Resolución de Minimum Vertex Cover utilizando un algoritmo cuántico

Grupo GSyA
Ismael Pérez Nieves
Manuel Ángel Serrano Martín

El problema que tenemos entre manos es una variación del típico Minimum Vertex Cover. Tenemos una tabla con una serie de incidencias que se distinguen entre sí por su ID. Cada incidencia tiene una gravedad y un tiempo estimado relacionados con ella. Además, tiene una serie de controles que tendrán que ser solucionados para considerar que la incidencia ha sido resuelta. Estos controles pueden coincidir de una incidencia a otra, de forma que si resolvemos una incidencia que comparte controles con otra, solucionamos esa otra incidencia a la vez. Lo que queremos hacer es quedarnos con un resultado óptimo quedándonos sólo con las incidencias que necesitemos, cumpliendo mientras a la vez las demás. Vamos a modelar este problema como un problema de optimización QUBO basándonos en parte en la ya conocida solución del MVC, por lo que es recomendable entender esta solución antes de continuar con este documento. Esta solución se explica en https://arxiv.org/pdf/1811.11538.pdf en el punto 4.1.

Primero, vamos a ver los requisitos que tiene que cumplir nuestro programa:

- Tenemos que elegir de un grupo de incidencias un resultado óptimo, de forma que todas las incidencias hayan sido resueltas ya sea de forma directa o indirecta.
- El resultado va a ser luego pasado por otro programa de optimización que considerará cuál es la mejor selección de incidencias teniendo en cuenta su relación precio/peso, siendo el precio la gravedad de la incidencia y el peso el tiempo. Por lo tanto, lo recomendable es que la salida de nuestra solución nos muestre las incidencias que cumplan nuestra restricción con el menor tiempo total.

Para modelar una matriz BQM y usarla en nuestra solución, debemos primero generar la ecuación QUBO. Vamos a hablar primero de la parte a optimizar.

Como hemos dicho, nuestro objetivo es minimizar el tiempo total de las incidencias escogidas. Esto se hace con un simple sumatorio de los tiempos estimados de todas las incidencias seleccionadas. Quedaría de la siguiente forma:

$$\Sigma(x_i * t_i)$$

Siendo x_i la variable binaria que determina que la incidencia i está o no seleccionada, y t_i el tiempo estimado relacionado con la incidencia i. Cabe destacar que nuestro objetivo es minimizar este objetivo, por lo que se tendrá que tener en cuenta cuando hagamos la ecuación QUBO final.

Modelar la restricción no va a ser tan sencillo. Vamos a analizar bien lo que tenemos entre manos. Las incidencias pueden ser cumplidas ya sea porque han sido seleccionadas o porque el conjunto de controles que la componen ya han sido resueltos por una o más incidencias seleccionadas. Si enfocamos este problema como un MVC, podríamos hacer que las incidencias que compartan controles estén relacionadas entre sí, y que sólo haga falta seleccionar a una de las dos incidencias para cumplir con nuestra solución. Desafortunadamente, no es tan sencillo. Tengamos en cuenta el siguiente caso:

Incidencia	Controles
A	1, 2
B	3, 4
C	1, 2, 3

Si seguimos el ejemplo del MVC, podríamos hacer que C estuviese relacionado con A y B, de forma que solucionando C solucionamos también esas dos incidencias. Sin embargo este no es el caso, ya que la incidencia B no estaría completa al estar el control 4 sin solucionar.

Podríamos hacer una complicada trama de relaciones entre las incidencias, haciendo que una incidencia estuviese relacionada a otra sólo en caso de que haya una tercera incidencia ya seleccionada. Sin embargo, he decidido ir por un camino más simple. Si en vez de centrarnos en las incidencias nos centramos en los controles, podemos encontrar una manera de sortear esta restricción.

Al final, lo que queremos es que todas las incidencias estén resueltas, y para determinar que una incidencia está solucionada nos fijamos en si sus controles han sido seleccionados o no. Dicho de otra forma, queremos que todos los controles tengan al menos una incidencia relacionada con él que haya sido seleccionada. Si todos los controles han sido solucionados, sabemos que todas las incidencias van a estarlo también. Vamos a formular esta restricción:

$$i \in C_k \rightarrow \Sigma(x_i) \geq 1$$

Siendo C el conjunto de incidencias relacionadas con el control k. Con esto controlamos que al menos 1 de las incidencias relacionadas con k han sido seleccionadas. Hacemos esto para todos los controles que tengamos y tenemos nuestra restricción completa:

$$\Sigma_k \left(\Sigma_{i \in Ck} \left(x_i - 1\right)^2\right)$$

Con esto ya podemos hacer nuestra ecuación QUBO final añadiendo un coeficiente de penalización (*P*). La pregunta ahora es, ¿cuál escogemos? Aún no se puede determinar a ciencia cierta cómo encontrar este coeficiente, por ahora solo tenemos el método empírico. El coeficiente que funciona en esta solución en concreto es el tiempo estimado más alto de entre todas las incidencias más 1, de forma que la penalización siga afectando la solución. Con todo esto en cuenta, la ecuación QUBO final que se nos queda es la siguiente:

$$\Sigma(x_i * t_i) + P * \Sigma_k \left(\Sigma_{i \in Ck} \left(x_i - 1\right)^2\right)$$

Con esto ya lo tendríamos todo para empezar a crear nuestro código.

Lo primero que queremos hacer es leer la tabla y quedarnos con los datos que necesitamos. Para esta solución he decidido modelar las incidencias y los controles como diccionarios porque luego nos ayudarán a buscar la información que necesitamos más rápidamente. Las incidencias guardarán la gravedad y el tiempo estimado y usaremos como clave su Id, y los controles guardarán las Ids de las incidencias que vayamos leyendo. El método que se encarga de esto es createList con la ayuda de addControl. createList simplemente irá leyendo el archivo .csv yendo línea por línea. Cuando lea una incidencia comparará su Id con la anterior para comprobar si es otra entrada de la misma incidencia. Si es nueva, será añadida al diccionario de incidencias como una nueva entrada. Cada vez que se lea una nueva línea se añadirá un nuevo control por medio del método addControl. Este método comprobará si el control introducido es nuevo o ya está dentro de controles. Si es nuevo se añadirá como una nueva entrada con el id de la incidencia leída, y si se encuentra una coincidencia simplemente se añadirá el id de la incidencia a esa entrada.

Una vez tengamos nuestros dos diccionarios, es hora de crear el BQM con createBQM. Queremos guardar de antemano las claves de los diccionarios de incidencias y controles, los tiempos estimados de las incidencias y el coeficiente de penalización. La parte fácil de la ecuación QUBO, el objetivo, se puede codificar con un simple bucle for que recorra la lista de incidencias y añada el tiempo de cada una como un término linear al BQM.

Para poder codificar la restricción, primero debemos quitarnos el cuadrado que tiene. Tenemos que tener en cuenta que estamos elevando al cuadrado un sumatorio; no tenemos un sólo término x, tenemos una matriz entera de términos. Para verlo más fácilmente, imaginemos que en vez del sumatorio de x, tenemos que elevar al cuadrado la suma de a, b, c y -1. Obtendríamos lo siguiente:

$$(a + b + c - 1)^2$$
$$a^2 + b^2 + c^2 + 1^2 + 2ab + 2ac + 2bc - 2a - 2b - 2c$$

Si volvemos a nuestro caso, a, b y c serían diferentes términos en nuestro sumatorio. Si aplicamos el cuadrado a nuestro caso, obtendremos lo siguiente:

$$P * \Sigma_k (\Sigma_{i \epsilon Ck} (x_i - 1)^2)$$
$$P * \Sigma_k (\Sigma_{i, j \epsilon Ck} (x_i^2 + 1^2 + 2x_i x_j - 2x_i))$$

Teniendo en cuenta que x solo puede tomar como valores 0 y 1, podemos eliminar el cuadrado que tiene ya que es irrelevante; así como el de 1:

$$P * \Sigma_k (\Sigma_{i, j \epsilon Ck} (x_i + 1 + 2x_i x_j - 2x_i))$$

Operamos:

$$P * \Sigma_k (\Sigma_{i, j \epsilon Ck} (-x_i + 1 + 2x_i x_j))$$
$$\Sigma_k (\Sigma_{i, j \epsilon Ck} (-Px_i + P + 2Px_i x_j))$$

Y quitamos la parte constante ya que no afecta a nuestra solución:

$$\Sigma_k (\Sigma_{i, j \epsilon Ck} (-Px_i + 2Px_i x_j))$$

La ecuación final nos da una parte linear ($-Px_i$) y una parte cuadrática ($2Px_i x_j$). Ahora tenemos que ir sumando de forma cumulativa estos términos siguiendo los índices de los sumatorios.

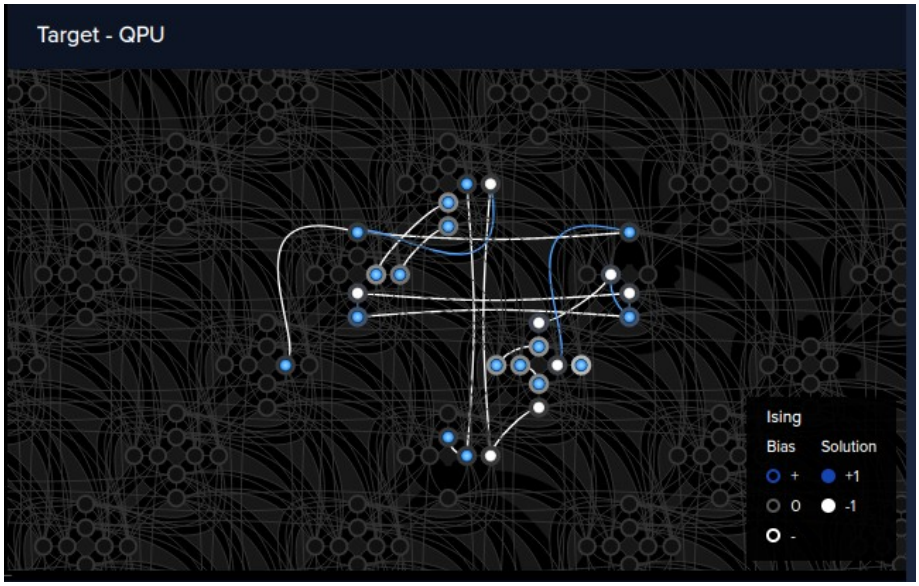
Justo después del bucle for que codificaba el objetivo, tenemos un doble bucle for. El término de fuera, k, irá tomando el valor de las Ids de los controles; e i tomará el valor de las Ids de las incidencias del control k. Por cada i, sumaremos $-P$ al término linear correspondiente a la incidencia i.

El siguiente paso es codificar la parte cuadrática. El primer doble bucle sirve para inicializar los términos lineares; es en el triple bucle en el que iremos modificando estos valores. Este bucle es igual que el anterior, pero añadimos un tercer índice j que nos ayudará a recorrer las Ids de forma cuadrática. Por cada combinación de i y j, sumaremos $2P$ al término cuadrático correspondiente a la incidencia i, j.

El resto del código es el creador del sample, el envío del BQM generado y el procesamiento de la solución.

Inspector

Hemos decidido usar la función del inspector creado por D-Wave para poder ver gráficamente los spins que se producen al resolver este problema. Para ello hemos creado otro archivo llamado inspector.py para poder usarlo. El resultado que se nos muestra es el siguiente:



La disposición de los nodos cambia cada vez que se ejecuta el código por lo que esta imagen puede variar, pero lo que no cambian son las relaciones entre esos nodos. Podemos distinguir dos grupos aislados entre sí. Están compuestos por los nodos 1-6-9 y 4-7-11. Estas relaciones se forman debido a la ecuación BQM que construimos antes:

$$\Sigma_k \left(\Sigma_{i,\, j\, \epsilon\, Ck}\, (-P x_i + 2P x_i x_j)\right)$$

Aparecen debido a la parte cuadrática de la relación, que aumenta la energía de los casos en los que dos nodos relacionados estén activados. Bajando el nivel de abstracción, podemos ver de dónde vienen estas relaciones. Son justo las incidencias que comparten uno o más controles entre sí. Si seguimos con el ejemplo de las 50 incidencias, la incidencia 1 comparte el control 12.1.3 con la 6 y el control 12,3,1 con la 9, y la incidencia 4 comparte el control 9.4.1 con la 7 y el 9.2.4 con la 11.

Al final, lo que nos permite ver el inspector en este problema son las decisiones que debemos tomar. Tenemos que evaluar si coger la incidencia 1 es mejor que coger las incidencias 6 y 9, y si la incidencia 4 es más óptima que coger la 7 y la 11.

El histograma que aparece a la derecha representa la frecuencia absoluta de los resultados obtenidos, agrupadas por la energía. La columna señalada de blanco es la respuesta elegida. Si nos damos cuenta, hay otro grupo que tiene la misma energía que la solución. Esto significa que hay más de una solución posible, y se alternarán entre ellas en cada ejecución del programa.

Comparación de tiempos

Para comprobar la mejora que ofrece el software cuántico respecto al tradicional, hemos decidido resolver este problema utilizando un algoritmo de backtracking y comparar los tiempos de ejecución de ambos programas. La complejidad del algoritmo desarrollado es de:

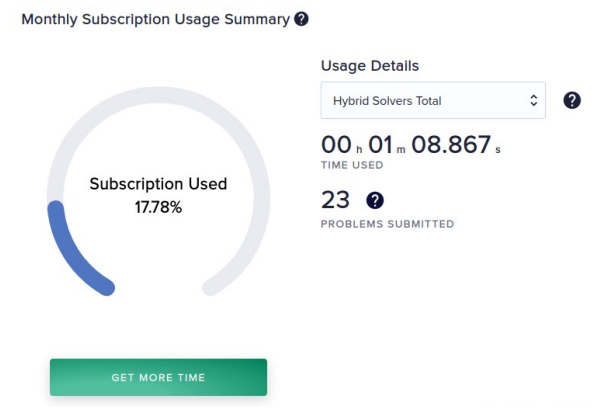
$$2^n + n\ m$$
$$O(2^n)$$

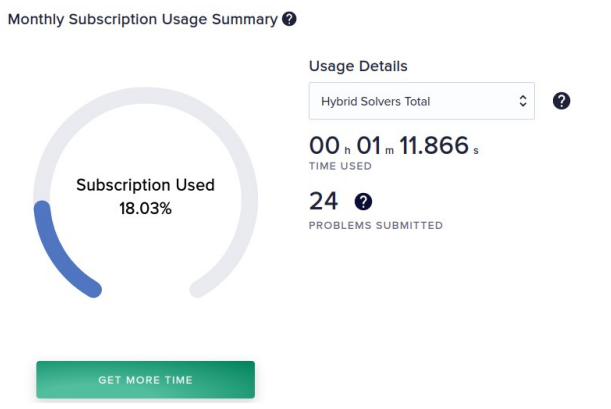
Siendo n el número de incidencias y m el número de controles. Dicho esto, procedemos a comparar los tiempos. Nótese que todo estos tiempos son sin ningún tipo de filtro previo a sus algoritmos para mejor comparación.

No. incidencias	Tiempo cuántico	Tiempo backtracking
12	2.999	0.008
30	2.990	30:24.043
50	2.990	6h+
100	2.997	

Cabe decir que el tiempo que tarda el algoritmo de backtracking a partir de las 50 incidencias está indeterminado debido a la masiva cantidad de tiempo que necesita. Se dejó ejecutando el algoritmo por más de 6 horas y no paró en ningún momento. Esto se podría solucionar con un ordenador más potente, pero con esto es suficiente para hacer la comparación.

Los tiempos del algoritmo cuántico se han sacado a partir de comparar el tiempo de uso de los ordenadores cuánticos de D-Wave que aparece en su página. Comparando el tiempo de uso de antes y el de después, sacamos cuánto tiempo se ha estado usando un ordenador cuántico. Por ejemplo, para sacar el primer ejemplo hemos usado estos dos tiempos:





Con esto obtenemos el tiempo puesto en la tabla, que es 2.999 segundos.

Mirando a la tabla, es obvio que el algoritmo cuántico es el ganador. Es cierto que el algoritmo de backtracking no es la solución más eficiente a este problema, pero incluso con un algoritmo más optimizado no se podría superar el tiempo del algoritmo cuántico ya que es constante.