también se almacena en la unidad de control de N. Así, N sabe cuál será el siguiente movimiento de M. Ahora N vuelve a acceder a cada uno de los marcadores de cabeza de su cinta, cambia el símbolo de la pista que representa a las cintas de M y mueve los marcadores de cabeza hacia la izquierda o derecha, si fuera necesario. Por último, N cambia el estado de M de acuerdo con lo que se haya registrado en su unidad de control. Así, N ha simulado un movimiento de M. Seleccionamos como estados de aceptación de N todos aquellos estados que se registran en el estado de M como uno de sus estados de aceptación. Así, cuando la máquina de Turing M simulada acepta, también N acepta y no aceptará en cualquier otro caso.

■

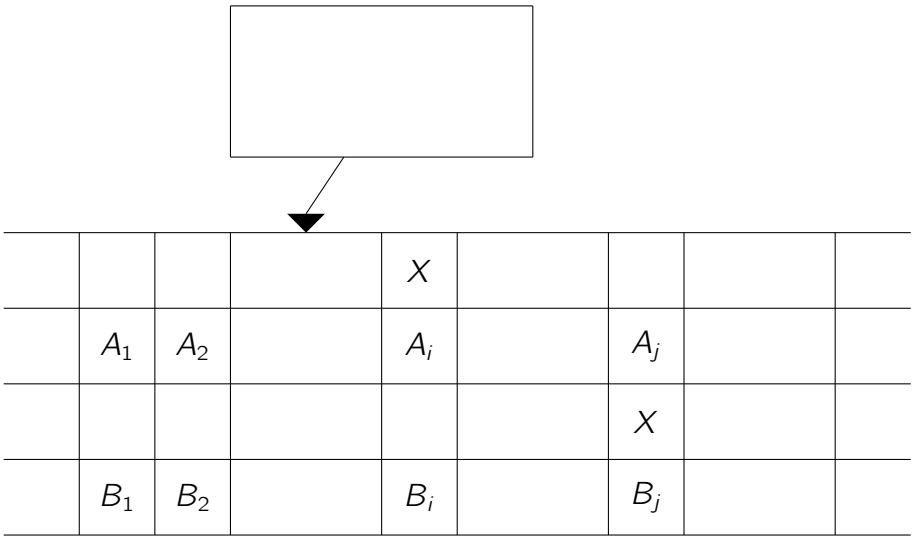


Figura 1.3: Simulación de una máquina de Turing de dos cintas mediante una máquina de Turing de una sola cinta.

Teorema 1.4. El tiempo invertido por la máquina de Turing N definida en el teorema anterior para simular n movimientos de la Máquina de Turing de k cintas de M es $O(n^2)$.

Demostración: Observemos que después de n movimientos de M, los marcadores de la cabeza de las cintas no pueden estar separados una distancia mayor a $2n$ casillas. Luego si M comienza en el marcador mas a la izquierda, no puede moverse más de $2n$ casillas hacia la derecha, para encontrar todos los marcadores de cabeza. Puede entonces realizar una excursión hacia la izquierda cambiando el contenido de las cintas simuladas de M, y moviendo los marcadores de cabeza hacia la izquierda o hacia la derecha según sea necesario. Este proceso no requiere más de $2n$ movimientos hacia la izquierda, más un máximo de $2k$ movimientos para invertir el sentido de movimiento y escribir un marcador X en la casilla situada en la derecha. Por lo tanto el número de movimietos que N necesita para simular uno de los n primeros movimientos no es mayor que $4n + 2k$. Dado que k es una constante, independiente del número de movimientos simulados, esta cantidad de movimientos es $O(n)$. Para simular n movimietos no se necesita más de n veces esta cantidad, es decir $O(n^2)$.

■

Por último, una **máquina de Turing no determinista** se diferencia de una máquina determinista en que tiene una función de transición δ tal que para cada estado q y símbolo de cinta X, $\delta(q, X)$ es un conjunto de tuplas:

$$\{(q_1, Y_1, D_1), (q_2, Y_2, D_2), ..., (q_k, Y_k, D_k)\}$$

Donde k es un número entero positivo. La máquina de Turing no determinista puede elegir en cada paso, cual de las tuplas será el siguiente movimiento. Una máquina de Turing no determinista acepta un lenguaje, si para una entrada w existe cualquier secuencia de movimientos que lleva desde la configuración inicial con w como entrada hasta una configuración con un estado de aceptación.

En el siguiente teorema veremos que las máquinas de Turing no deterministas no aceptan algún lenguaje que no sea aceptado por una máquina de Turing determinista.

Teorema 1.5. Todo lenguaje aceptado por una máquina de Turing no determinista, M_N puede ser aceptado por una máquina de Turing determinista M_D.

Demostración: Al igual que con las máquinas de turing de varias cintas, construiremos una máquina de Turing determinista, tal que la primera cinta de la Maquina de Turing determinista almacena una secuencia de configuraciones de M_N, incluyendo su estado. Para poder procesar la configuración actual, M_D hace lo siguiente:

1. M_D examina el estado y el símbolo al que señala la cabeza de la cinta de la configuración actual. La unidad de control de M_D conoce los movimientos de M_N para cada estado y símbolo. Si el estado de la configuración actual es de aceptación, entonces M_D acepta y termina la simulación.

2. Si el estado no es de aceptación y la combinación estado-símbolo da lugar a k movimientos, entonces M_D utiliza su segunda cinta para copiar la configuración y hacer a continuación k copias de dicha configuración al final de la secuencia de configuraciones de la cinta 1.

3. M_D modifica cada una de las k configuraciones de acuerdo con una de las k diferentes secuencias de movimientos que M_N puede realizar partiendo de su configuración actual.

4. M_D devuelve la configuración actual marcada, borra la marca y mueve la marca a la siguiente configuración situada a la derecha. El ciclo se repite desde el paso 1.

De esta forma M_D sólo aceptará si comprueba que M_N puede entrar en una configuración de aceptación, pero debemos confirmar que si M_N entra en un estado de aceptación después de n movimientos propios, entonces dicha configuración estará marcada en M_D y por tanto entrará también en un estado de aceptación.

Supongamos que m es el mayor número de movimientos que M_N tiene en cualquiera de sus configuraciones. Entonces existe una configuración inicial de M_N, entonces m es el máximo de configuraciones que M_N podría alcanzar después de un movimiento. Un máximo de m^2 configuraciones que M_N podría

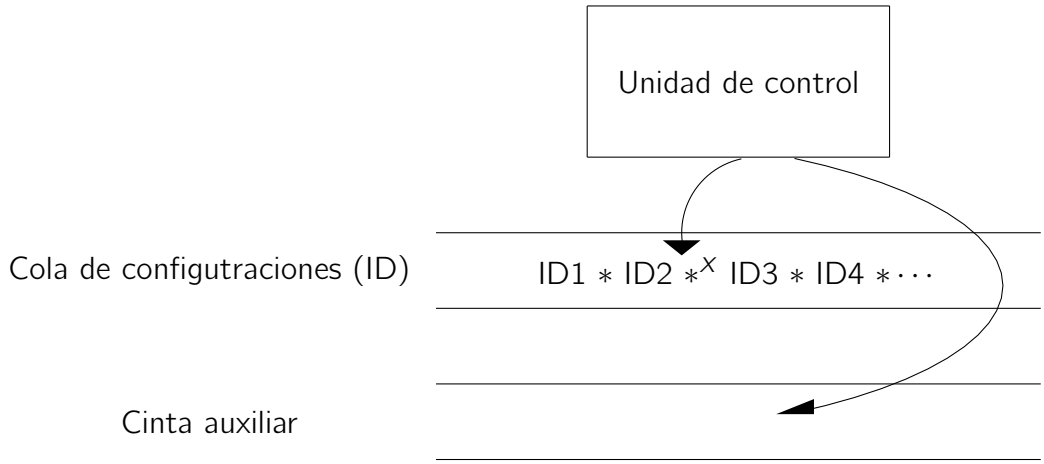


Figura 1.4: Simulación de una MTN mediante una MTD.

alcanzar después de dos movimientos, y así sucesivamente. Por lo tanto, después de n movimientos, M_N puede alcanzar un máximo de $1+m+m^2+...+m^n$ configuraciones. El orden en que M_D explora las configuraciones de M_N es la siguiente: explora todas las configuraciones alcanzables mediante cero movimientos (la configuración inicial), luego todas las configuraciones alcanzables mediante un movimiento, después aquellas que son alcanzables mediante dos movimientos, etc. En particular, M_D procesará como configuración actual toda configuración que se pueda alcanzar después de hasta n movimientos antes de considerar cualquier otra configuración que sólo sea alcanzable con más de n movimientos.

Así, Mx_D considerará la configuración de aceptación de M_N de entre las nm^n primeras configuraciones. Sólo debemos preocuparnos de que M_D considere esta configuración en un tiempo finito y este límite sea suficiente para asegurarnos de que finalmente se accederá a la configuración de aceptación. Por tanto, si M_N acepta, entonces M_D también lo hará. De esta forma, podremos asegurarnos de que todo lenguaje aceptado por una máquina de Turing no determinista, M_N puede ser aceptado por una máquina de Turing determinista M_D.

■

Debemos observar que la máquina de Turing determinista que construi-

mos, pude emplear un tiempo exponencialmente mayor que la de la máquina de Turing no determinista.

Capítulo 2

Problemas intratables

2.1. Las clases P y NP

En esta sección presentaremos las clases de problemas que se pueden resolver en tiempo polinomial mediante máquinas de Turing deterministas y no deterministas. Se dice que una máquina de Turing M tiene **complejidad temporal** $T(n)$ (o tiene tiempo de ejecución $T(n)$) si siempre que M recibe una entrada w de longitud n, M se para después de realizar como máximo $T(n)$ movimientos, sin importar si la máquina acepta o no.

Decimos que un lenguaje L pertenece a la **clase P** si existe alguna $T(n)$ polinomial tal que $L = L(M)$ para alguna máquina de Turing determinista M de complejidad temporal $T(n)$.

Veamos un ejemplo de un problema que vive en P. Para esto ocuparemos ciertos conceptos de la teoría de gráficas en los cuales ahondaremos más adelante, junto con los algoritmos DFS y BFS. Una **gráfica dirigida** o **digráfica** D, es una pareja ordenada $(V(D), A(D))$ que consta de un conjunto finito no vacío, V(D), de vértices y de un conjunto, $A(D)$, posiblemente vacío, de parejas ordenadas de elementos distintos de $V(D)$, a estos elemento les llamamos **arcos** o **flechas**. Éstas se representan por diagramas donde los arcos se dibujan como flechas, de manera que el arco (u, v) se representa por medio de una flecha del vértice u al vértice v y en este caso se dice que u **domina** a v. Un camino dirigido en una digráfica entre dos vértices u y v, es una sucesión de vértices $W = \{u = v_0, v_1, ..., v_n = v\}$ donde $(u_i, ui + 1) \in A(D)$ para toda $(0 \leq i \leq n - 1)$. Continuendo con nuestro ejemplo, definiremos el siguiente problema derivado de dar condiciones a SAT:

2-SAT Instancia: Sea U un conjunto de variables y C un conjunto de expresiones sobre U, tal que cada expresión en C tiene exactamente dos

literales.

Pregunta: ¿Existe una asignación de verdad que satisfaga C?

Teorema 2.1. 2-SAT $\in P$.

Demostración:

Sea $U = \{u_1, u_2, ..., u_n\}$ un conjunto de variables y $C = \{c_1, ..., c_m\}$ un conjunto de expresiones formando una entrada de 2-SAT. Construimos la digráfica $D = (V, E)$ donde el conjunto de vértices estará definido de la siguiente manera, $V = \{u, \overline{u} : u \in U\}$. Y existirá una flecha entre dos vértices, α y β, si y sólo si existe alguna equivalencia lógica en C de la expresión $(\neg\alpha \vee \beta)$.

Observemos que si D tiene un camino de α a β entonces también contiene un camino de $\neg\beta$ a $\neg\alpha$. Pues si $\alpha, p_1, p_2, ..., p_k, \beta$ es un camino, tenemos, por construcción, que si existe la flecha de x a y entonces hay una flecha de $\neg y$ a$\neg x$ puesto que $(\neg x \vee y) \vdash\dashv (\neg\neg y \vee \neg x)$. Por lo tanto, existen las flechas $(\neg\beta, \neg p_k), (\neg p_k, p_{k-1}\neg), ..., (\neg p_2, \neg p_1), (\neg p_1, \neg\alpha)$, formando un camino de $\neg\beta$ a $\neg\alpha$.

Afirmación 1. Si existe un vértice x en la digráfica D tal que hay un camino de x a $\neg x$ y un camino de $\neg x$ a x, entonces C es insatisfacible.

Prueba: Haremos esta prueba por contradicción. Supongamos que hay un camino de x a $\neg x$ y un camino de $\neg x$ a x para algún vértice $x \in V$ y supongamos que hay una asignación de verdad ρ que hace satisfacible a C.

Caso 1.- Si $\rho(x) =$ verdadero. Consideremos el camino $x, ..., \alpha, \beta, ..., \neg x$. Por construcción hay una flecha de α a β en D si y sólo si $(\neg\alpha \vee \beta) \in C$, esta flecha nos dice que si α es verdadera, entonces β también lo es. Como x es verdadero, entonces todas los vértices del camino de x a α (incluyendo α) deben ser verdaderas. De manera similar , todas las literales en el camino de β a $\neg x$ (incluyendo a β) deben tomar el valor de falso, ya que $\neg x$ es falso. Así, tenemos la expresión $\neg\alpha \vee \beta$ que es falsa, contradiciendo el supuesto de que existe una asignación de verdad ρ que haga satisfacieble a C.

Caso 2.- Si $\rho(x) =$ falso. La prueba es análoga al caso anterior.

Por lo tanto, si existe una variable x tal que hay un camino de x a $\neg x$ y un camino de $\neg x$ a x en la digráfica D, entonces C es insatisfacible.

□

Afirmación 2. Si C es insatisfacible, entonces existe un vértice x en la digráfica D tal que hay un camino de x a $\neg x$ y un camino de $\neg x$ a x.

Prueba: Esta prueba la haremos por contrapuesta. Supongamos que no hay caminos de x a $\neg x$ y tampoco hay caminos de $\neg x$ a x. Daremos la asignación de verdad a C utilizando el siguiente algoritmo:

1. Elegimos una literal x sin asignación, tal que no tenga un camino de x a $\neg x$ y le asignamos el valor de verdadero.

2. Asignamos el valor de verdadero a los vértices y tales que $(x, y) \in E(D)$ o $(y, x) \in E(D)$.

3. Asignamos el valor de falso a las negaciones de las variables anteriormente asignadas.

4. Repetir hasta que todas las literales tengan una asignación.

Observemos que este algoritmo está bien definido puesto que si hubiese un vértice x tal que haya un camino de x a y y un camino de x a $\neg y$, entonces hay un camino de x a $\neg y$ y un camino de $\neg y$ a $\neg x$, por la primera observación. Lo cual contradice el hecho de que no haya caminos de x a $\neg x$.

Así, existe una asignación de verdad que satisface a C.

□

La existencia de un camino de un vértice a otro está determinado por los algoritmos de búsqueda DFS o BFS, los cuales son ejecutados en tiempo polinomial. Así, 2-SAT $\in P$.

■

Por otro lado, decimos que un lenguaje L pertenece a la **clase NP** (polinomial no determinista) si existe una máquina de Turing no determinista M y una complejidad de tiempo polinomial $T(n)$ tal que $L = L(M)$, y cuando M recibe una entrada de longitud n, no existe ninguna sucesión de más de $T(n)$ movimientos de M.

2.1.1. Problemas NP-Completos

Veamos ahora una familia de problemas que pueden ser resueltos por una máquina de Turing no determinista. Decimos que L es **NP-completo** si cumple las siguientes condiciones:

1. L pertenece a NP.

2. Para todo lenguaje L' perteneciente a NP existe una reducción en tiempo polinomial de L' desde L.

Con una reducción en tiempo polinomial nos referimos a lo siguiente: En general, dados A y B dos problemas de decisión cuyas instancias requieren una respuesta del tipo "sí" o "no". Una reducción de A a B es un algoritmo polinomial R que transforma la entrada de A a una entrada equivalente de B. Esto es, dada una entrada x del problema A, R produce una entrada $R(x)$ del problema B, tal que x es una "sí" entrada de A si y sólo si $R(x)$ es una "sí" entrada de B.

Teorema 2.2. Si P_1 NP-completo y existe una reducción en tiempo polinomial desde P_1 a P_2, donde P_2 es NP, entonces P_2 es NP-completo.

Demostración: Debemos demostrar que todo lenguaje L de NP se reduce en tiempo polinomial a P_2. Sabemos que existe una reducción de L a P_1 en tiempo polinomial $p(n)$ puesto que por hipótesis P_1 es NP-completo. Por lo tanto una cadena w de L de longitud n se convierte en una cadena x de P_1 de longitud máxima $p(n)$.

También, tenemos que existe una reducción en tiempo polinomial de P_1 a P_2, supongamos que esta reducción tarda un tiempo polinomial $q(m)$. Entonces esta reducción transforma, en particular, a x en alguna cadena y de P_2, invirtiendo un tiempo máximo de $q(p(n))$. Así, la transformación de w en y tarda un tiempo máximo de $p(n)+q(p(n))$, que es polinomial. En conclusión, L es reducible en tiempo polinomial a P_2. Puesto que L puede ser cualquier lenguaje perteneciente a NP, hemos demostrado que P_2 es NP-completo.

■

2.2. Teorema de Cook

En 1971, fue que Stephen Cook demostró la existencia de un problema NP-completo, este problema es saber si, dada una expresión booleana, puede satisfacerse. Esto lo hace reduciendo el lenguaje de cualquier máquina de Turing no determinista que opera en tiempo polinomial al problema de satisfacibilidad (SAT).

Recordemos que en la sección de lógica vimos que cualquier fórmula bien formada la podemos expresar en su forma normal conjuntiva. Es decir, estar formada únicamente por una cantidad a lo más numerable de símbolos proposicionales, por el conjunto de símbolos lógicos $\{\neg, \vee, \wedge\}$ y finalmente por los símbolos auxiliares $\{(,)\}$. También definimos cuando es que una fórmula bien formada es satisfacible (existe una asignación de verdad que la satisface).

Entonces, nuestro problema de satisfacibilidad se reduce a lo siguiente:

Dada una expresión φ en su forma normal subjuntiva, ¿existe una asignación de verdad que haga a φ satisfacible?

Primero, debemos ver cómo es que representaremos a las fórmulas booleanas, para tener un alfabeto. Los símbolos que utilizaremos son los siguientes:

1. Los símbolos $\vee, \wedge, \neg, ($ y $)$ se representarán tal como están.
2. Si α_i es un símbolo proposicional se representa mediante el símbolo α seguido de ceros y unos que representan i en binario.

Así, el alfabeto del problema SAT esta compuesto únicamente por 8 elementos. Observemos que la longitud de una expresión booleana codificada es aproximadamente la misma que el número de posiciones de la expresión, contabilizando cada aparición de las letras proposicionales como 1. La razón de la diferencia está en que si la expresión tienen m posiciones, podemos tener $O(m)$ letras proposicionales, de modo que las letras proposicionales pueden emplear $O(log(m))$ para cada codificación. Por lo tanto, una expresión cuya longitud es igual a m posiciones puede tener un código de longitud $O(mlog(m))$ símbolos.

Pasemos a la demostración del teorema:

Teorema 2.3. SAT es NP-Completo.

Demostración:

Para demostrar que SAT es un problema NP-completo , necesitamos mostrar en primer lugar que está en NP.

Dada una expresión, E, utilizamos la capacidad no determinista de una MTN para conjeturar una asignación de verdad T para E. Si la E codificada tienen longitud n, entonces un tiempo $O(n)$ basta en una MTN de varias citas para esto. La MTN tiene muchas opciones de movimiento y puede tener tantas configuraciones como 2^n al final del proceso, donde cada camino representa la conjetura correspondiente a cada asignación de verdad.

Evaluamos E para la asignación de verdad T. Si T(E)=1, entonces acepta. Esta parte es determinista. El hecho de que otros caminos de la MTN no lleven a la aceptación no tiene consecuencias, ya que incluso, aunque sólo satisfaga una asignación de verdad, la MTN acepta.

La evaluación puede llevarse a cabo en un tiempo $O(n^2)$ sobre una MTN de varias cintas. Por lo tanto, el reconocimiento del problema SAT por parte de una MTN de varias cintas tarda un tiempo $O(n^2)$. La conversión a una MTN de una sola cinta puede elevar al cuadrado el tiempo, por lo que un tiempo $O(n^4)$ es suficiente en una MTN de una sola cinta. Así, SAT es un problema que está en NP.

Ahora tenemos que ver que si L es un lenguaje que pertenece a NP, entonces existe una reducción en tiempo polinomial de L a SAT. Sea MTN una máquina de Turing no determinista de una sola cinta M y un polinomio $p(n)$ tal que M no emplea más de $p(n)$ pasos para una entrada de longitud n, a lo largo de cualquier camino. Podemos restringir a MTN de tal manera que M nunca escriba un espacio en blanco y que nunca mueva su cabeza a la izquierda de la posición inicial de la misma. Por tanto, si M acepta una entrada w y $|w| = n$, entonces existe una sucesión de movimientos de M tal que:

1. α_0 es la configuración inicial de M para la entrada w.

2. $\alpha_0 \vdash \alpha_1 \vdash ... \vdash \alpha_k$ donde $k \leq p(n)$.

3. α_k es una configuración con estado de aceptación.

4. Cada α_i está formada sólo por símbolos distintos del espacio en blanco (excepto si α_i termina en un estado y en un espacio en blanco) y se

extiende desde la posición inicial de la cabeza (el símbolo de entrada más a la izquierda) hacia la derecha.

La estrategia puede resumirse como sigue:

a. Cada α_i puede escribirse como una sucesión de símbolos $X_{i0}X_{i1}...X_{i,p(n)}$. Uno de estos símbolos es un estado y los restantes son símbolos de cinta. Puesto que los estado y los símbolos de cinta son conjuntos ajenos, podemos decir que X_{ij} es un estado y en dónde se encuentra la cabeza de la cinta. No hay razón para representar los símbolos a la derecha de los primeros $p(n)$ símbolos de la cinta, porque no pueden influir en un movimiento de M, ya que garantizamos que M se para después de M movimientos.

b. Para describir la sucesión de las configuraciones en función de variables booleanas, creamos la variable y_{ijA} para representar la proposición de que $X_{ij} = A$, donde i y j son cada uno de los enteros pertenecientes al intervalo de 0 a $p(n)$, y A cualquier símbolo de cinta o estado.

c. Expresaremos la condición de que la sucesión de configuraciones representa la aceptación de una entrada w escribiendo una expresión booleana que es satisfacible si y sólo si M acepta w mediante una sucesión de, como máximo, $p(n)$ movimientos. La asignación de verdad será aquella que "diga la verdad" sobre las configuraciones; es decir, y_{ijA} será verdadera si y sólo solo si $X_{ij} = A$. Para garantizar que la reducción en tiempo polinomial de $L(M)$ a SAT es correcta, escribiremos esta expresión booleana de forma que:

   i. **Inicio correcto**. La configuración inicial es q_0w seguida por espacios en blanco.

   ii. **El siguiente movimiento es correcto**. Cada configuración subsiguiente procede de la anterior gracias a uno de los posibles movimientos válidos de M.

   iii. **Terminación correcta**. Es decir, existe alguna configuración que es un estado de aceptación.

Veamos algunos detalles que nos ayudarán en la construcción de las expresiones booleanas.

- Hemos especificado que las configuraciones terminan cuando comienza una cola infinita de espacios en blanco. Pero es más conveniente pensar que todas las configuraciones tienen la misma longitud, $p(n)+1$. Por tanto, podemos tener una cola de espacios en blanco en una configuración.

- Es mejor suponer que todos los cálculos duran exactamente $p(n)$ movimientos y por tanto tienen $p(n)+1$ configuraciones, incluso aunque la aceptación tenga lugar antes. Así conseguimos que cada configuración con un estado de aceptación sea su propio sucesor. Es decir, si α tiene un estado de aceptación, permitimos un "movimiento" $\alpha \vdash \alpha$. Por tanto, podemos suponer que si existe un cáculo de aceptación, entonces $\alpha_{p(n)}$ tendrá una configuración de aceptación y es todo lo que tendremos que comprobar para asegurar la condición "terminación correcta".

La siguiente figura muestra el aspecto de un cálculo en tiempo polinomial de M. Las filas corresponden a la sucesión de configuraciones y las columnas son las casillas de la cinta que podemos emplear para llevar a cabo el cálculo. Observemos que el número de cuadrados en la tabla es $(p(n)+1)^2$. Ademas el número de variables que representa cada cuadrado es finito, dependiendo sólo de M; es la suma de estados y de símbolos de cinta de M.

Configuración	0	1	...				...	$p(n)$
α_0	X_{00}	X_{01}						$X_{0,p(n)}$
α_1	X_{10}	X_{11}						$X_{1,p(n)}$
...	...	...	...	...	...	...	...	...
α_i				$X_{i,j-1}$	$X_{i,j}$	$X_{i,j+1}$		
α_{i+1}				$X_{i+1,j-1}$	$X_{i+1,j}$	$X_{i+1,j+1}$		
...	...	...	...	...	...	...	...	...
$\alpha_{p(n)}$	$X_{p(n),0}$	$X_{p(n)1}$						$X_{p(n),p(n)}$

Ahora proporcionamos un algoritmo para construir a partir de M y w una expresión booleana $E_{M,w}$. La forma general de $E_{M,w}$ es $U \wedge S \wedge N \wedge F$, donde S, N y F son expresiones que establecen que M se inicia, mueve y termina correctamente y U indica que sólo existe un símbolo en cada celda.

Unicidad

U es la conjunción de todos los términos de la forma

$$\neg(y_{ij\alpha} \wedge y_{ij\beta})$$

donde $\alpha \neq \beta$. Observemos que la cantidad de estos términos es $O(p^2(n))$.

Inicio correcto

X_{00} tiene que ser el estado inicial q_0 de M, X_{01} hasta X_{0n} definen w (donde n es la longitud de w) y el resto de las X_{0j}, tienen que ser espacios en blanco B. Es decir si $w = a_1 a_2 ... a_n$, entonces:

$$S = y_{00q_0} \wedge y_{01a_1} \wedge y_{02a_2} \wedge ... \wedge y_{0na_n} \wedge y_{0n+1,B} \wedge y_{0n+2,B} \wedge ... \wedge y_{0p(n),B}$$

Dada la codificasción de M y dada w, podemos escribir S en un tiempo $o(p(n))$ en una segunda cinta de la MT de varias cintas.

Terminación correcta

Como suponemos que siempre se repite una configuración de aceptación, la aceptación por parte de M es lo mismo que encontrar un estado de aceptación $\alpha_{p(n)}$. Recordemos que hemos supuesto que M es una MTN que, si acepta, lo hace en como máximo $p(n)$ pasos. Por lo tanto, F es la disyunción de las expresiones F_j, para $j = 0, 1, ..., p(n)$, donde F_j establece que $X_{p(n),j}$ es un estado de aceptación. Es decir, F_j es:

$$y_{pn,j,a_1} \vee y_{pn,j,a_2} \vee ... \vee y_{pn,j,a_k}$$

donde $a_1, a_2, ..., a_k$ son todos los estados de aceptación de M. Por tanto,

$$F = F_0 \vee F_1 \vee ... \vee F_{p(n)}$$

Observemos que cada F_i utiliza un número constante de símbolos, que depende de M, pero no de la longitud de n de la entrada w. Por lo tanto, F tiene una longitud $O(n)$. Más aún, F puede escribirse en un tiempo $O(p(n))$ en una MT de varias cintas.

El siguiente movimiento es correcto

Garantizar que el siguiente movimiento es correcto es la parte más complicada. La expresión N sera la operación lógica 'Y' aplicada a las expresiones

N_i, para $i = 0, 1, ..., p(n) - 1$ y cada N_i se diseña para garantizar que la configuración α_{i+1} es una de las configuraciones que M permita que siga a α_i. Para inicial la explicación de como escribir N_i fijémonos en la tabla anterior, que siempre podemos determinar $X_{i+1,j}$ a partir de:

1. Los tres símbolos anteriores a él: $X_{i,j-1}$, X_{ij} y $X_{i,j+1}$.
2. Si uno de estos símbolos es el estado α_i, entonces utilizamos también la elección concreta de movimiento de la MTN M.

Expresamos N_i con la operación $\wedge$ de las expresiones $A_{ij} \vee B_{ij}$, donde $j = 0, 1, ..., p(n)$.

- La expresión A_{ij} establece que:
  a. El estado de α_i está en la posición j (es decir, $X_{i,j}$ es el estado).
  b. Existe una opción de movimiento M, donde X_{ij} es el estado y $X_{i,j+1}$ es el símbolo explorado, tal que este movimiento transforma la sucesión de símbolos:

     $$X_{i,j-1}X_{i,j}X_{i,j+1}$$

     en

     $$X_{i+1,j-1}X_{i+1,j}X_{i+1,j+1}.$$

     Si $X_{i,j}$ es un estado de aceptación, existe la opción de no hacer ningún movimiento en absoluto, de modo que todas las configuraciones subsiguientes son las mismas que la que llevó en primer lugar la aceptación.

- La expresión B_{ij} establece que:
  a. El estado de α_i no está en la posición j; es decir, $X_{i,j}$ no es un estado.
  b. Si el estado de α_i no es adyacente a la posición j (es decir $X_{i,j}$ y $X_{i,j+1}$ no son estados), entonces $X_{i+1,j} = X_{i,j}$.
     Si el estado es adyacente a la posición j,entonces la corrección de la posición j será tenida en cuenta por $A_{i,j-1}$ o $A_{i,j+1}$.

B_{ij} es más fácil de escribir. Sean $q_1, q_2, ..., q_m$ estados de M y sean $Z_1, Z_2, .., Z_r$ los símbolos de cinta. Entonces:

$$\begin{aligned} B_{ij} =& (y_{i,j-1,q_1} \vee y_{i,j-1,q_2} \vee ... \vee y_{i,j-1,q_r}) \\ & \vee (y_{i,j-1,q_1} \vee y_{i,j,q_2} \vee ... \vee y_{i,j,q_r}) \\ & \vee (y_{i,j-1,Z_1} \vee y_{i,j-1,Z_2} \vee ... \vee y_{i,j-1,Z_r}) \\ & \wedge ((y_{i,j,Z_1} \wedge y_{i+1,j,Z_1}) \vee (y_{i,j,Z_2} \wedge y_{i+1,j,Z_2}) \vee ... \vee (y_{i,j,Z_r} \wedge y_{i+1,j,Z_r})). \end{aligned}$$

Las primeras dos filas de B_{ij} garantizan que B_{ij} se cumple cuando el estado α_i es adyacente a la posición j. Las tres primeras líneas garantizan que si el estado de α_i está en la posición, entonces B_{ij} es falso y la veracidad de N_i depende únicamente de que A_{ij} sea verdadero; es decir, de que el movimiento sea válido. Y cuando el estado está al menos separado dos posiciones de la posición j, las dos últimas líneas aseguran que el símbolo no debe cambiarse. Observemos que la última línea dice que $X_{i,j} = X_{i+1,j}$ enumerando todos los símbolos de cinta posibles Z y diciendo que ambos son Z_1, o ambos son Z_2, y así sucesivamente.

Existen dos casos especiales importantes: $j = 0$ o $j = p(n)$. En el primer caso no existen las variables $y_{i,j-1,X}$ y en el otro no existen las variables $y_{i,j+1,X}$. Pero, sabemos que la cabeza no se mueve hacia la izquierda de su posición inicial y sabemos que no tendrá tiempo de llegar más allá de $p(n)$ casillas a la derecha respecto de la posición en la que comenzó. Por lo tanto podemos eliminar los términos B_{i0} y $B_{i,p(n)}$.

Consideremos ahora las expresiones A_{ij}. Estas expresiones reflejan todas las relaciones posibles entre los 2×3 símbolos de la matriz de la tabla: $X_{i,j-1}, X_{i,j}, X_{i,j+1}, X_{i+1,j-1}, X_{i+1,j}$ y $X_{i+1,j+1}$. Una asignación de símbolos para cada una de estas seis variables es válida si:

1. $X_{i,j}$ es un estado, pero $X_{i,j-1}$ y $X_{i,j+1}$ son símbolos de cinta.
2. Existe un movimiento de M que explica como $X_{i,j-1}X_{i,j}X_{i,j+1}$ se transforma en:
$$X_{i+1,j-1}X_{i+1,j}X_{i+1,j+1}.$$

Existe por tanto un número finito de asignaciones de símbolos a las seis variables que son válidas. Sea A_{ij} la disyunción de varios términos, con un

término por cada conjunto de seis variables que forman una asignación. Por ejemplo, supongamos que un movimiento de M procede del hecho de que $\delta(q, A)$ contiene (p,C,L). Sea D un símbolo de cinta de M. Entonces una asignación válida es:

$$X_{i,j-1}X_{i,j}X_{i,j+1} = DqA \quad \text{y} \quad X_{i+1,j-1}X_{i+1,j}X_{i+1,j+1} = pDC.$$

Esta asignación refleja el cambio en la configuración debido al hacer este movimiento de M. El término que refleja esta posibilidad es:

$$y_{i,j-1,D} \wedge y_{i,j,q} \wedge y_{i,j+1,A} \wedge y_{i+1,j-1,D} \wedge y_{i+1,j,C} \wedge y_{i+1,j+1,p}.$$

A_{ij} es la disyunción de todos los términos válidos. También eliminamos los casos en que $j = 0$ y $j = p(n)$ para reflejar la no existencia de las variables y_{ijZ} para $j < 0$ o $j > p(n)$. Por último,

$$N_i = (A_{i0} \vee B_{i0}) \wedge (A_{i1} \vee B_{i1}) \wedge ... \wedge (A_{ip(n)} \vee B_{ip(n)})$$

y entonces: $N = N_0 \wedge N_1 \wedge ... \wedge Np(n) - 1$

Aunque A_{ij} y B_{ij} pueden ser muy grandes si M tiene muchos estados ó símbolos de cinta, su tamaño realmente es una constante siempre y cuando la longitud de la entrada w no intervenga; es decir, su tamaño es independiente de n, la longitud de w. Así, la longitud de N_i es $O(p(n))$, y la longitud de N es $O(p^2(n))$. Podemos escribir en una cinta de una MT de varias cintas en una cantidad de tiempo que es proporcional a su longitud, y dicho tiempo es polinomial en n.

Aunque hemos descrito la construcción de la expresión:

$$E_{M,w} = U \wedge S \wedge N \wedge F$$

como una función que depende tanto de M como w, el hecho es que sólo el inicio correcto S depende únicamente de w. N y F depende sólo de M y n.

Finalmente, para cualquier MTN M que opere en un tiempo polinomial $p(n)$, podemos diseñar un algoritmo que tome una entrada w de longitud n y generar $E_{M,w}$. El tiempo de ejecución de este algoritmo en una MT determinista de varias cintas es $O(p^2(n))$, y dicha MT de varias cintas puede convertirse en una MT de una sola cinta que opere en un tiempo $O(p^4(n))$. La salida de este algoritmo es una expresión booleana $E_{M,w}$ que es satisfacible si y sólo si M acepta w en como máximo $p(n)$ movimientos.

■

Finalmente, si existe una reducción en tiempo de un problema A desde un problema B NP-completo, denotado por $A \leq B$, diremos que A es B-fácil. Y si existe una reducción en tiempo de B desde A, $B \leq A$, diremos que A es B-difícil. Si A es B-fácil y B-difícil, entonces diremos que es B-equivalente. Si las reducciones fueron hechas en tiempo lineal, entonces lo denotaremos por $A \leq^{lin} B$.

Capítulo 3

Teoría de gráficas

3.1. Conceptos básicos

En esta sección daremos una pequeña introducción a la teoría de gráficas e introduciremos algunos problemas decidibles de la teoría de gráficas, con los cuales trabajaremos más adelante. Una **gráfica** consta de un conjunto no vacío V de objetos llamados vértices y un conjunto A de pares no ordenados de elementos de V, llamados aristas. Denotamos por $G = (V, A)$ a la gráfica dada por el par ordenado de los conjuntos V y A. También denotamos como $V(G)$ y $A(G)$ al conjunto de vértices y aristas de G, respectivamente. Dos gráficas G y H son **iguales** si y sólo si $V(G) = V(H)$ y $A(G) = A(H)$. En dado caso, escribimos $G = H$.

Es común representar a una gráfica con un diagrama en el plano, donde los vértices serán representados por puntos, mientras que las aristas son representadas por un segmento de linea o curva entre dos puntos en el plano correspondientes a los vértices apropiados. Por ejemplo, ver Figura 3.1, sea $G = (V, A)$ donde $V(G) = \{a, b, c, d, e\}$ y $A(G) = \{ab, ac, ad, ae, bc, bd, cd, de\}$.

Sean $u \in V(G)$ y $e \in A(G)$, decimos que el vértice u y la arista e son **incidentes** si y sólo si $u \in e$. Si $e \in A(G)$ y $d \in A(G)$ decimos que e y d son aristas **adyacentes** si y sólo si tienen un vértice en común. Dos vértices que son incidentes en una misma arista son **adyacentes** y diremos que son **vecinos** uno del otro. El conjunto de vecinos de un vértice v en una gráfica G será denotado por $N_G(v)$.

Al número de vértices de G lo llamaremos **orden**, mientras que al número de aristas lo llamaremos **tamaño** y utilizaremos la letras n y m respectivamente para referirnos a ellos. Puesto que el conjunto de vértices de una gráfica

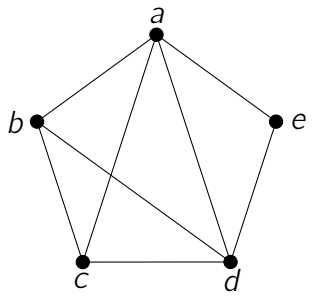


Figura 3.1: $G = \{V, A\}$.

no es vacío, el orden de toda gráfica es al menos 1. Una gráfica G es **trivial** si y sólo si $|V(G)| = 0$. Una gráfica es **finita** si y sólo si ambos conjuntos, los vértices y las aristas de la gráfica, son conjuntos finitos, en esta tesis trabajaremos con gráficas finitas, así que al referirnos a una "gráfica" nos estaremos refiriendo a una gráfica finita. Para todo $v \in V(G)$ decimos que $gr_G(v)$ es el **grado** de v donde $gr_G(v)$ es el número de aristas de G que inciden en v. En general, el subíndice G puede omitirse siempre que se tenga claro cual es la gráfica en la que se considera a v. Observemos que el grado de cada vértice está acotado por $n-1$, ya que cada vértice puede ser, a lo más, adyacente a todos los demás vértices. Es decir, $0 \leq gr_G(v) \leq n-1$ para todo $v \in G$.

Teorema 3.1. Si G es una gráfica de tamaño m, entonces,

$$2m = \sum_{v \in V(g)} gr(v).$$

Demostración: En la suma de los grados se cuenta el número de aristas que incide en cada vértice. Como cada arista tiene dos extremos, cada arista está contada dos veces en la suma.

■

Una gráfica H es una **subgráfica** de una gráfica G, lo denotamos como $H \subseteq G$, si $V(H) \subseteq V(G)$ y $E(H) \subseteq E(G)$. Si $H \subseteq G$ y $V(H)$ es un subconjunto propio de $V(G)$ o $E(H)$ es un subconjunto propio de $E(G)$, entonces H es

una **subgráfica propia** de G. Cuando se cumple que $V(H) = V(G)$ diremos que que H es una **subgráfica generadora** de G.

Una subgráfica F de una gráfica G es llamada una **subgráfica inducida** de G si para cualesquiera par de vértices, u y v, en $V(F)$ tales que $\{u, v\} \in A(G)$ entonces $\{u, v\} \in A(F)$. Si S es un conjunto de vértices, F es la **subgráfica inducida por** S de G ($F = G[S]$) si F es una subgráfica inducida de G y $V(F) = S$. Para un conjunto no vacío X de aristas, F es la **subgráfica inducida por aristas** X de G ($F = G[X]$) si $A(F) = X$ y $V(F)$ consiste de todos los vértices que son incidentes con al menos una arista en X.

Cualquier subgráfica propia de una gráfica G se puede obtener removiendo vértices y aristas de G. Para una arista e de G, escribimos $G - e$ para la subgráfica generadora de G cuyo conjunto de aristas consta de todas las aristas de G excepto e. Esto se puede generalizar para un conjunto X de aristas donde $G - X$ es la subgráfica generadora de G tal que $E(G - X) = E(G) \setminus X$.

Para un vértice v de una gráfica no trivial G, la subgráfica $G - v$ consta de todos los vértices de G excepto v y todas las aristas de G excepto por las aristas que son incidentes con v. Análogamente, esto se puede generalizar para un conjunto U de vértices de G donde la subgráfica propia $G - U$ tiene por vértices al conjunto $V(G) \setminus U$ y su conjunto de aristas consta de todas las aristas de G que no son incidentes con vértices de U.

Dos gráficas, cuyos vértices están etiquetados, G y H son **isomorfas**, $G \cong H$, si existe una función biyectiva $f : V(G) \to V(H)$ tal que $\{u, v\} \in E(G)$ si y sólo si $\{f(u), f(v)\} \in E(H)$. En este caso, f será llamado **isomorfismo** de G a H.

Varios de los conceptos de la Teoría de Gráficas corresponden a como podemos "movernos" en una gráfica. Si pensamos que los vértices de una gráfica son ciudades y las aristas como carreteras entre las ciudades, entonces la gráfica se puede considerar como un modelo de un país.

Formalmente, una sucesión de vértices es un uv-**camino** si la sucesión empieza con u y termina con v y es tal que los vértices consecutivos en la sucesión son adyacentes.

Denotamos a un uv-camino W de la siguiente forma:

$$W = (u = v_0, v_1, ..., v_k = v),$$

donde $\{v_i, v_{i+1}\}$ es una arista para $i \in \{0, 1, 2, ..., k\}$, y $k \geq 0$ es la longitud de W. Un camino que tiene longitud cero es un **camino trivial**. Podemos

observar que en nuestra definición de camino no se especifica que los vértices en la sucesión deban ser distintos, incluso puede suceder que $u = v$. Sea W un uv-camino si $u = v$, se dice que W es un **camino cerrado**. Si $u \neq v$, W es llamado **camino abierto** .

Podemos dar ciertas restricciones a los caminos como se muestra a continuación. Un uv-camino es un **uv-paseo** si en el uv-camino no se repiten aristas.

El siguiente camino no es un paseo:

$$W = (u, y, w, y, x, v),$$

puesto que la arista $\{w, y\}$ se recorre dos veces. Sin embargo, en esta definición no se excluye el caso en que un paseo repita vértices. Por ejemplo:

$$P = (u, w, y, x, w, v),$$

es un paseo en que el vértice w aparece dos veces. Así podemos sugerir una nueva restricción a nuestros caminos. Diremos que un uv-camino W es una **uv-trayectoria** si en W no se repiten vértices. Es fácil ver que toda trayectoria es un paseo.

Un **circuito** en una gráfica G es un camino cerrado de longitud 3 o más. Un circuito que no repite vértices, excepto el primero y el último, es llamado **ciclo** . Diremos que que C es un k-**ciclo** si C es de longitud k. La distancia entre dos vértices de una gráfica, u y v, es la longitud de la trayectoria mínima en G, en caso de existir, y es ∞ en otro caso; lo denotaremos por $d_G(u, v)$. A esta trayectoria también se le puede llamar **uv-geodésica**.

Se tiene un interés especial por las gráficas G en las cuales es posible viajar de un vértice a cualquier otro. Diremos que G es **conexa** si para cualquier par de vértices de G, u y v, existe una uv-trayectoria. Una gráfica que no sea conexa decimos que es **inconexa**.

Sean G un gráfica y $H \subset G$. Decimos que H es una **componente conexa** si y sólo si H es máxima por contención con respecto a la propiedad de ser conexa. Al número de componentes conexas de G lo denotaremos por $k(G)$. Así, G es conexa si y sólo si $k(G) = 1$.

Dadas H y G gráficas, $H \cup G$ es la **unión** de H y G si y sólo si $V(H \cup G) = V(H) \cup V(G)$ y $A(H \cup G) = A(H) \cup A(G)$. Así, cualquier gráfica se puede definir como la unión de sus componentes conexas.

3.1. Conceptos básicos

La teoría de gráficas abarca una enorme cantidad de conceptos, dado que es innecesario para este trabajo hablar de todos ellos, nos limitaremos a hablar de los conceptos que ocuparemos más adelante.

Veamos primero algunas gráficas que se han encontrado que cumplen ciertas características y es útil estar familiarizados con ellas.

Diremos que una gráfica es **completa** si cualesquiera dos vertices de la gráfica son adyacente. Una gráfica completa de orden n será denotada por K_n, estas son únicas salvo isomorfismo.

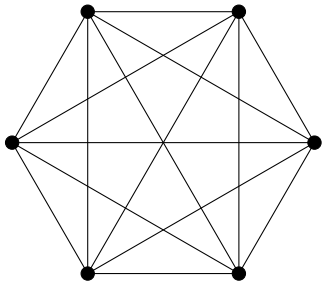


Figura 3.2: K_6

Una gráfica que cumple que todos sus vértices tengan grado r la llamaremos **r-regular** . En el caso en el que G sea 3-regular, es decir que todos sus vértices tienen grado 3, la llamaremos gráfica **cúbica.**

Estas definiciones nos dan pie a plantear nuestro primer problema decidible:

Dada una gráfica G, ¿esta contiene una subgráfica H cúbica?

Si los grados de la gráfica son todos menores a 3, es evidente que no podremos encontrar una subgráfica cúbica. En la Figura 3.3 tenemos una gráfica en la cual todos sus vértices son de grado 3, entonces la misma gráfica es una subgráfica cúbica.

Sin embargo existen gráficas donde el grado de sus vértices es mayor o igual a 3 y no necesariamente contienenen una subgráfica cúbica. Un ejemplo de ello es la Figura 3.4

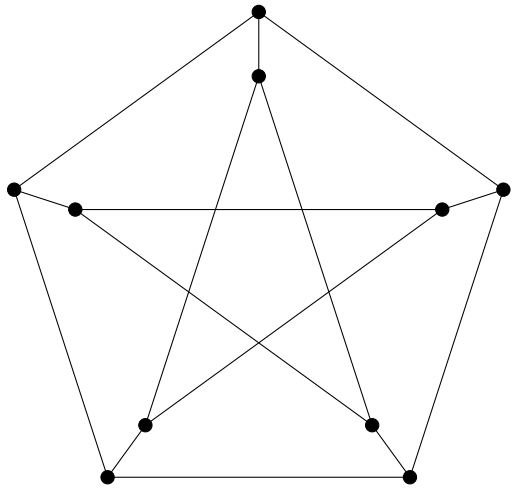


Figura 3.3: La gráfica de Petersen es 3-regular.

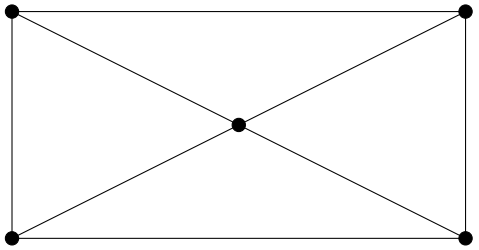


Figura 3.4: G no contiene subgráficas cúbicas.

Veamos ahora las gráficas en las que su conjunto de vértices admite una partición en conjuntos de tal manera que cumplan ciertas condiciones. Una gráfica G es **bipartita** si $V(G)$ admite una partición en dos cubconjuntos U y W, tal que toda arista de G contiene un vértice del conjunto U y otro del conjunto W.

Ahora diremos que un conjunto de aristas es **independiente** si ningún par de aristas del conjunto son adyacentes. Por un **apareamiento** en una gráfica G, nos referimos a un conjunto independiente de aristas de G.

Por ejemplo, sea G una gráfica bipartita con la partición de sus vértices en los conjuntos U y W, donde $r = |U| \leq |W|$. Un apareamiento en G es entonces un conjunto $M = \{e_1, e_2, ..., e_k\}$ de aristas, donde $e_i = u_i w_i$ para $1 \leq i \leq k$ tal

que $u_1, u_2, ..., u_k$ son k vertices distintos de U y $w_1, w_2, ..., w_k$ son k vertices distintos de W. Entonces decimos que M aparea al conjunto $\{u_1, u_2, ..., u_k\}$ con el conjunto $\{w_1, w_2, ..., w_k\}$. Observemos que se debe cumplir que $k \leq r$.

Diremos que un apareamiento es **máximo** si no está contenido en algún otro apareamiento. Si G es una gráfica de orden n, entonces un apareamiento máximo no pude exceder de $\lfloor n/2 \rfloor$. Si G es una gráfica con orden $2k$ tiene un apareamiento M de cardinalidad k, entonces este apareamiento (que es máximo) es llamado **apareamiento perfecto.** En otras palabras, un apareamiento perfecto es aque que cubre a todos los vértices de G.

Un **n-apareamiento perfecto** es una n-partición de los vértices de G en los conjuntos $V_1, V_2, ..., V_n$ de tal forma que $G[V_i]$ es un conjunto independiente de aristas y $V_1 \cup V_2 \cup ... \cup V_n$ es un apareamiento perfecto de G.

La gráfica de Petersen (Figura 3.3) es una gráfica que no contiene un 2-apareamiento perfecto. Esto nos lleva a plantear nuestro siguiente problema decidible:

Dada una gráfica G, ¿ésta contiene un 2-aparamiento perfecto?

Ahora hablaremos de ciertas propiedades con las que se puede dotar a las gráficas, primero veamos una propiedad que se usará mucho en este trabajo. Sea G una gráfica, una **coloración** de los vértices de G es una asignación de elementos de un conjunto, al los que llamaremos colores, a los vértices de G, de manera que a cada vértice le corresponda un único color. Una **coloración propia** o **buena coloración** de G es una asignación de colores a los vértices de G de manera que cualquier par de vértices vecinos tienen distinto color. Si es posible colorear G con k colores distintos de tal manera que la coloración sea propia, diremos que G es **k-coloreable**.

Dada una gráfica G, ¿ésta es k-coloreable?

En este trabajo nos interesa el caso cuando $k = 3$, la Figura 3.5 es un ejemplo de una gráfica que no es 3-coloreable.

Diremos que una gráfica es **plana** si puede ser dibujada en el plano de tal manera que sus aristas sólo se intersectan en vértices de la gráfica misma. Una gráfica G es **aplanable** si es isomorfa a una gráfica plana. Ahora podemos formular una nueva cuestión con respecto a la pregunta anterior: Dada una gráfica G aplanable, ¿G es 3-coloreable?.

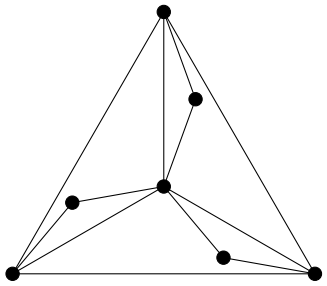


Figura 3.5: G no es 3-coloreable.

Ahora veamos algunas propiedades de las digráficas, las cuales hemos definido anteriormente. Mucha de la terminología utilizada para digráficas es muy similar a la utilizada para gráficas. Por ejemplo, dada una digráfica D, el orden de la digráfica es el cardinal del conjunto de vértices y el tamaño de D es el cardinal del conjunto de arcos de D.

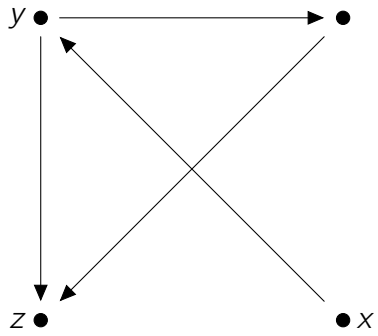


Figura 3.6: Digráfica dirigida.

Sean $D = (V(D), A(D))$ una digráfica y $v \in V(D)$. La **ex-vecindad** de v es el conjunto $N^+(v) = \{u \in V(D) : (v, u) \in A(D)\}$ a cuyos elemento se les llama **ex-vecinos** de v. El cardinal de $N^+(v)$ es el **ex-grado** de v que denotamos con $d^+(v)$. Analogamente, la **in-vecindad** de un vértice v es el conjunto $N^-(v) = \{u \in V(D) : (u, v) \in A(D)\}$ a cuyos elemento se les llama **in-vecinos** de v. El cardinal de $N^-(v)$ es el **in-grado** de v que denotamos con $d^-(v)$. Además, para todo $B \subseteq V(D)$ se define su ex-vecindad e in-vecindad como $N^+(B) = \bigcup_{v \in B} N^+(v) \backslash B$ y $N^-(B) = \bigcup_{v \in B} N^-(v) \backslash B$ respectivamente.

La **ex-vecindad cerrada** de B es su ex-vecindad junto con los elementos de B. De manera análoga se define su **in-vecindad cerrada** .

Decimos que un subconjunto, K, de vértices de una digráfica $D = (V, A)$ es un **núcleo** si para cualquier par de vértices del conjunto no existe una flecha entre ellos, es decir que es un conjunto independiente de vértices, y la in vecindad cerrada de K, es igual a $V(D)$. El ejemplo de la Figura 3.6 es una gráfica que no tiene núcleo. Si cambiamos la dirección de la flecha (x, y) a (y, x), esta nueva digráfica tendrá núcleo y estará conformado por los vértices x y z. Esto nos lleva a preguntarnos, dada una digráfica D, ¿ésta contiene un núcleo?.

Por último veremos algunos algoritmos de búsqueda en la teoría de gráficas que nos serán útiles en el siguiente capítulo. Estos algoritmos son conocidos como algoritmo de búsqueda de profundidad y algoritmo de búsqueda de amplitud (DFS y BFS por sus siglas en inglés). En resumen, estos algoritmos atraviesan toda la gráfica para encontrar cierto tipo de vértices o aristas.

Veamos primero el **algoritmo de búsqueda de profundidad**. El procedimiento en gráficas difiere en algunos aspectos del procedimiento en digráficas, así que trabajaremos por separado cada uno de ellos.

Empecemos con una gráfica no dirigida. Elegimos un vértice inicial r, al cual lo llamaremos *raiz*, para empezar la búsqueda. Después atravesaremos la arista $e = (r, v)$ para ir al vértice v. Al mismo tiempo, dirigiremos la arista e de r a v. Ahora diremos que la arista e está *examinada* y la llamaremos *rama*. Al vértice x lo llamaremos el *padre* de v y lo denotaremos por $r = padre(v)$. Continuaremos la búsqueda en general. Para un vértice x, hay dos casos:

1. Si toda arista incidente en x fue examinada, regresar al padre de x y continuar el proceso desde $padre(x)$. El vértice x se dice que fue *completamente escaneado.*

2. Si hay aristas que no han sido examinadas incidentes a x, elegimos una de ellas, digamos $e = (x, y)$ y la dirigimos de x a y. Esta arista es ahora una arista examinada. Tenemos ahora dos subcasos:

   a) Si el vértice y no fue visitado anteriormente, entonces atravesaremos la arista (x, y), visitaremos y y continuaremos la busqueda desde y. En este caso, e es una rama y $padre(y) = x$.

b) Si y a sido visitada con anterioridad, entonces seleccionaremos alguna otra arista incidente a x y a la arista e la llamaremos *arista de retroceso.*

Cada vez que lleguemos a un vértice x que no fue visitado con anterioridad, le asignaremos un número distinto que denotaremos por $DFN(x)$. El número que será asignado a la raíz es $DFN(raiz) = 1$.

La búsqueda termina cuando atravesamos la gráfica hacia la raíz y hemos visitado todos los vértices o cuando hemos hallado el vértice deseado.

DFS divide las arista de G en aristas que son ramas y aristas de retroceso. Las aristas que son ramas forman un árbol generador de G, también conocido como *árbol-DFS.*

En lo siguiente denotaremos:

$$K(x) = \begin{cases} 0 & \text{si el vértice } x \text{ no ha sido visitado} \\ 1 & \text{si el vértice } x \text{ ha sido visitado} \end{cases}$$

y $RAMA$ y $RETROCESO$ a los conjuntos que contienen a las ramas y a las aristas de retroceso respectivamente.

Algoritmo de búsqueda de profundidad para gráficas:

1. Ajustar: $RAMA \leftarrow \emptyset$, $RETROCESO \leftarrow \emptyset$, y $i \leftarrow 1$. Para todo vértice x de G, $padre(x) = \emptyset$ y $K(x) = 0$.

2. Elegir un vértice r, para el cual $K(r) = 0$ (esta condición es exclusiva para gráficas no conexas, ver paso 6). Ajustar $DNF(r) \leftarrow i$, $K(r) \leftarrow 1$ y $u \leftarrow r$.

3. Si toda arista incidente en u ha sido examinada, pasar al paso 5. En otro caso, elegir una arista $e_1 = (u, v)$ que no ha sido examinada.

4. Dirigir la arista e de u a v.

   a) Si $K(v) = 0$ entonces incrementar i en 1, ajustar $DFN(v) \leftarrow i$, añadir $e_1 \in RAMA$, $K(v) = 1$, $padre(v) = u$. Regresar al vértice u y al paso 3.

   b) Si $K(v) = 1$, entonces añadir $e_1 \in RETROCESO$ y regresar al paso 3.

5. Si $padre(u) \neq \emptyset$, entonces regresar al vértice $padre(u)$ y regresar al paso 3.

6. (Solo para gráficas no conexas.) Si hay un vértice r tal que $K(r) = 0$, entonces aumentar en 1 a i y regresar al paso 2.

7. Parar.

Denotaremos por T al árbol-DFS y $G^{\rightarrow}$ a la gráfica dirigida obtenida del algoritmo. T es al árbol dirigido generador de $G^{\rightarrow}$. Si hay una trayectoria dirigida de u a v en T, entonces llamaremos a u un *antecesor* de v y a v un *descendiente* de u. Los vértices u y v están *relacionados* si uno de ellos es un antecesor del otro. En particular, si (u, v) es una arista de T, entonces u es el padre de v y v es el hijo de u. Una arista (u, v) de G, donde u y v no están relacionados, será llamada *arista de cruce*.

Ahora veamos el algoritmo para gráfica dirigidas. El algoritmo no difiere mucho en comparación a las gráficas no dirigidas. Ahora DFS dividirá a las flechas en 4 clases diferentes. Si la búsqueda procede de una flecha no examinada $e = (x, y)$, entonces las cuatro clases posibles son:

1. Si y no ha sido visitado, entonces e es una rama.

2. Si y ha sido visitado, entonces existen tres casos:

   a) y es un descendiente de x en la subgráfica inducida por las ramas existentes. Entonces e es una *arista delantera* y $DNF(y) > DNF(x)$.

   b) x es un descendiente de y en la subgráfica inducida por las ramas existentes. Entonces e es una *arista de retroceso* y $DNF(y) < DNF(x)$.

   c) x y y no están relacionados por ninguna rama existente. Entonces e es una *arista de cruce* y $DFN(y) < DNF(x)$.

La gráfica dirigida G inducida por las ramas es llamada *bosque-DFS.*

En lo que sigue, K, $padre(x)$, $RAMA$ y $RETROCESO$ están definidos como en lo anterior. Tendremos dos nuevos conjuntos $DELANTERA$ Y $CRUCE$ para las aristas definidas con los nombres respectivos. Y finalmente:

$$L(x) = \begin{cases} 1 & \text{si el vértice } x \text{ está completamente escaneado,} \\ 0 & \text{en otro caso.} \end{cases}$$

Algoritmo de búsqueda de profundidad para digráficas:

1. Ajustar $RAMA \leftarrow \emptyset$, $DELANTERA \leftarrow \emptyset$, $RETROCESO \leftarrow \emptyset$, $CRUCE \leftarrow \emptyset$ e $i \leftarrow 1$. Para todo vértice en G, $padre(x) = \emptyset$, $K(x) = 0$ y $L(x) = 0$.

2. Elegir un vértice r (la raíz), para el cual $K(r) = 0$. Ajustar $DNF(r) \leftarrow i$, $K(r) \leftarrow 1$, sea para u un ex-vecino de r, $e = (r, u)$,añadir $e \in RAMA$. Incrementar i por 1 y ajustar $DFN(u) \leftarrow i$.

3. Si toda flecha que sale de u ha sido examinada, entonces ajustamos $L(u) \leftarrow 1$ y pasar al paso 5. En otro caso, elegir una arista no examinada $e_1 = (u, v)$.

4. Determinar:

   a) Si $K(v) = 0$ entonces: aumentar en 1 a i, fijar $DNF(v) \leftarrow i$, añadir $e \in RAMA$, $K(v) = 1$, ajustar $padre(v) \leftarrow u$ y regresar al vértice u. Pasar nuevamente al paso 3.

   b) Si $K(v) = 1$ y $DNF(v) > DNF(u)$, entonces añadir $e \in DELANTERA$. Regresar al paso 3.

   c) Si $K(v) = 1$ y $DNF(v) < DNF(u)$ y $L(v) = 0$, entonces añadir $e \in RETROCESO$. Regresar al paso 3.

   d) Si $K(v) = 1$ y $DNF(v) < DNF(u)$ y $L(v) = 1$, entonces añadir $e \in CRUCE$. Regresar al paso 3.

5. Si $padre(u) \neq \emptyset$, entonces regresar a vértice $padre(u)$ y regresar al paso 3.

6. Si hay un vértice r tal que $K(r) = 0$, entonces aumentar i en 1 y regresar al paso 2.

7. Parar.

3.1. Conceptos básicos

Pasemos ahora a analizar el **algoritmo de búsqueda de amplitud.** Al igual que con DFS, analizaremos los casos para gráficas y digráficas.

Algoritmo de búsqueda de amplitud en gráficas:

1. En principio, ningún vértice está etiquetado. Fijar $i \leftarrow 0$.
2. Elegir un vértice, no etiquetado, r (raíz) y etiquetar con i.
3. Buscar el conjunto J de vértices que no están etiquetados y son adyacentes a algún vértice etiquetado con i.
4. Si $J \neq \emptyset$, entonces aumentar i en 1. Etiquetar los vértices de J con i y pasar al paso 3.
5. (Unicamente para gráficas no conexas.) Si un vértice no está etiquetado, entonces fijar $i = 0$ y seguir al paso 2.
6. Parar.

BFS también forma un árbol generador llamado *árbol-BFS*, cuando tomamos las aristas

(vértice etiquetado con i, vértice no etiquetado)

mientras se forma J.

Algoritmo de búsqueda de amplitud en digráficas:

1. En principio, ningún vértice está etiquetado. Fijar $i \leftarrow 0$.
2. Elegir un vértice, no etiquetado, r (raíz) y etiquetar con i.
3. Buscar el conjunto J de vértices terminales de flechas cuyos vértices iniciales han sido etiquetados con i pero que sus vértices terminales no han sido etiquetados.
4. Si $J \neq \emptyset$, entonces aumentar i en 1. Etiquetar los vértices de J con i y pasar al paso 3.
5. Si un vértice no está etiquetado, entonces fijar $i = 0$ y seguir al paso 2.
6. Parar.

BFS en digráficas forma un *bosque-BFS*, cuando tomamos las aristas

(vértice etiquetado con i, vértice no etiquetado)

mientras se forma J.

Observemos que en los dos algoritmos cada vértice se examina una vez y todas las aristas son examinadas, así el tiempo que utilizan estos algoritmos es $O(\text{máx}\{|V|, |A|\})$.

Capítulo 4

Problemas de Teoría de Gráficas NP-completos

4.1. 3-coloración

Para probar que ciertos problemas de la Teoría de Gráficas son NP-completos, necesitamos ver que el problema de la 3-coloración es NP-Completo, para más adelante, reducir dichos problemas desde 3-coloración.

Para esto definiremos algunos problemas derivados de dar condiciones a SAT, los cuales nos ayudarán como pasos intermedios para ver que 3-coloración es SAT-equivalente:

3-SAT Instancia: Sea U un conjunto de variables y C un conjunto de expresiones sobre U, tal que cada expresión en C tiene exactamente tres literales.

Pregunta: ¿Existe una asignación de verdad que satisfaga C?

4-SAT Instancia: Sea U un conjunto de variables y C un conjunto de expresiones sobre U, tal que cada expresión en C tiene exactamente cuatro literales.

Pregunta: ¿Existe una asignación de verdad que satisfaga C?

NE5SAT Instancia: Sea U un conjunto de variables y C un conjunto de expresiones sobre U, tal que cada expresión en C tiene exactamente 5 literales.

Pregunta: ¿Existe una asignación de verdad que satisfaga C tal que cada expresión tenga una literal verdadera y otra falsa?

Teorema 4.1. 3-Coloracion $\leq^{lin}$ SAT

Demostración: Supongamos que tenemos una entrada para 3-coloración, esto es una gráfica $G = (V, E)$. Debemos construir una conjunción de expresiones proposicionales en tiempo lineal, tal que esta conjunción sea satisfacible si y sólo si, la gráfica G es 3-coloreable. Consideremos el siguiente conjunto de variables proposicionales:

$$U = \{p_x{}^0, p_x{}^1, p_x{}^2 : x \in V\}.$$

Donde, $p_x{}^i$ será verdadero si y sólo si $x \in V_i$ (donde V_i es el conjunto de vértices de color i). La fórmula F como entrada de SAT correspondiente a la gráfica G es la conjunción de las siguientes dos fórmulas:

$$F_1 = \bigwedge_{x \in V} \bigvee_{\{i,j,k\}=\{0,1,2\}} p_x{}^i \wedge \neg(p_x{}^j \vee p_x{}^k)$$

$$F_2 = \bigwedge_{\{x,y\} \in E} \bigwedge_{i=0}^{2} \neg(p_x{}^i \wedge p_y{}^i).$$

La primera fórmula nos dice que cada vértice tendrá un solo color, mientras que la segunda nos asegura que por cada arista de E, sus vértices extremos serán de distinto color. Así, es fácil ver que G es 3-coloreable si y sólo si F es satisfacible.

Por último, debemos verificar que la reducción se hace en tiempo lineal:

- Para obtener F_1 sólo se reescribió tres veces cada $x \in V$ como $p_x{}^0, p_x{}^1, p_x{}^2$
- Para obtener F_2 sólo se reescribió tres veces cada $x, y \in E$ como $\neg(p_x{}^i \wedge p_y{}^i)$

Entonces, la longitud de la conjunción de F_1 y F_2 es lineal en la longitud de G, así es como una máquina de Turing puede construir en tiempo lineal F desde G.

■

Lema 4.2. SAT $\leq$ 3-SAT.

Prueba: Sea $U = \{u_1, u_2, ..., u_n\}$ un conjunto de variables y $C = \{C_1, C_2, ..., C_m\}$ un conjunto de expresiones formando un estado en SAT. Construiremos una colección C' de expresiones con exactamente tres literales sobre U' un conjunto de variables tal que C' es satisfacible si y sólo si C es satisfacible. La construcción de C' se limita a sustituir cada expresión $C_j \in C$ por una expresión equivalente C'_j que tiene exactamente 3 literales basado en el conjunto U original y añadiendo algunas variables U'_j. Así, formamos:

$$U' = U \cup \Big(\bigcup_{j=1}^{m} U'_j\Big) \quad \text{y} \quad C' = \bigcup_{j=1}^{m} C'_j$$

Esto sólo lo necesitamos para mostrar como C'_j y U'_j se pueden construir desde C_j. Sea C_j dado por $\{z_1, z_2, ..., z_k\}$ donde z_i es la literal derivada de las variables en U. La forma en que cada C'_j y U'_j son formadas depende del valor de k.

- Caso 1. $k = 1$. $U'_j = \{{v_j}^1, {v_j}^2\}$,
  $C'_j = \{\{z_1, {v_j}^1, {v_j}^2\}, \{z_1, {v_j}^1, \overline{v}_j{}^2\}, \{z_1, \overline{v}_j{}^1, {v_j}^2\}, \{z_1, \overline{v}_j{}^1, \overline{v}_j{}^2\}\}$.
- Caso 2. $k = 2$. $U'_j = \{{v_j}^1\}$,
  $C'_j = \{\{z_1, z_2, {v_j}^1\}, \{z_1, z_2, \overline{v}_j{}^1\}\}$.
- Caso 3. $k = 3$. $U'_j = \varnothing$, $C'_j = \{\{c_j\}\}$.
- Caso 4. $3 < k$. $U'_j = \{{v_j}^i : 1 \leq i \leq k-3\}$,
  $C'_j = \{\{z_1, z_2, {v_j}^1\}\} \cup \{\{\overline{v}_j{}^i, z_{i+2}, \overline{v}_j{}^{i+1}\} : 1 \leq i \leq k-4\} \cup \{\{\overline{v}_j{}^{k-3}, z_{k-1}, z_k\}\}$.

Veamos que esta transformación cumple que C' es satisfacible si y sólo si C es satisfacible. Primero supongamos que $s : U \to \{F, V\}$ es una asignación de verdad que satisface C. Probaremos que s se puede extender a una asignación de verdad $s' : U' \to \{F, V\}$ que satisfaga C'. Puesto que las variables en $U' - U$ están divididas en los conjuntos U'_j y puesto que las variables en cada U'_j ocurren únicamente en las expresiones que perteneces a C'_j, necesitamos probar como s puede ser extendida a los conjuntos U'_j uno por uno, y en cada caso sólo necesitamos verificar que todas las expresiones en la correspondiente C'_j se satisface. Podemos hacer esto como sigue:

- Si U'_j se construyó como en el caso 1 o 2, entonces las expresiones en C'_j ya son satisfechas por t, así que podemos extender a s arbitrariamente en U'_j como $s'(v) = V$ para toda $v \in U'_j$.

- Si U'_j fue construido como en el caso 3, entonces U'_j es vacío y la única expresión en C'_j ya se satisface bajo s.

- El último caso, en que $3 < k$, puesto que s es una asignación de verdad que satisface C, tiene que haber al menos un entero l tal que en la literal z_l es verdadera bajo s. Si l es 1 o 2, entonces tendremos que $s'({v_j}^i) = F$ para $1 \leq i \leq k-3$. Si l es menos que $k-1$ o k, entonces $s'({v_j}^i) = V$ para $1 \leq i \leq k-3$. En el otro caso tendremos que $s'({v_j}^i) = V$ para $1 \leq i \leq l-2$ y $s'({v_j}^i) = F$ para $l-1 \leq i \leq k-3$.

En los tres casos se cumple que si C es satisfacible entonces C' es satisfacible por como lo construimos. Ahora, si s' es una asignación de verdad para C' que lo satisface, tomamos la restricción de s' para las variables en U que satisfacen a C. Así, C es satisfacible si y sólo si C' es satisfacible. El número de expresiones con tres literales en C' está acotado por un polinomio en mn. Por tanto, nuestra transformación es polinomial.

□

Teorema 4.3. SAT $\leq$ 3-Coloración.

Demostración: Haremos esta reducción usando las siguientes transformaciones:

SAT $\rightarrow$ 3-SAT $\rightarrow$ 4-SAT $\rightarrow$ NE5SAT $\rightarrow$ 3-COL

La primera transformación la hemos demostrado anteriormente en el Lema 4.2.

Pasemos a la segunda transformación, sea $C = \{C_1, ..., C_p\}$ con $C_i = \{{l_i}^0 \vee {l_i}^1 \vee {l_i}^2\}$ un conjunto de p 3-expresiones definidas en un conjunto de variables U dado como una entrada de 3-SAT. La correspondiente entrada para 4-SAT es el conjunto de $2p$ 4-expresiones $C' = \{C'_1, C''_1, ..., C'_p, C''_p\}$ definida sobre $U' = U \cup \{v\}$ donde v es una nueva variable con $C'_i = \{{l_i}^0 \vee {l_i}^1 \vee {l_i}^2 \vee v\}$ y $C''_i = \{{l_i}^0 \vee {l_i}^1 \vee {l_i}^2 \vee \overline{v}\}$. Si existe una asignación de verdad que satisfaga C es claro que se puede extender a una asignación de verdad, sin importar la asignación de v, que satisfaga a C'. Además, si hay una asignación de verdad que satisfaga a C', podemos restringir la asignación de verdad y escoger entre C'_i o C''_i para que se satisfaga a C. Así, existe una asignación de verdad que satisface a C si y sólo si existe una asignación de verdad que satisface a C'.

Realicemos la tercera transformación. Sea $C = \{C_1, C_2, ..., C_p\}$ con $C_i = \{l_i^0 \vee l_i^1 \vee l_i^2 \vee l_i^3\}$ un conjunto de p 4-expresiones definidas en un conjunto de variables U dado como una entrada de 4-SAT. La correspondiente entrada para NE5SAT es el conjunto de p 5-expresiones $C = \{C'_1, C'_2, ..., C'_p\}$ definidas sobre $U' = U \cup \{f\}$ donde f es una nueva variable con $C'_i = \{l_i^0 \vee l_i^1 \vee l_i^2 \vee f\}$. Supongamos que existe una asignación de verdad $s : U \to \{F, V\}$ que satisfaga a C, podemos extenderla a una asignación de verdad s' de tal forma que $s'(f) = F$. Así, si C es satisfacible bajo s entonces existe s' que hace satisfacible a C'. Por otro lado, si s' es una asignación de verdad que satisfaga a C' tal que $s'(f) = F$ tomamos la restricción de s' a U. Por otro lado, si $s'(f) = V$ tomamos la restricción de la negación de s' para garantizar que existirá al menos una literal l_i^n que sea verdadera. Por tanto, C es satisfacible si y sólo si existe una asignación de verdad de C' tal que cada C'_i tenga una literal verdadera y otra falsa.

Finalmente, sea $C = \{C_1, ..., C_p\}$ con $C_i = \{l_i^0 \vee l_i^1 \vee l_i^2 \vee l_i^3 \vee l_i^4\}$ un conjunto de 5-expresiones definidas sobre U un conjunto de variables, dados como una entrada de NE5SAT. La correspondente entrada de 3-coloreabilidad es la gráfica $G = (V, E)$ formada como sigue:

El conjunto de vértices es $V = V_1 \cup V_2$, donde:

$$V_1 = \{control\} \cup \{v, \overline{v} : v \in U\},$$

$$V_2 = \{s_i(l_i^j) : 1 \leq i \leq p, 0 \leq j \leq 4\},$$

El conjunto de aristas es $E = E_1 \cup E_2$, donde:

$$E_1 = \{\{control, v\}, \{control, \overline{v}\}, \{v, \overline{v}\} : v \in U\},$$

$$\begin{aligned} E_2 =& \{\{l_i^j, s_i(l_i^j)\} : 1 \leq i \leq p, 0 \leq j \leq 4\} \\ & \cup \{\{s_i(l_i^j), s_i(l_i^k)\} : 1 \leq i \leq p, 0 \leq j \leq 4, k = (j+1) (\text{mod } 5)\} \end{aligned}$$

El ciclo $\{\{s_i(l_i^j), s_i(l_i^k)\} : 1 \leq i \leq p, 0 \leq j \leq 4, k = (j+1) (\text{mod } 5)\}$ representará la expresión C_i y será denotado por CT_i. Observemos que estos ciclos son impares. Sea t una asignación de verdad para U tal que cada expresión en C tenga al menos una literal falsa y una literal verdadera. Sin

pérdida de generalidad, supongamos que $t(l_i^0) = 0$ y $t(l_i^1) = 1$. Definiremos la 3-coloración $col : V \to \{0, 1, 2\}$ de G como sigue: $col(control) = 2$, $col(v) = t(v)$ y $col(\overline{v}) = 1 - t(v)$ para $v \in U$. Según nuestra suposición, $t(l_i^0) = 0$ y $t(l_i^1) = 1$, entonces tendremos que: $col(s_i(l_i^0)) = 1$, $col(s_i(l_i^1)) = 0$, $col(s_i(l_i^2)) = 2$, $col(s_i(l_i^3)) = 1 - col(l_i^3)$, $col(s_i(l_i^4)) = 2$. Así, por como construimos G, col define una 3-coloración válida para G.

Ahora, supongamos que $col : V \to \{0, 1, 2\}$ es una 3-coloración válida para G. Sin pérdida de generalidad supongamos que $col(control) = 2$. Como $v, \overline{v}, control$ forman un 3-ciclo, entonces cada variable $v \in U$ tenemos que $col(v) \in \{0, 1\}$ y $col(\overline{v}) = 1 - col(v)$. Esto nos define una asignación de verdad t en U de forma que $t(v) = col(v)$ por cada $v \in U$. Puesto que el ciclo CT_i es de longitud impar entonces los tres colores son usados para colorear sus vértices. Entonces, existen en particular dos literales l_i^j y l_i^k de C_i tal que $t(s(l_i^j)) = 0$ y $t(s(l_i^k)) = 1$ y así tenemos que $col((l_i^j)) = 1$ y $col((l_i^k)) = 0$ lo cual nos dice que $t(l_i^j) = 1$ y $t(l_i^k) = 0$. Entonces, t es una asignación de verdad que cumple que cada expresión de C tenga una literal falsa y una literal verdadera.

Observemos que nuestra primera transformación, 4.2, no es una transformación que se haya hecho en tiempo lineal, y a pesar que el resto de las transformaciones si lo sean, nuestra demostración indica que una máquina de turing lo podrá hacer en un tiempo polinomial.

Así, SAT $\leq$ 3-Coloración.

■

De esta forma, hemos demostrado que 3-Coloración es SAT-equivalente.

4.2. Reducciones Lineales

El propósito de este trabajo es mostrar ejemplos de como hacer las reducciones en problemas de la teoría de gráficas. Veremos ejemplos de problemas SAT-fácil y SAT-difícil, todas estas reducciones las haremos en tiempo lineal, centrándonos sobre todo en los problemas SAT-difícil y al final veremos un ejemplo de una reducción hecha en tiempo polinomial.

4.2.1. Problemas SAT-fácil

Proposición 4.4. Núcleo de una gráfica $\leq^{lin}$ SAT.

Demostración: Sea $G = (V, A)$ un gráfica dirigida. Construiremos una fórmula F tal que F es satisfacible si y sólo si existe un conjunto $S \subset V$ que es el núcleo de G. Observemos que el problema del Núcleo de una gráfica, es decidible, en el sentido no determinista, eligiendo el conjunto de vértices S, y verificando que ningún vértice en S tenga un predecesor en S y cualquier otro vértice que no está en S tiene un predecesor en S.

Consideremos el conjunto de letras proposicionales $U = \{p_v : v \in V\}$ como una entrada de SAT. Intuitivamente, la variable p_v es verdadera si y sólo si v es un vértice de S.

Sea F la fórmula:

$$F = \bigwedge_{v \in V} \left(\left(p_v \rightarrow \bigwedge_{(u,v) \in A} \neg p_u \right) \wedge \left(\neg p_v \rightarrow \bigvee_{(u,v) \in A} p_u \right) \right)$$

es decir, si $v \in S$ entonces para cualquier flecha $(u, v) \in A$, el vértice u no pertenece a S, además existe una u en S tal que $(u, v) \in A$.

Es claro que F es satisfacible si y sólo si G tiene un núcleo. Observemos que la longitud de F es lineal con respecto a la longitud de G, ya que F es una copia del conjunto de vértices y de los predecesores de estos.

■

4.2.2. Problemas SAT-difícil

Proposición 4.5. 3-Coloración $\leq^{lin}$ Núcleo de una Gráfica.

Demostración:

Dada una gráfica $G = (V, E)$ construiremos una gráfica dirigida $G' = (V', F')$ que tendrá un núcleo si y sólo si G es 3-colorable. La describiremos en dos pasos.

Primero, para cada x en V definimos la subgráfica G_x con vértices:

$$cp^x, \quad t^x, \quad r^x, \quad c_0(x), \quad c_1(x), \quad c_2(x)$$

y donde el conjunto de flechas es la unión de los siguientes tres conjuntos:

$$\{(cp^x, r^x); (r^x, t^x); (t^x, cp^x)\} \quad , \quad \{(c_k(x), c_l(x)) : (l, k) \in \{0, 1, 2\}^2\}$$

$$\text{y} \quad \{(r^x, c_k(x)) : k \in \{0, 1, 2\}\}$$

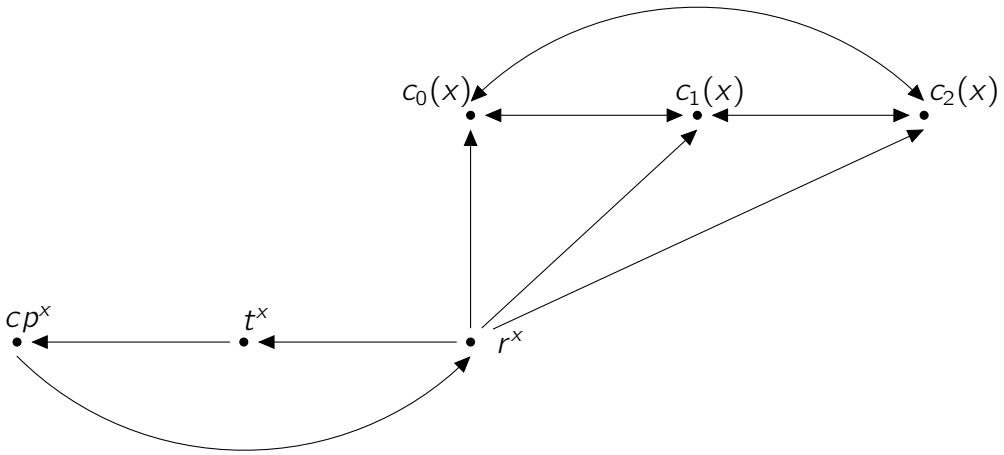


Figura 4.1: Gadget x.

Cada vértice $c_k(x)$ para $k = 0, 1, 2$ corresponde intuitivamente al predicado "el color del vértice x es k". Los tres vértices cp^x, t^x y r^x se utilizan para garantizar que al menos uno de los predicados anteriores es cierto. Segundo, uniremos todas las subgráficas G_x de la siguiente manera: si $\{x, y\} \in E$, entonces G' contiene las seis flechas de la forma $(c_k(x), c_k(y))$ y $(c_k(y), c_k(x))$ para $k = 0, 1, 2$. Esto completa la descripción de $G' = (V', F')$.

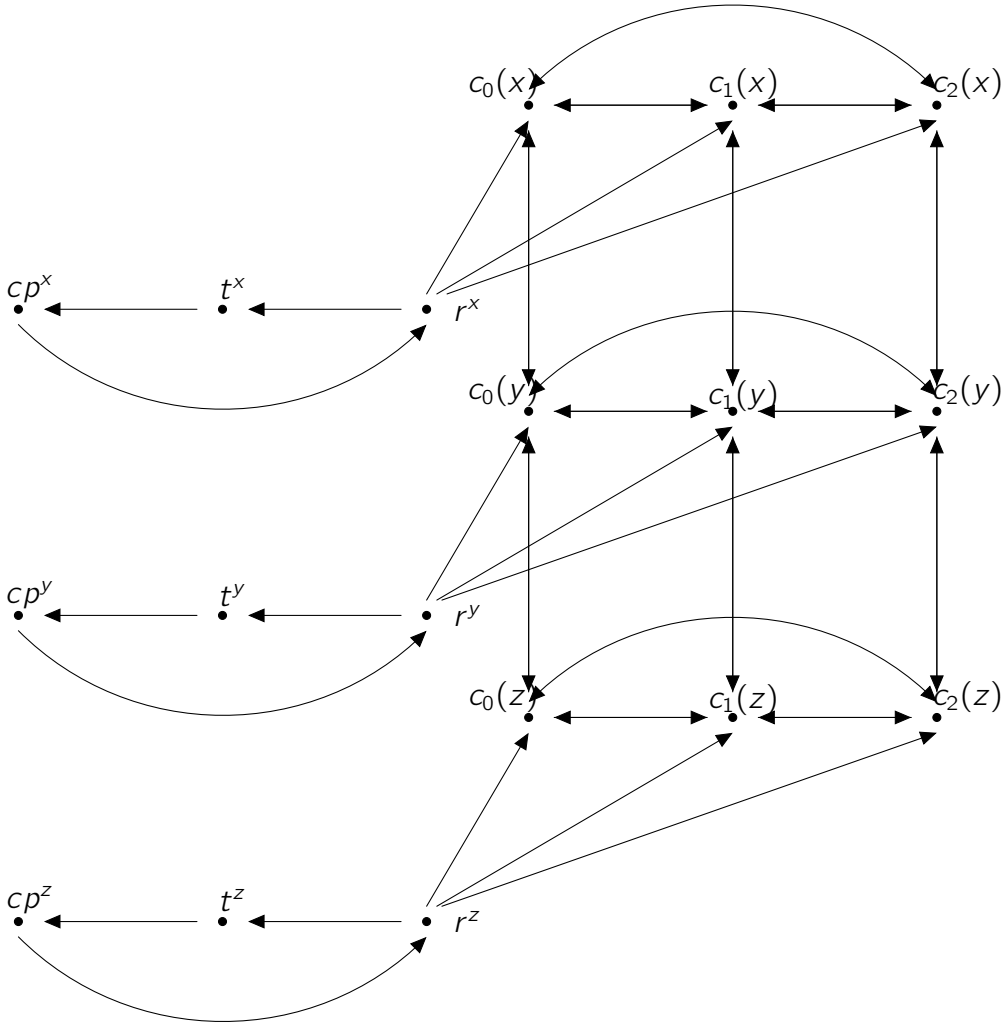


Figura 4.2: G'.

En este ejemplo nos muestra que G' se obtiene de una gráfica $G = (V, E)$ con $V = \{x, y, z\}$ y $E = \{\{x, y\}, \{y, z\}\}$. Si el orden y tamaño de G es n y t respectivamente, entonces el orden de G' es $6n$ y el tamaño es $12n + 6t$.

Afirmación 3. Si G' tiene un núcleo W' entonces G es 3-coloreable.

Prueba: Supongamos que G' tiene un núcleo W', entonces W' cumple las

siguientes propiedades presentadas en la siguiente afirmación:

Afirmación 4. Para todo x en V tenemos que:

- cp^x pertenece a W'
- existe una única $k \in \{0, 1, 3\}$ tal que $c_k(x)$ pertenece a W'

Prueba: Sea x un vértice de V. Primero mostraremos que $cp^x \in W'$. Supongamos que esto no es cierto. Entonces $cp^x \in V' \backslash W'$. Pero el único vértice que lo absorbe es r^x, por lo tanto $r^x \in W'$. Por otra parte, t^x sólo es absorbido por cp^x entonces $t^x \in W'$. Pero $(r^x, t^x) \in F'$, entonces tendríamos dos vértices en W' que son adyacentes, lo cual es una contradicción. Por tanto $cp^x \in W'$.

Ahora, veamos que existe una única k tal que $c_k(x) \in W'$. Puesto que $cp^x \in W'$, $(cp^x, r^x) \in F'$ y $(t^x, cp^x) \in F'$, tenemos que ni r^x, ni t^x pertenecen a W'. Entonces, $r^x \in V' \backslash W'$. Pero los únicos vértices absorbentes de r^x son t^x y $c_k(x)$ para $k = 0, 1, 2$, por tanto alguno de ellos debe pertenecer a W'. Pero ya vimos que $t(x) \notin W'$, por tanto al menos un vértice c_k pertenece a W. Más aún, $(c_k, c_l) \in F'$ para cualquier $k \neq l$. Por tanto, si $c_k(x) \in W'$ entonces $c_l(x) \notin W'$.

□

Por la Afirmación 4 podemos definir una 3-coloración de G de la siguiente manera:

$$col : x \mapsto col(x) = k \quad \text{si y sólo si} \quad c_k(x) \in W'.$$

Esta función está bien definida por la Afirmación 3. Sólo nos falta mostrar que esta es una buena 3-coloración para G. Sean x y y dos vértices distintos de G tales que $\{x, y\} \in E$. Entonces existen k y l tales que $c_k(x) \in W'$ y $c_l(y) \in W'$. Dado que W' es un conjunto independiente $(c_k(x), c_l(y)) \notin F'$. Pero $(c_k(x), c_k(y)) \in F'$ para cada $k \in \{0, 1, 2\}$ puesto que $\{x, y\} \in E$. Entonces, podemos deducir que $k \neq l$. Pero, por definición, $col(x) = k$ y $col(y) = l$. Así, hemos mostrado que para cualesquiera dos vértices adyacentes en G estos son de diferente color.

□

Afirmación 5. Si G es 3-coloreable entonces G' tiene un núcleo W'.

Prueba:

Sea $col : V \to \{0, 1, 2\}$ una 3-coloración válida para G. Definimos W' por:

$$W' = \{cp^x, c_k(x) : x \in V, k = col(x)\}.$$

Primero observemos que para cada $(x, y) \in V \times V$, la flecha $(c_k(x), c_l(y))$ pertenece a A' si y sólo si $k = l$ y $\{x, y\} \in E$. Observemos que cualesquiera dos vértices en W' no son adyacentes por construcción y por lo mencionado anteriormente, por tanto W' es un conjunto independiente. Veamos que cada vértice que no está en W' es absorbido por alguno que sí lo esté. Para cada $x \in V$ tal que $k = col(x)$, el vértice t^x es el predecesor de cp^x, y el vértice r^x junto con el vértice y $c_l(x)$ son absorbidos por $c_k(x)$ para $l \neq k$. Por tanto, todo vértice en $V'\backslash W'$ es absorbido por algún vértice en W'.

□

■

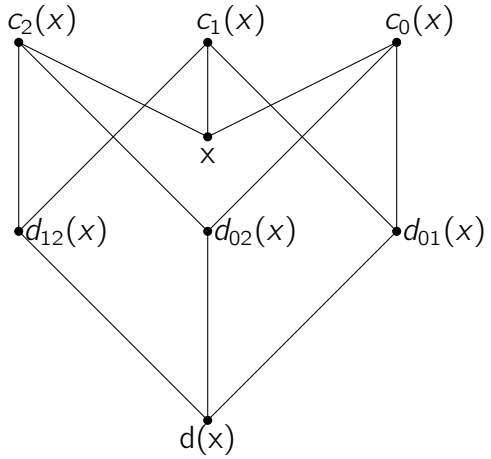


Figura 4.3: Gadget x

Proposición 4.6. 3-coloración $\leq^{lin}$ 2-partición en apareamientos perfectos.

Demostración: Dada una gráfica $G = (V, E)$, mostraremos como construir en tiempo lineal, una gráfica $G' = (V', E')$ en dónde podemos encontrar una partición en dos apareamientos perfectos si y sólo si G es 3-coloreable. Para construir G' necesitaremos dos tipos de gráficas, una por cada vértice y otra por cada adyacencia, necesitaremos también dos vértices de control llamados: **control-link** y **big-control**.
Así, definimos para cada vértice $x \in V$ el gadget $G_x = (V_x, E_x)$, independientemente del número de incidencias en E, de la siguiente manera:

El conjunto de vértices:

$$V_x = \{x, d(x), c_0(x), c_1(x), c_2(x), d_{01}(x), d_{01}(x), d_{12}(x)\}.$$

El conjunto de aristas:

$$\begin{aligned} E_x =& \{\{x, c_k(x)\} : k \in \{0,1,2\}\} \cup \{\{d_{01}(x), c_0(x)\}, \{d_{01}(x), c_1(x)\}\} \\ & \cup \{\{d_{0,2}(x), c_0(x)\}, \{d_{02}(x), c_2(x), \}\} \\ & \cup \{\{d_{12}(x), c_1(x)\}, \{d_{12}(x), c_2(x)\}\} \\ & \cup \{\{d(x), d_{ij}(x)\} : \{i,j\} \subset \{0,1,2\} \text{ y } i < j\} \end{aligned}$$

Ahora definimos la arista-gráfica $G_{xy} = (V_{xy}, E_{xy})$ para cada $\{x, y\} \in E$. El conjunto de vértices es:

$$V_{x,y} = \{c_k(x,y), c'_k(x,y), d_k(x,y) : k \in \{0,1,2\}\} \cup \{\alpha(x,y), \alpha'(x,y), \beta(x,y)\}.$$

Su conjunto de aristas es, para $k \in \{0,1,2\}$:

$$\begin{aligned} E_{x,y} =& \{\{c_k(x,y), c'_k(x,y)\}, \{c_k(x,y), d_k(x,y)\}, \{c'_k(xy), d_k(x,y)\}\} \\ & \cup \{\{\alpha(x,y), c_k(x,y)\}, \{\alpha'(x,y), c'_k(x,y)\}, \{\beta(x,y), d_k(x,y)\}\}. \end{aligned}$$

Finalmente, para terminar con la descripción de G' debemos unir estas gráficas y el **control** de la siguiente manera. El conjunto de aristas de control que conecta todos los gadgets con la estructura control definida como sigue:

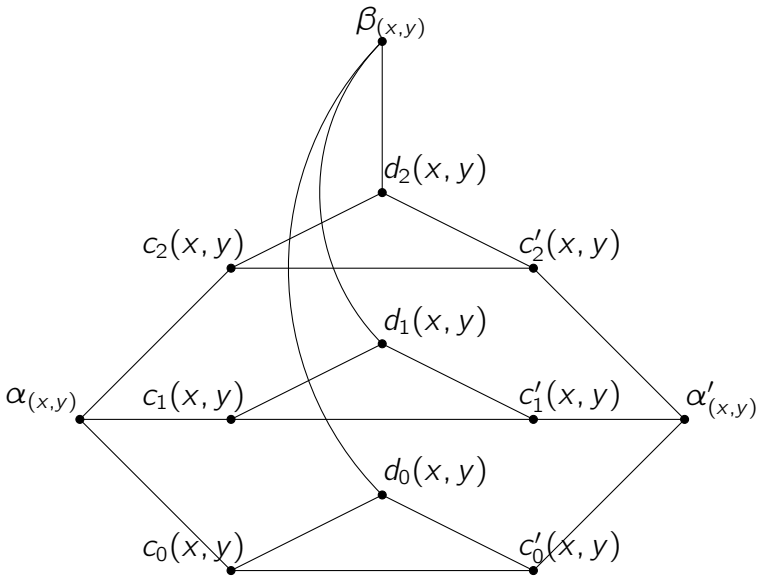


Figura 4.4: Arista-Gráfica.

$$\{\{\text{control-link, big-control}\}, \{\{x, \text{control-link}\} : x \in V\}\}.$$

Y para cada arista $\{x, y\} \in E$ unimos G_x y G_y a $G_{x,y}$ con las siguientes aristas:

$$\{\{c_k(x),\ c_k(x,y)\},\ \{c_k(y),\ c_k'(xy)\} : k \in \{0,1,2\}\}.$$

Afirmación 6. Si G es 3-coloreable entonces G' puede ser dividida en dos apareamientos perfectos.

Prueba:

Sea $col : V \to \{0,1,2\}$ una 3-coloración válida para G. Entonces, definimos la partición de G' en dos apareamientos perfectos considerando los dos siguientes subconjuntos de V (con $\{h,k,l\} = \{0,1,2\}$):

$$\begin{aligned} V_1' =& \{x, d(x), c_k(x), d_{hl}(x) : x \in V, k = col(x)\} \\ & \cup \{c_h(x,y), c_h'(x,y), d_k(x,y), c_k'(x,y), d_l(x,y), c_l(x,y) : \\ & \{x,y\} \in E, k = col(x), l = col(y)\}, \end{aligned}$$

$V_2' = \{c_h(x), c_l(x), d_{kl}(x), d_{kh}(x) : x \in V, k = col(x)\}$
$\cup \{d_h(x,y), c_k(x,y), c_l'(x,y) : \{x,y\} \in E, k = col(x), l = col(y)\}$
$\cup \{\alpha(x,y), \alpha'(x,y), \beta(x,y) : \{x,y\} \in E\}$
$\cup \{\text{control-link, big-control}\}$.

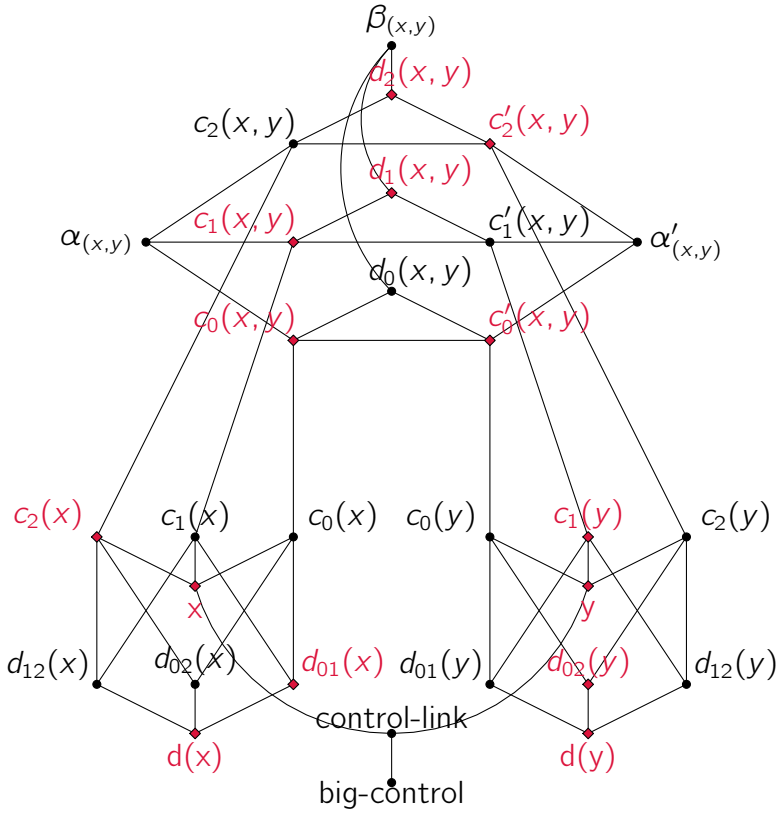


Figura 4.5: Apareamiento perfecto dado por $col(x) = 2$ y $col(y) = 1$

Notemos que si G es de orden n y tamaño m, entonces $|V'| = 8n+12m+2$ y $|E'| = 12n + 18m + 6m + n + 1$.

Observemos que $V_1' \cup V_2'$ es una partición de V puesto que a cada vértice le corresponde un color y así nuestros conjuntos están bien definidos. Por como construimos los apareamientos, estos serán perfectos.

□

Afirmación 7. Si G' puede ser dividida en dos apareamientos perfectos entonces G es 3-coloreable

Prueba: Sea $V' = V_1' \cup V_2'$ una partición de los vértices de G' tal que las subgráficas G_1' y G_2', inducidas respectivamente por V_1' y V_2' son apareamientos perfectos. Sean $E'|_{V_1'}$ y $E'|_{V_2'}$ denotados por E_1' y E_2' respectivamente, consideramos $G_1' = (V_1', E_1')$ y $G_2' = (V_1', E_2')$. La prueba de este lema necesita de otros dos lemas que a continuación mostraremos. Supondremos a partir de ahora, sin pérdida de generalidad, que big-control pertenece al conjunto de vértices V_2'

Afirmación 8. Sea x un vértice de V. Entonces, $x \in V_1'$ y existe una única $k \in \{0, 1, 2\}$ tal que $c_k(x) \in V_1'$.

Prueba: Por hipótesis, big-control pertenece a V_2'. Pero, su único vértice adyacente es control-link. Entonces, control-link también pertenece a V_2' puesto que este induce un apareamiento perfecto. Además, cada $x \in V$ es vecino de control-link entonces $x \in V_1'$ y los únicos vecinos de este son $c_k(x)$ para $k \in \{0, 1, 2\}$. Por tanto, uno y sólo uno de estos vértices pertenece a V_1'.

□

Afirmación 9. Cualquier arista $\{c_k, c_k(x, y)\}$ de E' tiene al menos uno de sus vértices finales en V_1' y el otro en V_2'.

Prueba:

Sea $\{c_k, c_k(x, y)\}$ una arista de E' (con $\{x, y\} \in E$). Tenemos dos casos, de acuerdo a qué subconjunto contiene a $c_k(x)$. Si $c_k(x) \in V_1'$ (que por la afirmación anterior sabemos que existe una única k que lo cumple) entonces x es el vecino de $c_k(x)$ en G_1' puesto que $x \in V_1'$. Por tanto, el vértice $c_k(x, y)$ pertenece a V_2'.
Si $c_k(x) \in V_2'$, se sigue del lema anterior que existe $l \neq k$ tal que $c_l(x) \in V_1'$. El vértice x es vecino de $c_l(x)$ en G_1'. Entonces, los otros vecinos de $c_l(x)$ en particular $d_{kl}(x)$ pertenecen a V_2'. Así, V_2' contiene a $c_k(x)$ y a $d_{kl}(x)$. Entonces, los otros vecinos de $c_k(x)$ pertenecen a V_1', en particular $c_k(x, y) \in V_1'$. Esto completa la prueba.

□

Ahora, definamos la 3-coloración $col : V \to \{0, 1, 2\}$ de G por:

$$col(x) = k \text{ si y sólo si } V_1' \text{ contiene a ambos, } x \text{ y } c_k(x).$$

Esta función está bien definida por el Afirmación 8. Ahora veamos que es una coloración válida para G. Supongamos que existe $\{x, y\} \in E$ tal que $col(x) = col(y) = k$, entonces $c_k(x)$ y $c_k(y)$ pertenecen a V_1'. Por otra parte, si $l \neq k$ entonces los vértices $c_l(x)$ y $c_l(y)$ pertenecen a V_2' (pues $y \in V_1'$ y $x \in V_1'$). De este modo, usando el lema 1.9, podemos observar que V_2' contiene a $c_k(x, y)$ y $c_k'(x, y)$ y por otro lado V_1' contiene a los vértices $c_l(x, y)$ y $c_l'(x, y)$ para $l \neq k$. En consecuencia, los vecinos de $c_k(x, y)$ distintos de $c_k'(x, y)$ pertenecen a V_1', en particular $\alpha(x, y) \in V_1'$. De la misma manera, los vecinos de $c_l(x, y)$ diferentes de $c_l'(x, y)$ pertenecen a V_2. Entonces $\alpha(x, y) \in V_2'$ lo cual es una contradicción.

□

Finalmente la función propuesta puede ser computada en tiempo lineal. Así, 3-coloración $\leq^{lin}$ 2-partición en apareamientos perfectos.

■

Ahora mostraremos que probar el problema de encontrar una subgráfica cúbica es SAT-difícil. La reducción desde 3-colorabilidad en este problema es más difícil que en los problemas anteriores. Sea $G = (V, E)$ una gráfica 3-coloreable. Construiremos una vértice-gráfica por cada ocurrencia en E de cada vértice de V.

Primero mostraremos algunos resultados que nos serán útiles para nuestra demostración.

Teorema 4.7. Toda gráfica $G = (V, E)$ de tamaño m tiene un número par de vértices de grado impar.

Prueba: Diremos que $d(s)$ es el grado de $s \in V$. Dividimos a los vértices en dos subconjuntos V_1 y V_2, donde V_1 contendrá a todos los vértices de grado impar y V_2 contiene a todos los vértices de grado par. Sabemos que para toda gráfica de tamaño m se cumple que:

$$\sum_{s \in V} d(s) = 2m.$$

entonces, tenemos que:

$$\sum_{s\in V} d(s) = \sum_{s\in V_1} d(s) + \sum_{s\in V_2} d(s) = 2m.$$

Puesto que $\sum_{s\in V_2} d(s)$ es una suma de números pares, entonces es un número par, además:

$$\sum_{s\in V_1} d(s) = 2m - \sum_{s\in V_2} d(s).$$

Por lo tanto, $\sum_{s\in V_1} d(s)$ es un número par.

□

Ahora, observemos que si $G = (V, E)$ es 3-coloreable, entonces podemos tomarnos por pares a los vértices de grado impar, y por cada par agregamos otro vértice que sea adyacente a ese par. Llamemos G_1 a la gráfica obtenida por añadir estos vértices y las respectivas aristas. Observemos que G_1 seguirá siendo una gráfica 3-coloreable, pues en el peor de los casos, el vértice añadido será adyacente a vértices de distinto color, pero aún podemos asignar el color restante al nuevo vértice y G_1 será 3-coloreable.

Proposición 4.8. 3-coloración $\leq^{lin}$ Subgráfica cúbica.

Demostración:

Mostraremos que una subgráfica cúbica puede ser obtenida por una 3-coloración. Dada una gráfica $G = (V, E)$, construiremos en varios pasos una gráfica $G' = (V', E')$ que tenga una subgráfica cúbica si y sólo si G es 3-coloreable. La idea es construir una gráfica en la que la mayoría de sus vértices sean de grado 3.

Primero supongamos que V está ordenado de manera lexicográfica. Asociamos dos vértices, $cp(x)$ y $cI(x)$, para cada x en V. Estos vértices serán unidos uno al otro para formar la estructura de control.

Ahora, analizando a E numeraremos el número de ocurrencias de cada vértice. Por ejemplo, si $V = \{w, x, y, z\}$ y $E = \{\{x, y\}, \{x, z\}, \{x, w\}, \{y, z\}\}$ entonces numeramos de la siguiente forma a los vértices $E = \{\{x_0, y_0\}, \{x_1, z_0\}, \{x_2, w_0\}, \{y_1, z_1\}\}$. Por cada ocurrencia de cada vértice construiremos una vértice-gráfica. En nuestro ejemplo debemos construir 8 vértice-gráficas: G_{x_0},

G_{x_1}, G_{x_2}, G_{y_0}, G_{y_1}, G_{z_0}, G_{z_1} y G_{w_0}. Por razones técnicas será útil tener un número par de ocurrencias en E por cada vértice x en V. Si este no es el caso, añadiremos un nuevo vértice a G, como describimos antes, por cada pareja de vértices de grado impar, el cual tendrá grado 2. Así, obtendremos un número par de ocurrencias, para cada vértice. Ahora daremos la construcción detallada de las vértice-gráficas, las arista-gráficas y la estructura de control. Sea $\{x_i : 0 \leq i \leq 2n\}$ todas las ocurrencias en E de un vértice $x \in V$. Definimos la vértice-gráfica $G_{x_i} = (V_{x_i}, E_{x_i})$ como sigue:

El conjunto de vértices será:

$$\begin{aligned} V_{x_i} =& \{x_i,\ c(x_i),\ c_0(x_i),\ c_1(x_i),\ c_2(x_i),\\ & c'(x_i),\ c'_0(x_i),\ c'_1(x_i),\ c'_2(x_i),\ b_0(x_i),\ b_1(x_i),\ b_2(x_i)\}. \end{aligned}$$

Y el conjunto de aristas será:

$$\begin{aligned} E_{x_i} =& \{\ \{x_i, c(x_i)\},\ \{x_i, c'_(x_i)\},\ \{c(x_i),\ c'(x_i)\}\ \}\\ & \cup \quad \{\{c(x_i), c_k(x_i)\},\ \{c'(x_i), c'_k(x_i)\} : k \in \{0,1,2\}\\ & \cup \quad \{\{b_k(x_i), c_k(x_i)\},\ \{b_k(x_i), c'_k(x_i)\} : k \in \{0,1,2\}\}. \end{aligned}$$

Intuitivamente, las aristas $\{c(x_i), c_0(x_i)\}$ y $\{c'(x_i), c'_0(x_i)\}$ corresponden al predicado "x_i tiene el color 0". Introducimos los vértices $c'(x_i)$ y $b_k(x_i)$, que son sólo copias de $c(x_i)$, para obtener la mayor cantidad de vértices de grado 3. El rol de los vértices $c_k(x_i)$, $c'_k(x_i)$ es diferente. Por un lado los vértices $c_k(x_i)$ aseguran que todas las ocurrencias del mismo vértice están coloreadas de la misma manera. Mientras que los vértices de la forma $c'_k(x_i)$ nos aseguran que dos vértices adyacentes no tendrán el mismo color.

Ahora definamos las arista-gráficas. Sea $\{x_i, y_j\}$ una arista de E. La arista-gráfica $G_{x_i,y_j} = (V_{x_i,y_j}, E_{x_i,y_j})$ esta formada de la siguiente forma:

El conjunto de vértices será:

$$\begin{aligned} V_{x_i,y_j} =& \{arc(x_i, y_j), link(x_i, y_j), test1(x_i, y_j), test2(x_i, y_j),\\ & c_0(x_i, y_j), c_1(x_i, y_j), c_2(x_i, y_j)\}. \end{aligned}$$

Y el conjunto de aristas estará dado por:

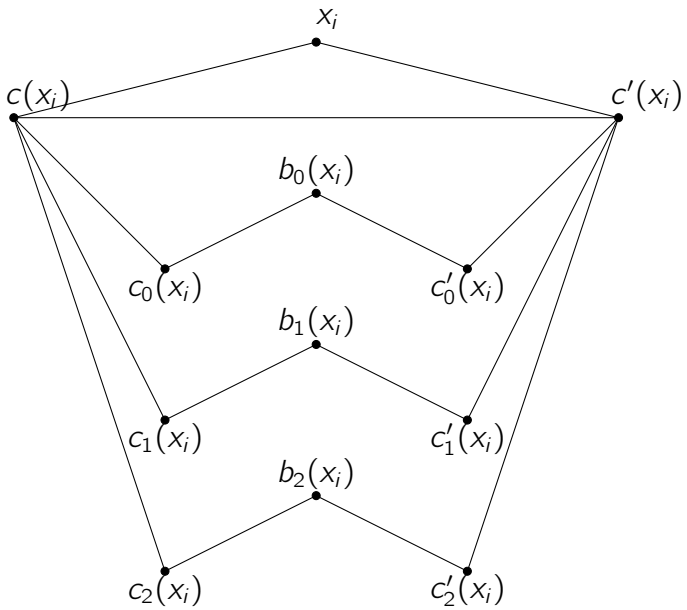


Figura 4.6: Vértice -gráfica G_{xi}.

$$
\begin{aligned}
E_{x_i,y_j} = &\{\{arc(x_i,y_j), link(x_i,y_j)\}\} \\
\cup \quad &\{\{link(x_i,y_j), test1(x_i,y_j)\},\ \{link(x_i,y_j), test2(x_i,y_j)\}\} \\
\cup \quad &\{\{test1(x_i,y_j), c_k(x_i,y_j)\} : 0 \leq k \leq 2\} \\
\cup \quad &\{\{test2(x_i,y_j), c_k(x_i,y_j) : 0 \leq k \leq 2\}\}.
\end{aligned}
$$

Informalmente, los vértices $c_k(x_i, y_j)$ serán útiles para definir un color para cada vértice final de la arista $\{x_i, y_j\}$. Más aún, cada vértice $c_k(x_i, y_j)$ nos asegurará que los dos vértices de la arista $\{x_i, y_j\}$ no tengan el mismo color. Finalmente, supongamos que $p(x)$ denota al predecesor de $x \in V$ en el orden lexicográfico y $s(x)$ al sucesor (si a es el primer elemento de V y z el último, entonces diremos que $p(a) = z$ y que s(z)=a). Entonces las aristas de la estructura de control estará definida por:

$$\{\{cp(x), cl(x)\},\ \{cl(x), cl(p(x))\},\ \{cl(x), cl(s(x))\}\}.$$

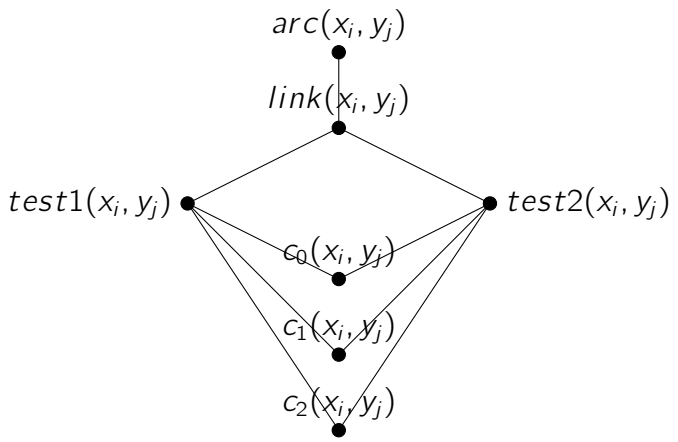


Figura 4.7: Arista-gráfica.

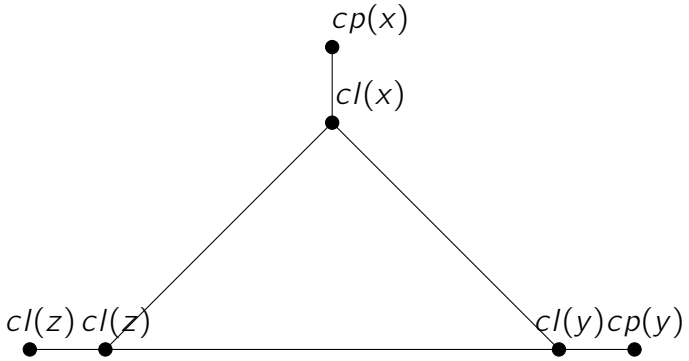


Figura 4.8: Estructura de control.

En la Figura 4.8 podemos notar que $cl(x)$ tiene exactamente grado 3. Ahora, para terminar la descripción de la gráfica G' uniremos las gráficas descritas anteriormente.
Primero, para cada $x \in V$ uniremos las vértices-gráficas correspondientes a las ocurrencias de x de la siguiente forma:

$\{\{c_k(x_0), c_k(x_1)\}, \{b_k(x_1), b_k(x_2)\}, \ldots,$
$\{c_k(x_{2i-1}), c_k(x_{2i})\}, \{b_k(x_{2i}), b_k(x_{2i+1})\}, \ldots,$
$\{c_k(x_{2n-1}), c_k(x_{2n})\} : 0 \leq k \leq 2; x \in V$ con 2n ocurrencias$\}$.

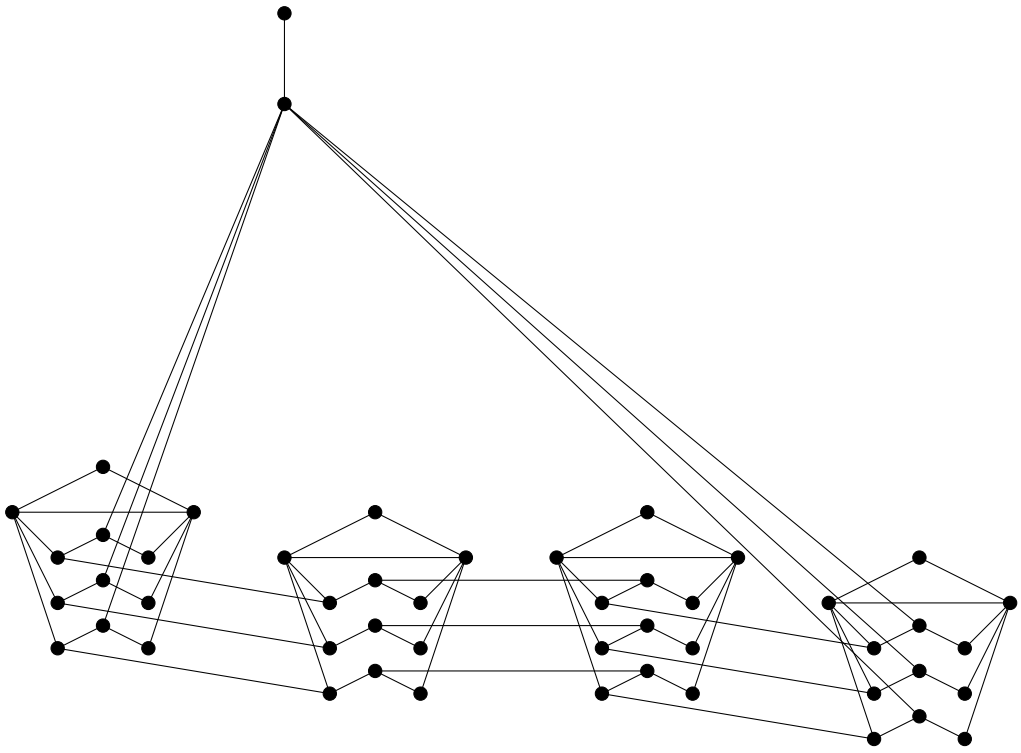


Figura 4.9: Unión a la estructura de control.

Ya que unimos las vértices-gráficas de cada vértice respectivamente, agregaremos las siguientes aristas para unirlas a la estructura de control. El conjunto de aristas está definido para cada $x \in V$ por:

$$\{\{b_k(x_1), cp(x)\}, \{b_k(x_{2n}), cp(x)\} : k \in \{0, 1, 2\}\}$$

Observemos que los vértices considerados anteriormente tienen grado 3, excepto por $cp(x)$. Los vértices $b_k(x_i)$ han sido añadidos únicamente para ob-

tener gráficas G_{x_i} que puedan ser unidas sin exceder del grado 3. Necesitamos un número par de estas por la misma razón.

Por otro lado, las aristas que necesitamos para unir cada arista-gráfica G_{x_i,y_j} a las dos vértice-gráficas correspondientes G_{x_i} y G_{y_j}, están definidas como sigue. Para cada arista $\{x_i, y_j\} \in E$ unimos G_{x_i} a G_{x_i,y_j}, con cuatro aristas: $\{x_i, arc\{x_i, y_j\}\}$ y $\{c'_k(x_i), c_k(x_i, y_j)\}$ para $k \in \{0, 1, 2\}$; y de manera similar unimos G_{y_j} a G_{x_i,y_j}.

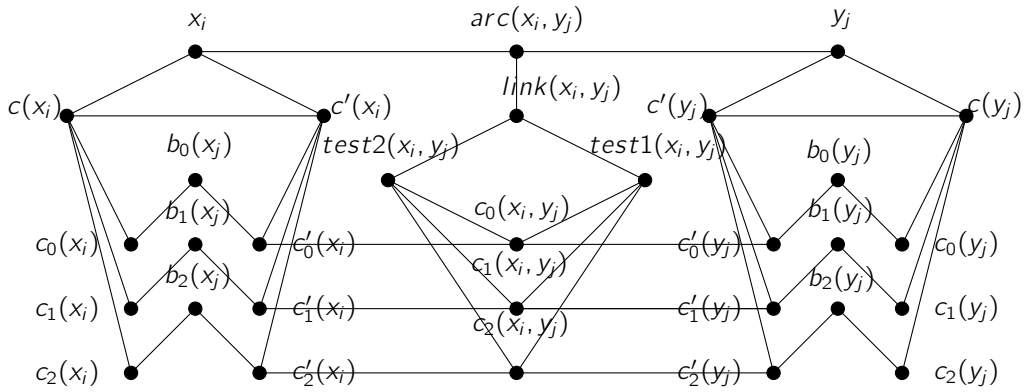


Figura 4.10: Unión de arista-gráfica G_{x_i,y_j} a las sus vértice-gráficas correspondientes G_{x_i} y G_{y_j}.

Observemos que los vértices x_i, y_j, $arc(x_i, y_j)$, $c'_k(x_i)$, $c'_k(y_j)$ (con $k \in \{0, 1, 2\}$) tienen grado 3, y los vértices $c_k(x_i, y_j)$,(con $k \in \{0, 1, 2\}$), $test1(x_i, y_j)$ y $test2(x_i, y_j)$ tienen grado 4.

Sea $G'_1 = (V'_1, E'_1)$ una subgráfica cúbica de G'. Veamos que esta subgráfica cumple con ciertas propiedades, independientemente de la estructura de G'.

Afirmación 10. Sea $G'_1 = (V'_1, E'_1)$ una subgráfica cúbica de G'. Por cada s en V, $d(s)$ denota al grado de s en G' y $d'(s)$ el grado en G'_1.

1. Si $d(s) = 3$ entonces, $d'(s) = 3$ si y sólo si E'_1 contiene a todas las aristas que incluyen a s.

2. Si $d(s) = 3$ y $\{s, t\}$ es una arista de E' entonces, $d'(t) = 0$ implica que $d'(s) = 0$.

3. Si $d(s) = d(t) = 3$ y $\{s, t\}$ es una arista de E' entonces, $d'(t) = 0$ si y sólo si $d'(s) = 0$.

4. Si hay un camino $s_1, s_2, ..., s_n$ en G' tal que $d(s_i) = 3$ para cada $i \in \{1, 2, ..., n\}$, entonces, $d'(s_1) \neq 0$ si y sólo si $d'(s_n) \neq 0$.

Prueba: Demostraremos cada uno de los incisos:

1. Primero supongamos que $d'(s) = 3$, puesto que $d(s) = 3$ entonces, en G'_1, s tiene el mismo número de ocurrencias que en G', por tanto E'_1 contiene a todas las aristas que incluyen a s.

   Ahora supongamos que E'_1 contiene a las aristas que contienen a s puesto que $d(s) = 3$ entonces $d'(s) = 3$.

2. Puesto que $\{s, t\}$ es una arista de E' y $d'(t) = 0$, entonces la arista $\{s, t\}$ no pertenece a las aristas de E'_1, así $d'(s) \leq 2$, como G'_1 es una gráfica cúbica, entonces $d'(s) = 0$.

3. Primero supongamos que $d'(s) = 0$, como $d(t) = 3$ y $\{s, t\}$ es una arista de E' entonces, por el inciso anterior, $d'(t) = 0$. Análogamente para la implicación restante.

4. En este caso, podemos utilizar los incisos anteriores para los vértices s_i y s_{i+1} con $i \in \{1, 2, ..., n-1\}$. Así, concluimos que $d'(s_1) \neq 0$ si y sólo si $d'(s_n) \neq 0$.

□

Afirmación 11. Si G es 3-coloreable, entonces G' tiene una subgráfica cúbica.

Prueba: Supongamos que G es 3-coloreable y sea $col : V \to \{0, 1, 2\}$ una coloración válida. Construiremos la subráfica cúbica G'_1 de la siguiente manera.
Para cada $a = \{x_i, y_j\}$ en E, si $col(x) = k$ y $col(y) = l$, entonces los vértices de G'_1 son:

x_i, y_j, $cp(x)$, $cp(y)$, $cl(x)$, $cl(y)$, $c(x_i)$, $c'(x_i)$, $b_k(x_i)$, $c_k(x_i)$, $c'_k(x_i)$, $c(y_j)$, $c'(y_j)$, $b_l(y_j)$, $c_l(y_j)$, $c'_l(y_j)$ $arc(a)$, $link(a)$ $test1(a)$, $test2(a)$ $c_k(a)$, $c_l(a)$.

□

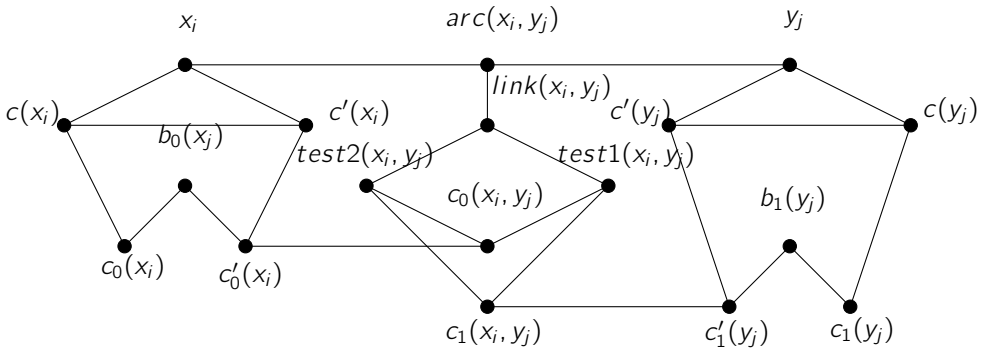


Figura 4.11: Subgráfica obtenida al tener $col(x) = 0$ y $col(y) = 1$.

Para completar la demostración de SAT-difícil de Subgráficas Cúbicas debemos mostrar el regreso.

Afirmación 12. Si $G' = (V', E')$ tiene una subgráfica cúbica entonces G es 3-coloreable.

Prueba: Sea $G'_1 = (V'_1, E'_1)$ una subgráfica cúbica, no trivial, de G. Primero diremos que para cualquier subgráfica H de G', $D'(H)$ denotará el grado máximo de los vértices de H en G'_1. Primero veremos algunas propiedades que tendrá G'_1 y después daremos la coloración de G

Afirmación 13. Sean x y y dos vértices de V. Sean $\{x_1, ..., x_{2n}\}$ y $\{y_1, ..., y_{2n}\}$ el conjunto de sus ocurrencias en E respectivamente. Las correspondientes vértice-gráficas y arista-gráfica serán G_{x_i}, G_{y_j} y G_{x_i,y_j}. Y sea $k \in \{0, 1, 2\}$. Entonces:

1. Para $i \in \{1, ..., 2n\}$; $d'(c_k(x_i)) = 0$ si y sólo si $d'(b_k(x_i)) = 0$ si y sólo si $d'(c'_k(x_i)) = 0$.
2. Para $i \in \{1, ..., 2n\}$ y $j \in \{1, ..., 2n\}$; $d'(b_k(x_i)) = 0$ si y sólo si $d'(b_k(x_j)) = 0$.
3. Para cada $x \in V$ y $y \in V$; $d'(cl(x)) = 0$ si y sólo si $d'(cl(y)) = 0$.

4. Para cada $x \in V$, $i \in \{1, ..., 2n\}$; $d'(cp(x)) = 0$ entonces $D'(G_{x_i}) = 0$.

5. Para cada $\{x_i, y_j\} \in E$; $D'(G_{x_i}) = D'(G_{y_j}) = 0$ entonces $D'(G_{x_i,y_j}) = 0$.

6. Para cada $x \in V$; $d'(cp(x)) = 0$ entonces $d'(cl(x)) = 0$.

Prueba: Demostraremos cada uno de los incisos:

1. Observemos que $\{c_k(x_i), b_k(x_i)\} \in E'$ y que $d(b_k(x_i)) = d(c_k(x_i)) = 3$, así por la Afirmación 10 (iii) $d'(c_k(x_i)) = 0$ si y sólo si $d'(b_k(x_i)) = 0$. De la misma forma, $\{b_k(x_i), c'_k(x_i)\} \in E'$ y $d(b_k(x_i)) = d(c'_k(x_i)) = 3$, por tanto también se cumple la segunda equivalencia.

2. Recordemos que en la construcción de la gráfica, tenemos a las aristas:

   $$\{ \{c_k(x_0), c_k(x_1)\}, \{b_k(x_1), b_k(x_2)\}, ..., \{c_k(x_{2i-1}), c_k(x_{2i})\}, \{b_k(x_{2i}), b_k(x_{2i+1})\}, ...,$$

   $$\{c_k(x_{2n-1}), c_k(x_{2n})\} : 0 \leq k \leq 2; x \in V \text{con 2n ocurrencias}\}.$$

   Observemos que existe una trayectoria de $b_k(x_i)$ a $b_k(x_j)$ en donde cada vértice tiene grado 3 en G', así, por la Afirmación 10.4 $d'(b_k(x_i)) = 0$ si y sólo si $d'(b_k(x_j)) = 0$.

3. Como en el inciso anterior, existe una trayectoria para cada $x \in V$ y $y \in V$ de $cl(x)$ a $cl(y)$. Además, todos los vértices de la trayectoria tienen grado 3 en G'. Por tanto, aplicando la Afirmación 10.4 se cumple que $d'(cl(x)) = 0$ si y sólo si $d'(cl(y)) = 0$.

4. Sea $x \in V$ tal que $d'(cp(x)) = 0$. Puesto que $d(b_k(x_{2n})) = 3$ y $\{b_k(x_{2n}), cp(x)\} \in E'$ para cada $k \in \{0, 1, 2\}$, por la Afirmación 10.2 tenemos que $d'(b_k(x_{2n})) = 0$. Por tanto, por el recíproco de la Afirmación 10.1, también tenemos que $d(c_k(x_{2n})) = d(c'_k(x_{2n})) = 0$. De esta forma $c(x_{2n})$ y $c'(x_{2n})$ sólo serían adyacentes entre sí y a x_{2n} en G'_1, pero deben tener grado 3, entonces $d(c(x_{2n})) = d(c'(x_{2n})) = 0$. Pero $d(x_{2n}) = 3$ y $\{x_{2n}, c(x_{2n})\} \in E$, entonces $d'(x_{2n}) = 0$. Así, demostramos que $D'(G_{x_{2n}}) = 0$. Pero por los incisos anteriores, tenemos que si $d'(b_k(x_{2n})) = 0$ entonces $d'(b_k(x_j)) = 0$ para toda $j \in \{1, ..., 2n\}$ y por lo tanto $d'(c_k(x_j)) = d'(c'_k(x_j)) = 0$. Así $D'(G_{x_j}) = 0$ para cada $x \in V$, $j \in \{1, ..., 2n\}$.

5. Sea $a = \{x_i, y_j\} \in E'$ tal que $D'(G_{x_i}) = D'(G_{y_j}) = 0$. En particular $d'(x_i) = d'(y_j) = 0$. Entonces $d'(arc(a)) \leq 1$ lo cual implica que $d'(arc(a)) = 0$. Pero $\{link(a), arc(a)\} \in E'$ y $d(link(a)) = 3$, entonces también tenemos que $d'(link(a)) = 0$. Por otra parte tenemos que $d'(c'(x_i)) = d'(c'(y_j)) = 0$ entonces $d'(c(a)) \leq 2$ y así tenemos que $d'(c(a)) = 0$. Lo que implica que $d'(test1(a)) = d'(test2(a)) = 0$. Así, mostramos que $D'(G_a) = 0$.

6. Observemos que $\{cp(x), cl(x)\} \in E'$ y $d(cl) = 3$, entonces por la Afirmación 10.2, $d'(cl(x)) = 0$.

□

Afirmación 14. Sea x un vértice de V. Entonces $d'(cp(x)) = 3$ y existe una única $k \in \{0, 1, 2\}$ tal que E'_1 contiene las siguientes tres aristas:

$$\{cp(x), cl(x)\}, \{cp(x), b_k(x_1)\}, \{cp(x), b_k(x_{2n})\}.$$

Prueba: Supongamos que $d'(cp(x)) = 0$ para algún $x \in V$. Por Afirmacion 13.4 y .6 tenemos que $D'(G_{x_i}) = 0$ y $d'(cp(x)) = 0$. Además, por Afirmación 13.3, $d'(cl(z)) = 0$ para cualquier otro $z \in V$. Puesto que $d'(cl(z)) = 0$, entonces para que $d'(cp(z)) \neq 0$ debe ocurrir cualquiera de las siguientes situaciones; o $\{cp(z), b_k(z_1)\} \in E'_1$ para cualquier $k \in \{0, 1, 2\}$ ($\{cp(z), b_k(z_{2n})\} \in E'_1$) o, el segundo caso, que las siguientes aristas pertenezcan a E'_1: $\{cp(z), b_k(z_1)\}$; $\{cp(z), b_l(z_1)\}$; $\{cp(z), b_m(z_{2n})\}$ donde $k, l, m \in \{0, 1, 2\}$ y $k \neq l$ (o $\{cp(z), b_k(z_{2n})\}$; $\{cp(z), b_l(z_{2n})\}$; $\{cp(z), b_m(z_1)\}$).

En el primer caso, puesto que no existen aristas de $cp(z)$ a $b_k(z_{2n})$ en E'_1 para toda $k \in \{0, 1, 2\}$ y $d(b_k(z_{2n})) = 3$ entonces $d'(b_k(z_{2n})) = 0$ pero por la Afirmación 13.2 tenemos que $d'(b_k(z_j)) = 0$ para toda $k \in 0, 1, 2$ y para toda $j \in \{1, ..., 2n\}$, lo cual contradice el hecho de que $\{cp(z), b_k(z_1)\} \in E'_1$ para cualquier $k \in 0, 1, 2$.

En el segundo caso, como $d(b_k(z_{2n}) = d(b_l(z_{2n}) = 3$ y las aristas $\{cp(z), b_k(z_{2n}\}$ y $\{cp(z), b_l(z_{2n}\}$ no están en E'_1 entonces $d'(b_k(z_{2n}) = d'(b_l(z_{2n}) = 0$. Entonces, por la Afirmación 13.2 se tiene que $d'(b_k(z_i) = d'(b_l(z_i) = 0$ para toda $i \in \{1, ..., 2n\}$. Lo mismo ocurre para el vértice $b_m(z_1)$, como

$d(b_m(z_1)) = 3$ y $\{cp(z), b_m(z_1\} \notin E_1'$ entonces $d'(b_m(z_1)) = 0$ y por Afirmación 13.2, $d'(b_m(z_j)) = 0$ para $j \in \{1, ..., 2n\}$, lo cual contradice el hecho de que $\{\{cp(z), b_k(z_1)\}, \{cp(z), b_l(z_1)\}, \{cp(z), b_m(z_{2n})\}\} \in E_1'$.

En conclusión, para cada $x \in V$, $i \in \{1, ..., 2n\}$; como $d'(cp(x)) = 0$ entonces $D'(G_{x_i}) = 0$. Además, para toda arista gráfica tendríamos que $D'(G_{x_i,y_j}) = 0$. Así, hemos demostrado que si $d'(cp(x)) = 0$ para algún $x \in V$ entonces $D'(G') = 0$ lo cual nos dice que $E_1' = \varnothing$ lo cual es una contradicción.

Ahora veamos que existe una única $k \in \{0, 1, 2\}$ tal que E_1' contiene las siguientes tres aristas: $\{cp(x), cl(x)\}$, $\{cp(x), b_k(x_1)\}$, $\{cp(x), b_k(x_{2n})\}$.

Ya vimos, en la primera parte de la demostración, que E_1' contiene necesariamente a la arista $\{cp(x), cl(x)\}$. Además, por las Afirmaciones 10.1 y 13.2 tenemos las siguientes equivalencias:

$$\{cp(x), b_k(x_1)\} \in E_1' \text{ si y sólo si } d'(b_k(x_i)) = 3$$

$$\text{si y sólo si } d'(b_k(x_{2n})) = 3 \text{ si y sólo si } \{cp(x), b_k(x_{2n})\} \in E_1'$$

Así, como $d'(cp(x)) = 3$, se cumple que $\{cp(x), b_k(x_1)\}$ y $\{cp(x), b_k(x_{2n})\}$ pertenecen a E_1' para una única k en $\{0, 1, 2\}$.

□

Afirmación 15. Para cualquier $x \in V$ tenemos que $d'(x_i) = 3$ para $i \in \{1, ..., 2n\}$, y existe una única $k \in \{0, 1, 2\}$ tal que $d'(c_k'(x_i)) = 3$ para cualquier $i \in \{1, ..., 2n\}$.

Prueba: Sea x un vértice de V. Sabemos, por el lema anterior que $d'(cp(x)) = 3$ y existe una única $k \in \{0, 1, 2\}$ tal que $\{cp(x), b_k(x_{2n})\} \in E_1'$. Como $d'(b_k(x_{2n})) = 3$ entonces $d'(c_k(x_{2n})) = d'(c_k'(x_{2n})) = 3$ por Afirmación 10 (i). Por la misma razón, puesto que $\{cp(x), b_l(x_{2n})\} \notin E_1'$ para $l \neq k$ tenemos que $d'(_l(x_{2n})) = d'(c_l(x_{2n})) = d'(c_l'(x_{2n})) = 0$ para $l \neq k$. Ahora, como $d'(c_k(x_{2n})) = 3$ y $d'(c_k'(x_{2n})) = 3$, las aristas $\{c(x_{2n}), c_k(x_{2n})\}$ y $\{c'(x_{2n}), c_k'(x_{2n})\}$ pertenecen E_1' por Afirmación 10.1. Además, todos los vértices restantes de $c(x_{2n})$, excepto por x_{2n} tienen grado cero en G_1'. En consecuencia $\{c(x_{2n}), x_{2n}\} \in E_1'$ y $d'(x_{2n}) = 3$. Por consiguiente, probamos que $d'(b_k(x_{2n}))$ implica que

$$d'(x_{2n}) = d'(c(x_{2n})) = d'(c_k(x_{2n})) = d' = (c_k'(x_{2n})) = 3$$

y que $d'(c_l(x_{2n})) = d' = (c'_l(x_{2n})) = 0$ para $l \neq k$. De la misma manera, podemos probar que $d'(b_k(x_i)) = 3$ implica que

$$d'(x_i) = d'(c(x_i)) = d'(c_k(x_i)) = d' = (c'_k(x_i)) = 3$$

y que

$$d'(c_l(x_i)) = d' = (c'_l(x_i)) = 0$$

para $l \neq k$. Pero, por Afirmación 13.2 sabemos que $d'(b_k(x_{2n})) = 3$ si y sólo si $d'(b_k(x_i)) = 3$. Esto completa la prueba.

□

Ahora, definamos una 3-coloración de G tal que:

$col : V \to \{0, 1, 2\}$ dada por $col(x) = k$ si y sólo si
$\{control(x), b_k(x_1)\} \in E'_1$

Esta función bien definida por la Afirmación 14. Veamos que es una buena coloración. Supongamos que no lo es. Entonces existe un arista $a = \{x_i, y_j\} \in E$ tal que $col(x) = col(y) = k$. Entonces, las aristas $\{cp(x), b_k(x_i)\}$ y $\{cp(y), b_k(y_j)\}$ pertenecen a E'_1. Además, tenemos que

$$d'(b_k(x_1)) = d'(b_k(y_1)) = 3$$

y, usando la Afirmación 13.2

$$d'(b_k(x_i)) = d'(b_k(y_j)) = 3.$$

Así, por la Afirmación 13.1 $d'(c'_k(x_i)) = d'(c'_k(y_j)) = 3$. Entonces, tenemos que $d'(c'_l(x_i)) = d'(c'_l(y_j)) = 0$ para $l \neq k$, usando el lema anterior. Ahora, $d'(c'_k(x_i)) = 3$ implica que $d'(c'_k(a)) = 3$ por Afirmación 10.1. Además, $d'(c'_l(x_i)) = d'(c'_l(x_i)) = 0$ para $l \neq k$ implica que $d'(c'_l(a)) = 0$ puesto que $c_l(a)$ tiene únicamente 4 vecinos y dos de ellos son $c'_l(x_i)$ y $c'_l(y_j)$.

Por otra parte, sabemos que $d'(x_i) = 3$. Entonces, si consideramos el camino $x_i, arc(a), link(a)$ podemos concluir, por la Afirmación 10.4 que $d'(link(a)) = 3$. Así, como $d'(link(a)) = 3$, entonces E'_1 contiene a todas las aristas que incluyen a este vértice, en particular, $\{link(a), test1(a)\}$. Los otros vecinos de $test1(a)$ son $c_k(a)$ y $c_l(a)$ para $l \neq k$. Pero, ya hemos probado que sólo uno de estos vértices, $c_k(a)$, no tiene grado 0. Finalmente, hemos probado que $1 \leq d'(test1(a)) \leq 2$, contradiciendo el hecho de que G'_1 es una gráfica cúbica. Por tanto col es una buena coloración para G.

□

Así, hemos probado que encontrar una Subgráfica Cúbica es un problema SAT-difícil.

■

Como vimos, todas las reducciones anteriores se pueden realizar en un tiempo lineal con una máquina de Turing. A continuación haremos un ejemplo de una reducción polinomial.

Proposición 4.9. 3-coloración $\leq$ gráfica plana 3-coloreable.

Demostración: Dada una gráfica $G = (V, E)$ queremos construir una gráfica $G' = (V', E')$ tal que G es 3-coloreable si y sólo si G' es plana 3-coloreable. Construiremos G' unicamente añadiendo a la gráfica G el siguiente Gadget H, plano, en los cruces de sus aristas:

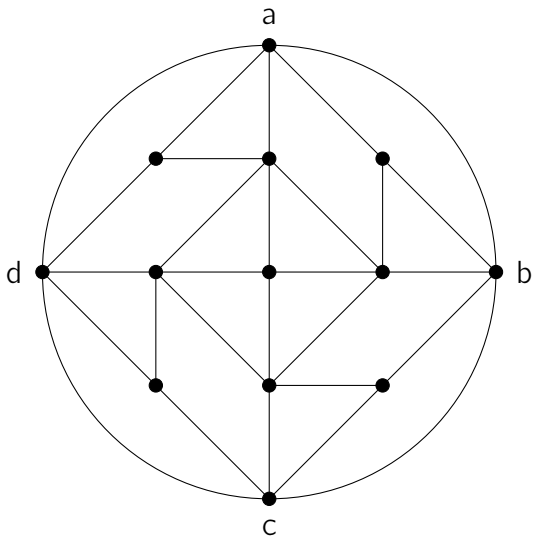


Figura 4.12: Gadget H.

Observemos que este Gadget tiene las siguientes dos propiedades:

Afirmación 16. 1. Cualquier 3-coloración válida f de H satisface que $f(a) = f(c)$, $f(b) = f(d)$ y $f(a) \neq f(b)$.

2. Al colorear los vértices a y c de un mismo color nos inducirá una 3-coloración válida de H.

Prueba: Para probar el primero, supongamos sin pérdida de generalidad que el vértice que está en el centro es de color 0, así, los vértices que son adyacentes a este deben tomar los colores 1 y 2 de manera alternada. Observemos que los vértices que están entre los vértices ab, bc, cd y da tienen que ser del mismo color del vértice central, en este caso, el color 0. Y puesto que los vértices $\{a, b, c, d\}$ están en un 4-ciclo y son adyacentes a los vértices de color cero, toman los colores 2 y 1 de forma alternada.

Para la segunda observación, supongamos, sin pérdida de generalidad, que los vértices a, c toman el color 0 y los vértices b, d toman el color 1 así los vértices que están entre los vértices ab, bc, cd y da, tendrán el color 2, haciendo que los vértices que son adyacentes al vértice del centro del Gadget tomen, alternadamente, los colores 0 y 1 y finalmente, el vértice central tendrá el color 2. Teniendo así una 3-coloración válida para el Gadget H.

□

Veamos cómo construiremos G'. Tomaremos en cuenta todos los vértices de G, junto con todas las aristas que no se intersecten en G. Si las aristas que se intersectan en G son $\{u, v\}$ y $\{x, w\}$, sin pérdida de generalidad, u tomará el lugar de a y x tomará el lugar de d.

Para facilitar la notación, diremos que los vértices u' y x' son lo extremos de u y x en H respectivamente. Ahora, veamos que G' es plana 3-coloreable si y sólo si G es 3-coloreable. Probaremos primero la suficiencia:

Afirmación 17. G' es plana 3-coloreable, entonces G es 3-coloreable.

Prueba: Sea $col : V' \to \{0, 1, 2\}$ una 3-coloración válida de G', tomaremos entonces la restricción de $col(v)$ para los vértices que pertenecen a G. Veamos que esta es una 3-coloración válida para G. Sea $a = \{u, v\} \in E(G)$, si a no tiene un cruce con alguna otra arista en G, como col es una 3-coloración válida de G', entonces u y v tendrán colores distintos en col'. Si $a = \{u, v\} \in E(G)$, tiene algún cruce con alguna o varias aristas, hemos visto que los vértices extremos del gadget tendrán, por pares, el mismo color en toda 3-coloración válida. Supongamos sin pérdida de generalidad que u es el vértice que pertenece al Gadget, así los vértices que están en el extremo de los

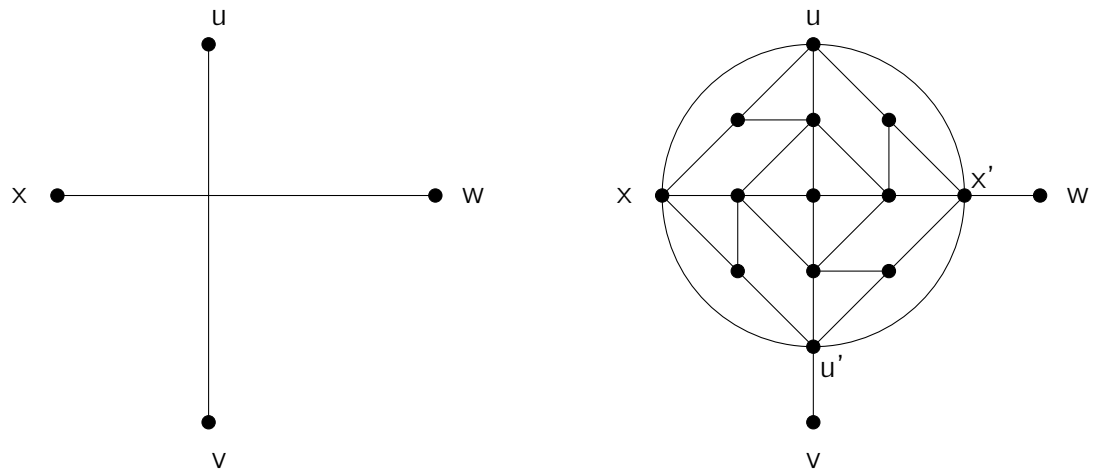


Figura 4.13: Construcción de G'.

gadgets que representan a cada uno de los cruces de a tendrán el mismo color que u, puesto que col es una buena 3-coloración de G', entonces v tendrá que tomar un color distinto al de todos vértices mencionados y por tanto $col'(u) \neq col'(v)$.

□

Probemos ahora la necesidad:

Afirmación 18. G es 3-coloreable, entonces G' es plana 3-coloreable.

Prueba: Esta prueba la realizaremos por inducción sobre el número de cruces entre las aristas de G. Sea $col : V(G) \to \{0, 1, 2\}$ un buena 3-coloración de G. Si G no tiene cruces, entonces $col' : V(G') \to \{0, 1, 2\}$ tal que $col'(v) = col(v)$ para todo $v \in G'$ es una 3-coloración válida para G'.

Supongamos por hipótesis inductiva, que existe una 3-coloración válida para G^n, la gráfica que resulta de sustituir las n cruces por sus respectivos gadgets. Probemos que se cumple para $n + 1$. Sabemos que existe una 3-coloración valida para G^n, digamos $col^n : V(G^n) \to \{0, 1, 2\}$, extenderemos esta 3-coloración a los vértices de G^{n+1}, de la siguiente forma:

Si $v \in G$ y $v \notin H$, entonces $col^{n+1}(v) = col^n$. Si el cruce pertenece a las aristas $a = \{u, v\}$ y $b = \{x, y\}$. Si $u \in G \cap H$, entonces asignaremos el mismo

color de u a su extremo, u' en H, es decir $col^n(u) = col^{n+1}(u) = col^{n+1}(u')$. Análogamente podemos suponer que $x \in G \cap H$, y así también asignaremos el mismo color que tiene el vértice x a su extremo, x'. Por la segunda parte de la Afirmación 16 se inducirá una buena 3- coloración al resto de los vértices del respectivo gadget. Y puesto que u' tiene el mismo color que u, entonces $col^{n+1}(u') \neq col^{n+1}(v)$, de la misma forma con los vértices x' y y. Así, col^{n+1} es una 3-coloración válida para G^{n+1}.

□

■

Capítulo 5

Conclusiones

En el artículo en el que se basó esta tesis, han desarrollado un método para probar que ciertos problemas de la Teoría de Gráficas son NP-completos, donde el problema de 3-coloración tiene un papel importante. La reducción del problema de para encontrar una subgráfica cúbica en una gráfica G, fue mejorada para garantizar que la reducción se pueda realizar en tiempo lineal. Por otra parte, el problema de planaridad 3-coloreable también ha sido mejorado, pues el gadget propuesto por Stockmeyer[11] no garantiza el hecho que al sustituir el gadget en las intersecciones, la gráfica resultante fuera 3-coloreable.

Con este trabajo se pretende que estudiantes de las carreras de matemáticas y ciencias de la computación, tengan material en español que los introduzca al tema de reducciones, ya sean en tiempo lineal y tiempo polinomial no-lineal. Se compiló el material que se cree necesario para dar pie a las reducciones en la teoría de gráficas, el modelo de reducción que se hizo desde 3-coloración da pie a ser utilizado en otros problemas. Se sabe de la existencia de muchos problemas NP-completos, no sólo en la teoría de gráficas, sino también en otras ramas de la matemática, con esto se espera que se pueda dar entrada a otras reducciones, que tal vez puedan ir de problemas del álgebra a problemas de gráficas o viceversa.

Bibliografía

[1] Chartrand, G., Zhang, P. (2012). *First Course in Graph Theory*. Nueva York: Dover Publications, INC.

[2] Creignou, Nadia (1995). *The class of problems that are linearly equivalent to Satisfiability or a uniform method for proving NP-completeness*. Theoretical Computer Science 145, 111-145.

[3] Enderton, H. B. (2001), *A mathematical introduction to logic*. Segunda Edición. Academic Press: Estados Unidos de América.

[4] Garey, M. R., Johnson, D. S., Stockmeyer, L. (1976). *Some simplified NP-complete graph problems*. Theoretical Computer Science 1, 237-267.

[5] Garey, M. R., Johnson, D. S. (1979). *Computers and Intractability. A guide to the theory of NP-completeness*. Nueva York: W. H. Freeman and Company.

[6] Hopcroft, J. E.; Motwani, R.; Ullman, J.D. (2007). *Introducción a la teoría de autómatas, lenguajes y computación*. Madrid: Pearson Educación S.A..

[7] Mendelson, Elliot (1997). *Introduction to Mathematical Logic*. Londres: Chapman & Hall

[8] Rojas Barbachano, R. *Notas de Clase: Lógica Matemática*. Por publicar.

[9] Ruohonen, Keijo. *Graph Theory*. Notas de clase. Traducción: Janne Tamminen, Kung-Chung Lee, Robert Piché.

[10] Solís Daun, J. E., Torres Falcón, Y. (1995). *Lógica Matemática*. México: Universidad Autónoma Metropolitana.

[11] Stockmeyer, L. (1973). *Planar 3-colorability is polynomial complete*. ACM SIGACT News 5,19?25.

Índice alfabético