T5.3. Informe de Evaluación de los Modelos (Bechmarket).

Ibermática – Quantumi3b

[15/07/2023]

Despregamento dunha infraestructura baseada en tecnoloxías cuánticas da información que permita impulsar a I+D+i en Galicia.

Apoiar a transición cara a una economía dixital.

Operación financiada pola Unión Europea, a través do FONDO EUROPEO DE DESENVOLVEMENTO REXIONAL (FEDER) como parte da resposta da Unión á pandemia da COVID-19

DATA	AUTOR	CAMBIOS	VERSIÓN
18/09/2023	Ibermática	Inicial	1

Tabla de contenidos

Lista de figuras

Lista de Tablas

Lista de acrónimos

LCU *Linear Combination of Unitaries*

Objetivo y alcance del proyecto

Introducción

La resolución de problemas combinatorios es una cuestión con un gran impacto tanto en la industria como en la investigación básica, pero que puede conllevar la utilización de un exceso de recursos inabordables clásicamente. Para evitar esta problemática, vamos a optar por una metodología cuántica.

En este documento trataremos los métodos de resolución que hemos desarrollado e implementado para la resolución de casos QUBO (Quadratic Unconstrained Binary Optimization) o su generalización, PUBO (Polynomial Unconstrained Binary Optimization), incluyendo el método de conversión del problema QUBO a un problema de Ising para ser introducido en el QAOA (Quantum Approximate Optimization Algorithm).

Comprobaremos las características del método y los hiperparámetros necesarios para obtener una buena precisión en el cálculo. También evaluaremos la capacidad de resolución del método para problemas generales aleatorios pequeños comparando su resultado con el resultado óptimo obtenido por fuerza bruta. Finalmente haremos un notebook en el cual se desarrolla el método para un problema general y se busca resolver el JSP (Job Scheduling Problem).

Debido a la escasez de recursos puramente cuánticos, también estudiaremos la posibilidad de desarrollar un algoritmo de resolución para el JSP con el método de resolución de problemas de optimización combinatoria en Tensor Networks.

Descripción del problema

El problema general que queremos resolver es el problema de optimización combinatoria binaria con una función de coste asociada. Nuestros elementos a optimizar son vectores o cadenas de componentes binarias $\vec{x} = (x_0, x_1, x_2, \ldots, x_{n-1}),\ x_i \in \{0,1\} \forall i \in [0, n-1]$. Si cada combinación $\vec{x}$ tiene un coste asociado $C\left(\vec{x}\right)$, nuestro problema es encontrar

$$\overrightarrow{x_{opt}} = \text{argmin}_{\vec{x}}\left(C\left(\vec{x}\right)\right),$$

la combinación con menor coste.

La resolución por fuerza bruta de este problema requiere la comprobación de 2n combinaciones, por lo que no es asumible para casos medianamente grandes. Por

ello, se han desarrollado diferentes métodos para obtener soluciones exactas más rápido a partir de las propiedades del problema concreto o métodos que obtienen soluciones aproximadas.

Método QAOA.

El método que implementaremos primero será el QAOA, un algoritmo variacional aproximado que utiliza ordenadores cuánticos de puertas [1]. El objetivo de este algoritmo es obtener la combinación óptima para un problema descrito mediante un hamiltoniano, que puede entenderse como una función de coste. La combinación estará codificada en los estados de la base computacional de los qubits del sistema. Este hamiltoniano se obtiene a partir de una función de coste clásica mediante diferentes tipos de transformaciones. En nuestro caso será del tipo QUBO al hamiltoniano de Ising.

Circuito QAOA.

Para el diseño del circuito (Fig. 1), empezamos con una superposición uniforme de todas las posibles combinaciones, mediante puertas H en cada qubit, y aplicamos una rotación de fase que hará que cada combinación tenga una fase proporcional al coste de la misma, mediante una puerta $e^{i\gamma H(x)}$usando el hamiltoniano del problema. Tras ello, aplicaremos una rotación en el eje X que hará que la amplitud de cada estado se reduzca más cuanto mayor sea su fase mediante una puerta $e^{i\beta X}$en cada qubit, que será la parte de mixing. Esto crea que, dada la normalización, los estados de menor coste tengan una mayor amplitud que los más costosos. Aplicaremos las puertas de rotación varias veces en capas, para ir amortiguando más unas amplitudes y amplificando más otras.

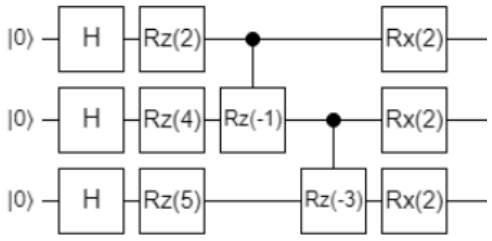

Figura 1: circuito QAOA de una capa de un problema Ising con Hamiltoniano $H = 1\,Z\otimes I\otimes I + 2\,I\otimes Z\otimes I + 2.5\,I\otimes I\otimes Z - 0.5\,Z\otimes Z\otimes I - 1.5\,I\otimes Z\otimes Z$*, con* γ*=-4 y* β*=-4.*

Los factores de proporción γ y β son debidos a la propiedad cíclica de la exponencial compleja, ya que existe la posibilidad de que añadamos una cantidad excesiva de fase. Sin estos factores una energía de 8 tendría una fase 8 - 2 π. Lo mismo sucede con la rotación en X. Por ello mismo tenemos que ajustar cuanto rotamos cada vez de forma óptima con un optimizador clásico.

Hay que tener en cuenta que el resultado del algoritmo no va a ser exactamente el estado óptimo, sino que va a ser una distribución en la cual el óptimo tendrá la mayor probabilidad. A mayor número de capas, más localizada será la distribución.

QUBO, Ising y métodos variacionales.

Los problemas tipo QUBO son problemas que se pueden expresar como un problema cuya función de coste viene dada por un polinomio de grado 2 de variables binarias. Se puede interpretar como un tipo de sistema con interacciones a 1 vecino. Una infinidad de problemas se pueden expresar como un QUBO, desde el Max-Cut hasta el travelling salesman problem [2]. Su forma general es:

$$C(\vec{x}) = \sum_{i}^{n} a_{ii}x_i + \sum_{i<j}^{n} a_{ij}x_i x_j; \qquad x_i \in \{0,1\} \forall i \in [0,n)$$

Estos problemas pueden resolverse de manera exacta, dependiendo del caso concreto, o por algoritmos variacionales, en los cuales se parte de un ciertos "ansatz" parametrizado donde se varían sus parámetros hasta poder alcanzar el mínimo de la función de coste. Nosotros nos centraremos en el QAOA, debido a que está especializado en la resolución de estos problemas, pero existen más "ansatz" disponibles.

El primer paso será trasladar el QUBO a un problema de Ising sin campo magnético. Esto lo conseguiremos mediante una función que realiza el mapeo como

$$H = \sum_{i}^{N} a_{ii}x_i + \sum_{i<j}^{N} a_{ij}x_i x_j \rightarrow H = \sum_{i}^{N} b_{ii}Z_i + \sum_{i<j}^{N} b_{ij}Z_i Z_j$$

$$b_{ii} \mathrel{+}= -\frac{1}{2}a_{ii};\ b_{ii} \mathrel{+}= -\frac{1}{4}a_{ij};\ b_{jj} \mathrel{+}= -\frac{1}{4}a_{ij};\ b_{ij} \mathrel{+}= \frac{1}{4}a_{ij};\ b_{ij} = b_{ji}$$

En el Ising del problema, el hamiltoniano del problema estará compuesto únicamente de combinaciones de puertas I y Z, las cuales conmutan entre sí, así que solo necesitaremos puertas RZ, RZZ y posteriores con el ángulo adecuado.

Una vez tenidos estos valores, se traducen a una lista cuyos elementos serán las posiciones de las puertas Z y los coeficientes asociados. En cada capa habrá que multiplicar el coeficiente asociado a su término con el ángulo γ de esa capa. De esta forma, por ejemplo, en la capa 0 para un término $b_{01}Z_0Z_1$

Aplicaremos una puerta RZZ en los qubits 0 y 1 con el ángulo $b_{01}\gamma_0$.

Podemos aplicar de forma consecutiva los términos uno a uno debido a que todas estas puertas Z y ZZ conmutan entre sí. Para casos con puertas de otro tipo tendríamos que calcular la exponencial de forma directa.

Esto es el circuito del QAOA, para el cual tendremos que determinar su energía en cada paso, esto es, su valor esperado en H(x). Sin embargo, antes discutiremos la divisibilidad del circuito.

Método de los M más probables

La primera idea de aproximación al problema fue utilizar un método "prepare and measure". Consiste en medir el estado resultante un cierto número de veces y luego considerar el coste de los M estados que más apareciesen ponderando con su frecuencia. El motivo para usar un método como este sería que, para medir el valor esperado de la energía para un estado general, tendríamos que medir todas las amplitudes de los estados base, lo cual aumenta de forma exponencial con el número de qubits, o una estimación de las mismas. Por ello, suponemos que el valor esperado va a venir determinado principalmente por los estados con una mayor amplitud.

Sin embargo, el método de los M más probables tiene diversos problemas:

- El número de mediciones necesarias para una buena resolución aumenta con el tamaño del espacio de Hilbert, esto es, exponencialmente.
- Para distribuciones de probabilidad generales, podemos estar haciendo un mal análisis, ya que suponemos que la distribución va a estar centrada en los estados de menor coste de forma suave, lo cual en general no sucede.
- Y el más importante es que, en un caso arbitrario, la inicialización de los ángulos nos llevará a un primer paso de optimización que puede no tener entre sus M más probables el estado óptimo.

Esto, unido a la localidad del método por usar un "learning rate" pequeño, hará que en general el método optimice la distribución de estos M más probables del primer paso sin tener en cuenta los demás, salvo un factor global.

Por estas razones, se ha optado por la creación de un nuevo algoritmo general de obtención de este valor esperado uniendo el Hadamard Test y el Linear Combination of Unitaries (LCU) modificado, una nueva técnica desarrollada por el equipo. Esta técnica nos permite obtener el valor esperado del hamiltoniano sin necesidad de medir directamente el estado, por lo que el muestreo necesario no crece con el espacio. Además, no es dependiente de la distribución de amplitudes del estado y solo requiere la medición de uno de los qubits "ancilla" que añadiremos. El único inconveniente que incluye es la necesidad de añadir qubits extra, aplicar un número notable de puertas condicionadas y un proceso de inicialización.

LCU+Hadamard Test

La idea es estimar el valor esperado de toda la distribución de probabilidad. El Hadamard Test se utiliza para estimar el valor esperado de un cierto operador unitario U mediante el circuito presentado en la Figura 2.

Su valor esperado será

Re[⟨U⟩] = 2P0-1

Siendo P0 la probabilidad de medir 0 en el qubit ”ancilla”.

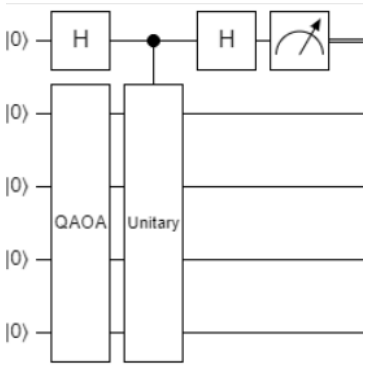

Fig. 2: Circuito para el Hadamard Test aplicado al estado proveniente de un circuito de QAOA. Testeamos el valor esperado del operador Unitary.

El método LCU es utilizado para aplicar una combinación lineal de operadores unitarios, lo cual no es necesariamente unitario, a un estado. Para el LCU estándar queremos aplicar

$$A = \sum_{i=0}^{n} \alpha_i U_i ,$$

Siendo las Ui operadores unitarios y los α valores positivos y normalizados de forma que

$$\sum_{i=0}^{n} \alpha_i = 1$$

Añadimos un conjunto de log2(n) ancillas e inicializamos en el estado:

$$V|0\rangle = \sum_{i=0}^{n} \sqrt{\alpha_i}|i\rangle$$

Con ello, aplicamos de forma controlada cada uno de los operadores unitarios que están asociados al estado base i para que su amplitud asociada se traslade a la amplitud de la puerta aplicada. Ahora aplicamos la inversa de V y obtenemos el estado:

$$|0\rangle \sum_{i=0}^{n} ||\alpha_i||U_i|\psi\rangle + |\perp\rangle B|\psi\rangle$$

De forma que el estado $|0\rangle$ de las ancillas es el estado al que se le ha aplicado A y el estado perpendicular $|\perp\rangle$ es el estado al que se le ha aplicado un operador B que desconocemos. Sin embargo, este estado ni nos interesa ni nos es necesario para el resto del análisis.

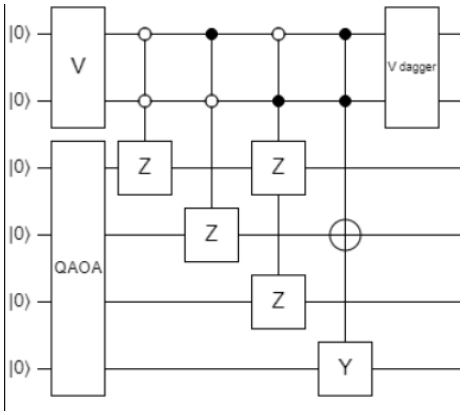

Fig. 3: Circuito LCU para un operador A combinación lineal de 4 operadores unitarios: ZxIxIxI, IxZxIxI, ZxIxZxI, IxXxIxY

Esto solo lo podemos hacer si los α de la matriz son no negativos, por lo que no podemos añadir fases y no podemos aplicarlo al Ising. Podemos generalizar el LCU mediante el uso de dos operadores para la inicialización y desinicialización de las ancillas. Para una matriz:

$$A = \sum_{i=0}^{n} \gamma_i \alpha_i U_i$$

Siendo los α números reales no negativos normalizados y los γ números complejos de módulo 1 (las fases). Vamos a crear el siguiente operador:

$$U|0\rangle = \sum_{i=0}^{n} \gamma_i \sqrt{\alpha_i}|i\rangle$$

De forma que si ejecutamos el siguiente circuito (Figura 4):

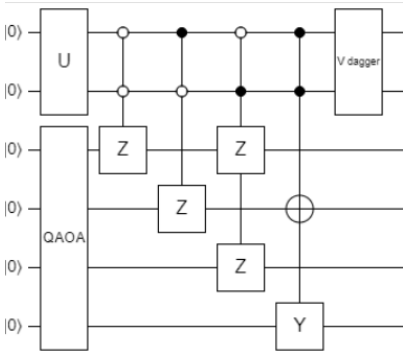

Fig. 4: circuito del LCU generalizado para el operador A.

Tenemos el estado con las fases aplicadas

$$|0\rangle \sum_{i=0}^{n} \gamma_i \alpha_i U_i |\psi\rangle + |\perp\rangle B|\psi\rangle$$

,

Donde en el estado 0 tenemos aplicado el operador A y en el perpendicular tenemos el estado residual que aplica un operador desconocido B, que no conocemos ni necesitamos.

Ahora vamos a unificar ambos métodos para calcular el valor esperado de A aplicado con el LCU, como si fuera el operador unitario del Hadamard Test. Si aplicamos el siguiente circuito (Figura 5):

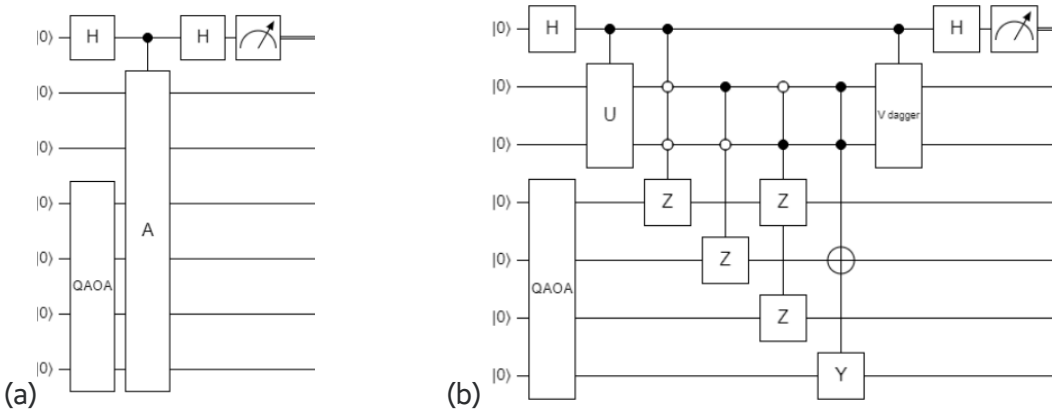

Figura 5: a) Circuito del LCU-Hadamard Test entendido como un Hadamard test del operador A extendido, que comprende todo el LCU de A. b) Circuito que realiza el LCU-Hadamard Test para un ejemplo de operador A.

Con este circuito vamos a aplicar el Hadamard test sobre la puerta 'A', siendo la puerta A la puerta unitaria que nos deja en el estado 0 en las ancillas del LCU nuestro estado A sobre Ψ. Uno podría pensar que necesita medir el 0 en esas ancillas, pero no es necesario, ya que $\mathrm{Re}[\langle A \rangle] = 2P_0 - 1$

Si tuviéramos un operador A con valores complejos, la parte imaginaria se mediría exactamente igual, pero añadiendo una puerta S justo antes de la última puerta H en la ancilla a medir. Esto nos permite identificar el factor de error que tenemos en un caso de operador de valores reales.

Para operadores donde sus coeficientes no están normalizados

$$C = \sum_i \zeta_i U_i; \quad \sum_i \zeta_i = N$$

$$A = \sum_i \frac{\zeta_i}{N} U_i = \sum_i \alpha_i U_i = \frac{C}{N}$$

$$\mathrm{Re}[\langle C \rangle] = (2P_0 - 1)N \qquad (1)$$

Sin embargo, el requerimiento de la inicialización con coeficientes arbitrarios y el gran número de puertas multicontroladas hace que este método no sea compatible con los niveles de ruido presentes en los ordenadores cuánticos de la era NISQ. Aun así, consideramos que puede ser parte de una investigación a futuro.

Resultados

Para poder comparar el método, hemos realizado pruebas para determinar el número de cuentas necesarias para obtener una precisión de al menos dos cifras significativas en el valor esperado. Experimentalmente hemos determinado que se necesitan al menos O(105) "shots". La ecuación (1) expresa que lo único necesario para determinar con precisión el valor esperado es determinar con precisión la P0. Estadísticamente, la precisión a la hora de determinar esta probabilidad crece con $^{O(\sqrt{shots})}$, por lo que necesitamos aumentar cuadráticamente el número de "shots" para obtener una precisión dada. Teniendo en cuenta el prefactor 2N, necesitamos un número de "shots" $^{O\left(\frac{2N}{\epsilon^2}\right)}$, siendo ϵ el error en el valor esperado. Ello mismo provoca que el número de "shots" necesarios no dependa del tamaño del problema.

- Esta aproximación es notablemente mejor que el método de sampleo simple, en el cual necesitamos caracterizar la distribución entera de manera eficiente, lo cual requiere un aumento del número de "shots" con el tamaño del problema, además de depender de la forma de la distribución implicada.
- También es mejor que el método de aplicar el Hadamard Test a cada uno de los operadores unitarios de la combinación lineal y luego ponderar sus resultados, ya que esto requiere el uso de un circuito específico para cada uno

de los operadores, cuyo número tiende a aumentar con el tamaño del problema. Así, necesitamos un aumento cuadrático del número de "shots" para cada operador, por lo que al final necesitamos $O\left(\frac{2Nn}{\epsilon^2}\right)$, siendo n el número de operadores en la combinación lineal.

Comparando el performance en simulador sin ruido de este método LCU+Hadamard Test con el método de sampleo usual, que consiste en medir el sistema entero un número de veces y calcular la función de coste de cada uno de los estados medidos. En la tabla 1, podemos ver los resultados de las pruebas. Resolvimos problemas QUBO con matrices aleatorias usando 105 "shots" para el método LCU+Hadamard y 106 para el sampleo simple. Usamos más "shots" en el caso de sampleo simple para tener la mayor precisión posible y así poder tomarlo como el valor exacto. Los estados para los que calculamos los valores esperados son los obtenidos tras optimizar el circuito QAOA del problema. Para evitar mínimos locales debidos a la inicialización, vamos a realizar 4 inicializaciones diferentes con parámetros iniciales aleatorios y nos quedaremos con el mejor resultado de las 4.

En la figura 6, enseñamos las distribuciones de frecuencias en los estados más frecuentes tras optimizar el circuito QAOA con diferente número de qubits y de capas. Para comparar, obtenemos por fuerza bruta la solución óptima del problema y las soluciones que tengan un coste hasta un 80% menor (mayor) que el óptimo si este tiene un coste negativo (positivo). Así, cuando las haya, las ordenaremos según su coste en la tabla. El óptimo es el 1º, el segundo mejor 2º y el tercero mejor 3º.

Caso	Número de qubits	Capas	Soluciones	⟨H⟩ sampleo simple	⟨H⟩ LCU+Hadamard
1	3	2	1º 011 2º 001	-0.8538	-0.8533
2	4	2	1000	-1.1898	-1.1837
3	5	2	00111	-0.7919	-0.7926
4	6	3	101000	-1.9420	-1.9356
5	7	3	1º 0010100 2º 0011000 3º 0010000	-1.8668	-1.8610

Tabla 1. Valores esperados obtenidos con el método de sampleo simple y con el LCU+Hadamard Test para diferente número de qubits. El valor esperado a medir es el de la matriz del Ising asociado al QUBO. Añadimos las soluciones exactas del problema y el número de capas usadas en el QAOA.

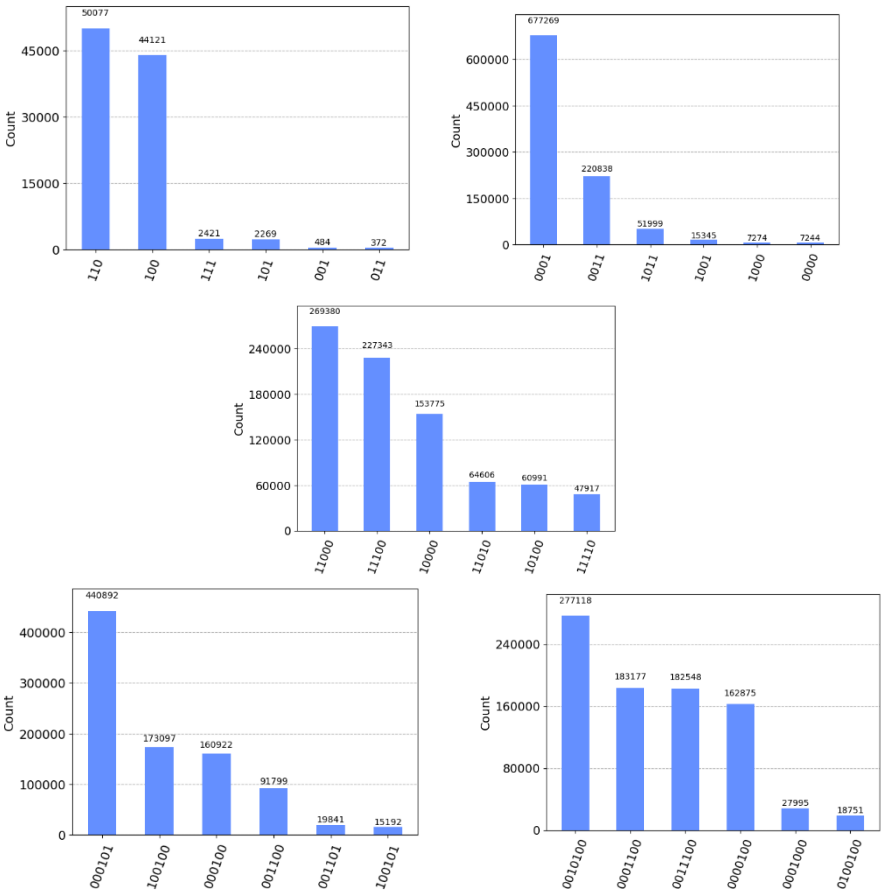

Fig. 6: Distribuciones de frecuencia obtenidas tras el mejor ajuste del QAOA para los casos de la tabla 1. Las strings deben leerse con los bits en el orden inverso, debido a la diferencia de codificación.

En la Figura 6, podemos observar que en el caso (1) se predicen bien los dos estados de menor coste, al igual que en el (2), estando los óptimos separados en frecuencia del resto. Para el caso (3) vemos que el estado más probable no es el óptimo, siendo este el segundo más frecuente. Esto es debido al bajo número de capas para el tamaño del problema y la similitud de los estados. Por otro lado, en el caso (4), ya con una capa más para superar el problema del caso (3), vemos que el óptimo se distingue notablemente, y en el caso (5) se predice bien el óptimo, quedando los subóptimos también separados del resto de estados, con un nivel inferior de frecuencia, aunque entre ellos hay un estado que no estaba en nuestras estimaciones, probablemente cercano a estos.

El método tiene una buena capacidad de resolución con un bajo número de capas y podemos predecir de forma notablemente eficaz el valor esperado de la energía con el método LCU+Hadamard.

Todos estos resultados se han ejecutado y comprobado con un Jupyter Notebook disponible en el equipo de CESGA Optimización que implementa todos los métodos, con su explicación, y el caso del JSP.

Tensor Networks

Ahora vamos a estudiar la posible utilidad de las tensor networks para la resolución de este problema. Esto es debido a que para un JSP suficientemente grande, no tendríamos disponibles suficientes qubits o de suficiente calidad. La ventaja que presentan las tensor networks es la simulación de sistemas cuánticos con bajo entrelazamiento o la implementación de algoritmos clásicos inspirados en algoritmos cuánticos.

La codificación será realizar una grid 2D de nodos para representar los qudits de los posibles jobs en máquinas y tiempos.

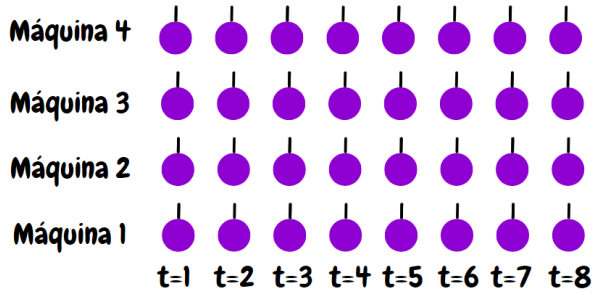

Fig. 7: nodos del estado de superposición básico para el JSP. Cada uno de los nodos es un estado en superposición uniforme, tal que su estado indica el job que se está realizando en el espacio de tiempo correspondiente en el tiempo correspondiente.

Si usamos el formalismo de señales para aplicar las restricciones, y así eliminar los estados no compatibles con el problema. Para ello tenemos que empezar haciendo que un job no pueda estar al mismo tiempo en dos máquinas.

Necesitamos una representación que nos permita visualizar las conexiones del tensor network, que sería un cubo en 3D, por lo que vamos a representar las uniones desde arriba. Si representamos los nodos como cuadrados en una grid, representaremos con líneas que los unen las capas de nodos lineales que los unen, siendo estos 3 y 4-tensores. Así, en la figura 6 podemos ver cómo tenemos nuestra señal para cada tiempo:

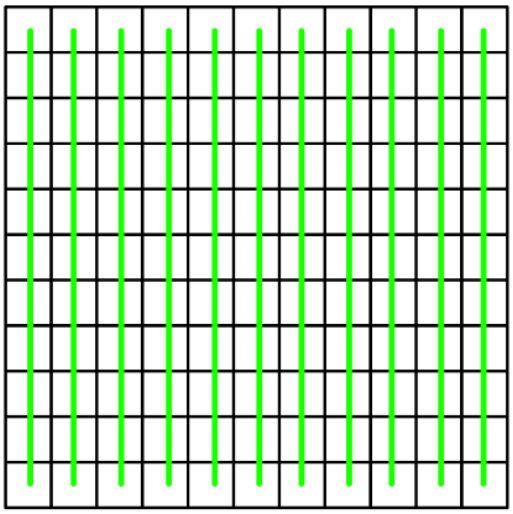

Fig. 8: Uniones para restricción de no coincidencia temporal. Cada línea verde, para n jobs, está compuesta de n capas 2 dimensionales en enlace, por lo que su contracción tiene un coste del tipo 2^n.

Por otro lado, tenemos que imponer también que las tareas duren un cierto tiempo, lo cual, como se puede observar en la figura 8, se cumple para las tareas con duración d, n capas d dimensionales, lo cual cuesta contraerse del tipo 2^n.

Además, necesitamos que se activen todas las capas, por lo que necesitaríamos algún otro tipo de condición horizontal, pero esta iría en los nodos finales de cada máquina como un proyector.

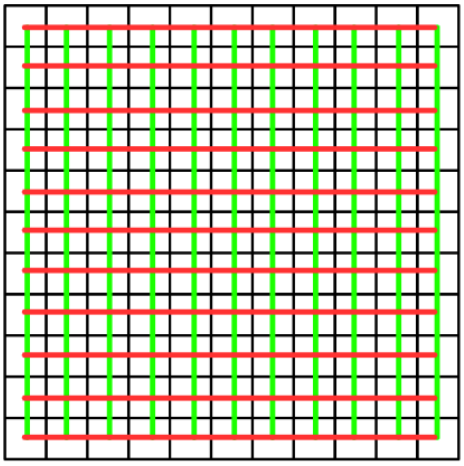

Fig. 9: uniones incluyendo la duración de las tasks. Cada línea roja está compuesta de j capas de dimensión de enlace d

Finalmente, necesitamos la restricción del orden de las tareas por máquina, con lo que necesitamos una señal que indique, para cada espacio de tiempo, si en las otras máquinas se han activado las otras tareas, cosa que iría dentro de las uniones de la duración, pero transmitiéndose en vertical, lo cual generaría una "grid" totalmente conexa.

Ahora en la parte del coste, solo necesitaríamos una capa exactamente igual que las primeras mencionadas en vertical, pero que aumente la amplitud del estado si todos los nodos de esa capa están en cero (sin job). Esto se puede lograr eficientemente.

Con esto, hemos visto que la conectividad del algoritmo es el máximo cuadrado, lo cual implica que su tiempo de contracción aumenta exponencialmente con la raíz del número de nodos, pero debido al número de capas, aumenta muchísimo más rápido que eso, por estar contrayendo un cubo. Por ello mismo no podemos usar el formalismo de transmisión de señales en optimización con Tensor Networks.

Usando un formalismo similar al usado en el QAOA con QUBO podríamos obtener un resultado comparable, aunque menos eficiente, por lo que no es tampoco viable.

Por el hecho de no ser eficientemente emulable en Tensor Networks, con los niveles de entrelazamiento y la forma del mismo, determinamos que el particionado del circuito cuántico no es viable, debido a que el número de cortes crece con el tamaño del problema.

Hamiltoniano

El problema se escribe en su forma QUBO, esto es, como un problema de minimización de una función de coste cuadrática de variables binarias a la que llamaremos Hamiltoniano. Dicho hamiltoniano tiene 4 términos:

- H0: Término de reducción temporal (“cuanto menos tiempo, mejor”). Es el término principal de nuestra optimización. Sólo se tiene en cuenta el tiempo de finalización de la última task de cada trabajo. Como a priori no se puede saber de antemano qué trabajo finalizará más tarde, la heurística intenta finalizar todos los trabajos lo antes posible: sin embargo, para evitar que (por ejemplo) los tiempos de finalización de tasks [5,7] equivalgan a [6,6] (puesto que 5+7=6+6), se elevan los tiempos a una potencia “r”. En el ejemplo, esto hace que 5^2+7^2 > 6^2 + 6^2 por lo que se daría preferencia a la configuración que acaba antes: [6,6].

$$H_0 = \lambda_0 \sum_{i,t} (t + d_{i,j_{fin}(i)})^r x_{i,j_{fin}(i),t} \; : r \geq 1$$

- H1: Término de la restricción ”todas las task deben ser realizadas en un único momento del tiempo”. Dicho de otra manera: para cada task, la suma de todas sus variables a lo largo del tiempo debe ser igual a uno. Se aplicará una penalización de no anularse el término entre paréntesis.

$$H_1 = \lambda_1 \sum_{\text{job,task}} \left(\sum_{\text{time}} x_{i,j,t} - 1 \right)^2$$

$$= \lambda_1 \sum_{i,j} \left(\sum_{t_1,t_2} x_{i,j,t_1} x_{i,j,t_2} - 2 \sum_{t} x_{i,j,t} + 1 \right)$$

$$= \lambda_1 \sum_{i,j} \left(\sum_{t} x_{i,j,t}^2 + 2 \sum_{t_1<t_2} x_{i,j,t_1} x_{i,j,t_2} - 2 \sum_{t} x_{i,j,t} + 1 \right)$$

$$= \sum_{i,j} \left(2\lambda_1 \sum_{t_1<t_2} x_{i,j,t_1} x_{i,j,t_2} - \lambda_1 \sum_{t} x_{i,j,t}^2 \right) + C$$

- H2: Coeficiente de la restricción "diferentes trabajos que requieren la misma máquina no pueden coincidir en el tiempo". Se aplicará una penalización a cada pareja de variables activa si se cumplen las tres siguientes restricciones:

  Son variables que corresponden a task distintas de job distintos.

  Las máquinas necesarias para hacer dichas task resultan ser la misma.

  El tiempo de inicio correspondiente a la segunda variable coincide con el de la primera variable, o con cualquiera entre dicho tiempo de inicio y el tiempo de finalización de la primera tarea.

$$H_2 = \lambda_2 \sum_{\{c_1,c_2\}\in T_1} x_{c_1} x_{c_2}$$

$$: \{[i_1, j_1, t_1], [i_2, j_2, t_2]\} \in T_1 \Leftrightarrow \left\{ \begin{array}{l} i_1 \neq i_2 \wedge j_1 \neq j_2 \\ M_1 = M_2 \\ t_1 \leq t_2 < t_1 + d_1 \end{array} \right\}$$

- H3: Término de la restricción "Tasks subsecuentes (j y j+1) del mismo trabajo deben respetar el orden temporal". Se aplicará una penalización a cada pareja de variables activa si se cumplen las tres siguientes restricciones:
  - Son variables que corresponden al mismo job.
  - Son variables que corresponden a tasks subsecuentes (j y j+1).
  - El tiempo de inicio correspondiente a la variable de la task j+1 está entre el tiempo 0 y el tiempo de la finalización de la variable correspondiente a la task j.

$$H_3 = \lambda_3 \sum_{\{c_1,c_2\}\in T_2} x_{c_1} x_{c_2}$$

$$: \{[i_1, j_1, t_1], [i_2, j_2, t_2]\} \in T_2 \Leftrightarrow \left\{ \begin{array}{l} i_1 = i_2 \\ j_2 = j_1 + 1 \\ 0 \leq t_2 < t_1 + d_1 \end{array} \right\}$$

Comparativas

En la siguiente tabla se muestra el resumen de las diferentes pruebas que se detallan más adelante en el documento. Las columnas de la tabla son las siguientes:

- Size: "Número de jobs x Número de tasks", que equivale a "Número de jobs x Número de máquinas"
- d: Tiempo máximo considerado. Aumentarlo asegura que nuestra solución está dentro del espacio considerado, pero también aumenta el número de variables necesarias.
- N: Número de variables necesarias para resolver el problema después de haber retirado las opciones trivialmente irrelevantes (por ejemplo: la task número 3 no puede empezar en el tiempo 0 si hay tasks previas con duración no nula).
- L0: Coeficiente del término de reducción temporal ("cuanto menos tiempo, mejor"). Como sólo el tamaño relativo de los coeficientes es relevante, lo fijaremos en L0=1 para todos los casos.
- r: Exponente del tiempo en el término H0 del hamiltoniano ("cuanto menos tiempo, mejor").
- L1: Coeficiente de la restricción "todas las task deben ser realizadas en algún momento del tiempo".
- L2: Coeficiente de la restricción "diferentes trabajos que requieren la misma máquina no pueden coincidir en el tiempo".
- L3: Coeficiente de la restricción "Tasks subsecuentes (j y j+1) del mismo trabajo deben respetar el orden temporal".
- Shots: número de shots medidos en el ordenador cuántico (o simulador).
- Óptimo: ¿ha encontrado el proceso una de las soluciones óptimas? En caso negativo, puede ser bien por no haber ofrecido una solución legal (No Legal, N.L.), o bien por la existencia de una solución mejor (No Optimizado, N.O.).
- Layers: Número de layers usados en el QAOA.

Size	d	N	L0	r	L1	L2	L3	Shots	Layers	Óptimo?
2x2	3	4	1	1.5	4	4	2	10^5	4	Sí
3x3	4	12	1	1.5	8	8	2	10^5	4	Sí
3x3	4	16	1	1.5	8	8	2	10^5	4	Sí
3x3	4	16	1	1.7	4	4	2	10^5	4	N.L.
3x3	4	12	1	1.5	10	8	2	10^5	4	Sí
3x3	4	16	1	1.7	8	8	2	10^5	4	N.L.
3x3	4	12	1	1.5	10	8	2	10^5	2	N.O.
”	”	”	”	”	”	”	”	”	3	N.L.
”	”	”	”	”	”	”	”	”	4	Sí
3x3	4	12	1	1.5	10	8	2	10^4	4	N.O.
”	”	”	”	”	”	”	”	10^3	”	N.O.

Tabla 2. Comparativa de Resultados.

Conclusiones

Las conclusiones a las que hemos podido llegar son las siguientes:

- El valor de l0 no es tan relevante, al menos para estos tamaños de problema, ya que el espacio de posibles valores de coste no es excesivamente grande para las configuraciones válidas. Posiblemente en problemas más grandes su efecto sería mucho más relevante. Lo mismo aplica al r.
- El valor de l1 y l2 tiene que ser aproximadamente del mismo orden, para que no se superpongan dos tareas ni desaparezcan. Además, en general, su valor tendrá que crecer con el tamaño del problema, como se pudo observar comparando pruebas en 2x2 y 3x3 a la hora de ajustar. Esto es debido a que una incidencia ya hace que la solución no sea válida, por lo que su contribución mínima es independiente del tamaño del problema, mientras que la contribución del l0 aumenta con el tamaño del problema.
- El valor de l3 parece no ser dependiente del tamaño del problema, pero puede ser por el tamaño escogido.
- La dependencia con el número de ”shots” es visible, ya que en los casos de menos ”shots”, faltan tareas. Esto debe ser por la mala estimación del valor

esperado del coste en el entrenamiento. Aun así, podemos ver que completar las tareas que faltan es relativamente sencillo a simple vista.

- La dependencia con el número de capas es relevante, debido a que con un menor número no se consigue un buen cumplimiento de las restricciones.
- En los casos de coincidencia de tareas, vemos que podemos quedarnos con la tarea del job que está incompleto (que no ha realizado todas sus tareas) y obtenemos la configuración óptima. No es seguro que esta metodología pueda generalizarse, pero sería una línea de trabajo para una futura investigación.

En el documento sito en el repositorio del proyecto, "T3.1. Informe de Evaluación de los Modelos y Resultados de las Pruebas.docx", están descritas en detalle los resultados de las distintas pruebas realizadas sobre el servidor Finisterrae III de CESGA.

Referencias

[1] Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices (arxiv.org).

[2] Max-Cut and Traveling Salesman Problem — Qiskit Optimization 0.5.0 documentation