Coloreado de grafo con 3 colores NP-Completo

Miguel Lentisco Ballesteros

Jesús Sánchez de Lechina Tejada

Demostración NP-Completo

El problema de colorear un grafo con 3 colores (COLOREAR3) consiste en:

Dado un grafo G y una función $f : G \to \{C1, C2, C3\}$ que asigna a cada nodo n de G un color de los 3 $\{C1, C2, C3\}$ entonces un grafo es coloreable con 3 colores si:

- Cada nodo n de G tiene un color asignado
- Para cada par de nodos conectados (u, v) de G entonces $f(u) \neq f(v)$ (tienen colores distintos).

Para ver que es NP-Completo veremos que está en NP y después haremos una reducción del problema NP-Completo SAT3 a COLOREAR3.

NP

Vemos que está en NP comprobando que podemos hacer una MTND que elija una forma de colorear el grafo de manera no determinista y compruebe si el grafo resultante cumple las condiciones (que cada par de nodos conectados tengan colores distintos).

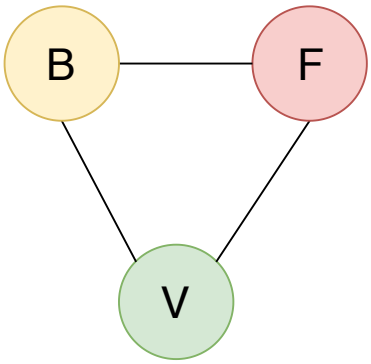
La comprobación se puede hacer con un algoritmo simple que por cada nodo del grafo se compruebe que su color es distinto de todos sus vecinos (nodos conectados); si alguno falla rechaza y si para todos acaba satisfactoriamente acepta.

Por tanto el problema COLOREAR3 está en NP.

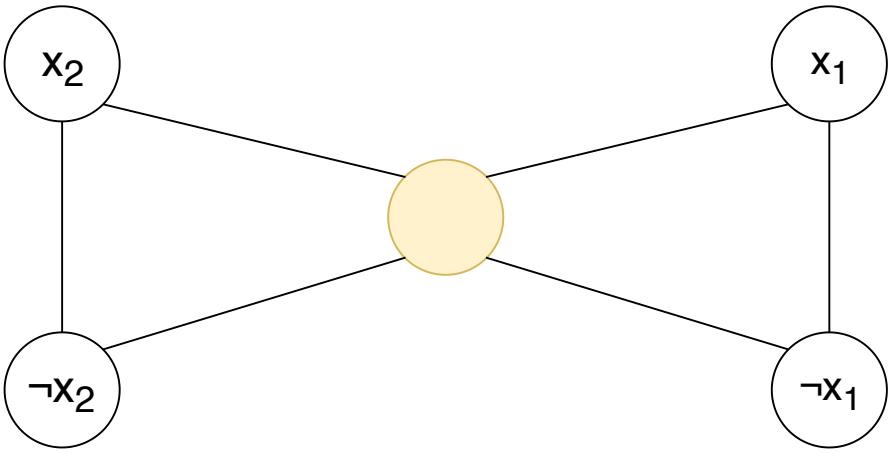
Reducción SAT3 a COLOREAR3

Sabemos que SAT3 consiste en ver si dado un conjunto de símbolos proposicionales (variables) $U = \{p_1, \ldots, p_n\}$ y la colección de cláusulas C donde cada cláusula tiene 3 literales (es decir, de la forma $x_1 \vee x_2 \vee x_3$); entonces ver si C es satisfacible (es decir, que cada cláusula de C sea verdad asignando a cada p_i un valor de verdad - V o F). Por tanto tendremos que encontrar una combinación de valores para cada p_i de manera que al menos un literal de cada cláusula de C sea V.

Tendremos que saber si las variables son V o F, así que diremos que nuestros 3 colores sean $\{V, F, B\}$ que representan respectivamente Verdadero, Falso y Base (este último color es cualquier otro, servirá como apoyo). Y al colorear tendremos que ponerles a nuestras variables un color V o F; por tanto tendremos que crear nodos que representen estas variables y además forzar que si se quisiera colorear el grafo entonces a los nodos de las variables solo puedan darles V o F. Por otro lado en las cláusulas aparecen las variables tal cual o negadas, por lo que también deberian de aparecer como nodos, junto a la misma condición de las variables sin negar.



Todo esto nos indica que por cada variable de SAT3 crearemos dos nodos: un nodo con la variable y otro con la variable negada; finalmente formamos un triangulo conenctandolas con el nodo "Base" de esta manera nos aseguramos que al colorear las variables y las negaciones solo puedan tener el valor F o V y además si una toma un valor (V/F) entonces la negación debe tomar la contraria forzosamente (F/V).

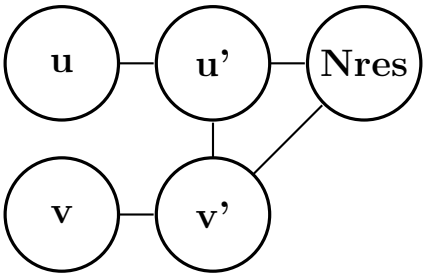


Ya tenemos la manera de que si el grafo es coloreable entonces se le asignaran colores (valores de verdad) correctos a nuestros nodos variable (variables), ahora tenemos que asegurar que si es coloreable entonces cada cláusula es verdad: es decir, para cada claúsula uno de los 3 nodos variable que aparecen en la cláusula

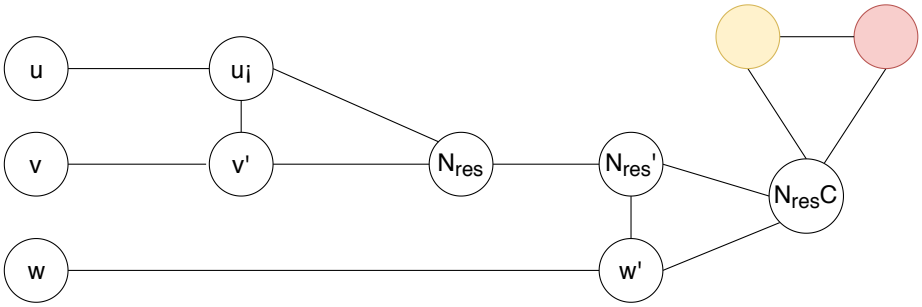
debe estar pintado con V (Verdad); si todos los nodos variable de la cláusula se pintaran con F entonces el grafo debe ser no coloreable.

Para implementar esto último nos servimos de la propia logica de SAT3 donde las cláusulas se implementan con OR ($\vee$), así que haremos una versión de puerta OR en grafos de manera que se cumpla lo que hemos dicho arriba.

Una puerta OR en grafo se construye de la siguiente manera: - Dados dos nodos u, v del grafo se crean 3 nodos adicionales u', v' y n_{res} - Se conectan en triángulo los nodos u', v' y n_{res} - u' se conecta con u y v' se conecta con v. - A n_{res} lo llamaremos el nodo resultado de la puerta OR.



Entonces una cláusula se representa tomando los dos primeros literales que aparecen, se buscan los nodos variables de esos literales y se les aplica la construcción de la puerta OR. Al nodo resultado de esa puerta OR y al nodo variable del tercer literal de la cláusula se le vuelve a aplicar la construcción de la puerta OR. Finalmente al nodo resultado de esa puerta OR lo llamaremos $n_{resClausula}$ (N_{resC}) que simula el resultado de la cláusula.



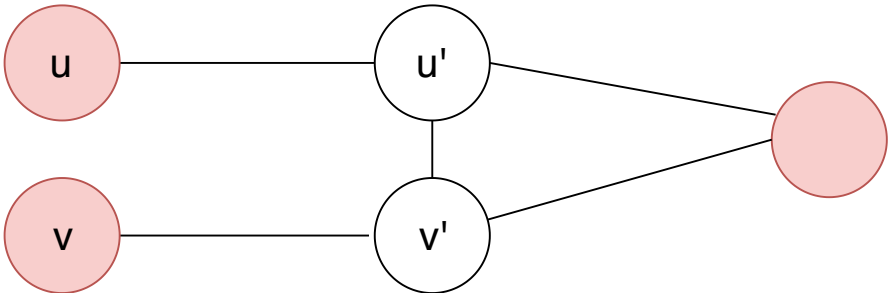
Donde B (en amarillo) estaría conectada con cada una de las variables u,w,v

Como queremos que las cláusulas sean siempre verdad entonces conectamos en triángulo $n_{resClausula}$ con los nodos base Falso y Base; de manera que si se puede colorear el grafo obligamos a que $n_{resClausula}$ se tenga que pintar de V, y por tanto todas las cláusulas son verdad.

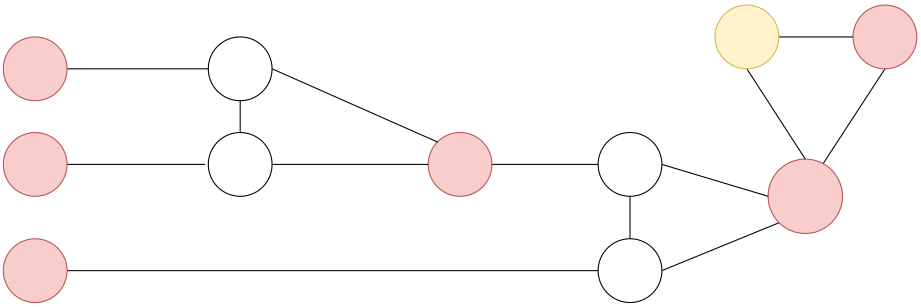
Nos falta ver que si todos los literales de una cláusula se pintan con F entonces $n_{resClausula}$ debería pintarse con F (y por tanto el grafo no sería coloreable) y que si alguno de los literales se pinta con V entonces $n_{resClausula}$ se puede pintar

con V (es decir, que el subgrafo de la cláusula es coloreable).

Veamos que en una puerta OR si los nodos de entrada se pintan con F entonces el nodo resultado se debe pintar con F para que sea coloreable. Obviamente llamemos u, v a los nodos entrada y supongamos que están pintados con F, como u', v' y n_{res} forman un triángulo para que sea coloreable cada vertice debe estar pintado de un color diferente. Al estar conectado v' con v que tiene F entonces v' puede pintarse con V ó B pero no F; e igual pasa con u' con u por lo que n_{res} debe ser F ya que es el único que puede.

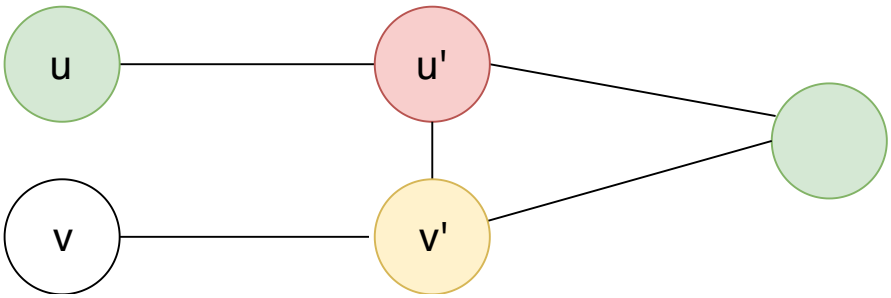


Por tanto al aplicar la puerta OR sobre los literales x_1, x_2 suponiendo que están pintados de F, el resultado se pinta de F y como el literal x_3 también está pintado de F aplicar la puerta OR sobre estos dos nos dice que $n_{resClausula}$ debe estar pintado de F pero está conectado con el nodo Falso pintado de F. Concluimos que entonces el grafo no es coloreable.

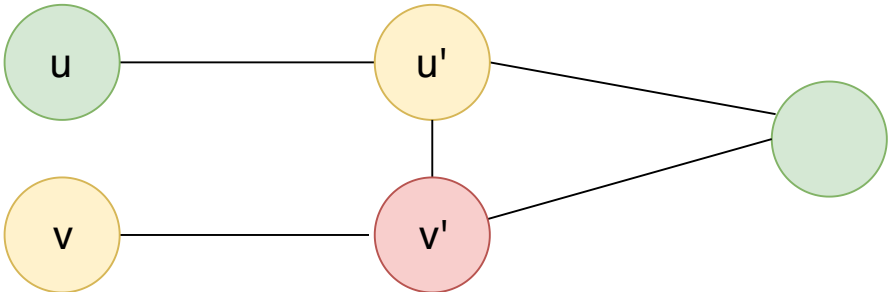


Veamos ahora que en una puerta OR si al menos uno de los nodos de entrada está pintado de V entonces existe una manera de colorear los nodos de la puerta de manera que el nodo resultado esté pintado de V.

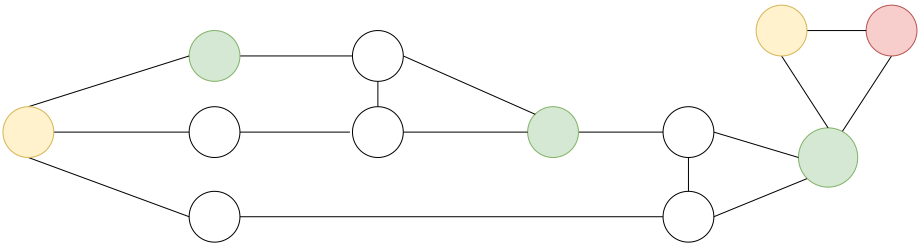
- Suponemos nodos de entrada u, v donde u está pintado de V y v de V o F, queremos que n_{res} tenga color V, por tanto u' y v' deben tomar B y F, sin más que pintando u' de F ya que u tiene V se puede y pintando v' de B dando igual lo que tenga v (F o V).



- Ahora si v tuviera B entonces ponemos v' de F y u' de B y seguiría siendo consistente.



Por tanto si el tercer nodo literal está pintado de V entonces $n_{resClausula}$ está pintado de V (existe al menos una manera de que sea coloreable), y si alguno de los dos primeros nodos literales está pintado de V entonces el nodo resultado de la primera puerta n_{res} estará pintado de V y por tanto aplicamos lo mismo y tendremos que $n_{resClausula}$ está pintado V.



Así concluye la reducción, si encontramos una manera de colorear este grafo habremos resuelto el problema SAT3 sin más que tomar los nodos variables y ver el color asignado sabiendo que V es Verdadero y F es Falso. Por tanto como COLOREAR3 era NP y hemos encontrado una reducción de SATA3 a COLOREAR3 donde SATA3 está en NP-Completo entonces COLOREA3 está en NP-Completo.

Implemetación de la reducción

Implementamos la reducción SAT3 a COLOREAR3 que hemos usado, para ello tenemos que implementar varias clases:

Clausula: representa una cláusula de SAT3, como son del estilo `x1 OR x2 OR x3` simplemente guardamos los 3 literales como strings. Es decir:

```
class Clausula:
  l1 # Literal 1 (string)
  l2 # Literal 2 (string)
  l3 # Literal 3 (string)
```

Nodo: un nodo en COLOREAR3 tendrá un nombre, una lista de nodos con los que está conectados y un color asignado ("F" - Falso, "V" - Verdadero, "B" - Base, "" - Nada aún).

```
class Nodo:
  nombre # Nombre (string)
  nodos_conectados # Nodos conectados (lista de Nodo)
  color # Color (string)
```

Grafo: finalmente guardamos la estructura del grafo que tendrá los nodos (divididos por comodidad a la hora de programar en 3 listas: nodos_base que representan los 3 nodos principales, V,F,B; los nodos_variable que son los nodos que representan las variables y su negación; y finalmente los nodos_puertas que son los nodos auxiliares implementados para hacer las puertas OR en el grafo) y la lista de cláusulas y de variables que provienen de SAT3.

```
class Grafo:
  variables # Las variables de SAT3 (lista de string)
  clausulas # Las cláusulas de SAT3 (lista de Clausula)
  nodos_base # Nodos base (lista de Nodo)
  nodos_variable # Nodos de variables (lista de Nodo)
  nodos_puertas # Nodos de puertas (lista de Nodo)
```

Primero leemos de un fichero mediante un parser la siguiente siguiente estructura, a modo de ejemplo:

```
Variables: x y z
Clausulas:
x or y or z
¬x or y or z
¬x or y or ¬z
```

Una vez leidas las variables y las cláusulas se las pasamos para crear un grafo nuevo, que lo hace con 3 etapas:

1. `inicializar_grafo`: crea los 3 nodos base ("Base", "Verdad", "Falso") y crea un triángulo entre ellos.

2. `crea_nodos_variables`: por cada variable crea un nodo con el nombre de la variable y otro con el nombre de la variable pero negado (por ejemplo "x" y "¬x") y finalmente crea un triángulo entre estos dos nuevos y el nodo "Base" (ninguno de los nodos variable tiene color aún).
3. `crea_nodos_puertas`: por cada cláusula toma los dos primeros literales (busca su nodo variable correspondiente) y les aplica la creación de una puerta OR (`crea_puerta_or`) que se encarga de hacer la estructura usada en la reducción y finalmente devuelve el último nodo (el resultado de la puerta); y después se vuelve a aplicar `crea_puerta_or` al resultado de la anterior puerta OR junto al tercer literal. Finalmente se hace el triángulo con el nodo resultado final y los nodos base "Falso" y "Base" para que se fuerce al que el color sea "Verdad". De todas formas ninguno de los nuevos nodos tiene color asignado (se deja a "").

Por tanto ya tenemos creado un grafo que de ser coloreable con 3 colores entonces podemos responder al problema SAT3 del que venía (el resultado será el color - valor de verdad que toman los nodos variables).

Nota: la implementación es muy simple por lo que hemos decidido solo explicarla.

Algoritmo de resolución

Un algoritmo muy fácil de implementar (pero obviamente de tiempo exponencial y por tanto se sugiere utilizarlo con ejemplos sencillos) para resolver el coloreado con 3 colores de este grafo sería el siguiente:

La idea básica es ver que nodos faltan por pintar, y darles un color cualquiera que esté disponible y ver si el subgrafo que nos queda es coloreable; si es coloreable he acabado, si no es entonces volvemos a la elección de color y probamos con otro, si en algún momento no tenemos colores disponibles (porque no se podía o ya hemos probado y no era coloreable) entonces es que no se puede colorear. Finalmente como caso base, si el grafo no tiene nodos sin pintar es que es coloreable.

Si el grafo es coloreable entonces la función devuelve True y el grafo queda coloreado; en caso contrario devuelve False y se quedan los nodos con color "".

La implementación sería `colorea_grafo`:

```
def colorea_grafo(nodos):
    # Si no hay nodos por pintar, el grafo es coloreable
    if nodos.vacio():
        return True
    # Si quedan nodos por pintar, tomo el primero
    nodo = nodos[0]
    # Tomo la lista de colores disponibles para pintar
    colores = colores_disponibles(nodo)
```

```
    # Si no se puede pintar el subgrafo no es coloreable
    if len(colores) == 0:
        return False
    # Quito el nodo (ya está coloreado)
    nodos.delete(nodo)
    # Por cada color disponible pinto el nodo y veo si es coloreable el subgrafo
    for color in colores:
        nodo.color = color
        res = self.colorea_grafo(nodos)
        # Si se puede colorear devuelvo que sí
        if res:
            return True
    # Si no se puede colorear de ninguna manera devuelvo que no
    nodo.color = ""
    return False
```

Transformación de la solución en SAT3

Una llamada a `to_sat3` nos imprime por pantalla en caso de que el grafo sea coloreable, los valores de verdad que deben de tomar las variables para poder resolver el problema SAT3; tomando en cuenta que solo hay que ver los colores de los nodos variable:

```
def to_sat3(self):
    es_resoluble = self.alg_colorear_3_colores()
    msg = ""
    if es_resoluble:
        msg = "Resolución SAT3, valores de verdad:"
        for i in range(0, len(self.nodos_variable), 2):
            msg = msg + "\n" + self.nodos_variable[i].nombre + " = " + self.nodos_variable[i
    else:
        msg = "No se puede resolver el problema SAT3/Colorear el grafo con 3 colores."
    return msg
```

Por ejemplo para:

```
Variables: x y z
Clausulas:
x or y or z
¬x or y or z
¬x or y or ¬z
```

Nos devuelve por pantalla:

```
Resolución SAT3, valores de verdad:
x = F
y = F
```

```
z = V
```

Que efectivamente resuelve SAT3.

Nota: también se imprime por pantalla el "grafo" (se imprimen la información de cada nodo: nombre, lista de nombres de nodos conectados y su color).