Segunda práctica de Inteligencia Artificial
Curso 2018-2019

Departament d'Informàtica i Enginyeria Industrial
Universitat de Lleida
carlos@diei.udl.cat
jojeda@diei.udl.cat
jponfarreny@diei.udl.cat

En esta práctica, tendréis que implementar un sistema que sea capaz de resolver los siguientes problemas de grafos, mediante su traducción al problema de la Máxima Satisfactibilidad y la utilización de un *MaxSAT solver*.

1. Minimum Dominating Set
2. Maximum Independent Set
3. Graph Coloring

1. Minimum Dominating Set (MDS)

Un conjunto dominante de vértices de un grafo $G = (V, E)$, es un subconjunto D de G, tal que por cada vértice en V que no este en D, al menos uno de sus nodos adyacentes está en D. El conjunto dominante mínimo es aquel que contiene el menor número de vértices, al tamaño de este conjunto se le llama "independent domination number".

$$G = (V, E),\ D \subset V$$
$$\forall_{v_i \in V}\ v_i \notin D \implies \exists_{v_j \in D}\ v_j \in neighbors(v_i))$$
$$minimize\ |D|$$

Una par de posibles aplicaciones del problema MDS podrían ser:

1. Calcular el número mínimo de *routers* wifi necesarios para dar cobertura a todos los dispositivos de una empresa.
2. Calcular el número mínimo de cámaras necesarias para vigilar las salas de un museo.

2. Maximum Independent Set (MIS)

Un conjunto independiente de vértices de un grafo $G = (V, E)$, es un conjunto de vértices $S \subseteq V$, tal que por cada vértice $v \in S$, ninguno de sus vecinos en G está también en S. El conjunto independiente máximo es aquel que contiene el mayor número de vértices, al tamaño de este conjunto se le llama "independence number".

$$G = (V, E),\ S \subseteq V$$
$$\forall_{v_i \in S}\ neighbors(v_i) \cap S = \emptyset$$
$$maximize\ |S|$$

Una aplicación del problema MIS sería: calcular el máximo de tareas que pueden realizarse simultáneamente sin que dos tareas compitan por el mismo recurso. Los vértices representarían las tareas y por cada par de tareas que compartan un recurso habría una arista.

3. Graph Coloring (GC)

En este problema se trata de asignar colores a los vértices de un grafo $G = (V, E)$, de forma que dos vértices adyacentes no tengan el mismo color. El problema de optimización consiste en calcular el "chromatic index", que es el menor número de colores con el que se puede colorear el grafo, respetando que dos vértices adyacentes no pueden pintarse con el mismo color.

Una aplicación de este problema seria el almacenaje de sustancias químicas, de forma que dos sustancias que producen una reacción peligrosa no se almacenen en el mismo contenedor.

Nota: Para codificar este problema debéis definir el número máximo de colores que se pueden utilizar. El tamaño de la formula final depende de este valor, intentad que sea lo más pequeño posible pero garantizando que la formula tiene solución.

4. Codificación (6 puntos)

El objetivo es, resolver cada uno de los problemas anteriores mediante su codificación a MaxSAT y su resolución mediante un MaxSAT solver. Para ello, debéis usar la estructura de código que se os entrega junto a este enunciado. La salida esperada por pantalla es:

```
INDEPENDENT DOMINATION NUMBER <number>
MDS <vertices>
INDEPENDENCE NUMBER <number>
MIS <vertices>
CHROMATIC INDEX <number>
GC <vertices> | <vertices> | ...
```

Donde *<number>* es un número entero y *<vertices>* es una lista de vértices separados por espacios. En el caso del *graph coloring* (GC) se reporta una lista de vértices por cada color, separándolas con el carácter "|". Por ejemplo, dado el grafo de Petersen el resultado sería:

```
INDEPENDENT DOMINATION NUMBER 3
MDS 1 8 9
INDEPENDENCE NUMBER 4
MIS 3 5 6 7
CHROMATIC INDEX 3
GC 1 3 7 | 2 4 6 10 | 5 8 9
```

4.1. Más de una solución (1 punto)

Estos problemas puede tener más de una solución, debéis extender el código para reportar:

- k soluciones (0.5 puntos)
- Todas las posibles soluciones (0.5 puntos)

Nota: El código proporcionado ofrece los *flags* necesarios para ejecutar la herramienta des de la línea de comandos. Vuestras modificaciones no deben alterar esta parte.

4.2. (1,3)-Weighted Partial MaxSAT (1 punto)

Generar la versión 1,3-Weighted Partial MaxSAT de los problemas anteriores. Para ello debéis Implementar el método *to_13wpm* que se encuentra en el archivo *wcnf.py*.

Para pasar una formula a 1,3-Weighted Partial MaxSAT debeis:

1. Reemplazar las clausulas *soft* por una variable de reificación: (C_i, w_i) se transforma en $(\overline{b_i}, w_i)$
2. Añadir las clausulas *hard* que relacionan C_i con b_i: $(\overline{C_i} \implies b_i, \infty)$
3. Transformar las clausulas *hard* a 3SAT.

5. Implementación (1 puntos)

Se valorará la calidad y eficiencia de las funciones implementadas. Es necesario tener en cuenta los siguientes aspectos de la implementación:

- El lenguaje de programación es **Python 3**.
- La utilización de un diseño orientado a objetos.
- La simplicidad y legibilidad del código.
- **Debéis** seguir la guía de estilo[1].

Nota: Se usara pylint3 para analizar el cumplimiento de la guía de estilo.

6. Documentación (1 puntos)

Se debe entregar un documento en pdf (máximo $\approx$ 3 páginas), que incluya una descripción de como habéis codificado los problemas en MaxSAT (que representan las variables y las clausulas), más la evaluación experimental que hayas realizado. Para ello debéis utilizar notación matemática en la medida que os sea posible.

7. Material a entregar

Todo el material requerido se entregará en un paquete comprimido de nombre ia-prac2.[tgz|tar.gz|zip].

```
ia-prac2.[tgz|tar.gz|zip]
├── informe.pdf
├── src
│   └── *.py
└── grphs
    └── *.dmg
```

[1] `https://www.python.org/dev/peps/pep-0008/`