UAEM | Universidad Autónoma del Estado de México

C.U. VALLE DE CHALCO

CENTRO UNIVERSITARIO XICO VALLE DE CHALCO

APROXIMACIÓN DE SOLUCIONES DEL PROBLEMA MAX-SAT USANDO CÓMPUTO CUÁNTICO ADIABÁTICO

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRA EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

MARTHA GUADALUPE MORALES HUERTA

TUTOR ACADÉMICO

DR. WILLIAM DE LA CRUZ DE LOS SANTOS

TUTORES ADJUNTOS

DRA. MARÍA DE LOURDES LÓPEZ GARCÍA

DR. JUVENAL RUEDA PAZ

VALLE DE CHALCO SOLIDARIDAD, MÉXICO. NOVIEMBRE 2018

Universidad Autónoma del Estado de México
Centro Universitario Valle de Chalco

Valle de Chalco Solidaridad, Edo de Méx. a jueves, 08 de noviembre de 2018

DR. EN C. JUVENAL RUEDA PAZ
COORDINADOR DE LA MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN
DEL CENTRO UNIVERSITARIO *UAEM* VALLE DE CHALCO.

P R E S E N T E.

Por este medio le comunico a usted que la comisión revisora designada para realizar Tesis denominada: **"Aproximación de soluciones al problema max-SAT usando cómputo cuántico adiabático"**, como parte de los requisitos para obtener el grado académico de Maestría en **Ciencias de la Computación** presenta **Martha Guadalupe Morales Huerta**, con número de cuenta **1630649** para sustentar el acto de evaluación de grado, ha dictaminado que dicho trabajo reúne las características de contenido para proceder a la impresión del mismo.

A T E N T A M E N T E

Tutor adjunto	Tutor Académico	Tutor Adjunto
Dra. María de Lourdes López García	Dr. William De la Cruz De los Santos	Dr. Juvenal Rueda Paz

Centro Universitario UAEM Valle de Chalco
Av. Hermenegildo Galeana No 3, Col. Ma. Isabel, Valle de Chalco, C.P. 56615,
Edo. De México, Tel: (55) 59714940, 59787577 y 30921763
Página: http://cux.uaemex.mx e-mail: rgcruzf@uaemex.mx

Universidad Autónoma del Estado de México
Centro Universitario Valle de Chalco

Valle de Chalco Solidaridad, Estado de México jueves, 08 de noviembre de 2018

MARTHA GUADALUPE MORALES HUERTA
CANDIDATO A GRADO DE MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN
CENTRO UNIVERSITARIO UAEM VALLE DE CHALCO

Presente

De acuerdo con el Reglamento de Estudios Avanzados de la Universidad Autónoma del Estado de México y habiendo cumplido con todas las indicaciones que la Comisión Revisora realizó con respecto a su trabajo **Tesis** titulado **"Aproximación de soluciones al problema max-SAT usando cómputo cuántico adiabático"** la Coordinación de la Maestría en **Ciencias de la Computación** del Centro Universitario UAEM Valle de Chalco concede la autorización para que proceda a la impresión de la misma.

Sin más por el momento, le reitero la seguridad de mi especial consideración y estima.

ATENTAMENTE
PATRIA, CIENCIA Y TRABAJO
"2018, Año del 190 Aniversario de la Universidad Autónoma del Estado de México"

CENTRO UNIVERSITARIO UAEM VALLE DE CHALCO MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

DR. EN C. JUVENAL RUEDA PAZ
COORDINADOR DE LA MAESTRÍA CIENCIAS DE LA COMPUTACIÓN
CENTRO UNIVERSITARIO UAEM
VALLE DE CHALCO

CARTA DE CESIÓN DE DERECHOS DE AUTOR

El que suscribe **Martha Guadalupe Morales Huerta** Autor del trabajo escrito de evaluación profesional en la opción de Tesis con el título "Aproximación de soluciones al problema max-SAT usando cómputo cuántico adiabático", por medio de la presente con fundamento en lo dispuesto en los artículos 5, 18, 24, 25, 27, 30, 32 y 148 de la Ley Federal de Derechos de Autor, así como los artículos 35 y 36 fracción II de la Ley de la Universidad Autónoma del Estado de México; manifiesto mi autoría y originalidad de la obra mencionada que se presentó en el **Centro Universitario UAEM Valle de Chalco** para ser evaluada con el fin de obtener el Grado de **Maestra en Ciencias de la Computación**.

Así mismo expreso mi conformidad de ceder los derechos de reproducción, difusión y circulación de esta obra, en forma NO EXCLUSIVA, a la Universidad Autónoma del Estado de México; se podrá realizar a nivel nacional e internacional, de manera parcial o total a través de cualquier medio de información que sea susceptible para ello, en una o varias ocasiones, así como en cualquier soporte documental, todo ello siempre y cuando sus fines sean académicos, humanísticos, tecnológicos, históricos, artísticos, sociales, científicos u otra manifestación de la cultura.
Entendiendo que dicha cesión no genera obligación alguna para la Universidad Autónoma del Estado de México y que podrá o no ejercer los derechos cedidos.
Por lo que el autor da su consentimiento para la publicación de su trabajo escrito de evaluación profesional.
Se firma la presente en la ciudad de Valle de Chalco, a los 08 días del mes de Noviembre de 2018.

Martha Guadalupe Morales Huerta

Dedicatoria

Esta tesis está dedicada a mi familia y a mis pocos
pero excelentes amigos,
con amor.

Agradecimientos

Primeramente, quiero agradecer a Dios por permitirme tener las fuerzas y el entusiasmo necesarios para poder aventurarme a mundos desconocidos con la convicción de que podré superar los retos que se me presenten aun a pesar de las dificultades.

Este trabajo de tesis no hubiese sido posible sin el ingenio y la inteligencia de mi Director de Tesis, el Dr. William De la Cruz, quien siempre me ha impulsado a buscar retos y a amar la investigación. Estoy extremadamente agradecida y en deuda con él por compartir, sin nungún tipo de reserva, todo su conocimiento y experiencias teniéndome mucha paciencia, lo cual valoro y atesoraré por el resto de mi vida. Asímismo, quiero agradecer a mis tutores adjuntos, el Dr. Juvenal Rueda y la Dra. María de Lourdes López, por sus valiosos consejos en mi formación y por darme de su tiempo en la construcción y revisión de este trabajo de investigación.

También, quiero agradecer de corazón a mi familia y amigos que me estuvieron dando constantemente palabras de apoyo y aliento para conseguir mis metas y nunca darme por vencida.

Por último, deseo agradecer al Centro Universitario UAEM Valle de Chalco por las instalaciones prestadas y al CONACyT por la beca otorgada durante la realización de los estudios de Maestría.

Resumen

Actualmente, se han propuesto tecnologías cuánticas basadas en el algoritmo de temple cuántico y cómputo cuántico adiabático con aplicaciones a problemas de optimización combinatorios. Diversos estudios teóricos y experimentales se han enfocado en determinar las ventajas y desventajas de resolver clases específicas de problemas tales como detección de fallas en redes de potencia, plegado de proteínas, satisfacción de restricciones, entre otros. En esta tesis se propone una formulación cuántica del problema de máxima satisfactibilidad booleana usando el algoritmo de temple cuántico y cómputo cuántico adiabático. Nuestra formulación consiste en la construcción de una función booleana cuadrática cuya optimización corresponde a la solución del problema de estudio. También, se proponen tres estrategias de mapeo directas para instancias del problema de máxima satisfactibilidad booleana sobre la topología de hardware cuántico de la computadora D-Wave. Para validar nuestra propuesta, realizamos simulaciones computacionales del modelo cuántico para aproximar soluciones del problema de estudio, y las comparamos con resultados obtenidos usando algoritmos clásicos. Los resultados de las simulaciones muestran que el problema de máxima satisfactibilidad booleana puede ser tratado por medios cuánticos con las tecnologías cuánticas actuales. Sin embargo, se tienen que llevar a cabo investigaciones futuras para determinar si el algoritmo de temple cuántico y el cómputo cuántico adiabático, para el problema de estudio, tienen ventajas en términos de complejidad cuando se comparan con los mejores algoritmos en el estado del arte.

Abstract

Currently, quantum technologies based on the algorithm of quantum annealing and adiabatic quantum computation have been proposed with applications to combinatorial optimization problems. Several theoretical and experimental studies have focused on determining the advantages and disadvantages of solving specific classes of problems such as power network failure detection, protein folding, constraint satisfaction, among others. In this thesis we propose a quantum formulation of the maximum satisfiability problem using the algorithm of quantum annealing and adiabatic quantum computation. Our formulation consists on the construction of a quadratic pseudo-Boolean function whose optimization corresponds to the solution of the study problem. Also, three direct embedding strategies are proposed for instances of the maximum satisfiability problem on the quantum hardware topology of the D-Wave computer. To validate our proposal, we perform computational simulations of the quantum model to approximate solutions to the study problem, and compare it with results obtained using classical algorithms. The results of the simulations show that the problem of maximum satisfiability can be treated by quantum means with the current quantum technologies. However, future research has to be carried out to determine whether the algorithm of quantum annealing and adiabatic quantum computation, for the study problem, have advantages in terms of complexity when compared with the best algorithms in the state of the art.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El *Cómputo Cuántico* (CC) fue propuesto por Richard Feynman [18] en 1982 como una idea de crear computadoras que usaran los efectos de la mecánica cuántica y demostrar si es posible simular un sistema cuántico por medio de un simulador cuántico universal. Entre los algoritmos cuánticos más conocidos destaca el algoritmo de Shor [49] para factorizar números enteros en sus factores primos y el algoritmo de Grover [24] para realizar búsquedas en una base de datos no estructurada.

La programación de una computadora cuántica es muy diferente al de una computadora tradicional. Un procesador cuántico considera todas las posibilidades simultáneamente y por medio de la aplicación de transformaciones unitarias puede cambiar las probabilidades de encontrar una solución a un problema dado. Una computadora cuántica es de naturaleza probabilística, lo cual significa que para obtener una solución a un problema, se tiene que repetir la ejecución del algoritmo cuántico un número determinado de veces y escoger la solución que aparece con mayor frecuencia [13].

El *Cómputo Cuántico Adiabático* [16] (CCA) se introdujo como una alternativa al paradigma de compuertas cuánticas para resolver problemas NP-difíciles, que clásicamente se conocen como *intratables*, es decir, que no pueden ser resueltos por un algoritmo clásico eficientemente. El CCA se introdujo principalmente para resolver problemas de optimización combinatoria; este tiene ventajas como robustez hacia ruido [34] y permite modelar

problemas de una forma general por medio del diseño de un hamiltoniano en donde se codifican todas las posibles soluciones.

La solución de un problema de optimización combinatorio consiste en encontrar un vector de estados que minimiza o maximiza una función de costo dada. Por lo regular, este tipo de problemas son NP-difícil [21]. El problema de *máxima satisfactibilidad booleana* (max-SAT) pertenece a esta clasificación, el cual es la versión de optimización del problema de decisión *satisfactibilidad booleana* (SAT). Este último fue planteado por primera vez en 1960 [14] para comprobar la satisfactibilidad de las fórmulas de la lógica proposicional en un conjunto de cláusulas unidas por conjunciones, es decir, determina si existe una asignación de verdad que satisfaga a todas las cláusulas de una fórmula booleana que se encuentra en forma conjuntiva (FC). Por otro lado, el problema max-SAT consiste en encontrar una asignación a las variables booleanas en una FC de tal forma que satisfaga el mayor número de cláusulas. Algunos ejemplos de su uso son: consistencia en sistemas expertos basados en conocimiento [42], restricciones de integridad en bases de datos [3] y síntesis de circuitos asíncronos [25].

El algoritmo de *temple cuántico* (TC) fue formulado para resolver problemas de optimización combinatorios, empleando fluctuaciones cuánticas para atravesar barreras de potencial, lo que permite explorar el espacio de soluciones de una forma más eficiente [37, 19, 33, 47]. En particular, el hardware D-Wave implementa el algoritmo TC para resolver instancias del modelo Ising en un campo magnético transversal, el cual también se conoce como un problema NP-difícil [5]. Esta tecnología ha tenido buenos resultados sobre algunos problemas de optimización específicos, comparados con el estado del arte de los algoritmos clásicos [36].

El proceso por el cual una instancia del modelo Ising es mapeada al hardware cuántico es conocido como *mapeo menor* [9, 8]. Este proceso es de importancia para la determinación de la cantidad de recursos físicos (*qubits*). En general, encontrar un mapeo menor con un número mínimo de qubits es un problema difícil [8]. Sin embargo, existen algoritmos heurísticos que encuentran mapeos menores que se aproximan a la solución óptima en

tiempo polinomial. Otros autores proponen algoritmos de mapeo específicos dependiendo del problema dado como los citados en [6, 51].

En este trabajo de tesis se propone el modelado de un algoritmo cuántico usando CCA para aproximar soluciones al problema max-SAT. Se modeló matemáticamente el problema max-SAT como un problema de optimización de una función pseudo-booleana. Es decir, dada una FC se construye una función pseudo-booleana correspondiente de forma tal que el vector que minimiza la función corresponde a una solución de la instancia max-SAT. Para demostrar que el modelo propuesto para el problema max-SAT es correcto, se realizaron simulaciones computacionales usando las librerías proporcionadas por D-Wave. Además, se diseñaron tres estrategias de mapeo a la topología de hardware en la computadora cuántica de D-Wave para el problema max-SAT. Las estrategias de mapeo propuestas aprovechan la estructura del problema max-SAT, permitiendo diseñar métodos directos de mapeo en la topología. Los resultados de las simulaciones llevadas a cabo, usando el algoritmo de *temple cuántico simulado* (TCS), muestran que el tamaño de los mapeos obtenidos son competitivos en comparación con las técnicas heurísticas.

1.1. Planteamiento del problema

El problema max-SAT es NP-difícil, por lo que no existen algoritmos eficientes para resolverlo, en otras palabras, no existe un algoritmo que lo resuelva en tiempo polinomial. El cómputo cuántico adiabático es un modelo que permite encontrar soluciones a problemas de optimización combinatorios y es aplicable, principalmente, a problemas NP-difíciles. Estudiar el comportamiento de algoritmos cuánticos para aproximar soluciones a problemas de optimización es de importancia y así determinar a que clase de problemas se pueden obtener ventajas en términos de complejidad en comparación con algoritmos clásicos. Inclusive, el diseño de un algoritmo cuántico por medio del CCA para aproximar soluciones al problema max-SAT.

1.2. Objetivos

1.2.1. General

Diseñar un algoritmo cuántico usando CCA para obtener soluciones del problema max-SAT.

1.2.2. Específicos

- Construir una función objetivo usando variables lógicas cuyo mínimo global corresponda a la solución del problema max-SAT.
- Construir una función pseudo-booleana equivalente con la función en el punto anterior, para ser optimizada por medio de un algoritmo cuántico.
- Diseñar algoritmos de mapeo a la topología del hardware cuántico (arquitectura D-Wave).
- Analizar estadísticamente las soluciones obtenidas.

1.3. Justificación de la investigación

En ciencias de la computación, los problemas que pertenecen a la clase NP-difícil son problemas de optimización, para los cuales no existen algoritmos que los resuelvan en tiempo polinomial. Existen varias técnicas de aproximación que consisten en algoritmos rápidos que obtienen soluciones aproximadas a problemas NP-difícil, como el algoritmo de temple simulado, métodos de Monte Carlo, heurísticas y algoritmos probabilistas. Por otro lado, el CC ha demostrado tener ventajas en diversos problemas difíciles. Conocer con mayor precisión qué clases de problemas son los más adecuados para resolverlos usando algoritmos cuánticos es de vital importancia en la comunidad de ciencias de la computación y computación cuántica.

1.4. Hipótesis

Por medio de simulaciones y ejecución del algoritmo a desarrollar, usando la arquitectura del hardware cuántico D-Wave, se cree que puede obtener una ventaja en tiempo de ejecución, superior a cualquier computadora convencional.

1.5. Metodología

Para alcanzar los objetivos de esta tesis, se realizaró una investigación documental sobre los fundamentos de complejidad computacional y el cómputo cuántico, así como del estado del arte de los algoritmos del temple cuántico y cómputo cuántico adiabático. Se usó el método analítico para la propuesta teórica del problema de estudio, y se usó el método experimental para validar el modelo matemático desarrollado. Además, se utilizaron herramientas técnicas como lenguajes de programación y herramientas estadísticas para el estudio la realización de pruebas y análisis de los resultados.

1.6. Contexto y delimitación de la tesis

El CCA es una subárea del CC el cual es una rama de las ciencias de la computación teórica. En este trabajo de tesis se desarrolla e implementa un algoritmo que encuentra soluciones aproximadas al problema max-SAT usando CCA. En la investigación se generan instancias difíciles del problema y se mapean a la arquitectura del hardware cuántico mediante tres estrategias propuestas y se realiza el análisis correspondiente para probar su efectividad con respecto a el método heurístico.

El algoritmo cuántico propuesto en esta tesis es aplicable a instancias del problema max-SAT, principalmente a instancias difíciles. Se usa el modelo continuo del CCA para resolver problemas de optimización. El modelado matemático del problema max-SAT se basa principalmente en la construcción de una función pseudo-booleana multivariada, la

cual puede ser optimizada por medio del hardware cuántico. Las estrategias de mapeo propuestas son aplicables a la topología existente de la computadora D-Wave 2000Q, que se pueden tomar como base para diseñar estrategias de mapeo en otras topologías como por ejemplo la arquitectura cuántica de IBM-Q [1].

1.7. Organización de la tesis

El presente trabajo está dividido en 5 capítulos y se organiza como sigue: en al capítulo 1 se plantean los objetivos e introducción de la problemática de esta tesis. En el capítulo 2 se introducen los conceptos importantes para el desarrollo de la tesis como son: complejidad computacional, principios básicos del CCA, así como la tecnología a utilizar para el desarrollo del algoritmo cuántico. En el capítulo 3 se plantea la formulación cuántica del problema max-SAT y construcción de una función pseudo-booleana multivariada del problema, así como simulaciones usando el algoritmo TCS. En el capítulo 4 se explican las estrategias de mapeo desarrolladas, así como los resultados de los experimentos realizados. Por último, en el capítulo 5, se dan las conclusiones de esta investigación y los trabajos futuros a realizar.

[1] IBM Quantum Experience (`https://www.research.ibm.com/ibm-q/`)

Capítulo 2

Marco teórico

En este capítulo se introducen los conceptos básicos para abordar este trabajo de investigación. Se comienza introduciendo conceptos básicos de complejidad computacional, los cuales forman la base para comprender la naturaleza del problema de estudio max-SAT. Además, se describen los fundamentos del cómputo cuántico y los algoritmos cuánticos. Finalmente, se habla de la tecnología D-Wave con la cual se realiza la implementación del problema de estudio.

2.1. Complejidad computacional

La teoría de la computación es una rama de las matemáticas y de la computación que centra su interés en el estudio y definición formal de la computación. Su objetivo principal es determinar cuáles son las capacidades y limitaciones de las computadoras. Para ello se vale de otras teorías tales como teoría de autómatas, teoría de computabilidad y teoría de complejidad computacional.

La historia de la teoría computacional comienza en 1936 con uno de los aportes más importantes: la máquina de Turing [50]. Ésta demostró ser un modelo de una computadora flexible y robusta, aunque luego se comprobó que fallaban al cuantificar el tiempo y la memoria requerida por una computadora, constituyendo un problema crítico. La idea de

medir el tiempo y espacio como una función de la longitud de la entrada, se originó a principios de los años 60's por Hartmanis y Stearns [28], y así nació la teoría de la complejidad computacional.

Inicialmente, los investigadores trataban de entender las nuevas medidas de complejidad y cómo se relacionaban unas con otras. En 1965, Edmonds [15] definió un algoritmo en el cual un polinomio acotaba el tiempo de ejecución (tiempo polinomial), lo que conllevó a la noción de eficiencia de algoritmos. Es decir, un algoritmo es eficiente siempre y cuando su complejidad en tiempo o número de iteraciones está acotado por un polinomio que depende del tamaño de la entrada.

El objetivo principal de esta teoría es la creación de mecanismos y herramientas capaces de describir y analizar la complejidad de un algoritmo y la complejidad intrínseca de un problema. Busca la razón de qué hace a algunos problemas computacionalmente difíciles y a otros sencillos.

La teoría de la complejidad computacional se centra en la clasificación de los problemas computacionales de acuerdo a su dificultad inherente, y en la relación entre dichas clases de complejidad. Así mismo, estudia la eficiencia de los algoritmos estableciendo su efectividad de acuerdo al tiempo de ejecución y al almacenamiento de datos requerido por la computadora, ayudando a evaluar la viabilidad de la práctica en tiempo y costo.

Se denota como $T(x)$ el tiempo de ejecución de un algoritmo $\mathcal{A}$ con una entrada $x \in \{0,1\}^n$. La complejidad en tiempo del algoritmo $\mathcal{A}$ está dado por

$$t(n) = \text{máx}\{T(x) \mid \forall x \in \{0,1\}^n, |x| \leq n\} \tag{2.1}$$

donde $|x|$ denota la longitud de la cadena x.

De forma similar, se denota como $S(x)$ el espacio requerido en la ejecución del algoritmo $\mathcal{A}$ con entrada $x \in \{0,1\}^n$. La complejidad en espacio de $\mathcal{A}$ se define por

$$s(n) = \text{máx}\{S(x) \mid \forall x \in \{0,1\}^n, |x| \leq n\}. \tag{2.2}$$

De aquí en adelante se considera como la complejidad de ejecución de un algoritmo como la complejidad en tiempo.

2.1.1. Orden de crecimiento de funciones

Sea $g(n)$ una función con $n \in \mathbb{Z}^+$. Se denota por $O(g(n))$ al conjunto de funciones

$$O(g(n)) \quad = \quad \{f(n) \mid \exists c, n_0 \in \mathbb{R}, 0 \leq f(n) \leq cg(n), \forall n \geq n_0\}. \tag{2.3}$$

En otras palabras, $f(n)$ está en $O(g(n))$ si existen constantes c, n_0 tal que $cg(n)$ acota superiormente a $f(n)$ para $n \geq n_0$. El orden O se usa comúnmente para representar el "peor caso" de ejecución de un algoritmo.

Del mismo modo, se denota por $\Theta(g(n))$ al conjunto de funciones

$$\Theta(g(n)) \quad = \quad \{f(n) \mid \exists c_1, c_2, n_0 \mathbb{R}, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0\}. \tag{2.4}$$

En otras palabras, $f(n)$ está en $\Theta(g(n))$ si y sólo si $f(n)$ está acotada tanto por arriba como por abajo por la función $g(n)$. La notación Θ se usa comúnmente para representar la complejidad "exacta" de un algoritmo.

Por último, se denota por $\Omega(g(n))$ al conjunto de funciones

$$\Omega(g(n)) \quad = \quad \{f(n) \mid \exists c, n_0, 0 \leq cg(n) \leq f(n), \forall n \geq n_0\}. \tag{2.5}$$

De manera similar, una función $f(n)$ pertenece a $\Omega(g(n))$ si $f(n)$ está acotada por abajo por la función $g(n)$. La notación Ω se usa comúnmente para representar la complejidad en el "mejor caso" de un algoritmo.

Los ordenes de crecimiento de funciones O, Θ, Ω, se usan para describir la complejidad en tiempo o espacio de un algoritmo [11]. Por ejemplo, si un algoritmo ejecuta un número de pasos dado por $t(n) = an^2 + b \log n$ para procesar una entrada de tamaño n, decimos que su complejidad es $O(n^2)$. Si, por otra parte, somos capaces de probar que para cada n, siendo suficientemente grande existe una instancia de entrada de tamaño n sobre la cual los algoritmos ejecutan al menos cn^2 pasos. La Figura 2.1 muestra ejemplos del crecimiento de algunas funciones.

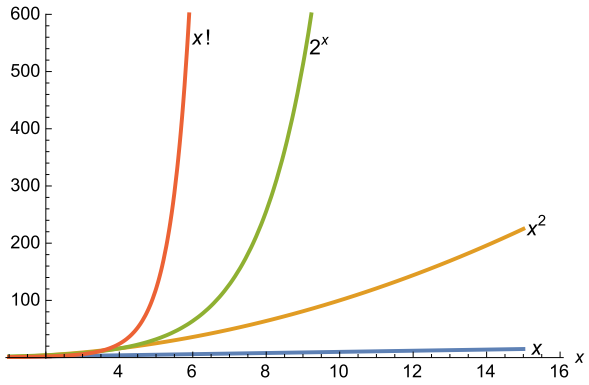


Figura 2.1: Ejemplo de crecimiento de funciones.

2.1.2. Cómputo determinista y no-determinista

Un autómata finito o máquina de estado-finito comparte con una computadora real el hecho de que tiene una unidad central de procesamiento con capacidad finita. Lo que hace que el autómata finito sea un modelo computacional restringido es la ausencia completa de memoria fuera de su procesador central.

Sea $\Sigma = \{0, 1\}$ un conjunto de los símbolos 0 y 1 llamado *alfabeto*. Una cadena $\sigma = \sigma_1\sigma_2 \cdots \sigma_n$ es la concatenación de símbolos $\sigma_j \in \Sigma$ con $j = 1, \ldots, n$. La *longitud* de la cadena σ, denotado por $|\sigma|$, es el número de símbolos que contiene. Una cadena de longitud 0 se le conoce como la *cadena vacía*. Denotamos por $\Sigma^* = \bigcup_{k>0} \Sigma^k$ un *diccionario* donde cada Σ^k es un conjunto de cadenas con símbolos en Σ de longitud k. Un *lenguaje* es cualquier subconjunto de cadenas en Σ^*.

Definición 1 (Autómata Finito Determinista (AFD) [40])**.** *Un AFD es una 5-tupla* $A = (Q, \Sigma, \delta, q_0, F)$ *donde* Q *es un conjunto finito de estados,* Σ *es un alfabeto,* $q_0 \in Q$ *es el estado inicial,* $F \subseteq Q$ *es el conjunto de estados finales, y* δ *es la función de transición definida de* $Q \times \Sigma$ *a* Q*.*

Las reglas de acuerdo al autómata A de la definición 1, que eligen el siguiente estado, se encuentran codificadas en la función de transición δ. Así, si A está en el estado $q \in Q$ y el símbolo leído es $a \in \Sigma$, entonces existe un único estado $p = \delta(q, a)$ con

el símbolo a. Para cualquier cadena $\sigma \in \Sigma^*$, la función extendida $\hat{\delta}$ se define como la aplicación sucesiva de la función δ con los símbolos en σ. Un AFD A reconoce una cadena $\sigma \in \Sigma^*$ si $\hat{\delta}(q_0, \sigma) \in F$ en cuyo caso escribimos $A(\sigma) = 1$.

El lenguaje de un AFD es el conjunto de cadenas que reconoce, es decir

$$L_A = \{\sigma \in \Sigma^* \mid A(\sigma) = 1\}. \tag{2.6}$$

El no-determinismo se refiere a la capacidad de estar en varios estados a la vez. Los autómatas finitos no-deterministas no están pensados como modelos realistas de computadoras sino como una generalización notacional útil de autómatas finitos, ya que pueden simplificar enormemente la descripción de estos autómatas. Entonces, el no-determinismo no es una característica esencial de los autómatas finitos: cada autómata finito no-determinista es equivalente a un autómata finito determinista.

Definición 2 (Autómata Finito No-determinista (AFN) [40])**.** *Un AFN es una 5-tupla* $N = (Q, \Sigma, \delta_N, q_0, F)$ *donde* Q, Σ, q_0, F *se definen como en un AFD y* δ_N *es una función tal que para cada* $q \in Q$ *y* $a \in \Sigma$ *se tiene que* $\delta_N(q, a) \in \mathcal{P}(Q)$ *donde* $\mathcal{P}(Q)$ *es el conjunto potencia de* Q*.*

La definición formal de computación basada en AFN es similar a la de AFD. Con la diferencia fundamental de que un AFN puede "escoger" la transición más adecuada para llegar a un estado final. El cómputo de un AFN se puede ver como un árbol de posibilidades donde cada rama representa un camino determinista. Un AFN reconoce una cadena $\sigma \in \Sigma^*$ si existe al menos una rama que llega a un estado final. Así, el lenguaje de un AFN es el conjunto de cadenas que reconoce.

2.1.3. Máquinas de Turing

Las máquinas de Turing son un modelo abstracto de computación que ofrece una formulación simple de un algoritmo (ver Figura. 2.2) [50]. Se define como sigue:

Definición 3 (Máquina de Turing Determinista (MTD))**.** *Sea una MTD M una 7-tupla $(Q, \Sigma, \Gamma, \Delta, q_0, \sqcup, F)$ donde Q es un conjunto finito de estados, Σ es el alfabeto de entrada, Γ es el alfabeto de la cinta, $q_0 \in Q$ es el estado inicial, $\sqcup$ es un símbolo especial, $F \subset Q$ es un conjunto de estados finales i.e. $F = \{q_Y, q_N\}$, y Δ es una función de transición. Aquí, se asume que existe un estado de aceptación q_Y y un estado de rechazo q_N.*

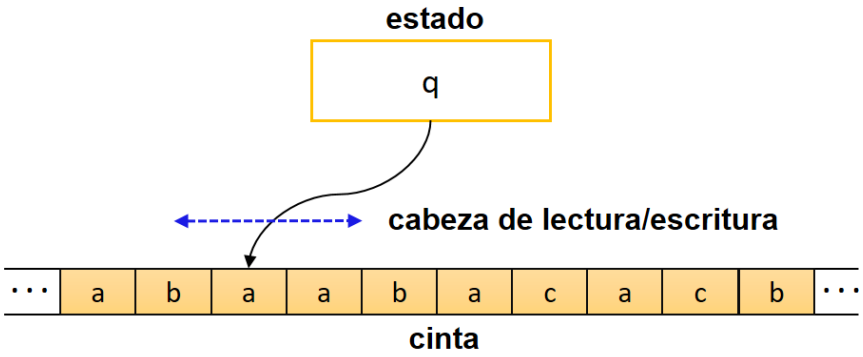


Figura 2.2: Esquema de una Máquina de Turing (imagen propia).

La función de transición Δ definida como $\Delta : (Q - \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{-1, 0, 1\}$ tal que $(q, X) \mapsto (p, Y, D)$ donde $p, q \in Q, X, Y \in \Gamma$ y D es la dirección de la cabeza lectora. Aquí $\{-1, 1, 0\}$ significa un movimiento a la izquierda, derecha o sin movimiento respectivamente.

Dada una MTD M y una cadena de entrada $x \in \Sigma^*$, se denota por $M(x)$ la aplicación de M sobre x. Inicialmente, se copia x en la cinta a partir de la posición número uno y se realizan los siguientes pasos según sea el caso:

1. Se inicia en el estado q_0 y la cabeza de lectura/escritura se sitúa sobre el primer símbolo de entrada.

2. Si el estado actual es q_Y o q_N la computación termina.

3. Si el estado actual es $q \in Q - \{q_Y, q_N\}$ se lee el símbolo s sobre la cinta en la que se encuentra la cabeza lectora y el siguiente estado es $\Delta(q, s) = (q', s', D)$. Es decir, se borra el símbolo s sobre la cinta y se escribe el símbolo s', la cabeza se mueve una

posición a la izquierda si $D = -1$ o una posición a la derecha si $D = 1$, finalmente el nuevo estado es q'.

Dada una MTD M y $x \in \Sigma^*$ se dice que M acepta a x si y sólo si, la computación de M sobre x termina en el estado q_Y y escribimos $M(x) = 1$, $M(x) = 0$ si es q_N y $M(x) = \perp$ en otro caso. El lenguaje que reconoce M está dado por $L_M = \{x \in \Sigma^* \mid M \text{ acepta a } x\}$. El tiempo de computación de M con entrada x es el número de pasos que realiza hasta llegar a un estado terminal. Si para toda $x \in \Sigma^*$, $M(x) \neq \perp$ se define la *complejidad en tiempo* de M como $T_M : \mathbb{Z}^+ \to \mathbb{Z}^+$ dada por

$$T_M(n) = \max_{x \in \Sigma^*, |x|=n} \{\text{tiempo computación de } M(x)\}.$$

Una MTD M es de *complejidad polinomial* en tiempo si existe un polinomio p tal que para toda $n \in \mathbb{Z}^+$, $T_M(n) \leq p(n)$.

Definición 4 (Máquina de Turing No-determinista (MTN))**.** *Una MTN N es una 7-tupla $(Q, \Sigma, \Gamma, \Delta_N, q_0, \sqcup, F)$ donde $Q, \Sigma, \Gamma, q_0 \in Q$ y F se definen como en una MTD y Δ_N se define de tal manera que para cualquier $q \in Q, X \in \Gamma : \Delta_N(q, X) \in \mathcal{P}(Q \times \Gamma \times \{-1, 0, 1\})$. Esto significa que para un estado actual de N pueden existir más de un estado siguiente. Para alguna entrada $x \in \Sigma^*$, $N(x)$ es un árbol de computación donde cada rama de computación es uno de los posibles caminos que toma N con la entrada x según Δ_N.*

Una MTN N funciona de la siguiente forma:

1. Toma como entrada $x \in \Sigma^*$, se escribe sobre la cinta a partir de la posición 1 hasta la número $|x|$, la cabeza lectora se localiza sobre el primer símbolo en la cinta y el estado inicial es q_0.

2. Explora todas las ramas de computación en paralelo a partir del estado inicial al igual que una MTD.

Se dice que una MTN N acepta a x si existe al menos una rama de computación que llegue al estado terminal q_Y . El lenguaje que reconoce N es $L_N = \{x \in \Sigma^* \mid N \text{ acepta a } x\}$. El tiempo de computación de N con la entrada $x \in L_N$ es la longitud máxima de todas las ramas de computación que terminan en el estado q_Y. La complejidad de N se define como $T_N : \mathbb{Z}^+ \to \mathbb{Z}^+$ dada por

$$T_N(n) = \max_{x \in \Sigma^*, |x|=n} \{\text{tiempo computación de } N(x)\}.$$

Una MTN N es de complejidad polinomial si existe un polinomio p tal que para toda $n \geq 1$, $T_N(n) \leq p(n)$.

2.1.4. Clases de problemas P, NP, NP-completos

La clase de problemas P consiste en problemas de decisión que se pueden resolver por MTDs de complejidad polinomial y la clase NP consiste en problemas de decisión que se pueden resolver por MTNs de complejidad polinomial. Informalmente hablando, la clase P es el tipo de problemas de decisión que son resolubles por algún algoritmo en un número de pasos delimitado por un polinomio en el tamaño de la entrada. Mientras que NP son aquellos problemas cuyas soluciones se pueden verificar eficientemente por una MTD. El problema P-vs-NP, formalizado por Cook en 1971 [10], consiste en determinar si acaso P=NP o P≠NP.

Formalmente, las clases P y NP se definen como [2, 4, 21, 23]:

Definición 5 (Clase P)**.** *La clase* P *consiste de todos los lenguajes L de manera que existen MTDs M de complejidad polinomial tal que $L = L_M$.*

Definición 6 (Clase NP)**.** *La clase* NP *consiste de todos los lenguajes L de manera que existen MTNs N de complejidad polinomial tal que $L = L_N$.*

Definición 7 (Reducción en tiempo polinomial)**.** *Un lenguaje L_1 sobre Σ_1 es reducible en tiempo polinomial a un lenguaje L_2 sobre Σ_2, denotado por $L_1 \leq_p L_2$, si existe una función*

$f : \Sigma_1^* \to \Sigma_2^*$ *de complejidad polinomial tal que para cada* x *se cumple*

$$x \in L_1 \textit{ si y sólo si } f(x) \in L_2.$$

Definición 8 (Problemas NP-completos)**.** *Un problema de decisión* Π *se dice que es NP-completo si* Π *pertenece a* NP *y para cualquier problema* Π' *en* NP *se cumple que* $\Pi' \leq_p \Pi$.

Por las definiciones 5 y 6 se satisface que P⊂NP. Retomando el problema de si P $\neq$ NP, se cumple si y sólo si por cada problema NP-completo $\mathcal{P}$, $\mathcal{P} \notin$ P. Hasta el momento, no se sabe si existen algoritmos en tiempo polinomial para los problemas NP-completos y solo existen actualmente algoritmos en tiempo *superpolinomial*, incluso si no se ha demostrado un límite inferior superpolinomial en la complejidad de tiempo para cada uno de esos problemas.

2.1.5. Problemas de optimización combinatorios

La optimización combinatoria es un campo emergente a la vanguardia de la combinatoria y la informática teórica que apunta a utilizar ciertas técnicas para resolver problemas discretos de optimización. Un problema de optimización discreto busca determinar la mejor solución posible a partir de un conjunto finito de posibilidades.

Definición 9 (Clase NPO [4])**.** *Un problema de optimización* $\mathcal{P} = (\mathcal{I}, \mathcal{S}, m, goal)$ *pertenece a la clase de problemas NP de optimización (NPO) si se cumple lo siguiente:*

1. *El conjunto de instancias* $\mathcal{I}$ *es identificado en tiempo polinomial.*
2. *Existe un polinomio* q *tal que dada una instancia* $x \in \mathcal{I}$*, para cada* $y \in \mathcal{S}(x)$*,* $|y| < q(x)$ *y, para cada* y *tal que* $|y| < q(x)$*, es decidible en tiempo polinomial si* $y \in \mathcal{S}(x)$.
3. *La función de medida* m *es computable en tiempo polinomial.*

La optimización combinatoria busca mejorar los algoritmos mediante el uso de métodos matemáticos, ya sea para reducir el tamaño del conjunto de soluciones posibles o para

hacer que la búsqueda sea más rápida. Surgió originalmente en teóricas de gráficas con problemas de la coloración gráficas y apareamientos de gráficas bipartitas. En la era moderna, la optimización combinatoria es útil para el estudio de algoritmos, con especial relevancia para la inteligencia artificial, el aprendizaje automático y la investigación operativa.

2.1.6. Clase de problemas NP-difíciles

Un problema A es NP-difícil si existe una reducción en tiempo polinomial de un problema NP-completo B a A. Tal que una reducción implica que un algoritmo que resuelve A en tiempo polinomial puede ser usado como una subrutina para resolver B en tiempo polinomial. Un problema es llamado NP-difícil porque este es difícil como (o más difícil que) un problema NP-completo. El diagrama de la Figura 2.3, basado en [38], muestra el dominio de los problemas NP-difíciles sobre todo el conjunto de problemas.

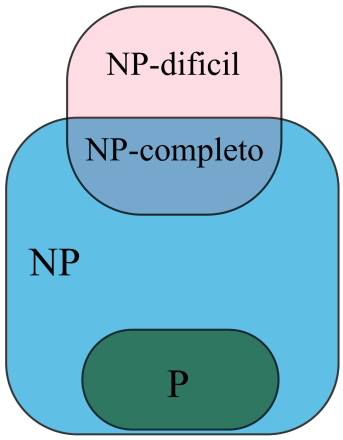


Figura 2.3: Dominio de los problemas NP-difícil (imagen propia).

Los problemas P no pertenecen a NP-difícil y la intersección de P con NP-difícil está vacía. Esto es porque si un problema NP-difícil puede ser resuelto en tiempo polinomial entonces los problemas NP-completos pueden ser resueltos también en tiempo polinomial, así como los NP. Si esto se cumple podemos determinar que P=NP.

No se sabe, hasta ahora, cuáles problemas NP-difícil no son NP-completos. Por tanto, si encontramos dicho problema, podemos probar que P≠NP. Suponiendo que P =

NP, entonces todos los problemas NP pueden ser resueltos en tiempo polinomial y, entonces, todos los problemas NP-completos pueden ser resueltos en tiempo polinomial y, por lo tanto, todos los problemas son NP-difícil.

2.2. Algoritmo de temple simulado

El algoritmo de temple simulado (TS) fue propuesto por Kirkpatrick [37] en 1983 como un intento de equilibrar la exploración con la explotación. El TS busca el mímimo global x^* en un camino indirecto y estocástico. También, busca una distribución de probabilidad p sobre el espacio de búsqueda. La probabilidad asociada con el elemento x en el espacio de búsqueda se denota con $p(x)$. La probabilidad p se determina por una función de minimización balanceada en términos explorativos y explotativos, dada por

$$F(p) = \mathbb{E}_p(E) - T \cdot S(p) \tag{2.7}$$

donde $\mathbb{E}_p(E)$ es la energía esperada debajo de la distribución p establecida como

$$\mathbb{E}_p(E) = \sum_x p(x)E(x) \tag{2.8}$$

y es minimizada por la distribución de probabilidad $p(x)$ tal que $p(x) = 1$ si $x = x^*$, y $p(x) < 1$ en otro caso. El término $S(p)$ corresponde a la entropía, la cual está dada por

$$E(x) = -\sum_x p(x) \ln p(x). \tag{2.9}$$

La inclusión de la entropía favorece la exploración de todo el espacio de búsqueda y T corresponde a la temperatura. El TS comienza con una temperatura alta la cual favorece la exploración del espacio de búsqueda. Conforme el algoritmo progresa, la temperatura decrece hasta regiones donde $\mathbb{E}_p(E)$ es baja. Periódicamente, T es llevada a 0 para que el mínimo global x^* sea obtenido.

En [29] se define el TS como una técnica de optimización que puede: a) procesar funciones de costo que poseen grados bastante arbitrarios de no-linealidad, discontinuidades, y estocasticidad; b) procesar condiciones de frontera y restricciones bastante arbitrarias

impuestas a estas funciones de costo; c) ser implementado con bastante facilidad con un grado de codificación mínimo en relación con otros algoritmos de optimización no lineal; y d) garantizar estadísticamente la búsqueda de una solución óptima. El algoritmo 1 muestra el TS.

Algorithm 1: Temple simulado.

Input : instancia x, l máximo número de iteraciones, m función de medida, y $r \in (0, 1)$ razón de decrecimiento de la temperatura.
Output: Solución s.
$T :=$ temperatura inicial t_0;
$s :=$ solución inicials_0;
while *la temperatura* T *no sea cero* **do**
  for l *veces* **do**
    Seleccionar algún $s' \in \mathcal{N}(s)$;
    if $m(x, s') < m(x, s)$ **then**
      $s := s'$
    else
      $\delta = m(x, s') - m(x, s)$;
      $s := s'$ con probabilidad $\exp^{-\frac{\delta}{T}}$
  $T := rT$;

2.3. Fundamentos del cómputo cuántico

En esta sección se introducen los fundamentos básicos del cómputo cuántico tales como: espacios de Hilbert, transformaciones lineales; los postulados de la mecánica cuántica, el cómputo cuántico adiabático, el temple cuántico y finalmente, una breve descripción de la tecnología D-Wave.

2.3.1. Qubits y quregistros

Sea $\mathbb{H}_1 = \mathbb{C}^2$ un espacio de Hilbert de dimensión 1. Sean $|0\rangle = [\,1 \quad 0\,]^T$ y $|1\rangle = [\,0 \quad 1\,]^T$ una base de $\mathbb{H}_1$. Un *qubit* es una superposición lineal de los estados $|0\rangle$ y $|1\rangle$ dado por

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.10}$$

donde $\alpha, \beta \in \mathbb{C}$ y se cumple que $|\alpha|^2 + |\beta|^2 = 1$.

Sea $\mathbb{H}_n = \mathbb{H}_1 \otimes \mathbb{H}_{n-1}$ un espacio de Hilbert de dimensión n donde la operación $\otimes$ denota el producto tensorial. Dada $\sigma \in \{0,1\}^n$ una cadena con símbolos 0 y 1 de longitud n, se define el estado $|\sigma\rangle$ como

$$|\sigma\rangle = \bigotimes_{i=1}^{n} |\sigma_i\rangle \in \mathbb{H}_n. \tag{2.11}$$

El conjunto de estados $\{|\sigma\rangle\}_{\sigma\in\{0,1\}^n}$ forma una base de $\mathbb{H}_n$. Un *quregistro* es un vector $|\psi\rangle$ en $\mathbb{H}_n = \mathbb{C}^N$ con $N = 2^n$, representado por

$$|\psi\rangle = \sum_{\sigma\in\{0,1\}^n} \alpha_\sigma |\sigma\rangle \;\text{ con }\; \alpha_\sigma \in \mathbb{C}. \tag{2.12}$$

Un operador $T : \mathbb{H}_n \to \mathbb{H}_n$ es lineal si $T(\sum_{i=1}^k a_i |\psi_i\rangle) = \sum_{i=1}^k a_i T(|\psi_i\rangle)$ donde $|\psi_i\rangle \in \mathbb{H}_n$ y $a_i \in \mathbb{C}$. Todo operador lineal T se puede representar por medio de una matriz $T = (t_{ij}) \in \mathbb{C}^{N\times N}$ tal que

$$T(|\psi_i\rangle) = \sum_{j=1}^{N} t_{ij} |\psi_j\rangle \text{ para } i = 1, \ldots, N \tag{2.13}$$

y vectores $\{|\psi_k\rangle\}$ con $k = 1, \ldots, N$ forman una base ortonormal en $\mathbb{H}_n$. De aquí en adelante, para cada operador lineal T se usará su representación matricial. Un operador lineal T es *hermítico* si $T^\dagger = T$ y es *unitario* si $T^\dagger T = I_N$ donde $\dagger$ representa el conjugado transpuesto e I_N denota la matriz identidad de dimensión N.

Sea $T : \mathbb{H}_n \to \mathbb{H}_n$ un operador lineal y sea $|\psi\rangle \in \mathbb{H}_n$ un vector no nulo. Se dice que $|\psi\rangle$ es invariante bajo la acción de T si y sólo si $T|\psi\rangle = \lambda|\psi\rangle$ para alguna constante $\lambda \in \mathbb{C}$. A la constante λ se le llama *autovalor* de T y al vector $|\psi\rangle$ se le llama *autovector* de T asociado al autovalor λ. Se define por $\Lambda(T) = \{\lambda_1, \ldots, \lambda_N\}$ el *espectro* de T como el conjunto de todos los autovalores de T.

Algunos de los operadores sobre $\mathbb{H}_1$ más conocidos son las matrices de Pauli:

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \qquad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}. \tag{2.14}$$

2.3.2. Postulados de la mecánica cuántica

La *Mecánica Cuántica* es un modelo formal matemático para describir el mundo físico que nos rodea. Los postulados de la mecánica cuántica nos proporcionan una conexión entre la realidad y el formalismo matemático. Estos describen a los estados cuánticos, evolución cuántica y medición cuántica [41, 43].

– *Estados*: cualquier sistema físico se le asocia un espacio de estados, esto es, un espacio de Hilbert. La descripción completa de un sistema está dado por un vector de estado que corresponde a un vector complejo $|\psi\rangle$ en el espacio de Hilbert. El sistema cuántico más simple es el qubit.

– *Evolución*: este describe la forma en la que un estado $|\psi\rangle$ cambia en el tiempo. La evolución de un sistema cuántico cerrado se describe por medio de transformaciones unitarias. Es decir, la evolución del estado $|\psi\rangle$ en un tiempo inicial t_1 está relacionado con el estado $|\psi'\rangle$ en el tiempo t_2 por medio de una transformación unitaria U como

$$|\psi'\rangle = U\,|\psi\rangle\,. \tag{2.15}$$

Lo anterior se cumple siempre y cuando el sistema físico sea cerrado, es decir, un sistema que no interactúa con su entorno o con otros sistemas. Un sistema que evoluciona de acuerdo a (2.15) se le llama unitario o discreto.

Para sistemas físicos cerrados continuos en el tiempo, la evolución del estado $|\psi\rangle$ se describe por medio de la ecuación de Schrödinger como

$$i\hbar\frac{d\,|\psi\rangle}{dt} = H\,|\psi\rangle \tag{2.16}$$

donde $\hbar$ es la constante de Planck y H es un operador hermítico o *hamiltoniano* que describe la dinámica del sistema físico cerrado.

Dado que H es un operador hermítico, entonces por su descomposición espectral se tiene que

$$H = \sum_{\lambda} \lambda\,|\psi\rangle\,\langle\psi| \tag{2.17}$$

con autovalores λ y correspondientes autovectores $|\psi\rangle$. A los autovectores $|\psi\rangle$ se les conoce como *estados de energía* y a los autovalores λ como *energías* del estado $|\psi\rangle$. Al estado con energía más baja se le conoce como *estado firme*.

La evolución continua de un estado $|\psi\rangle$ desde un tiempo t_1 a un tiempo t_2 se puede ver como evolución discreta de la siguiente forma:

$$|\psi(t_2)\rangle = \exp\left[\frac{-iH(t_2-t_1)}{\hbar}\right]|\psi(t_1)\rangle \tag{2.18}$$

$$= U(t_1,t_2)\,|\psi(t_1)\rangle \tag{2.19}$$

donde se define el operador unitario

$$U(t_1,t_2) = \exp\left[\frac{-iH(t_2-t_1)}{\hbar}\right]. \tag{2.20}$$

– *Mediciones*: una medición cuántica es un evento probabilista que altera el estado cuántico de un sistema físico cerrado al ser observado. Las mediciones cuánticas se describen por una colección $\{M_m\}$ de operadores que actúan en el espacio de estados. Los subíndices m se refieren a todos los posibles resultados que se pueden obtener en la medición. Si el estado de un sistema físico cerrado es $|\psi\rangle$ antes de realizar una medición, entonces la probabilidad de que $|\psi\rangle$ se encuentre en el estado m es

$$p(|\psi\rangle = m) = \left\langle \psi \,\middle|\, M_m^\dagger M_m \,\middle|\, \psi \right\rangle \tag{2.21}$$

y el estado final después de la medición es

$$|\psi'\rangle = \frac{M_m\,|\psi\rangle}{\sqrt{\left\langle \psi \,\middle|\, M_m^\dagger M_M \,\middle|\, \psi \right\rangle}}. \tag{2.22}$$

La función p es una distribución de probabilidad ya que para cualquier m, $p(|\psi\rangle = m) \geq 0$ y $\sum_m p(|\psi\rangle = m) = 1$.

Por ejemplo, suponga que se quiere realizar una medición al estado cuántico $|\psi\rangle$ dado por $|\psi\rangle = \sum_{\sigma\in\{0,1\}^n} a_\sigma\,|\sigma\rangle \in \mathbb{H}_n$ sobre la base $\{|\sigma\rangle\}_{\sigma\in\{0,1\}^n}$. El conjunto de operadores de medición es $\{M_\sigma\}_{\sigma\in\{0,1\}^n}$ dados como $M_\sigma = |\sigma\rangle\,\langle\sigma|$. Así, la probabilidad

de que el estado $|\psi\rangle$ se encuentre en el estado $|\sigma\rangle$ es

$$p(|\psi\rangle = \sigma) = \left\langle \psi \,\middle|\, M_\sigma^\dagger M_\sigma \,\middle|\, \psi \right\rangle = \langle \psi \,|\, M_\sigma \,|\, \psi \rangle = |a_\sigma|^2 \tag{2.23}$$

y el estado después de la medición es $|\psi'\rangle = (1/|a_\sigma|^2) M_\sigma |\psi\rangle = (a_\sigma/|a_\sigma|)\,|\sigma\rangle$.

2.3.3. Cómputo cuántico adiabático

El cómputo cuántico adiabático (CCA) se introdujo como una técnica para resolver problemas de optimización combinatorios [17, 16]. El CCA usa el *Teorema Adiabático* para aproximar soluciones de la ecuación de Shrödinger. El Teorema Adiabático establece que si la evolución de un sistema cuántico, descrito por un hamiltoniano dependiente del tiempo, es lo suficientemente grande, el estado instantáneo del sistema se mantendrá cerca de su estado firme en cada instante de tiempo.

Los pasos principales de un algoritmo de CCA son los siguientes:

1. Construir un hamiltoniano inicial H_0 cuyo estado firme sea conocido y fácil de preparar.

2. Construir un hamiltoniano final H_f cuyas energías correspondan a los valores de medición del espacio de soluciones en un problema de optimización. De forma tal que el estado firme de H_f corresponda a la solución del problema dado.

3. Definir un hamiltoniano total dependiente del tiempo dado por

$$H(t) = (1 - \frac{t}{T})H_0 + \frac{t}{T}H_f \tag{2.24}$$

para un tiempo total T.

4. Evolucionar el sistema cuántico descrito por el hamiltoniano $H(t)$ desde un tiempo inicial $t_0 = 0$ a un tiempo final $t_f = T$, por medio de la ecuación de Schrödinger. Así, de acuerdo al Teorema Adiabático, si T es lo suficientemente grande, entonces el estado final del sistema estará cerca del estado firme de H_f.

5. Realizar una medición cuántica al estado final.

Observe que en el tiempo inicial t_0, en el hamiltoniano total $H(t)$ solo se tiene la contribución del hamiltoniano inicial H_0 y en el tiempo final t_f se tiene la contribución del hamiltoniano H_f. La evolución en el paso 4 debe entonces iniciar en el estado firme de H_0, razón por la cual debe ser fácil de preparar.

De manera general, cualquier problema de optimización combinatoria se puede expresar como la minimización/maximización de una función $f : \{0,1\}^n \to \mathbb{R}^+$ sobre todas las posibles cadenas σ en $\{0,1\}^n$. Para resolver el problema de optimización dado por la función f usando un algoritmo de CCA, se tienen que construir los hamiltonianos H_0 y H_f como sigue. El hamiltoniano $H_f : \mathbb{H}_n \to \mathbb{H}_n$ está dado por

$$H_f = \sum_{\sigma \in \{0,1\}^n} f(\sigma) \left|\sigma\right\rangle \left\langle\sigma\right| \tag{2.25}$$

tal que para cualquier cadena $\sigma \in \{0,1\}^n$ se satisface que $H_f \left|\sigma\right\rangle = f(\sigma) \left|\sigma\right\rangle$. De esta forma, el problema de minimización de la función f coincide con el problema de conocer el estado firme del hamiltoniano H_f.

Sea $W : \mathbb{H}_1 \to \mathbb{H}_1$ la transformación Hadamard cuya representación matricial es

$$W = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.26}$$

Se define por $W^{\otimes n} : \mathbb{H}_n \to \mathbb{H}_n$ el producto tensorial de la matriz Hadamard W consigo misma n veces. Entonces, el hamiltoniano inicial $H_0 : \mathbb{H}_n \to \mathbb{H}_n$ se puede construir como

$$H_0 = \sum_{\sigma \in \{0,1\}^n} h(\sigma)(W^{\otimes n} \left|\sigma\right\rangle)(W^{\otimes n} \left|\sigma\right\rangle)^\dagger \tag{2.27}$$

para cualquier función $h : \{0,1\}^n \to \mathbb{R}^+$ tal que $h(0^n) = 0$ y $h(\sigma) \geq 1$ con $\sigma \in \{0,1\}^n - \{0^n\}$. El estado firme de H_0 es

$$\left|\psi_0\right\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{\sigma \in \{0,1\}^n} \left|\sigma\right\rangle, \tag{2.28}$$

que se puede construir fácilmente por medio de la transformada de Hadamard como $\left|\psi_0\right\rangle = W^{\otimes n} \left|0^n\right\rangle$.

2.3.4. Algoritmo de temple cuántico

En el algoritmo TS se usan fluctuaciones térmicas para explorar el espacio de soluciones con respecto a una función de costo de un problema de optimización dado. Mientras que el algoritmo de *temple cuántico* (TC) usa el proceso cuántico de tunel para hacer transiciones de estados y así explorar el espacio de soluciones de manera más eficiente [19, 33, 47].

El algoritmo TC se puede describir por el siguiente hamiltoniano

$$H = -\sum_{1\leq i<j\leq n} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z - h\sum_{i=1}^{n}\hat{\sigma}_i^z - \Gamma(t)\sum_{i=1}^{n}\hat{\sigma}_i^x \tag{2.29}$$

donde $\Gamma(t)$ es una función dependiente del tiempo que controla el efecto cuántico de tunel; J_{ij} y h son coeficientes que describen la fuerza de interacción entre parejas de qubits y la magnitud del campo longitudinal, respectivamente. Las transformaciones $\hat{\sigma}_i^z : \mathbb{H}_n \to \mathbb{H}_n$ y $\hat{\sigma}_i^x : \mathbb{H}_n \to \mathbb{H}_n$ se definen como

$$\hat{\sigma}_i^z = I_2 \otimes \cdots \underbrace{\otimes\sigma_z\otimes}_{i\text{-ésimo qubit}} \cdots \otimes I_2$$

y

$$\hat{\sigma}_i^x = I_2 \otimes \cdots \underbrace{\otimes\sigma_x\otimes}_{i\text{-ésimo qubit}} \cdots \otimes I_2$$

donde σ_z, σ_x son las transformaciones de Pauli que actuan sobre el i-ésimo qubit, según sea el caso.

Los dos primeros términos en (2.29) corresponden al llamado modelo Ising con un campo magnético longitudinal, y el último término corresponde a un campo magnético transversal cuya fuerza es controlado por la función $\Gamma(t)$. Entonces, la idea del algoritmo TC es evolucionar el sistema cuántico descrito por el hamiltoniano H, variando $\Gamma(t)$ desde un valor grande hasta cero, y se espera que el sistema en algún momento se aproxime al estado firme de H.

2.3.5. La tecnología D-Wave

La tecnología D-Wave implementa físicamente el proceso de TC para resolver instancias del modelo Ising. El hardware cuántico usa qubits superconductores a muy bajas temperaturas organizados sobre una topología específica con interacción local [26, 27].

El comportamiento del hardware cuántico se modela por el hamiltoniano

$$H(\tau) = A(\tau)H_0 + B(\tau)H_p \tag{2.30}$$

donde H_0 representa el campo magnético transversal, H_p es una instancia del modelo Ising, las funciones $A(\tau)$ y $B(\tau)$ controlan la contribución de los hamiltonianos H_0 y H_p durante el proceso de evolución con razón de tiempo $\tau = t/T$ para $0 \leq t \leq T$ y tiempo total T. La Figura 2.4 muestra el comportamiento de las funciones de evolución $A(\tau)$ y $B(\tau)$ implementadas por el hardware cuántico.

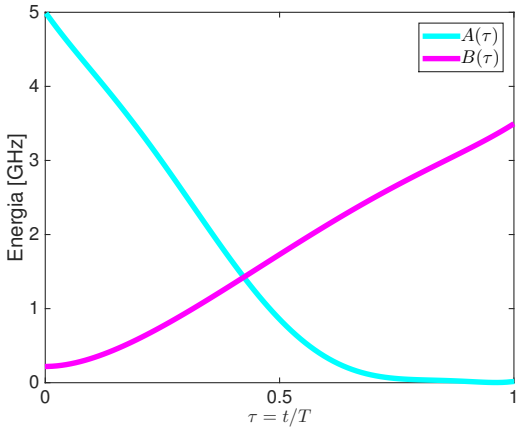


Figura 2.4: Funciones de evolución del proceso de temple cuántico [32]. Aquí se observa que conforme el tiempo evoluciona desde $t = 0$ hasta $t = 1$, la influencia del hamiltoniano H_0 (H_p) es predominante (nulo) hasta anularse (ser predominante), respectivamente.

La computadora D-Wave ejecuta un algoritmo de TC para aproximar el estado firme de un modelo Ising de la forma

$$H_p = \sum_{1 \leq i < j \leq n} J_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z + \sum_{i=1}^{n} h_i\hat{\sigma}_i^z \tag{2.31}$$

que corresponde a encontrar la mínima energía de la función clásica

$$E(\mathbf{s}) = \sum_{1 \leq i < j \leq n} J_{ij} s_i s_j + \sum_{i=1}^{n} h_i s_i \tag{2.32}$$

donde $\mathbf{s} = (s_1, \ldots, s_n)$ con $s_i \in \{-1, 1\}$, $J_{ij} \in [-1, 1]$ es la fuerza de unión entre las variables s_i y s_j, y $h_i \in [-1, 1]$ es la magnitud de la componente longitudinal del modelo Ising para la variable s_i. De hecho, el problema de encontrar valores -1/1 para las variables $s_1, \ldots, s_n$ tal que la función (2.32) sea mínima es un problema NP-difícil [5].

Un problema equivalente al modelo clásico de Ising (2.32) es el modelo QUBO (*Quadratic Unconstrained Binary Optimization*) dado por

$$\mathcal{E}(\mathbf{x}) = \sum_{1 \leq i < j \leq n} J_{ij} x_i x_j + \sum_{i=1}^{n} h_i x_i, \tag{2.33}$$

por medio de un cambio de variables dado por $s_i = 1 - 2x_i$. Cualquier instancia del modelo QUBO/Ising con n variables se puede representar como una gráfica ponderada $G = (V, E)$ tal que $V = \{1, \ldots, n\}$ y $E = \{\{i, j\} \mid J_{ij} \neq 0\}$, con pesos sobre los vértices $i \in V : i \mapsto h_i$ y pesos sobre las aristas $\{i, j\} \in E : \{i, j\} \mapsto J_{ij}$.

La forma en la que se programa la computadora D-Wave consiste en representar un problema de optimización como un problema de optimización de una función QUBO/Ising sobre variables booleanas/Ising. Una vez obtenida una representación QUBO/Ising, se asignan valores a los coeficientes J_{ij} y h_i para configurar el hardware cuántico. La computadora D-Wave puede resolver instancias del modelo QUBO/Ising cuyas gráficas correspondientes G se puedan representar sobre la topología del hardware cuántico.

Capítulo 3

Formulación cuántica del problema max-SAT

En este capítulo se muestra el modelado y simulación del problema max-SAT. Primero, se describe brevemente el problema de decisión de satisfactibilidad booleana, y su versión de optimización, max-SAT. El modelado matemático propuesto consiste en la construcción de una función pseudo-booleana cuya optimización corresponda a resolver el problema de estudio. Se aplicaron técnicas de reducción de grado a la función pseudo-boolean propuesta, con el fin de obtener una función pseudo-booleana cuadrática que pueda ser optimizada usando el enfoque del computo cuántico adiabático. Se consideraron las versiones de optimización de los problemas 3-SAT y 1-en-3-SAT. También, se muestra una comparación de eficiencia para resolver instancias difíciles del problema max-SAT usando algoritmos del estado del arte y el algoritmo de temple simulado.

3.1. El problema SAT y max-SAT

El problema SAT es considerado como el primer problema NP-completo demostrado por Cook-Levin [10] en 1971. En 1972, Karp [35] introdujo 21 problemas que se consideran NP-completos, donde figura el problema SAT como el primero de ellos. SAT es un proble-

ma de gran importancia práctica, con aplicaciones que abarcan desde la prueba de chips y el diseño de computadoras hasta la ingeniería de imágenes y software [46].

Sea $\mathcal{X} = (x_j)_{j=1}^n$ un conjunto de n variables booleanas. Una *literal* tiene la forma x^δ con $x \in \mathcal{X}$ y $\delta \in \{0,1\}$ tal que $x^1 = x$ y $x^0 = \overline{x}$. Una *cláusula* es una disyunción de literales, y una *forma conjuntiva* (FC) es una conjunción de cláusulas. Una *asignación* $\varepsilon = (\varepsilon)_{j=1}^n \in \{0,1\}^n$ es un punto en el hipercubo de n dimensiones. Tal asignación satisface la literal x_j^δ si y sólo si $\varepsilon_j = \delta$; satisface una cláusula siempre que satisface al menos una literal en la cláusula; y satisface una FC siempre que satisfaga todas las cláusulas de la FC. Una m-cláusula es una cláusula que consiste exactamente en m literales, y una m-FC es una FC que consta de solo m-cláusulas.

El problema *satisfactibilidad booleana (SAT)* se enuncia como sigue:

Problema 1. *SAT*

Input: Una FC Φ sobre un conjunto $\mathcal{X} = (x_j)_{j=1}^n$ de n variables booleanas.

Objetivo: Decidir si existe una asignación $\varepsilon \in \{0,1\}^n$ tal que $\Phi(\varepsilon) = 1$.

Un ejemplo de una 3-FC definida sobre 7 variables y 27 cláusulas:

$$\begin{aligned}\Phi \;=\; & (\overline{x}_2 \vee \overline{x}_6 \vee x_7) \wedge (\overline{x}_1 \vee \overline{x}_4 \vee \overline{x}_6) \wedge (x_1 \vee \overline{x}_6 \vee \overline{x}_7) \wedge (x_2 \vee x_5 \vee \overline{x}_7) \wedge \\ & (x_3 \vee \overline{x}_5 \vee \overline{x}_6) \wedge (\overline{x}_2 \vee x_5 \vee \overline{x}_7) \wedge (x_1 \vee x_2 \vee x_6) \wedge (\overline{x}_4 \vee \overline{x}_5 \vee \overline{x}_7) \wedge \\ & (\overline{x}_2 \vee x_3 \vee x_4) \wedge (x_1 \vee x_4 \vee \overline{x}_5) \wedge (x_2 \vee \overline{x}_3 \vee \overline{x}_6) \wedge (\overline{x}_1 \vee x_2 \vee x_6) \wedge \\ & (\overline{x}_2 \vee \overline{x}_5 \vee x_6) \wedge (x_1 \vee \overline{x}_3 \vee x_5) \wedge (\overline{x}_1 \vee x_3 \vee \overline{x}_6) \wedge (x_4 \vee \overline{x}_5 \vee \overline{x}_6) \wedge \\ & (\overline{x}_2 \vee x_4 \vee \overline{x}_5) \wedge (\overline{x}_3 \vee x_6 \vee x_7) \wedge (\overline{x}_4 \vee x_5 \vee \overline{x}_6) \wedge (\overline{x}_1 \vee x_5 \vee \overline{x}_6) \wedge \\ & (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_5) \wedge (x_3 \vee \overline{x}_4 \vee \overline{x}_6) \wedge (\overline{x}_1 \vee x_4 \vee x_5) \wedge (x_1 \vee \overline{x}_4 \vee x_7) \wedge \\ & (\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_6) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_5) \wedge (\overline{x}_2 \vee x_4 \vee x_6)\end{aligned}$$

El problema SAT es NP-completo [10] y 3-SAT (la restricción de SAT a 3-FC's) también es NP-completo [39]. Sin en cambio, 2-SAT se puede resolver en tiempo polinomial. La versión de optimización del SAT, llamado *max-SAT*, consiste en satisfacer la

mayor cantidad de cláusulas en una FC dada. El problema max-SAT es NP-difícil [21]. De manera formal:

Problema 2. *max-SAT*

Input: Una FC Φ *sobre un conjunto* $\mathcal{X} = (x_j)_{j=1}^n$ *de* n *variables booleanas.*

Objetivo: Encontrar una asignación $\varepsilon \in \{0,1\}^n$ *tal que* ε *satisfaga el mayor número posible de cláusulas en* Φ.

3.2. Formulación QUBO del problema max-SAT

Para aproximar soluciones del problema max-SAT usando el algoritmo de TC o por medio de la tecnología D-Wave, se necesita modelar cualquier instancia del problema max-SAT como instancia equivalentes del modelo QUBO/Ising. Aquí, se considera la construcción de una función pseudo-booleana de grado igual a 3 para instancias 3-SAT y su versión de optimización. La función pseudo-booleana mencionada es convertida a un modelo o función QUBO por medio de la aplicación de métodos de reducción de grado.

3.2.1. Construcción de una función pseudo-booleana para el max-SAT

Sea $\Phi = \bigwedge_{j=1}^{t} C_j$ una m-FC con t cláusulas. Asuma que sin pérdida de generalidad, cada cláusula es de la forma $C_j = x_1^{\delta_1} \vee \cdots \vee x_m^{\delta_m}$, sea $h_{C_j} : \{0,1\}^n \to \mathbb{R}^+ \cup \{0\}$ una función tal que por cada asignación $\varepsilon \in \{0,1\}^n : h_{C_j}(\varepsilon) = 0$ si ε satisface a C_j, y $h_{C_j}(\varepsilon) = 1$ en otro caso.

La función h_{C_j} se puede definir como

$$h_{C_j} = \prod_{k=1}^{m} (1 - x_k^{\delta_k}). \tag{3.1}$$

Sea $h_\Phi : \{0,1\}^n \to \mathbb{R}^+ \cup \{0\}$ dada por

$$h_\Phi = \sum_{j=1}^{t} h_{C_j} \tag{3.2}$$

tal que $h_\Phi(\varepsilon) = 0$ si y sólo si Φ satisface por alguna asignación $\varepsilon \in \{0,1\}^n$. Note que la función $h_\Phi(\varepsilon)$ representa el número de cláusulas que no se satisfacen.

El problema max-SAT se puede reformular en términos de la función h_Φ como:

Problema 3. *max-SAT (versión 2)*

Input: Una FC Φ sobre un conjunto $\mathcal{X} = (x_j)_{j=1}^n$ de n variables booleanas.

Objetivo: Encontrar una asignación $\varepsilon \in \{0,1\}^n$ tal que $h_\Phi(\varepsilon)$ sea mínimo.

La expansión de la función h_{C_j} dada en (3.1) resulta en un polinomio multivariado de grado m. Es decir, un polinomio o *función pseudo-booleana* de la forma

$$f(\mathbf{x}) = \sum_{S \subseteq \{1,\ldots,n\}} c(S) \prod_{j \in S} x_j \tag{3.3}$$

donde $\mathbf{x} = (x_1, \ldots, x_n)$ y $c(S)$ es un coeficiente real. Aquí, el grado de f, denotado por $\deg(f)$, es igual a la cardinalidad del subconjunto S más grande para el cual $c(S) \neq 0$. Note que el modelo QUBO dado en (2.33) es una función pseudo-booleana de grado 2.

De este modo, para obtener una formulación cuántica del problema max-SAT, dentro del enfoque del cómputo cuántico adiabático o TC, se necesita reducir el grado de la función h_Φ dada en (3.2). En la siguiente sección, se muestran algunas de las técnicas de reducción más conocidas y que se aplicaron en este trabajo de tesis.

3.2.2. Métodos de reducción

Muchos de los problemas de optimización combinatorios se pueden expresar como el siguiente problema de minimización

$$\min_{\mathbf{x} \in \{0,1\}^n} f(\mathbf{x}) \tag{3.4}$$

para la función pseudo-booleana $f : \{0,1\}^n \to \mathbb{R}$ como en (3.3). Asumiendo que $\deg(f) > 2$, entonces la reducción de f a un problema de minimización de una función cuadrática consiste en encontrar $g : \{0,1\}^{n+m} \to \mathbb{R}$ con $\deg(g) \leq 2$ tal que

$$\forall \mathbf{x} \in \{0,1\}^n : f(\mathbf{x}) = \min_{\mathbf{w} \in \{0,1\}^m} g(x_1, \ldots, x_n, w_1, \ldots, w_m) \tag{3.5}$$

donde $(w_1, \ldots, w_m)$ es un conjunto de variables nuevas y $(x_1, \ldots, x_n)$ comúnmente se les llaman variables originales del problema.

En [20], Freedman y Drineas proponen un método para reducir cualquier término negativo de grado d a un término cuadrático usando una sola variable nueva. Así, para cualquier monomio negativo $-x_1 \cdots x_d$ se tiene que

$$-\prod_{i=1}^{d} x_i = \min_{w \in \mathbb{B}} w \left((d-1) - \sum_{j=1}^{d} x_j \right) \tag{3.6}$$

donde w es una nueva variable booleana. También se tiene que para cada asignación de minimización de f hay un mínimo correspondiente de la forma cuadrática g obtenida por las aplicaciones repetidas de (3.6) que satisfacen $w = x_1 x_2 \ldots x_d$.

Por otro lado, Ishiwaka [30] propone un método para reducir términos positivos de grado d utilizando pocas variables adicionales. De manera precisa, sea $t(x) = x_1 x_2 \cdots x_d$ de grado d, $k = \lfloor \frac{d-1}{2} \rfloor$ y $w = (w_1, w_2, \ldots, w_k)$ son k variables nuevas. Defina,

$$\begin{aligned} S_1 &= \sum_{j=1}^{d} x_j, \\ S_2 &= \sum_{1 \le i < j \le d} x_i x_j, \\ A &= \sum_{j=1}^{k} w_j, \\ B &= \sum_{j=1}^{k} (4j-1) w_j. \end{aligned}$$

Entonces, se tienen las siguientes igualdades:

$$\prod_{j=1}^{d} x_j = S_2 + \min_{w \in \mathbb{B}^k} \{ B - 2AS_1 \} \tag{3.7}$$

si $d = 2k + 2$, mientras que

$$\prod_{j=1}^{d} x_j = S_2 + \min_{w \in \mathbb{B}^k} \{ B - 2AS_1 + w_k(S_1 - d + 1) \} \tag{3.8}$$

si $d = 2k + 1$. S_1 y S_2 son funciones simétricas de x y A una función simétrica de w. Este método introduce $d - 2$ variables nuevas.

3.2.3. Reducción de la función h_Φ

Como se mencionó anteriormente, el problema 3-SAT es NP-completo. Aquí, consideramos la versión de optimización de 3-SAT para simplificar nuestro análisis, el cual también se puede demostrar que es un problema NP-difícil [21].

Para el problema 3-SAT tenemos cláusulas de la forma $C_j = x_1^{\delta_1} \vee x_2^{\delta_2} \vee x_3^{\delta_3}$. Por lo tanto, h_{C_j} puede escribir como

$$\begin{aligned} h_{C_j} &= \delta_1\delta_2\delta_3 + (-1)^{2-\delta_1}\delta_2\delta_3 x_1 + (-1)^{2-\delta_2}\delta_1\delta_3 x_2 + \\ &\quad (-1)^{2-\delta_3}\delta_1\delta_2 x_3 + (-1)^{4-\delta_1-\delta_2}\delta_3 x_1 x_2 + \\ &\quad (-1)^{4-\delta_1-\delta_3}\delta_2 x_1 x_3 + (-1)^{4-\delta_2-\delta_3}\delta_1 x_2 x_3 + \\ &\quad (-1)^{6-\delta_1-\delta_2-\delta_3} x_1 x_2 x_3. \end{aligned} \tag{3.9}$$

Como se puede ver en (3.9), hay un término cúbico $ax_1x_2x_3$ que se puede convertir en uno cuadrático agregando una variable adicional al problema. Observe que el coeficiente $a = (-1)^{6-\delta_1-\delta_2-\delta_3}$ puede ser positivo o negativo según los valores de $\delta_k, k = 1, 2, 3$.

Tabla 3.1: Tabla de verdad para los términos $x_1x_2x_3$ y $x_1x_2 + x_1x_3 + x_2x_3 + z_j(1 - x_1 - x_2 - x_3)$.

z_j	x_1	x_2	x_3	$x_1x_2x_3$	$x_1x_2 + x_1x_3 + x_2x_3 + z_j(1 - x_1 - x_2 - x_3)$
0 (1)	0	0	0	0	0 (1)
0 (1)	0	0	1	0	0 (0)
0 (1)	0	1	0	0	0 (0)
0 (1)	0	1	1	0	1 (0)
0 (1)	1	0	0	0	0 (0)
0 (1)	1	0	1	0	1 (0)
0 (1)	1	1	0	0	1 (0)
0 (1)	1	1	1	1	3 (1)

En este trabajo de investigación, usamos los métodos de reducción de Ishikawa [30] y de Freedman [20] cuando el coeficiente a es positivo y negativo, respectivamente. Estos dos métodos fueron elegidos ya que introducen coeficientes pequeños a la expresión reducida. El método de reducción de Rosenberg [44] se puede usar para el caso general de la función h_Φ. La reducción de la función dada en (3.9) es como sigue:

- Si $a > 0$, entonces el término positivo $x_1x_2x_3$ se puede expresar como

$$\begin{aligned} x_1x_2x_3 &= x_1x_2 + x_1x_3 + x_2x_3 + \min_{z_j\in\{0,1\}} z_j(1 - x_1 - x_2 - x_3) \\ &\leq x_1x_2 + x_1x_3 + x_2x_3 + z_j(1 - x_1 - x_2 - x_3) \end{aligned} \tag{3.10}$$

- Si $a < 0$, entonces el término negativo $-x_1x_2x_3$ se puede expresar como

$$\begin{aligned} -x_1x_2x_3 &= \min_{z_j\in\{0,1\}} z_j\,(2 - x_1 - x_2 - x_3) \\ &\leq z_j\,(2 - x_1 - x_2 - x_3) \end{aligned} \tag{3.11}$$

  donde z_j es una variable nueva por cada cláusula C_j.

Tabla 3.2: Tabla de verdad para los términos $-x_1x_2x_3$ y $z_j\,(2 - x_1 - x_2 - x_3)$.

z_j	x_1	x_2	x_3	$-x_1x_2x_3$	$z_j\,(2 - x_1 - x_2 - x_3)$
$0\,(1)$	0	0	0	0	$0\,(2)$
$0\,(1)$	0	0	1	0	$0\,(1)$
$0\,(1)$	0	1	0	0	$0\,(1)$
$0\,(1)$	0	1	1	0	$0\,(0)$
$0\,(1)$	1	0	0	0	$0\,(1)$
$0\,(1)$	1	0	1	0	$0\,(0)$
$0\,(1)$	1	1	0	0	$0\,(0)$
$0\,(1)$	1	1	1	-1	$0\,(-1)$

En la Tabla 3.1, se puede verificar que siempre haya un valor para la variable z_j cuyo término $x_1x_2 + x_1x_3 + x_2x_3 + z_j(1 - x_1 - x_2 - x_3)$ es mayor o igual que el término cúbico

positivo $x_1x_2x_3$. Esta última propiedad afirma que si reemplazamos el lado izquierdo con el lado derecho de (3.10) en h_{C_j}, entonces se conserva el mínimo de h_{C_j}. Esto también es cierto cuando el término cúbico es negativo, como se puede verificar en la Tabla 3.2.

Finalmente, usando (3.10) and (3.11) podemos obtener una expresión QUBO h_Φ^{qubo} por cada FC $\Phi = \bigwedge_{j=1}^{t} C_j$. La construcción de la función h_Φ^{qubo} requiere al menos $n+t$ variables, entre ellas t son nuevas variables. Por lo tanto, el problema max-SAT es equivalente a minimizar la función h_Φ^{qubo} sobre $n+t$ variables.

Consideremos un ejemplo de una instancia 3-SAT con 30 cláusulas y 7 variables. Sea Φ_1 una FC dada por

$$\begin{aligned}\Phi_1 \;=\;& (\overline{x}_1 \vee x_2 \vee \overline{x}_6) \wedge (\overline{x}_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge (x_1 \vee \overline{x}_3 \vee x_5) \wedge (x_1 \vee \overline{x}_5 \vee \overline{x}_7) \wedge \\ & (x_1 \vee x_3 \vee \overline{x}_6) \wedge (\overline{x}_2 \vee \overline{x}_4 \vee \overline{x}_7) \wedge (x_2 \vee x_4 \vee \overline{x}_7) \wedge (x_1 \vee x_3 \vee x_4) \wedge \\ & (\overline{x}_2 \vee x_5 \vee x_7) \wedge (x_2 \vee \overline{x}_5 \vee x_7) \wedge (x_1 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee \overline{x}_4 \vee x_6) \wedge \\ & (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_5) \wedge (\overline{x}_1 \vee \overline{x}_4 \vee \overline{x}_7) \wedge (x_1 \vee \overline{x}_5 \vee x_7) \wedge (\overline{x}_5 \vee \overline{x}_6 \vee x_7) \wedge \\ & (x_1 \vee \overline{x}_4 \vee \overline{x}_7) \wedge (x_1 \vee x_2 \vee \overline{x}_7) \wedge (x_3 \vee x_6 \vee x_7) \wedge (x_1 \vee x_5 \vee \overline{x}_6) \wedge \\ & (x_2 \vee \overline{x}_5 \vee x_6) \wedge (\overline{x}_1 \vee x_6 \vee \overline{x}_7) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \overline{x}_4) \wedge \\ & (\overline{x}_2 \vee \overline{x}_3 \vee x_4) \wedge (x_1 \vee x_3 \vee x_5) \wedge (x_1 \vee x_2 \vee x_7) \wedge (x_1 \vee \overline{x}_5 \vee \overline{x}_6) \wedge \\ & (x_1 \vee x_4 \vee \overline{x}_7) \wedge (\overline{x}_1 \vee x_2 \vee x_7). \end{aligned} \tag{3.12}$$

La expresión cúbica correspondiente h_{Φ_1} como en (3.2) para Φ_1 es

$$\begin{aligned} h_{\Phi_1} \;=\;& 4 - x_3 + x_6 + x_7 - 2x_1 + x_4 + 2x_5 - x_1x_6 + x_1x_3 - x_5x_7 - 2x_2x_7 - \\ & x_4x_7 - x_1x_4 - 3x_2x_5 - x_1x_7 + x_3x_7 + x_6x_7 - x_2x_4 - x_1x_2x_6 + \\ & x_3x_4x_5 + x_1x_3x_6 + 2x_2x_4x_7 + 2x_2x_5x_7 - x_3x_4x_6 + x_1x_2x_5 + x_1x_4x_7 - \\ & x_5x_6x_7 + x_1x_2x_7 - x_3x_6x_7 + x_2x_5x_6 - x_1x_6x_7 + x_1x_2x_4. \end{aligned} \tag{3.13}$$

Aplicando los métodos de reducción en las ecuaciones (3.10) y (3.11) a la función

h_{Φ_1}, obtenemos

$$
\begin{aligned}
h_{\Phi_1}^{\text{qubo}} \;=\; & 4 - x_3 + x_6 + x_7 - 2x_1 + x_4 + 2x_5 + 2x_8 + x_9 + x_{10} + 2x_{11} + 2x_{12} + \\
& 2x_{13} + x_{14} + x_{15} + 2x_{16} + x_{17} + 2x_{18} + x_{19} + 2x_{20} + x_{21} + 2x_1x_3 + \\
& x_5x_7 + 3x_2x_7 + 2x_4x_7 + x_1x_4 + x_2x_5 + x_1x_7 + x_3x_7 + x_6x_7 + \\
& 2x_2x_4 - x_1x_8 - x_2x_8 - x_6x_8 + x_3x_4 + x_3x_5 + x_4x_5 - x_3x_9 - x_4x_9 - \\
& x_5x_9 + x_3x_6 - x_1x_{10} - x_3x_{10} - x_6x_{10} - 2x_2x_{11} - 2x_4x_{11} - 2x_7x_{11} - \\
& 2x_2x_{12} - 2x_5x_{12} - 2x_7x_{12} - x_3x_{13} - x_4x_{13} - x_6x_{13} + 3x_1x_2 + x_1x_5 - \\
& x_1x_{14} - x_2x_{14} - x_5x_{14} - x_1x_{15} - x_4x_{15} - x_7x_{15} - x_5x_{16} - x_6x_{16} - \\
& x_7x_{16} - x_1x_{17} - x_2x_{17} - x_7x_{17} - x_3x_{18} - x_6x_{18} - x_7x_{18} + x_2x_6 + \\
& x_5x_6 - x_2x_{19} - x_5x_{19} - x_6x_{19} - x_1x_{20} - x_6x_{20} - x_7x_{20} - x_1x_{21} - \\
& x_2x_{21} - x_4x_{21}. \qquad (3.14)
\end{aligned}
$$

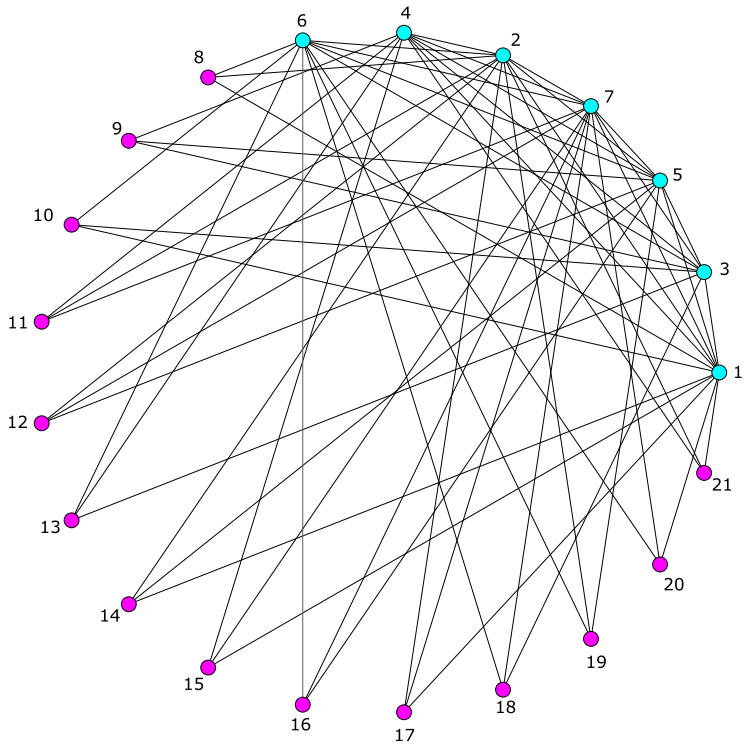


Figura 3.1: Gráfica lógica $G_{\Phi_1} = (V, E)$ asociada a la función $h_{\Phi_1}^{\text{qubo}}$ dada en (3.14).

En la Figura 3.1 se muestra la gráfica lógica $G_{\Phi_1} = (V, E)$ asociada a la función QUBO $h^{\text{qubo}}_{\Phi_1}$ dada en (3.14) donde los vértices de color cyan son las variables originales del problema y los vértices de color magenta son las variables ancillas o auxiliares. Observe también que cada variable ancilla es adyacente a exactamente 3 variables originales del problema, y las auxiliares no son adyacentes entre sí.

3.2.4. 1-en-3 SAT

El problema 1-en-3 SAT es una variante de 3-SAT el cual consiste en satisfacer exactamente una literal en cada cláusula. Se sabe que 1-en-3 SAT es NP-completo y su versión de optimización es NP-difícil [21]. Por construcción, la función h_{C_j} dada en (3.9) no se puede usar para determinar si la cláusula C_j se satisface cuando una sola literal es verdadera. Por lo tanto, para cualquier cláusula $C_j = x_1^{\delta_1} \vee x_2^{\delta_2} \vee x_3^{\delta_3}$ sobre $\mathcal{X} = (x_j)_{j=1}^n$, se define la función h'_{C_j} tal que $h'_{C_j}(\varepsilon) = 0$ si se satisface exactamente una literal en C_j, y $h'_{C_j}(\varepsilon) = 1$ en otro caso, para toda asignación $\varepsilon \in \{0,1\}^n$.

La Tabla 3.3 muestra los valores de verdad que la función h'_{C_j} debe satisfacer con respecto a las variables x_1, x_2 y x_3. Para encontrar la forma de la función h'_{C_j} se puede usar el método de Karnaugh [45], el cual obtiene un circuito booleano que coincide con las entradas y salidas de una tabla de verdad dada.

La expresión booleana de la función h'_{C_j} es la siguiente:

$$h'_{C_j}(x_1, x_2, x_3) = 1 - x_1 - x_2 - x_3 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3 - 3x_1x_2x_3, \tag{3.15}$$

cuyos valores de verdad coinciden con la Tabla 3.3. Note que el último término de h'_{C_j} es cúbico, el cual se puede reducir a un término cuadrático como se muestra en la Tabla 3.2. La reducción de la función h'_{C_j} queda como

$$\begin{aligned} h^{\text{qubo'}}_{C_j}(x_1, x_2, x_3) &= 1 - x_1 - x_2 - x_3 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3 + \\ &\quad 3w_j(2 - x_1 - x_2 - x_3) \end{aligned} \tag{3.16}$$

donde w_j es una nueva variable del problema.

x_1	x_2	x_3	$h'_{C_j}(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Tabla 3.3: Tabla de verdad de la función booleana h'_{C_j}.

Para obtener una función h'_{C_j} en términos de las variables $x_1^{\delta_1}, x_2^{\delta_2}, x_3^{\delta_3}$, basta con sustituir las variables $x_j^{\delta_j} := \delta_j - (-1)^{1-\delta_j} x_j$ para $j = 1, 2, 3$ en la expresión (3.16). Por ejemplo,

$$\begin{aligned} h_{C_j}^{\text{qubo'}} &= 1 - \delta_1 + 2\delta_1\delta_2 - \delta_2 + 2\delta_1\delta_3 + 2\delta_2\delta_3 - \delta_3 - (-1)^{2-\delta_1} x_1 + \\ &\quad 2(-1)^{-\delta_1-\delta_2+4} x_2 x_1 + 2(-1)^{-\delta_1-\delta_3+4} x_3 x_1 + 2(-1)^{2-\delta_1}\delta_2 x_1 + \\ &\quad 2(-1)^{2-\delta_1}\delta_3 x_1 - (-1)^{2-\delta_2} x_2 - (-1)^{2-\delta_3} x_3 + 2(-1)^{-\delta_2-\delta_3+4} x_2 x_3 + \\ &\quad 2(-1)^{2-\delta_2}\delta_1 x_2 + 2(-1)^{2-\delta_3}\delta_1 x_3 + 2(-1)^{2-\delta_3}\delta_2 x_3 + 2(-1)^{2-\delta_2}\delta_3 x_2 + \\ &\quad 3w_j(2 - \delta_1 - \delta_2 - \delta_3 - (-1)^{2-\delta_1} x_1 - (-1)^{2-\delta_2} x_2 - (-1)^{2-\delta_3} x_3) \end{aligned} \tag{3.17}$$

Finalmente, dada una m-FC $\Phi = \bigwedge_{j=1}^{t} C_j$, se tiene la función $h_\Phi^{\text{qubo'}}$ para el problema 1-en-3 SAT dada por

$$h_\Phi^{\text{qubo'}} = \sum_{j=1}^{t} h_{C_j}^{\text{qubo'}}. \tag{3.18}$$

Es importante mencionar que las funciones dadas en (3.2) y (3.18), son tales que existen asignaciones que las minimizan y cuya evaluación corresponden al máximo número de cláusulas que se pueden satisfacer. Sin embargo, para cualquier otra asignación no óptima, su evaluación no necesariamente corresponde al número de cláusulas que satisface. Por

lo tanto, se tiene que realizar un procedimiento de verificación para aceptar asignaciones cuya evaluación de su función pseudo-booleana cuadrática correspondiente coincida con el número de cláusulas que satisface.

3.3. Simulación

En esta sección se presentan simulaciones que se realizaron con el propósito de comparar tres herramientas diferentes para encontrar soluciones al problema max-SAT con instancias 3-FC, esto con el objetivo de conocer las soluciones óptimas por medio de un *solver* o algoritmo clásico y las soluciones obtenidas con representaciones QUBO. Estas herramientas son las siguientes:

- *ahmaxsat*[1]: Es un algoritmo usado para resolver instancias del problema max-SAT que detecta inconsistencias en subconjuntos usando simulaciones unitarias de propagación basado en el algoritmo heurístico Jeroslow-Wang [1, 31]. Es considerado uno de los mejores algoritmos para el problema max-SAT en instancias aleatorias de los últimos años, obtuvo el primer lugar en tres categorías de la *MAX-SAT Evaluation* en los años 2014 y 2015.
- *qbsolv*[2]: Es un algoritmo híbrido (cuántico/clásico) de código abierto que aproxima soluciones a instancias del modelo QUBO. Este algoritmo particiona instancias del modelo QUBO en subproblemas más pequeños que pueden ser resueltos ya sea de manera clásica usando búsqueda Tabu o por medio del algoritmo de TC. La ejecución del algoritmo TC se ejecutan en el hardware cuántico de D-Wave que pueden ser representadas usando los recursos cuánticos disponibles.
- *dwave-sapi*[3]: Es un conjunto de librerías cliente que se utiliza para interactuar con el sistema D-Wave de manera remota/local. Las bibliotecas cliente utilizan un para-

[1] http://www.lsis.org/abramea/ressources.html

[2] https://github.com/dwavesystems/qbsolv

[3] https://docs.dwavesys.com/docs/latest/c_ml_2.html

digma tradicional de cliente-servidor donde el código de la aplicación se ejecuta en un sistema cliente, y los comandos del cliente se traducen en llamadas REST/HTTP y luego se transmiten del cliente al servidor. Esta herramienta permite resolver instancias del modelo QUBO/Ising directamente sobre el hardware cuántico. También permite resolver instancias QUBO/Ising de manera local por medio del algoritmo de temple cuántico simulado (TCS) [12].

En este trabajo se usó una computadora sobre el sistema operativo Windows con un procesador Intel Celeron$^{\text{TM}}$ N2820 a 2.13 GHz y 4 GB de RAM. Se usaron los lenguajes de programación C, Python, Mathematica y Matlab.

3.3.1. Generación de instancias aleatorias

Sea $A_{k,n}$ el conjunto de todas las $2^k\binom{n}{k}$ cláusulas con exactamente k literales distintas y no-complementarias sobre n variables booleanas. Se denota por $F_k(n,m)$ a una instancia aleatoria del problema k-SAT formada al seleccionar uniformemente m cláusulas en $A_{k,n}$ con reemplazo. Estudios teóricos y experimentales muestran que las instancias aleatorias $F_k(n,m)$ con $r_k = m/n \approx 4.24$ son las más difíciles de resolver [48, 22]. A las instancias con razón $r_k \approx 4.24$ se les llaman instancias difíciles k-SAT y al valor r_k se le conoce como una transición de fase. Las instancias difíciles son de importancia ya que caracterizan la complejidad del problema k-SAT, y son usadas comúnmente como conjunto de prueba para evaluar el desempeño de algoritmos dedicados.

Dada una k-FC $\Phi = \bigwedge_{i=1}^{m} C_i$ con m cláusulas sobre un conjunto de n variables booleanas. Para cualquier cláusula $C_i = x_1^{\delta_1} \vee \cdots \vee x_k^{\delta_k}$, se define a π_{C_i} como el conjunto de índices de las variables booleanas que pertenecen a C_i. Se representa a Φ como una matriz $M_\Phi = (m_{ij}) \in \{-1, 0, 1\}^{m\times n}$ tal que para cualquier cláusula C_i,

$$\forall j \in \{1, \ldots, n\} : m_{ij} = \begin{cases} 2\delta_j - 1 & \text{si } j \in \pi_{C_i}, \\ 0 & \text{en otro caso} \end{cases} \tag{3.19}$$

donde δ_j corresponde al exponente de la variale x_j en C_i. De este modo, las instancias

aleatorias $F_k(n, m)$ coinciden con matrices $M_\Phi \in \{-1, 0, 1\}^{m\times n}$ donde por cada renglón M_i para $i \in \{1, \ldots, m\}$, M_i contiene k posiciones diferentes de cero. El algoritmo 2 muestra el procedimiento para la generación de instancias aleatorias y su implementación se llevó a cabo en Matlab.

Algorithm 2: Generación de instancias aleatorias.

Input : k número de literales por cláusula
m número de cláusulas
n número de variables.
Output: instancia aleatoria de k-SAT M_Φ.
Inicializar con ceros la matriz M_Φ de tamaño $m \times n$.
$i = 1$;
while $i \leq m$ **do**
  Escoger índices $j_1, \ldots, j_k \in \{1, \ldots, n\}$ uniformemente y distintos entre sí;
  for $r = 1, \ldots, k$ **do**
    Escoger $\delta \in \{-1, 1\}$ uniformemente;
    $M_\Phi(i, j_r) = \delta$;
  if *el renglón* $M_\Phi(i)$ *no existe en* M_Φ **then**
    $i = i + 1$;
return M_Φ;

Cada una de las herramientas consideradas para las simulaciones, ahmaxsat, qbsolv y dwave-sapi, requieren de un formato de entrada específico. El algoritmo ahmaxsat requiere el formato Dimacs para instancias de k-SAT; los archivos de entrada con formato Dimacs, en el primer renglón del archivo requiere el número de variables y el número de cláusulas con el prefijo `p cnf`. En los renglones siguientes del archivo se especifican cada una de las cláusulas donde por cada cláusula se listan los índices de sus variables seguido de un 0 que indica el fin de renglón.

El formato de entrada para qbsolv consiste en instancias del modelo QUBO donde en el primer renglón del archivo se tiene la cadena `p qubo` seguido de un 0 que significa que se trata de una instancia sin restricciones; el número de variables, la cantidad de términos lineales y la cantidad de términos cuadráticos. En los siguientes renglones se especifican primero los términos lineales seguidos de los términos cuadráticos. Un término lineal cx_j o cuadrático cx_ix_j se escriben como `j j c` y `i j c`, respectivamente.

Por último, el formato para dwave-sapi también son instancias del modelo QUBO similares al formato de qbsolv. Se usó la versión de Python para dwave-sapi por lo que una instancia QUBO se representa como una estructura de datos de diccionario. Un diccionario en Python es un conjunto de elementos, términos lineales y cuadráticos, con el siguiente formato: un término lineal cx_j o cuadrático cx_ix_j se escriben como `(j, j): c` y `(i, j): c`, respectivamente.

La Figura 3.2 muestra un ejemplo de una instancia aleatoria para 3-SAT en los formatos para ahmaxsat, qbsolv y dwave-sapi. En la Figura 3.2(izquierda) se muestra el formato Dimacs, en este formato un número negativo significa la negación de su variable correspondiente. En las Figuras 3.2(centro) y (derecha), se muestra la representación en el modelo QUBO de la instancia 3-SAT de la Figura 3.2(izquierda), propuesta en la sección 3.2.3, para qbsolv y dwave-sapi, respectivamente.

```
p cnf 4 17
1 2 3 0
1 3 4 0
2 -3 -4 0
-1 -3 -4 0
-2 -3 -4 0
1 3 -4 0
2 -3 4 0
-1 -2 3 0
1 -2 4 0
-1 -2 4 0
-1 -2 -4 0
-2 3 -4 0
-1 -3 4 0
1 2 4 0
2 3 4 0
-2 3 4 0
1 -3 -4 0
```

```
p qubo 0 6 6 11
c Diagonal elements
0 0 -3
1 1 -1
2 2 -2
3 3 -2
4 4 4
5 5 2
c off-diagonals
0 1 3
0 2 3
0 3 1
2 3 2
1 3 1
0 4 -2
1 4 -2
2 4 -2
0 5 -1
2 5 -1
3 5 -1
```

```
QUBO = {
    (0, 0): -3,
    (1, 1): -1,
    (2, 2): -2,
    (0, 1): 3,
    (0, 2): 3,
    (3, 3): -2,
    (0, 3): 1,
    (2, 3): 2,
    (1, 3): 1,
    (4, 4): 4,
    (0, 4): -2,
    (1, 4): -2,
    (2, 4): -2,
    (5, 5): 2,
    (0, 5): -1,
    (2, 5): -1,
    (3, 5): -1 };
```

Figura 3.2: Ejemplo de una instancia aleatoria de 4 variables y 17 cláusulas: (izquierda) formato Dimacs, (centro) formato qbsolv y (derecha) formato dwave-sapi.

Se construyeron tres conjuntos de instancias aleatorias de 3-SAT con razón r_k cerca de la transición de fase para llevar a cabo las simulaciones y evaluación del modelo propuesto. Los conjuntos son los siguientes:

- *Conjunto A*: Consta de 100 instancias con 50 variables y 212 cláusulas donde se

desconoce si existen asignaciones que las satisfacen.

- *Conjunto B*: Consta de 100 instancias con 8 variables y 34 cláusulas con una sola solución cada una.
- *Conjunto C*: Consta de 100 instancias con 9 variables y 38 cláusulas con una sola solución cada una.

El conjunto de instancias A se usaron para las herramientas ahmaxsat y qbsolv ya que pueden resolver instancias con un número grande de variables. Mientras que los conjuntos de instancias B y C se ususaron para la herramienta dwave-sapi ya que solo puede resolver instancias del modelo QUBO con un número pequeño de variables. Las razones r_3 para los conjuntos de instancias A, B y C son de 4.24, 4.25 y 4.22, respectivamente. Finalmente, por cada instancia 3-SAT en los conjuntos A, B y C, se construyeron sus correspondientes funciones pseudo-booleanas cuadráticas propuesta en la sección 3.2.3 para ser resueltas por qbsolv o dwave-sapi.

3.3.2. Análisis estadístico

La Figura 3.3 muestra una comparación entre ahmaxsat y qbsolv para resolver el problema max-SAT con respecto al número de cláusulas que se satisfacen por cada una de las instancias del conjunto A. Se puede observar que ahmaxsat encuentran las asignaciones de verdad que satisfacen todas las cláusulas para cada instancia. Mientras que qbsolv encuentra soluciones que oscilan entre un mínimo de 206 y máximo de 212 de cláusulas que se satisfacen entre las 100 instancias. Se puede ver que solo encuentra el óptimo de cláusulas que se satisfacen para 7 instancias. La media del número de cláusulas que se satisfacen es de 209.63 y la desviación estándar es de 1.35. Cabe mencionar que para todas las ejecuciones de qbsolv se estableció un máximo de 1000 iteraciones.

La Figura 3.4(arriba) muestra las energías ordenadas de menor a mayor del modelo QUBO por cada instancia en el conjunto A encontradas por qbsolv, y en la Figura 3.4(abajo) se muestran sus frecuencias. Las primeras 7 energías más bajas en la Figura 3.4(arriba)

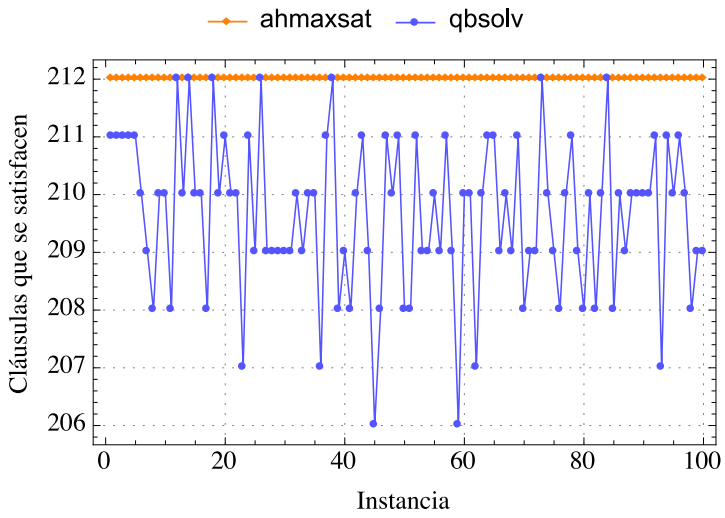


Figura 3.3: Número de cláusulas que se satisfacen por cada instancia del conjunto A usando ahmaxsat y qbsolv.

corresponden a las instancias 3-SAT para las cuales se encontró el valor óptimo. Note que los niveles de energía no corresponden al número de cláusulas que satisfacen, ya que el formato qbsolv no permite representar los términos constantes y el modelo QUBO propuesto en la sección 3.2.3 no permite contar el número de cláusulas que se satisfacen. Por lo anterior, las asignaciones encontradas por qbsolv son evaluadas directamente en las instancias 3-SAT correspondientes para poder contar el número de cláusulas que satisfacen.

La Figura 3.5(arriba) muestra una comparación del tiempo de ejecución para ahmaxsat y qbsolv con las instancias del conjunto A. La media de los tiempos de ejecución en segundos para ahmaxsat y qbsolv son de 0.024 y 4.48, respectivamente. El algoritmo ahmaxsat es casi tres órdenes de magnitud más rápido que qbsolv sobre las instancias del conjunto A. La desviación estándar de los tiempos de ejecución en segundos para ahmaxsat y qbsolv son de 0.00489 y 1.423, respectivamente. La Figura 3.5 (abaajo) muestra con más detalle los tiempos de ejecución para ahmaxsat sobre el conjunto A.

La herramienta dwave-sapi se puede ejecutarse de manera remota/local para resolver instancias del modelo QUBO/Ising. Cuando se ejecuta de manera remota, se realiza una conexión con el servidor y se encola un proceso en espera de ejecución en el hardware

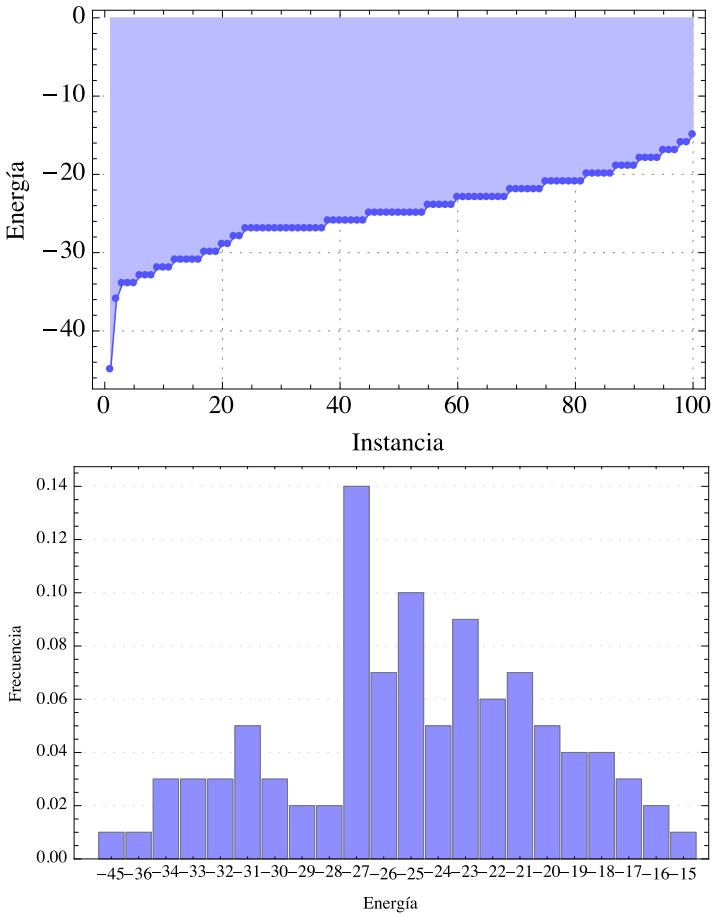


Figura 3.4: Energías Ising encontradas con qbsolv en las instancias del conjunto A (izquierda) ordenadas de menor a mayor y (derecha) la frecuencia de cada una de ellas.

cuántico. Por otro lado, cuando se usa de manera local, se ejecuta en el cliente un simulador que resuelve instancias del modelo QUBO/Ising usando el algoritmo de TCS. Ya sea de manera remota/local se tienen que establecer los parámetros como: el número de lecturas rep o veces que se va a ejecutar el proceso cuántico y el tiempo de TC (*annealing time* t_a). Los pasos a seguir para resolver una instancia QUBO/Ising usando dwave-sapi son los siguientes:

1. Construir el archivo de entrada para una instancia del modelo QUBO/Ising.

2. Establecer el número de lecturas rep y el tiempo t_a.

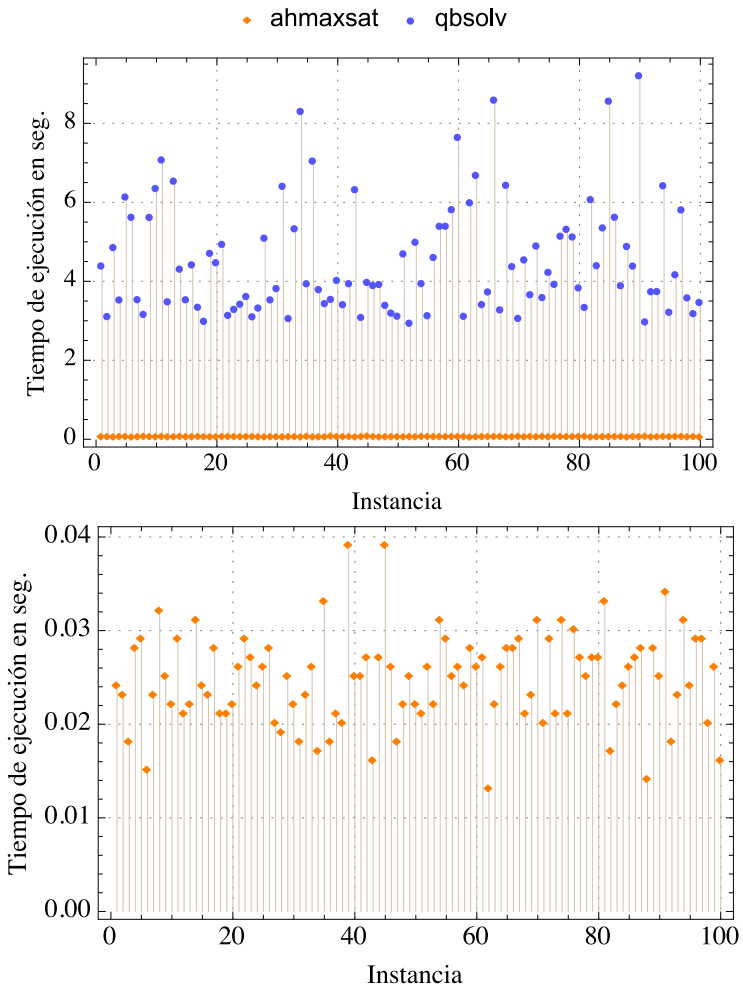


Figura 3.5: Comparación del tiempo de ejecución para ahmaxsat y qbsolv: (arribla) tiempo de ejecución de ahmaxsat y qbsolv para las instancias en el conjunto A y (abajo) tiempo de ejecución para ahmaxsat sobre el conjunto de instancias A.

3. Realizar un mapeo de la instancia de entrada sobre la topología del hardware.

4. Ejecutar el proceso de TC

El resultado de la ejecución del proceso de TC ya sea de manera remota/local consiste en un conjunto de rep configuraciones para las variables booleanas/Ising del problema junto con sus energías correspondientes. Las configuraciones son los valores obtenidos por medio de una medición cuántica al final del proceso de TC. Las ejecuciones realizadas en

este trabajo fueron de manera local, con un número $rep = 1000$ y t_a igual a 1.

La Figura 3.6(arriba) muestran la cantidad de qubits necesarios para el proceso de mapeo para cada instancia del modelo QUBO de los conjuntos B y C sobre arquitectura del hardware cuántico. El proceso de mapeo se lleva a cabo por medio de un algoritmo heurístico cuya implementación está disponible en dwave-sapi. Por cada cantidad de qubits obtenidos por el algoritmo heurístico, se ordenaron de menor a mayor para poder apreciar la tendencia del número de recursos necesarios para representar las instancias sobre el hardware cuántico. La media de qubits necesarios para el conjunto B es de 94.42 qubits con desviación estándar de 11.05 y la media para el conjunto C es de 112.87 qubits con desviación estándar de 7.28.

La Figura 3.6 (abajo) muestra la frecuencia de aparición de la solución óptima para las instancias de los conjuntos B y C, ordenadas de menor a mayor. La media de las frecuencias de aparición de la solución para los conjuntos B y C son 0.23 y 0.15, respectivamente. Note que entre mayor cantidad de variables, se requiere mayor cantidad de qubits y la probabilidad de encontrar la solución disminuye.

Finalmente, la Figura 3.7(arriba) muestran las energías QUBO promedio, ordenadas de menor a mayor, encontradas en los conjuntos B y C. Las cuatro energías más bajas con su número de ocurrencias total se muestran en el histograma de la Figura 3.7(abajo). Aunque se trate de diferentes conjuntos de instancias y herramientas, qbsolv y dwave-sapi, el histograma de la Figura 3.4(abajo) y el de la Figura 3.7 (abajo), en ambos casos las energías bajas tienen la menor frecuencia.

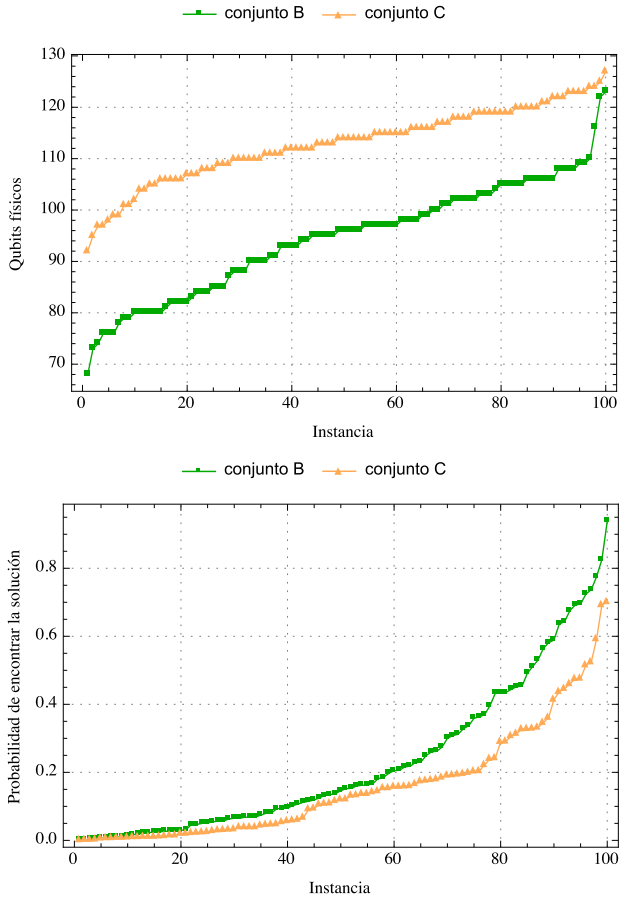


Figura 3.6: Resultados de la simulación usando dwave-sapi: (arriba) número de qubits físicos necesarios para mapear el problema en la arquitectura del hardware por cada una de las instancias de los conjuntos B y C ordenados de menor a mayor y (abajo) probabilidad de encontrar la solución por cada una de las instancias de los conjuntos B y C ordenadas de menor a mayor.

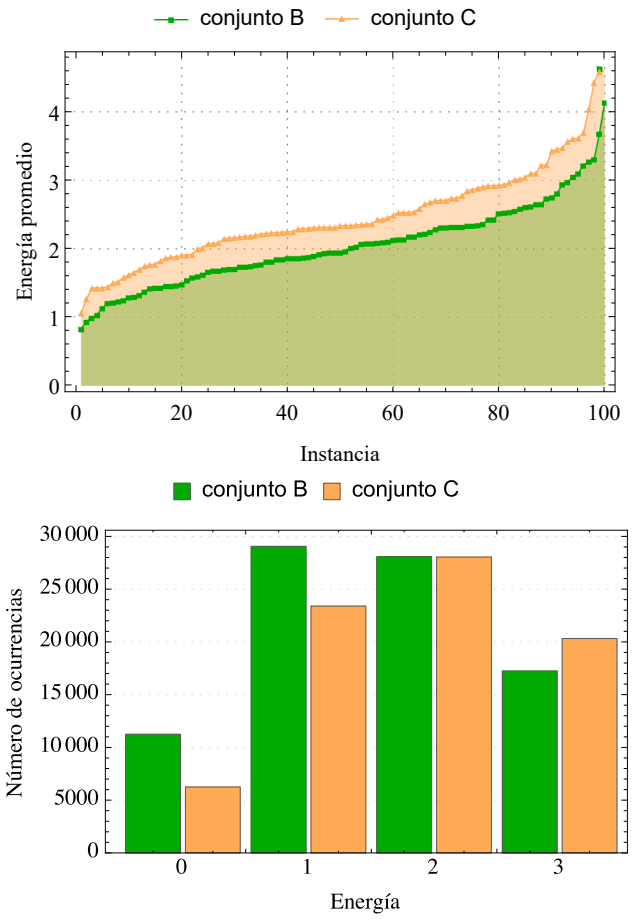


Figura 3.7: Resultados de la simulación usando dwave-sapi: (izquierda) energías promedio encontradas en los conjuntos de instancias B y C ordenadas de menor a mayor y (derecha) el número de ocurrencias para las cuatro energías más bajas encontradas en los conjuntos B y C.

Capítulo 4

Estrategias de mapeo

En este capítulo se aborda el problema de mapeo o *embedding* de instancias QUBO/Ising sobre la topología del hardware cuántico desarrollado por D-Wave. Se proponen tres estrategias directas de embedding para el problema max-SAT que toman ventaja de la estructura matemática del problema. Nuestras propuestas se enfocan en dos aspectos del proceso de embedding: 1) calcular el embedding de manera eficiente y directa por medio de algoritmos deterministas y 2) optimizar la cantidad de recursos físicos (qubits) para representar un problema dado. Los resultados de simulaciones muestran que las estrategias propuestas son competitivas en comparación con algoritmos de embedding heurísticos reportados en el estado del arte.

4.1. Topología

La tecnología cuántica actual D-Wave está construida sobre una topología específica de qubits físicos que interactúan de manera local [26, 27, 32]. Por lo que, limita la clase de problemas QUBO/Ising que se pueden representar directamente al hardware cuántico. La topología física del hardware se le llama *gráfica Chimera* $\mathcal{G}_{M,N,L}$, la cual consiste en una retícula de dimensión $M \times N$ bloques donde cada bloque tiene $2L$ vértices o qubits, que en total tiene $2LMN$ vértices. La Figura 4.1 muestra una gráfica Chimera $\mathcal{G}_{3,3,4}$ y la forma

en la que se conectan los vértices. Note que cada bloque en la gráfica Chimera corresponde a una gráfica L-bipartita.

Los vértices en una gráfica Chimera $\mathcal{G}_{M,N,L}$ se pueden indexar por medio de una 4-tupla (i, j, o, p) donde $0 \leq i \leq M-1$, $0 \leq j \leq N-1$, $o = 0, 1$, y $0 \leq p \leq L-1$. La pareja (i, j) indica el renglón y columna de un bloque en $\mathcal{G}_{M,N,L}$, el índice o corresponde a los vértices izquierdos (0) o derechos (1) en un bloque, y el índice p indica el número de vértice ya sea izquierdo o derecho en un bloque. Por convención, los bloques de una gráfica Chimera se enumeran en columnas de izquierda a derecha, y los renglones de abajo hacia arriba. De este modo, el bloque en la esquina inferior izquierda de la Figura 4.1 corresponde al bloque $(0, 0, o, p)$ donde $o = 0(1)$ son los vértices a la izquierda (derecha), y el índice p se asigna de abajo hacia arriba. Cualquier 4-tupla (i, j, o, p) se puede convertir en un solo índice por medio de la función $\rho(i, j, o, p) = 1 + 2NLi + 2Lj + Lo + p$.

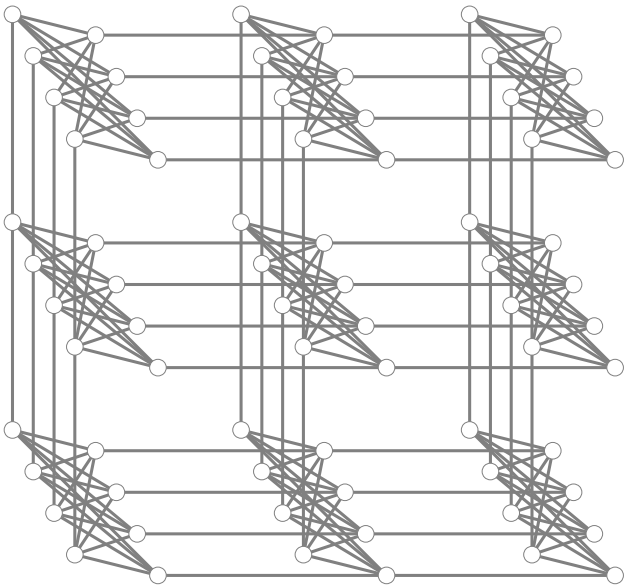


Figura 4.1: Topología de la gráfica Chimera $\mathcal{G}_{3,3,4}$.

4.1.1. El problema de embedding

Dada una instancia del modelo clásico QUBO/Ising como en (2.33) o (2.32), se le asocia una gráfica ponderada $G = (V, A)$ donde $V = \{1, \ldots, n\}$ y $E = \{\{i, j\} \mid J_{ij} \neq 0\}$, para cada vértice $i \in V : i \mapsto h_i$ y para toda arista $\{i, j\} \in E : \{i, j\} \mapsto J_{ij}$ (vea el capítulo 2), a G se le conoce como *gráfica lógica* del problema QUBO/Ising. Comúnmente, no siempre es posible encontrar una correspondencia uno-a-uno entre variables lógicas y qubits físicos en la gráfica Chimera, es decir, no siempre G es una subgráfica de $\mathcal{G}_{M,N,L}$.

El problema de embedding consiste en encontrar una subgráfica G' en $\mathcal{G}_{M,N,L}$ tal que G se puede obtener a partir de G' por medio de *contracciones de aristas*. La contracción de aristas toma una arista $\{i, j\}$, la cual es removida de la gráfica y los vértices incidentes i, j son fusionados en un nuevo vértice k, de modo tal que las aristas incidentes a k son las aristas incidentes de i y j. Si G' existe, entonces se dice que G es un *menor* de $\mathcal{G}_{M,N,L}$.

En [34] se propone una estrategia para encontrar una subgráfica $G' \subseteq \mathcal{G}_{M,N,L}$ para G de forma tal que cada vértice en G corresponde a un subárbol en G', y cada arista en G corresponde al menos a una arista en G'. A este problema se le llama embedding menor y se define de manera formal como sigue:

Definición 10. *Sea $\mathcal{G}_{M,N,L} = (V_{\mathcal{G}}, E_{\mathcal{G}})$ la gráfica Chimera y sea $G = (V, E)$ una instancia QUBO/Ising. El embedding menor de G se define como una función*

$$\phi : G \to \mathcal{G}_{M,N,L}$$

tal que

1. *Cada vértice $i \in V$ es mapeado a un subárbol conectado $T_i = (V_{T_i}, E_{T_i})$ en $\mathcal{G}_{M,N,L}$.*
2. *Existe una función $\tau : V \times V \to E_{\mathcal{G}_{M,N,L}}$ tal que para cada $\{i, j\} \in E$, existen vértices $i_{\tau(i,j)} \in V_{T_i}$ y $j_{\tau(j,i)} \in V_{T_j}$ con $\{i_{\tau(i,j)}, j_{\tau(j,i)}\} \in E_{\mathcal{G}_{M,N,L}}$.*

Encontrar un embedding menor con el mínimo número de vértices y aristas es un problema NP-difícil [8, 9]. Algunos autores han propuesto algoritmos heurísticos de com-

plejidad en tiempo polinomial para encontrar un embedding menor cuyo número de recursos (qubits) es cercano al embedding menor más pequeño [51, 6]. En [7], se propone una estrategia directa de embedding para el problema max SAT sobre una gráfica Chimera. Sin embargo, el número de qubits físicos que obtienen no es el óptimo tomando en cuenta la estructura matemática del problema, y por otro lado, no se muestra un estudio analítico o experimental de la efectividad del embedding propuesto para obtener soluciones del problema.

4.2. Estrategias de embedding propuestas

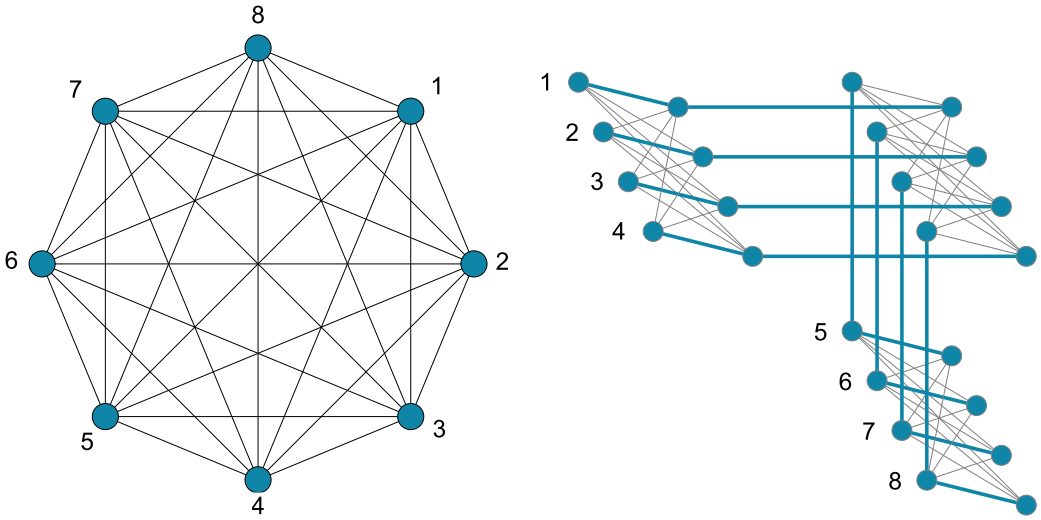
A continuación, se detallan las tres estrategias directas de embedding diseñadas para el problema max-SAT que se basan en [7]. Se realizó una comparación de nuestras propuestas de embedding y la propuesta en [7] en función de la cantidad de recursos que emplean y su efectividad para obtener soluciones del problema max-SAT.

4.2.1. Estrategia por bloques

La estrategia de embedding por bloques consiste en usar una sub-reticula triangular superior de la gráfica Chimera para mapear las variables originales de una instancia QUBO/Ising. Mientras que las variables nuevas del problema son mapeadas por medio de cadenas de qubits verticales y horizontales. Aquí, se asume que las instancias QUBO/Ising de las instancias del problema max-SAT están dadas como en la expresión (3.2) con su respectiva reducción de grado dadas en (3.10) y (3.11).

Consideremos un ejemplo de la estrategia de embedding por bloques para el mapeo de la gráfica completa K_8. La Figura 4.2(a) muestra la gráfica completa K_8 y la Figura 4.2(b) muestra el embedding encontrado usando la estrategia por bloques. En la Figura 4.2(b), cada cadena de qubits de color cian corresponde a una variable lógica o vértice de la Figura 4.2(a). Mientras que cada arista en la gráfica K_8 de la Figura 4.2(a) se mapea a una sola arista en la gráfica Chimera de la Figura 4.2(b). Note que por cada bloque de

la gráfica Chimera, solo es posible conectar entre sí a lo más 4 vértices. Por esta razón, se tienen que extender las cadenas de qubits hasta el bloque superior derecho, para conectar los vértices de cada bloque.



(a) (b)

Figura 4.2: (a) gráfica completa K_8 y (b) embedding de K_8.

La Figura 4.3 muestra el embedding obtenido usando la estrategia por bloques para la gráfica lógica de la Figura 3.1 con función QUBO correspondiente dada en (3.14). En este caso se tienen dos tipos de cadenas, las cadenas de qubits de Chimera y mapear que representan variables lógicas originales del problema, y las cadenas de color magenta que representan variables nuevas introducidas en la reducción. Cada cadena en la Figura 4.3 está enumerada con el vértice correspondiente de la Figura 3.1. Note que los vértices 1,2,3,4 y 5,6,7 son mapeados usando los bloques inferiores, y sus cadenas correspondientes se extienden vertical y horizontalmente para poder conectarse con las variables nuevas. Las variables nuevas son mapeadas a cadenas de longitud 1, ya que estas se conectan con exactamente 3 variables originales.

El Algoritmo 3 muestra la estrategia de embedding por bloques. Recibe como entrada una instancia max-SAT Φ sobre n variables con t cláusulas y su función QUBO correspondiente H_Φ^{qubo}. La función H_Φ^{qubo} está representada por $n+t$ variables lógicas, de

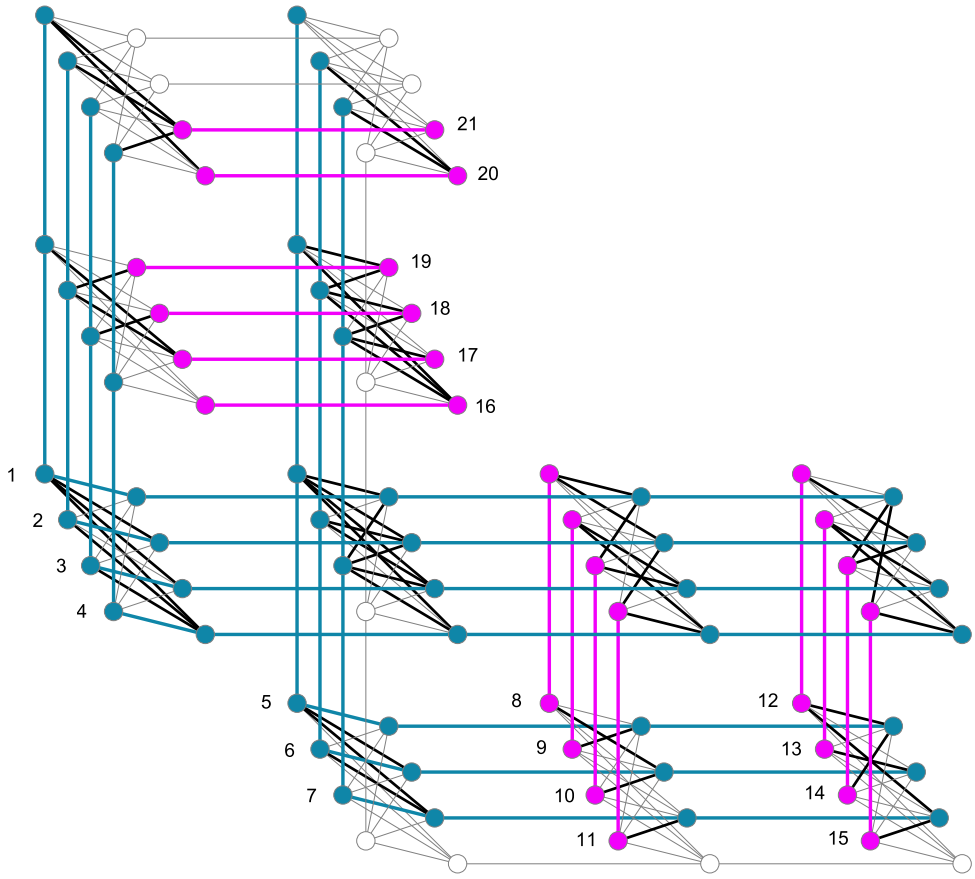


Figura 4.3: Embedding de la gráfica lógica de la Figura 3.1 con función QUBO dada en (3.14). Las aristas de color gris y los vértices representados como círculos vacíos no son parte del embedding.

las cuales n son variables originales del problema y las t restantes son variables auxiliares que se agregaron en el proceso de reducción de grado. Para calcular un embedding de una función H_Φ^{qubo} con $n+t$ variables, la estrategia por bloques requiere de una gráfica Chimera $\mathcal{G}_{M,N,L}$ donde $M = \lceil n/4 \rceil + \lceil t/8 \rceil$, $N = \lceil n/4 \rceil + \lfloor t/8 \rfloor$ and $L = 4$. El número de qubits del embedding que usa la estrategia por bloques es $2LNM - base(base-1)/2$ donde $base = \lceil n/4 \rceil$. La longitud máxima de las cadenas de qubits es de $2(N-1)$ asumiendo $M = N$.

El algoritmo de embedding por bloques es similar a la propuesta en [7], en donde

Algorithm 3: Estrategia de embedding por bloques

Input : Una gráfica $G = (V, E)$, n variables o vértices, anc número de variables nuevas y gráfica Chimera $\mathcal{G}_{M,N,L}$.
Output: Un embedding menor de G sobre $\mathcal{G}_{M,N,L}$.

$V' = \emptyset$, $E' = \emptyset$;
$base = \lceil n/4 \rceil$;
$blocks = \lceil anc/4 \rceil$;
$blockh = base + \lceil blocks/2 \rceil$;
$blockv = base + \lfloor blocks/2 \rfloor$;
`//Mapea cadenas de variables originales`
$cont = 0$, $g = 0$, $h = base - 1$;
for $a = 0$ *to* $base - 1$ **do**
  for $p = 0$ *to* $L - 1$ **do**
    if $cont < n$ **then**
      $cont++$;
      for $b = g$ *to* $blockh - 1$ **do**
        $E' = E' \cup \{\rho(h, b, 1, p), \rho(h, b + 1, 1, p)\}$;
      for $b = h$ *to* $blockv - 1$ **do**
        $E' = E' \cup \{\rho(b, g, 0, p), \rho(b + 1, g, 0, p)\}$;
  $g++$, $h--$;
`//Mapea horizontalmente variables nuevas`
$cont = 0$, $j = base$;
for $a = base + 1$ *to* $blockh$ **do**
  for $p = 0$ *to* $L - 1$ **do**
    if $cont < anc$ **then**
      for $i = base - 1$ *to* 0 **do**
        $E' = E' \cup \{\rho(i, j, 0, p), \rho(i - 1, j, 0, p)\}$;
      $cont++$;
  $j++$;
`//Mapea verticalmente variables nuevas`
$i = base$;
for $a = base + 1$ *to* $blockv$ **do**
  for $p = 0$ *to* $L - 1$ **do**
    if $cont < anc$ **then**
      for $j = 0$ *to* $base - 1$ **do**
        $E' = E' \cup \{\rho(i, j, 1, p), \rho(i, j - 1, 1, p)\}$;
      $cont++$;
  $i++$;
Obtener V' a partir de E';
Return $G' = (V', E')$;

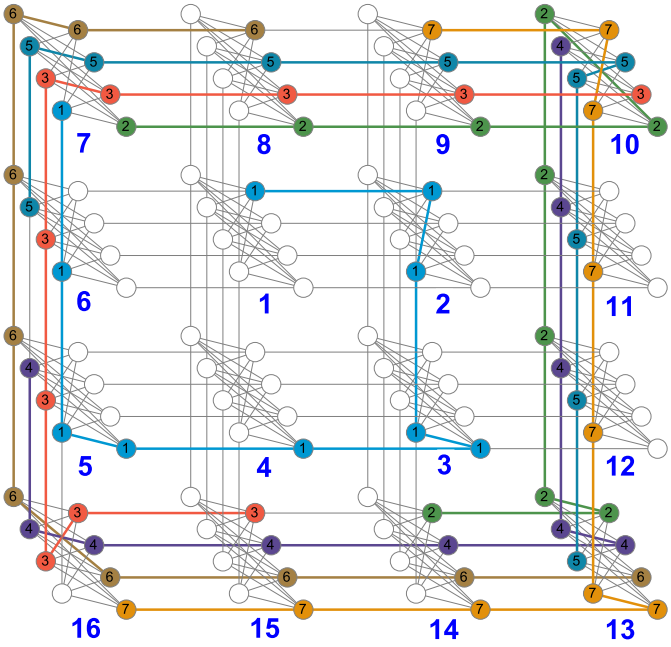
solo se usan cadenas verticales. Nuestra propuesta usa cadenas de qubits verticales y horizontales para aprovechar la estructura cuadrada de la gráfica Chimera. De esta forma, es posible mapear una cantidad mayor de qubits de una función QUBO dada.

4.2.2. Estrategia espiral

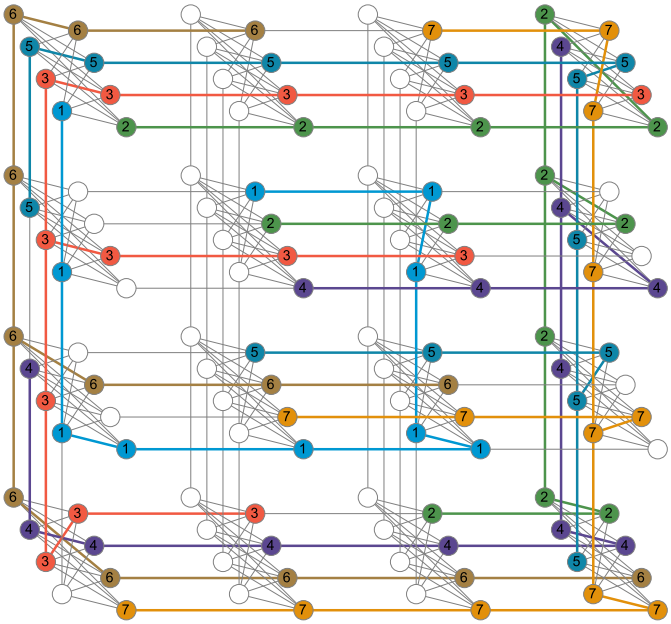
La estrategia de embedding en espiral se basa en construir cadenas de qubits para mapear variables lógicas en forma de espiral a lo largo de la gráfica Chimera. La forma en espiral permite distribuir mejor las variables o cadenas de qubits, desde el centro hacia afuera de la Chimera. De este modo, es más fácil conectar variables adyacentes debido a la cercanía de las cadenas en la forma en espiral. Asumiendo que se tienen instancias del problema max-SAT como la expresión dada en (3.2) con su respectiva reducción de grado dadas en (3.10) y (3.11). La estrategia en espiral consiste en tres pasos: (1) mapear las variables originales del problema en forma de espiral desde del centro hacia afuera de la Chimera, (2) extender vertical u horizontalmente las cadenas de forma tal que alcancen a sus variables lógicas adyacentes, y (3) mapaear las variables nuevas usando los qubits más cercanos a sus variables lógicas adyacentes.

Por ejemplo, los pasos de la estrategia en espiral para la Figura 3.1 con función QUBO dada en (3.14) son los siguientes: el paso (1) se muestra en la Figura 4.4(a) donde las cadenas de qubits en forma de espiral desde adentro hacia afuera de la Chimera. Para ello, se enumeran los bloques de la Chimera en forma de espiral y se crean las cadenas desde el primer bloque interior. En este paso solo se mapean las variables originales del problema por medio de cadenas de longitud 8 y 9. En el paso (2), que se muestra en la Figura 4.4(b), se visualiza la extensión de las cadenas vertical u horizontal hacia los bloques más cercanos donde pasan las variables originales adyacentes del problema. Finalmente, el paso (3) se muestra en la Figura 4.5 donde cada arista de la gráfica lógica es mapeada a una sola arista del Chimera, y cada variable nueva se mapea a un solo qubit.

El Algoritmo 4 muestra la estrategia de embedding en espiral. En el Algoritmo 4 se usa la función `indexing_blocks` (Algoritmo 8 del apéndice) para asignar un índice a



(a)



(b)

Figura 4.4: Pasos (1) y (2) y de la estrategia de embedding en espiral para la gráfica de la Figura 3.1.

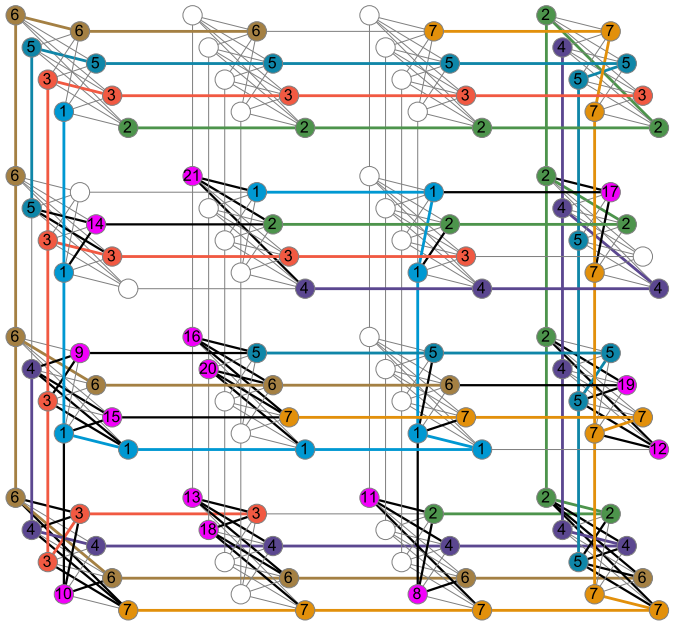


Figura 4.5: Resultado de embedding de la gráfica en la Figura 3.1 usando la estrategia en espiral. Las aristas de color gris y los vértices representados como círculos vacíos no son parte del embedding.

cada bloque del Chimera en forma de espiral, la función `spiral_chain` (Algoritmo 6 del apéndice) que mapea cada variable original del problema en una cadena de qubits y la extiende a los bloques más cercanos donde pasan las variables lógicas adyacentes. Finalmente, la función `partitioning_chain` (Algoritmo 7 del apéndice) que crea las cadenas de qubits en forma de espiral.

4.2.3. Estrategia diagonal

La estrategia de embedding diagonal usa los bloques en la diagonal de la gráfica Chimera para mapear las variables originales de una instancia 3-SAT. Aquí se asume que se tienen instancias como en (3.2) con su respectiva reducción de grado dadas en (3.10) y (3.11). Esta consiste de cuatro pasos importantes: (1) seleccionar los bloques diagonales de la gráfica Chimera y mapear a lo más dos variables originales del problema por cada bloque; (2)

Algorithm 4: Estrategia de embedding espiral.

Input : Una gráfica $G = (V, E)$, n variables o vértices y gráfica Chimera $\mathcal{G}_{M,N,L}$.
Output: Un embedding menor de G sobre $\mathcal{G}_{M,N,L}$.

$V' = \emptyset$, $E' = \emptyset$;
$total\_squares = 2$;
$spiral = \emptyset$, $blocks = \emptyset$;
$max = 2, min = max - 1$;
for *cada* $square$ *de* $total\_squares$ **do**
    //Índices de los bloques
    $iblock =$indexing_blocks(min, max);
    //Crea cadenas en espiral
    $beg = 0$;
    if $square == 1$ **then**
        $beg = 3$;
    $spiral = spiral \cup$ spiral_chain$(iblock, min, max, L, beg)$;
    $min - -$; $max + +$;
//Particiona la espiral en n cadenas
$chainSize = \lfloor |spiral|/n \rfloor$;
$\{i, j\} = blocks[1]$;
$E' =$partitioning_chain$(chainSize, spiral, n, i, j, E, E')$;
//Asigna las variables nuevas a los vértices vacíos
Obtain V' from E';
//Vértices vacíos de la Chimera
$V''' = V'' - V'$;
for *cada variable nueva de* E **do**
    Sea $\{x, y, z\}$ los vertices adyacentes a la variable nueva
    for *cada vértice* v *de* V''' **do**
        if $\{v, x\} \cap E''$ *y* $\{v, y\} \cap E''$ *y* $\{v, z\} \cap E''$ **then**
            $E' = E' \cup \{v, x\} \cup \{v, y\} \cup \{v, z\}$;
            $V''' = V''' - \{v\}$;
Obtener V' a partir de E';
Return $G' = (V', E')$;

expandir tanto vertical como horizontalmente las variables iniciales a lo largo de la gráfica Chimera; (3) expandir ya sea vertical u horizontalmente los qubits, de forma tal que se garantice la conexión con las variables nuevas; y (4) mapear las variables nuevas usando los qubits más cercanos a sus variables adyacentes.

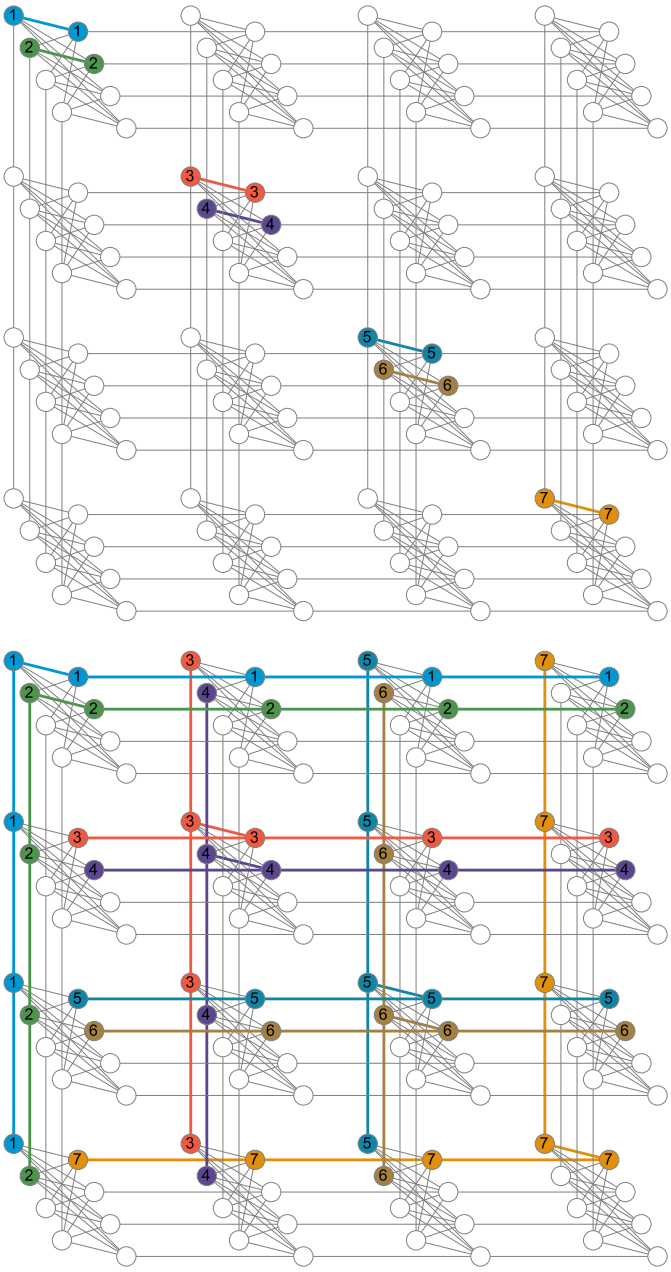


Figura 4.6: Pasos (1) y (2) de la estrategia de embedding diagonal.

Por ejemplo, los pasos de la estrategia diagonal para la Figura 3.1 con función QUBO dada en (3.14) son los siguientes: la Figura 4.6(arriba) muestra el paso (1), se puede ver que se mapean a lo más dos variables originales del problema sobre los bloques diagonales. La Figura 4.6(abajo) se muestra el paso (2), extendiendo las cadenas vertical y horizontalmente. La Figura 4.7(arriba) muestra el paso (3), donde cada cadena por variable original del problema se extiende a un qubit en el mismo bloque (observe las flechas). Finalmente, la Figura 4.7(abajo) muestra el embedding final, mapeando las variables nuevas a los qubits más cercanos con sus variables adyacentes.

El Algoritmo 5 muestra la estrategia de embedding diagonal. Este algoritmo se puede generalizar para instancias arbitrarias del problema max-SAT. En general, la estrategia diagonal crea cadenas de longitud a lo más $2N$ qubits para cualquier gráfica Chimera $\mathcal{G}_{N,M,L}$, suponiendo $N = M$. Finalmente, esta estrategia crea cadenas de qubits más grandes que la estrategia por bloques.

4.3. Comparación y desempeño

En la Figura 4.8 se muestra una comparación del tamaño del embedding las estrategias propuestos en esta tesis y la técnica heurística implementada en la libreria D-Wave. Se utilizó una gráfica Chimera de tamaño 4×4 y se generó un conjunto de 100 instancias aleatorias del problema max-3-SAT con una proporción de 30 cláusulas y 7 variables. Esta proporción del número de variables sobre cláusulas se eligió con respecto al tamaño máximo de instancia que se puede mapear sobre la Chimera usando el algoritmo en espiral. Las estrategias por bloques, espiral y diagonal, para un número fijo de variables por cláusulas, obtienen el mismo tamaño de embedding. El método heurístico de D-Wave siempre regresa un embedding de tamaño diferente de qubits con una desviación estándar de 4.78 qubits.

La Figura 4.9(a) muestra la frecuencia promedio para encontrar la solución en más de 100 instancias aleatorias de max-3-SAT para las estrategias de embedding propuestas en esta tesis y el método heurístico. Todas las instancias aleatorias de 3-SAT utilizadas se

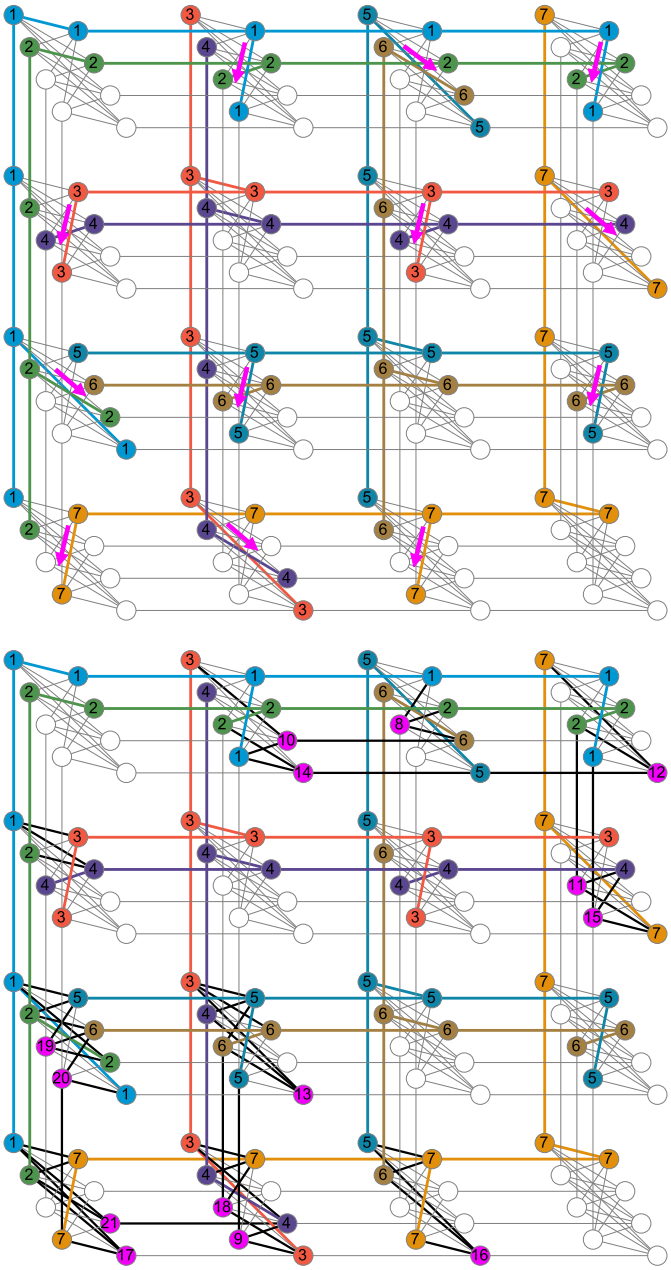


Figura 4.7: Pasos (3) y (4) de la estrategia de embedding diagonal. Las aristas de color gris y los vértices representados como círculos vacíos no son parte del embedding.

Algorithm 5: Estrategia de embedding diagonal.

Input : Una gráfica $G = (V, E)$, n variables o vértices y gráfica Chimera $\mathcal{G}_{M,N,L}$.
Output: Un embedding menor de G sobre $\mathcal{G}_{M,N,L}$.

$V' = \emptyset$, $E' = \emptyset$; $i = \lceil n/2 \rceil - 1, max = i, var = 0$;
`//Mapea variables originales`
for $a = 1$ *to* n **do**
  for $b = 0$ *to* max **do**
    $E' = E' \cup \{\rho(i, b, 1, 3 - var), \rho(b, max - i, 0, var)\}$;
    `//qubits adicionales`
    if $\text{mód}\ (i, 2) > 0$ *y* $\text{mód}\ (b, 2) > 0$ **then**
      if $i \neq max - b$ **then**
        $E' = E' \cup \{\rho(i, b, 1, 3 - var), \rho(b, max - i, 0, 3 - var)\}$;
      if $b \neq i$ **then**
        $E' = E' \cup \{\rho(b, max - i, 0, var), \rho(b, max - i, 1, var)\}$;
    else if $\text{mód}\ (i, 2) == 0$ *and* $\text{mód}\ (b, 2) == 0$ **then**
      if $i \neq max - b$ **then**
        $E' = E' \cup \{\rho(i, b, 1, 3 - var), \rho(b, max - i, 0, 3 - var)\}$;
      if $b \neq i$ **then**
        $E' = E' \cup \{\rho(b, max - i, 0, var), \rho(b, max - i, 1, var)\}$;
  if $var == 1$ **then**
    $i - -$; $var = 0$;
  else
    $var + +$;
`//Asigna variables nuevas los vértices vacíos`
Obtener V' a partir de E'; $V''' = V'' - V'$;
for *cada variable nueva de* E **do**
  Sean $\{x, y, z\}$ los vértices adyacentes de la variable nueva
  for *cada vértice* v *de* V''' **do**
    if $\{v, x\} \cap E''$ *y* $\{v, y\} \cap E''$ *y* $\{v, z\} \cap E''$ **then**
      $E' = E' \cup \{v, x\} \cup \{v, y\} \cup \{v, z\}$;
      $V''' = V''' \setminus \{v\}$;
Obtener V' a partir de E';
Return $G' = (V', E')$;

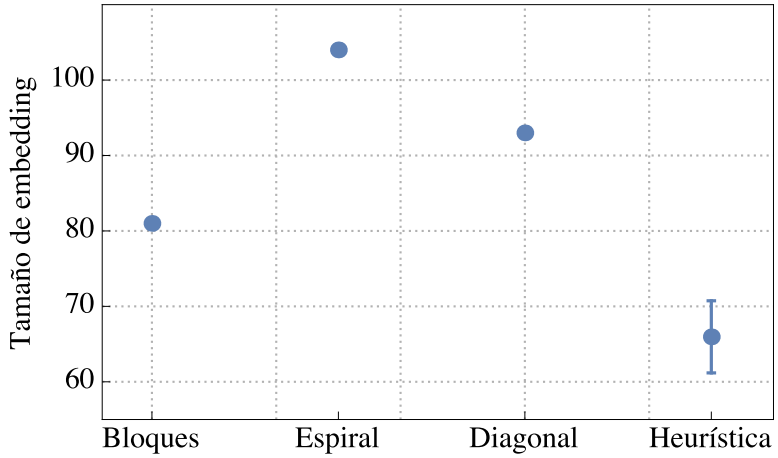
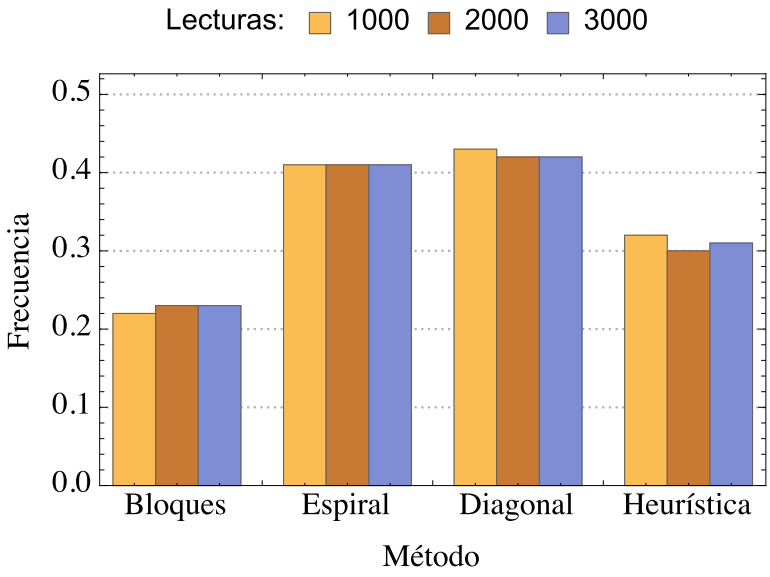


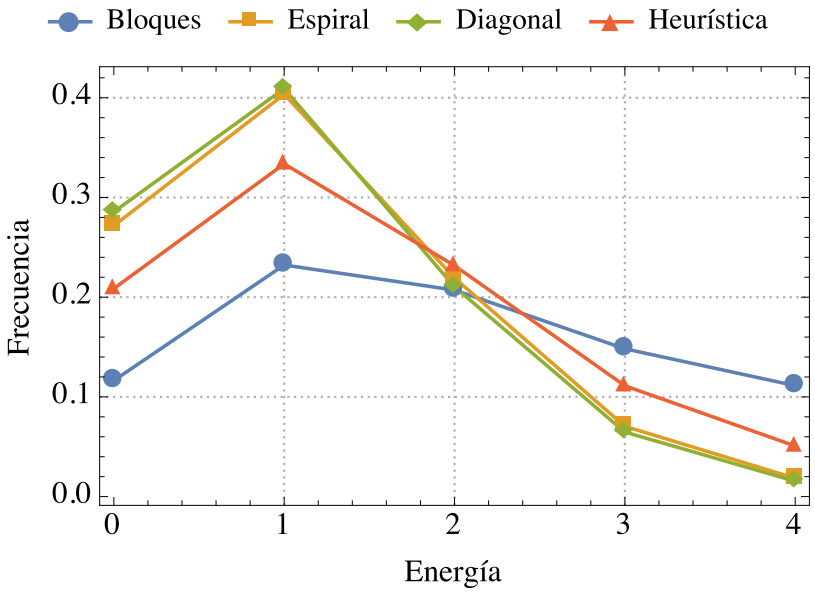
Figura 4.8: Tamaño en qubits del embedding obtenido por las estrategias por bloques, espiral, diagonal y heurístico.

generaron de tal manera que tienen una única solución. Usamos la propuesta de [9] para asignar los pesos a las cadenas de qubits de los embedding obtenidas para las estrategias por bloques, espiral y diagonal.

La Figura 4.9(a) muestra que la estrategia por bloques obtiene la peor frecuencia para encontrar asignaciones de verdad; mientras que las estrategias espiral y diagonal tienen los mejores resultados. Además, se puede ver que las frecuencias obtenidas no cambian en promedio para un número de lecturas de 1000, 2000 y 3000. Por otro lado, el método heurístico obtiene un menor número de qubits en el embedding pero no tiene la mejor frecuencia para encontrar asignaciones de verdad. La Figura 4.9(b) muestra la frecuencia promedio para las primeras energías más bajas que corresponden al número más grande de cláusulas satisfechas, para un número de lecturas igual a 1000. Como puede verse en la Figura 4.9(b), la estrategia de embedding por bloques obtiene la peor frecuencia de energía más baja de acuerdo con la Figura 4.9(a).



(a)



(b)

Figura 4.9: Comparación: (a) frecuencia promedio para encontrar soluciones y (b) frecuencia promedio de las energías más pequeñas para encontrar soluciones de las estrategias de embedding por bloques, espiral, diagonas y heurística.

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo de investigación, se describió la formulación clásica y cuántica del problema max-SAT y 1-en-3 sat. Se usaron los métodos propuestos por Ishikawa y Freedman para la reducción de grado de las funciones QUBO para el problema max-SAT. Esta reducción de una expresión pseudo-booleana de grado arbitrario a una expresión QUBO comúnmente implica agregar nuevas variables al problema.

Con base en la expresión QUBO obtenida, se diseñaron tres estrategias de embedding a la arquitectura Chimera: algoritmos por bloques, espiral y diagonal. También se realizaron simulaciones clásicas usando algoritmos para encontrar soluciones del problema max-SAT, la herramienta qbsolv y el algoritmo de temple cuántico simulado. Los experimentos realizados fueron los siguientes:

- Aproximación de soluciones al problema max-SAT usando algoritmos clásicos en el estado del arte.
- Búsqueda de soluciones a instancias aleatorias del max-SAT con su representación QUBO usando la herramienta qbsolv.
- Simulación del temple cuántico usando el algoritmo de temple cuántico simulado para resolver instancias aleatorias sobre una gráfica Chimera $\mathcal{G}_{4,4,4}$.
- Cálculo del embedding para instancias aleatorias del problema max-SAT sobre una

gráfica Chimera $\mathcal{G}_{4,4,4}$.

Se observó que la estrategia de embedding por bloques obtuvo los peores resultados, en comparación con las estrategias espiral y diagonal. El algoritmo heurístico disponible en las librerías de D-Wave, fue la que obtuvo el menor tamaño de embeddig en nuestras comparaciones. Sin embargo, se observó en los resultados que el embedding que consume más qubits físicos (diagonal) obtiene una probabilidad más alta de éxito que el resto de las estrategias.

Por otro lado, las reducciones presentadas son efectivas para obtener funciones QUBO con un número polinomial de nuevas variables. Hasta el momento, en la mayoría de los problemas de optimización, no se ha podido evadir la reducción de las funciones pseudo-booleanas con el fin de mapear directamente a la topología Chimera.

Se espera que este trabajo sea el punto de partida para otras investigaciones en el área de cómputo cuántico adiabático y temple cuántico, en la búsqueda de soluciones a problemas de optimización combinatorios. Como trabajo futuro de esta tesis se considera lo siguiente:

- Estudiar la importancia de las variables auxiliares introducidas en la reducción de grado, en el proceso adiabático y cómo afecta en la exploración del espacio de soluciones y energía.
- Desarrollar estrategias de embedding que sean tolerantes a errores debido a la longitud de las cadenas de qubits. Se propone usar qubits redundantes para hacer más robusto el embedding obtenido.
- Realizar un estudio de todas las funciones booleanas cuadráticas con un número constante de variables, usadas para contar soluciones (cláusulas) en problemas combinatorios. Lo anterior con el fin de caracterizar el tipo de funciones QUBO o problemas que se puede representar usando el modelo Ising.

Apéndice A

Algoritmos complementarios de estrategias de embedding

A.1. Listado de los algoritmos

Se muestran los algoritmos complementarios de las estrategias de embedding descritos en el capítulo 4 de esta tesis. El orden del listado es el siguiente:

- Algoritmo 1: Spiral chain function.
- Algoritmo 2: Partitioning chain function.
- Algoritmo 3: Indexing blocks function.

Se usó la herramienta de software Mathematica versión 11 para la implementación y visualización de la gráfica Chimera y el embedding resultante. Se pone a disposición de manera pública los códigos en Mathematica de los algoritmos aquí propuestos en la siguiente URL: `https://goo.gl/qeanZc`

Algorithm 6: Spiral chain function

Function *spiral_chain(*$iblock, min, max, L, beg$*)*
- $spiral = \emptyset$;
- **for** $a = beg$ *to* $L - 1$ **do**
  - **for** *each* $block$ *of* $iblock$ **do**
    - $\{i, j\} = block$;
    - **if** $a == 0$ *and* $block == 1$ **then**
      - $spiral = spiral \cup \rho(i, j, 1, 0) \cup \rho(i, j, 0, 3)$;
    - **else if** $i == min$ **then**
      - **if** $j == min$ **then**
        - $spiral = spiral \cup \rho(i, j, 1, 3 - a)$;
        - **if** $a < L - 1$ **then**
          - $spiral = spiral \cup \rho(i, j, 0, 2 - a)$;
      - **else if** $j == max$ **then**
        - $spiral = spiral \cup \rho(i, j, 0, a) \cup \rho(i, j, 1, 3 - a)$;
      - **else**
        - $spiral = spiral \cup \rho(i, j, 1, 3 - a)$;
    - **else if** $i == max$ **then**
      - **if** $j == min$ **then**
        - **if** $square == 1$ **then**
          - $spiral = spiral \cup \rho(i, j, 1, a)$;
        - **else**
          - $spiral = spiral \cup \rho(i, j, 0, 3 - a) \cup \rho(i, j, 1, a)$;
      - **else if** $j == max$ **then**
        - $spiral = spiral \cup \rho(i, j, 1, 0 + a) \cup \rho(i, j, 0, a)$;
      - **else**
        - $spiral = spiral \cup \rho(i, j, 1, a)$;
    - **else**
      - **if** $j == min$ **then**
        - $spiral = spiral \cup \rho(i, j, 0, 3 - a)$;
      - **else if** $j == max$ **then**
        - $spiral = spiral \cup \rho(i, j, 0, a)$;
- Return $spiral$;

Algorithm 7: Partitioning chain function

Function *partitioning_chain(*$chainSize, spiral, n, i, j, E, E''$*)*
- $index = 0, p = 2;$
- $E' = \emptyset;$
- `//Spiral edges`
- **for** $a = 1$ *to* n **do**
  - $E'_a = \emptyset;$
  - **for** $b = 0$ *to* $chainSize - 2$ **do**
    - $E'_a = E'_a \cup \{spiral[index], spiral[index++]\};$
  - `//Adding the horizontal edges for inner square`
  - **if** $[(a > 1$ *and* $a \leq 3)$ *and* $(p == 2$ *or* $p == 0)]$ *or* $[(a > 3$ *and* $a < n)$ *and* $(p == 3$ *or* $p == 1)]$ **then**
    - $E'_a = E'_a \cup \{\rho(i, j, 1, p), \rho(i, j + 1, 1, p)\};$
    - $E'_a = E'_a \cup \{\rho(i, j + 1, 1, p), \rho(i, j + 2, 1, p)\};$
    - **if** $p == 0$ **then**
      - $p = 3;$
      - $i - -;$
    - **else**
      - $p - -;$
  - **else**
    - $E'_a = E'_a \cup \{\rho(i, j - 1, 1, p), \rho(i, j, 1, p)\};$
    - $E'_a = E'_a \cup \{\rho(i, j, 1, p), \rho(i, j + 1, 1, p)\};$
    - $p - -;$
  - $E' = E' \cup E'_a;$
- `//Relationing the` E `edges in the minor embedding`
- $V_1 = \emptyset, V_2 = \emptyset;$
- `//...for every vertice of edges of logic graph`
- **for** *each edge* $\{v_1, v_2\}$ *of* E **do**
  - Obtain V'_1 of E'_{v_1};
  - Obtain V'_2 of E'_{v_2};
  - $E' = E' \cup (E'' \cap \text{Tuples}\{\{V_1\}, \{V_2\}\});$
- Return E';

Algorithm 8: Indexing blocks function

Function *indexing_blocks(*min, max*)*
- $blocks = \emptyset, cont = 0$;
- **for** $a = min + 1$ *to* max **do**
  - $cont++$;
  - $blocks[cont] = \{a, max\}$;
  - $blocks[cont + (max - min)] = \{min, a\}$;
- $cont += max - min$;
- **for** $a = max - 1$ *to* min **do**
  - $cont++$;
  - $blocks[cont] = \{a, max\}$;
  - $blocks[cont + (max - min)] = \{min, a\}$;
- Return $blocks$;

Anexo A

Productos de investigación

Figura A.1: Participación en la X Reunión de la División de Información Cuántica de la Sociedad Mexicana de Física (dICu) 2017.

Figura A.2: Participación en el Taller de Óptica Cuántica 2017 en las instalaciones del Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE).

Embedding strategies for the maximum satisfiability problem

Martha Morales Huerta · William Cruz-Santos · Salvador E. Venegas Andraca · Marco Lanzagorta

Received: date / Accepted: date

Abstract Harnessing quantum hardware resources is of vital importance to assess their actual capabilities when compared with state of the art digital computers. In particular, the D-Wave technology allows to solve combinatorial optimization problems that can be mapped into a specific hardware topology that implements a two-local Ising model. The process by which an instance of the Ising problem is mapped into the quantum hardware is called minor embedding, which maps logical variables to physical qubits. Finding custom embedding methods allows to take advantage of the mathematical structure of specific problems. In this article, three direct embedding strategies are proposed for the max k-SAT problem. Simulations results show us that, using our embedding strategies, the occurrence to find solutions of hard random instances of the max k-SAT problem can be increased when compared with heuristic embedding algorithms.

Keywords Quantum annealing · satisfiability · combinatorial optimization

Martha Morales Huerta
CU-UAEM Valle de Chalco, Hermenegildo Galeana 3, Valle de Chalco 56615, Estado de México, México
E-mail: martha_morales@live.com.mx

William Cruz-Santos
CU-UAEM Valle de Chalco, Hermenegildo Galeana 3, Valle de Chalco 56615, Estado de México, México
E-mail: wdelacruzd@uaemex.mx

Salvador E. Venegas Andraca
Tecnologico de Monterrey, Escuela de Ingenieria y Ciencias. Ave. Eugenio Garza Sada 2501, Monterrey, NL 64849, Mexico
E-mail: salvador.venegas-andraca@keble.oxon.org and svenegas@itesm.mx

Marco Lanzagorta
US Naval Research Laboratory, 4555 Overlook Ave. SW, Washington DC, 20375, USA.
E-mail: marco.lanzagorta@nrl.navy.mil

Figura A.3: Portada de un artículo próximo a publicarse acerca de las estrategias de embedding desarrolladas en este trabajo de investigación.

Referencias

[1] A. Abrame and D. Habet. *Ahmaxsat: Description and Evaluation of a Branch and Bound Max-SAT Solver*. Journal of Satisfiability, Boolean Modeling and Computation, 9:89–128, 2015.

[2] S. Arora and B. Barak. *Computacional complexity: a modern approach*. Cambridge University Press, 2009.

[3] P. Asirelli, M. De Santis, and M. Martelli. *Integrity constraints in logic databases*. The Journal of Logic Programming, 2(3):221–232, 1985.

[4] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation: combinatorial optimization problems and their approximability properties*. Springer-Verlag, Berlin, Heidelberg, 1999.

[5] F. Barahona. *On the computational complexity of Ising spin glass models*. Journal of Physics A: Mathematical and General, 15(10):3241, 1982.

[6] Z. Bian, F. Chudak, R. B. Israel, B. Lackey, W. G. Macready, and A. Roy. *Mapping Constrained Optimization Problems to Quantum Annealing with Application to Fault Diagnosis*. Frontiers in ICT, 3:14, 2016.

[7] N. Chancellor, S. Zohren, P. Warburton, S. Benjamin, and S. Roberts. *A direct mapping of max k-SAT and high order parity checks to a Chimera graph*. Scientific reports, 6:37107, 2016.

[8] V. Choi. *Minor-embedding in adiabatic quantum computation: I. The parameter setting problem*. Quantum Information Processing, 7(5):193–209, 2008.

[9] V. Choi. *Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design*. Quantum Information Processing, 10(3):343–353, 2011.

[10] S. A. Cook. *The complexity of theorem-proving procedures*. In Proceedings of the third annual ACM symposium on Theory of computing, pages 151–158. ACM, 1971.

[11] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2009.

[12] E. Crosson and A. Harrow. *Simulated Quantum Annealing Can Be Exponentially Faster Than Classical Simulated Annealing*. In 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), pages 714–723, 2016.

[13] E. Dahl. *Programming with D-wave: Map coloring problem*. D-Wave Official Whitepaper, 2013.

[14] M. Davis and H. Putnam. *A computing procedure for quantification theory*. Journal of the ACM (JACM), 7(3):201–215, 1960.

[15] J. Edmonds. *Paths, trees, and flowers*. Canadian Journal of mathematics, 17(3):449–467, 1965.

[16] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda. *A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem*. Science, 292(5516):472–475, 2001.

[17] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. *Quantum Computation by Adiabatic Evolution*. eprint arXiv:quant-ph/0001106, 2000.

[18] R. Feynman. *Simulating physics with computers*. International journal of theoretical physics, 21(6):467–488, 1982.

[19] A. B. Finnila, M. A. Gomez, C. Sebenik, C. Stenson, and J. D. Doll. *Quantum annealing: A new method for minimizing multidimensional functions*. Chemical Physics Letters, 219(5):343 – 348, 1994.

[20] D. Freedman and P. Drineas. *Energy Minimization via Graph Cuts: Settling What is Possible*. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA, pages 939–946, 2005.

[21] M. R. Garey and S. Johnson, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[22] I. P. Gent and T. Walsh. *The SAT phase transition*. In ECAI, volume 94, pages 105–109. PITMAN, 1994.

[23] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, New York, NY, USA, 2008.

[24] L. K. Grover. *A fast quantum mechanical algorithm for database search*. In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pages 212–219. ACM, 1996.

[25] J. Gu and R. Puri. *Asynchronous circuit synthesis with boolean satisfiability*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 14(8):961–973, 1995.

[26] R. Harris, J. Johansson, A. J. Berkley, M. W. Johnson, T. Lanting, Siyuan Han, P. Bunyk, E. Ladizinsky, T. Oh, I. Perminov, E. Tolkacheva, S. Uchaikin, E. M. Chapple, C. Enderud, C. Rich, M. Thom, J. Wang, B. Wilson, and G. Rose. *Experimental demonstration of a robust and scalable flux qubit*. Phys. Rev. B, 81:134510, 2010.

[27] R. Harris, M. W. Johnson, T. Lanting, A. J. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, F. Cioata, I. Perminov, P. Spear, C. Enderud, C. Rich, S. Uchaikin, M. C. Thom, E. M. Chapple, J. Wang, B. Wilson, M. H. S.

Amin, N. Dickson, K. Karimi, B. Macready, C. J. S. Truncik, and G. Rose. *Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor*. Phys. Rev. B, 82:024511, 2010.

[28] J. Hartmanis and R. E. Stearns. *On the computacional complexity of algorithms*. Transactionsof the American Mathematical Society, 117:285–306, 1965.

[29] L. Ingber. *Simulated annealing: Practice versus theory*. *Mathematical and computer modelling*, 18(11):29–57, 1993.

[30] H. Ishikawa. *Transformation of General Binary MRF Minimization to the First-Order Case*. IEEE Trans. Pattern Anal. Mach. Intell., 33(6):1234–1249, 2011.

[31] R. G. Jeroslow and J. Wang. *Solving propositional satisfiability problems*. Annals of mathematics and Artificial Intelligence, 1(1-4):167–187, 1990.

[32] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J S Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. *Quantum annealing with manufactured spins*. Nature, 473(7346):194–198, 2011.

[33] T. Kadowaki and H. Nishimori. *Quantum annealing in the transverse Ising model*. Phys. Rev. E, 58:5355–5363, 1998.

[34] W. M. Kaminsky and S. Lloyd. *Scalable architecture for adiabatic quantum computing of NP-hard problems*. In Quantum computing and quantum bits in mesoscopic systems, pages 229–236. Springer, 2004.

[35] R. M. Karp. *Reducibility among combinatorial problems*. In Complexity of computer computations, pages 85–103. Springer, 1972.

[36] J. King, S. Yarkoni, M. M. Nevisi, J. P. Hilton, and C. C. McGeoch. *Benchmarking a quantum annealing processor with the time-to-target metric*. ArXiv e-prints, 2015.

[37] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. *Optimization by Simulated Annealing*. Science, 220(4598):671–680, 1983.

[38] V. Lakshmikantham and S. K. Sen. *Computational Error and Complexity in Science and Engineering: Computational Error and Complexity*. Mathematics in Science and Engineering. Elsevier Science, 2005.

[39] L. A. Levin. *Universal sequential search problems*. Problemy Peredachi Informatsii, 9(3):115–116, 1973.

[40] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall PTR, 1997.

[41] A. Messiah. *Quantum Mechanics*. Dover books on physics. Dover Publications, 1999.

[42] T. A Nguyen, W. A. Perkins, T. J. Laffey, and D. Pecora. *Checking an Expert Systems Knowledge Base for Consistency and Completeness*. In IJCAI, volume 85, pages 375–378, 1985.

[43] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 2011.

[44] I. G. Rosenberg. *Reduction of bivalent maximization to the quadratic case*. Cahiers du Centre d'etudes de Recherche Operationnelle, 17:71–74, 1975.

[45] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[46] D. Sanjoy, P. Christos, and V. Umesh. *Algorithms*. McGraw-Hill Higher Education, 2007.

[47] G. E. Santoro and E. Tosatti. *Optimization using quantum mechanics: quantum annealing through adiabatic evolution*. Journal of Physics A: Mathematical and General, 39(36):R393–R431, 2006.

[48] B. Selman, H. Levesque, and D. Mitchell. *A New Method for Solving Hard Satisfiability Problems*. In Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI'92, pages 440–446. AAAI Press, 1992.

[49] P. W. Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. SIAM J. Comput., 26(5):1484–1509, 1997.

[50] A. M. Turing. *On computable numbers, with an application to the Entscheidungsproblem*. Proceedings of the London mathematical society, 2(1):230–265, 1936.

[51] D. Venturelli, D. J. Marchand, and G. Rojo. *Quantum Annealing Implementation of Job-Shop Scheduling*. ArXiv e-prints, 2015.