

UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Tesi di Laurea Triennale in Ingegneria Informatica

**Quantum Annealing: Fondamenti e
Applicazioni a Problemi di Ottimizzazione
di Interesse Ingegneristico**

Relatore:

Prof. Marco Cococcioni

Candidato:

Francesco Panattoni

A mio padre Claudio, mia madre Rita e mia sorella Chiara. Per il loro costante supporto, che mi ha permesso di raggiungere questo traguardo.

A mia nonna Gabriela, che con il suo amore e i suoi insegnamenti continua a guidarmi anche da lontano.

Indice

1	Introduzione	3
2	Postulati della Meccanica Quantistica	6
2.1	Introduzione alla Meccanica Quantistica	6
2.2	Primo Postulato della Meccanica Quantistica	6
2.3	Secondo Postulato della Meccanica Quantistica	7
2.4	Terzo Postulato della Meccanica Quantistica	8
2.5	Quarto Postulato della Meccanica Quantistica	9
3	Quantum Bit	10
3.1	Dal Bit al Qubit	10
3.2	Proprietà del Qubit	11
3.2.1	Sovrapposizione	11
3.2.2	Entanglement	12
3.2.3	Interferenza	13
3.2.4	Effetto Tunnel	13
3.3	Sfera di Bloch	14
3.4	Vantaggi computazionali del Qubit	15
4	Quantum Annealing	17
4.1	Introduzione alla Computazione Quantistica	17
4.2	Problemi di Ottimizzazione	19
4.3	Modello di Ising	20
4.4	Quadratic Unconstrained Binary Optimization	22
4.5	Teorema Adiabatico e Processo Adiabatico	25
4.6	Campo Trasversale e Matrici di Pauli	29
4.7	Effetto Tunnel nel Quantum Annealing	31
4.8	Un confronto con il Simulated Annealing	32
5	Topologie Hardware e Architettura del Quantum Annealing	37
5.1	Quantum Annealer	37
5.2	Distinzione fra Qubit Logico e Qubit Fisico	37
5.3	Grafo Chimera	38
5.4	Embedding	39

5.5	Tool per l'Embedding	40
6	Quantum Annealing per il Problema del Commesso Viaggiatore	41
6.1	Problema del Commesso Viaggiatore	41
6.2	TSP nella forma QUBO	42
6.3	Progettazione Algoritmo	45
6.4	Esempio pratico di risoluzione di un TSP	47
6.4.1	Passo 1: Definizione del Problema	47
6.4.2	Passo 2: Rappresentazione della Soluzione	48
6.4.3	Passo 3: Definizione dei Vincoli	48
6.4.4	Passo 4: Costruzione della matrice QUBO	49
6.4.5	Passo 5: Conversione nel Modello di Ising	50
6.4.6	Passo 6: Embedding con grafico Chimera	50
6.4.7	Passo 7: Determinazione del Minimo di Energia	52
6.4.8	Passo 8: Interpretazione della soluzione	52
7	Implementazione dell'Algoritmo di Grover attraverso il Quantum Annealing	54
7.1	Introduzione all'Algoritmo di Grover	54
7.2	Porta di Hadamard	56
7.3	Oracolo Quantistico	57
7.4	Iterazione di Grover	60
7.5	Applicazione dell'Algoritmo di Grover	64
7.6	Algoritmo di Grover con Quantum Annealing	67
7.7	Perché implementare l'Algoritmo di Grover con il Quantum Annealing	73
8	Conclusioni	74
A	Appendice A: Codice di risoluzione del TSP	75
B	Appendice B: Prova che QUBO e Ising sono Isomorfi	81

Capitolo 1: Introduzione

La **computazione quantistica** rappresenta una delle più affascinanti frontiere della scienza moderna, un crocevia dove fisica teorica, matematica e informatica si incontrano per sfidare i limiti del calcolo tradizionale. Sin dall'invenzione dei primi computer, avvenuta a metà del XX secolo, l'uomo ha cercato di costruire macchine sempre più veloci e potenti, in grado di risolvere problemi complessi in tempi ridotti. Tali macchine, dai primi calcolatori meccanici fino ai moderni supercomputer, hanno però sempre operato entro i vincoli imposti dai principi della fisica classica. È nel regno quantistico che si dischiudono nuove possibilità: fenomeni come l'entanglement, la sovrapposizione e il tunneling promettono di ridefinire radicalmente il concetto stesso di elaborazione dell'informazione.

Il percorso che ha portato alla nascita della computazione quantistica affonda le radici nel lavoro pionieristico di figure come *Richard Feynman* e *David Deutsch*. Negli anni '80, Feynman intuì che un computer basato su principi quantistici avrebbe potuto simulare sistemi fisici quantistici in modo più efficiente rispetto ai computer classici. Successivamente, Deutsch formalizzò questa intuizione introducendo il concetto di "computer quantistico universale", una macchina teorica in grado di eseguire qualsiasi algoritmo classico e, potenzialmente, di risolvere alcuni problemi in modo esponenzialmente più veloce.

Alla base di questa rivoluzione vi è il **quantum bit** (o in forma abbreviata **qubit**) unità fondamentale dell'informazione quantistica, il cui comportamento trascende la dicotomia binaria del bit. Il bit, nel paradigma classico, può esistere esclusivamente nello stato 1 o nello stato 0. Il qubit, invece, può esistere simultaneamente in più stati grazie al principio di sovrapposizione, aprendo la strada a forme di parallelismo computazionale prima impensabili. L'entanglement, una correlazione profonda e non locale tra qubit, permette di intrecciare informazioni in modi che sfidano l'intuizione umana e che trovano applicazioni tanto nella crittografia quanto nell'ottimizzazione.

Questa disciplina, un tempo confinata al regno della teoria, ha visto negli ultimi decenni un'accelerazione straordinaria. Dalla dimostrazione degli algoritmi di *Shor* e *Grover* negli anni '90, che hanno messo in evidenza il potenziale dei computer quantistici nel fattorizzare numeri primi e cercare elementi in un database non ordinato, fino ai progressi odierni nella realizzazione di computer quantistici scalabili, il campo della computazione quantistica non solo ridefinisce le capacità computazionali, ma sfida il confine tra il mondo classico e quello quantistico.

I computer quantistici possono essere programmati utilizzando linguaggi di alto livello come **Python**, grazie a framework avanzati che traducono il codice in istruzioni comprensibili dall'hardware quantistico. Strumenti come **Qiskit**, **Cirq** e **Amazon Braket** consentono ai programmati di descrivere algoritmi quantistici, progettare circuiti e interagire con dispositivi quantistici o simulatori. Sebbene Python non venga eseguito direttamente sull'hardware quantistico, funge da interfaccia per inviare comandi e analizzare i risultati. L'architettura sottostante traduce il codice in linguaggi a basso livello, come **QASM**, eseguibili dai processori quantistici. Questa metodologia permette di sfruttare la potenza della computazione quantistica mantenendo la semplicità e la leggibilità tipiche dei linguaggi di programmazione classici.

Due sono i principali paradigmi che si sono affermati per sfruttare le peculiarità della meccanica quantistica: il **Gate-Based Quantum Computing** e l'**Adiabatic Quantum Computing**. Sebbene entrambi condividano la capacità di elaborare l'informazione sfruttando fenomeni come la sovrapposizione e l'entanglement, i loro approcci concettuali e implementativi sono profondamente differenti, riflettendo due visioni distinte di come un sistema quantistico possa essere manipolato per risolvere problemi computazionali.

L'approccio più studiato, e quello che attualmente appare come il futuro predominante della computazione quantistica, è il *Gate-Based Quantum Computing*. Questo modello gode di una maggiore flessibilità teorica e applicativa, oltre a essere sostenuto da un ampio consenso nella comunità scientifica e da investimenti significativi da parte delle principali aziende tecnologiche. Tuttavia questo approccio non è esente da problematiche. Il principale ostacolo è rappresentato dalla **decoerenza quantistica**, il fenomeno per cui l'interazione di un sistema quantistico con l'ambiente esterno porta alla perdita di informazioni quantistiche. Poiché i qubit sono estremamente sensibili a perturbazioni esterne (come fluttuazioni termiche o elettromagnetiche), il loro stato quantistico tende a collassare rapidamente in uno stato classico, interrompendo i calcoli e riducendo la fedeltà dei risultati. Il fenomeno diventa critico nelle operazioni con porte quantistiche, che necessitano di coerenza temporale significativamente maggiore rispetto ai tempi di calcolo.

Il principale argomento di cui tratteremo in questa tesi sarà il *Quantum Annealing*, una forma specializzata dell'*Adiabatic Quantum Computing* progettata specificamente per risolvere problemi di ottimizzazione. A differenza dell'*Adiabatic Quantum Computing*, che si propone come paradigma computazionale universale [1], il *Quantum Annealing* è ottimizzato per individuare il minimo globale di una funzione di costo definita in uno spazio di soluzioni discreto.

Nasce come estensione del **Simulated Annealing**, una tecnica classica di ottimizzazione introdotta nel 1983 da *Kirkpatrick, Gelatt e Vecchi*, come un metodo per risolvere problemi di ottimizzazione combinatoria utilizzando analogie con i processi fisici di ricottura dei metalli.

Il *Quantum Annealing* fu formalizzato come concetto nel 1998 grazie al lavoro di *T. Kadowaki e H. Nishimori*, che dimostrarono la possibilità di utilizzare il **Teorema Adiabatico** per trovare il minimo energetico di un sistema [2].

Negli anni 2000, si passò dalla teoria alla pratica con lo sviluppo di prototipi hardware per implementare il modello. In questo periodo, aziende come *D-Wave Systems*, fondata nel 1999, iniziarono a costruire i primi dispositivi quantistici basati sul *Quantum Annealing*.

Nel 2007, D-Wave presentò pubblicamente il primo computer quantistico commerciale, il *D-Wave One*, in grado di risolvere problemi di ottimizzazione combinatoria utilizzando il *Quantum Annealing*.

Il *Quantum Annealing* si configura, quindi, come un approccio molto promettente, vista l'efficacia nella risoluzione dei problemi di ottimizzazione. Tuttavia, essendo una tecnica specializzata, esso resta meno versatile rispetto al *Gate-Based Quantum Computing*, almeno per il momento.

Le sfide tecnologiche ed epistemologiche che accompagnano la realizzazione dei computer quantistici rendono questo campo un simbolo del nostro tempo: un'era in cui la scienza e la tecnologia non solo spiegano il mondo, ma creano strumenti per plasmarlo. La computazione quantistica invita a ripensare profondamente le nostre idee su ciò che è possibile calcolare, aprendo nuovi orizzonti per affrontare problemi che la computazione classica non può risolvere in tempi ragionevoli.

Capitolo 2: Postulati della Meccanica Quantistica

2.1 Introduzione alla Meccanica Quantistica

La **meccanica quantistica** è una delle teorie fondamentali per la descrizione del comportamento della materia e dell'energia a scale microscopiche. Contrariamente alla meccanica classica, che si basa sull'evoluzione deterministica di variabili osservabili (come posizione e velocità), la meccanica quantistica introduce una descrizione probabilistica dei sistemi fisici, basata su una struttura matematica fortemente astratta.

I principi fondamentali della meccanica quantistica sono formalizzati attraverso quattro postulati, che stabiliscono le regole matematiche e fisiche per la rappresentazione degli stati quantistici, la loro evoluzione nel tempo e il processo di misurazione.

2.2 Primo Postulato della Meccanica Quantistica

Lo stato di un sistema quantistico isolato è completamente descritto da un vettore di stato (o funzione d'onda) in uno spazio di Hilbert associato al sistema.

Il primo postulato [3] stabilisce che ogni sistema quantistico può essere rappresentato matematicamente da un vettore di stato $|\psi\rangle$, appartenente a uno spazio vettoriale complesso, detto **spazio di Hilbert**. Questo spazio, dotato di un prodotto scalare, permette di calcolare lunghezze e angoli tra i vettori, conferendo così una struttura geometrica ai concetti di probabilità e sovrapposizione quantistica.

Il vettore $|\psi\rangle$ è detto anche **ket** ed è espresso secondo la *notazione di Dirac*.

Un vettore di stato può essere scritto come una combinazione lineare di stati base ortonormali, detti autostati, secondo la relazione:

$$|\psi\rangle = \sum_{i=1}^N \lambda_i |e_i\rangle$$

dove $|e_i\rangle$ rappresenta una base ortonormale dello spazio di Hilbert e λ_i sono coefficienti complessi che esprimono le ampiezze di probabilità.

La normalizzazione di un ket richiede che la somma delle probabilità su tutti gli stati possibili sia pari a 1, il che si traduce nella condizione:

$$\int_{-\infty}^{\infty} ||\psi\rangle|^2 dx = 1$$

Questo requisito riflette il fatto che la probabilità totale di trovare la particella da qualche parte nello spazio deve essere uguale a 1. Il modulo quadro della funzione d'onda o del vettore di stato fornisce una misura della probabilità di trovare il sistema in un particolare stato.

Un esempio pratico riguarda il caso di un sistema a una dimensione, dove $|\psi\rangle$ rappresenta la funzione d'onda di una particella. La probabilità di trovare la particella in un intervallo specifico è proporzionale al valore di $||\psi\rangle|^2$.

2.3 Secondo Postulato della Meccanica Quantistica

L'evoluzione temporale di un sistema quantistico isolato è descritta da un'operazione unitaria governata dall'equazione di Schrödinger.

L'equazione di Schrödinger [4] può essere espressa come:

$$i\hbar \frac{d|\psi(t)\rangle}{dt} = H |\psi(t)\rangle$$

Dove:

- $|\psi(t)\rangle$ il vettore di stato del sistema quantistico in funzione del tempo t ;
- \hbar è la **costante di Planck ridotta**. Il valore esatto non è importante ed è comune assorbire il fattore \hbar in H ponendo $\hbar = 1$. Per completezza: $\hbar = 1.0545718 \times 10^{-34} \text{ J} \cdot \text{s}$;
- i è l'unità immaginaria;
- H è l'**Hamiltoniano**, un operatore hermitiano che rappresenta l'energia totale del sistema e agisce sullo spazio degli stati del sistema stesso, combinando l'energia cinetica e potenziale.

L’evoluzione temporale unitaria implica che la norma del vettore di stato sia preservata nel tempo, assicurando la conservazione della probabilità totale del sistema.

Se uno stato iniziale $|\psi(0)\rangle$ evolve fino a uno stato $|\psi(t)\rangle$ dopo un tempo t , la relazione tra i due stati è data da un operatore unitario $U(t)$, tale che:

$$|\psi(t)\rangle = U(t) |\psi(0)\rangle = e^{-\frac{i}{\hbar} H t} |\psi(0)\rangle$$

In conclusione, risulta ovvio che se si conosce l’Hamiltoniano di un sistema, insieme al valore della costante di Planck ridotta \hbar (che per l’appunto è una costante), è possibile determinare l’evoluzione temporale dello stato quantistico del sistema attraverso l’equazione di Schrödinger. Questa equazione, infatti, descrive come lo stato del sistema evolva nel tempo, fornendo un quadro teorico completo per prevedere la dinamica del sistema.

Tuttavia, la determinazione precisa dell’Hamiltoniano di un sistema fisico specifico risulta essere un compito estremamente complesso, in quanto richiede una conoscenza approfondita delle forze e delle interazioni che governano il comportamento delle sue componenti. Infatti gran parte della fisica del ventesimo secolo è stata incentrata sulla determinazione dell’Hamiltoniano di un sistema, o almeno sulla comprensione di come costruire e approssimare correttamente l’Hamiltoniano per descrivere vari sistemi fisici.

L’Hamiltoniano è oggetto di particolare attenzione in quanto rappresenta uno strumento fondamentale nell’analisi e nella formulazione del *Quantum Annealing*.

2.4 Terzo Postulato della Meccanica Quantistica

In un sistema quantistico isolato, ogni grandezza fisica misurabile è rappresentata da un operatore hermitiano \hat{O} che agisce nello spazio di Hilbert. Questo operatore, chiamato osservabile, ha autovettori che formano una base per lo spazio di Hilbert, che corrispondono a stati quantistici. I risultati della misurazione di un’osservabile sono dati dagli autovalori di questo operatore. Quando si effettua una misurazione, lo stato del sistema collassa in uno degli autovettori associati all’autovalore misurato.

Sebbene i sistemi quantistici isolati seguano un’evoluzione unitaria determinata dall’equazione di Schrödinger, è fondamentale prevedere momenti in cui uno strumento di **misura** (che rappresenta un’interazione con un sistema fisico esterno) interviene per osservare il comportamento del sistema. Questo processo interrompe l’isolamento del sistema, trasformandolo in un sistema aperto e introducendo un’interazione che altera la sua evoluzione, rendendola non più unitaria.

Il terzo postulato [5] formalizza questo concetto tramite l'**osservabile**, un operatore hermitiano \hat{O} i cui autovalori (reali, poiché hermitiano) rappresentano i possibili risultati della misura e i cui autovettori formano una base dello spazio di Hilbert. Questo implica che **il processo di misura è intrinsecamente probabilistico**.

La probabilità di ottenere un particolare autovalore λ_i dell'osservabile durante la misura è data da:

$$P(\lambda_i) = |\langle \phi_i | \psi \rangle|^2$$

dove $|\phi_i\rangle$ è l'autovettore associato a λ_i e $|\psi\rangle$ è lo stato quantistico del sistema prima della misurazione. Una volta effettuata la misura, lo stato del sistema collassa nello stato $|\phi_i\rangle$, perdendo la sovrapposizione originaria. Questo fenomeno rende la misurazione un atto non neutro, che modifica irreversibilmente lo stato del sistema.

2.5 Quarto Postulato della Meccanica Quantistica

Lo spazio degli stati di un sistema composto è il prodotto tensore degli spazi degli stati dei sottosistemi componenti.

Per descrivere sistemi quantistici composti, il quarto postulato [6] stabilisce che lo stato totale è ottenuto combinando gli stati dei sottosistemi attraverso il prodotto tensore:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$$

Il prodotto tensore tra due spazi vettoriali V e W è uno spazio vettoriale che contiene combinazioni lineari di elementi della forma $v \otimes w$, con $v \in V$ e $w \in W$. In pratica è un modo per unire spazi vettoriali per formare spazi vettoriali più grandi.

Ciò consente di rappresentare stati di sistemi composti da più sottosistemi. Tuttavia l'**entanglement** si verifica proprio quando lo stato del sistema totale non può essere separato in stati separabili dei singoli sottosistemi, cioè non può essere scritto come il prodotto tensore degli stati dei sottosistemi:

$$|\psi\rangle \neq |\psi_1\rangle \otimes |\psi_2\rangle \text{ se entangled}$$

Questo implica che le misurazioni su uno dei sottosistemi possono influenzare istantaneamente lo stato dell'altro sottosistema, anche se separato spazialmente.

L'entanglement è un fenomeno che è alla base di molte applicazioni della computazione quantistica, come la crittografia quantistica.

Capitolo 3: Quantum Bit

3.1 Dal Bit al Qubit

Nella computazione classica, il **bit** è l'unità fondamentale dell'informazione e può assumere esclusivamente uno dei due stati definiti: 0 o 1. In ambito quantistico, l'analogo del bit è il **quantum bit**, o **qubit**, un'entità che sfrutta le proprietà della meccanica quantistica per rappresentare e processare informazioni.

A differenza del bit classico, il qubit non è limitato a due stati distinti, ma può esistere in una sovrapposizione di entrambi gli stati base. Formalmente, lo stato di un qubit [7] è descritto come:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Dove:

- $|0\rangle$ e $|1\rangle$ rappresentano gli stati base del qubit rappresentati nella notazione di Dirac tramite i ket;
- α e β sono numeri complessi che rappresentano l'ampiezza di probabilità degli stati $|0\rangle$ e $|1\rangle$, rispettivamente.

Le ampiezze devono rispettare la condizione $|\alpha|^2 + |\beta|^2 = 1$, che garantisce che la somma delle probabilità sia sempre pari a 1.

La rappresentazione dello stato del qubit si inserisce nel contesto del **primo postulato della meccanica quantistica**, secondo cui un sistema quantistico è descritto da un vettore di stato appartenente a uno spazio di Hilbert complesso. Questo vettore contiene tutte le informazioni relative al qubit, inclusa la probabilità di osservare specifici stati base durante una misurazione.

La realizzazione fisica di un qubit richiede l'impiego di un sistema quantistico che possa essere manipolato per rappresentare, controllare e misurare stati quantistici. Una delle implementazioni fisiche più comuni si basa sullo spin elettronico, ovvero

il momento angolare intrinseco associato a un singolo elettrone. Gli elettroni sono intrappolati in strutture di semiconduttori, che sfruttano la tecnologia già esistente nel mondo dei chip classici. In questo contesto, il qubit è codificato nei due possibili orientamenti dello spin dell'elettrone.

Gli stati $|0\rangle$ e $|1\rangle$ sono rappresentati, rispettivamente, dagli stati di spin "up" (\uparrow) e spin "down" (\downarrow). Lo spin può essere manipolato tramite campi magnetici o impulsi elettrici. Sebbene siano possibili diverse implementazioni fisiche dei qubit, nel contesto della presente analisi sul *Quantum Annealing* si adotterà quella basata sullo spin elettronico.

3.2 Proprietà del Qubit

Il qubit possiede proprietà intrinseche che derivano direttamente dai principi fondamentali della meccanica quantistica e che meritano un'analisi più approfondita.

3.2.1 Sovrapposizione

La prima proprietà fondamentale del qubit, l'abbiamo già introdotta ed è la **sovraposizione** [8]. Questa proprietà consente a un singolo qubit di rappresentare un insieme più ampio di informazioni rispetto a un bit classico, grazie alla possibilità di trovarsi in una sovrapposizione degli stati base 0 e 1. Tuttavia ciò non implica che sia possibile accedere simultaneamente a entrambi gli stati base durante un'operazione di misurazione.

In base al **terzo postulato della meccanica quantistica**, la misurazione di un qubit proietta lo stato quantistico in uno dei suoi stati base, distruggendo la sovrapposizione preesistente. In particolare, il risultato della misurazione sarà 0 con probabilità $|\alpha|^2$, o 1 con probabilità $|\beta|^2$.

Un esempio classico di stato di sovrapposizione di un qubit, fondamentale nel calcolo quantistico, è il seguente:

$$|\psi\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

Questo stato è noto come stato di sovrapposizione equiprobabile ed è una combinazione lineare in cui il qubit ha una probabilità del 50% di collassare nello stato $|0\rangle$ e una probabilità del 50% di collassare nello stato $|1\rangle$ quando viene misurato nella base computazionale. Il fattore $\frac{1}{\sqrt{2}}$ è necessario per soddisfare la condizione di normalizzazione: $|\alpha|^2 + |\beta|^2 = 1$.

3.2.2 Entanglement

L'**entanglement** è una delle proprietà più straordinarie della meccanica quantistica, che si manifesta quando due o più qubit vengono correlati in un modo che il loro stato complessivo non può essere descritto indipendentemente. Quando due o più qubit sono in uno stato di entanglement, lo stato di ciascun qubit dipende istantaneamente dallo stato dell'altro, indipendentemente dalla distanza che li separa, grazie alla natura non-locale della meccanica quantistica. Questo fenomeno, descritto da *Einstein* come "spooky action at a distance", non ha equivalenti nel mondo classico.

Per comprendere meglio questo fenomeno, consideriamo il caso di due qubit descritti da uno spazio di Hilbert, che include tutti i possibili stati combinati. Gli stati base di due qubit possono essere scritti come:

$$|00\rangle |01\rangle |10\rangle |11\rangle$$

Qualsiasi stato sovrapposto può essere descritto come:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

Tuttavia, nel caso dell'entanglement, lo stato complessivo del sistema non può essere scritto come un semplice prodotto di stati di ciascun qubit. Lo abbiamo visto con il **quarto postulato della meccanica quantistica**.

Gli **stati di Bell** rappresentano un esempio classico di stati entangled e sono definiti come stati di sovrapposizione massimale tra due qubit. Essi costituiscono il paradigma fondamentale per lo studio dell'entanglement quantistico. Gli stati di Bell, che sono i seguenti, rappresentano i più semplici esempi di stati entangled:

$$\begin{aligned} |\Phi^+\rangle &= \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \\ |\Phi^-\rangle &= \frac{1}{\sqrt{2}} (|00\rangle - |11\rangle) \\ |\Psi^+\rangle &= \frac{1}{\sqrt{2}} (|01\rangle + |10\rangle) \\ |\Psi^-\rangle &= \frac{1}{\sqrt{2}} (|01\rangle - |10\rangle) \end{aligned}$$

Quando si misura un sistema entangled, il processo di misurazione di un qubit influenza direttamente lo stato del secondo qubit.

In uno stato di Bell come $|\Phi^+\rangle$ o $|\Phi^-\rangle$, i due qubit sono "allineati", ossia misurare uno stato come $|0\rangle$ implica che l'altro sarà misurato come $|0\rangle$, e se uno è $|1\rangle$, anche l'altro lo sarà. In uno stato come $|\Psi^+\rangle$ o $|\Psi^-\rangle$, i due qubit sono in stati opposti: se misuriamo uno come $|0\rangle$, l'altro sarà $|1\rangle$ e viceversa.

In pratica gli stati di Bell descrivono come due qubit possono essere entangled.

3.2.3 Interferenza

I qubit possono anche manifestare il fenomeno dell'interferenza, che si verifica quando le ampiezze di probabilità degli stati si combinano in modi che possono aumentare o ridurre la probabilità di determinati risultati. Essendo l'equazione di Schrödinger un'equazione d'onda, l'interferenza può essere sfruttata per:

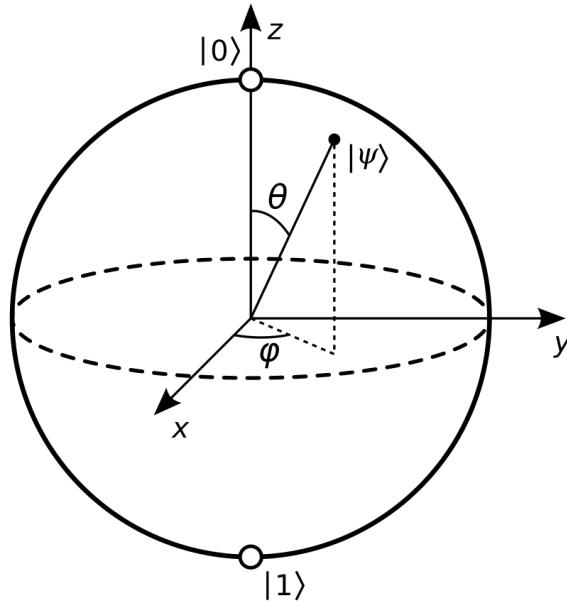
- Ridurre la probabilità che lo stato finale non sia indesiderato (**interferenza quantistica distruttiva**);
- Aumentare la probabilità di trovare i qubit su uno stato finale che codifica la soluzione al problema dato (**interferenza quantistica costruttiva**).

Le tecniche di interferenza sono utilizzate nei circuiti quantistici per manipolare gli stati dei qubit, rendendo alcune configurazioni più probabili di altre. Questo permette di amplificare le probabilità degli stati desiderati e ridurre quelle degli stati indesiderati, migliorando l'efficienza degli algoritmi quantistici.

3.2.4 Effetto Tunnel

Un'altra proprietà fondamentale del qubit è l'**effetto tunnel**, che riveste un ruolo cruciale nel contesto del *Quantum Annealing*. L'effetto tunnel consente a un sistema quantistico, come quello associato a un qubit, di attraversare una barriera energetica anche quando, secondo la meccanica classica, la sua energia non sarebbe sufficiente a superarla. In un problema di ottimizzazione, ciò implica che, se una soluzione prossima al minimo globale è separata da una barriera energetica più alta rispetto a un minimo locale, il sistema quantistico può passare attraverso tale barriera, consentendo di raggiungere il minimo globale.

3.3 Sfera di Bloch



Il comportamento di un qubit può essere rappresentato graficamente sulla **Sfera di Bloch**, una rappresentazione geometrica che ne illustra l'orientamento in uno spazio tridimensionale [9]. La sfera ha raggio unitario, con gli stati base $|0\rangle$ e $|1\rangle$ situati rispettivamente ai poli nord e sud della sfera, mentre gli stati sovrapposti sono rappresentati da punti sulla superficie della sfera, dove ciascun punto corrisponde a una possibile combinazione degli stati base.

Ogni stato del qubit può essere descritto nella forma parametrica:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

Dove:

- θ è l'angolo di latitudine (o angolo zenitale) che misura la distanza angolare dallo stato $|0\rangle$ (polo nord);
- φ è l'angolo di longitudine (o angolo azimutale) che determina la fase relativa tra gli stati $|0\rangle$ e $|1\rangle$.

Pertanto, un generico vettore di stato $|\psi\rangle$ è rappresentato sulla Sfera di Bloch e può essere caratterizzato da un vettore unitario, individuato dai due angoli θ e φ . Questa rappresentazione geometrica aiuta a visualizzare la manipolazione e l'evoluzione di un qubit in un sistema quantistico, specialmente quando si applicano operazioni (ad

esempio con porte quantistiche) che modificano l'orientamento del qubit sulla sfera di Bloch.

La sfera di Bloch visualizza anche il processo di misurazione. In base al **terzo postulato**, una misurazione lungo un asse specifico provoca il collasso dello stato del qubit in uno degli stati base associati a quell'asse.

3.4 Vantaggi computazionali del Qubit

La capacità dei qubit di esistere in più stati contemporaneamente, combinata con fenomeni come l'entanglement e l'interferenza, conferisce ai sistemi quantistici un'enorme potenza computazionale.

In un sistema classico, per rappresentare n bit sono necessarie 2^n possibili combinazioni di stati, ma solo uno di questi stati può essere rappresentato e processato in un dato momento. Questo limita la capacità di esplorare soluzioni simultaneamente, costringendo il sistema a una ricerca sequenziale.

In un sistema quantistico, invece, n qubit possono rappresentare simultaneamente una sovrapposizione di 2^n stati distinti. Grazie a ciò, un singolo qubit può trovarsi in una combinazione di entrambi gli stati base ($|0\rangle$ e $|1\rangle$), e quindi un sistema di n qubit può rappresentare simultaneamente tutte le possibili configurazioni di n bit. Ciò implica che un computer quantistico con un numero relativamente ridotto di qubit ha il potenziale di esplorare uno spazio di soluzioni esponenzialmente più vasto rispetto a un computer classico.

Inoltre, grazie all'entanglement quantistico, i qubit non solo esistono in sovrapposizione, ma possono anche essere correlati in modi che non hanno analoghi nei sistemi classici. Questa interazione permette a un computer quantistico di eseguire operazioni su più stati contemporaneamente, rendendo possibili calcoli molto più complessi in tempi significativamente più brevi.

Pertanto, un algoritmo quantistico può risolvere problemi complessi, come la fattorizzazione di numeri grandi o la ricerca in spazi di soluzione molto ampi, molto più velocemente di quanto potrebbe fare un algoritmo classico.

Un esempio concreto del potenziale dei computer quantistici è la loro capacità di generare numeri casuali. Nei computer classici, la generazione di numeri casuali è generalmente realizzata tramite algoritmi deterministicici noti come generatori di numeri pseudo-casuali. Tali algoritmi, a partire da un valore iniziale chiamato "*seme*" (**seed**), producono una sequenza di numeri che simula la casualità, ma che in realtà è completamente deterministica. Ciò significa che, dato lo stesso seme, verrà sempre generata la stessa sequenza di numeri.

Nei computer quantistici, invece, la generazione di numeri casuali può essere intrinseca al processo stesso, grazie alla natura probabilistica della meccanica quantistica. Questo processo è fondamentalmente non deterministico, poiché il risultato della misurazione è casuale e non può essere previsto con certezza. Sebbene la generazione di numeri casuali possa sembrare un argomento banale, essa riveste un'importanza cruciale, poiché costituisce la base per numerosi algoritmi di crittografia.

Nonostante la potenza computazionale offerta dai qubit, i bit classici rimangono fondamentali nei computer quantistici. Un problema significativo nel contesto del calcolo quantistico è rappresentato dalla **decoerenza quantistica**, che si riferisce alla perdita di coerenza degli stati quantistici a causa di interazioni con l'ambiente esterno, come fluttuazioni termiche o l'influenza di campi elettromagnetici. Ad esempio, una piccola fluttuazione termica può generare l'emissione di un fotone a bassa energia, che interagendo con il qubit ne altera lo stato e ne compromette l'affidabilità del risultato del calcolo.

Sebbene il processo di elaborazione avvenga tramite qubit, l'output finale deve essere convertito in bit classici. Questa conversione è necessaria anche perché i dispositivi di output, come memorie e schermi, operano nel dominio classico e non sono in grado di interpretare direttamente gli stati quantistici. Pertanto, i bit classici svolgono un ruolo cruciale nel rendere i risultati delle operazioni quantistiche comprensibili e utilizzabili nell'ambito delle tecnologie computazionali tradizionali.

Capitolo 4: Quantum Annealing



Quantum Annealer della D-Wave

4.1 Introduzione alla Computazione Quantistica

Il modello predominante di computazione quantistica è il **Gate-Based Quantum Computing**, che rappresenta un'estensione quantistica del paradigma classico dei circuiti logici. In questo approccio, i calcoli e le trasformazioni vengono eseguiti applicando una sequenza di operazioni ai qubit, attraverso **porte logiche quantistiche**. Queste operazioni manipolano lo stato quantistico dei qubit, sfruttando fenomeni come la sovrapposizione e l'entanglement, che permettono di elaborare informazioni in modi impossibili per i computer classici.

Il processo inizia con l'inizializzazione dei qubit in uno stato noto. Successivamente, le porte quantistiche vengono applicate per modificare gli stati dei qubit, codificando il problema e realizzando il calcolo desiderato.

Grazie a un insieme definito di operazioni logiche di base, è possibile progettare algoritmi scalabili e, teoricamente, universali.

Tuttavia, questo approccio richiede un controllo estremamente preciso e costante degli stati dei qubit. Una delle principali sfide consiste nel mantenere la coerenza quantistica, ossia la capacità dei qubit di conservare la sovrapposizione di stati per tempi sufficientemente lunghi. Questa caratteristica è essenziale per il corretto funzionamento del calcolo quantistico, ma al momento rappresenta una sfida tecnica significativa.

In alternativa al modello a porte quantistiche, si distingue il *Quantum Annealing* [10], un approccio euristico specializzato per la risoluzione di problemi di ottimizzazione combinatoria (sebbene possa essere applicato anche ad altre classi di problemi).

Questo metodo si basa sul principio dell'**evoluzione adiabatica**, che sfrutta anche l'**effetto tunnel** per esplorare lo spazio delle soluzioni. Il problema di ottimizzazione viene formulato come la minimizzazione di un **paesaggio energetico**, una rappresentazione geometrica in cui ogni configurazione del sistema (ossia ogni possibile soluzione) è associata a un livello di energia specifico. In questa rappresentazione, le soluzioni corrispondono a punti del paesaggio e l'altezza di ogni punto rappresenta il valore di energia associato. Il *minimo globale*, che identifica la soluzione ottimale, corrisponde al punto più basso del paesaggio, mentre i *minimi locali* rappresentano soluzioni subottimali che non soddisfano il criterio di globalità.

Il *Quantum Annealing* sfrutta la naturale tendenza dei sistemi fisici a evolvere verso lo stato ad energia minima. Il problema da risolvere viene codificato in un **Hamiltoniano**, che descrive la dinamica del sistema quantistico e lo spettro degli stati energetici associati. L'obiettivo è identificare lo stato fondamentale del sistema, corrispondente al minimo globale del paesaggio energetico. In termini formali, l'Hamiltoniano H è un operatore hermitiano che agisce nello spazio di Hilbert e definisce sia i possibili valori di energia del sistema (i suoi autovalori) sia la sua evoluzione temporale (come visto nella sezione dedicata al **secondo postulato della meccanica quantistica**).

Nel *Quantum Annealing*, il sistema parte da un **Hamiltoniano iniziale** H_i , noto e facile da preparare, ed evolve gradualmente verso un **Hamiltoniano finale** H_f , che codifica il problema di interesse. Se l'evoluzione è sufficientemente lenta, il sistema rimarrà nello stato fondamentale durante l'intero processo, consentendo di identificare il minimo globale una volta completato il processo.

Un elemento distintivo del *Quantum Annealing*, rispetto agli approcci classici, è l'utilizzo dell'**effetto tunnel**, che permette al sistema di attraversare le barriere energetiche anziché scalarle, riducendo il rischio di rimanere bloccato nei minimi locali. Per analogia, si può immaginare un viaggiatore che attraversa un paesaggio collinare per raggiungere la valle più profonda: un algoritmo classico dovrebbe

"scalare" ogni collina e "descendere" in ogni valle, rischiando di fermarsi in una valle che non è quella più bassa. Il *Quantum Annealing*, invece, sfrutta il tunnel quantistico per "passare attraverso" le colline, raggiungendo con maggiore efficienza il punto più basso.

Un'altra differenza significativa rispetto al Gate-Based Quantum Computing risiede nella natura del paradigma operativo. Mentre il Gate-Based si basa su sequenze discrete di operazioni logiche applicate ai qubit, il *Quantum Annealing* opera in modo continuo e analogico, modificando gradualmente l'Hamiltoniano del sistema. Questo approccio è inoltre più robusto rispetto agli effetti di rumore e decoerenza, poiché non richiede un controllo preciso di sequenze logiche, ma solo la preservazione delle condizioni per il **processo adiabatico**.

4.2 Problemi di Ottimizzazione

Il *Quantum Annealing* si è dimostrato particolarmente efficace nella risoluzione di **problemi di ottimizzazione combinatoria**, ma è utile chiarire cosa si intenda esattamente per questo tipo di problemi.

Un **problema di ottimizzazione** consiste nel determinare la soluzione ottimale, ossia il massimo o il minimo di una funzione obiettivo, rispettando un insieme di vincoli. Formalmente, può essere espresso come segue:

$$\begin{cases} \min_{x \in S}(f(x)) \\ g_i(x) \leq 0 \\ h_i(x) = 0 \end{cases}$$

Dove:

- x rappresenta la variabile, che può essere un vettore multidimensionale;
- $f(x)$ è la funzione obiettivo da ottimizzare;
- S è lo spazio delle possibili soluzioni;
- $g(x)$ e $h(x)$ definiscono i vincoli del problema, rispettivamente, come diseguaglianze e uguaglianze.

Nel caso specifico dei problemi di ottimizzazione combinatoria, l'obiettivo è trovare la migliore combinazione tra un insieme finito (o numerabile) di soluzioni. Un esempio tipico è il **problema del Comessoso Viaggiatore** (*Travelling Salesman Problem*, o **TSP**), che consiste nel determinare il percorso più breve che attraversa un insieme di città, visitandole una sola volta ciascuna, per poi tornare al punto di partenza.

La difficoltà di risoluzione è descritta dalla **teoria della complessità computazionale**, che classifica i problemi in base al tempo e alla memoria richiesti dagli algoritmi per risolverli.

I problemi di classe **P** includono i problemi che possono essere risolti da un algoritmo deterministico in tempo polinomiale rispetto alla dimensione dell'input. I problemi di classe **NP** includono i problemi per cui una soluzione può essere verificata in tempo polinomiale da un algoritmo deterministico, anche se trovarla potrebbe richiedere un tempo non polinomiale. Tra questi rientrano anche i problemi **NP-Completi**. Sono quelli più difficili della classe NP. Se uno di questi problemi può essere risolto in tempo polinomiale, allora tutti i problemi NP possono esserlo. Tuttavia, la dimostrazione che un problema NP-Completo possa essere risolto in tempo polinomiale, è uno dei **problemi del millennio**, tutt'ora irrisolto. Poi ci sono i problemi **NP-Hard**, che sono almeno tanto difficili quanto i problemi NP-completi, ma non appartengono necessariamente alla classe NP. Il problema del Commesso Viaggiatore è un problema *NP-Hard*.

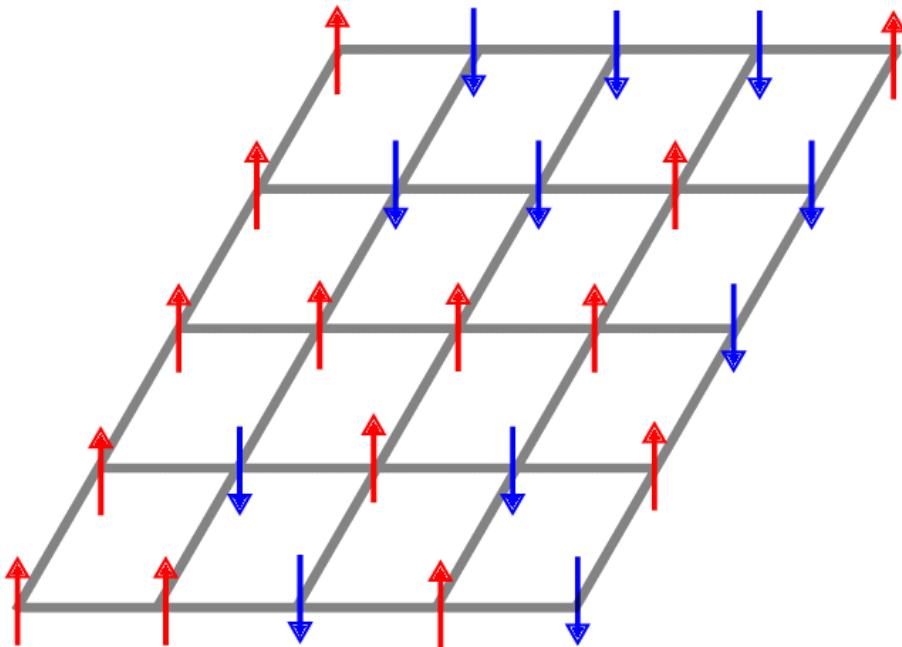
4.3 Modello di Ising

Il **modello di Ising** è il fondamento teorico su cui si basa il *Quantum Annealing* ed è stato originariamente introdotto per descrivere il comportamento dei materiali ferromagnetici [11]. Questo modello rappresenta un sistema di N spin, ognuno dei quali può assumere due stati (-1 o +1), interagenti tra loro attraverso un reticolo. La configurazione del sistema è descritta da un Hamiltoniano che include le interazioni tra spin adiacenti e un campo magnetico esterno (se presente):

$$H = -\frac{1}{2} \sum_{i,j} J_{ij} s_i \cdot s_j - \sum_i h_i s_i$$

Dove:

- $s_i \in \{-1; 1\}$ è lo stato dello spin i ;
- J_{ij} è un parametro che rappresenta la forza di interazione tra gli spin i e j :
 - Se $J_{ij} > 0$, gli spin tendono ad allinearsi, promuovendo il *ferromagnetismo*;
 - Se $J_{ij} < 0$, si tende all'*antiferromagnetismo*, in cui gli spin tendono ad orientarsi in direzioni opposte.
- h_i è un campo magnetico esterno che agisce sullo spin i , influenzando la probabilità che lo spin si orienti in una direzione.



Raffigurazione grafica del Modello di Ising

Il fattore $\frac{1}{2}$ viene introdotto questa formulazione per evitare di contare due volte le interazioni tra coppie di spin. L'obiettivo è trovare la configurazione degli spin che minimizza l'Hamiltoniano, corrispondente al minimo globale dell'energia del sistema.

In pratica, la configurazione di ciascun spin dipende dalle interazioni con gli spin adiacenti e dai campi magnetici esterni. Il modello di Ising descrive il sistema in termini energetici, dove lo stato fondamentale è la configurazione che minimizza l'energia complessiva. A temperature elevate, gli spin si distribuiscono casualmente, mentre a basse temperature il sistema tende a ridurre l'energia totale, favorendo l'allineamento degli spin. Questo processo porta a una transizione di fase tra uno stato disordinato a temperatura alta e uno ordinato a temperatura bassa.

Il modello di Ising è particolarmente adatto per rappresentare problemi di ottimizzazione combinatoria, poiché molte funzioni di costo possono essere tradotte in termini di interazioni tra spin. Le interazioni J_{ij} e i campi magnetici h_i sono scelti in modo da modellare i vincoli e gli obiettivi del problema. Il *Quantum Annealing* sfrutta proprio questo formalismo: l'ottimizzazione viene formulata come la minimizzazione di un paesaggio energetico, e questo modello fornisce una rappresentazione efficace di tale paesaggio. In altre parole, il problema di ottimizzazione è tradotto in un sistema fisico che evolve verso il minimo energetico, attraverso un processo (adiabatico) che consente di esplorare lo spazio delle soluzioni.

Il modello di Ising può essere rappresentato in una forma alternativa che ci risulta particolarmente utile per la trattazione di problemi di ottimizzazione:

$$H = \sum_{i=1}^N \sum_{j < i}^N x_i x_j Q_{ij} + \sum_{i=1}^N x_i Q_{ii}$$

In questa versione, x_i rappresenta lo stato dell' i -esimo spin (s_i) e il prodotto $s_i s_j$ è ora espresso come $x_i x_j$ senza modificare la natura dell'interazione tra gli spin.

Q è una matrice quadrata di dimensione NxN, definita come:

$$Q_{ij} = \begin{cases} -\frac{1}{2} J_{ij}, & \text{se } i \neq j, \\ -h_i, & \text{se } i = j. \end{cases}$$

In questa rappresentazione, Q incapsula sia le interazioni J_{ij} (nei termini non diagonali), sia i campi esterni h_i (nei termini diagonali). In pratica, Q rappresenta il paesaggio energetico del sistema.

Questa formulazione compatta semplifica notevolmente il trattamento computazionale del problema, rendendo più efficienti le operazioni di calcolo e facilitando l'implementazione in contesti di ottimizzazione quantistica.

4.4 Quadratic Unconstrained Binary Optimization

Il modello di Ising, sebbene utile per rappresentare vari problemi di ottimizzazione, presenta alcune limitazioni pratiche. In primo luogo, la formulazione di problemi utilizzando questo modello può risultare laboriosa e poco intuitiva, specialmente quando si cerca di tradurre un problema in termini di interazioni tra spin. Inoltre, una delle principali restrizioni di questo modello è che le variabili di stato possono assumere solo i valori +1 e -1, escludendo la possibilità dello stato 0.

Un'alternativa al modello di Ising è il **Quadratic Unconstrained Binary Optimization**, abbreviato in **QUBO**, un altro modello matematico che si presta particolarmente bene alla formulazione di problemi di ottimizzazione combinatoria [12]. La funzione obiettivo di un problema QUBO è espressa come:

$$f(x) = x^T Q x = \sum_{i=1}^n \sum_{j=i}^n Q_{ij} \cdot x_i \cdot x_j$$

dove:

- x è un vettore binario ($x_i \in \{0; 1\}$);
- Q è una matrice simmetrica di dimensioni NxN, che definisce le interazioni tra le variabili x_i e x_j .

Gli elementi Q_{ij} rappresentano i pesi associati a ciascuna coppia di indici i e j del vettore binario x . In modo intuitivo, il peso Q_{ij} contribuisce al valore complessivo della funzione obiettivo solo se entrambe le variabili x_i e x_j sono uguali a 1.

Questo tipo di problema è definito senza vincoli ("unconstrained"), in quanto la formulazione non include esplicitamente vincoli, sebbene essi possano essere incorporati indirettamente all'interno della matrice Q , sotto forma di *ricompense* e *penalità*.

Nel caso del QUBO, l'obiettivo è minimizzare una funzione quadratica con variabili binarie. Nel modello di Ising, invece, si cerca di minimizzare un Hamiltoniano che descrive le interazioni tra spin binari. Nonostante le differenze apparenti, i due modelli sono sostanzialmente equivalenti dal punto di vista formale: l'unica differenza significativa riguarda i valori che le variabili di stato possono assumere. Infatti, la formulazione QUBO e il modello di Ising sono **isomorfi** (*vedi Appendice B*), ossia un problema in forma QUBO può essere espresso attraverso il modello di Ising e viceversa.

La formulazione QUBO è spesso preferita rispetto al modello di Ising, in quanto risulta generalmente più intuitiva e conveniente per esprimere un problema di ottimizzazione. In pratica, i problemi vengono inizialmente formulati in forma QUBO, dopodiché questa viene trasformata nel modello di Ising per applicare il processo di *Quantum Annealing*. Una volta trovata la soluzione nel modello di Ising, è possibile riconvertirla in una soluzione per il problema QUBO originale.

Esiste un algoritmo in grado di portare una matrice Q in forma QUBO ad una in forma di Ising.

```

1 import numpy as np
2
3 # Funzione per convertire una matrice qubo [creata attraverso
4 # numpy] in una matrice di Ising
5 def qubo_to_ising(qubo):
6     if qubo.shape[0] != qubo.shape[1]:
7         return None
8
9     N = qubo.shape[0]
10    Ising = np.zeros((N, N))
11
12    for i in range(N):
13        Ising[i, i] = 0.5 * qubo[i, i]
14
15    for i in range(N):
16        for j in range(i + 1, N):
17            Ising[i, j] = 0.25 * qubo[i, j]
18            Ising[i, i] += 0.25 * qubo[i, j]
19            Ising[j, j] += 0.25 * qubo[i, j]
20
21    return Ising

```

Una volta applicato il *Quantum Annealing*, otterremo la soluzione del problema nella forma del modello di Ising, che indicheremo come x_I . Per ottenere la soluzione corrispondente nel formato del QUBO, denotata come x_Q , è possibile utilizzare la seguente trasformazione:

$$x_Q = \frac{x_I + 1}{2}$$

Abbiamo già specificato che il QUBO è particolarmente efficace per rappresentare i problemi di ottimizzazione combinatoria, ma non è l'unica classe di problemi che è in grado di formalizzare. Ad esempio i problemi di Machine Learning (Clustering, Feature Selection) e i problemi di teoria dei grafi (Max-Cut Problem, il Graph Coloring, la Community Detection) sono formalizzabili attraverso il QUBO.

Ising model

$$\operatorname{argmin}_{x \in \{-1,1\}^N} \sum_{i=0} \sum_{j < i} x_i x_j Q_{ij} + \sum_{i=0} x_i Q_{ii}$$

QUBO formulation

$$\operatorname{argmin}_{x \in \{0,1\}^N} \sum_{i=0} \sum_{j < i} x_i x_j Q_{ij} + \sum_{i=0} x_i Q_{ii}$$

Analogia tra il Modello di Ising e il QUBO

4.5 Teorema Adiabatico e Processo Adiabatico

Teorema (Teorema Adiabatico). *Se un sistema quantistico è inizialmente in uno degli autostati (ad esempio lo stato fondamentale) di un Hamiltoniano H_i e se l'Hamiltoniano cambia abbastanza lentamente nel tempo, il sistema rimarrà in uno degli autostati corrispondenti dell'Hamiltoniano istantaneo $H(t)$ per tutta l'evoluzione.*

Introdotto formalmente da Born e Fock nel 1928, il **Teorema Adiabatico** è un principio fondamentale della meccanica quantistica che descrive il comportamento di un sistema quantistico sottoposto a un'evoluzione lenta e continua del suo Hamiltoniano [13]. Abbiamo visto con il **secondo postulato della meccanica quantistica** che l'Hamiltoniano rappresenta l'energia totale del sistema e governa la sua dinamica, determinando l'evoluzione dello stato quantistico nel tempo.

Lo **stato fondamentale** rappresenta la configurazione ad energia minima del sistema e corrisponde alla soluzione ottimale del problema di ottimizzazione. In pratica è lo stato ad energia minima descritto dall'Hamiltoniano. Gli **stati eccitati**, invece, sono configurazioni di energia superiore che rappresentano soluzioni subottimali o non valide.

Il **Quantum Annealing** si fonda sull'applicazione del Teorema Adiabatico. Questo approccio prevede l'evoluzione di un sistema quantistico descritto da un Hamiltoniano dipendente dal tempo, noto come **Hamiltoniano totale**, il quale cambia gradualmente seguendo un'evoluzione adiabatica.

All'inizio del processo, il sistema è caratterizzato da un **Hamiltoniano iniziale** H_i , il cui stato fondamentale è noto e facilmente preparabile. Generalmente, questo Hamiltoniano è indipendente dal problema specifico e descrive uno stato quantistico "neutro", ma stabile. Ad esempio, un campo magnetico uniforme applicato ai qubit può indurli in uno stato di sovrapposizione, in cui ognuno ha la stessa probabilità di trovarsi negli stati $|0\rangle$ e $|1\rangle$.

Durante l'evoluzione, l'Hamiltoniano totale viene lentamente modificato fino a raggiungere l'**Hamiltoniano finale** H_f (o **Hamiltoniano del problema**). Questo Hamiltoniano rappresenta il problema di ottimizzazione da risolvere e descrive la funzione obiettivo che il sistema deve minimizzare [14]. A differenza dell'Hamiltoniano iniziale, che è progettato per essere semplice e facilmente preparabile, H_f incorpora le specificità del problema da risolvere, includendo i vincoli e le caratteristiche che definiscono le soluzioni ammissibili. In particolare, l'Hamiltoniano finale è strutturato secondo il **modello di Ising**. In questo contesto, lo stato fondamentale di H_f rappresenta la configurazione di spin che minimizza l'energia totale del sistema. Tale configurazione corrisponde alla soluzione ottimale del problema di ottimizzazione.

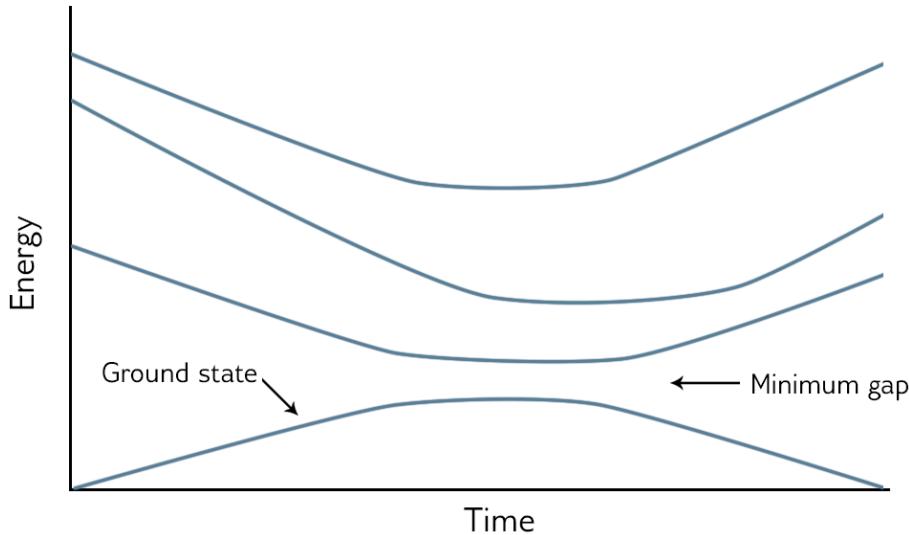


Grafico dell'energia in funzione del tempo durante un processo di Quantum Annealing. L'origine degli assi è lo stato fondamentale (ground state) dell'Hamiltoniano iniziale, lo stato di partenza dell'Hamiltoniano totale. Se l'evoluzione è adiabatica seguiremo la curva tracciata dallo stato fondamentale, supereremo il gap minimo (minimum gap) fino ad arrivare allo stato fondamentale dell'Hamiltoniano finale, senza "salti" energetici a stati eccitati.

Il comportamento del sistema è governato dall'evoluzione dell'Hamiltoniano totale, che può essere formalizzata come segue:

$$H(t) = s(t) \cdot H_f + [1 - s(t)] \cdot H_i$$

Dove è $s(t)$ la funzione di programmazione, o **scheduling function**, e T rappresenta il tempo totale di evoluzione. $s(t) : [0; T] \rightarrow [0; 1]$ è una funzione temporale monotona crescente con $s(0) = 0$ e $s(T) = 1$. Essa determina la transizione dall'Hamiltoniano iniziale all'Hamiltoniano finale.

La transizione da H_i a H_f deve avvenire lentamente, in modo da garantire che il sistema rimanga nello stato fondamentale durante tutto il processo, come previsto dal Teorema Adiabatico. Inizialmente lo stato fondamentale di H_i è ben separato dagli stati eccitati. Una volta introdotto l'hamiltoniano del problema, gli stati eccitati si avvicinano allo stato fondamentale e viceversa. In prossimità di queste transizioni, si verificano fenomeni di **Avoided Crossing** [15], ovvero situazioni in cui il gap energetico tra due livelli si riduce notevolmente senza che i due livelli si intersechino completamente.

Matematicamente, ciò significa che, durante l'evoluzione temporale dell'Hamiltoniano totale, due autovalori si avvicinano sempre più senza mai toccarsi, ma si allontanano evitando effettivamente un'intersezione diretta. Un gap energetico ridotto aumenta il rischio di transizioni non adiabatiche, in cui il sistema potrebbe passare dallo stato fondamentale a uno degli stati eccitati, compromettendo la soluzione ottimale. Per minimizzare questo rischio, il processo evolutivo deve essere sufficientemente adiabatico, permettendo al sistema di rimanere nello stato fondamentale e garantire la corretta evoluzione verso la soluzione ottimale.

Per analizzare la probabilità che il sistema attraversi un Avoided Crossing, si utilizza la **teoria di Landau-Zener** [16], che fornisce una stima di tale probabilità in relazione alla velocità di variazione dell'Hamiltoniano e al gap energetico. Secondo questa teoria, se il sistema attraversa rapidamente un Avoided Crossing, la probabilità di transizione a uno stato eccitato aumenta significativamente. Al contrario, se il sistema evolve lentamente, la probabilità di rimanere nello stato fondamentale resta elevata. La probabilità di transizione P_{LZ} è data dalla formula:

$$P_{LZ} = e^{-\frac{\pi g^2}{2\hbar v}}$$

dove:

- g è il gap energetico minimo;
- \hbar è la costante di Planck ridotta;
- v è la velocità di variazione dell'Hamiltoniano totale.

Si osserva facilmente come una transizione lenta (piccolo v) favorisca l'evoluzione adiabatica, mentre una transizione rapida aumenti la probabilità di errori.

Il Gap Minimo è la minima separazione energetica tra lo stato fondamentale e il livello eccitato più vicino e rappresenta un caso particolare di Avoided Crossing. Se tale gap è troppo piccolo, il sistema può facilmente saltare a uno stato eccitato a causa di fluttuazioni energetiche.

Il gap minimo riveste un'importanza cruciale nell'analisi delle transizioni adiabatiche, poiché fornisce una condizione fondamentale per garantire che il processo evolutivo avvenga in maniera adiabatica. In particolare, per evitare transizioni non adiabatiche e mantenere il sistema nello stato fondamentale, è necessario che il tempo di evoluzione T sia sufficientemente grande rispetto all'inverso del quadrato del gap minimo.

Formalmente, questo requisito può essere espresso come:

$$T \approx O\left(\frac{1}{g_{min}^2}\right)$$

dove g_{min} rappresenta il gap minimo tra lo stato fondamentale e il livello eccitato più vicino. Un gap molto piccolo, perciò, richiede tempi di evoluzione più lunghi per mantenere l'adiabaticità del processo e prevenire transizioni verso stati eccitati.

Nei sistemi di grandi dimensioni, la riduzione del gap energetico rappresenta una sfida, poiché il gap minimo diminuisce esponenzialmente all'aumentare del numero di qubit. Quando questo gap diventa troppo piccolo, il tempo necessario per mantenere l'evoluzione adiabatica in condizioni praticabili diventa estremamente lungo, poiché la separazione energetica tra lo stato fondamentale e gli stati eccitati si riduce drasticamente. In tali circostanze, per evitare che il sistema salti a stati eccitati durante l'evoluzione, il tempo di evoluzione T dovrebbe essere sufficientemente lungo. Tuttavia, se il tempo disponibile è insufficiente, il sistema rischia di transitare a stati non ottimali, compromettendo la qualità della soluzione trovata.

I dispositivi che implementano il *Quantum Annealing*, come quelli prodotti dalla *D-Wave*, affrontano questa sfida utilizzando una versione approssimata del processo adiabatico. Questi dispositivi sono progettati per ottimizzare la rapidità e la scalabilità delle soluzioni, sacrificando però una perfetta aderenza al Teorema Adiabatico.

4.6 Campo Trasversale e Matrici di Pauli

Il **campo trasversale** è una componente dell'Hamiltoniano iniziale e rappresenta un termine che introduce **fluttuazioni quantistiche** nel sistema. Questo campo serve a inizializzare il sistema quantistico in uno stato fondamentale ben definito, che è una sovrapposizione uniforme di tutti gli stati possibili nello spazio delle soluzioni.

Per rappresentare l'effetto del campo trasversale, è necessario introdurre le **matrici di Pauli** [17]. Esse sono definite come matrici 2x2 e rappresentano gli operatori di spin lungo tre direzioni ortogonali.

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

- σ_x : rappresenta le transizioni tra stati di spin opposti. È associata al campo trasversale, che induce fluttuazioni quantistiche;
- σ_y : utilizzata in altri contesti della meccanica quantistica, ma meno rilevante nel contesto trattato;
- σ_z : rappresenta lo stato dello spin lungo l'asse z , che corrisponde ai valori discreti +1 (spin up \uparrow) o -1 (spin down \downarrow). È usata per descrivere l'Hamiltoniano finale.

L'Hamiltoniano iniziale, che include il campo trasversale, è comunemente espresso come:

$$H_i = \Gamma(t) \sum_i \sigma_i^x$$

dove:

- $\Gamma(t)$ è la forza del campo trasversale, una funzione monotona decrescente;
- σ_i^x è una matrice di Pauli che descrive le interazioni lungo l'asse x per il qubit i .

All'inizio del processo il campo trasversale è molto forte. Questo introduce una sovrapposizione tra stati quantistici, incoraggiando la ricerca di soluzioni globali nel paesaggio energetico. Durante il processo, il campo trasversale viene ridotto gradualmente fino ad essere nullo. Con la diminuzione del campo trasversale, il sistema diventa sempre più dominato dall'Hamiltoniano del problema, che rappresenta la funzione obiettivo da ottimizzare.

Il modello di Ising è formalizzato utilizzando la matrice di Pauli σ_z , che per l'appunto caratterizza le interazioni tra gli spin:

$$H_f = \sum_{i=0}^N \sum_{j=i}^N J_{ij} \sigma_i^z \sigma_j^z + \sum_{i=0}^N h_i \sigma_i^z$$

dove:

- J_{ij} : Coefficiente di accoppiamento tra i qubit i e j , che descrive l'interazione tra di essi;
- h_i : Campo magnetico locale applicato al qubit i ;
- σ_i^z : Matrice di Pauli lungo l'asse z , che descrive l'orientamento dello spin (valori +1 o -1).

Quando la forza del campo trasversale $\Gamma(T)$ diventa trascurabile, cioè tende a zero ($\Gamma(T) \approx 0$), il sistema cessa di evolversi in modo dinamico e raggiunge uno stato di equilibrio. In questa fase, il sistema si "congela" nel suo stato fondamentale, corrispondente al minimo energetico dell'Hamiltoniano finale H_f .

Considerando gli effetti del campo trasversale e utilizzando il formalismo delle matrici di Pauli, possiamo esprimere l'evoluzione temporale dell'Hamiltoniano totale come segue:

$$H(t) = s(t) \cdot H_f + [1-s(t)] \cdot H_i = s(t) \left[\sum_{i=0}^N \sum_{j=i}^N J_{ij} \sigma_i^z \sigma_j^z + \sum_{i=0}^N h_i \sigma_i^z \right] + [1-s(t)] \sum_{i=0}^N \sigma_i^x$$

Il termine $[1-s(t)]$ modula l'intensità dell'interazione tra i qubit mediante il campo trasversale. Poiché il campo trasversale è già integrato nella formulazione tramite $[1-s(t)]$, non è necessario introdurre un altro parametro, come $\Gamma(t)$, per controllare l'intensità di tale campo. In altre parole, la funzione $s(t)$ assorbe già il ruolo di $\Gamma(t)$, rendendo superfluo un ulteriore fattore di moltiplicazione per il termine $\sum_{i=0}^N \sigma_i^x$.

Riassumendo, il campo trasversale consente di inizializzare il sistema e introdurre fluttuazioni quantistiche, mentre le matrici di Pauli forniscono il formalismo matematico per descrivere sia le interazioni tra qubit sia la dinamica dello stato quantistico.

4.7 Effetto Tunnel nel Quantum Annealing

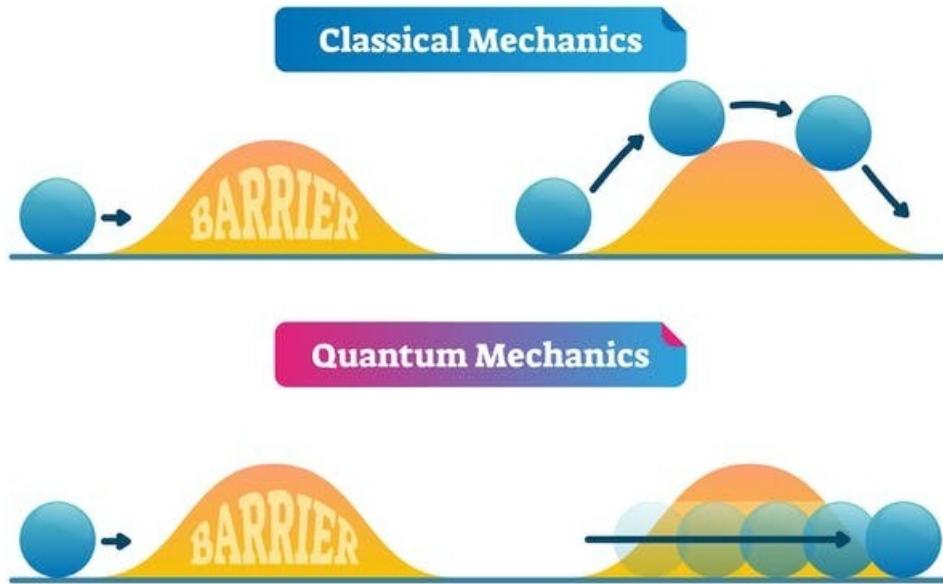


Illustrazione grafica dell'Effetto Tunnel

Le *fluttuazioni quantistiche*, introdotte dal , permettono agli spin di "ruotare" e di non essere confinati negli stati classici $\{-1; +1\}$ del *modello di Ising*. Questo facilita il **tunneling**.

Nella meccanica quantistica, una particella è descritta da una funzione d'onda che non è limitata a valori distinti o a una regione dello spazio ben definita, ma si estende potenzialmente in tutto lo spazio, con un'ampiezza che può essere diversa da zero anche in quelle regioni che, secondo la meccanica classica, sarebbero inaccessibili per la particella, note come **regioni proibite**.

Quando una particella si trova in prossimità di una barriera di potenziale (una regione dello spazio in cui l'energia potenziale è superiore all'energia cinetica della particella) la funzione d'onda non si annulla immediatamente al di là della barriera. Al contrario, la funzione d'onda mostra un decadimento esponenziale all'interno della barriera, mantenendo però una probabilità finita di attraversarla. Questo fenomeno, chiamato **effetto tunnel**, permette alla particella di superare la barriera senza dover possedere l'energia necessaria per scalarla, come invece sarebbe richiesto dalla meccanica classica [18].

Come già detto, nel *Quantum Annealing*, l'effetto tunnel consente al sistema quantistico di attraversare barriere energetiche che separano i minimi locali dal minimo globale del paesaggio energetico. Questo è particolarmente vantaggioso nella risoluzione dei problemi di ottimizzazione.

La probabilità di tunneling è determinata da tre fattori principali:

- **Altezza della barriera (Δ):** rappresenta la differenza di energia tra il minimo locale e il picco della barriera potenziale. Barriere più alte riducono la probabilità che il sistema attraversi il picco energetico per raggiungere un minimo globale;
- **Larghezza della barriera (ω):** corrisponde alla distanza che il sistema deve percorrere nello spazio delle configurazioni degli stati per attraversare la barriera. Una barriera più larga rende il tunneling meno probabile;
- **Forza del campo trasversale ($\Gamma(t)$):** controlla la probabilità di tunneling e viene ridotta durante il processo di *Quantum Annealing*. Maggiore è $\Gamma(t)$, maggiore è la capacità del sistema di "tunnelizzare" attraverso le barriere.

Può essere espressa come:

$$P \approx e^{-\frac{\sqrt{\Delta}\omega}{\Gamma(t)}}$$

4.8 Un confronto con il Simulated Annealing

La ricottura (o annealing), è una tecnica metallurgica finalizzata a migliorare le proprietà meccaniche e strutturali di un materiale, come l'aumento della duttilità e la riduzione delle tensioni interne. Questo processo viene utilizzato per ottimizzare la microstruttura dei metalli, riducendo difetti cristallini e migliorandone sia la lavorabilità che le prestazioni meccaniche.

Il processo di annealing si articola in tre fasi principali:

1. **Riscaldamento:** Il materiale viene portato a una temperatura sufficientemente alta da consentire il movimento degli atomi all'interno della struttura cristallina. La temperatura raggiunta è tipicamente vicina al punto di ricristallizzazione del materiale.
2. **Mantenimento della temperatura:** Durante questa fase, il materiale viene mantenuto ad un'alta temperatura per un periodo di tempo specifico. Questo permette agli atomi di stabilirsi in configurazioni energetiche più favorevoli, riducendo difetti strutturali.
3. **Raffreddamento lento:** Dopo il periodo di mantenimento, la temperatura viene gradualmente ridotta in modo controllato. Questo consente al materiale di raggiungere uno stato stabile con una struttura cristallina migliorata e minime tensioni residue. Un raffreddamento lento è fondamentale per evitare la formazione di nuovi difetti.

Sia il **Simulated Annealing** che il **Quantum Annealing** prendono spunto da questo processo, solo che il primo utilizza dei metodi classici per trovare la soluzione ad un problema di ottimizzazione, mentre il secondo usa delle proprietà della meccanica quantistica [19].

Il Simulated Annealing è un algoritmo euristico che affronta problemi di ottimizzazione formulati tramite una funzione obiettivo di molte variabili, soggetta a vincoli matematici. Il principio di funzionamento simula un processo di raffreddamento lento per individuare il minimo globale della funzione obiettivo. Ogni configurazione del sistema è associata a un valore di "energia" e l'obiettivo è minimizzare tale energia individuando la configurazione ottimale.

La temperatura simula l'energia termica. Inizialmente è alta, permettendo al sistema di esplorare configurazioni meno favorevoli. Gradualmente, la temperatura viene ridotta, restringendo la ricerca a configurazioni energeticamente favorevoli.

L'algoritmo inizia con una configurazione iniziale generata casualmente e un valore di temperatura elevato. La configurazione rappresenta uno stato specifico del sistema da ottimizzare, associato a un valore della funzione obiettivo, interpretato nel contesto dell'algoritmo come energia (cioè il costo della soluzione).

Dopo l'**inizializzazione**, l'algoritmo procede secondo i seguenti passi:

1. **Generazione di una nuova configurazione:** Una nuova configurazione viene ottenuta applicando una modifica locale a quella corrente. Ad esempio, nel problema del Commesso Viaggiatore (TSP), ciò potrebbe consistere nello scambiare due città nell'ordine di visita.
2. **Calcolo della variazione di energia:** La differenza di energia, ΔE , tra la nuova configurazione e quella corrente viene calcolata utilizzando la funzione obiettivo.
3. **Accettazione o rifiuto della nuova configurazione:**
 - Se $\Delta E \leq 0$: La nuova configurazione, avendo un'energia minore o uguale, viene accettata e sostituisce quella corrente.
 - Se $\Delta E > 0$: La configurazione con energia maggiore viene accettata con probabilità
$$P \approx e^{-\frac{\Delta E}{T}}$$

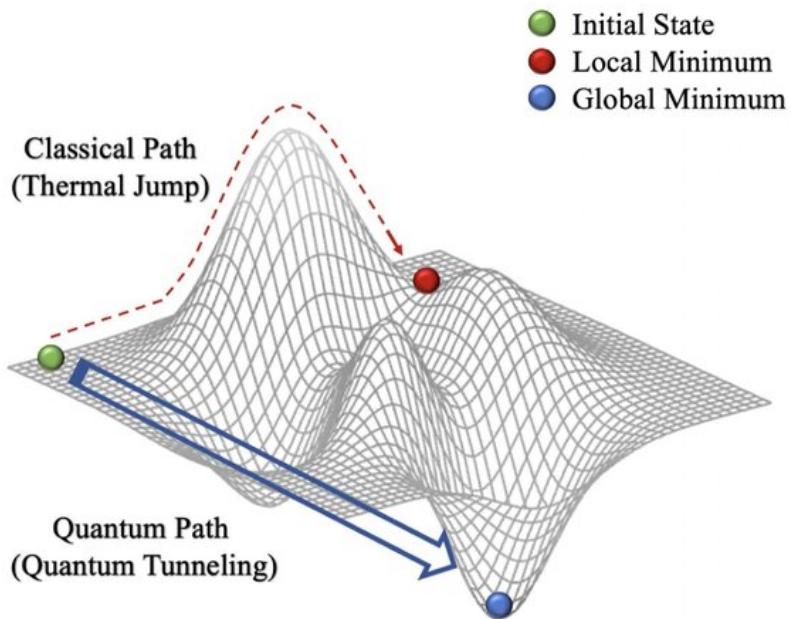
dove T è la temperatura corrente. Questa probabilità segue la **distribuzione di Boltzmann**, che descrive la probabilità di trovare un sistema fisico in uno stato di energia E a una temperatura T . Ciò consente al sistema di saltare occasionalmente verso configurazioni meno vantaggiose, evitando di rimanere bloccato in minimi locali.

4. **Aggiornamento della temperatura:** La temperatura viene ridotta seguendo una legge di raffreddamento, tipicamente: $T_i = \alpha \cdot T_{i-1}$ dove $\alpha \in (0; 1)$ è il fattore di raffreddamento. Valori di α vicini a 1 garantiscono un raffreddamento lento, essenziale per esplorare efficacemente lo spazio delle soluzioni.
5. **Condizione di arresto:** L'algoritmo termina quando la temperatura raggiunge una soglia minima prestabilita. A questo punto, l'ultima configurazione accettata viene presa come soluzione finale. In caso contrario, si ritorna al passo 1.

```

1 def simulated_annealing(init_temp, final_temp, init_placement,
2                           cost_function, perturb_function, schedule_function,
3                           inner_loop_criterion):
4     """
5         Algoritmo Simulated Annealing
6         :param init_temp: Temperatura iniziale
7         :param final_temp: Temperatura finale
8         :param init_placement: Posizionamento (o soluzione
9             iniziale)
10        :param cost_function: Funzione per valutare il costo di un
11            posizionamento
12        :param perturb_function: Funzione per generare una nuova
13            soluzione
14        :param schedule_function: Funzione per aggiornare la
15            temperatura
16        :param inner_loop_criterion: Funzione per verificare se il
17            ciclo interno deve terminare
18        :return: Il miglior posizionamento trovato
19    """
20
21    temp = init_temp
22    place = init_placement
23
24    while temp > final_temp:
25        while not inner_loop_criterion():
26            new_place = perturb_function(place)
27            delta_cost = cost_function(new_place) -
28                cost_function(place)
29
30            if delta_cost < 0:
31                place = new_place
32            else:
33                if random.random() > math.exp(-delta_cost /
34                    temp):
35                    place = new_place
36
37            temp = schedule_function(temp)
38
39    return place

```



Confronto fra l'applicazione del Simulated Annealing e del Quantum Annealing, mostrando il vantaggio dell'effetto tunnel rispetto al salto termico

La convergenza ad una soluzione dipende dalla strategia di raffreddamento, che regola come la temperatura diminuisce nel corso del processo. Inizialmente, è accettato un numero significativo di soluzioni meno ottimali, tipicamente circa il 50% delle soluzioni peggiori. Durante ogni fase del processo di raffreddamento, la temperatura viene ridotta progressivamente, solitamente di un 10% rispetto al valore precedente.

Il numero di iterazioni per ogni fase di raffreddamento è generalmente compreso tra 1 e 10, permettendo al sistema di esplorare adeguatamente lo spazio delle soluzioni. Al termine del processo, la temperatura raggiunge un livello tale che non vengono più accettate soluzioni peggiori, simile a un algoritmo di **greedy search**. Questo approccio garantisce che, nelle fasi finali, il sistema esplori più selettivamente le configurazioni, dirigendosi verso il minimo globale.

Le principali similitudini tra Simulated Annealing e *Quantum Annealing* riguardano il loro obiettivo comune di trovare il minimo globale di una funzione obiettivo, esplorando diverse configurazioni per evitare minimi locali. Entrambi gli algoritmi seguono un comportamento esponenziale per passare a stati energeticamente sfavorevoli, simile alla distribuzione di Boltzmann, e la temperatura (per il Simulated Annealing) o il campo trasversale (per il *Quantum Annealing*) controllano l'esplorazione dello spazio delle soluzioni.

Le principali differenze tra i due metodi risiedono nel tipo di fluttuazioni utilizzate: il Simulated Annealing si basa su fluttuazioni termiche, mentre il *Quantum Annealing* sfrutta fluttuazioni quantistiche. Nell'approccio classico, la probabilità di superare una barriera energetica dipende dalla temperatura, mentre nell'approccio quantistico la probabilità di attraversarla dipende anche dalle dimensioni della barriera e dalle proprietà quantistiche dei qubit (effetto tunnel).

Il *Quantum Annealing* può risultare più efficace del Simulated Annealing, soprattutto in contesti complessi, grazie alla capacità di attraversare barriere alte tramite fenomeni quantistici, rendendo l'ottimizzazione più veloce ed efficiente su computer quantistici. Tuttavia, la sua efficacia dipende dalle specifiche del problema e dell'implementazione del dispositivo. In ambienti con pochi minimi locali e con spazi di soluzioni di dimensioni relativamente piccole il Simulated Annealing può essere una scelta preferibile.

Capitolo 5: Topologie Hardware e Architettura del Quantum Annealing

5.1 Quantum Annealer

Il **Quantum Annealer** è un tipo specifico di *hardware quantistico* progettato specificamente per risolvere problemi di ottimizzazione, utilizzando i principi della meccanica quantistica.

Il Quantum Annealer agisce evolvendo il sistema lungo una traiettoria determinata dall'Hamiltoniano, in modo che il sistema raggiunga, con alta probabilità, il minimo globale o una soluzione molto vicina ad esso. Questo processo è reso possibile dalla natura parallela e probabilistica delle interazioni quantistiche tra i qubit, che permettono di esplorare simultaneamente molteplici configurazioni di stato.

I problemi di ottimizzazione vengono formulati tramite il modello matematico del **QUBO**, che viene successivamente convertito nel **modello di Ising**, che come abbiamo visto, è una rappresentazione che il quantum annealer è in grado di "trattare" per trovare la soluzione ottimale.

5.2 Distinzione fra Qubit Logico e Qubit Fisico

Fino ad ora abbiamo trattato il **qubit** come una rappresentazione astratta dello stato dello spin di una particelle subatomica. D'ora in avanti è necessaria una distinzione fra **qubit Logico** e **qubit Fisico**.

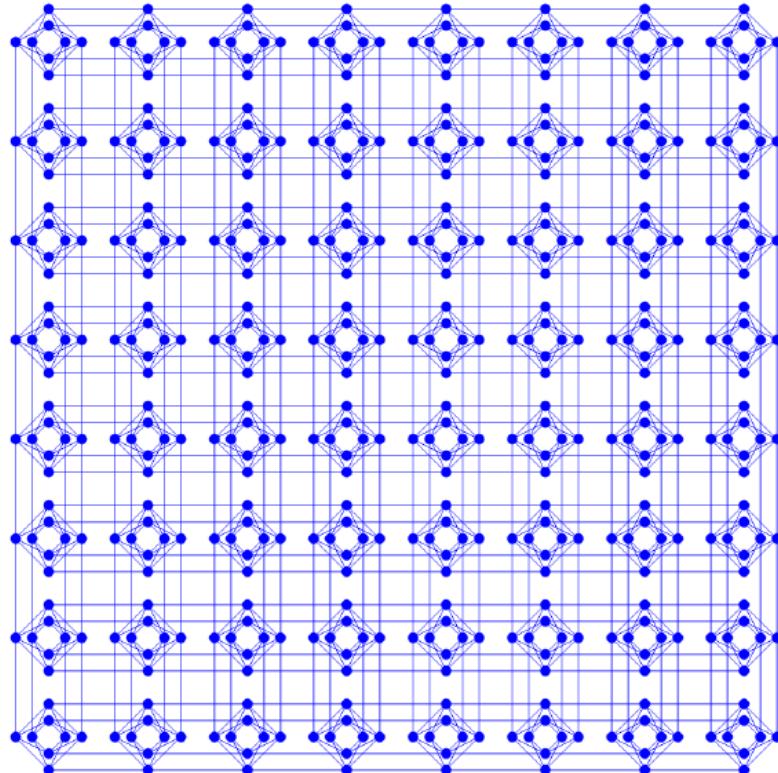
Il *qubit logico* è un'astrazione che rappresenta una variabile di un problema matematico. Nei modelli di Ising o nei QUBO, ogni qubit logico codifica uno stato binario, che può essere +1 o -1, oppure 1 o 0, a seconda della convenzione adottata.

Un *qubit fisico*, invece, è un'entità concreta e fisica, come lo spin di una particella subatomica, che può essere implementata su hardware quantistico.

Perché è stata necessaria questa distinzione? Ogni qubit fisico è soggetto a rumore, decoerenza e imperfezioni che ne compromettono la stabilità e la capacità di mantenere le informazioni. Se un qubit logico fosse rappresentato da un singolo qubit fisico, un errore nel qubit fisico potrebbe compromettere l'informazione contenuta in quel qubit logico. Per preservare le informazioni quantistiche, sono necessari meccanismi di correzione degli errori, mantenimento della coerenza e difesa dai disturbi esterni. Sebbene non ci concentreremo sui dettagli di questi meccanismi, è importante sottolineare che un **singolo qubit logico** viene spesso rappresentato da una **catena di qubit fisici interconnessi**.

Questa tecnica, nota come **embedding**, consente di mappare le interazioni richieste dal modello logico sulla topologia fisica del Quantum Annealer, garantendo così una maggiore robustezza e stabilità nell'esecuzione del calcolo.

5.3 Grafo Chimera



Una rappresentazione del Grafo Chimera

Il **grafo Chimera** è una **struttura topologica** progettata per organizzare e connettere i qubit fisici nei processori quantistici sviluppati da *D-Wave Systems* [20].

Questa architettura riflette le limitazioni hardware dei Quantum Annealer, che non possono implementare connessioni completamente arbitrarie tra i qubit fisici. La topologia Chimera offre un compromesso tra flessibilità e semplicità costruttiva, ottimizzando l'implementazione di interazioni locali e non locali per rappresentare problemi complessi di ottimizzazione.

Nel grafo Chimera, i qubit fisici sono rappresentati come **nodi**, mentre le interazioni tra di essi (denominate "**coupler**") sono rappresentate da archi. Ogni cella del grafo contiene 8 qubit fisici, suddivisi in due gruppi di 4 qubit (configurazione $K_{4,4}$), dove ogni gruppo è completamente connesso internamente, e può interagire con i qubit del gruppo opposto. Le celle sono disposte in una griglia $L \times L$ e le connessioni tra celle adiacenti sono limitate.

Questa configurazione consente di rappresentare sottoproblemi locali altamente connessi all'interno di ciascuna cella. Tuttavia, la connettività limitata tra le celle può rappresentare un ostacolo per problemi con molte interazioni non locali. Ad esempio, per un processore con $L = 4$, il grafo Chimera contiene $N = 8L^2 = 128$ qubit fisici e circa $16L^2 - 4L = 240$ connessioni.

Per mappare un problema del modello di Ising su un grafo Chimera, le interazioni J_{ij} e i campi locali h_i devono essere tradotti in connessioni e pesi compatibili con la struttura del grafo. Tuttavia, le limitazioni della connettività richiedono un processo di embedding, in cui le variabili logiche (s_i) sono rappresentate da più qubit fisici interconnessi.

5.4 Embedding

L'**embedding**, ossia, il processo di **traduzione del modello di Ising al grafo Chimera**, avviene in diverse fasi.

In primo luogo, ogni variabile logica s_i del modello di Ising viene rappresentata da un qubit logico che viene poi mappato da una catena di qubit fisici interconnessi.

Le interazioni J_{ij} del modello devono essere assegnate agli archi del grafo, e se i qubit fisici corrispondenti non sono direttamente collegati, vengono introdotte catene di qubit per mediare l'interazione.

Per garantire che una catena di qubit fisici rappresenti correttamente una singola variabile logica, vengono introdotti vincoli di coerenza. Questi vincoli impongono che tutti i qubit fisici della catena assumano lo stesso valore:

$$E_{\text{catena}} = \sum_{(i,k) \in \text{catena}} J_{\text{catena}} s_i s_k$$

dove J_{catena} è un peso assegnato alle connessioni all'interno della catena.

Infine termini h_i vengono assegnati ai nodi corrispondenti del grafo Chimera, rappresentando campi locali applicati ai qubit fisici.

L'ottimizzazione dell'embedding nel grafo Chimera richiede algoritmi specifici per ridurre il numero di catene necessarie e minimizzare gli errori tra i qubit fisici. Tuttavia, la traduzione del modello di Ising presenta sfide legate alla limitata connettività, che richiede l'uso di catene per simulare connessioni non locali, e all'aumento del numero di qubit fisici, che riduce la capacità del sistema. Inoltre, la sensibilità agli errori dei vincoli di catena può compromettere la rappresentazione delle variabili logiche, e la complessità computazionale del processo deve essere bilanciata con i benefici del calcolo quantistico.

Il grafo Chimera è stato una delle prime architetture pratiche per quantum annealers, permettendo la risoluzione di problemi di ottimizzazione complessi. Tuttavia, ne abbiamo riscontrato diverse limitazioni.

Per superarle, è stato sviluppato il grafo Pegasus, che offre una connettività superiore. Pegasus, con una struttura a $K_{5,5}$, consente una maggiore densità di connessioni tra i qubit, riducendo la necessità di catene di qubit e migliorando la rappresentazione diretta delle interazioni tra variabili logiche. Ciò ottimizza l'uso delle risorse hardware, aumentando l'efficienza del sistema.

5.5 Tool per l'Embedding

Diversi strumenti e algoritmi sono stati sviluppati per trovare embedding efficienti per un dato modello di Ising.

Minorminer [21] è uno strumento open-source progettato per affrontare il problema dell'embedding dei grafi logici nei grafi fisici dei quantum annealer. Disponibile pubblicamente, Minorminer è scritto in *Python* e può essere integrato facilmente in altri framework o applicazioni personalizzate.

La stessa D-Wave offre un'interfaccia software che include strumenti per l'embedding automatico. La **D-Wave API** [22] e il **Software Development Kit (SDK)** [23] permettono ai programmati di modellare, inviare e analizzare problemi ottimizzati per l'hardware quantistico. Permettono anche di simulare i risultati su un computer classico prima di eseguire il problema sull'hardware quantistico, utile per debugging e test.

Capitolo 6: Quantum Annealing per il Problema del Commesso Viaggiatore

6.1 Problema del Commesso Viaggiatore

Il **problema del Commesso Viaggiatore** (*Traveling Salesman Problem, TSP*) è un classico problema di ottimizzazione combinatoria [24]. Esso richiede di determinare il percorso più breve che permette a un commesso viaggiatore di visitare un insieme di città, passando per ciascuna esattamente una volta, per poi tornare al punto di partenza.

Il TSP è noto per la sua rilevanza sia teorica che pratica, ed è classificato come problema *NP-hard*, il che implica che non esistono algoritmi noti in grado di risolverlo in tempo polinomiale per ogni caso possibile (a meno che $P = NP$, ma ricordo che determinarlo è uno dei *problemi del millennio*).

Può essere descritto matematicamente come segue:

- Siano $G = (N, A)$ un grafo completo, $N = \{1, 2, \dots, n\}$ l'insieme delle città e A l'insieme degli archi;
- A ciascun arco $(i; j) \in A$ è associato un costo c_{ij} , che rappresenta la distanza o il peso del collegamento tra le città i e j .

Un **ciclo hamiltoniano** è un percorso in un grafo che visita ciascun nodo esattamente una volta e ritorna al nodo di partenza, formando così un ciclo chiuso. L'obiettivo del TSP è determinare un ciclo hamiltoniano $x = (x_1, x_2, \dots, x_n)$, dove ogni città è visitata esattamente una volta, minimizzando la somma totale dei costi associati al percorso.

Il TSP è **simmetrico** se $c_{ij} = c_{ji} \forall (i; j)$, altrimenti si dice **asimmetrico**.

x_{ij} è la generica variabile binaria del problema. In pratica $x_{ij} = 1$ se l'arco $(i; j)$ appartiene al circuito, altrimenti $x_{ij} = 0$. Una possibile formulazione matematica del problema è:

$$\begin{aligned} \min & \left(\sum_{i=1}^N \sum_{j=1}^N c_{ij} x_{ij} \right), \\ & \sum_{j=1, j \neq i}^N x_{ij} = 1 \quad \forall i \in N, \\ & \sum_{i=1, i \neq j}^N x_{ij} = 1 \quad \forall j \in N, \\ & \sum_{(i,j) \in A, i \in S} \sum_{j \notin S} x_{ij} \geq 1 \quad \forall S \subset N, S \neq 0, S \neq N, \\ & x_{ij} \in \{0, 1\} \quad \forall i, j \in N. \end{aligned}$$

I primi due tipi di vincoli mi assicurano che ogni città sia visitata una sola volta e sono detti **vincoli di assegnazione**. Il primo nello specifico mi dice che ogni città deve essere lasciata una sola volta, il secondo che ogni città deve essere raggiunta una sola volta.

Il terzo tipo di vincoli evita cicli interni che non coprono tutte le città, garantendo un unico ciclo hamiltoniano e l'assenza di sottocicli. Sono detti **vincoli di connessione**. In particolare essi definiscono che, comunque si scelga un sottoinsieme proprio di nodi S , deve esistere almeno un arco che colleghi un nodo di S con un nodo non appartenente a S .

6.2 TSP nella forma QUBO

Per risolvere il problema tramite il *Quantum Annealing*, è necessario prima rappresentarlo in forma **QUBO**. Tuttavia, il TSP, nella sua formulazione classica, è difficile da esprimere in questa forma.

Questo limite deriva dalla difficoltà di rappresentare esplicitamente i **vincoli di connessione**. La modellazione di tali vincoli spesso richiede espressioni matematiche più complesse di quelle quadratiche e comporta un aumento significativo del numero di variabili, rendendo il problema inefficiente da risolvere.

Nei metodi classici, come l'ottimizzazione lineare, questi vincoli vengono gestiti iterativamente durante il processo di ottimizzazione. Nel caso del QUBO, invece, tutti i vincoli devono essere incorporati direttamente nella funzione obiettivo sotto forma di **penalità e/o ricompense**.

Rappresentare in modo efficace vincoli complessi, come quelli relativi agli archi nel TSP, risulta quindi impraticabile senza introdurre semplificazioni o approssimazioni. Per superare queste difficoltà, è possibile adottare una formulazione alternativa del problema, basata **sull'ordine di visita delle città**.

In questa nuova rappresentazione, il problema viene modellato utilizzando variabili binarie x_{ip} che assumono il valore 1 se una città i occupa una posizione p lungo il percorso e 0 altrimenti. La funzione obiettivo e i vincoli del problema si possono così esprimere nella forma seguente:

$$\begin{aligned} \min & \left(\sum_{i=1}^N \sum_{j=1}^N \sum_{p=1}^N c_{ij} \cdot x_{ip} \cdot x_{j(p+1)} \right), \\ & \sum_{p=1}^N x_{ip} = 1 \quad \forall i \in N, \\ & \sum_{i=1}^N x_{ip} = 1 \quad \forall p = \{1, \dots, n\}, \\ & x_{ip} \in \{0, 1\} \quad \forall i \in N, \forall p = \{1, \dots, n\}. \end{aligned}$$

dove:

- c_{ij} rappresenta la distanza (il costo) tra le città i e j ;
- Il primo tipo di vincoli assicura che ogni città i sia visitata esattamente una volta e occupi una posizione unica nel percorso;
- Il secondo tipo di vincoli garantisce che ogni posizione p nel percorso sia assegnata a una sola città.

Inoltre, per completare il ciclo, si impone $x_{j,n+1} = x_{j,1}$ per tornare al punto di partenza.

Per passare dal problema del TSP alla formulazione QUBO, costruiamo una matrice Q che incorpora la funzione obiettivo e i vincoli sotto forma di penalità e ricompense.

La **ricompensa** R è un numero positivo, il cui valore deve essere maggiore della distanza più grande, mentre la **penalità** P è un numero positivo, che deve essere maggiore o uguale alla ricompensa. Questo requisito garantisce che le penalità associate alla violazione dei vincoli prevalgano sui contributi positivi della ricompensa nella funzione obiettivo.

Sebbene la ricompensa sia un valore positivo, essa viene successivamente sottratta nella formulazione matematica, contribuendo così a bilanciare l'influenza del termine di ottimizzazione e dei vincoli.

La matrice Q del QUBO viene aggiornata con i costi effettivi se due città i e j occupano posizioni adiacenti:

$$Q_{ip,jq} = c_{ij} \quad \text{se } i \neq j \text{ e } q = p + 1$$

Se una città i appare in due posizioni diverse p e q , questa combinazione è vietata:

$$Q_{ip,jq} = P \quad \text{se } i = j \text{ e } p \neq q$$

Se due città i e j occupano la stessa posizione p , questa combinazione è vietata:

$$Q_{ip,jq} = P \quad \text{se } i \neq j \text{ e } p = q$$

Infine per evitare che la soluzione imponga la stessa città su più posizioni, togliamo la ricompensa sulla diagonale:

$$Q_{ip,jq} = -R \quad \text{se } i = j \text{ e } p = q$$

Se non rientra in nessuno dei precedenti casi:

$$Q_{ip,jq} = 0$$

Riassumendo:

$$Q_{ip,jq} = \begin{cases} Q_{ip,jq} = c_{ij} & \text{se } i \neq j \text{ e } q = p + 1; \\ Q_{ip,jq} = P & \text{se } i = j \text{ e } p \neq q; \\ Q_{ip,jq} = P & \text{se } i \neq j \text{ e } p = q; \\ Q_{ip,jq} = -R & \text{se } i = j \text{ e } p = q; \\ Q_{ip,jq} = 0 & \text{altrimenti} \end{cases}$$

Non entreremo nel dettaglio della scelta dei termini P ed R , in quanto questa parte del processo dipende da considerazioni specifiche che esulano dallo scopo di questo lavoro. Tuttavia, è importante sottolineare che, generalmente, il calcolo di questi termini si basa sulla dimensione del problema e sui pesi delle connessioni del grafo.

6.3 Progettazione Algoritmo

La **progettazione di un algoritmo di Quantum Annealing** segue una serie di passi strutturati che permettono di tradurre un problema di ottimizzazione in un modello risolvibile tramite il quantum annealer.

La prima fase consiste nella **definizione del problema**. Ogni problema deve essere espresso come una funzione obiettivo che rappresenti un costo da minimizzare o una qualità da massimizzare. Il *Quantum Annealing* è progettato per risolvere problemi di minimo, quindi se si ha un problema di massimo si applica questa formula alla funzione obiettivo:

$$\max(f(x)) = -\min(-f(x))$$

I vincoli del problema rimangono invariati durante questa conversione. Se sono presenti penalità associate ai vincoli, anch'esse restano additive nella funzione obiettivo trasformata. Le ricompense restano sottrattive.

Successivamente, si passa alla **rappresentazione della soluzione**. Nel *Quantum Annealing* è determinata dal numero di qubit necessari per modellare tutte le possibili configurazioni del problema. Ogni qubit rappresenta una decisione binaria o uno stato del sistema. Bisogna porre particolare attenzione a questo step, perché è necessario per rendere più facile la costruzione della matrice QUBO.

La fase seguente riguarda la **definizione dei vincoli**, che deve garantire che le soluzioni candidate rispettino le regole del problema. Questi vincoli vengono tradotti in termini matematici attraverso penalizzazioni e/o ricompense inseriti nella funzione obiettivo.

Una volta definiti i vincoli, è necessario **costruire la matrice QUBO**, che rappresenta matematicamente il problema sotto forma di una matrice quadrata simmetrica. Gli elementi diagonali della matrice descrivono i costi o i premi associati alle decisioni individuali, mentre gli elementi non diagonali rappresentano le interazioni tra diverse decisioni.

Dopo aver costruito la matrice QUBO, il problema viene **convertito nel modello di Ising**, che è direttamente interpretabile dai quantum annealer. La conversione mantiene la struttura e il significato matematico del problema.

Il passo successivo è l'**embedding del modello di Ising nel grafo Chimera** o in una topologia hardware compatibile. Questo processo richiede un numero maggiore di qubit fisici rispetto a quelli logici, ma strumenti software specializzati semplificano questa operazione.

Una volta completato l'embedding, il quantum annealer esegue la **determinazione del minimo di energia** attraverso il processo di *Quantum Annealing*. È qui che il sistema viene inizializzato nello stato fondamentale di un Hamiltoniano iniziale e successivamente evoluto verso lo stato fondamentale dell'Hamiltoniano del problema. L'adiabaticità del processo garantisce che il sistema rimanga nello stato di energia minima, a meno di interruzioni causate da decoerenza o transizioni non adiabatiche.

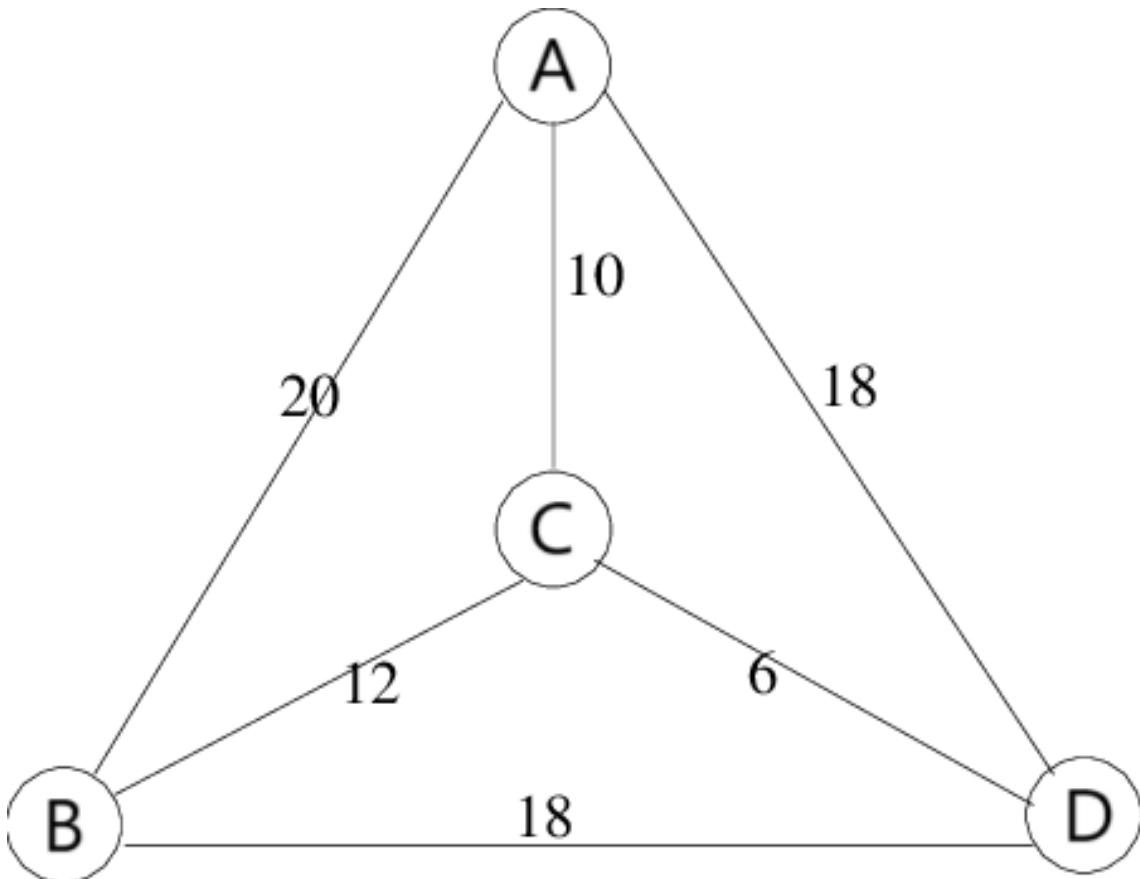
Infine, si giunge all'**interpretazione della soluzione**, in cui lo stato finale del sistema viene decodificato per estrarre la configurazione ottimale delle variabili originali. La soluzione restituita è nel modello di Ising e viene poi convertita in forma QUBO in modo che sia più facile da interpretare.

Riassumendo, i passaggi per la progettazione di un algoritmo [25] sono:

1. **Definizione del Problema;**
2. **Rappresentazione della Soluzione;**
3. **Definizione dei Vincoli;**
4. **Costruzione della matrice QUBO;**
5. **Conversione nel Modello di Ising;**
6. **Embedding con grafico Chimera;**
7. **Determinazione del minimo di energia;**
8. **Interpretazione della soluzione.**

6.4 Esempio pratico di risoluzione di un TSP

Per illustrare l'applicazione del *Quantum Annealing*, consideriamo un esempio pratico basato su un problema del Commesso Viaggiatore simmetrico (rappresentato nella figura sotto). La rappresentazione grafica del problema da risolvere è la seguente:



La città A deve essere allo stesso tempo, la città di partenza e la città di arrivo.

6.4.1 Passo 1: Definizione del Problema

Come discusso in precedenza, la formulazione classica del TSP, in cui l'obiettivo è determinare direttamente gli archi che compongono il percorso ottimale, risulta complessa da convertire in una forma compatibile con il modello QUBO. Pertanto proponiamo una formulazione alternativa del problema, focalizzandoci sull'**ordine delle città da visitare**. In questa rappresentazione, l'obiettivo è determinare la sequenza ottimale di città che minimizza il costo totale del percorso, rispettando i vincoli di unicità e ciclicità.

6.4.2 Passo 2: Rappresentazione della Soluzione

In un problema di TSP con N città, è necessario garantire che ogni città sia assegnata a una posizione specifica nel percorso. Per rappresentare questa relazione nella formulazione QUBO, sono richiesti N^2 qubit logici: ciascuna città può occupare N posizioni nel percorso e ogni combinazione è rappresentata da un qubit.

Nel nostro problema vi sono 4 città. Teoricamente, sarebbero necessari 16 qubit logici per rappresentare tutte le possibili assegnazioni. Tuttavia, è possibile ridurre il numero di qubit richiesti sfruttando le proprietà del problema e introducendo alcune ottimizzazioni.

Poiché in un TSP la città di partenza e di arrivo sono la stessa (in questo caso è A), è sufficiente escluderla dalla rappresentazione. Questo riduce il problema a determinare l'ordine delle altre $N - 1$ città. Di conseguenza, con 4 città, il numero di qubit logici necessari si riduce a:

$$(N - 1)^2 = 3^2 = 9$$

Questa riduzione sfrutta la simmetria del problema e semplifica il modello, pur preservando la correttezza della rappresentazione e la capacità di determinare il percorso ottimale.

City	B	B	B	C	C	C	D	D	D
Position	2	3	4	2	3	4	2	3	4

6.4.3 Passo 3: Definizione dei Vincoli

Abbiamo già visto come si formulano i vincoli del TSP con la formulazione basata sull'ordine delle città. Essi possono essere riassunti come segue:

1. **Ogni città deve essere visitata una sola volta;**
2. **Ogni posizione del percorso deve essere occupata da una sola città;**
3. **La distanza tra città consecutive deve influire sul costo totale del percorso.**

La formalizzazione matematica di questi vincoli è la seguente:

$$\begin{aligned} & \min \left(\sum_{i \in \{B,C,D\}} \sum_{j \in \{B,C,D\}} \sum_{p=2}^4 c_{ij} \cdot x_{ip} \cdot x_{ip+1} \right), \\ & \sum_{p=2}^4 x_{ip} = 1 \quad \forall i \in \{B,C,D\} \\ & \sum_{i \in \{B,C,D\}} x_{ip} = 1 \quad \forall p = \{2, 3, 4\} \\ & x_{ip} \in \{0, 1\} \quad \forall i \in \{B, C, D\}, \forall p = \{2, 3, 4\}. \end{aligned}$$

6.4.4 Passo 4: Costruzione della matrice QUBO

Con queste semplificazioni, possiamo esprimere Q nel seguente modo:

$$Q_{ip,jq} = \begin{cases} Q_{ip,jq} = c_{ij} & \text{se } i \neq j \text{ e } q = p + 1; \\ Q_{ip,jq} = P & \text{se } i = j \text{ e } p \neq q; \\ Q_{ip,jq} = P & \text{se } i \neq j \text{ e } p = q; \\ Q_{ip,jq} = c_{ij} - R & \text{se } i = j \text{ e } p = q \text{ con } p = 2 \text{ o } p = N; \\ Q_{ip,jq} = -R & \text{se } i = j \text{ e } p = q \text{ con } p \neq 2 \text{ e } p \neq N; \\ Q_{ip,jq} = 0 & \text{altrimenti} \end{cases}$$

Applicandola al nostro problema, troviamo la seguente matrice:

Il TSP è simmetrico e di conseguenza pure questa matrice lo è. Questo implica che i valori nella matrice siano speculari rispetto alla diagonale, ovvero i riquadri grigi sono identici per le coppie di città simmetriche.

6.4.5 Passo 5: Conversione nel Modello di Ising

Il modello QUBO viene tradotto in un modello di Ising per l'implementazione sul quantum annealer. Applicando l'algoritmo di conversione otteniamo la seguente matrice:

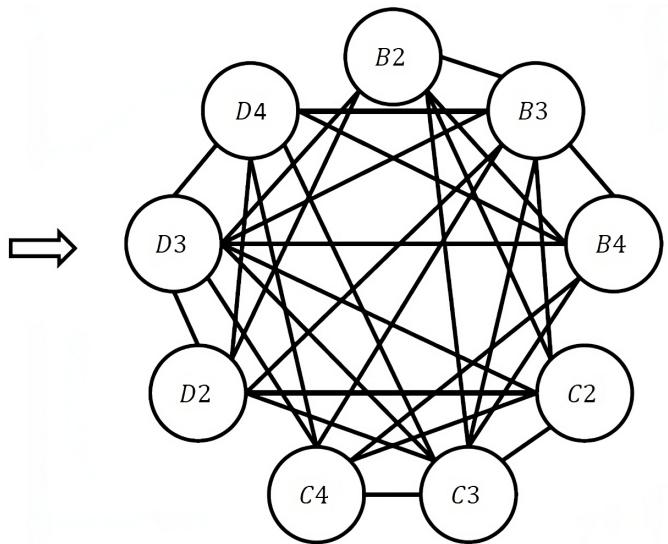
	<i>B2</i>	<i>B3</i>	<i>B4</i>	<i>C2</i>	<i>C3</i>	<i>C4</i>	<i>D2</i>	<i>D3</i>	<i>D4</i>
<i>B2</i>	35	7,5	7,5	7,5	3	0	7,5	4,5	0
<i>B3</i>		32,5	7,5	3	7,5	3	4,5	7,5	4,5
<i>B4</i>			35	0	3	7,5	0	4,5	7,5
<i>C2</i>				27	7,5	7,5	7,5	1,5	0
<i>C3</i>					26,5	7,5	1,5	7,5	1,5
<i>C4</i>						27	0	1,5	7,5
<i>D2</i>							32,5	7,5	7,5
<i>D3</i>								29,5	7,5
<i>D4</i>									32,5

6.4.6 Passo 6: Embedding con grafico Chimera

Il modello di Ising può essere rappresentato come un grafo. In questa rappresentazione:

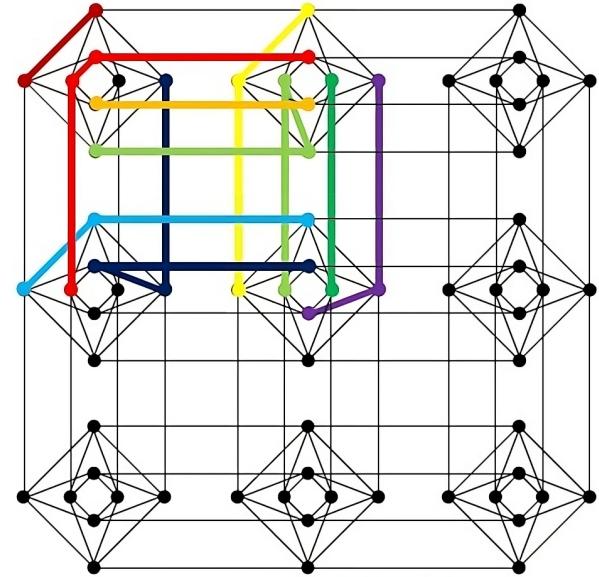
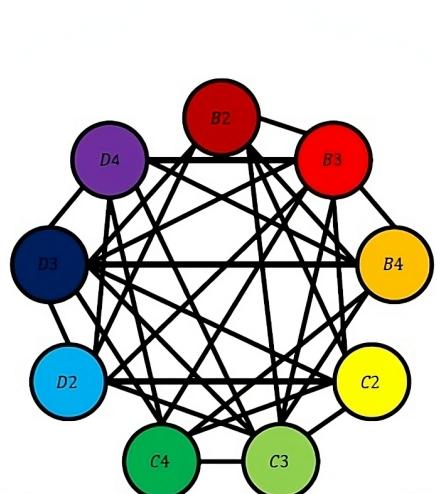
- I vertici del grafo corrispondono ai qubit logici;
- Un arco viene introdotto tra due vertici se il valore corrispondente nella matrice Q (nella forma di Ising) è diverso da zero.

	<i>B2</i>	<i>B3</i>	<i>B4</i>	<i>C2</i>	<i>C3</i>	<i>C4</i>	<i>D2</i>	<i>D3</i>	<i>D4</i>
<i>B2</i>	35	7,5	7,5	7,5	3	0	7,5	4,5	0
<i>B3</i>		32,5	7,5	3	7,5	3	4,5	7,5	4,5
<i>B4</i>			35	0	3	7,5	0	4,5	7,5
<i>C2</i>				27	7,5	7,5	7,5	1,5	0
<i>C3</i>					26,5	7,5	1,5	7,5	1,5
<i>C4</i>						27	0	1,5	7,5
<i>D2</i>							32,5	7,5	7,5
<i>D3</i>								29,5	7,5
<i>D4</i>									32,5



Questo grafico ci rende più facile la comprensione del meccanismo di **embedding**. N qubit logici richiedono N^2 qubit fisici nel caso peggiore. Il nostro modello di Ising deve essere mappato sulla topologia fisica del grafo Chimera.

Utilizzando dei tool specifici è stato possibile ottenere un embedding che richiede 27 qubit fisici per rappresentare 9 qubit logici. Considerando che nel caso peggiore sarebbe stato necessario l'uso di 81 qubit fisici, il risultato ottenuto è soddisfacente.



6.4.7 Passo 7: Determinazione del Minimo di Energia

Il quantum annealer risolve il problema determinando lo stato fondamentale (minimo energetico) dell'Hamiltoniano totale. Il *Quantum Annealing* sfrutta l'effetto tunnel per superare barriere energetiche, evitando di rimanere bloccato nei minimi locali e trovando il minimo globale.

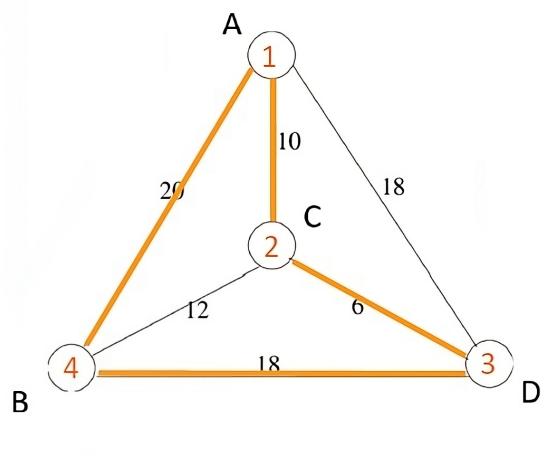
6.4.8 Passo 8: Interpretazione della soluzione

Una volta identificata la configurazione dei qubit che corrisponde all'energia minima, il passo successivo consiste nel tradurre questa configurazione nella soluzione del problema originale.

Nel caso del TSP, ciò implica determinare il percorso ottimale, ossia la sequenza di città che minimizza la distanza totale, risolvendo così il problema di ottimizzazione.

Questa fase richiede l'interpretazione della configurazione dei qubit in termini di variabili logiche, seguendo la mappatura stabilita durante l'embedding, per ricostruire il percorso che rappresenta la soluzione migliore.

		-1	-1	1	1	-1	-1	-1	1	-1
		B2	B3	B4	C2	C3	C4	D2	D3	D4
-1	B2	35	7,5	7,5	3	0	7,5	4,5	0	
-1	B3		32,5	7,5	3	7,5	3	4,5	7,5	4,5
1	B4			35	0	3	7,5	0	4,5	7,5
1	C2				27	7,5	7,5	7,5	1,5	0
-1	C3					26,5	7,5	1,5	7,5	1,5
-1	C4						27	0	1,5	7,5
-1	D2							32,5	7,5	7,5
1	D3								29,5	7,5
-1	D4									32,5



Soluzione del Problema

Ricostruita la soluzione per il modello di Ising, viene poi presentata in forma QUBO.

La soluzione al nostro problema specifico è: $A \rightarrow C \rightarrow D \rightarrow B \rightarrow A$. In formulazione QUBO si traduce in:

$$x = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Capitolo 7: Implementazione dell’Algoritmo di Grover attraverso il Quantum Annealing

7.1 Introduzione all’Algoritmo di Grover

L’**algoritmo di Grover**, o algoritmo di ricerca quantistica, è una tecnica di ricerca quantistica ideata per identificare un elemento specifico in un database non ordinato. Fu ideato da *Lov Grover* nel 1996 [26]. Questo algoritmo è significativamente più efficiente rispetto ai metodi classici, poiché riduce il numero di query necessarie da $O(N)$, in media, a $O(\sqrt{N})$, dove N è la dimensione del database. Si dice appunto che l’algoritmo di Grover fornisce un’**accelerazione quadratica**, che deriva dal fatto che $O(\sqrt{N})$ cresce molto più lentamente di N , offrendo un vantaggio significativo, specialmente per database molto grandi.

L’algoritmo è ideale per problemi che possono essere ridotti a una ricerca in un database non strutturato, come l’inversione di una funzione e la soluzione di equazioni con risultati multipli.

Come altri algoritmi quantistici, l’algoritmo di Grover inizia con una sovrapposizione uguale di tutti i possibili stati partendo da un registro composto da n qubit. Questo implica che una stessa ampiezza pari a $\frac{1}{\sqrt{2^n}}$ è associata ad ogni possibile configurazione di qubit nel sistema e corrisponde ad una probabilità $\frac{1}{2^n}$ che il sistema sia in uno di quei 2^n stati.

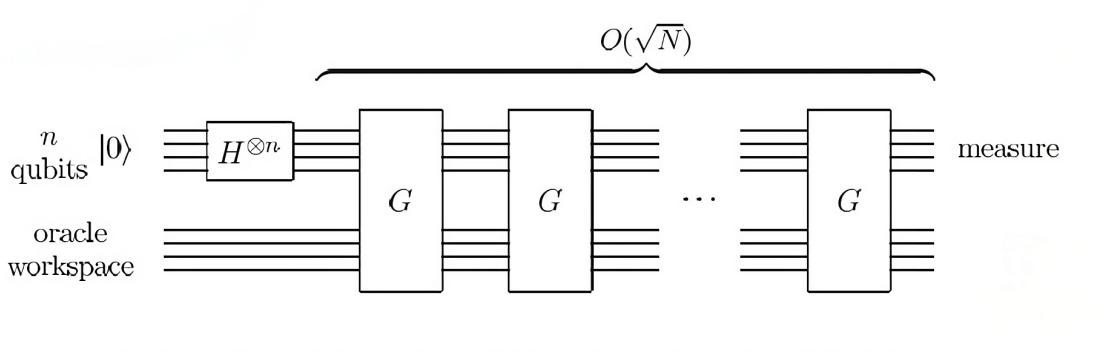
L’algoritmo di Grover sfrutta l’**amplificazione di ampiezza**, cioè una proprietà caratteristica delle ampiezze quantistiche. In pratica si ha uno sfasamento selettivo di uno stato di un sistema quantistico. Uno sfasamento di π equivale a moltiplicare l’ampiezza di quello stato per -1 . Sebbene il segno dell’ampiezza cambi, la probabilità associata allo stato rimane invariata, in quanto dipende solo dal modulo dell’ampiezza.

Le successive trasformazioni sfruttano questa inversione di fase per amplificare gradualmente l'ampiezza dello stato desiderato, aumentando così la probabilità di osservarlo durante una misurazione. Questo approccio è reso possibile dal fatto che le ampiezze quantistiche contengono sia informazioni sulla probabilità che sulla fase dello stato, consentendo manipolazioni che vanno oltre ciò che è realizzabile nei sistemi classici.

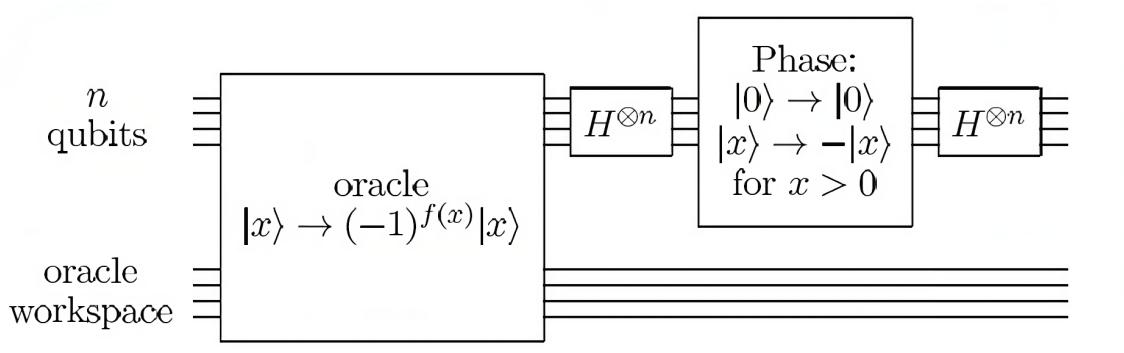
L'algoritmo di ricerca opera come segue [27]:

1. L'inizializzazione avviene ponendo gli n qubit di un registro nello stato di sovrapposizione tramite la **trasformazione di Hadamard**: $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$.
2. Si applica un'**iterazione di Grover** G , composta da:
 - (a) *Applicazione dell'oracolo* O .
 - (b) *Trasformata di Hadamard* $H^{\otimes n}$.
 - (c) *Fase condizionale*, dove gli stati base $|x\rangle$ ricevono una fase di -1 tranne per $|0\rangle$: $|x\rangle \rightarrow -(-1)^{\delta_{x0}}|x\rangle$.
 - (d) *Un'altra trasformata di Hadamard* $H^{\otimes n}$.

Ogni passo può essere implementato in modo efficiente su un computer quantistico.



Circuito schematico che rappresenta l'algoritmo di Grover.



Circuito per l'iterazione di Grover G .

7.2 Porta di Hadamard

La **trasformazione di Hadamard** è una delle operazioni fondamentali nell’informatica quantistica e gioca un ruolo cruciale nella creazione di stati di sovrapposizione e nella manipolazione delle ampiezze di probabilità. La **porta di Hadamard** è la realizzazione pratica di questa trasformazione per un singolo qubit all’interno di un circuito quantistico. È una porta unitaria a un qubit e si rappresenta graficamente con il simbolo H . Il simbolo $H^{\otimes n}$ indica che stiamo usando n porte di Hadamard.

Come già visto, la trasformazione di Hadamard può essere espressa come:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

Quando si considera un registro di n qubit, la trasformazione di Hadamard agisce su ogni qubit individualmente, e si rappresenta come il prodotto tensoriale (vedi **il quarto postulato della meccanica quantistica**) di n matrici di Hadamard:

$$H^{\otimes n} = H \otimes H \otimes \cdots \otimes H$$

Nell’algoritmo di Grover, la porta di Hadamard inizializza il registro di n qubit in una sovrapposizione uniforme di tutti gli stati possibili, rappresentando un punto di partenza per il processo di amplificazione.

Durante le iterazioni di Grover, la porta di Hadamard è utilizzata per trasformare gli stati base in stati in cui è più facile applicare operazioni come la riflessione rispetto alla media.

Alla fine di un’iterazione di Grover, la porta di Hadamard può essere usata per riportare il sistema in un formato leggibile prima della misurazione.

Da un punto di vista fisico, la porta di Hadamard può essere vista come una rotazione di un qubit nello spazio delle fasi, che cambia la rappresentazione della funzione d’onda del sistema quantistico. È cruciale per sfruttare la natura probabilistica e interferenziale dei qubit.

Da un punto di vista matematico, la porta di Hadamard trasforma lo stato quantistico dalla **base computazionale** alla **base di Fourier** e viceversa. La base di Fourier è un insieme di vettori che rappresentano stati periodici o oscillazioni sinusoidali con diverse frequenze e fasi. In un contesto quantistico, è una base alternativa per descrivere stati quantistici, in cui ogni vettore è una combinazione lineare di stati base con coefficienti complessi che variano in modo ciclico (esponenziali complesse). Serve a decomporre informazioni nei loro componenti di fase e frequenza, utile per analizzare e manipolare segnali periodici o stati quantistici.

7.3 Oracolo Quantistico

Un **oracolo** è un'entità teorica o una funzione utilizzata nei modelli di calcolo per fornire risposte immediate a domande specifiche. Si tratta di uno strumento concettuale che consente di analizzare algoritmi e la loro complessità, assumendo che l'oracolo possa risolvere problemi o verificare ipotesi in modo istantaneo, senza entrare nei dettagli del processo computazionale necessario per produrre tali risposte.

Nel contesto della computazione classica, un oracolo è concepito come un'entità *ideale*, priva di costo computazionale, che fornisce risposte binarie, come "sì" o "no", in relazione a un dato input. Tuttavia, la realizzazione pratica di un oracolo classico è estremamente complessa, se non addirittura impossibile, poiché richiederebbe capacità di calcolo illimitate o l'accesso a conoscenze predefinite.

Nella computazione quantistica, il concetto di oracolo assume una forma più concreta grazie all'uso delle operazioni quantistiche. Un **oracolo quantistico** può essere implementato utilizzando **trasformazioni unitarie** che agiscono su stati quantistici. In particolare, le operazioni che modificano la fase di un qubit possono essere sfruttate per rappresentare una funzione oracolo.

Questa implementazione consente di codificare informazioni su un problema all'interno di una funzione unitaria che opera su uno spazio di Hilbert, offrendo la possibilità di integrare l'oracolo all'interno di algoritmi quantistici, come l'algoritmo di Grover.

Come input per l'algoritmo di Grover, supponiamo di avere una funzione $f : \{0, 1\}^N \rightarrow \{0, 1\}$. Nell'analogia del "database non strutturato", il dominio rappresenta gli indici di un database e $f(x) = 1$ se e solo se i dati a cui punta x soddisfano il criterio di ricerca. Inoltre, supponiamo che solo un indice soddisfi $f(x) = 1$ e chiamiamo questo indice β . Il nostro obiettivo è identificare β . Quindi:

$$f(x) = \begin{cases} 1 & \text{se } x = \beta, \\ 0 & \text{se } x \neq \beta. \end{cases}$$

L'oracolo quantistico è implementato come operatore unitario U_f , che agisce su uno stato quantistico $|x\rangle$ secondo la regola:

$$U_f|x\rangle \otimes |q\rangle = |x\rangle|q \oplus f(x)\rangle$$

Dove:

- x rappresenta un elemento del database;
- q è un qubit ausiliario;
- $f(x)$ è una funzione booleana che restituisce 1 per l'elemento desiderato e 0 per gli altri;

- \oplus è l'operazione XOR;
- \otimes è il prodotto tensore.

Nell'algoritmo di Grover, l'oracolo è progettato per identificare un elemento specifico del database β :

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle$$

Questa operazione inverte la fase dello stato associato all'elemento desiderato ($|\beta\rangle$), lasciando inalterati gli altri.

L'oracolo nell'algoritmo di Grover ha il compito di "*marcare*" le soluzioni del problema di ricerca. Questo avviene modificando la fase degli stati quantistici che rappresentano le soluzioni. Consideriamo un problema di ricerca su N elementi con M soluzioni: su un computer quantistico, è sufficiente applicare l'oracolo $O(\sqrt{N/M})$ volte per individuare una soluzione. Tuttavia, descrivere l'oracolo senza spiegare come viene effettivamente implementato può sembrare astratto e controintuitivo. A prima vista, sembra che l'oracolo conosca già la risposta al problema di ricerca. Qual è, dunque, il beneficio di un algoritmo di ricerca quantistica che si basa su questo tipo di consultazione?

La chiave per comprendere l'utilità dell'oracolo sta nel distinguere due concetti: riconoscere una soluzione e conoscerla a priori. [28]

Un esempio semplice per illustrare questa distinzione è il problema della fattorizzazione. Supponiamo di avere un numero intero m che sappiamo essere il prodotto di due numeri primi p e q . Per determinare p e q su un computer classico, il metodo ovvio è testare tutti i numeri da 2 a \sqrt{m} per trovare il divisore più piccolo. Questo approccio comporta una serie di divisioni di prova (**trial divisions**), ciascuna delle quali verifica se un dato numero divide m esattamente. Quando viene trovato il fattore minore, il fattore maggiore può essere determinato dividendo m per esso.

In un algoritmo classico, questo processo richiede circa \sqrt{m} divisioni di prova. Al contrario, l'algoritmo di Grover consente di velocizzare significativamente questa ricerca. Utilizzando un oracolo quantistico che verifica se un dato x divide m esattamente, possiamo applicare l'algoritmo di ricerca quantistica per identificare il divisore più piccolo in $O(m^{1/4})$ consultazioni dell'oracolo. Questo rappresenta un miglioramento quadratico rispetto al metodo classico. Per far funzionare l'algoritmo, è necessario costruire un circuito efficiente che implementi l'oracolo. Si inizia definendo una funzione:

$$f(x) = \begin{cases} 1 & \text{se } x \bmod m = 0, \\ 0 & \text{altrimenti.} \end{cases}$$

La funzione $f(x)$ indica se una divisione di prova ha successo. Grazie alla **computazione reversibile**, che consente di preservare le informazioni durante ogni trasformazione e garantisce l'invertibilità delle operazioni tramite l'aggiunta di registri ausiliari e modifiche al design delle porte logiche, è possibile progettare un circuito classico reversibile che trasforma (x, q) in $(x, q \oplus f(x))$, dove q è un registro di output inizialmente impostato a 0.

Questo circuito, preservando il legame biunivoco tra input e output caratteristico della computazione reversibile, può essere tradotto direttamente in un circuito quantistico che opera sui registri $|x\rangle|q\rangle$, producendo $|x\rangle|q \oplus f(x)\rangle$, come richiesto dall'oracolo. Non ci soffermeremo ulteriormente sulla computazione reversibile per una questione di brevità.

Quindi l'oracolo è un componente operativo costruito ad *hoc* per problemi specifici, come la ricerca nell'algoritmo di Grover. La sua implementazione è legata al problema che si vuole risolvere ed è uno strumento essenziale per sfruttare i vantaggi dell'interferenza quantistica.

Alla fine l'oracolo può essere concepito come una **scatola nera** o come una **query**, in quanto agisce come un'entità che fornisce risposte definite a una determinata domanda.

L'esempio della fattorizzazione è concettualmente interessante, ma non pratico: ci sono algoritmi classici per la fattorizzazione che funzionano molto più velocemente della ricerca attraverso tutti i possibili divisori, ma soprattutto non compete con l'algoritmo di Shor. La complessità dell'algoritmo di Shor è $O(\log(N)^3)$, che è molto più veloce rispetto a qualsiasi algoritmo classico e a Grover (per numeri di grandi dimensioni). Tuttavia, questo esempio, illustra il modo generale in cui può essere applicato l'algoritmo di ricerca quantistica.

7.4 Iterazione di Grover

Avendo un registro di n qubit, è possibile applicare una **porta di Hadamard** a ciascun qubit per portarli in uno stato di sovrapposizione. Così viene inizializzato il nostro algoritmo. La sequenza di operazioni successive viene definita **iterazione di Grover**.

L'iterazione di Grover [29] è un'operazione che viene ripetuta più volte dall'algoritmo e che può essere formalizzata nel seguente modo:

$$G = (2|\psi\rangle\langle\psi| - Id)O$$

dove:

- ψ è lo stato quantistico attuale, che rappresenta la sovrapposizione di tutti gli stati possibili del sistema;
- Id è l'operatore identità;
- O è l'operatore oracolo che agisce sullo stato quantistico per identificare la soluzione del problema.

Possiamo interpretarla anche come una rotazione in uno spazio bidimensionale. Consideriamo due stati normalizzati:

$$|\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_{x \notin S} |x\rangle \quad |\beta\rangle = \frac{1}{\sqrt{M}} \sum_{x \in S} |x\rangle$$

dove S è l'insieme delle soluzioni. Dato che $x = \sum_{x \notin S} |x\rangle + \sum_{x \in S} |x\rangle$ e applicando la trasformazione di Hadamard, si ha che lo stato iniziale può essere espresso come:

$$|\psi\rangle = \sqrt{\frac{N-M}{N}} |\alpha\rangle + \sqrt{\frac{M}{N}} |\beta\rangle$$

G agisce come una rotazione in questo piano di due dimensioni. Ogni volta che applichiamo l'oracolo e l'operazione di diffusione $2|\psi\rangle\langle\psi| - Id$, otteniamo delle riflessioni successive, che alla fine ci avvicinano sempre di più allo stato $|\beta\rangle$, cioè alla soluzione del problema.

In pratica l'iterazione di Grover funziona così:

1. Viene applicato l'oracolo, che "segna" (modificando la fase di π e quindi moltiplicando per -1) lo stato che rappresenta la soluzione del problema, lasciando invariati gli altri stati.
2. Viene applicata una trasformazione di Hadamard, per trasformare lo stato quantistico dalla base computazionale alla base di Fourier. Per amplificare la probabilità dello stato target, serve calcolare e applicare l'inversione rispetto alla media. Questo calcolo avviene naturalmente nella base di Fourier, dove la trasformazione è più semplice da eseguire.
3. Successivamente, viene applicata una riflessione rispetto allo *stato medio* (tramite l'operatore di diffusione: $2|\psi\rangle\langle\psi| - Id$), che aumenta la probabilità di misurare la soluzione corretta, aumentando l'ampiezza dello stato marcato dall'oracolo. Invece gli stati non appartenenti alla soluzione vengono ridotti di ampiezza.
4. Dopo la riflessione rispetto alla media, la trasformazione di Hadamard viene applicata una seconda volta per ritornare dalla base di Fourier alla base computazionale. Questo consente di riportare lo stato quantistico alla base originale in cui i risultati possono essere misurati e conservare gli effetti delle precedenti operazioni. Dopo di che si reitera interamente il processo per R volte.
5. Dopo un certo numero di iterazioni R , la probabilità di trovare la soluzione aumenta e, al termine dell'algoritmo, si misura lo stato quantistico, ottenendo la soluzione del problema.

Per determinare quante iterazioni dell'algoritmo di Grover siano necessarie per far convergere lo stato quantistico iniziale $|\psi\rangle$ verso la soluzione $|\beta\rangle$, occorre analizzare la dinamica di evoluzione dello stato nel corso delle iterazioni [30].

Lo stato iniziale del sistema è definito come:

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle,$$

dove $|\alpha\rangle$ rappresenta lo stato di non-soluzione, e $|\beta\rangle$ è lo stato corrispondente alla soluzione del problema. L'algoritmo di Grover utilizza una serie di operazioni per "ruotare" lo stato iniziale verso lo stato $|\beta\rangle$. La rotazione angolare necessaria per portare lo stato da $|\psi\rangle$ a $|\beta\rangle$ è data da un angolo di rotazione $\theta = \arccos(\sqrt{\frac{M}{N}})$, dove M è il numero di soluzioni nel problema di ricerca e N è il numero totale di possibili risultati. Essendo, sia N che M , due numeri interi positivi, possiamo dire che θ è compreso tra 0 e $\frac{\pi}{2}$ (compresi).

La funzione $CI(x)$ restituisce l'intero più vicino al valore reale x , con la convenzione di arrotondare per difetto in caso di valori medi (es. $CI(3.5) = 3$).

Possiamo esprimere il numero di rotazioni R necessarie per avvicinare lo stato $|\psi\rangle$ a $|\beta\rangle$ come:

$$R = \text{CI} \left(\frac{\arccos(\sqrt{\frac{M}{N}})}{\theta} \right).$$

Ripetendo l'iterazione di Grover un numero di volte pari a R , quindi, lo stato $|\psi\rangle$ viene ruotato di fino a raggiungere un angolo $\frac{\theta}{2} \leq \frac{\pi}{4}$ rispetto allo stato obiettivo $|\beta\rangle$. A questo punto, l'osservazione dello stato del sistema nella base computazionale permette di ottenere una soluzione al problema di ricerca con una probabilità di successo almeno pari al 50%.

In determinate condizioni, è possibile migliorare ulteriormente questa probabilità. Ad esempio quando il numero di soluzioni M è molto minore di N ($M \ll N$), la probabilità di successo può essere ulteriormente migliorata, poiché l'angolo θ si approssima come:

$$\theta \approx \sin(\theta) \approx 2\sqrt{\frac{M}{N}}.$$

In questo caso, l'errore angolare massimo nello stato finale è pari a $\frac{\theta}{2} \approx \sqrt{\frac{M}{N}}$, il che implica una probabilità di errore di almeno $\frac{M}{N}$.

Si osservi che il numero di iterazioni R dipende solo dal numero di soluzioni M , ma non dall'identità delle soluzioni stesse. Pertanto, una volta conosciuto il valore di M , è possibile applicare l'algoritmo di Grover per trovare la soluzione senza bisogno di ulteriori informazioni sulle soluzioni specifiche.

La formula di R sopra riportata fornisce un'espressione esatta per il numero di chiamate all'oracolo necessarie per eseguire l'algoritmo di ricerca, ma sarebbe utile disporre di una versione semplificata che riassume il comportamento essenziale di R .

Per ottenere un limite superiore approssimato, notiamo che dalla relazione $R = \text{CI} \left(\frac{\arccos(\sqrt{M/N})}{\theta} \right)$ si ha che:

$$R \leq \lceil \frac{\pi}{2\theta} \rceil,$$

dove θ rappresenta l'angolo di rotazione. Questo perché al massimo la funzione arcoseno può raggiungere un valore di $\frac{\pi}{2}$.

Utilizzando il limite inferiore per θ e assumendo che $M \leq \frac{N}{2}$, si ottiene: $\frac{\theta}{2} \geq \sin(\frac{\theta}{2}) = \sqrt{\frac{M}{N}}$, da cui si deduce il limite superiore sul numero di iterazioni necessarie: $R \leq \left\lceil \frac{\pi}{4} \sqrt{\frac{N}{M}} \right\rceil$.

In altre parole, il numero di iterazioni R richieste per ottenere una soluzione con alta probabilità cresce come $O(\sqrt{N/M})$, comportando così un miglioramento quadratico rispetto al metodo classico che richiederebbe $O(N/M)$ chiamate all'oracolo per risolvere il problema di ricerca.

Possiamo quindi approssimare il numero di iterazioni attraverso la formula:

$$R = \left\lceil \frac{\pi}{4} \sqrt{\frac{N}{M}} \right\rceil$$

Cosa succede quando più della metà degli elementi sono soluzioni al problema di ricerca, cioè quando $M \geq \frac{N}{2}$?

L'angolo θ tra $|\psi\rangle$ e $|\alpha\rangle$ è definito dalla relazione:

$$\sin \theta = 2 \sqrt{\frac{M}{N} \frac{N-M}{N}} = 2 \frac{\sqrt{M(N-M)}}{N}$$

Questo deriva dal calcolo della probabilità di trovare lo stato iniziale nel sottospazio target e nel sottospazio ortogonale $|\alpha\rangle$ (rispetto a $|\beta\rangle$), con la normalizzazione del sistema quantistico.

Prendendo l'arcoseno per ottenere l'angolo θ si ha:

$$\theta = \arcsin \left(2 \frac{\sqrt{M(N-M)}}{N} \right)$$

Da quest'ultima espressione, osserviamo che l'angolo θ diventa più piccolo man mano che M varia da $\frac{N}{2}$ a N . Di conseguenza, il numero di iterazioni richieste dall'algoritmo di ricerca aumenta con M , per $M \geq \frac{N}{2}$.

Questa è una proprietà poco intuitiva per un algoritmo di ricerca: ci aspettiamo che il problema diventi più facile da risolvere quando il numero di soluzioni aumenta, non il contrario!

Se $M \geq \frac{N}{2}$ si può aggirare il problema usando uno di questi due metodi.

- Possiamo semplicemente selezionare un elemento a caso dallo spazio di ricerca e verificare se è una soluzione utilizzando l'oracolo. Questo approccio ha una probabilità di successo di almeno 50%, e richiede una sola consultazione dell'oracolo. Lo svantaggio è che potremmo non conoscere in anticipo il numero di soluzioni M .
- L'idea è di raddoppiare il numero di elementi nello spazio di ricerca aggiungendo N elementi extra, nessuno dei quali è una soluzione. Di conseguenza, meno della metà degli elementi nel nuovo spazio di ricerca saranno soluzioni.

Questo viene realizzato aggiungendo un singolo qubit all'indice di ricerca, raddoppiando il numero di elementi da cercare a $2N$.

Il nuovo problema di ricerca ha solo M soluzioni su $2N$ elementi, quindi eseguendo l'algoritmo di ricerca, vediamo che al massimo sono richieste:

$$R = \frac{\pi}{4} \sqrt{\frac{2N}{M}}$$

E nel caso in cui non conosciamo M ? Come facciamo a stimare R ?

Non approfondiremo molto questo caso. Diremo solo che esiste un algoritmo di **Quantum Counting** che se integrato all'algoritmo di Grover è in grado di stimare il numero M di soluzioni.

In particolare, Quantum Counting si basa sulla **trasformata quantistica di Fourier** per ottenere una stima del valore di M . La tecnica si affida all'**interferenza** per ottenere informazioni statistiche sul numero di soluzioni a partire dall'oracolo di Grover, senza bisogno di esplorare ogni singola soluzione.

7.5 Applicazione dell'Algoritmo di Grover

Consideriamo un sistema composto da 3 qubit. Lo stato del sistema può essere rappresentato come una sovrapposizione lineare di tutti gli $N = 2^3 = 8$ stati, espressa dalla seguente equazione:

$|\psi\rangle = \alpha_0|000\rangle + \alpha_1|001\rangle + \alpha_2|010\rangle + \alpha_3|011\rangle + \alpha_4|100\rangle + \alpha_5|101\rangle + \alpha_6|110\rangle + \alpha_7|111\rangle$
dove α_i è l'ampiezza dello stato $|i\rangle$. Possiamo rappresentare quindi $N = 2^3 = 8$ stati.

Supponiamo di voler identificare, utilizzando l'algoritmo di Grover, uno specifico elemento del database rappresentato dallo stato $|011\rangle$. Questo stato è l'obiettivo della nostra ricerca.

Applicando la trasformazione di Hadamard, avremo che:

$$\alpha_i = \frac{1}{\sqrt{8}} = \frac{1}{2\sqrt{2}} \quad \forall i = 0, 1, \dots, 7$$

Nel contesto dell'algoritmo di Grover, il numero di stati target M è pari a 1, poiché il risultato desiderato è un unico stato. D'altra parte, il numero totale di stati N è pari ad 8. Poiché $M \leq \frac{N}{2}$, stimiamo il numero di iterazioni:

$$R = \left\lceil \frac{\pi}{4} \sqrt{\frac{8}{1}} \right\rceil = \left\lceil \frac{\pi}{4} \cdot 2 \cdot \sqrt{2} \right\rceil = \left\lceil \frac{\pi}{2} \cdot \sqrt{2} \right\rceil \approx \lceil 2.22 \rceil = 2$$

Applichiamo l'interazione di Grover al sistema

$$G|\psi\rangle = (2|\psi\rangle\langle\psi| - Id)O|\psi\rangle$$

Il primo passo è usare l'oracolo per cambiare segno all'ampiezza dello stato $|011\rangle$. Ponendo $|x\rangle = O|\psi\rangle$,abbiamo:

$$|x\rangle = \frac{1}{2\sqrt{2}}|000\rangle + \frac{1}{2\sqrt{2}}|001\rangle + \frac{1}{2\sqrt{2}}|010\rangle - \frac{1}{2\sqrt{2}}|011\rangle + \frac{1}{2\sqrt{2}}|100\rangle + \frac{1}{2\sqrt{2}}|101\rangle + \frac{1}{2\sqrt{2}}|110\rangle + \frac{1}{2\sqrt{2}}|111\rangle$$

Dopo di che si applica l'operatore di diffusione $2|\psi\rangle\langle\psi| - Id$ al vettore $|x\rangle$.

Possiamo esprimere $|x\rangle$ come la somma dello stato iniziale del sistema $|\psi\rangle$ e una componente aggiuntiva che riflette l'effetto dell'oracolo. In pratica:

$$|x\rangle = |\psi\rangle - \frac{2}{2\sqrt{2}}|011\rangle$$

Pertanto:

$$G|\psi\rangle = (2|\psi\rangle\langle\psi| - Id) \cdot (|\psi\rangle - \frac{2}{2\sqrt{2}}|011\rangle) = 2|\psi\rangle\langle\psi| - |\psi\rangle - \frac{2}{\sqrt{2}}|\psi\rangle\langle\psi| + \frac{1}{\sqrt{2}}|011\rangle$$

$\langle\psi| \psi\rangle$ è un prodotto scalare di $|\psi\rangle$ con se stesso. Questo è semplicemente la norma quadrata dello stato $|\psi\rangle$, che deve essere 1 per un sistema quantistico normalizzato.

$$\langle\psi| \psi\rangle = \left(\frac{1}{\sqrt{8}}\right)^2 \sum_{x=0}^7 \langle x | x \rangle = \frac{1}{8} \sum_{x=0}^7 1 = 1$$

Quando calcoliamo il prodotto scalare $\langle \psi | 011 \rangle$, stiamo cercando di ottenere la proiezione dello stato $|\psi\rangle$ su $|011\rangle$, ossia quanto lo stato $|\psi\rangle$ "contiene" $|011\rangle$.

Ogni prodotto scalare tra $\langle x | 011 \rangle$ è 0, tranne quando $x = 011$, poiché lo stato $|011\rangle$ è ortogonale a tutti gli altri stati della base computazionale tranne se stesso. Quindi, il termine che conta è: $\langle 011 | 011 \rangle = 1$. Di conseguenza:

$$\langle \psi | 011 \rangle = \frac{1}{\sqrt{8}} \cdot 1 = \frac{1}{2\sqrt{2}}$$

Questo è il prodotto scalare tra $|\psi\rangle$ e $|011\rangle$. Il valore che otteniamo, $\frac{1}{\sqrt{8}}$, è proprio il coefficiente con cui lo stato $|011\rangle$ appare nella sovrapposizione di $|\psi\rangle$.

Ritorniamo a dove eravamo prima:

$$G|\psi\rangle = \dots = 2|\psi\rangle - |\psi\rangle - \frac{2}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} \cdot |\psi\rangle + \frac{1}{\sqrt{2}}|011\rangle = |\psi\rangle - \frac{1}{2}|\psi\rangle + \frac{1}{\sqrt{2}}|011\rangle = \frac{1}{2}|\psi\rangle + \frac{1}{\sqrt{2}}|011\rangle$$

Prendendo come risultato della prima iterazione di Grover $|\psi_1\rangle$ abbiamo:

$$\begin{aligned} |\psi_1\rangle &= G|\psi\rangle = \frac{1}{4\sqrt{2}}|000\rangle + \frac{1}{4\sqrt{2}}|001\rangle + \frac{1}{4\sqrt{2}}|010\rangle + \frac{5}{4\sqrt{2}}|011\rangle + \\ &\quad + \frac{1}{4\sqrt{2}}|100\rangle + \frac{1}{4\sqrt{2}}|101\rangle + \frac{1}{4\sqrt{2}}|110\rangle + \frac{1}{4\sqrt{2}}|111\rangle \end{aligned}$$

Dobbiamo fare una seconda iterazione di Grover:

$$|\psi_2\rangle = G|\psi_1\rangle = (2|\psi\rangle\langle\psi| - Id)O|\psi_1\rangle$$

Ci risparmiamo i calcoli, visto che sono analoghi a quelli visti precedentemente con la prima iterazione. Ci limitiamo a dire che, preso $y = O|\psi_1\rangle$, abbiamo:

$$|y\rangle = \frac{1}{2}|\psi\rangle - \frac{3}{2\sqrt{2}}|011\rangle$$

Infine il risultato che a noi interessa è:

$$\begin{aligned} |\psi_2\rangle &= -\frac{1}{8\sqrt{2}}|000\rangle - \frac{1}{8\sqrt{2}}|001\rangle - \frac{1}{8\sqrt{2}}|010\rangle + \frac{11}{8\sqrt{2}}|011\rangle - \frac{1}{8\sqrt{2}}|100\rangle - \frac{1}{8\sqrt{2}}|101\rangle \\ &\quad - \frac{1}{8\sqrt{2}}|110\rangle - \frac{1}{8\sqrt{2}}|111\rangle \end{aligned}$$

Abbiamo trovato il coefficiente di $|011\rangle$, ossia $\alpha_3 = \frac{11}{8\sqrt{2}}$. Possiamo calcolare la probabilità di misurare lo stato corrispondente alla soluzione corretta:

$$\left|\frac{11}{8\sqrt{2}}\right|^2 = \frac{121}{128} \approx 94.5\%$$

Di conseguenza, la probabilità di trovare uno stato errato è circa il 5,5%.

Un 94.5% di possibilità di successo è circa 17 volte la probabilità di errore.

7.6 Algoritmo di Grover con Quantum Annealing

L'algoritmo di Grover è concepito per essere implementato all'interno del paradigma del **Gate-Based Quantum Computing**, sfruttando circuiti quantistici che utilizzano porte logiche come quella di Hadamard. Tuttavia, nonostante questa connessione intrinseca con i circuiti quantistici basati su porte, è possibile adattare l'algoritmo di Grover anche al paradigma del **Quantum Annealing** [31].

Nel *Quantum Annealing*, l'evoluzione dinamica del sistema è descritta dall'Hamiltoniano totale, la cui formulazione matematica è data dall'espressione:

$$H(t) = s(t) \cdot H_f + [1 - s(t)] \cdot H_i$$

L'implementazione dell'algoritmo di Grover all'interno di questo paradigma richiede una progettazione accurata dell'Hamiltoniano iniziale H_i , dell'Hamiltoniano finale H_f e della scheduling function $s(t)$ che regola la transizione tra H_i e H_f .

Diciamo subito che nel *Quantum Annealing*, l'oracolo non è implementato in modo diretto come nel caso Gate-Based, dove è definito come un'operazione unitaria (un "giro" del circuito quantistico). Qui, l'approccio è più legato all'evoluzione del sistema attraverso un Hamiltoniano, e l'oracolo deve essere incorporato in questo Hamiltoniano di evoluzione.

Iniziamo costruendo l'Hamiltoniano iniziale. Definiamolo in modo che $|\psi\rangle$ sia il suo stato fondamentale. Lo stato fondamentale di un Hamiltoniano H è il vettore nello spazio di Hilbert associato al valore più basso dell'energia (autovalore minimo) del sistema. Definiamo $|\psi\rangle$ come una sovrapposizione uniforme di tutti i vettori base $|i\rangle_n$ in uno spazio di Hilbert di dimensione $N = 2^n$ (vedi la **trasformazione di Hadamard**):

$$|\psi\rangle \equiv \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle_n$$

Torniamo all'implementazione. L'Hamiltoniano iniziale è definito come:

$$H_i = Id - |\psi\rangle\langle\psi|$$

dove:

- Id rappresenta l'operatore identità;
- $|\psi\rangle\langle\psi|$ è il proiettore sullo stato $|\psi\rangle$, ossia un operatore che lascia invariato $|\psi\rangle$ e annulla ogni vettore ortogonale a $|\psi\rangle$.

Questa costruzione garantisce che:

- $|\psi\rangle$ sia uno stato proprio di H_i con autovalore 0:

$$H_i|\psi\rangle = (Id - |\psi\rangle\langle\psi|)|\psi\rangle = |\psi\rangle - |\psi\rangle = 0.$$

- Gli stati ortogonali a $|\psi\rangle$ abbiano autovalore 1. Se $|\phi\rangle$ è uno stato ortogonale a $|\psi\rangle$, allora:

$$\langle\psi|\phi\rangle = 0 \implies |\psi\rangle\langle\psi|\phi\rangle = 0 \implies H_i|\phi\rangle = |\phi\rangle.$$

In pratica, la costruzione dell'Hamiltoniano iniziale non differisce sostanzialmente da quanto già visto in precedenza. Infatti, tutti i qubit del sistema sono ora in uno stato di sovrapposizione equiprobabile, che rappresenta sia la costruzione dell'Hamiltoniano iniziale, sia l'inizializzazione dell'algoritmo di Grover.

Passiamo a definire l'Hamiltoniano finale:

$$H_f = Id - |\beta\rangle\langle\beta|$$

dove $|\beta\rangle$ rappresenta lo stato fondamentale, con autovalore 0, di H_f , ergo l'elemento da cercare. Ciò è analogo a quello che abbiamo fatto con l'Hamiltoniano iniziale.

Abbiamo visto che lo stato iniziale può essere espresso in questo modo:

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle$$

dove:

- $|\alpha\rangle$ rappresenta la sovrapposizione normalizzata di tutti gli stati che non sono soluzioni del problema ($x \notin S$);
- $|\beta\rangle$ rappresenta la sovrapposizione normalizzata di tutti gli stati che sono soluzioni del problema ($x \in S$).

Questi due stati formano una base in un sottospazio bidimensionale, in cui l'iterazione di Grover opera come una rotazione, progressivamente orientando lo stato iniziale $|\psi\rangle$ verso $|\beta\rangle$, cioè verso la soluzione.

Possiamo esprimere $|\alpha\rangle$ in questo modo:

$$|\alpha\rangle = \sqrt{\frac{N}{N-M}}|\psi\rangle - \sqrt{\frac{M}{N-M}}|\beta\rangle$$

Quando applichiamo l'iterazione di Grover, essa si traduce in una rotazione nel piano generato da $|\alpha\rangle$ e $|\beta\rangle$. Più precisamente, ogni iterazione effettua due riflessioni successive: la prima rispetto al sottospazio ortogonale a $|\beta\rangle$, operata dall'oracolo O , e la seconda rispetto allo stato iniziale $|\psi\rangle$, tramite l'operazione di diffusione $2|\psi\rangle\langle\psi| - Id$. Queste riflessioni, combinate, producono una rotazione che riduce progressivamente l'angolo tra lo stato corrente e $|\beta\rangle$.

Per semplificare i calcoli poniamo M=1.

Quindi:

$$H(t) = s(t) \cdot H_f + [1 - s(t)] \cdot H_i$$

Introduciamo un importante concetto che useremo più avanti. La **Legge di Born** ci dice la probabilità di ottenere un risultato specifico quando si misura un sistema quantistico. Se un sistema è in uno stato quantistico $|\psi\rangle$, e si misura una grandezza osservabile associata ad uno stato $|\beta\rangle$, la probabilità di ottenere il risultato associato a $|\beta\rangle$ è data da:

$$P(\beta) = |\langle\beta|\psi\rangle|^2.$$

Questa espressione dice che la probabilità di osservare lo stato $|\beta\rangle$ durante una misura del sistema nello stato $|\psi\rangle$ è il quadrato del modulo del prodotto scalare tra i due stati.

Nel nostro caso:

$$P(\beta) = |\langle\beta|\psi\rangle|^2 = \left| \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \langle\beta|i\rangle \right|^2 = \left| \frac{1}{\sqrt{N}} \right|^2 = \frac{1}{N}.$$

Questo perché, se $|\beta\rangle$ è uno degli stati base $|j\rangle$, allora $\langle\beta|i\rangle$ è 1 solo quando $i = j$ e 0 altrimenti. Pertanto, il prodotto scalare sarà: $\frac{1}{\sqrt{N}}$. Questo perché c'è solo un singolo termine non nullo nella somma, quando $i = \beta$, e il suo valore è $\frac{1}{\sqrt{N}}$, che poi elevato al quadrato sarà $\frac{1}{N}$.

Notiamo che $\langle \beta | \alpha \rangle = 0$, $\langle \alpha | \alpha \rangle = 1$. Inoltre, usando $\langle \beta | \psi \rangle = \frac{1}{\sqrt{N}}$ (come abbiamo visto con la legge di Born) e esprimendo $|\psi\rangle$ come combinazione lineare di $|\alpha\rangle$ e $|\beta\rangle$, otteniamo:

$$\begin{aligned} H_i |\alpha\rangle &= \frac{1}{N} |\alpha\rangle - \frac{\sqrt{N-1}}{N} |\beta\rangle, \\ H_i |\beta\rangle &= \frac{N-1}{N} |\beta\rangle - \frac{\sqrt{N-1}}{N} |\alpha\rangle, \\ H_f |\alpha\rangle &= |\alpha\rangle, \\ H_f |\beta\rangle &= 0. \end{aligned}$$

Per analizzare l'evoluzione dell'Hamiltoniano totale nel tempo, ci possiamo concentrare solo sul sottospazio bidimensionale $\{|\alpha\rangle; |\beta\rangle\}$. In questo sottospazio l'Hamiltoniano agisce come un generatore di rotazioni. L'Hamiltoniano totale, in questo sottospazio, può essere rappresentato come una matrice quadrata:

$$\begin{pmatrix} \langle \alpha | H(t) | \alpha \rangle & \langle \alpha | H(t) | \beta \rangle \\ \langle \beta | H(t) | \alpha \rangle & \langle \beta | H(t) | \beta \rangle \end{pmatrix}$$

Risolvendo queste operazioni, la matrice completa nel sottospazio è:

$$\hat{H}(t) = \begin{pmatrix} s(t) + \frac{1-s(t)}{N} & -\frac{[1-s(t)]\sqrt{N-1}}{N} \\ -\frac{[1-s(t)]\sqrt{N-1}}{N} & \frac{[1-s(t)](N-1)}{N} \end{pmatrix}.$$

La matrice di \hat{H} può essere separata in una parte diagonale e una parte non diagonale. La parte diagonale è proporzionale all'identità e rappresenta una trasformazione scalare, mentre la parte non diagonale descrive la rotazione tra gli stati, che dipende dai parametri $s(t)$ e N (poiché $M = 1$). Così facendo la matrice può essere espressa come:

$$\hat{H}(t) = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \frac{1}{2} \begin{pmatrix} \frac{2[s(t)-1]}{N} - 2s(t) + 1 & 2[1-s(t)]\sqrt{\frac{N-1}{N}} \\ 2[1-s(t)]\sqrt{\frac{N-1}{N}} & \frac{2[1-s(t)]}{N} + 2s(t) - 1 \end{pmatrix}.$$

Per caratterizzare la forza della rotazione in funzione dei parametri $s(t)$ e N , introduciamo il parametro $g(t)$. Questo parametro è definito come:

$$g(t) = \sqrt{\frac{4s(t)[s(t)-1](N-1)}{N} + 1}$$

Questo parametro è utilizzato per normalizzare la rotazione e ottenere i coseni e seni dell'angolo di rotazione, ma non è solo questo. Esso rappresenta anche il gap energetico tra $|\alpha\rangle$ e $|\beta\rangle$.

Successivamente, possiamo definire i parametri angolari associati alla rotazione come segue:

$$\cos \theta(t) = \frac{1 - 2s(t) + \frac{2[s(t)-1]}{N}}{g(t)}$$

$$\sin \theta(t) = \frac{2[1 - s(t)]}{g(t)} \sqrt{\frac{N-1}{N}}$$

In questo modo, l'Hamiltoniano $\hat{H}(t)$ può essere riscritto come:

$$\hat{H}(t) = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \frac{g(t)}{2} \begin{pmatrix} \cos \theta(t) & \sin \theta(t) \\ \sin \theta(t) & -\cos \theta(t) \end{pmatrix}.$$

Se $s(t)$ una funzione lineare, il gap raggiunge il minimo quando $t = \frac{T}{2}$ e quindi $s = \frac{1}{2}$.

$$g_{min} = g\left(\frac{1}{2}\right) = \frac{1}{\sqrt{N}}$$

Poiché il tempo di evoluzione è approssimativamente dato da $T \approx O\left(\frac{1}{g_{min}^2}\right)$, si potrebbe inizialmente concludere che il fattore di ritardo T necessario è lineare, cioè $O(N)$. Tuttavia, conoscendo la funzione $g(t)$, possiamo ridurre il tempo di esecuzione a $O(\sqrt{N})$ [32].

La formula $g(t)$ descrive come varia il gap energetico nel tempo, fornendo il valore del fattore di ritardo T necessario in ogni istante. Questo consente di variare dinamicamente T , rispettando comunque le condizioni che garantiscono l'adiabaticità dell'evoluzione del sistema. Per ottenere un tempo di esecuzione $T \approx O(\sqrt{N})$, possiamo utilizzare la seguente scheduling function [33]:

$$s(t) = \frac{1}{2} + \frac{1}{2\sqrt{N-1}} \tan\left((2\frac{t}{T}-1) \arctan \sqrt{N-1}\right)$$

In questo modo, la formula completa per l'Hamiltoniano nel contesto dell'algoritmo di Grover, implementato tramite *Quantum Annealing*, diventa:

$$H(t) = [\frac{1}{2} + \frac{1}{2\sqrt{N-1}} \tan\left((2\frac{t}{T}-1) \arctan \sqrt{N-1}\right)][Id - |\beta\rangle\langle\beta|] -$$

$$[-\frac{1}{2} + \frac{1}{2\sqrt{N-1}} \tan\left((2\frac{t}{T}-1) \arctan \sqrt{N-1}\right)][Id - |\psi\rangle\langle\psi|]$$

Per quanto riguarda invece il tempo di evoluzione T , possiamo calcolarlo nel seguente modo:

$$T \approx \frac{N}{\sqrt{N-1}} \arctan(\sqrt{N-1}) \rightarrow T \approx \frac{\pi}{2} \sqrt{N}$$

I risultati presentati si generalizzano facilmente nel caso in cui esistano $M \geq 1$ soluzioni.

Invece di evolvere in un sottospazio bidimensionale, il sistema evolve in un sottospazio di dimensione $M+1$, generato da $\{|\alpha\rangle; |\beta\rangle\}$. L'Hamiltoniano totale può essere scritto in questa base come:

$$H(t) = \begin{pmatrix} [1-s(t)] \left(1 - \frac{1}{N}\right) & -\frac{[1-s(t)]}{N} & \cdots & -([1-s(t)])\sqrt{\frac{N-M}{N}} \\ -\frac{[1-s(t)]}{N} & ([1-s(t)])(1-\frac{1}{N}) & \cdots & -([1-s(t)])\sqrt{\frac{N-M}{N}} \\ \vdots & \vdots & \ddots & \vdots \\ -([1-s(t)])\sqrt{\frac{N-M}{N}} & -([1-s(t)])\sqrt{\frac{N-M}{N}} & \cdots & s(t) + ([1-s(t)])(1-\frac{N-M}{N}) \end{pmatrix}.$$

Questo Hamiltoniano può essere facilmente diagonalizzato. Si trova che ci sono $M-1$ autovalori uguali a $[1-s(t)]$ e due autovalori:

$$\lambda_{1,2}(t) = \frac{1}{2} \pm \frac{1}{2} \sqrt{[1-2s(t)]^2 + \frac{4M}{N}s(t)[1-s(t)]}$$

che determinano il minimo gap rilevante:

$$g(t) = \sqrt{[1-2s(t)]^2 + \frac{4M}{N}s(t)[1-s(t)]}$$

Gli altri $N-M-1$ autovalori sono irrilevanti per l'evoluzione adiabatica.

Come nel caso $M=1$, il gap raggiunge il minimo quando $t = \frac{T}{2}$ e quindi $s(\frac{T}{2}) = \frac{1}{2}$, dove:

$$g_{min} \approx \sqrt{\frac{M}{N}}$$

che si riduce al caso di un singolo stato marcato per $M=1$. Pertanto, l'evoluzione adiabatica con il programma ottimale descritto nella sezione precedente porta a un tempo totale:

$$T \approx O(\sqrt{\frac{N}{M}})$$

7.7 Perché implementare l’Algoritmo di Grover con il Quantum Annealing

Dopo aver introdotto l’algoritmo di Grover e aver discusso la sua implementazione attraverso il *Quantum Annealing*, dimostrando che raggiunge una complessità $O(\sqrt{N})$, ci poniamo una domanda tanto banale quanto fondamentale. Perché implementare l’algoritmo di Grover con il Quantum Annealing?

In primo luogo, perché così l’algoritmo sfrutta la dinamica di evoluzione continua, che si adatta perfettamente alle tecnologie di computer, come quelli della D-Wave, dove il sistema quantistico evolve attraverso un Hamiltoniano adiabatico. Questa evoluzione permette di risolvere i problemi in modo più semplice rispetto ai modelli basati su porte, i quali richiedono la costruzione di circuiti quantistici complessi che operano in maniera discreta e con un numero maggiore di porte logiche.

In secondo luogo l’uso del *Quantum Annealing* consente di ridurre la necessità di una precisa temporizzazione delle operazioni quantistiche. Nella versione Gate-Based dell’algoritmo di Grover, le iterazioni devono essere regolate con precisione, e un errore temporale potrebbe compromettere significativamente l’efficacia dell’algoritmo. Al contrario, con il *Quantum Annealing*, l’evoluzione del sistema è governata da un parametro continuo che, se progettato correttamente, garantisce una ricerca adiabaticamente ottimale senza dover necessariamente gestire la complessità di una sequenza di porte discrete. Questo approccio fornisce una robustezza maggiore rispetto a piccoli errori nelle implementazioni hardware, come quelli che potrebbero verificarsi in un contesto di calcolo quantistico basato su porte.

Infine, un vantaggio importante riguarda la scalabilità del sistema. Sebbene l’algoritmo di Grover, implementato tramite porte quantistiche, possa affrontare determinati problemi in modo efficace per un numero relativamente ridotto di qubit, il *Quantum Annealing* è particolarmente utile per sistemi con un numero molto elevato di variabili, che sono tipici di molte applicazioni pratiche. La capacità del *Quantum Annealing* di eseguire ottimizzazioni globali senza l’intervento manuale di operazioni logiche complesse rende questo approccio adatto a sistemi ad alta dimensione, con una complessità computazionale che cresce meno rapidamente rispetto alle tecniche basate su porte.

Finora, l’algoritmo di Grover nell’Adiabatic Quantum Computing è rimasto confinato alla teoria, privo di implementazioni pratiche. Tuttavia, studi recenti suggeriscono che il *Quantum Annealing* potrebbe permetterne l’implementazione, aprendo nuove possibilità per l’ottimizzazione di problemi complessi, pur essendo ancora in fase preliminare di ricerca. Questo approccio potrebbe superare le limitazioni degli algoritmi quantistici attuali, offrendo vantaggi nella risoluzione di problemi complessi.

Capitolo 8: Conclusioni

Il **Quantum Annealing** rappresenta una delle applicazioni più promettenti della computazione quantistica nel campo dell'ottimizzazione combinatoria. Basandosi su fenomeni quali l'effetto tunnel e il Teorema Adiabatico, questo metodo consente di esplorare lo spazio delle soluzioni con un'efficienza unica, superando le limitazioni degli algoritmi classici e riducendo il rischio di rimanere nei minimi locali del paesaggio energetico.

Uno degli aspetti più significativi del *Quantum Annealing* è la sua capacità di sfruttare l'effetto tunnel per superare le barriere energetiche che separano i minimi locali dal minimo globale. Questo approccio, unito alla flessibilità nella rappresentazione dei problemi e alla possibilità di implementare vincoli complessi, lo rende una tecnologia versatile per applicazioni in ambiti come la **logistica**, l'**intelligenza artificiale**, la **bioinformatica** e l'**analisi finanziaria**.

Sebbene l'implementazione pratica presenti ancora sfide significative, come la riduzione del gap minimo e il miglioramento della scalabilità nei dispositivi quantistici, i progressi tecnologici nel design di architetture hardware, come il grafo Pegasus, e nelle tecniche di embedding aprono nuove prospettive per un futuro in cui il Quantum Annealing potrà essere utilizzato per risolvere problemi complessi che oggi risultano inaccessibili ai calcolatori classici.

Il Quantum Annealing non rappresenta solo un nuovo paradigma computazionale, ma anche un esempio di come i principi fondamentali della fisica quantistica possano essere sfruttati per risolvere problemi reali. Questo dimostra che l'innovazione tecnologica non si limita a spingere i confini della conoscenza teorica, ma contribuisce attivamente a trasformare le nostre capacità di affrontare e risolvere problemi complessi. Con il continuo avanzamento della ricerca, il Quantum Annealing potrebbe diventare un pilastro fondamentale nella prossima **rivoluzione computazionale**.

Appendice A: Codice di risoluzione del TSP

Il codice riportato di seguito implementa una soluzione al problema del Commesso Viaggiatore utilizzando tecniche di **Quantum Annealing**, in particolare sfruttando l'architettura ibrida del Quantum Annealer della D-Wave. Per questo motivo, l'approccio adottato risulta essere leggermente diverso. La soluzione è applicata al problema trattato nel capitolo 6, sezione 4.

```
1 import itertools
2 import networkx as nx
3 import dimod as di
4 import pandas as pd
5 from collections import defaultdict
6 from dwave.system import LeapHybridSampler
7
8 # Il codice prende ispirazione da quello contenuto in:
9 # https://github.com/dwavesystems/dwave-networkx/blob/main/
10 #   dwave_networkx/algorithms/tsp.py
11 # Questo per avere una funzione "tsp_to_qubo" in grado di
12 #   fornire una matrice Q ottimizzata per l'architettura dei
13 #   Quantum Annealer della D-Wave
14 # Questo approccio permette di ridurre il numero di variabili
15 #   e semplificare la complessità del problema, rendendolo più
16 #   adatto alla risoluzione
17 # tramite Quantum Annealing.
18
19 def tsp_to_qubo(graph, lagrange=None, weight='weight',
20                 missing_weight=None):
21     """
22         Genera il modello QUBO per risolvere il problema del
23         commesso viaggiatore (TSP) su un grafo dato.
24
25         Parametri:
26         -----
27         graph : networkx.Graph
28             Grafo completo in cui ogni arco ha un attributo di
29             peso associato.
```

```

22
23     lagrange : float, opzionale (default None)
24         Fattore di penalizzazione per garantire che i vincoli
25         siano rispettati.
26
27     weight : str, opzionale (default 'weight')
28         Nome dell'attributo che rappresenta il peso degli
29         archi del grafo.
30
31     missing_weight : float, opzionale (default None)
32         Peso da assegnare agli archi mancanti nel caso di un
33         grafo incompleto. Se non specificato, viene usata
34         la somma totale dei pesi esistenti nel grafo.
35
36     Restituisce:
37     -----
38     qubo : dict
39         Dizionario che rappresenta il modello QUBO. Le chiavi
40         sono tuple di variabili, e i valori i coefficienti.
41     """
42
43     num_nodes = graph.number_of_nodes()
44
45     if lagrange is None:
46         # Determina un valore di default per lagrange basato
47         # sulla dimensione e sui pesi del grafo
48         if graph.number_of_edges() > 0:
49             lagrange = graph.size(weight=weight) * num_nodes /
50             graph.number_of_edges()
51         else:
52             lagrange = 5
53
54     if num_nodes < 3:
55         raise ValueError("Il grafo deve avere almeno 3 nodi.")
56
57     qubo = defaultdict(float)
58
59     # Vincoli di presenza
60     for node in graph:
61         for pos in range(num_nodes):
62             qubo[((node, pos), (node, pos))] -= 2 * lagrange
63
64     # Vincoli di unicita' del nodo
65     for node in graph:
66         for pos1 in range(num_nodes):
67             for pos2 in range(pos1 + 1, num_nodes):
68                 qubo[((node, pos1), (node, pos2))] += 2 *
lagrange
69
70     # Vincoli di unicita' della posizione
71     for pos in range(num_nodes):
72         for node1 in graph:
73             for node2 in set(graph) - {node1}:
74

```

```

67             qubo[((node1, pos), (node2, pos))] += lagrange
68
69     # Termini obiettivo
70     for u, v in graph.edges():
71         cost = graph[u][v][weight]
72         for pos in range(num_nodes):
73             next_pos = (pos + 1) % num_nodes
74             qubo[((u, pos), (v, next_pos))] += cost
75             qubo[((v, pos), (u, next_pos))] += cost
76
77     return qubo
78
79
80 def solve_qubo(Q, **sampler_args):
81     """
82         Risolve un modello QUBO utilizzando un solver esatto (ExactSolver).
83
84     Parametri:
85     -----
86     Q : dict
87         Modello QUBO da risolvere.
88
89     sampler_args : kwargs
90         Argomenti opzionali per il campionatore.
91
92     Restituisce:
93     -----
94     sample : dict
95         Soluzione del modello QUBO con le variabili binarie e i loro valori.
96     """
97     sampler = di.ExactSolver()
98     response = sampler.sample_qubo(Q, **sampler_args)
99
100    sample = response.first.sample
101    print("(Nodo, Posizione) [non contano il vincolo sulla partenza]:")
102    for var, value in sample.items():
103        print(f" {var}: {value}")
104
105    print("\n")
106    return sample
107
108
109 def solve_tsp_quantum_annealing(Q, start=None, **sampler_args):
110     """
111         Risolve il modello QUBO per il problema TSP utilizzando il Quantum Annealer ibrido di D-Wave.
112
113     Parametri:

```

```

114     -----
115     Q : dict
116         Modello QUBO da risolvere.
117
118     start : hashable, opzionale
119         Nodo da considerare come punto di partenza del
120         percorso.
121
122     sampler_args : kwargs
123         Argomenti opzionali per il campionatore.
124
125     Restituisce:
126     -----
127     route : list
128         Lista ordinata dei nodi che rappresentano il percorso
129         ottimale.
130         """
131         sampler = LeapHybridSampler()
132         response = sampler.sample_qubo(Q, **sampler_args)
133         sample = response.first.sample
134
135         route = [None] * len(G)
136         for (city, time), val in sample.items():
137             if val:
138                 route[time] = city
139
140         if start is not None and route[0] != start:
141             # Ruota il percorso per iniziare dal nodo specificato
142             idx = route.index(start)
143             route = route[idx:] + route[:idx]
144
145         print("Soluzione: ")
146         print(route)
147         print("\n")
148
149     return route
150
151
152 def print_matrix(m):
153     """
154     Stampa un dizionario come matrice leggibile utilizzando
155     pandas.
156
157     Parametri:
158     -----
159     m : dict
160         Dizionario che rappresenta una matrice.
161         """
162     variables = sorted(set(var for pair in m.keys() for var in
163                         pair))

```

```
161     matrix = pd.DataFrame(0.0, index=pd.MultiIndex.from_tuples
162                           (variables), columns=pd.MultiIndex.from_tuples(variables))
163
164     for (var1, var2), value in m.items():
165         matrix.loc[var1, var2] = value
166
167     print(matrix.to_string(index=False, header=False))
168     print("\n")
169
170 def create_graph(distance_matrix):
171     """
172     Genera un grafo completo basato su una matrice delle
173     distanze.
174
175     Parametri:
176     -----
177     distance_matrix : numpy.ndarray
178         Matrice contenente le distanze tra i nodi.
179
180     Restituisce:
181     -----
182     G : networkx.Graph
183         Grafo completo con pesi sugli archi.
184
185     G = nx.Graph()
186     num_nodes = distance_matrix.shape[0]
187     for i in range(num_nodes):
188         for j in range(i + 1, num_nodes):
189             weight = distance_matrix[i, j]
190             G.add_edge(i, j, weight=weight)
191
192     return G
193
194 def order_graph(graph):
195     """
196     Ordina i nodi e gli archi di un grafo in modo
197     deterministico.
198
199     Parametri:
200     -----
201     graph : networkx.Graph
202         Grafo da ordinare.
203
204     Restituisce:
205     -----
206     G : networkx.Graph
207         Grafo ordinato.
208
209     G = nx.Graph()
210     G.add_nodes_from(sorted(graph.nodes(data=True)))
211     G.add_edges_from(graph.edges(data=True))
```

```
210     return G
211
212
213 def print_graph(G):
214     """
215     Stampa i nodi e gli archi di un grafo.
216
217     Parametri:
218     -----
219     graph : networkx.Graph
220         Grafo da stampare.
221     """
222     print("Nodi:", sorted(G.nodes()))
223     print("Archi con pesi:")
224     for u, v, data in sorted(G.edges(data=True)):
225         print(f"({u}, {v}) - peso: {data['weight']}")"
226     print("\n")
227
228
229
230 # Risoluzione Problema del TSP
231
232 # Creazione Grafo
233 G = nx.Graph()
234 edges = [("A", "B", 20), ("A", "C", 10), ("A", "D", 18), ("B",
235 "C", 12), ("B", "D", 18), ("C", "D", 6)]
236 G.add_weighted_edges_from(edges)
237 print_graph(G)
238
239 # Creazione QUBO
240 Q = tsp_to_qubo(G)
241 print("Matrice QUBO:")
242 print_matrix(Q)
243
244 # Risoluzione Problema
245 solve_qubo(Q)
246 solve_tsp_quantum_annealing(Q, start="A")
```

Appendice B: Prova che QUBO e Ising sono Isomorfi

La dimostrazione è in Inglese.

$$H(\vec{s}) = - \sum_i^N \sum_{j < i} J_{ij} s_i s_j - \mu \sum_i^N h_i s_i$$

To clean up the equation, let's recast the coefficients.

$$\begin{aligned} -J_{ij} &\rightarrow J_{ij} \\ -\mu h_i &\rightarrow h_i \end{aligned}$$

$$H(\vec{s}) = \sum_i^N \sum_{j < i} J_{ij} s_i s_j + \sum_i^N h_i s_i$$

Recall that $s_i, s_j \in \{-1, 1\}$

Replace s_i, s_j with $x_i, x_j \in \{0, 1\}$ using the formula below.

$$s = 2x - 1$$

(i.e. $-1 = 2(0) - 1$ and $1 = 2(1) - 1$)

$$\begin{aligned} &= \sum_i^N \sum_{j < i} J_{ij} (2x_i - 1)(2x_j - 1) + \sum_i^N h_i (2x_i - 1) \\ &= \sum_i^N \sum_{j < i} J_{ij} (4x_i x_j - 2x_i - 2x_j + 1) + \sum_i^N (2h_i x_i - h_i) \end{aligned}$$

Expand and group constants into the constant k_0

$$= \sum_i^N \sum_{j < i} 4J_{ij} x_i x_j - \sum_i^N x_i \sum_{j < i} 2J_{ij} - \sum_i^N \sum_{j < i} 2J_{ij} x_j + \sum_i^N 2h_i x_i + k_0$$

Group x_i terms together. Have constant $a_i = \sum_{j < i} 2J_{ij} + 2h_i$

$$= \sum_i^N \sum_{j < i} 4J_{ij} x_i x_j + \sum_i^N a_i x_i - 2 \sum_i^N \sum_{j < i} J_{ij} x_j + k_0$$

We can make the third term, $\sum_i^N \sum_{j < i} J_{ij} x_j$, in terms of x_i .

(See “Third Term Expansion” section for how $b_i x_i$ substitution was made.)

$$= \sum_i^N \sum_{j < i} 4J_{ij}x_i x_j + \sum_i^N a_i x_i - 2 \sum_i^N b_i x_i + k_0$$

Let $J'_{ij} = 4J_{ij}$

Let $h'_i = a_i - 2b_i$

$$= \sum_i^N \sum_{j < i} J'_{ij}x_i x_j + \sum_i^N h'_i x_i + k_0$$

$$= H(\vec{x}) + k_0$$

Therefore Ising ($H(\vec{s})$) and QUBO ($H(\vec{x})$) are isomorphic

$$\sum_i^N \sum_{j < i} J_{ij}x_j = J_{10}x_0$$

$$+ J_{20}x_0 + J_{21}x_1$$

$$+ J_{30}x_0 + J_{31}x_1 + J_{32}x_2$$

...

$$+ J_{N0}x_0 + J_{N1}x_1 + J_{N2}x_2 + J_{N3}x_3 + J_{N,N-1}x_{N-1}$$

Observe that each column has like terms.

$$= (J_{10} + J_{20} + J_{30} + \dots + J_{N0})x_0 + (J_{21} + \dots + J_{N1})x_1 + \dots + (J_{N,N-1})x_{N-1}$$

Cast the J_{ij} sums as some constant b_i

$$= b_0 x_0 + b_1 x_1 + \dots + b_{N-1} x_{N-1} + b_N x_N$$

Since there is no x_N term in the original sum, b_N is 0

$$= \sum_i^N b_i x_i$$

"Third Term Expansion"

Bibliografia

- [1] Wim van Dam Dorit Aharonov and Julia Kempe. Adiabatic quantum computation is equivalent to standard quantum computation. November 2004. Available at https://www.researchgate.net/publication/4109376_Adia...
- [2] T. Kadowaki e H. Nishimori. Quantum annealing in the transverse ising model. November 1998. Available at <https://journals.aps.org/pre/pdf/10.1103/PhysRevE.58.5355>.
- [3] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *State space*, pages 80–81, October 2000.
- [4] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *Evolution*, page 82, October 2000.
- [5] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *Quantum Measurement*, pages 84–85, October 2000.
- [6] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *Composite systems*, page 94, October 2000.
- [7] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *Quantum bits*, page 13, October 2000.
- [8] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *Quantum bits*, page 14, October 2000.
- [9] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *Quantum bits*, page 15, October 2000.
- [10] Differences between quantum annealers and gate-based quantum computing. January 2023. Available at <https://quantumzeitgeist.com/differences-between-quantum-annealers-and-gate-based-quantum-computing/>.
- [11] Wikipedia. Ising model. June 2023. Available at https://en.wikipedia.org/wiki/Ising_model.

- [12] Wikipedia. Quadratic unconstrained binary optimization. November 2022. Available at https://en.wikipedia.org/wiki/Quadratic_unconstrained_binary_optimization.
- [13] Dr. Steven Herbert. Quantum Computing (CST Part II) - Lecture 15: Adiabatic Quantum Computing. pages 3–4. Available at https://www.cl.cam.ac.uk/teaching/1920/QuantComp/Quantum_Computing_Lecture_15.pdf.
- [14] D-Wave. What is quantum annealing? Available at https://docs.dwavesys.com/docs/latest/c_gs_2.html.
- [15] Bernard Zygelman. A first introduction to quantum computing and information. *Avoided Crossings*, pages 249–250, October 2019.
- [16] Bernard Zygelman. A first introduction to quantum computing and information. *Landau-Zener Transitions, and the Scheduling Function*, pages 250–252, October 2019.
- [17] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *The Pauli matrices*, pages 65–68, October 2000.
- [18] Dr. Steven Herbert. Quantum computing (cst part ii) - lecture 15: Adiabatic quantum computing. page 14. Available at https://www.cl.cam.ac.uk/teaching/1920/QuantComp/Quantum_Computing_Lecture_15.pdf.
- [19] Wikipedia. Quantum annealing. August 2021. Available at https://en.wikipedia.org/wiki/Quantum_annealing.
- [20] Hristo Djidjev, Stefanie Zbinden, Andreas Bärtschi and Stephan Eidenbenz. Embedding algorithms for quantum annealers with chimera and pegasus connection topologies. 2019. Available at https://public.lanl.gov/djidjev/papers/2019_QUBO_EMBEDDINGS_on_Chimera_and_Pegasus_Host_Graphs.pdf.
- [21] D-Wave Systems. Minorminer. Available at <https://github.com/dwavesystems/minorminer>.
- [22] D-Wave Systems. D-wave api. Available at https://docs.dwavesys.com/docs/latest/doc_rest_api.html.
- [23] D-Wave Systems. Sdk. Available at <https://github.com/dwavesystems/dwave-ocean-sdk>.
- [24] Massimo Pappalardo and Mauro Passacantando. Ricerca operativa. *Compresso Viaggiatore*, pages 275–281, May 2013.
- [25] Dr. Sebastian Feld. Quantum annealing for operation research and machine learning. *QA from a CS/CE perspective*, April 2022. Provided by Professor Cococcioni.

- [26] Lov K. Grover. A fast quantum mechanical algorithm for database search. July 2022. Available at <https://dl.acm.org/doi/10.1145/237814.237866>.
- [27] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *The Procedure*, pages 250–251, October 2000.
- [28] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *The Oracle*, page 248, October 2000.
- [29] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *Geometric Visualization*, page 252, October 2000.
- [30] Michael A. Nielsen and Isaac L. Chuang. Quantum computation and quantum information. *Performance*, pages 253–255, October 2000.
- [31] Bernard Zygelman. A first introduction to quantum computing and information. *The Grover Algorithm*, pages 252–255, October 2019.
- [32] Wimvan Dam, Michele Mosca, Umesh Vazirani. How powerful is adiabatic quantum computation? March 2007. Available at <https://people.eecs.berkeley.edu/~vazirani/pubs/adiabatic.pdf>.
- [33] Tameem Albash and Daniel Lidar. Adiabatic quantum computation. November 2016. Available at https://www.researchgate.net/publication/310235960_Adiabatic_Quantum_Computing.