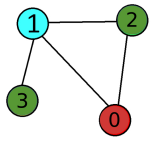
Lab10: Codificar el 3-coloreado de grafos en instancias de SAT

1. **Explicación de la codificación**

   El problema de colorear un grafo con tres colores consiste en decidir si los nodos del grafo pueden colorearse empleando 3 colores de manera que dos nodos adyacentes no tengan el mismo color. La codificación del problema de la 3-coloración para el grafo siguiente como una instancia del problema SAT se explica a continuación.



   - Supongamos que disponemos de los colores $\{r, g, b\}$ (rojo, verde y azul). El conjunto de variables proposicionales es:

     $$\{x_{0,r}\ ,\ x_{0,v}\ ,\ x_{0,a}\ ,\ x_{1,r}\ ,\ x_{1,v}\ ,\ x_{1,a}\ ,\ x_{2,r}\ ,\ x_{2,v}\ ,\ x_{2,a}\ ,\ x_{3,r}\ ,\ x_{3,v}\ ,\ x_{3,a}\}$$

     Una variable $x_{i,j}$ se evalúa a True si, y sólo si, el nodo i tiene asignado el color j.

   - La instancia SAT está formada por las siguientes cláusulas:

     Un conjunto de cláusulas que expresan que cada nodo tiene asignado un único color. Para ello, primero se añaden unas cláusulas expresando que cada nodo tiene asignado al menos un color:

     $$(x_{0,r} \vee x_{0,g} \vee x_{0,b}) \wedge (x_{1,r} \vee x_{1,g} \vee x_{1,b}) \wedge$$

     $$(x_{2,r} \vee x_{2,g} \vee x_{2,b}) \wedge (x_{3,r} \vee x_{3,g} \vee x_{3,b})$$

     y, luego, se añaden las cláusulas expresando que cada nodo tiene asignado a lo sumo un color. Es importante tener en cuenta que $\neg(x_{0,r} \wedge x_{0,g})$ es equivalante a escribir $(\neg x_{0,r} \vee \neg x_{0,g})$.

     $$(\neg x_{0,r} \vee \neg x_{0,g}) \wedge (\neg x_{0,r} \vee \neg x_{0,b}) \wedge (\neg x_{0,g} \vee \neg x_{0,b}) \wedge$$

     $$(\neg x_{1,r} \vee \neg x_{1,g}) \wedge (\neg x_{1,r} \vee \neg x_{1,b}) \wedge (\neg x_{1,g} \vee \neg x_{1,b}) \wedge$$
     $$(\neg x_{2,r} \vee \neg x_{2,g}) \wedge (\neg x_{2,r} \vee \neg x_{2,b}) \wedge (\neg x_{2,g} \vee \neg x_{2,b}) \wedge$$
     $$(\neg x_{3,r} \vee \neg x_{3,g}) \wedge (\neg x_{3,r} \vee \neg x_{3,b}) \wedge (\neg x_{3,g} \vee \neg x_{3,b})$$

     Finalmente, se añade otro conjunto de cláusulas expresando que nodos adyacentes tienen asignados colores diferentes.

     $$(\neg x_{0,r} \vee \neg x_{1,r}) \wedge (\neg x_{0,g} \vee \neg x_{1,g}) \wedge (\neg x_{0,b} \vee \neg x_{1,b}) \wedge$$

     $$(\neg x_{0,r} \vee \neg x_{2,r}) \wedge (\neg x_{0,g} \vee \neg x_{2,g}) \wedge (\neg x_{0,b} \vee \neg x_{2,b}) \wedge$$
     $$(\neg x_{1,r} \vee \neg x_{2,r}) \wedge (\neg x_{1,g} \vee \neg x_{2,g}) \wedge (\neg x_{1,b} \vee \neg x_{2,b}) \wedge$$
     $$(\neg x_{1,r} \vee \neg x_{3,r}) \wedge (\neg x_{1,g} \vee \neg x_{3,g}) \wedge (\neg x_{1,b} \vee \neg x_{3,b})$$

Observa que, a partir de una interpretación que satisface la fórmula, se puede generar una coloración válida: si la variable $x_{i,j}$ tiene asignado el valor True, se asigna el color j al nodo i.

2. **Tarea a realizar**

   Debes implementar la función `reduce_3colorable_to_SAT` que dado un grafo no dirigido devuelve la lista de cláusulas que codifica el problema del 3-coloreado en formato DIMACS (para minisat). En el fichero `Lab10_3colorableToSAT.py` dispones de la función `list2dimacs` que ya está implementada. Esta función escribe en un fichero la lista devuelta por `reduce_3colorable_to_SAT`.

   ATENCIÓN:

   - La lista que devuelve tu función debe estar en formato DIMACS. Por tanto la primera sublista debe ser: `["p", "cnf", num_vars, num_clauses]` donde `num_vars` es el número de variables de tu fórmula y `num_clauses` el número de cláusulas. Además las sublistas siguientes tienen que acabar con un 0.
   - Si el grafo tiene m nodos, el número de variables será $3 \times m$, de forma que la variable i corresponderá al nodo $(i-1)/3$. Si el valor $(i-1)\%3 = 0$ el color es r. Si el valor $(i-1)\%3 = 1$ el color es g. Si el valor $(i-1)\%3 = 2$ el color es b. Siguiendo con el ejemplo, la variable 7 será exactamente $x_{2,r}$.

3. **Solución al problema 3-coloreado usando minisat y plots de grafos**

   Para obtener una solución al problema del 3-coloreado, una vez terminada la función `reduce_3colorable_to_SAT`, la ejecución del programa generará seis ficheros. Cada fichero será una entrada para minisat.

   La fórmula del fichero `g1.txt` corresponde con el grafo de nuestro ejemplo y es satisfactible. Puedes obtener una asignación a las variables que la haga cierta si escribes: `minisat g1.txt asig-1.txt`, se creará un fichero `asig-1.txt` con la siguiente respuesta: $[1, -2, -3, -4, -5, 6, -7, 8, -9, 10, 11, -12]$. Significará que las únicas variables con asignación True son la 1, 6, 8, y 11. Es decir, $x_{0,r}$, $x_{1,b}$, $x_{2,g}$, $x_{3,g}$. Para facilitar la visualización, el método `visualizeGXGraph`, que se invoca desde los test, llama directamente a minisat y, cuando la entrada es satisfactible, crea una imagen con el grafo coloreado.

4. **Solución al problema 3-coloreado usando tu 3SAT-solver**

   Puedes probar tu SAT-solver usando los seis ficheros generados al ejecutar el programa que contiene la función `reduce_3colorable_to_SAT`. Recuerda la función `list_minisat2list_our_sat`. Esta función, dado un fichero en formato DIMACS, devuelve una tupla (t1, t2) donde t1 = número de variables y t2 = fórmula en nuestro formato. Para usarla hemos dejado el fichero `tools.py`, desde donde se importa la función.

5. **Entrega**

   Una tabla con los tiempos que tarda tu SAT_solver en calcular la respuesta para esos seis ficheros.