eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Ingeniería de Computadores

Trabajo de Fin de Grado

Resolución de problemas de planificación horaria usando computación cuántica adiabática

Autor

Aingeru Ramos Auzmendi

2022

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Ingeniería de Computadores

Trabajo de Fin de Grado

Resolución de problemas de planificación horaria usando computación cuántica adiabática

Autor

Aingeru Ramos Auzmendi

Director

Jose A. Pascual

Resumen

El objetivo de este proyecto es la utilización del computador cuántico adiabático de DWave para ejecutar el problema de programación lineal que la empresa Plannam ha cedido a la Facultad de Informática de San Sebastián para la elaboración de este proyecto. Concretamente, el problema que nos ataña trata sobre la asignación de tareas de un año a un conjunto de trabajadores teniendo en cuenta restricciones de tiempo y habilidad de cada trabajador.

Para ello, se hace un estudio de los pasos necesarios para resolver este tipo de problemas en el computador de DWave. En particular, se estudia el modelo de computación cuántica adiabática, los problemas de programación lineal (y algunas propiedades útiles para el caso que nos ocupa) y se explica el modelo Quadratic Unconstrained Binary Optimization (QUBO) que estas máquinas son capaces de ejecutar.

El análisis de los resultados muestran como el modelo desarrollado puede obtener resultados del problema de optimización si la configuración de la máquina de DWave es la correcta. También se muestran formas de como ir mejorando los modelos una vez conseguidos los primeros resultados e ir puliendo los valores clave para la ejecución correcta del problema. Por último, se proponen distintas líneas de investigación que aportarían nuevas formas de mejorar el modelo y conseguir que converja en una solución mejor con menor tiempo y esfuerzo.

Índice general

Índice de figuras

Índice de tablas

1. CAPÍTULO

Introducción

La mecánica cuántica a día de hoy esta más que aceptada por la comunidad científica y ha aportado luz a grandes misterios sobre el funcionamiento de las entidades más pequeñas de la naturaleza y los extraños fenómenos que los rodean. Sin embargo, esto no siempre fue así. Fue durante el siglo XX que apareció el primer indicio de este nuevo tipo de mecánica como una herramienta para predecir experimentos que, con la óptica clásica, no tenían sentido. Desde la conceptualización de que la energía no es continua, sino que tiene caracter discreto y cuantificable; de ahí el nombre cuánto, al entendimiento de como funcionan las superposiciones de las propiedades de un ente puramente cuántico.

No pocas han sido las controversias que ha causado esta mecánica debido a la interpretación probabilística de la realidad que muestra y su aparente total desentendimiento de la física desarrollada desde el siglo XVI que pretende usar el determinismo para describir la naturaleza. Incluso hoy se pueden oír científicos que siguen rechazando la interpretación de estas leyes. Pero a pesar de todo, la mecánica cuántica se ha demostrado capaz de superar obstáculos y erigirse como el conjunto de leyes que rigen el mundo en pequeñas escalas.

Muchas aplicaciones han surgido desde que comprendemos mejor las leyes más fundamentales de la naturaleza, pero la que más auge está teniendo (bajo la poca humilde opinión de un informático) a día de hoy es el desarrollo de la computación cuántica.

La primera vez que se oyó de este campo fue en 1981 en boca de Paul Benioff. Benioff expuso como, usando la superposición de las propiedades cuánticas, podíamos codificar información en ellos y computar para obtener resultados más rápido que ningún ordena-

dor clásico podría jamas. Aquí nació la conceptualización del cúbit (sobre su definición y propiedades se hablara en un apartado futuro). Después Richard Feynman postuló que gracias a los computadores cuánticos se podrían simular sistemas cuánticos de gran tamaño debido a la naturaleza cuántica del propio computador.

Mucho se ha avanzado desde entonces y aunque se empiezan a ver las primeras empresas especializadas en este campo y aplicaciones reales en funcionamiento, esta tecnología se encuentra en un estado fácilmente comparable a la era del comienzo de la computación. Una era en la que las máquinas tenían tamaños de habitaciones enteras y cada grupo de investigación proponía distintas formas de representar un bit o como operar con el.

En este proyecto nos centraremos en la propuesta de la empresa privada DWave. Su computador, al contrario de otros muchos, no es un computador general, por lo que no ejecuta cualquier algoritmo. Concretamente solo ejecuta uno, el *recocido cuántico* más reconocido por su nombre en inglés *quantum anneling*. Haciendo uso de esta tecnología trataremos de resolver un problema de planificación horaria propuesta por la empresa Plannam. Un problema de planificación horaria, sin entrar mucho en detalles dado que la descripción de este problema ocupa todo un capítulo del documento, es un problema de optimización cuya función objetivo y restricciones describen un modelo que pretende asignar trabajadores a tareas repartidas a lo largo de un periodo de tiempo definido, en nuestro caso deberemos asignar las tareas de todo un año a una plantilla de trabajadores cumpliendo restricciones de tiempo y habilidad de cada trabajador.

2. CAPÍTULO

Objetivos del Proyecto

El objetivo principal de este proyecto es el estudio de los problemas de optimización de planificación horaria, concretamente del problema de planificación de Plannam y él como poder ejecutarlo en un computador adiabático cuántico.

Para lograr el objetivo principal se deberá, en primer lugar, estudiar en profundidad el problema de planificación de Plannam. Desde el significado de cada restricción a él como se relacionan las variables entre sí para componer el problema.

Además, se deberá dedicar tiempo al análisis de la computación cuántica adiabática, tanto de su funcionamiento físico en detalle como de los modelos matemáticos que son usados para operar en estas máquinas (modelos BQM). Incluido está en este apartado el estudio de la reformulación de nuestro problema de planificación a estos modelos.

En último lugar, una vez desarrollado el modelo BQM, para el caso que nos ocupa un modelo QUBO, se deberán realizar distintas pruebas para valorar su eficacia a la hora de obtener resultados válidos del problema de planificación que buscamos resolver y si estos resultados son mejores que los que brinda la computación clásica. Esto nos llevará a analizar los resultados para, si fuera el caso, poder mejorar nuestro modelo e ir obteniendo mejores resultados de forma continua.

3. CAPÍTULO

Planificación

En el transcurso de cualquier proyecto, la gestión y planificación resultan indispensables a la hora de mantener una evolución y progreso del mismo, así como de cara a prevenir posibles riesgos. Por ello, se hace necesario dividir todo el trabajo en distintas fases bien diferenciadas entre sí. En este proyecto, las fases que se han determinado son las siguientes: gestión, documentación, desarrollo, evaluación y comunicación.

Además se han definido paquetes de trabajo dentro de cada una de las fases. Estos paquetes tratan de establecer aquellos elementos que determinan duración del proyecto y resultan la base para la planificación del mismo. En última instancia, cada paquete de trabajo estará compuesto por un conjunto de tareas a realizar que se describirán con posterioridad.

3.1. Diagrama EDT

Con el fin de mostrar la relación existente entre los paquetes de trabajo y la composición de los mismos se presenta un diagrama EDT (Estructura de Descomposición del Trabajo) en la Figura 3.1. Mediante el uso del EDT, se representa la descomposición de todo el trabajo que se debe llevar a cabo para cumplir los objetivos del TFG.

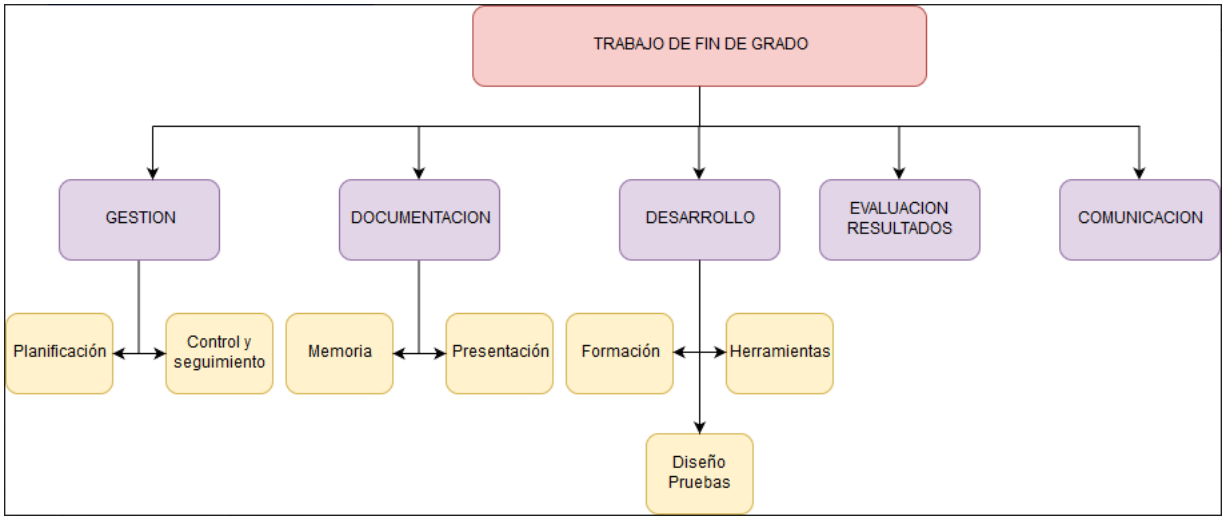

Figura 3.1: Diagrama EDT

3.2. Descripción de las fases que componen el EDT

A continuación, se proceden a explicar la descripción de las tareas contenidas en los paquetes de trabajo que componen el proyecto. Hay que mencionar que las tareas han sido diseñadas en el inicio del proyecto, durante la fase de planificación. A pesar de ello, se ha de recalcar que algunas de las tareas han aparecido durante el desarrollo del proyecto y que en un principio no fueron previstas.

3.2.1. Gestión

Esta fase transcurre a lo largo de todo el ciclo de vida del proyecto, concentrándose sobre todo en el primer mes de desarrollo del mismo. Ésto se debe a que un proyecto de éstas dimensiones requiere de una planificación detallada antes de empezar la fase de implementación, con el objetivo de dividir la carga de trabajo equitativamente a lo largo de todo el desarrollo.

A continuación se detallarán las tareas planificadas para la fase de gestión. Hay que remarcar que las tareas 1 y 2 se corresponden al paquete de trabajo *Planificación* y las tareas 3 y 4 al paquete de trabajo *Seguimiento y control*.

1. **Estudio de los objetivos:** Concretar de la manera más específica posible los obje-

tivos a cumplir por el proyecto. Definir bien estos objetivos ayudará a la identificación del trabajo a realizar a lo largo del ciclo de vida del proyecto.

2. **Diseño de los paquetes de trabajo y su coste:** Identificar los paquetes de trabajo que componen cada una de las fases del proyecto y realizar una estimación del coste temporal de los mismos. Además, hay que jerarquizarlos y clasificar cada uno de ellos en unidades más pequeñas, derivando en última instancia en tareas a desarrollar.

3. **Cálculo y análisis de las desviaciones:** Contabilizar las desviaciones entre el tiempo real y estimado de cada paquete. Llegado el momento del cierre del proyecto, se llevará a cabo un análisis del porqué de las desviaciones más significativas en caso de haberlas.

4. **Control del desarrollo del proyecto:** Llevar un control y seguimiento preciso del proyecto. Cada día que se dedique a desarrollar el proyecto, serán documentadas las horas invertidas en el paquete de trabajo en cuestión. De esta forma, se sabrá en todo momento el tiempo real dedicado a cada fase del proyecto. Además, esta tarea nos facilitará el trabajo a la hora de calcular y analizar las desviaciones.

3.2.2. Fase de documentación

Esta fase incluye la documentación del trabajo realizado a lo largo del ciclo de vida del proyecto. Para ello haremos uso de la herramienta *Overleaf*, clasificando la información correspondiente a cada fase del proyecto en diferentes ficheros. En el periodo final del proyecto, se procederá a la redacción del documento final mediante la unión de los diversos ficheros, dotando a la memoria un cierto orden y temporalidad. A continuación, se describen cada una de las tareas a implementar en esta fase. Hay que remarcar que la tarea 1 corresponde al paquete de trabajo *Memoria* mientras que las tareas 2,3 y 4 corresponden al paquete de trabajo *Presentación*.

1. **Documentación del proyecto:** Documentar en los ficheros de *Overleaf* cada uno de los paquetes de trabajo implementados además de una redacción formal del documento final.

2. **Hacer presentación:** Realizar la presentación del trabajo realizado. Tener en cuenta que la duración de la misma deberá ser de unos 15-20 minutos.

3. **Ensayar presentación:** Ensayo de la presentación con el director antes de la exposición del trabajo.

4. **Crear póster:** Crear un póster que resuma el contenido del TFG.

3.2.3. Fase de desarrollo

Esta fase es la de mayor duración, ya que va a requerir del aprendizaje de nuevas herramientas y tecnologías que son nuevas para mi.

A continuación, se proceden a describir cada una de las tareas a llevar a cabo en la fase de desarrollo. Decir que la tarea número 1 pertenece al paquete de trabajo *Formación*, mientras que la tarea 2 pertenece al paquete *Herramientas*. Por su parte las tareas 3 y 4 corresponden al paquete de trabajo *Diseño y pruebas*.

1. **Formación:** Esta tarea tiene como objetivo el aprendizaje de las tecnologías que se utilizarán y el entorno de trabajo donde se pondrán en marcha. Dentro de este paquete de trabajo realizaremos el estudio de diversos elementos.

    - **Estudio del computador cuántico:** Estudiar el funcionamiento del computador y sus peculiaridades.
    - **Estudio del modelo QUBO y transformación de problemas:** Estudiar el modelo matemático usado para describir los problemas y entender el cómo transformar los problemas de optimización a ese modelo.

2. **Herramientas:** Dentro de este paquete de trabajo se desarrollan las herramientas y entornos de los que se harán uso en esta fase.

    - **Familiarizarse con la API del computador cuántico:** Entender la comunicación con el computador cuántico mediante la librería de DWave.
    - **Familiarizarse con los nodos de la facultad:** Aprender el uso de los nodos de la facultad para optimizar el tiempo de calculo de los problemas.

3. **Desarrollo del código:** Tarea en la que se incluye la preparación del código que transforma los problemas de optimización al modelo QUBO y los ejecuta en el computador cuántico. A su vez, el programa guarda los resultados en disco en formato JSON para su posterior análisis.

4. **Ejecución de las pruebas:** Diseñar y ejecutar las pruebas para minimizar el tiempo de uso de la QPU y obtener los mejores resultados posibles.

3.2.4. Fase de evaluación de los resultados

En esta fase se analizan y documentan los resultados obtenidos por cada uno de los experimentos realizados. El paquete de trabajo *Evaluación de los resultados* se encuentra dividido en 2 tareas:

1. **Documentación de los resultados:** Descripción del entorno experimental que sera utilizado incluyendo las métricas que se emplearán para la evaluación de los resultados.
2. **Análisis de los resultados:** Análisis de los resultados con las métricas más importantes.

3.2.5. Comunicación

En esta fase se detallan las formas de comunicación que se utilizarán para transmitir el estado de desarrollo del proyecto. La fase en cuestión se encuentra compuesta por las siguientes tareas:

1. **Reuniones periódicas:** Con el fin de establecer la comunicación entre el alumno y el tutor se realizarán reuniones, normalmente de carácter mensual, de entre 30 y 45 minutos. En ellas se analizará en qué punto del proyecto nos encontramos y se fijarán nuevos objetivos a corto plazo. De esta forma, se tratará de conseguir equilibrar la carga de trabajo y hacer un seguimiento continuo por parte del tutor durante todo el ciclo de vida del proyecto.

3.3. Estimación y desviación de las tareas

Una vez explicadas las tareas que componen cada fase del proyecto, pasaremos a representar en la Tabla 3.1 la información referente al coste temporal estimado de cada una de las tareas y la información sobre el tiempo real invertido en cada tarea, identificando el paquete de trabajo al que pertenecen.

	Estimación (h)	Real (h)
Gestión	36	34
Planificación	19	19
Estudio objetivos	4	4
Diseño de los paquetes de trabajo	15	15
Control y seguimiento	17	15
Cálculo y análisis desviaciones	7	5
Control del desarrollo del proyecto	10	10
Documentación	100	105
Memoria	75	80
Documentar trabajo	75	80
Presentación	25	20
Hacer presentación	10	10
Ensayar presentación	10	5
Crear póster	5	5
Desarrollo	190	157
Formación	50	40
Estudio de políticas de scheduling	15	20
Estudio Multi-label	35	20
Herramientas	40	45
Estudio del simulador SSE	20	30
Estudio de Scikit	20	15
Diseño y pruebas	100	72
Desarrollo del framework	95	65
Diseño de los experimentos	5	7
Evaluación y resultados	20	17
Documentación de los resultados	10	8
Análisis de los resultados	10	30
Comunicación	4	5
Reuniones periódicas	4	5
TFG	350	334

Tabla 3.1: Tabla de desviaciones.

3.4. Plan de riesgo

En proyectos de una duración tan larga y con tantas tareas es necesario contar con un plan de riesgo para prevenir posibles desviaciones debidas a factores que son imprevisibles a priori. El desarrollo de este punto nos proporcionará una mayor flexibilidad para abordar circunstancias no previstas. Por lo tanto, para poder reaccionar ante retrasos y problemas que seguramente ocurrirán, se pone en marcha el siguiente plan de riesgo, el cual se tendrá en cuenta a lo largo del ciclo de vida del proyecto.

Con el fin de evitar problemas a la hora de cumplir con el plazo de entrega del proyecto, se ha previsto la finalización del mismo con anterioridad. En concreto, la fecha límite para entregar el proyecto es el día 6 de Febrero, fecha en la cual el director debe dar su visto bueno. Teniendo en cuenta esa fecha, la planificación de este proyecto ha sido diseñada para que la memoria esté lista y revisada para una fecha acordada con el director. En particular, se decidió el día 18 de Enero, 10 días antes de la fecha en la que el estudiante solicita la defensa y el director emite el informe favorable y 19 días hasta la entrega del proyecto.

Uno de los riesgos que más podrían lastrar la finalización del proyecto es el tiempo limitado de cómputo que tenemos en el computador cuántico. Al solo disponer de 4 horas de tiempo de cómputo en QPU, será necesario planificar las pruebas para usar el mínimo tiempo posible. La finalización de este recurso sin haber obtenido las suficientes pruebas podría equivaler a no poder terminar de forma óptima el proyecto, por lo que este riesgo tendrá mucho peso durante todo el proyecto, sobre todo en la fase de pruebas.

3.5. Metodología de trabajo

De cara a establecer una rutina a la hora del desarrollo del proyecto, se presenta a continuación una metodología de trabajo que se intentará cumplir desde el principio.

En mi caso particular, al no tener asignaturas que cursar durante el 1º cuatrimestre, puedo dedicarme completamente a la elaboración del TFG. Desde el día de inicio del proyecto, 11 del Octubre del 2021, hasta el día 6 de Febrero de 2022, he decidido dedicarle una media de 25 horas semanales (5 horas diarias de Lunes a Viernes); excepto en el periodo de Navidad si no ha habido muchas desviaciones.

En total se invertirán un total de 350 horas computadas hasta el día 18 de Enero. Esto nos permitirá dejar un colchón de días hasta el 6 de Febrero como se comenta en el plan de riesgo.

3.6. Análisis de las desviaciones

En general, se puede decir que dos han sido las principales factores que han producido la desviaciones más significativas respecto a las horas estimadas en los diferentes paquetes.

La primera desviación fue en el paquete de desarrollo del código. Encontrar una formulación general con la que modelar todas las restricciones del problema permitió crear una función general y ahorrar tiempo en el desarrollo de cada restricción. Esto supuso un ahorro de 30 horas en este paquete.

La segunda desviación se produjo en el paquete de pruebas. La inexperiencia en el uso de la máquina y correcciones del código produjo una desviación de horas invertidas que se propagó al paquete de documentación, retrasando la finalización de la memoria. Para solventar esta desviación, se usaron las horas de colchón que se mencionan en el plan de riesgo.

4. CAPÍTULO

Computación Cuántica Adiabática

El modelo de circuitos cuánticos es el que está recibiendo más atención por parte de diversas empresas como IBM, Google o Riggeti para construir un computador cuántico capaz de resolver problemas con aplicación real. Estos computadores están basados en puertas cuánticas, similares a las puertas lógicas que se usan en la computación tradicional, pero que en vez de usar lógica booleana, manipulan las superposiciones de los cúbits que pasan por esas puertas. Sin embargo, el computador que vamos a usar en este proyecto utiliza otro paradigma, el llamado *quantum anneling* o computación adiabática cuántica. Antes de hablar de este paradigma de computación, hay que entender algunos conceptos básicos como por ejemplo el cúbit que es la unidad mínima de información en este tipo de máquinas, y sus propiedades, que son las que brindan a estas máquinas su capacidad masiva de computación.

4.1. El Cúbit

Un bit es la unidad mínima de información. Los bits tienen dos estados: el 0 y el 1. Estos estados vienen representados por distintos niveles de voltaje. Los cúbits, al igual que los bits tienen dos estados; pero estos se representan con propiedades cuánticas tales como el *"spin"* de una partícula o el campo magnético de un superconductor en temperaturas cercanas al cero absoluto. Sin embargo, los cúbits al estar representados con estos fenómenos, la información que codificamos en ellos goza de propiedades cuánticas como la **superposición** y el **entrelazamiento**.

La superposición es el fenómeno por el cual un cúbit está en un estado indeterminado de sus dos estados básicos. Este estado se puede describir como la combinación lineal de sus estados básicos. Matemáticamente se describe así:

$$Q = a|0\rangle + b|1\rangle \tag{4.1}$$

Los coeficientes a y b son las amplitudes de probabilidad. Para saber con cuanta probabilidad se medirá cada estado en el cúbit, se hace el cuadrado de cada término. De esta manera, a^2 es la probabilidad de medir el estado $|0\rangle$ en el cúbit y b^2 la probabilidad de medir $|1\rangle$.

Como es evidente, ambos valores tienen que obedecer la expresión $a_2 + b_2 = 1$. Esto es porque la suma de la probabilidad de todos los sucesos posibles debe ser exactamente igual a 1. De esta forma, se puede definir el cúbit que tiene el 50 % de medir $|0\rangle$ y 50 % de medir $|1\rangle$ sería $1/\sqrt{2}|0\rangle + 1/\sqrt{2}|1\rangle$ o su forma simplificada $1/\sqrt{2}(|0\rangle + |1\rangle)$.

Un sistema de múltiples cúbits, supongamos 2 cúbits similares al de arriba descrito, se definiría como el producto tensorial de ambos cúbits:

$$1/\sqrt{2}(|0\rangle + |1\rangle) \otimes 1/\sqrt{2}(|0\rangle + |1\rangle) \tag{4.2}$$

Pero para verlo de forma más clara se escribe con la siguiente notación donde se perciben todos los estados posibles del sistema:

$$S = 1/2\sqrt{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \tag{4.3}$$

Si en este sistema midiésemos el primer cúbit, recibiríamos $|0\rangle$ o $|1\rangle$ con la misma probabilidad. Después de esta acción, medir el segundo cúbit sería totalmente igual que en el cúbit anterior, recibiríamos un resultado aleatorio. En este sistema los cúbits son independientes y no se afectan los unos a los otros.

Usando la superposición de cúbits se pueden crear sistemas de N cúbits donde se representan 2^N estados con los que operar todos a la vez. Es la superposición la que encierra la potencia de estas máquinas.

El entrelazamiento es un caso particular de superposición entre objetos cuánticos. Supongamos el siguiente sistema:

$$S = |10\rangle + |01\rangle \tag{4.4}$$

En este caso los cúbits están entrelazados. Si midiésemos el primer cúbit, automáticamente determinaríamos el estado del cúbit siguiente.

Otra forma de saber si un sistema de cúbits están entrelazados es intentando factorizar el sistema en sus cúbits fundamentales. Es decir, si ese sistema es posible construirlo a través del producto tensorial entre cúbits. Si no es el caso, el sistema es un sistema entrelazado.

4.2. Sistemas Adiabáticos

Para poder llegar a entender este paradigma de computación, primero hay que entender dos conceptos: como evoluciona un sistema físico y el efecto túnel.

- **Evolución de un sistema físico**: Todos los sistemas físicos tienden a sus estados de mínima energía. Como el caso de un objeto caliente, que de forma natural buscará su estado de mínima energía; es decir, se enfriará. De la misma forma, los sistemas cuánticos tienden a sus estados de mínima energía de forma natural. Sin embargo, este tipo de sistemas pueden llegar a estados de mínima energía en recorridos que si fueran analizados desde una perspectiva física clásica nunca podrían haber llegado. Este es el fenómeno conocido como el **efecto túnel**.

- **El Efecto Túnel**: Como se ha mencionado antes, en el mundo cuántico, las partículas tienen sus propiedades no están definidas y no es hasta que se miden que determinan esas propiedades. Esto permite situaciones raras como la siguiente:

  En la Figura 4.1 se puede ver una pelota en un mínimo local. La pelota podría estar en un estado menos energético si llegase al mínimo global. Sin embargo, tiene un

obstáculo; la montaña. Para poder pasarlo, según la física clásica, se le debería dar energía a la pelota para que pudiese superar esa barrera energética que le impide llegar a un estado menos energético. En la física cuántica no sucede lo mismo.

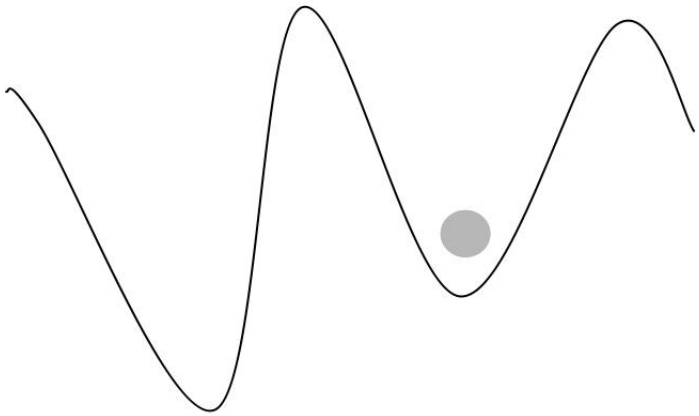

Figura 4.1: Pelota en un mínimo local de una función

Ahora en vez de una pelota tenemos una partícula (marcada con una P en la Figura 4.2) y su posición está indeterminada. En este estado la posición de la partícula está definida por una nube de probabilidad. La partícula es más probable que aparezca en el mínimo local (zona más oscura); sin embargo, aunque sea poco probable, también hay una pequeña posibilidad de que aparezca al otro lado del obstáculo y caer en el mínimo global (zona clara).

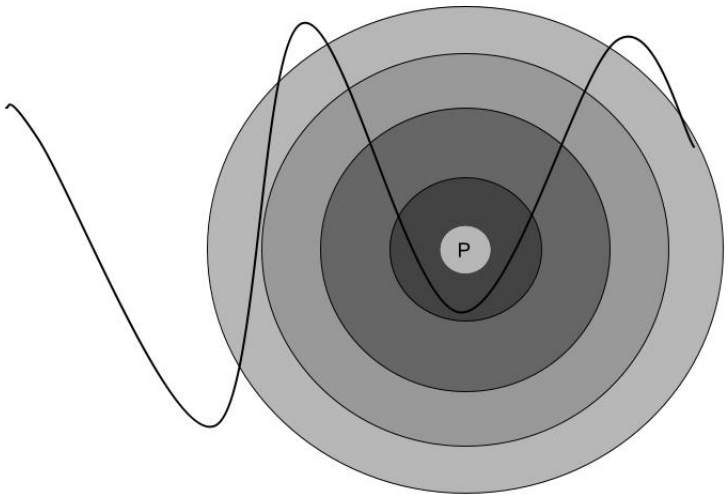

Figura 4.2: Partícula con posición indeterminada con su nube de probabilidad

Las máquinas que usan el adiabatismo cuántico explotan este fenómeno para poder llegar a estados de mínima energía en sistemas de cúbits.

Ahora imaginemos el siguiente escenario: Tenemos un sistema de múltiples cúbits. Cada combinación de valores de cúbits es un estado del sistema. Nosotros, como usuarios, podemos influir sobre la cantidad de energía que cada estado tiene en el sistema. Entonces podemos hacer que ciertas combinaciones de cúbits tengan energías menores al resto haciendo que el sistema sea favorable en caer en esos estados. ¿Y si hacemos que esos estados representen una solución a un problema de optimización de tipo minimización? Esa es precisamente la premisa de DWave.

4.3. Decoherencia

Los fenómenos cuánticos son difíciles de imaginar y cuesta más creer que leyes probabilísticas rijan el mundo macroscópico en el que vivimos; un mundo que nuestra física determinista describe con precisión milimétrica. Sin embargo, analizando más profundamente estos fenómenos uno llega a la conclusión que ambos modelos pueden convivir gracias a la decoherencia.

Resulta que los efectos cuánticos solo ocurren cuando las partículas se encuentran aisladas o en conjuntos pequeños de ellos. A mayores escalas, las partículas pierden las propiedades de superposición y quedan determinadas; funcionando como un objeto clásico. Este es actualmente el mayor problema para conseguir computadores cuánticos a gran escala y el objetivo de la siguiente generación de máquinas cuánticas: evitar la decoherencia.

4.4. DWave y su Computador

DWave es una empresa canadiense dedicada exclusivamente a la investigación y comercialización de computadores cuánticos. Al contrario que otras empresas, que intentan mediante puertas cuánticas lograr un computador cuántico general, DWave tras unos intentos poco exitosos, centró su investigación en este paradigma de computación adiabática. Tras esa decisión, la empresa ha conseguido grandes logros; como ofrecer al público el primer computador cuántico de 7000 cúbits.

4.4.1. Representación de Cúbits

En el computador de DWave los cúbits son representados por la dirección de la corriente y su respectivo campo magnético en anillos superconductores (Ver Figura 4.3).

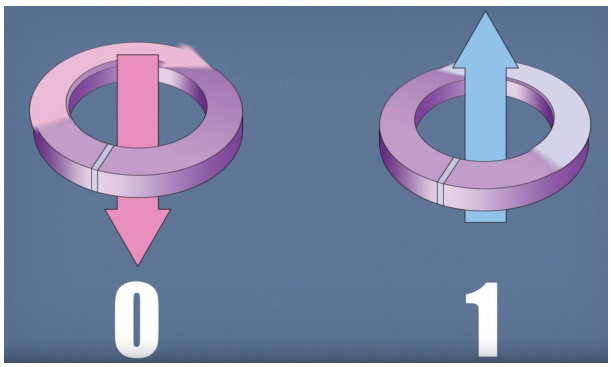

Figura 4.3: Diagrama de la representación de los cúbits de DWave

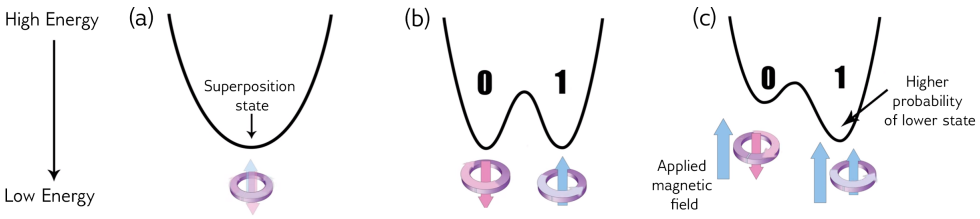

Figura 4.4: Evolución de cúbit en el sistema DWave

En la Figura 4.4 puede verse la evolución de un cúbit en el computador de DWave según su nivel de energía. Al comienzo, el cúbit se encuentra en estado de superposición. Los valles que representan cada estado están en el mismo valle energético, no están determinados, están superpuestos. Cuando el cúbit es medido los niveles energéticos de cada estado aparecen. En este caso los dos valles tienen el mismo nivel energético, por lo que el cúbit caerá equiprobablemente en el estado $|0\rangle$ o $|1\rangle$.

Pero podemos alterar la superposición si sometemos al cúbit a un campo magnético externo. De esta forma, al momento de la medición, el nivel energético del estado que contrafavorece ese campo magnético externo se vuelve más costoso, puesto que necesita más energía para mantenerse "a contracorriente" de la influencia externa, y favorecemos al estado que "sigue" esa influencia externa.

4.4.2. Arquitectura y Biases

Los campos magnéticos mencionados arriba son los que usamos como usuarios para manipular los estados energéticos de los estados del sistema de cúbits. Estos campos magnéticos son llamados *biases* y son estos los valores con los que modelamos los problemas dentro de la máquina. Por su parte, dentro del computador los cúbits están organizados siguiendo el grafo de la imagen 4.5:

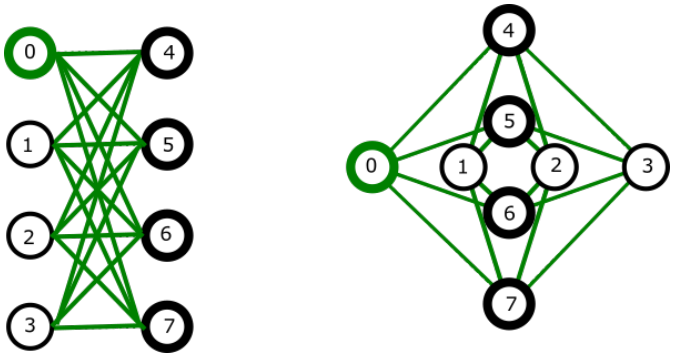

Figura 4.5: Grafo de conexión de los cúbits en el computador de DWave

Cada nodo del grafo es un cúbit y cada cúbit se relaciona con otros 4. Cada nodo tiene su propio *bias* al igual que cada conexión. Por lo que tenemos 2 tipos de *biases*, los lineales y los cuadráticos. La diferencia de estos dos se explica en el apartado BQM en el futuro.

4.4.3. Modelo Escalable

El grafo anterior es la mínima expresión de unidad de cómputo cuántico en DWave. Pero, ¿cómo consiguen escalar este modelo para crear computadoras de miles de cúbits? Para lograr ese objetivo se usan los *couplers*. Los *couplers* son cables que conectan distintas unidades de procesamiento haciendo que los cúbits externos se entrelazan para comportarse como si fueran el mismo. Estas conexiones son mostradas en la Figura 4.6

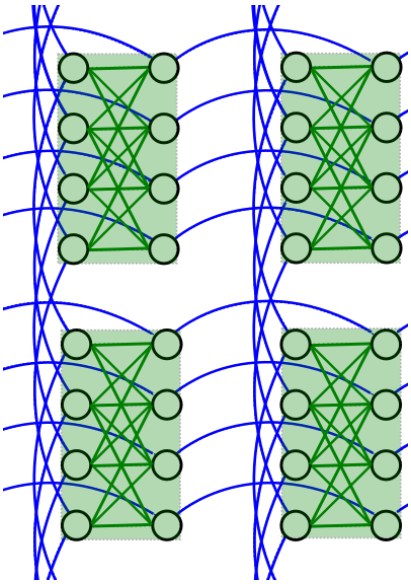

Figura 4.6: Conexión de *couplers*

De esta forma podemos mapear cúbits en múltiples cúbits si fuese necesario para que este se relacione con un cúbit lejano dentro del computador. Del mapeo de estos cúbits se encarga el propio sistema de DWave.

5. CAPÍTULO

Programación Lineal

La Programación Lineal (LP por sus siglas en inglés) es el área de la matemática que estudia la optimización (maximizar o minimizar) de funciones lineales sujetas, o no, a una serie de restricciones representadas mediante un sistema de ecuaciones. Por ello, la representación de un problema de programación lineal consta de:

- **Función objetivo**: Esta es la función que modela, usando variables independientes entre sí, el comportamiento de un sistema. El valor que devuelve dicha función es el valor que se busca optimizar. Dependiendo del problema que se quiera modelar, uno buscará el valor máximo o mínimo de dicha función. Por ello, los problemas de optimización se distinguen entre problemas de **maximización** o de **minimización**.
- **Restricciones**: Las restricciones son las ecuaciones o inecuaciones que limitan los valores de las variables. Una selección arbitraria de valores para las variables puede obtener, a priori, un valor óptimo en la función objetivo, pero si la combinación de ciertos valores no satisface una o varias restricciones la solución queda invalidada.

La forma general de describir matemáticamente estos problemas es la siguiente:

$$\begin{aligned} \text{minimize} \quad & z = \sum_{i=1}^{n} c_i * x_i \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{i,j} * x_j \leq b_i \quad i = 1, 2, 3, ..., p \end{aligned}$$

En el campo de la programación lineal (un subcampo de la optimización), todas las funciones, función objetivo y restricciones, deben de ser funciones lineales.

5.1. Propiedades

Muchas veces nos encontraremos que los problemas de optimización tienen formatos que no son los adecuados a nuestras necesidades y requerirán de transformaciones para poder operar con ellos. Durante este proyecto se han usado dos propiedades o técnicas para adaptar los modelos de optimización.

- **De MAX a MIN y viceversa**: Como se comentó en el apartado de las máquinas de DWave, las funciones que puede optimizar el computador son problemas de minimización. Esto podría resultar un problema si no fuese porque las funciones objetivo pueden cambiar su forma de maximización a una de minimización y viceversa usando la siguiente expresión:

$$max \quad (\sum_{i=0}^{N} c_i x_i) = min \quad (-\sum_{i=0}^{N} c_i x_i) \tag{5.1}$$

  Este cambio en el signo de la función objetivo no altera el valor absoluto de los resultados, dado que lo que antes era un máximo global (la solución a hallar) en la función de maximización, ahora es el mínimo global en la función de minimización.

- **Inecuaciones a Ecuaciones**: Mucas veces las restricciones no son igualdades, sino que vienen definidas como desigualdades. Para poder transformar una desigualdad a una igualdad, se añade una variable auxiliar que llamaremos *"slack variable"*. Con ella podemos trasformar las restricciones en forma de inecuaciones en ecuaciones.

$$x_0 + x_1 < 3 \tag{5.2}$$

  A esta otra:

$$x_0 + x_1 + s = 3 \tag{5.3}$$

Si la inecuación en vez ser de tipo *"menor que"* fuera de tipo *"mayor que"* la variable s estaría restando en la ecuación final.

6. CAPÍTULO

Binary Quadratic Model

Como se ha mencionado antes, la forma de modelar nuestros problemas dentro del computador cuántico es usando campos magnéticos que afecten a los cúbits. A estos campos magnéticos se les llama *biases* y ya se mencionaron en apartados anteriores explicando su utilidad. Para describir como deben de ser esos *biases* se usa el modelo BQM. El modelo BQM (Binary Quadratic Model en inglés) se basa en el uso de funciones cuadráticas multivariables, donde cada una de las variables obtiene su valor de un conjunto binario de valores. Los dos conjuntos binarios que usan las máquinas D-Wave son los conjuntos {-1,+1} y {0,1}.

6.1. Formulación General

Las funciones cuadráticas son aquellas que cumplen la siguiente ecuación:

$$\sum_{i=0}^{N} a_i x_i + \sum_{i<j} b_{i,j} x_i x_j + \sum_{i=0}^{N} c_i x_i^2 \qquad \forall a_i, b_{i,j}, c_i \in R \tag{6.1}$$

Pero debido a los dos conjuntos binarios que usan en DWave los términos al cuadrado se suprimen, por lo que el modelo BQM es:

$$\sum_{i=0}^{N} a_i x_i + \sum_{i<j} b_{i,j} x_i x_j \qquad \begin{array}{l} \forall a_i, b_{i,j} \in R \\ \forall x_i \in \{b_0, b_1\} \end{array} \tag{6.2}$$

Los términos lineales son los que describen los *biases* lineales que afectan a cada cúbit de forma individual. Los términos cuadráticos son los *biases* cuadráticos que ponderan la fuerza con la que dos cúbits se relacionan dentro del problema.

Cambiando el tipo de conjunto binario usado para las variables de la ecuación se obtienen los dos modelos que entran dentro del modelo BQM de D-Wave: ISING y QUBO.

6.2. ISING

En este proyecto no se trabaja con el modelo ISING por lo que no entraremos de forma detallada a explicar el uso y funcionamiento de este modelo. Lo único a destacar es que el conjunto binario que se usa en este modelo es el conjunto {-1,+1}. Esto es debido a que este modelo pretende describir los valores de los cúbits como si fueran ”spines” de partículas. Este modelo es usado mayormente para describir problemas físicos y mecánica estadística; un campo que se aleja enormemente de nuestro campo de estudio.

6.3. QUBO

Este modelo es el que más se asemeja a la computación tradicional dado que su conjunto binario es el clásico {0,1}. Usando este conjunto con la expresión general se obtiene:

$$\sum_{i=0}^{N} a_i x_i + \sum_{i<j} b_{i,j} x_i x_j \qquad \begin{array}{l} \forall a_i, b_{i,j} \in R \\ \forall x_i \in \{0, 1\} \end{array} \tag{6.3}$$

Debido al uso del conjunto binario {0, 1} los términos lineales se pueden describir también como el cuadrado de la misma variable.

$$x_i = x_i^2 \qquad \forall x_i \in \{0,1\} \tag{6.4}$$

Por lo que podemos simplificar la ecuación a su forma más reducida:

$$\sum_{i \leq j} x_i Q_{i,j} x_j \qquad \begin{array}{l} \forall Q_{i,j} \in R \\ \\ \forall x_i \in \{0,1\} \end{array} \tag{6.5}$$

De esta expresión se puede ver como podemos representar la función como una matriz triangular superior Q donde se almacena cada valor $Q_{i,j}$ teniendo en cuanta el índice de las variables a las que referencia.

$$Q = \begin{pmatrix} Q_{0,0} & Q_{0,1} & Q_{0,2} \\ 0 & Q_{1,1} & Q_{1,2} \\ 0 & 0 & Q_{2,2} \end{pmatrix} \tag{6.6}$$

Sin embargo, por claridad, usaremos la expresión de la Ecuación 6.3 donde los términos lineales son los valores de la diagonal principal y los valores cuadráticos los valores del resto de la matriz. Este es el modelo que usaremos para escribir el problema dado a la máquina cuántica de DWave

7. CAPÍTULO

Transformación a QUBO

En este apartado se explican los pasos a seguir para transformar cualquier problema de programación lineal a una versión QUBO que pueda usar la máquina cuántica de DWave. Para simplificar la explicación asumiremos que todas las variables del problema dado son binarias. Más adelante se explicará como lidiar con variables no binarias.

7.1. Primeros pasos

Tomemos como expresión general de un problema de optimización esta expresión de aquí:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} c_i * x_i \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{i,j} * x_j \leq b_i \quad i = 1,2,3,...,p \end{aligned}$$

En primer lugar, se trasforma la función objetivo. La forma de proceder puede ser de dos formas dependiendo del tipo de función objetivo. Si la función objetivo es una función de minimización no es necesario hacer nada; ya está en su formato QUBO. De lo contrario, si la función objetivo es una función de maximización, usando las propiedades vistas en el apartado de Programación Lineal, podemos invertir el tipo de optimización.

En nuestro caso la función objetivo terminaría con la siguiente forma:

$$\text{minimize} \quad \sum_{i=1}^{n} -c_i * x_i$$

7.2. Funciones de penalización

La forma de plantear las restricciones es más elaborada. Estas son concebidas como funciones de penalización.

Una función de penalización es aquella función que devuelve sus valores mínimos cuando el valor de las variables dentro de la restricción cumplen con la restricción. Si las variables se alejan del valor óptimo para cumplir la restricción, esta devolverá un valor más elevado. Este valor será la penalización que aplicaremos cuando las restricciones no se cumplan, haciendo menos óptimo esa solución. Debido a esta propiedad que deben de cumplir, las funciones de penalización son funciones parabólicas

Los pasos para crear una función de penalización de una restricción son simples, pero difieren mínimamente dependiendo del tipo de restricción.

7.2.1. Función de Penalización: Ecuaciones

Dado una ecuación como la siguiente y llevando el término independiente al lado de las variables:

$$\sum_{i=0}^{N} a_i x_i = d \tag{7.1}$$

$$\sum_{i=0}^{N} a_i x_i - d = 0 \tag{7.2}$$

Podemos obtener la función parabólica; que será la función de penalización de la restricción; simplemente haciendo el cuadrado de la ecuación.

$$fp(x) = (\sum_{i=0}^{N} a_i x_i - d)^2 \tag{7.3}$$

7.2.2. Función de Penalización: Inecuaciones

Dado una inecuación como la siguiente:

$$\sum_{i=0}^{N} a_i x_i < d \tag{7.4}$$

Transformamos la inecuación en una ecuación usando las propiedades vistas en el apartado PL. Del como se representa la nueva variable con cúbits se habla a posteriori.

$$\sum_{i=0}^{N} a_i x_i + s_i = d \tag{7.5}$$

Y procediendo como en el caso anterior, creamos la función parabólica que será nuestra función de penalización.

$$fp(x) = (\sum_{i=0}^{N} a_i x_i + s_i - d)^2 \tag{7.6}$$

7.3. Expresión QUBO

Sumando todo lo anterior obtenemos lo siguiente:

$$Z = -\sum_{i=1}^{N} c_i * x_i + \sum_{i=1}^{N} fp_i() \tag{7.7}$$

El primer sumatorio expresa la función objetivo en su forma de minimización y el segundo sumatorio expresa la suma de todas las funciones de penalización de cada restricción del problema.

Si algunas variables no respetan las restricciones, las funciones de penalización devolverán valores altos perjudicando al valor Z final. En definitiva, la solución óptima es aquella que menor valor Z obtenga de la función superior, pues esta minimizara la función objetivo y minimizará el valor de las funciones de penalización.

Para controlar el nivel de penalización que aportan las penalizaciones se puede multiplicar por un factor. A este factor se le llama el factor de Lagrange. Así, la ecuación QUBO de todo problema de optimización (maximización) sería de la siguiente forma:

$$Z = -\sum_{i=1}^{N} c_i * x_i + \delta \sum_{i=1}^{N} fp_i() \tag{7.8}$$

7.4. Representación de valores no binarios en QBits

En todo el ejemplo del apartado anterior hemos usado variables binarias, por lo que no era necesario más que usar un cúbit para representar cada variable. Pero a la hora de enfrentar problemas reales, nos encontraremos con variables no binarias, tales como enteros o decimales. ¿Como representamos esos valores con cúbits?.

Pongamos el ejemplo de un byte. En computación clásica, un byte es representado con 8 bits. Cada bit aporta al valor 2^i, siendo i la posición del bit dentro de la representación.

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
128	64	32	16	8	4	2	1

Matemáticamente puede describirse de la siguiente forma.

$$128b_7 + 64b_6 + 32b_5 + 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0 = \sum_{i=0}^{7} 2^i b_i \tag{7.9}$$

Usando la representación matemática arriba descrita podemos introducir valores enteros en los problemas QUBO. Lo único que cambia es que en vez de usar bits tradicionales; b_i, usamos cúbits; q_i

8. CAPÍTULO

Descripción del problema

Este apartado está dirigido exclusivamente a la explicación del problema dado para este proyecto. El problema con el que se ha trabajado en este proyecto es un problema de optimización dirigido a la planificación horaria; es decir, el problema de optimización pretende asignar trabajadores a ciertas tareas establecidas de forma que la productividad de la plantilla sea la máxima posible.

8.1. Variables

Las variables de este problema son de la forma $x_{t,w}$. Estas son variables binarias y expresan que el trabajador w hace la tarea t

8.2. Parámetros

Los parámetros usados para describir las funciones que completan el problema son:

- h_t: Horas que dura la tarea t
- jp_t: La tarea t es una jornada partida
- $sx_{t,s}$: La tarea t se encuentra en la semana s
- $dx_{t,d}$: la tarea t se encuentra en el día d

- HA_w: Máximo de horas anuales del empleado w
- HS_w: Máximo de horas semanales del empleado w
- ST_w: Máximo de días trabajando en una semana para el empleado w
- $p_{t,w}$: El trabajador w puede tiene las habilidades para realizar la tarea t

8.3. Función Objetivo

La función objetivo es una función de maximización. Se puede ver que el objetivo de esta función es maximizar la productividad de la plantilla haciendo las máximas horas posibles.

$$\text{maximize} \quad \sum_{t=1}^{T} \sum_{w=1}^{W} h_t * x_{t,w}$$

8.4. Restricciones

A continuación se muestra una lista de las restricciones del problema explicando brevemente la funcionalidad de cada restricción.

- **Restricción 1**: Cada tarea t_j puede ser realizada por un solo empleado

$$\sum_{w=1}^{W} x_{t,w} \leq 1 \quad \forall t \in T \tag{8.1}$$

- **Restricción 2**: Cada empleado w puede realizar como máximo una tarea al día.

$$\sum_{t=1}^{T} dx_{t,d} * x_{t,w} \leq 1 \quad \forall d \in D, \forall w \in W \tag{8.2}$$

- **Restricción 3**: Cada empleado w puede realizar tareas que no superen las HA_w horas en un año.

$$\sum_{t=1}^{T} h_t * x_{t,w} \leq HA_w \quad \forall w \in W \tag{8.3}$$

- **Restricción 4**: Cada empleado w puede realizar tareas que no superen las HSw horas en una misma semana.

$$\sum_{t=1}^{T} sx_{t,s} * h_t * x_{t,w} \leq HS_w \quad \forall w \in W, \forall s \in S \tag{8.4}$$

- **Restricción 5**: Cada empleado w puede realizar hasta S_w tareas en una misma semana.

$$\sum_{t=1}^{T} sx_{t,s} * x_{t,w} \leq S_w \quad \forall w \in W, \forall s \in S \tag{8.5}$$

- **Restricción 6**: Cada empleado w no puede realizar más de una jornada partida en la misma semana.

$$\sum_{t=1}^{T} sx_{t,s} * jp_t * x_{t,w} \leq 1 \quad \forall w \in W, \forall s \in S \tag{8.6}$$

- **Restricción 7**: Cada empleado w solo puede realizar tareas en las que tiene la habilidad requerida.

$$\sum_{t=1}^{T} (1 - p_{t,w}) * x_{t,w} = 0 \quad \forall w \in W \tag{8.7}$$

8.5. Transformación

En primer lugar, se ha transformado la función objetivo a la forma de minimización quedando de la siguiente manera:

$$\text{minimize} \quad \sum_{t=1}^{T} \sum_{w=1}^{W} -h_t * x_{t,w}$$

En cuanto a las funciones de penalización se ha seguido un proceso más elaborado para ganar tiempo y facilidad en la programación. Si uno se fija en las restricciones puede percatarse que todas las restricciones siguen el siguiente patrón.

$$\sum_{i=1}^{\alpha} m_i x_i \leq d \tag{8.8}$$

Pero para simplificar los siguientes cálculos simplificamos con $m_i x_i = Y_i$, obteniendo:

$$\sum_{i=1}^{\alpha} Y_i \leq d \tag{8.9}$$

Siguiendo el proceso explicado en el apartado donde se explican la obtención de las funciones de penalización se consigue:

$$(\sum_{i=1}^{\alpha} Y_i + s + (-d))^2 = 0 \tag{8.10}$$

La variable s es nuestra variable "slac" que nos permite pasar de una inecuación a ecuación. Desarrollando la expresión obtenemos:

$$(\sum_{i=1}^{\alpha} Y_i)^2 + \sum_{i=0}^{\alpha} 2sY_i - \sum_{i=0}^{\alpha} 2dY_i + s^2 - 2sd + d^2 \tag{8.11}$$

Finalmente, desarrollando el primer sumatorio tenemos la expresión:

$$\sum_{i=1}^{\alpha} Y_i^2 + \sum_{i=0}^{\alpha} \sum_{j=i+1}^{\alpha} 2Y_i Y_j + \sum_{i=0}^{\alpha} 2sY_i - \sum_{i=0}^{\alpha} 2dY_i + s^2 - 2sd + d^2 \tag{8.12}$$

Rehaciendo la expresión $Y_i = m_i x_i$ introducimos de nuevo nuestras variables. Pero en vez de usar la notación x_i usaremos q_i, denotando que nuestras variables ya son cúbits. Por lo que queda:

$$\sum_{i=1}^{\alpha} m_i^2 q_i + \sum_{i=0}^{\alpha} \sum_{j=i+1}^{\alpha} 2m_i m_j q_i q_j + \sum_{i=0}^{\alpha} 2sm_i q_i - \sum_{i=0}^{\alpha} 2dm_i q_i + s^2 - 2sd + d^2 \tag{8.13}$$

Simplificando sumatorios se obtiene:

$$\sum_{i=1}^{\alpha}(m_i^2 - 2dm_i)q_i + \sum_{i=0}^{\alpha}\sum_{j=i+1}^{\alpha} 2m_i m_j q_i q_j + \sum_{i=0}^{\alpha} 2sm_i q_i + s^2 - 2sd + d^2 \tag{8.14}$$

Para terminar debemos representar nuestra variable s en una variable compuesta por cúbits. Estas variables auxiliares tienen la propiedad de solo poder obtener valores del rango entre [0, d], por lo que se ha optado por el siguiente formato.

$$s = \sum_{k=0}^{n} 2^k s_k, \quad n = \lceil log_2 d \rceil - 1 \tag{8.15}$$

Donde s_k es cada cúbit que representa a la variable auxiliar. De esta forma introducimos las variables en las ecuaciones y minimizaremos el uso de cúbits a los estrictamente necesarios. Sustituyendo esto último en la ecuación y simplificando los sumatorios resultantes queda la ecuación final que buscábamos.

$$\begin{aligned} &\textstyle\sum_{i=1}^{\alpha}(m_i^2 - 2dm_i)q_i + \sum_{i=0}^{\alpha}\sum_{j=i+1}^{\alpha} 2m_i m_j q_i q_j + \sum_{i=0}^{\alpha}\sum_{k=0}^{n} m_i 2^{k+1} q_i s_k \\ &+ \textstyle\sum_{k=0}^{n}(2^{2k} - d2^{k+1})s_k + \sum_{k=0}^{n}\sum_{t=k+1}^{n} 2^{k+t+1} s_k s_t + d^2 \end{aligned} \tag{8.16}$$

De esta última expresión, modificando los valores de α, el vector de valores m y el valor d, podemos obtener la función de penalización de cualquiera de las restricciones descritas arriba, ahorrando tiempo en el cálculo de cada función y pudiendo implementar una función general dentro del programa, facilitando su uso.

9. CAPÍTULO

Análisis de resultados

A continuación se muestra el análisis de los resultados con la implementación realizada. La evaluación se ha dividido en dos partes: la parte de preprocesado, donde se ha medido el tiempo requerido para transformar un problema de programación lineal a uno QUBO, y la parte de ejecución en el computador cuántico, que medirá su capacidad para obtener un resultado óptimo del problema.

9.1. Preprocesado

Como se ha comentado a lo largo del documento, los problemas de optimización requieren ser trasformados a un modelo QUBO para poder ser ejecutados en el computador cuántico. Obviamente, esto añade un tiempo de preprocesado a todos los problemas. En la Figura 9.1 se muestra el coste temporal de este preprocesado según el número de cúbits que necesite el problema para ser representado.

Como se puede ver el coste del preprocesado es lineal. Teniendo en cuenta que esta es la parte que es ejecutada en el computador clásico se puede decir que el algoritmo implementado para realizar la transformación es lo suficientemente eficiente como para ejecutar problemas de gran tamaño.

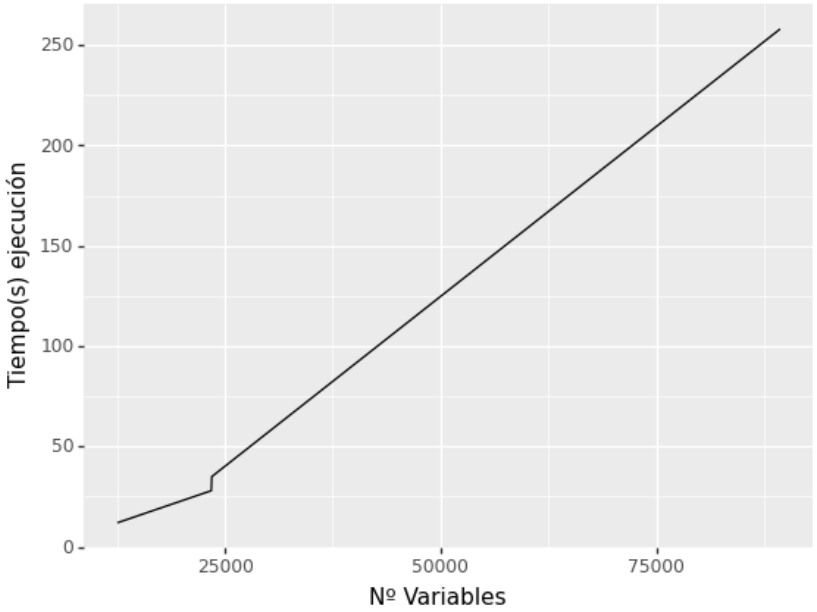

Figura 9.1: Tiempo de preprocesado (segundos) respecto número de cúbits

9.2. Procesado en QPU

Esta es la parte más delicada de la ejecución de las pruebas. Aunque se haya desarrollado un modelo QUBO que modele el problema a resolver, la configuración de la máquina de DWave puede alterar la eficiencia del modelo en la QPU.

Los dos parámetros para especificar la configuración de la máquina de DWave serán: el factor de Lagrange y el tiempo de ejecución en QPU. Para un correcto funcionamiento de la QPU, estos dos valores deben ser muy bien medidos y requiere de un proceso de prueba y error para poder llegar a ellos. Aunque sea un proceso de prueba y error, de las sucesivas ejecuciones realizadas, se puede obtener información que nos dirija a los valores óptimos de esos parámetros.

A continuación se muestra el proceso que se ha seguido hasta obtener una configuración óptima para el modelo que estamos ejecutando. Durante todo este proceso se ha usado como problema de prueba el set de datos más pequeño que se nos ha proporcionado, el que llamaremos a partir de ahora el problema ”14”, por el *id* del set de datos.

Como primer paso se ejecutó el modelo QUBO con la siguiente configuración para obtener los primeros resultados y tener un primer vistazo de como funciona el modelo en la práctica.

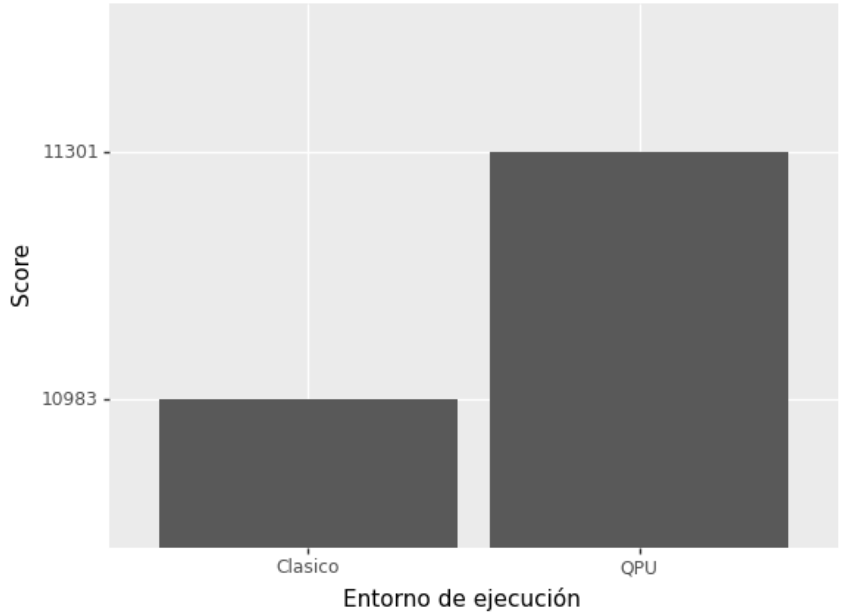

Figura 9.2: Score del problema 14 con FL:10 y TE:600

- **Factor de Lagrange**: 10 para todas las restricciones del problema.
- **Tiempo en QPU**: 600s (10 minutos)

Los resultados de dicha prueba se muestran en las Figuras 9.2 y 9.3.

Como se puede observar en la gráfica de la Figura 9.2, el *score*, es decir, el valor que nos devuelve la función objetivo, es superior a la que nos devuelve la versión clásica del problema. Sin embargo, la gráfica de la Figura 9.3 nos muestra que la solución de la QPU incumple muchas de las restricciones del problema.

Analizando la Figura 9.3 se observa que hay restricciones que son incumplidas con mayor facilidad que otras, por lo que uno pudiera pensar que las funciones de penalización que modelan esas restricciones no son lo suficientemente fuertes. Debido a eso, se probó la siguiente configuración:

- **Factor de Lagrange**: 10, excepto restricciones 1, 2 y 6 que será de un valor de 100.
- **Tiempo en QPU**: 600s (10 minutos)

En esta prueba se obtuvo un resultado similar al anterior. Dado que los factores de Lagrange no parecían surtir efecto, se remodeló las restricciones 1, 2 y 6 para poder aumentar

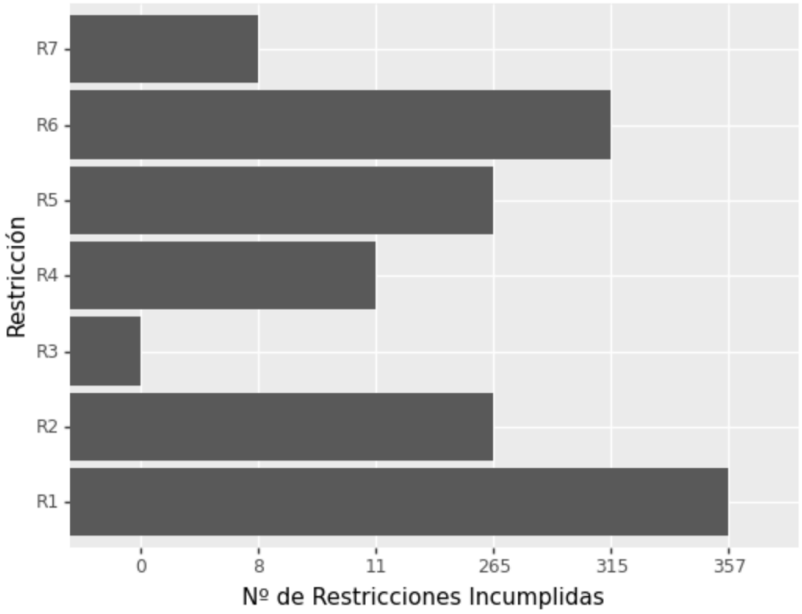

Figura 9.3: Desglose de restricciones incumplidas del problema 14 con FL:10 y TE:600

su fuerza dentro del computador cuántico. Esta idea llevó a sustituir las restricciones por estas otras:

- **Restricción 1:**

$$\sum_{w=1}^{W} h_t * x_{t,w} \leq h_t \quad \forall t \in T \tag{9.1}$$

- **Restricción 2:**

$$\sum_{t=1}^{T} dx_{t,d} * h_t * x_{t,w} \leq max(D \cdot T) \quad \forall d \in D, \forall w \in W \tag{9.2}$$

- **Restricción 6:**

$$\sum_{t=1}^{T} sx_{t,s} * jp_t * h_t * x_{t,w} \leq max(S \cdot JP \cdot T) \quad \forall w \in W, \forall s \in S \tag{9.3}$$

Con las nuevas restricciones y sus nuevas funciones de penalización, la configuración que se eligió para la prueba fue:

- **Factor de Lagrange**: 10 para todas las restricciones
- **Tiempo en QPU**: 600s (10 minutos)

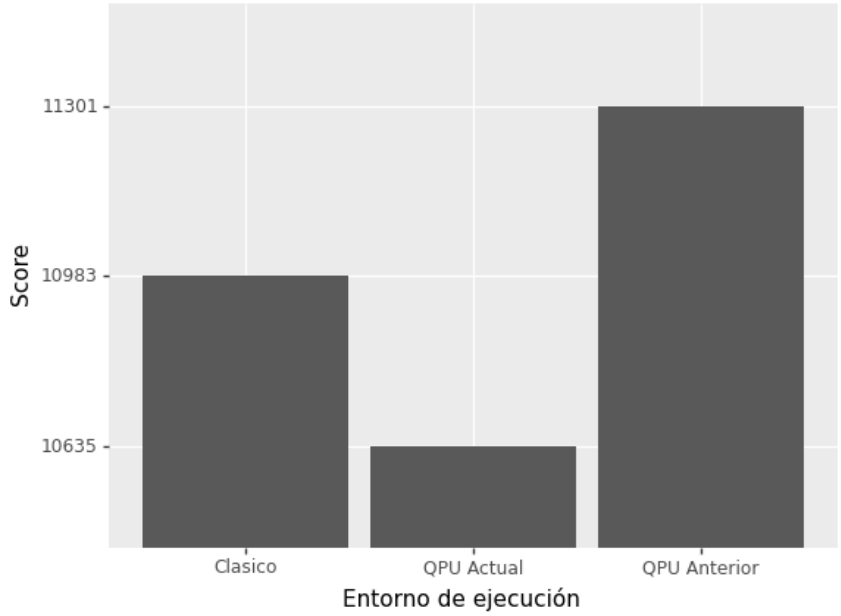

Figura 9.4: Score del problema 14 (con restricciones nuevas) con FL:10 y TE:600

Esta prueba devolvió el resultado de las Figuras 9.4 y 9.5.

Se puede ver como el "score" ha decrecido, incluso por debajo del resultado de la versión clásica, pero se ha conseguido que las restricciones 1 y 2 sean cumplidas. En cuanto la restricción 6, se le aumentó el factor de Lagrange y fue aumentado el tiempo de ejecución en QPU para observar si el mayor tiempo y penalización hacía que la restricción pesase más en el resultado final, por lo que la configuración quedó de esta manera:

- **Factor de Lagrange**: 10, excepto para la restricción 6 que será de valor 100.
- **Tiempo en QPU**: 1200s (20 minutos)

Esta configuración no devolvió un resultado mejor. De la misma forma que la vez anterior, el factor de Lagrange no es suficiente para hacer converger a la restricción R6.

Tras esta prueba, se ejecutó el programa para observar como se comportaba la restricción 6 de forma individual con la misma configuración. En esta prueba se comprobó como la función de penalización de la restricción 6 no se comporta de forma óptima. Incluso repitiendo la prueba con un factor de Lagrange superior no se solventa el problema.

Como últimas pruebas se ejecutaron el set de datos más pequeño (problema 14) y el set de datos más grande (problema 31) sin la restricción 6. Obviamente, los "scores" en estas

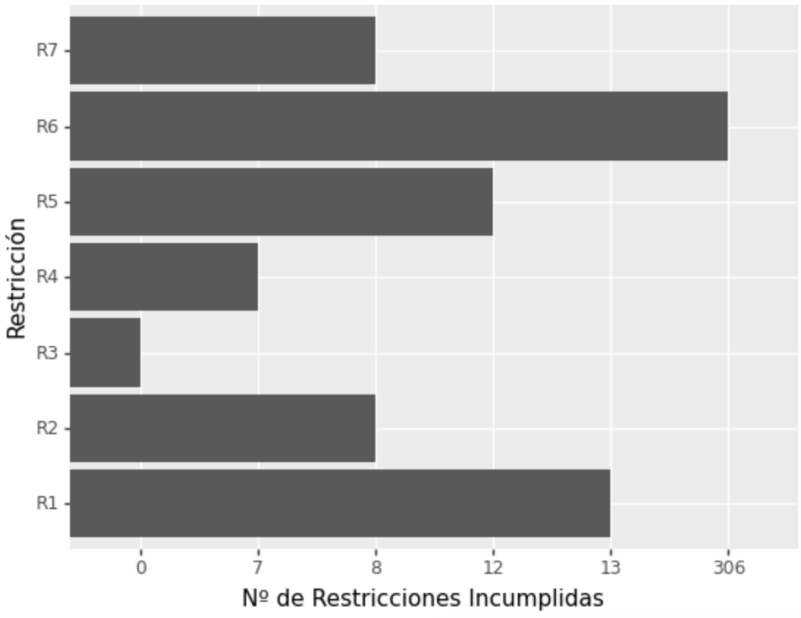

Figura 9.5: Desglose de restricciones incumplidas del problema 14 (con restricciones nuevas) con FL:10 y TE:600

pruebas no tienen valor, solo nos fijaremos en el número de restricciones incumplidas. Los resultados son los siguientes:

Las Figuras 9.6 y 9.7 muestran los resultados de los problemas 14 y 31 con la siguiente configuración.

- **Factor de Lagrange**: 10

- **Tiempo en QPU**: 1200s (20 minutos)

Se puede apreciar como el problema 31, al tener mayor número de variables, con el mismo tiempo de ejecución se obtienen peores resultados. Para solucionarlo se ha aumentado el factor de Lagrange de 10 a 100. El resultado se muestra en la figura 9.8

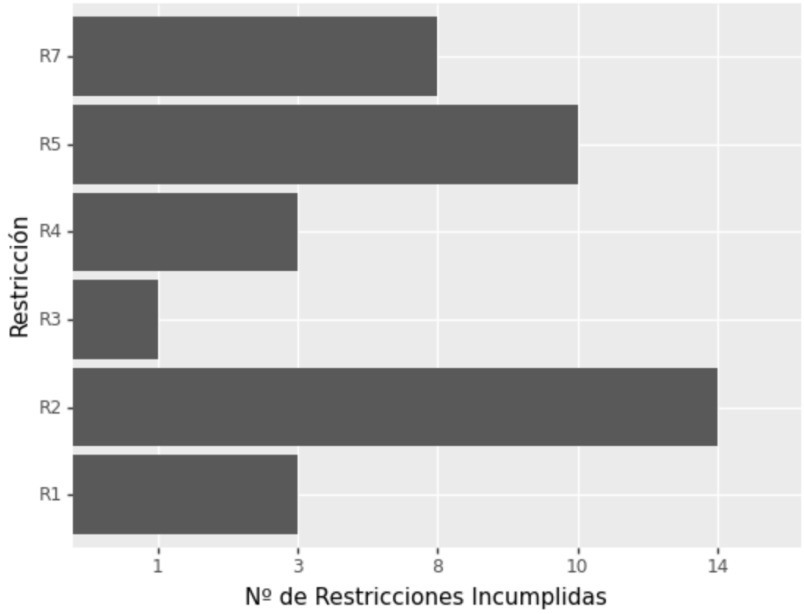

Figura 9.6: Desglose de restricciones incumplidas del problema 14 (con restricciones nuevas) con FL:10 y TE:1200

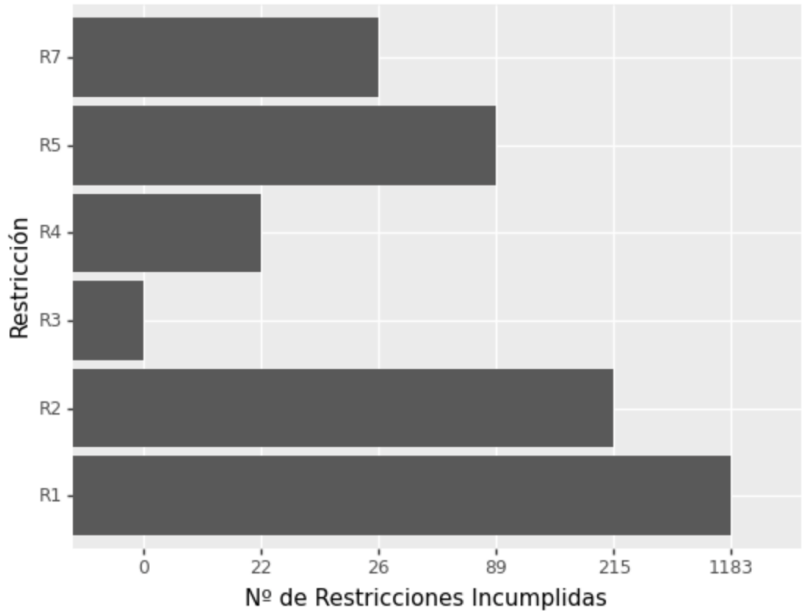

Figura 9.8: Desglose de restricciones incumplidas del problema 31 (con restricciones nuevas) con FL:100 y TE:1200

Respecto al resultado anterior, aunque en la restricción 1 haya aumentado el número de restricciones incumplidas, hay una mejora general en el número total de restricciones cumplidas.

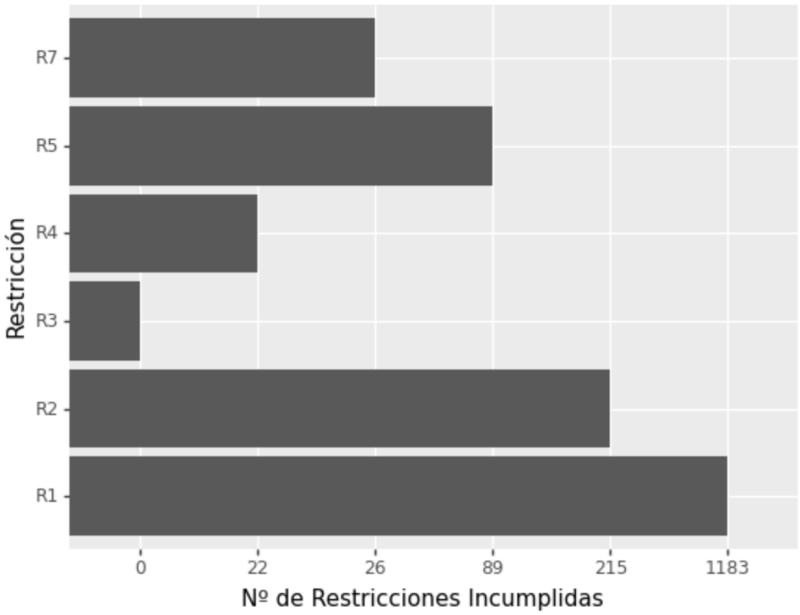

Figura 9.7: Desglose de restricciones incumplidas del problema 31 (con restricciones nuevas) con FL:10 y TE:1200

De este análisis se pueden sacar dos conclusiones: la primera es que la búsqueda de valores óptimos para la configuración de la ejecución no es una tarea trivial y requiere de un análisis exhaustivo y continuo sobre el modelo y los resultados y, en segundo lugar, que la búsqueda de estos valores de configuración es vital para obtener buenos resultados a nuestros problemas de optimización.

10. CAPÍTULO

Conclusiones y Trabajo Futuro

Debido a la falta de tiempo al final del proyecto y, sobre todo, falta de tiempo en la QPU para ejecutar otras pruebas que midiesen otros aspectos del modelo desarrollado, no se ha podido indagar más en la restricción 6, que es la que más problemas estaba dando a la hora de obtener una solución óptima. Sin embargo, a la vista de los resultados se puede ver que la implementación realizada es capaz de obtener resultados del problema de optimización respetando la gran mayoría de restricciones y en un tiempo inferior al que un computador clásico requeriría.

Hay que destacar también, que incluso añadiendo el tiempo de preprocesado, el tiempo total es inferior al que un ordenador clásico requeriría para obtener un resultado óptimo. Hay que recordar que el factor que influye en el tiempo de cómputo en un computador cuántico es el número de cúbits usados para la representación del problema y no el número de restricciones. Por ello, añadir una versión funcional de la restricción 6 no aumentaría el tiempo de cómputo (las variables "slack" de la función de penalización si aumentarían el tiempo de cómputo, pero es irrisorio) y se esperaría obtener un resultado mucho mejor.

La tecnología cuántica ha brindado muchos avances palpables en nuestra sociedad hoy en día, y está claro que este paradigma de computación, con algo más de desarrollo, divulgación y expertos que sepan usar y sacar el potencial de estas máquinas, será ampliamente usado por las empresas para optimizar sus procesos.

Respecto al trabajo futuro, la principal línea de investigación a seguir sería él como poder deducir de una forma más efectiva la configuración del computador cuántico para poder ejecutar las pruebas y obtener buenos resultados. Como se ha mencionado arriba, debido a

la falta de tiempo al final del proyecto no se ha podido mejorar los resultados de la función de penalización de la restricción 6, aunque tenemos varias ideas para solucionarlo

Otra cuestión que queda abierta es encontrar el porqué esta restricción en concreto no ha sido posible hacerla funcionar con buen rendimiento y poder encontrar una reformulación que consiga implementar la restricción como una función de penalización de forma óptima.

Por motivos académicos, se ha optado por desarrollar este proyecto a ”nivel de máquina”. Personalizar nosotros la propia expresión QUBO y sus valores; valores que reflejan la fuerza de los biases dentro de la máquina cuántica de DWave. Sin embargo, la API de DWave y su biblioteca de código ofrece soluciones para, a través de sus funciones y objetos, describir problemas de optimización de forma más natural; es decir, como si lo estuviésemos escribiendo matemáticamente en un papel.

Esto plantearía la duda de si merece la pena desarrollar nosotros un modelo QUBO personalizado y único a un problema dado o usar esta biblioteca y que se encargue de todo el proceso de transformación a un modelo QUBO de forma automática, ahorrando tiempo tanto para la representación del problema como para la implementación. ¿Cuál sería el mejor? ¿El segundo obtendría mejores resultados? Esta podría ser otra de las líneas que pueden estudiarse.

10.1. Material Generado

A lo largo del proyecto se ha desarrollado código tanto para transformar los problemas de planificación horaria de Plannam a un modelo QUBO, como pequeños ”scripts” para analizar los resultados que devolvía la QPU. Todo el código generado durante el periodo de vida del proyecto se encuentra disponible en aquí.

Bibliografía

[1] Chris Bernhart (2019), *Quantum ComputingFor Everyone*, MIT.

[2] DWave Systems, *Getting Started with D-Wave Solvers*, DWave System Documentation

[3] DWave Systems, *Problem-Solving Handbook*, DWave System Documentation

[4] DWave Systems, *DWave Main Web Page* (https://www.dwavesys.com/)

[5] Amit Verma, Mark Lewis (2021), *Variable Reduction For Quadratic Unconstrained Binary Optimization*, Missouri Western State University

[6] Silvestre G. Paredes, *Apuntes de Fundamentos de Optimización*, Universidad Politécnica de Cartagena