ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

L'algoritmo QAOA applicato al MaxCut su grafi

Relatore:

Prof.ssa Elisa Ercolessi

Correlatore:

Dott. Federico Dell'Anna

Presentata da:

Mattia Chiurco

Anno Accademico 2022/2023

Sommario

Nel cuore della rivoluzione della computazione quantistica, il Quantum Approximate Optimization Algorithm (QAOA) emerge come uno strumento potente per risolvere problemi di ottimizzazione complessi, come il noto problema del MaxCut. Quest'ultimo, con le sue applicazioni che spaziano dallo studio del magnetismo alla logistica, si presta come un eccellente banco di prova per l'efficacia del QAOA. Questo lavoro indaga l'applicazione del QAOA al MaxCut su una varietà di grafi, tra cui i grafi generici e i Barabási-Albert. Partendo da una descrizione delle basi della computazione quantistica, si affrontano le sfide tecniche relative all'implementazione del QAOA, con particolare attenzione alla strategia del parameter fixing, adottata per migliorare i risultati. Gli esperimenti condotti hanno confermato la capacità del QAOA di risolvere efficacemente il problema del MaxCut, mettendo in luce l'importanza crescente di questo algoritmo nel panorama della computazione quantistica.

Indice

Introduzione **5**

1 La Computazione Quantistica **7**

Capitolo 1 **7**

1.1 Introduzione alla Computazione Quantistica 7

1.1.1 Cenni Storici 7

1.1.2 I Qubit 8

1.1.3 Gate quantistici 12

1.1.4 Circuiti quantistici 15

1.2 Algoritmi quantistici 18

1.2.1 Computazione classica su un computer quantistico 18

1.2.2 Parallelismo quantistico 19

1.2.3 Esempi di algoritmi quantistici 21

1.3 Quantum Approximate Optimization Algorithm 22

1.3.1 Principio variazionale 22

1.3.2 Algoritmi quantistici variazionali 24

1.3.3 MaxCut 24

1.3.4 Quadratic Unconstrained Binary Optimization (QUBO) 27

1.3.5 Circuito per il QAOA 29

1.3.6 Teorema adiabatico e QAOA 31

2 Qiskit e QAOA: Dall'Analisi all'Implementazione **33**

Capitolo 2 **33**

2.1 Introduzione a Qiskit 33

2.2 Il codice 35

2.2.1 Parameter fixing 38
2.3 Implementazione su computer reali 39

3 Analisi Sperimentale del QAOA su Grafi Generici e Barabási–Albert 43

Capitolo 3 43
3.1 Modello Barabási-Albert 43
3.2 Un esempio preliminare 46
3.3 Risultati su grafi Barabási–Albert 56

Conclusione 59

Appendice A 61

Bibliografia 64

Elenco delle figure

Figura 1 Sfera di Bloch, utile per rappresentare un singolo qubit, presa da [1] . 11
Figura 2 Controlled-NOT gate 15
Figura 3 Circuito utilizzato per scambiare due qubit 15
Figura 4 Gate di Toffoli 18
Figura 5 Algoritmo di Deutsch 22
Figura 6 Algoritmo di Deutsch-Jozsa 22
Figura 7 Variational Quantum Eigensolvers 24
Figura 8 Esempio di circuito per il QAOA, presa da [2] 28

Figura 9 Grafo con 5 nodi non pesato. 46
Figura 10 Istogramma contenente tutti i cut del grafo di 5 nodi non pesato, cioè dove tutti gli archi hanno peso 1. 47
Figura 11 Soluzione del MaxCut sul grafo non pesato a 5 nodi. 49
Figura 12 Circuito QAOA a $p = 1$. 49
Figura 13 Istogrammi con distribuzione di probabilità ottenuta dal QAOA al variare di p per il grafo non pesato da 5 nodi. In Figura (a) risultati per $p = 1$, in Figura (b) risultati per $p = 2$, in Figura (c) risultati per $p = 3$, in Figura (d) risultati per $p = 4$. 50
Figura 14 Istogrammi con distribuzione di probabilità ottenuta dal QAOA con aggiunta della tecnica del parameter fixing al variare di p per il grafo non pesato da 5 nodi. In Figura (a) risultati per $p = 1$, in Figura (b) risultati per $p = 2$, in Figura (c) risultati per $p = 3$, in Figura (d) risultati per $p = 4$. 50
Figura 15 Grafo con 5 nodi pesato. 51
Figura 16 Istogramma contenente tutti i cut del grafo di 5 nodi pesato. 51

Figura 17 Istogrammi con distribuzione di probabilità ottenuta dal QAOA al variare di p per il grafo pesato da 5 nodi. In Figura (a) risultati per $p = 1$, in Figura (b) risultati per $p = 2$, in Figura (c) risultati per $p = 3$, in Figura (d) risultati per $p = 4$. 53

Figura 18 Istogrammi con distribuzione di probabilità ottenuta dal QAOA con aggiunta della tecnica del parameter fixing al variare di p per il grafo pesato da 5 nodi. In Figura (a) risultati per $p = 1$, in Figura (b) risultati per $p = 2$, in Figura (c) risultati per $p = 3$, in Figura (d) risultati per $p = 4$. 53

Figura 19 Istogrammi con distribuzione di probabilità ottenuta dal QAOA a $p = 2$ su un computer reale. A sinistra il grafo non pesato e a destra quello pesato. 54

Figura 20 Grafo Barabási–Albert con 10 nodi e connettività $m = 4$ generato utilizzando la libreria networkx di python come descritto nel Capitolo 2 55

Figura 21 Istogramma contenente tutti i cut del grafo Barabási–Albert con 10 nodi e connettività $m = 4$. 55

Figura 22 Istogrammi con distribuzione di probabilità ottenuta dal QAOA con aggiunta della tecnica del parameter fixing al variare di p. In Figura (a) risultati per $p = 1$, in Figura (b) risultati per $p = 2$, in Figura (c) risultati per $p = 3$, in Figura (d) risultati per $p = 4$, in Figura (e) risultati per $p = 5$, in Figura (f) risultati per $p = 6$ 57

Figura 23 Nella figura è rappresentato l'andamento della fidelity in funzione del parametro p su grafi Barabási–Albert di 6 nodi a m costante. Per ogni valore di m, sono stati generati casualmente 10 grafi, e per ciascuno di questi è stata calcolata la fidelity per vari valori di p. Ogni punto del grafico rappresenta la media di tali valori, con la corrispondente incertezza data dalla deviazione standard. In particolare, la Figura (a) mostra i risultati per $m = 1$, la Figura (b) per $m = 2$, la Figura (c) per $m = 3$, la Figura (d) per $m = 4$, e infine, la Figura (e) presenta i risultati per $m = 5$. 58

Introduzione

La computazione quantistica, un concetto rivoluzionario basato sulla sovrapposizione e l'entanglement dei qubit, sta ridisegnando i limiti dell'elaborazione dell'informazione. I computer quantistici, grazie alla loro potenziale capacità di eseguire calcoli estremamente complessi molto più rapidamente dei tradizionali computer classici, aprono nuove frontiere in una serie di campi, tra cui la crittografia, l'ottimizzazione e la simulazione di molecole e materiali.

Tra i vari sviluppi nell'ambito della computazione quantistica, il Quantum Approximate Optimization Algorithm (QAOA) è emerso come uno strumento particolarmente potente. Il QAOA fa parte di una classe di algoritmi noti come algoritmi quantistici ibridi, che sfruttano il meglio di entrambi i mondi classici e quantistici. Tali algoritmi sono progettati per funzionare su dispositivi quantistici detti NISQ (Noisy Intermediate Scale Quantum), una classe di dispositivi quantistici che, sebbene non siano ancora in grado di supportare l'elaborazione quantistica a errore zero, sono tuttavia considerati come la prossima generazione di macchine quantistiche. Il QAOA è diventato rilevante grazie alla sua capacità di risolvere problemi di ottimizzazione combinatoria, tra cui il problema del taglio massimo (MaxCut). Il problema MaxCut è importante non solo dal punto di vista teorico, ma ha anche numerose applicazioni pratiche. Può essere utilizzato per modellare e risolvere problemi in una vasta gamma di settori, come l'assegnazione di risorse, la pianificazione, il routing di reti, l'ottimizzazione di portafigli finanziari e molte altre applicazioni in ambiti quali la logistica, la produzione, le telecomunicazioni, l'informatica e la finanza.

In questa tesi, poniamo un'attenzione particolare alla risoluzione del problema del MaxCut sui grafi di Barabási-Albert utilizzando il QAOA. I grafi di Barabási-Albert, che derivano il loro nome dai fisici Albert-László Barabási e Réka Albert, sono distinti per il loro principio di "attaccamento preferenziale". Questa proprietà li rende modelli ideali per numerose reti complesse e interconnesse che si riscontrano in natura e nelle società, includendo reti di computer, di telecomunicazioni, sociali, neurali e metaboliche. Applicare il QAOA per risolvere il problema del MaxCut su tali grafi potrebbe rivelarsi di grande valore. Potrebbe offrire nuovi spunti su come massimizzare l'efficienza di queste reti, o suggerire strategie per interrompere la diffusione di informazioni o malattie.

La forza del QAOA risiede nella sua capacità di fornire soluzioni approssimate in tempi ragionevoli, che è fondamentale quando si tratta di problemi complessi e di grandi dimensioni

che i metodi classici non possono gestire efficacemente. L'efficacia di QAOA nel risolvere il problema del MaxCut e altre sfide di ottimizzazione ne fa una componente fondamentale nella futura evoluzione della computazione quantistica.

Questa tesi si propone di esplorare l'applicazione del QAOA al problema del MaxCut, focalizzandosi in particolare sui grafi di Barabási-Albert. Analizzeremo in dettaglio la teoria alla base dell'algoritmo partendo da una descrizione delle basi della computazione quantistica in modo da avere una comprensione completa del funzionamento del QAOA. Discuteremo sia questioni teoriche che pratiche, concentrandoci sull'efficacia e l'efficienza del QAOA, sulle sue potenziali limitazioni e sulle sfide che possono sorgere nell'implementazione su dispositivi NISQ. Esploreremo anche strategie per migliorare la precisione dell'algoritmo, come la tecnica del *parameter fixing*. Presenteremo, inoltre, il modello Barabási-Albert con le sue peculiarità. L'obiettivo è di fornire una visione completa e approfondita del QAOA, del suo potenziale ruolo nell'evoluzione della computazione quantistica, e delle implicazioni pratiche del suo utilizzo sui grafi.

Capitolo 1

La Computazione Quantistica

Nel primo capitolo di questa tesi, si pongono le fondamenta per la comprensione del Quantum Approximate Optimization Algorithm (QAOA). Iniziamo con un'introduzione alla computazione quantistica, fornendo una panoramica storica e presentando i concetti fondamentali. Segue una discussione sui principi fondamentali degli algoritmi quantistici, in preparazione alla successiva descrizione del QAOA. Tra i temi affrontati, un particolare rilievo è dato al problema del MaxCut e ai problemi Quadratic Unconstrained Binary Optimization (QUBO). Questi ultimi sono di importanza critica nel contesto del QAOA, in quanto rappresentano classi di problemi per i quali l'algoritmo è particolarmente efficace e a cui molti problemi di ottimizzazione possono essere ricondotti. L'inizio di questo capitolo si basa in larga misura sulla struttura del libro 'Quantum Computation and Quantum Information' di Nielsen, M. A. e Chuang, I. L. [3], un punto di riferimento fondamentale nel campo della computazione quantistica.

1.1 Introduzione alla Computazione Quantistica

1.1.1 Cenni Storici

La computazione quantistica che oggi conosciamo deriva dalla confluenza di diverse discipline scientifiche, tra cui meccanica quantistica, scienza dell'informazione e informatica. Questo campo di ricerca, in continua evoluzione, rappresenta un approccio innovativo alla risoluzione di problemi complessi e all'elaborazione di informazioni.

Durante i primi due decenni del XX secolo, le limitazioni della fisica classica hanno stimolato l'emergere della meccanica quantistica. Questa nuova teoria si è rivelata straordinariamente efficace nell'interpretare una moltitudine di fenomeni, che vanno dalla struttura atomica, alla conformazione del DNA, fino ai processi di fusione nucleare che alimentano le stelle. Parallelamente, nel corso dello stesso secolo, l'informatica ha compiuto passi da gigante grazie a pionieri come Alan Turing. Turing ha introdotto il concetto di "macchina di Turing", un modello teorico che rappresenta l'idea di un computer programmabile. Questo ha portato alla

formulazione della tesi di Church-Turing [4], che stabilisce una corrispondenza tra i procedimenti fisici computazionali e il concetto matematico di una macchina di Turing universale. Le potenziali abilità dei computer quantistici, che promettono di superare le capacità della computazione classica, mettono alla prova la validità della tesi di Church-Turing. Nel 1985, David Deutsch [5] introdusse l'idea di un computer quantistico in grado di simulare in modo efficiente qualsiasi sistema fisico, una proposta che continua a mettere in dubbio la tesi forte di Church-Turing.

Un'importante pietra miliare nella teoria quantistica è stata la scoperta del teorema del non-clonazione [6, 7]. Questo principio afferma che, al contrario dell'informazione classica, uno stato quantistico sconosciuto non può essere replicato o duplicato. Tale peculiarità segna una significativa differenza tra il mondo dell'informazione quantistica e quello dell'informazione classica. L'impulso di governare singoli sistemi quantistici ha dato una significativa spinta alla computazione e alla scienza dell'informazione quantistica. La teoria dell'informazione quantistica ha preso forma parallelamente alla computazione quantistica. È in questo quadro che Ben Schumacher ha introdotto i "bit quantistici", meglio noti come "qubit".

Nel frattempo, l'industria dei semiconduttori si avvicinava sempre più ai limiti fisici dell'ulteriore miniaturizzazione. La legge di Moore, che ha sostenuto l'evoluzione tecnologica per decenni, prevedeva un raddoppio della densità dei transistor su un chip ogni 18 mesi. Tuttavia, con i transistor che tendono a raggiungere dimensioni quasi atomiche, non è più possibile mantenere tale ritmo di riduzione delle dimensioni. Questo prossimo confine fisico ha reso evidente l'esigenza di esplorare nuovi modelli di computazione. I computer quantistici, che sfruttano le proprietà uniche della meccanica quantistica, come la sovrapposizione e l'entanglement, rappresentano una promettente direzione di sviluppo.

1.1.2 I Qubit

L'elaborazione dell'informazione quantistica avviene attraverso l'uso di quantum bit, o qubit, l'equivalente quantistico dei bit nell'informatica classica. Un bit tradizionale può assumere uno *stato* 0 o 1; in maniera simile, un qubit può essere in uno stato $|0\rangle$ o $|1\rangle$. Tuttavia, la caratteristica distintiva dei qubit è la possibilità di trovarsi anche in una combinazione lineare dei due stati, un fenomeno conosciuto come sovrapposizione di stati, rappresentata come:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1.1}$$

dove α e β sono numeri complessi. Dal punto di vista matematico, un qubit è rappresentato come un vettore in uno spazio bidimensionale definito sopra il campo dei numeri complessi $\mathbb{C}$. Questo spazio è conosciuto come spazio di Hilbert $\mathcal{H}$, ed in esso gli stati $|0\rangle$ e $|1\rangle$ costituiscono una base ortonormale. Ogni stato di un qubit può quindi essere espresso come una combinazione lineare di questi. É importante sottolineare che, mentre è sempre possibile misurare lo stato di un bit, non è possibile fare lo stesso con i qubit misurando direttamente α e β. I principi della meccanica quantistica affermano che, ad ogni misura, otterremo lo stato $|0\rangle$ con una probabilità $|\alpha|^2$ oppure lo stato $|1\rangle$ con una probabilità $|\beta|^2$. Queste probabilità devono sommare a 1, ovvero varrà la relazione: $|\alpha|^2 + |\beta|^2 = 1$, geometricamente questo significa che $|\psi\rangle$ è un vettore unitario in uno spazio complesso bidimensionale. Un qubit può essere manipolato durante la computazione e successivamente misurato. Ripetendo questo processo molte volte è possibile ricavare una distribuzione di probabilità da cui è possibile estrapolare informazioni sullo stato $|\psi\rangle$, fornendo di conseguenza dettagli sul risultato della computazione. Un caso fondamentale di sovrapposizione di stati è rappresentato dallo stato $|+\rangle$ definito come:

$$|+\rangle = \sqrt{\frac{1}{2}}|0\rangle + \sqrt{\frac{1}{2}}|1\rangle \tag{1.2}$$

Misurando questo stato avremo la stessa probabilità di ottenere $|0\rangle$ e $|1\rangle$. Qualsiasi base di $\mathbb{C}^2$ può essere utilizzata come base computazionale. Consideriamo ad esempio i due stati, $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ e $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, questi due vettori sono linearmente indipendenti e possono essere utilizzati come base alternativa. È possibile ricavare facilmente che:

$$|0\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle) \quad \text{e} \quad |1\rangle = \frac{1}{\sqrt{2}}(|+\rangle - |-\rangle).$$

In questo contesto, un qubit generico della forma $\alpha|0\rangle + \beta|1\rangle$ può essere riformulato nella nuova base in questo modo:

$$\alpha|0\rangle + \beta|1\rangle = \frac{\sqrt{\alpha}}{2}(|+\rangle + |-\rangle) + \frac{\sqrt{\beta}}{2}(|+\rangle - |-\rangle) = \frac{\alpha+\beta}{\sqrt{2}}|+\rangle + \frac{\alpha-\beta}{\sqrt{2}}|-\rangle \tag{1.3}$$

È possibile eseguire misurazioni in riferimento a una base diversa dalla base standard $\{|0\rangle, |1\rangle\}$. In tal caso, il qubit misurato collasserà in uno degli stati corrispondenti alla base di calcolo in considerazione. Nel caso sopra descritto, questi stati sono rappresentati da $|+\rangle$ e $|-\rangle$.

É possibile riscrivere $|\psi\rangle$ in Eq. (1.1) in funzione di tre numeri reali θ, ϕ e γ:

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right) \tag{1.4}$$

Notiamo che, essendo $|\psi\rangle$ un elemento dello spazio $\mathcal{H}$, una fase globale applicata ad esso non influisce sul risultato misurato. Infatti, nello spazio $\mathcal{H}$ due vettori che differiscono solo da una fase globale sono considerati equivalenti. Quindi possiamo ignorare il parametro γ e scrivere:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{1.5}$$

Ciò vuol dire che possiamo rappresentare $|\psi\rangle$ come un punto su una sfera (spesso chiamata *Sfera di Bloch*, come mostrato in Figura 1)

Consideriamo ora la possibilità di avere più di un solo qubit. Utilizziamo il prodotto tensoriale [8], scritto come $|v\rangle \otimes |w\rangle$, per descrivere uno stato che coinvolge più di un qubit. Lo spazio creato da un sistema composto da n qubit avrà una dimensione pari a 2^n. Ogni vettore normalizzato presente in questo spazio è uno stato possibile del sistema, che è comunemente riferito come *registro quantistico di n qubit*. Per esempio se abbiamo due qubit: il qubit A nello stato $|1\rangle_A$ e il qubit B nello stato $|0\rangle_B$ possiamo esprimere lo stato risultante come

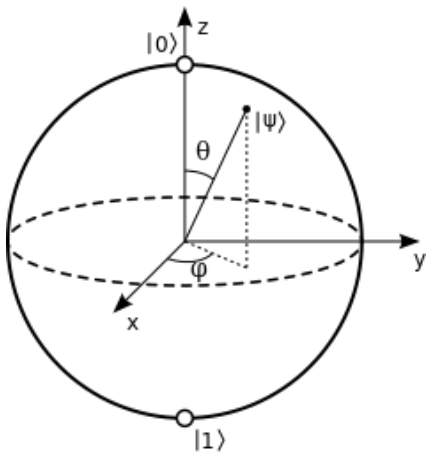

Figura 1: Sfera di Bloch, utile per rappresentare un singolo qubit, presa da [1]

$$|10\rangle_{AB} := |1\rangle_A \otimes |0\rangle_B = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Stati definiti in questo modo sono chiamati non correlati. Esistono degli stati, detti correlati o entangled, che non possono essere rappresentati semplicemente come un prodotto tensoriale dei singoli stati dei qubit. Un esempio famoso di questi è rappresentato dagli stati di Bell, che esamineremo più avanti. Questi particolari stati possiedono una proprietà affascinante: misurando uno dei qubit che compongono lo stato entangled, siamo in grado di acquisire informazioni sullo stato degli altri qubit. Questo avviene nonostante non vi sia un'interazione diretta tra di loro, sottolineando il fenomeno di correlazione quantistica.

Possiamo definire lo stato di un sistema formato da due qubit in funzione di una base contenente quattro elementi $|00\rangle$, $|01\rangle$, $|10\rangle$ e $|11\rangle$. $|\psi\rangle$ sarà quindi espresso nella forma:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{1.6}$$

Effettuando una misurazione sullo stato $|\psi\rangle$ descritto in eq. (1.6), la probabilità di ottenere un risultato $|x\rangle$, dove x può essere uno tra $(00, 10, 01, 11)$, è data dal modulo quadro del corrispondente coefficiente, cioè $|\alpha_x|^2$. La condizione di normalizzazione è espressa nella relazione $\sum_{x\in(0,1)^2} \quad |\alpha_x|^2 = 1$, dove la notazione $x \in (0,1)^2$ indica un set di stringhe bidimensionali in cui ciascuna componente può essere 0 o 1. Un importante esempio sono gli *stati di Bell* :

$$|\psi^{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{1.7}$$

$$|\psi^{10}\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{1.8}$$

$$|\psi^{01}\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{1.9}$$

$$|\psi^{11}\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{1.10}$$

Lo stato di Bell presenta la caratteristica che la misurazione del primo qubit produce due possibili risultati: 0 con probabilità $\frac{1}{2}$, lasciando lo stato post-misurazione come $|\phi\rangle = |00\rangle$, e 1 con probabilità $\frac{1}{2}$, lasciando $|\phi\rangle = |11\rangle$. Gli stati di Bell sono spesso detti stati con la massima correlazione (o *maximally entangled*) e sono molto utilizzati in algoritmi come quello del teletrasporto quantistico. In generale, se abbiamo n qubit la base dello spazio vettoriale sarà della forma $|x_1x_2\ldots x_n\rangle$ e sarà caratterizzato da 2^n coefficienti, i quali soddisfano la regola di normalizzazione.

1.1.3 Gate quantistici

É possibile manipolare i qubit utilizzando i gate quantistici. Essi possono essere esclusivamente operazioni lineari per poter rispettare le leggi della meccanica quantistica. I gate quantistici per un singolo qubit sono matrici 2x2 unitarie ($U^\dagger U = I$), è possibile dimostrare che ogni matrice unitaria può essere considerata un gate quantistico. Iniziamo con il definire una controparte quantistica al gate NOT classico (spesso chiamato gate X). Esso dovrà invertire $|0\rangle$ e $|1\rangle$ in modo che

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{1.11}$$

É quindi possibile definire X come una matrice nel seguente modo:

$$X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1.12}$$

Così definito il NOT gate non è altro che una rotazione di π intorno all'asse $\hat{x}$ della sfera di Bloch. Altri due importanti gate sono: il gate Z e l'Hadamard gate (H). Il gate Z lascia invariato lo stato $|0\rangle$ e inverte il segno allo stato $|1\rangle$ facendolo diventare $-|1\rangle$. Nella sfera di Bloch questo corrisponde ad una rotazione di π rispetto all'asse $\hat{z}$. Può essere descritto dalla matrice:

$$Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{1.13}$$

Notiamo che questo gate inverte la fase del qubit. L'Hadamard gate è uno dei più importanti ed è utilizzato per portare il qubit in una sovrapposizione di stati. É definito dalla matrice:

$$H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.14}$$

e permette di trasformare lo stato $|0\rangle$ in $|+\rangle$ e lo stato $|1\rangle$ in $|-\rangle$. Nella sfera di Bloch questo gate corrisponde ad una rotazione di $\frac{\pi}{2}$ intorno all'asse $\hat{y}$ seguita da una rotazione di π intorno all'asse $\hat{x}$. La forma più generale di un gate a singolo qubit è la matrice unitaria:

$$U(\theta, \phi, \lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{1.15}$$

Partendo dal gate U è possibile definire le rotazioni di un angolo θ attorno ai tre assi $\hat{x}, \hat{y}, \hat{z}$ in questo modo:

$$R_x(\theta) = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} = u(\theta, -\pi/2, \pi/2) \tag{1.16}$$

$$R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ -\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} = u(\theta, 0, 0) \tag{1.17}$$

$$R_z(\phi) = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix} = u(0, 0, \phi) \tag{1.18}$$

É possibile definire gate che agiscono su più qubit. Senza dilungarci troppo, definiamo il CNOT gate equivalente quantistico del XOR, il quale verrà usato spesso in questa tesi essendo parte fondamentale dell'algoritmo QAOA. La porta Controlled-NOT (CNOT) in Figura 2 agisce su due qubit, il primo è chiamato qubit di *controllo* mentre il secondo qubit *target*. Se il controllo si trova nello stato $|0\rangle$ il target è lasciato inalterato, se il controllo è nello stato $|1\rangle$ il target viene invertito. Se prendiamo il BPS (Bit più significativo) come qubit di controllo la matrice apparirà come segue:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.19}$$

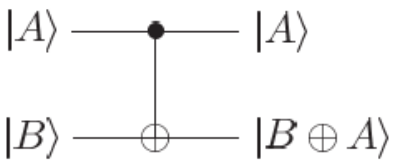

Figura 2: Controlled-NOT gate

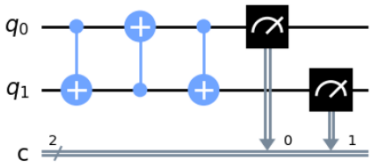

Figura 3: Circuito utilizzato per scambiare due qubit

Adesso abbiamo tutti gli strumenti per poter poter parlare di algoritmi quantistici.

1.1.4 Circuiti quantistici

Guardiamo nel dettaglio gli elementi di un circuito, possiamo analizzare un esempio di circuito in Figura 3 dove gli stati dei due qubit q_0 e q_1 vengono scambiati. Generalmente, i qubit vengono configurati inizialmente nello stato $|0\rangle$. Nonostante ciò, esiste la possibilità di inizializzarli impiegando una base differente. Per effetturare lo scambio tra qubit vengono applicati tre CNOT che eseguono la seguente operazione:

$$
\begin{aligned}
|a, b\rangle &\rightarrow |a, a \oplus b\rangle \\
&\rightarrow |a \oplus (a \oplus b), a \oplus b\rangle \\
&= |b, a \oplus b\rangle \\
&\rightarrow |b, (a \oplus b) \oplus b\rangle \\
&= |b, a\rangle
\end{aligned}
$$

Alla fine i due qubit vengono misurati e il loro valore viene inserito in bit classici. Ogni linea che si può vedere nel circuito è detta *filo*, esse non corrispondono a veri e propri fili ma indicato il passaggio del tempo e quindi quale operazione viene eseguita prima e quale dopo. É importante notare che nei circuiti quantistici non sono permessi cicli, vengono infatti detti *aciclici*, cioè non è possibile utilizzare un feedback da una parte del circuito quantistico a un'altra. In secondo luogo, nei circuiti classici è possibile 'unire' diversi fili, mediante un'operazione nota come FANIN. Questa operazione genera un unico filo che contiene la funzione OR applicata bit a bit agli input. È evidente che tale operazione non è reversibile e, pertanto, non unitaria; di conseguenza, non possiamo implementare il FANIN nei nostri circuiti quantistici. In terzo luogo, l'operazione inversa, chiamata FANOUT, che consiste nella produzione di molteplici copie di un bit, non è consentita nei circuiti quantistici. Questa limitazione deriva direttamente dal teorema del *no cloning*, un principio fondamentale della meccanica quantistica, il quale afferma che *non esiste una trasformazione unitaria, denotata con M, che, applicata a ogni stato* $|\psi\rangle|0\rangle$*, lo trasformi in* $|\psi\rangle|\psi\rangle$. Di conseguenza, l'operazione di FANOUT, simile al processo di clonazione, è preclusa nei circuiti quantistici. É possibile dimostrare il teorema supponendo di possedere una macchina quantistica con due qubit, identificati come A e B. Il qubit A, da noi definito come il qubit dei dati, si trova inizialmente in uno stato quantistico puro, ma sconosciuto, che indicheremo con $|\psi\rangle$. Questo è lo stato che abbiamo intenzione di replicare sul qubit B, che indicheremo come il qubit bersaglio. Supponiamo che il qubit bersaglio parta da uno stato puro standard, indicato con $|s\rangle$. Pertanto, lo stato iniziale del nostro dispositivo di clonazione risulta essere $|\psi\rangle \otimes |s\rangle$. Supponiamo che esista una trasformazione unitaria U che possa svolgere l'operazione di copia:

$$|\psi\rangle \otimes |s\rangle \xrightarrow{U} U|\psi\rangle \otimes |s\rangle = |\psi\rangle \otimes |\psi\rangle \tag{1.20}$$

Immaginiamo ora che il nostro processo di clonazione funzioni perfettamente per due specifici stati puri, $|\psi\rangle$ e $|\phi\rangle$. Allora avremo che:

$$\begin{aligned} U|\psi\rangle \otimes |s\rangle &= |\psi\rangle \otimes |\psi\rangle \\ U|\phi\rangle \otimes |s\rangle &= |\phi\rangle \otimes |\phi\rangle \end{aligned} \tag{1.21}$$

Svolgendo il prodotto scalare di queste due equazioni, otteniamo $\langle\psi|\phi\rangle = (\langle\psi|\phi\rangle)^2$. Ma $x = x^2$ ha solamente due soluzioni possibili: $x = 0$ e $x = 1$. Di conseguenza, o $|\psi\rangle$ è uguale a $|\phi\rangle$, o $|\psi\rangle$ e $|\phi\rangle$ sono stati ortogonali. Questo implica che un dispositivo di clonazione può replicare soltanto stati che sono ortogonali tra loro, il che rende impraticabile la costruzione di un dispositivo di clonazione quantistica universale.

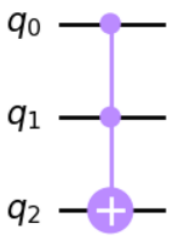

Figura 4: Gate di Toffoli

1.2 Algoritmi quantistici

1.2.1 Computazione classica su un computer quantistico

Un principio chiave che distingue i circuiti tradizionali da quelli quantistici è che, mentre le porte logiche classiche possono essere irreversibili, le porte logiche quantistiche sono unitarie e quindi reversibili. Per garantire un funzionamento reversibile, è indispensabile che la funzione in questione sia biettiva. Ciascun circuito classico può essere convertito in un circuito equivalente che incorpora esclusivamente elementi reversibili attraverso l'impiego di un gate noto come *gate di Toffoli*. Quest'ultimo è caratterizzato da tre bit di ingresso e tre bit di uscita, come si può vedere nella Figura 4, la tabella di verità del gate Toffoli è:

Inputs			Outputs		
a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Due di questi bit fungono da bit di controllo, rimanendo immutati indipendentemente dall'azione del gate di Toffoli. Il terzo bit, detto bit target, subisce un'inversione qualora en-

trambi i bit di controllo siano impostati su $|1\rangle$, altrimenti rimane inalterato. Applicando due volte il gate di Toffoli otteniamo lo stato iniziale verificando che l'operazione è reversibile $(a, b, c) \rightarrow (a, b, c \oplus ab) \rightarrow (a, b, c)$. Il gate di Toffoli assume un ruolo fondamentale nella computazione classica reversibile, dato che è possibile costruire qualsiasi calcolo classico in maniera reversibile utilizzando il gate di Toffoli. Questa affermazione si fonda sull'universalità delle operazioni NAND e FANOUT nei calcoli classici, e sulla capacità di implementare entrambe queste operazioni attraverso il gate di Toffoli. Ad esempio, se si esegue l'operazione con $c = 1$, si ottiene $a' = a$, $b' = b$ e $c' = 1 \oplus ab = \neg ab$, rappresentando così la simulazione reversibile del NAND. Analogamente, il FANOUT reversibile si può ottenere applicando il gate di Toffoli con $a = 1$ e $c = 0$, consentendo così la copia del bit b. Similmente a NAND e FANOUT, per costruire un circuito reversibile per un'operazione classica arbitraria f tramite il gate di Toffoli, è necessario utilizzare alcuni bit di servizio sia in input che in output. Una volta rimossi questi bit di servizio, il circuito risultante compie la trasformazione $(x, y) \mapsto (x, y \oplus f(x))$, dove x è l'input di f e y è il registro destinato a contenere l'output, e può essere considerato come il circuito reversibile standard per l'implementazione di f.

1.2.2 Parallelismo quantistico

Un computer quantistico consente di valutare una funzione $f(x)$ su vari valori di x simultaneamente. Questo concetto è noto come *parallelismo quantistico*, ed è una caratteristica fondamentale dei circuiti quantistici. Consideriamo, ad esempio, una funzione booleana della forma $f(x) : \{0, 1\} \rightarrow \{0, 1\}$. Per calcolare $f(x)$ tramite un calcolo quantistico, è necessario definire la trasformazione $f(x)$ come una trasformazione unitaria U_f. Questo si realizza applicando una sequenza appropriata di porte logiche quantistiche (che indicheremo come una "scatola nera" denominata U_f) sullo stato di input $|x, y\rangle$, conosciuto come *registro dei dati*. Questo trasforma $|x, y\rangle$ nello stato $|x, y \oplus f(x)\rangle$, che chiamiamo *registro target*. Se $y = 0$, allora lo stato finale del secondo qubit rappresenterà precisamente il valore di $f(x)$. Ipotizziamo che l'input sia $\frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes |0\rangle$, applicando U_f a questo registro dei dati, si ottiene $\frac{|0,f(0)\rangle+|1,f(1)\rangle}{\sqrt{2}}$. Questo stato contiene informazioni sia su $f(0)$ che su $f(1)$, dunque abbiamo valutato f contemporaneamente su $x = 0$ e $x = 1$. Questo processo può essere generalizzato per calcolare funzioni su un numero arbitrario di bit, usando una generalizzazione della porta di Hadamard nota come trasformata di Walsh-Hadamard. Questa operazione implica n porte di Hadamard che agiscono simultaneamente su n qubit. L'effetto di applicare la trasformazione di Hada-

mard a n qubit, inizialmente nello stato $|0\rangle$, risulta essere $\sqrt{\frac{1}{2^n}} \sum |x\rangle$ dove la sommatoria si estende a tutti i valori possibili di x. Indichiamo questa operazione con $H^{\otimes n}$. In altre parole, la trasformazione di Hadamard genera una sovrapposizione equiprobabile di tutti gli stati di base computazionali. Essa realizza questo processo con estrema efficienza, creando una sovrapposizione di 2^n stati facendo ricorso a soltanto n gate. La procedura per la valutazione quantistica parallela di una funzione, che ha un input x di n bit e un output di 1 bit, $f(x)$, può essere sintetizzata nel modo seguente: si inizia preparando uno stato di $n + 1$ qubit $|0\rangle^{\otimes n}|0\rangle$, poi si applica la trasformazione di Hadamard ai primi n qubit, seguita dall'applicazione del circuito quantistico che implementa U_f. Questa serie di operazioni conduce allo stato

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle|f(x)\rangle. \tag{1.22}$$

Il parallelismo quantistico ci dà la possibilità di valutare in modo simultaneo tutti i possibili valori della funzione f, benché possa apparire come se avessimo valutato f una sola volta. Tuttavia, questo parallelismo non fornisce un beneficio immediato. Per esempio, nel nostro caso con un singolo qubit, la misurazione dello stato fornisce solo $|0, f(0)\rangle$ o $|1, f(1)\rangle$. Allo stesso modo, nel caso generale, la misurazione dello stato

$$\sum_x |x, f(x)\rangle \tag{1.23}$$

risulterà in $f(x)$ per un solo valore di x. Questo principio trova applicazione nell'algoritmo di Deutsch-Jozsa, che sarà dettagliatamente illustrato nel paragrafo successivo. Grazie a questo algoritmo, è possibile determinare se una funzione è bilanciata o costante eseguendo un'unica iterazione.

1.2.3 Esempi di algoritmi quantistici

Due degli algoritmi molto famosi sono l'algoritmo di Deutsch e l'algoritmo di Deutsch-Jozsa, rispettivamente in Figura 5 e 6. L'algoritmo di Deutsch [5], proposto per la prima volta da David Deutsch nel 1985, è fondato sulla nozione di interferenza quantistica. Consideriamo una funzione booleana $f : \{0, 1\} \to \{0, 1\}$. Essa può essere o costante (il suo valore è sempre 0 o sempre 1) o bilanciata (il suo valore è 0 per una metà dei suoi input e 1 per l'altra metà). In un computer classico, dovremmo valutare la funzione due volte per determinare con certezza se è costante o bilanciata. Tuttavia, l'algoritmo di Deutsch ci permette di determinare se f è costante o bilanciata eseguendo una sola valutazione della funzione. Si prepara un registro quantistico in uno stato di superposizione degli stati base 0 e 1. Successivamente si applica un'operazione chiamata oracolo quantistico, che implementa la funzione f. Infine, si misura il primo registro. Durante questa fase, avviene un processo chiamato collasso della funzione d'onda. L'interferenza quantistica fra gli stati di superposizione nel nostro sistema fa sì che, nonostante stiamo misurando solo un singolo qubit, otteniamo informazioni globali sulla funzione f. In particolare, se la funzione f è costante, la misurazione darà come risultato 0. Se, invece, la funzione f è bilanciata, la misurazione darà come risultato 1. Questo avviene perché gli stati in cui l'output di f è diverso interferiscono tra loro in modo da cancellarsi a vicenda, lasciando solo gli stati che rivelano se f è costante o bilanciata. L'algoritmo di Deutsch-Jozsa [9], invece, viene utilizzato per funzioni booleane su n bit. Considerando una funzione f tale che $f : \{0, 1\}^n \to \{0, 1\}$, e supponendo che f possa essere o costante oppure bilanciata - ovvero assume valore 0 su esattamente metà degli input e valore 1 sulla restante metà - con un algoritmo classico, nel caso peggiore, avremmo bisogno di valutare la funzione su almeno $2^{n-1} + 1$ valori per poter stabilire con certezza se f è costante o bilanciata. Tuttavia, l'approccio quantistico dell'algoritmo di Deutsch-Jozsa permette di determinare questa proprietà in un solo passo.

Altri algoritmi quantistici famosi sono: l'algoritmo di Shor per la fattorizzazione di numeri interi, il quale si basa sulla trasformata di Fourier quantistica e sul principio di interferenza quantistica, o l'algoritmo di Grover per la ricerca in un database non strutturato, in grado di trovare un elemento in un database non strutturato con N elementi in $\sqrt{N}$ passaggi rispetto agli N passaggi necessari per un algoritmo classico.

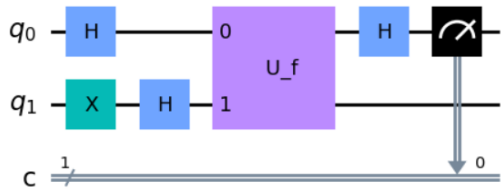

Figura 5: Algoritmo di Deutsch

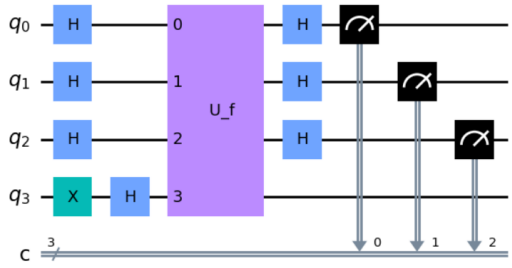

Figura 6: Algoritmo di Deutsch-Jozsa

1.3 Quantum Approximate Optimization Algorithm

1.3.1 Principio variazionale

Quando studiamo un sistema meccanico quantistico, siamo spesso particolarmente interessati allo stato fondamentale (o *ground state*). Questo è lo stato con l'energia più bassa possibile, o equivalentemente, l'autostato dell'Hamiltoniana con l'autovalore più basso possibile. Nella maggior parte dei sistemi a cui siamo interessati non è possibile calcolare lo stato fondamentale analiticamente, dobbiamo quindi ricorrere ad alcune tecniche per poter trovare una soluzione approssimata. Il principio variazionale ci permette di trovare un limite superiore all'energia dello stato fondamentale. Il principio variazionale si basa su una procedura molto semplice:

- Inizialmente, si formula un'ipotesi preliminare (o *ansatz*) riguardo una famiglia di stati fondamentali (normalizzati), caratterizzati da un parametro θ. Questi stati possono essere indicati come $|\psi(\theta)\rangle$.
- Successivamente, il principio variazionale [10] afferma che l'energia del vero stato fondamentale, indicata con E_0, è sempre minore o uguale al valore atteso dell'Hamiltoniana per qualsiasi stato nella famiglia $|\psi(\theta)\rangle$:

$$E_0 \leq \langle\psi(\theta)|\hat{H}|\psi(\theta)\rangle \tag{1.24}$$

- Si procede quindi a far variare il parametro θ in modo da minimizzare l'energia e trovare un limite superiore più accurato:

$$E_{min} = \min_{\theta} \langle \psi(\theta) | \hat{H} | \psi(\theta) \rangle \geq E_0 \tag{1.25}$$

Possiamo chiamare θ^* l'angolo che minimizza l'energia della famiglia di stati, $|\psi(\theta^*)\rangle$ il corrispondente stato che sarà un'approssimazione dello stato fondamentale $|\psi^*\rangle$. Per dimostrare la disuguaglianza del principio variazionale possiamo scrivere gli autovalori dell'Hamiltoniana come $|n\rangle$. Non sappiamo quali siano questi stati, ma sappiamo che esistono. Possiamo quindi utilizzare una risoluzione dell'identità $I = \sum_n |n\rangle\langle n|$ per scrivere:

$$\begin{aligned}
\hat{H}|n\rangle &= E_n|n\rangle \\
\langle\psi|\hat{H}|\psi\rangle &= \sum_n \langle\psi|\hat{H}|n\rangle\langle n \mid \psi\rangle \\
&= \sum_n E_n \langle\psi \mid n\rangle\langle n \mid \psi\rangle \\
&= \sum_n E_n |\langle n \mid \psi\rangle|^2 \\
&= \sum_n E_0 |\langle n \mid \psi\rangle|^2 + \sum_n (E_n - E_0) |\langle n \mid \psi\rangle|^2 \\
&\geq E_0 \sum_n |\langle n \mid \psi\rangle|^2 \\
&= E_0 \sum_n \langle\psi \mid n\rangle\langle n \mid \psi\rangle \\
&= E_0 \langle\psi \mid \psi\rangle \\
\Rightarrow \langle\psi|\hat{H}|\psi\rangle &\geq E_0.
\end{aligned}$$

Questa relazione è valida per ogni stato $|\psi\rangle$. Il punto debole principale del principio variazionale è che la qualità con cui si approssima lo stato fondamentale dipende dalla qualità della nostra ipotesi preliminare. Una scelta possibile per migliorare il nostro risultato è di parametrizzare il nostro stato con un circuito quantistico variazionale.

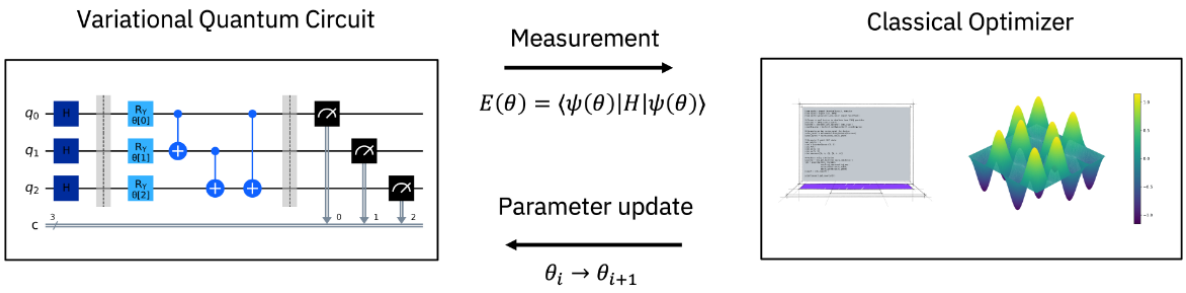

Figura 7: Variational Quantum Eigensolvers

1.3.2 Algoritmi quantistici variazionali

Un circuito variazionale è un circuito che dipende da alcuni parametri θ_i. In Figura 7 è possibile osservare un esempio di circuito parametrico di tre qubit sulla sinistra. É possibile utilizzare questo tipo di circuito per preparare un ansatz. L'idea dietro il Variational Quantum Eigensolvers (VQE) è quella di preparare uno stato quantistico con un circuito variazionale e di aggiornare i parametri calcolando su un computer quantistico il valore dell'energia per quello specifico ansatz $E(\theta)$ e successivamente passare questo valore ad un ottimizzatore classico che aggiornerà i parametri. In questo modo otteniamo un feedback loop ibrido che ci permette di approssimare il ground state dell'Hamiltoniana. Questo algoritmo è molto utilizzato in chimica o fisica quantistica ma può essere utilizzato anche per problemi di ottimizzazione classica codificando il problema di ottimizzazione in un'Hamiltoniana il cui ground state corrisponde con la soluzione del problema. Questa è l'idea utilizzata anche nel QAOA che verrà presentato successivamente.

1.3.3 MaxCut

Il problema del MaxCut è un noto problema di ottimizzazione combinatoria nella teoria dei grafi, che ha trovato applicazioni in una vasta gamma di settori, tra cui la progettazione di circuiti, la statistica e la fisica teorica.

Per prima cosa definiamo cos'è un grafo. Un *grafo* G è una struttura matematica definita come una coppia ordinata $G = (V, E)$, dove:

- V è un insieme non vuoto di *vertici* (o nodi).

- E è un insieme di coppie ordinate (per i grafi diretti o orientati) o non ordinate (per i grafi non diretti o indiretti) di vertici, noti come *archi*.

Un *grafo pesato* è un grafo in cui ogni arco ha un valore associato, noto come *peso*. Formalmente, un grafo pesato può essere definito come una tripla ordinata $G = (V, E, w)$, dove:

- V è un insieme non vuoto di vertici.
- E è un insieme di coppie ordinate o non ordinate di vertici.
- $w : E \to \mathbb{R}$ è una funzione che assegna un peso reale a ogni arco.

In termini semplici, il problema MaxCut richiede di dividere i nodi di un grafo in due gruppi in modo tale da massimizzare il numero di archi che connettono i due gruppi. In altre parole, si cerca di trovare il "taglio" che separa il maggior numero possibile di archi. Formalmente, dato un grafo non diretto G = (V, E), un taglio è una partizione di V in due sottoinsiemi. Il peso del taglio è la somma dei pesi degli archi che hanno un'estremità in ciascun sottoinsieme. Nonostante la sua apparente semplicità, il problema MaxCut è NP-hard, il che significa che non esiste un algoritmo noto che possa risolverlo in tempo polinomiale per tutti i grafi. Tra le varie tecniche utilizzate per affrontare il problema MaxCut, l'algoritmo di Goemans-Williamson [11] (che verrà presentato in dettaglio nell'Appendice A) si distingue per l'uso di un metodo chiamato 'arrotondamento casuale dell'iperpiano'. Questa tecnica consente di convertire la soluzione del problema di programmazione semidefinita, che è una matrice, in una soluzione del problema originale, che è un taglio del grafo. L'efficacia di questo metodo ha permesso all'algoritmo di Goemans-Williamson di ottenere un rapporto di approssimazione garantito di $0.87856 < \alpha < 0.87857$, il migliore conosciuto per il problema MaxCut [11, 12]. Il rapporto di approssimazione è definito come il rapporto tra la somma dei pesi degli archi che attraversano il cut ottenuto dall'algoritmo e la somma massima possibile dei pesi degli archi che potrebbero attraversare un cut:

$$\alpha = \frac{C}{C_{\text{max}}} \tag{1.26}$$

Dove C rappresenta la somma dei pesi degli archi nel cut ottenuto dall'algoritmo e C_{max} rappresenta la somma massima possibile dei pesi degli archi in un cut. Questo rapporto, α, ci fornisce una misura di quanto sia buona l'approssimazione fornita dall'algoritmo rispetto alla soluzione ottima in un grafo pesato.

La Congettura dei Giochi Unici (Unique Games Conjecture, o UGC) [13, 14] suggerisce che l'algoritmo di Goemans-Williamson abbia la migliore prestazione possibile in tempo polinomiale sul MaxCut [15]. Questa affermazione deriva dall'importanza fondamentale dell'UGC nel campo della teoria della complessità computazionale, in particolare in relazione alla complessità di approssimazione. Formulata da Subhash Khot nel 2002 [13], la UGC coinvolge problemi di ottimizzazione combinatoria detti *giochi unici*, in cui si ha un insieme di variabili, ciascuna delle quali può assumere valori da un insieme finito di opzioni, e un insieme di vincoli, ciascuno dei quali specifica che due variabili devono avere valori correlati in un certo modo. L'obiettivo è trovare l'assegnazione di valori che soddisfi il maggior numero possibile di vincoli. La UGC sostiene che, in un gioco unico, è estremamente complesso trovare una soluzione approssimata che soddisfi una frazione elevata dei vincoli, anche nel caso in cui esista una soluzione che soddisfi quasi tutti i vincoli. Per essere più precisi, la congettura propone che, per ogni numero reale positivo ε, esiste un intero k tale che è NP-difficile distinguere tra i due seguenti scenari per un gioco unico con alfabeto di dimensione k: 1) esiste una soluzione che soddisfa almeno $1-\varepsilon$ dei vincoli, oppure 2) ogni soluzione può soddisfare al massimo una frazione ε dei vincoli. Se questa congettura si rivelasse vera, avrebbe profonde implicazioni per una vasta gamma di problemi di ottimizzazione. Inoltre, se si riuscisse a trovare un algoritmo con performance migliore di quello proposto da Goemans-Williamson, ciò implicherebbe che P = NP, sconvolgendo completamente la nostra comprensione attuale della complessità computazionale.

Più recentemente, l'attenzione si è rivolta verso l'uso di tecniche di ottimizzazione quantistica, come il Quantum Approximate Optimization Algorithm (QAOA), per risolvere il problema MaxCut. Questi metodi sfruttano le proprietà uniche dei sistemi quantistici per cercare soluzioni in uno spazio di soluzioni molto più ampio rispetto a quello accessibile con metodi classici.

Iniziamo considerando un grafo pesato $G = (V, E)$, composto da n vertici. Per facilitare la nostra analisi, numeriamo questi vertici come $v_1, v_2, \ldots, v_n$. Ad ogni vertice può essere assegnato un valore di 0 o 1, che può rappresentare, ad esempio, su quale lato di un taglio si trova il vertice. In questo modo, possiamo rappresentare qualsiasi taglio del grafo come un vettore x in $\{0,1\}^n$, dove la dimensione n del vettore corrisponde al numero di vertici nel grafo. Il peso totale di un taglio del grafo viene calcolato attraverso la funzione di costo definita come:

$$C(x) = \sum_{i,j=1}^{n} W_{ij} x_i (1 - x_j) \tag{1.27}$$

In questa equazione, $W_{ij} = W_{ji}$ rappresenta il peso del lato che collega i vertici v_i e v_j. Se i vertici v_i e v_j non sono collegati da un lato, allora $W_{ij} = 0$. Notiamo che il peso di un lato contribuisce alla somma totale nella funzione di costo solo se i due vertici che collega si trovano su lati opposti del taglio, cioè se $x_i \neq x_j$. Questo accade perché solo in questo caso uno dei fattori nella moltiplicazione $x_i(1 - x_j)$ sarà 1 e l'altro sarà 0, dando come risultato 0 se i due vertici sono sullo stesso lato (entrambi 0 o entrambi 1) e il peso del lato se sono su lati opposti (uno 0 e l'altro 1). Quindi, la funzione di costo $C(x)$ calcola efficacemente il peso totale di un dato taglio nel grafo.

1.3.4 Quadratic Unconstrained Binary Optimization (QUBO)

La Quadratic Unconstrained Binary Optimization (QUBO) è una forma di problema di ottimizzazione che si presenta frequentemente in informatica, matematica applicata e in vari campi dell'ingegneria. In un problema QUBO, l'obiettivo è minimizzare una funzione obiettivo che è una forma quadratica definita su variabili binarie con vincoli lineari e quadratici. In altre parole preso $x = (x_1, x_2, \ldots, x_n)$, può essere scritto nella forma:

$$\begin{aligned} &\text{minimizza} && x^T Q x + c^T x \\ &\text{soggetto a} && Ax \leq b \\ & && x^T Q_i x + a_i^T x \leq r_i \\ & && l_j \leq x_j \leq u_j \end{aligned}$$

dove $Q \in \mathbb{R}^{n \times n}$ è una matrice simmetrica con valori reali $n \times n$ e $c \in \mathbb{R}^n$ è un vettore reale che definisce la parte quadratica e lineare della funzione obiettivo. Le variabili di ottimizzazione x_i con $i \in \{1, \ldots, n\}$ possono essere binarie, intere o reali in base al problema che si sta cercando di risolvere, nel nostro caso ci concentreremo su variabili binarie. Esistono varie tecniche euristiche [16] e approssimate per risolvere problemi QUBO, tra cui metodi di raffreddamento simulato [17], algoritmi genetici, e più recentemente, tecniche di ottimizzazione quantistica [18]. In particolare, i problemi QUBO sono di grande interesse nel campo della computazione quantistica, in quanto molti problemi di ottimizzazione possono essere riformulati come problemi QUBO e quindi risolti utilizzando algoritmi quantistici come il QAOA. Possiamo formulare qualsiasi problema MaxCut come un programma quadratico. Consideriamo di nuo-

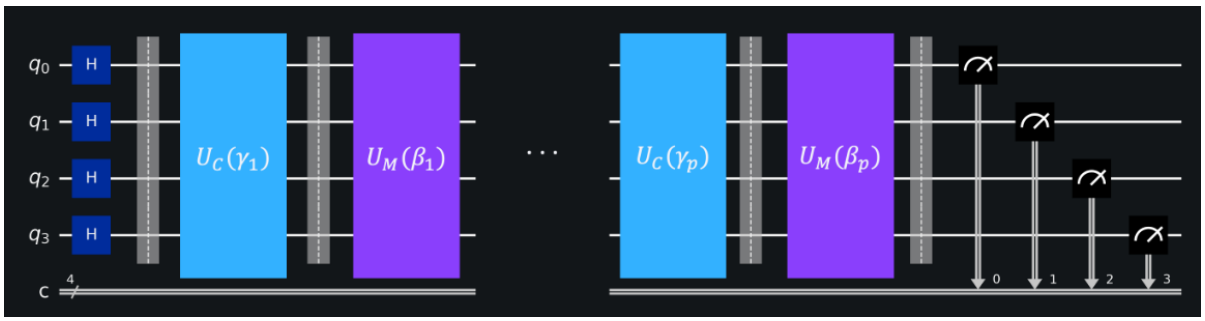

Figura 8: Esempio di circuito per il QAOA, presa da [2]

vo la funzione di costo per un problema MaxCut con matrice dei pesi simmetrica W Equazione (1.27). Questa è chiaramente una funzione di costo quadratico e possiamo riformularla nella forma standard per i programmi quadratici come scritto sopra:

$$\begin{aligned}\sum_{i,j=1}^{n} W_{ij}x_i(1-x_j) &= \sum_{i,j=1}^{n} W_{ij}x_i - W_{ij}x_ix_j \\ &= \sum_{i=1}^{n}\left(\sum_{j=1}^{n} W_{ij}\right)x_i - \sum_{i,j=1}^{n} W_{ij}x_ix_j \\ &= c^T x + x^T Q x,\end{aligned}$$

per Q e c definiti come segue:

$$Q_{ij} = -W_{ij} \qquad c_i = \sum_{j=1}^{n} W_{ij}. \tag{1.28}$$

Quindi, otteniamo un'istanza di ottimizzazione con variabili binarie, una funzione obiettivo quadratica e senza alcun vincolo di variabile.

1.3.5 Circuito per il QAOA

L'Algoritmo di Ottimizzazione Approssimata Quantistica [20] (QAOA, Quantum Approximate Optimization Algorithm) è un algoritmo quantistico sviluppato per risolvere problemi di ottimizzazione combinatoria. Questo algoritmo è un esempio di algoritmo variazionale quantistico, che sfrutta l'interazione tra un computer classico e uno quantistico per trovare la soluzione di un problema. Il QAOA opera applicando una serie di trasformazioni quantistiche a un gruppo di qubit, che sono inizialmente preparati in uno stato di base. Queste trasformazioni sono regolate da un insieme di parametri che possono essere ottimizzati per minimizzare l'energia dello stato finale dei qubit. L'ottimizzazione di questi parametri viene effettuata su un computer classico, mentre l'applicazione delle trasformazioni quantistiche e la misurazione dello stato finale dei qubit sono compiti che vengono eseguiti su un computer quantistico. Il QAOA, come algoritmo ibrido, sfrutta il meglio di entrambi i mondi computazionali. L'uso di un computer quantistico permette di esplorare un ampio spazio di soluzioni, potenzialmente conducendo a soluzioni migliori o più velocemente raggiungibili rispetto agli algoritmi classici per problemi di ottimizzazione combinatoria. Uno degli obiettivi principali della computazione quantistica, e in particolare di algoritmi come il QAOA, è di risolvere problemi che sono intrattabili per i computer classici. In teoria, gli algoritmi quantistici possono risolvere certi problemi di ottimizzazione in un tempo che scala in modo polinomiale con la dimensione del problema, mentre gli algoritmi classici migliori conosciuti richiedono un tempo che scala esponenzialmente. Tuttavia, a causa delle limitazioni attuali della tecnologia quantistica, questa promessa non è ancora stata realizzata nella pratica.

Nella Figura 8 è possibile vedere un esempio di circuito per il QAOA. In questa figura, i blocchi QAOA sono indicati come U_C e U_M. $U_C = e^{-i\gamma H_C}$ rappresenta l'esponenziale di matrice dell'Hamiltoniana di costo H_C, mentre $U_M = e^{-i\beta H_M}$ rappresenta l'esponenziale di matrice dell'Hamiltoniana di mix H_M. È importante sottolineare che questi esponenziali di matrice rappresentano operatori di evoluzione temporale in meccanica quantistica, i quali determinano l'evoluzione di un sistema nel tempo in base all'Hamiltoniana del sistema. La funzione costo del problema di ottimizzazione viene codificata in un'Hamiltoniana H_C [21] mentre H_M è l'Hamiltoniana di mix che permette di esplorare lo spazio delle soluzioni. Viene successivamente applicato il principio variazionale visto in precedenza per trovare un'approssimazione del ground state che corrisponde alla migliore soluzione del problema. La tecnica si basa sul trotterized adiabatic process che verrà spiegato nel paragrafo successivo. Nella versione

originale di QAOA, lo stato iniziale preparato è lo stato di sovrapposizione $|+\rangle$:

$$|+\rangle = \sum_{x\in\{0,1\}^n} \frac{1}{\sqrt{2^n}}|x\rangle \tag{1.29}$$

e l'Hamiltoniana del mixer è, solitamente, definita come la somma dei singoli operatori Pauli X su tutti i qubit

$$H_M = \sum_{i=1}^{n} X_i. \tag{1.30}$$

In questo contesto, è fondamentale sottolineare che l'Hamiltoniana del mixer varia in funzione del problema specifico che si intende risolvere. Inoltre, un aspetto cruciale nella scelta di questa Hamiltoniana consiste nella necessità che non commuti con l'Hamiltoniana di costo $[H_C, H_M] \neq 0$. Se le due Hamiltoniane commutassero, infatti, la nostra capacità di esplorare in maniera completa l'intero spazio delle soluzioni verrebbe compromessa. Per un'istanza di QUBO con matrice corrispondente Q, vorremmo codificare la funzione di costo $C(x)$ come l'Hamiltoniana di costo H_C, cioè stiamo cercando di trovare un operatore H_C, tale che

$$H_C|x\rangle = \left(x^T Q x + c^T x\right)|x\rangle = \left(\sum_{i,j=1}^{n} x_i Q_{ij} x_j + \sum_{i=1}^{n} c_i x_i\right)|x\rangle \tag{1.31}$$

per tutti gli stati di base computazionali $x \in \{0,1\}^n$. Utilizzando il fatto che

$$Z_i|x\rangle = (-1)^{x_i}|x\rangle = (1-2x_i)|x\rangle \implies x_i|x\rangle = \frac{\mathbb{I} - Z_i}{2}|x\rangle, \tag{1.32}$$

dove $\mathbb{I}$ denota l'operatore identità e Z_i la matrice di Pauli-Z per il qubit i, possiamo scrivere H_C come una somma di operatori Pauli-Z, effettuando la sostituzione $x_i \to \frac{\mathbb{I}-Z_i}{2}$ nell'espressione della funzione di costo $C(x)$. Questo fornisce la seguente espressione per H_C:

$$H_C = \sum_{i,j=1}^{n} \frac{1}{4} Q_{ij} Z_i Z_j - \sum_{i=1}^{n} \frac{1}{2}\left(c_i + \sum_{j=1}^{n} Q_{ij}\right) Z_i + \left(\sum_{i,j=1}^{n} \frac{Q_{ij}}{4} + \sum_{i=1}^{n} \frac{c_i}{2}\right) \tag{1.33}$$

Applicando l'operatore esponenziale a entrambe le Hamiltoniane, otteniamo una forma variazionale costituita da gate R_Z e R_{ZZ} nel layer di costo e gate R_X nel layer del mixer:

$$e^{-i\gamma H_C} = \prod_{i,j=1 i\neq j}^{n} R_{Z_i Z_j}\left(\frac{1}{2}Q_{ij}\gamma\right)\prod_{i=1}^{n} R_{Z_i}\left(\left(c_i + \sum_{j=1}^{n} Q_{ij}\right)\gamma\right) \tag{1.34}$$

$$e^{-i\beta H_M} = \prod_{i=1}^{n} R_{X_i}(2\beta) \tag{1.35}$$

É possibile applicare più di un layer di questi due operatori, il numero di layer viene chiamato profondità e viene spesso indicato con p. In linea teorica $p \to \infty$ assicura di trovare la soluzione al problema.

1.3.6 Teorema adiabatico e QAOA

Il teorema adiabatico in meccanica quantistica afferma che se un sistema quantistico si trova inizialmente nello stato fondamentale (o ground state) di un'Hamiltoniana e quest'ultima viene variata molto lentamente nel tempo, il sistema rimarrà nello stato fondamentale della nuova Hamiltoniana, assumendo che non vi siano degenerazioni. Questo significa che se l'evoluzione è abbastanza lenta, il sistema ha la possibilità di adattarsi continuamente ai cambiamenti dell'Hamiltoniana, mantenendosi nello stato fondamentale. Nel nostro caso possiamo applicare il teorema adiabatico in questo modo:

- Prima di tutto, codifichiamo il problema di ottimizzazione in un'Hamiltoniana di costo, che chiameremo H_C.
- In seguito, prepariamo il sistema nel ground state di un'Hamiltoniana iniziale H_0 che è più semplice e di cui possiamo facilmente preparare il ground state.
- Infine, facciamo evolvere adiabaticamente il sistema da H_0 ad H_C. In termini pratici, ciò significa che cambiamo lentamente l'Hamiltoniana del sistema da H_0 a H_C nel corso di un periodo di tempo sufficientemente lungo. Se l'evoluzione è abbastanza lenta, il teorema adiabatico ci garantisce che il sistema rimarrà nello stato fondamentale del-

l'Hamiltoniana corrente, dunque alla fine dell'evoluzione il sistema si troverà nel ground state di H_C.

In questo modo, se tutto va come previsto, alla fine dell'evoluzione adiabatica il sistema sarà nello stato fondamentale di H_C, che codifica la soluzione ottimale del problema che stiamo cercando di risolvere. Per evolvere il sistema utilizziamo la tecnica della trotterizzazione (dal nome del matematico britannico Hugh Trotter). Questa tecnica offre un modo per approssimare l'evoluzione dividendo l'Hamiltoniana in pezzi più semplici che sono più facili da gestire. Consideriamo l'Hamiltoniana $H(t) = H_C + H_M$, l'operatore di evoluzione temporale può essere scritto come $U(t) = e^{-\frac{iH(t)t}{\hbar}} = e^{-\frac{i(H_C+H_M)t}{\hbar}}$. Sappiamo che se due operatori A e B commutano vale la relazione $e^{A+B} = e^A e^B$. Possiamo quindi scrivere la formula di Trotter Suzuki [22] per approssimare l'esponenziale della somma di due matrici:

$$e^{-i(H_C+H_M)t} \simeq (e^{-\frac{iH_C t}{r}} e^{-\frac{iH_M t}{r}})^r \tag{1.36}$$

Questa approssimazione vale a meno di termini che dipendono da $[H_C, H_M] \neq 0$, i quali tendono a 0 per $r \to \infty$.

Consideriamo ora un'evoluzione adiabatica di tempo totale T della forma:

$$H(t) = \frac{t}{T} H_C + \left(1 - \frac{t}{T}\right) H_M \tag{1.37}$$

Se l'Hamiltoniana di mix è definita come Equazione (1.30) e partiamo dal ground state di questa Hamiltoniana possiamo finire nel ground state dell'Hamiltoniana di costo H_C. Gli autostati di H_M sono $|+\rangle$ e $-|-\rangle$ quindi nel QAOA vengono applicati degli Hadamard gate all'inizio del circuito per preparare i qubit nello stato di massima energia di H_M. I layer del QAOA corrispondono a segmenti della trotterizzazione dell'operatore evoluzione temporale. In questo modo finiremo nello stato di massima energia di H_C, cambiando di segno otterremo il ground state. Guardando il QAOA da questa prospettiva capiamo il perché per $p \to \infty$ abbiamo la sicurezza di trovare la soluzione al nostro problema. É importante notare che non è possibile in pratica avvicinarsi a $p = \infty$ perché all'aumentare di p aumentano le variabili da minimizzare e a p molto grandi ci troveremo una grandissima quantità di variabili da ottimizzare che è ovviamente impossibile.

Capitolo 2

Qiskit e QAOA: Dall'Analisi all'Implementazione

Questo capitolo si propone di esplorare in profondità il funzionamento e le capacità del Qiskit, la piattaforma per la programmazione quantistica utilizzata nelle simulazioni. Esamineremo le componenti chiave che definiscono Qiskit, fornendo una panoramica delle principali librerie che costituiscono la sua struttura. Successivamente, ci addentreremo nel dettaglio del codice che alimenta le simulazioni, soffermandoci in particolare sulla strategia del "parameter fixing", come delineato nel [23]. Concluderemo il capitolo affrontando le sfide che emergono quando si utilizzano computer quantistici reali per testare il Quantum Approximate Optimization Algorithm (QAOA).

2.1 Introduzione a Qiskit

Il Quantum Approximate Optimization Algorithm (QAOA) può essere implementato e sperimentato utilizzando vari framework software, ma uno strumento particolarmente popolare e potente in questo campo è Qiskit. Qiskit [24] è un framework di software open source in continua evoluzione sviluppato da IBM per l'informatica quantistica. Si tratta di una suite di librerie Python che consente agli sviluppatori e ai ricercatori di creare e ottimizzare programmi quantistici, simulare l'esecuzione di tali programmi su un computer quantistico e persino eseguirli su veri computer quantistici IBM attraverso il cloud. Qiskit si articola in cinque elementi principali: capacità fondamentali, applicazioni ad alto livello, applicazioni a basso livello, simulatori e fornitori di hardware. Le capacità fondamentali sono racchiuse nella libreria Qiskit Terra [25], la quale è la base su cui sono costruiti tutti gli altri componenti di Qiskit. Fornisce gli strumenti per definire, manipolare e ottimizzare i circuiti quantistici a basso livello e le istruzioni quantistiche.

Le applicazioni ad alto livello sono contenute in quattro librerie Qiskit Nature [26], Qiskit

Finance [27], Qiskit Machine Learning [28], Qiskit Optimization [29]. Qiskit Nature è un framework open source progettato per affrontare problemi derivanti dalle scienze naturali e dalla meccanica quantistica mediante l'uso di algoritmi di calcolo quantistico. Le sue capacità spaziano dalla determinazione degli stati fondamentali e degli stati eccitati per problemi legati alla struttura elettronica, alla misurazione dei momenti di dipolo in sistemi molecolari. Inoltre, può risolvere modelli complessi come quelli di Ising e Fermi-Hubbard su reticoli. Qiskit Finance è un framework open source che contiene componenti di incertezza per problemi di azioni/titoli, applicazioni per problemi finanziari, come l'ottimizzazione del portafoglio, e fornitori di dati per reperire dati reali o casuali per creare esperimenti nel campo della finanza. D'altra parte, Qiskit Optimization è un framework open-source che offre un'ampia gamma di funzionalità, dalla modellazione di alto livello dei problemi di ottimizzazione, con la capacità di convertire automaticamente i problemi in diverse rappresentazioni necessarie, a un insieme di algoritmi di ottimizzazione quantistica pronti per essere eseguiti sia su simulatori classici che su dispositivi quantistici reali attraverso Qiskit. Questi algoritmi vanno da algoritmi quantistici variazionali, come il QAOA, a Grover Adaptive Search implementato tramite GroverOptimizer, tutti sfruttando gli algoritmi fondamentali forniti da Terra. Inoltre, la struttura modulare del modulo di ottimizzazione consente una facile estensione e favorisce lo sviluppo e il testing rapido di nuovi algoritmi. Sono inoltre disponibili ottimizzatori classici compatibili per i test, la validazione e il benchmarking. Qiskit Machine Learning introduce elementi base del calcolo quantistico, quali Quantum Kernels e Quantum Neural Networks, utilizzabili in diverse applicazioni come classificazione e regressione. Il design intuitivo facilita l'uso da parte degli utenti e la prototipazione di modelli, senza la necessità di una profonda conoscenza del calcolo quantistico. Inoltre, è possibile estendere facilmente Qiskit Machine Learning per sostenere la ricerca nel campo del machine learning quantistico. Vengono forniti strumenti come la classe FidelityQuantumKernel, l'EstimatorQNN e il SamplerQNN per risolvere rapidamente problemi di classificazione o regressione. Inoltre, l'interfaccia permette di valutare le reti neurali per un dato input e di calcolare i gradienti corrispondenti, essenziali per l'addestramento. Qiskit Machine Learning fornisce anche una serie di algoritmi di apprendimento e consente di integrare le reti neurali quantistiche nella libreria PyTorch grazie al TorchConnector.

Le applicazioni di basso livello sono contenute nelle librerie Qiskit Metal [30], Qiskit Dynamics [31] e Qiskit Experiments [32]. Qiskit Metal è dedicata alla progettazione di qubit superconduttivi e al layout dei circuiti quantistici. Fornisce un ambiente grafico interattivo per la progettazione di componenti quantistici, consentendo ai ricercatori e agli ingegneri di progettare e analizzare la geometria e la fisica dei qubit e dei circuiti quantistici. Qiskit Dynamics fornisce strumenti per la simulazione e l'analisi delle dinamiche dei sistemi quantistici. Questa libreria consente di modellare e simulare l'evoluzione temporale dei sistemi quantistici, inclusi gli effetti del rumore e della dissipazione. È particolarmente utile per studiare la dinamica

dei qubit e dei circuiti quantistici nel tempo. Infine, Qiskit Experiments offre un quadro per l'esecuzione e l'analisi degli esperimenti quantistici. Questo strumento semplifica il processo di definizione, esecuzione e analisi degli esperimenti su computer quantistici reali o simulati. Include supporto per una varietà di esperimenti, tra cui la tomografia dei qubit, la calibrazione dei pulsanti e la misura dell'errore del gate.

I simulatori sono contenuti nella libreria Qiskit Aer [33], la quale offre simulazioni ad alta performance di circuiti quantistici, inclusi rumore e ottimizzazioni specifiche per macchine. Infine, Qiskit supporta una varietà di fornitori di hardware quantistico. IBM Quantum fornisce l'accesso a una gamma di sistemi quantistici basati su qubit superconduttori attraverso la libreria Qiskit Runtime [34]. AQT e IonQ offrono entrambi accesso a dispositivi quantistici basati su ioni intrappolati, utilizzando rispettivamente ioni di calcio e itterbio. IQM e Rigetti offrono accesso a sistemi quantistici basati su qubit superconduttori, con Rigetti che fornisce qubit superconduttori sintonizzabili. Infine, Quantinuum fornisce accesso a sistemi a ioni di itterbio intrappolati con qubit altamente fedeli e completamente connessi, e la capacità di eseguire misurazioni a metà circuito.

In conclusione, Qiskit emerge come uno strumento fondamentale per l'informatica quantistica e, in particolare, per l'implementazione e la sperimentazione del QAOA. La sua architettura modulare consente un controllo dettagliato e un'ottimizzazione accurata dei circuiti quantistici, fornendo nel contempo strumenti per gestire il rumore, simulare l'esecuzione e implementare una vasta gamma di algoritmi quantistici. La forza di Qiskit risiede non solo nella sua versatilità e potenza come framework di programmazione quantistica, ma anche nella sua comunità di sviluppatori e ricercatori. Questa comunità attiva e impegnata continua a spingere i limiti di ciò che è possibile con Qiskit, contribuendo con miglioramenti e innovazioni che mantengono Qiskit all'avanguardia del campo in rapida evoluzione dell'informatica quantistica. Per quanto riguarda il QAOA, Qiskit offre gli strumenti necessari per esplorare e ottimizzare questo algoritmo in modo efficace, consentendo sia la simulazione che l'esecuzione su veri computer quantistici.

2.2 Il codice

Passiamo ora alla descrizione del codice. Inizialmente, il codice utilizza la libreria Python networkx per generare un grafo. L'analisi effettuata ha preso in considerazione diversi tipi di

grafi, partendo da un grafo non pesato per estendersi successivamente a grafi pesati. La fase finale dell'analisi ha esplorato il comportamento del QAOA su grafi formati secondo il modello Barabasi-Albert. Per ulteriori dettagli sul modello Barabasi-Albert e sui risultati ottenuti, si rimanda al Capitolo 3.

Proseguendo, ho calcolato in maniera classica i valori di cut per tutte le possibili combinazioni all'interno del set $\{0,1\}^n$, dove n rappresenta il numero di nodi presenti nel grafo, corrispondente anche al numero di qubit utilizzati nell'analisi. Per ogni possibile bitstring, ho determinato il cut sommando i pesi di tutti gli archi che lo separano. Ho successivamente creato un istogramma per visualizzare i valori di cut per ogni possibile bitstring, identificando così il cut di valore più alto il quale ci servirà per validare i risultati ottenuti dal QAOA.

A questo punto, ho convertito il problema MaxCut in un problema QUBO mediante la libreria Qiskit Optimization. Questa conversione è fondamentale per l'applicazione delle tecniche di ottimizzazione quantistica, poiché le piattaforme quantistiche come Qiskit utilizzano il formato QUBO per esprimere problemi di ottimizzazione. Dopo aver ottenuto il problema nel formato QUBO, ho calcolato l'Hamiltoniana di Ising, o Hamiltoniana di costo H_C. Una volta ottenuta H_C, il QAOA può essere utilizzato per trovare il suo ground state, che rappresenta la soluzione del problema. Ho condotto vari test variando il numero di layer attraverso il parametro p, utilizzando l'algoritmo QAOA presente nella libreria "qiskit.algorithms" e la libreria Qiskit Aer per simulare l'esecuzione su un computer quantistico, ho utilizzato la libreria Qiskit Runtime quando ho effettuato simulazioni su computer reali. Come ottimizzatore classico ho utilizzato il COBYLA. L'algoritmo COBYLA (Constrained Optimization BY Linear Approximations), implementato in Qiskit, è un metodo di ottimizzazione senza derivata utilizzato per risolvere problemi di ottimizzazione vincolati non lineari. Inizia con un punto iniziale e un modello lineare approssimato, poi esplora l'area di ricerca muovendosi lungo le direzioni scelte per minimizzare il modello corrente e soddisfare i vincoli. Dopo ogni passo, il modello viene aggiornato con nuovi dati. Ho settato il parametro della tolleranza a 10^{-4}, ciò vuol dire che l'ottimizzazione viene bloccata quando la differenza tra il valore della funzione obiettivo calcolato nelle iterazioni successive è inferiore a 10^{-4}. In altre parole, l'algoritmo continuerà ad iterare finché il miglioramento ottenuto nella funzione obiettivo rispetto all'iterazione precedente è inferiore a 10^{-4}. Sebbene COBYLA non possa garantire di trovare il minimo globale, è spesso efficace per una vasta gamma di problemi di ottimizzazione.

L'Hamiltoniana di mix è stata mantenuta al valore predefinito, ovvero quello presente nell'Equazione (1.30). Dopo aver ottenuto i risultati del QAOA, ho generato un istogramma simile a quello precedentemente descritto, ordinando però le bitstring dalla più alla meno probabile. Il colore di ogni barra corrisponde al valore del cut della relativa bitstring, questo valore è stato calcolato in modo classico in modo da poter capire meglio la distribuzione di probabi-

lità. Ho poi calcolato il valore del cut della bitstring più probabile e l'ho confrontato con il miglior cut ottenuto classicamente. Questo confronto ha permesso di calcolare il *rapporto di approssimazione*, definito come il rapporto tra la somma dei pesi degli archi che attraversano il cut ottenuto dall'algoritmo QAOA e la somma massima possibile dei pesi degli archi che potrebbero attraversare un cut:

$$\alpha = \frac{C_{\text{QAOA}}}{C_{\text{max}}} \tag{2.1}$$

Dove C_{QAOA} rappresenta la somma dei pesi degli archi nel cut ottenuto dal QAOA e C_{max} rappresenta la somma massima possibile dei pesi degli archi in un cut. Questo rapporto, α, ci fornisce un indicatore dell'efficacia del QAOA nel risolvere il problema considerato.

Un'altra grandezza utilizzata successivamente nelle nostre analisi è la fidelity, la quale descrive la similitudine tra il risultato ottenuto e il risultato ideale. Questa è stata calcolata creando un vettore soluzione formato dalla somma degli stati corrispondenti alle bitstring con il cut massimo (individuate in modo classico) normalizzato:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{s} |s\rangle \tag{2.2}$$

e facendo il prodotto scalare con lo stato output del QAOA. Questo secondo stato può essere calcolato sommando tutti gli stati possibili moltiplicati per la radice quadrata della probabilità ottenuta dal QAOA:

$$|\phi\rangle = \sum_{i} \sqrt{p(i)}|i\rangle \tag{2.3}$$

La fidelity sarà quindi definita dalla formula:

$$\text{fidelity} = |\langle\phi|\psi\rangle| \tag{2.4}$$

2.2.1 Parameter fixing

Mentre il QAOA offre alcune garanzie di performance già a $p = 1$ [20] per poter raggiungere un vantaggio sugli algoritmi classici è necessario esplorare profondità del circuito maggiori [35]. Tuttavia, l'aumento di p porta a un landscape di ottimizzazione progressivamente più complesso, caratterizzato da un gran numero di minimi locali sub-ottimali. Si è dimostrato [36] che la convergenza degli algoritmi di ottimizzazione classici in tali soluzioni sub-ottimali potrebbe essere un potenziale collo di bottiglia delle prestazioni di QAOA, poiché trovare un minimo quasi ottimale di solito richiede un numero di inizializzazioni dell'algoritmo di ottimizzazione classico esponenziale in p. Da notare che il problema dei minimi locali sub-ottimali è diverso da quello dei barren plateaus [37, 38], i quali sono regioni dello spazio di ricerca in cui il gradiente della funzione obiettivo da ottimizzare diventa molto piccolo, quasi piatto, rendendo difficile per gli algoritmi di ottimizzazione trovare una direzione per migliorare la soluzione corrente, problema che viene amplificato all'aumentare di p.

Per risolvere questi problemi, Lee, X. *et al.* [23] hanno presentato la strategia del *parameter fixing*. In sintesi, questa strategia consiste nei seguenti passaggi:

- Si inizia con una profondità di circuito $p = 1$. A questo livello, si risolve il problema con 20 set di parametri casuali, affidandosi all'ottimizzatore per trovare la soluzione desiderata.
- Si selezionano i parametri con il rapporto di approssimazione più alto e si considerano questi come i parametri ottimali per $p = 1$.
- Questo set di parametri viene fissato, e si inserisce un altro paio di parametri casuali per la successiva profondità di circuito, $p = 2$. Si passa poi questo nuovo set di parametri iniziali nel risolvere per $p = 2$.
- Per assicurarsi di trovare il massimo globale, si utilizzano 20 coppie di parametri casuali e si seleziona quella che produce il rapporto di approssimazione più alto. In questo modo, si ottengono i parametri ottimali per $p = 2$.
- Questa procedura viene ripetuta fino a raggiungere il p desiderato.

Nonostante il termine "parameter fixing" possa suggerire che i parametri vengono bloccati in un certo stato, la simulazione effettiva consente ulteriori ottimizzazioni. In pratica, quando l'algoritmo passa a un livello di profondità superiore, i parametri ottimizzati precedentemente (i migliori selezionati per il livello precedente) sono utilizzati come punto di partenza. Tuttavia, non restano statici. Anche questi, insieme ai nuovi parametri introdotti, sono soggetti a un'ulteriore fase di ottimizzazione. Pertanto, sebbene questi parametri siano "fissati" come base di partenza per ogni livello di profondità successivo, possono ancora essere raffinati per

adattarsi meglio al nuovo livello di profondità.

Questa strategia ha dimostrato di essere efficace nell'ottenere buoni risultati nel QAOA. Inoltre, è stato riscontrato che per i grafici 3-regolari con il parameter fixing, la deviazione standard del rapporto di approssimazione diminuisce con l'aumento di p, indicando che l'importanza dei nuovi parametri casuali diventa minore al crescere di p. Nel Capitolo 3 vedremo come questa tecnica può migliorare i risultati in diversi tipi di grafi.

Un'altra idea che ha dimostrato ottimi miglioramenti nelle performance a p alti è quella proposta da Sack, S. H. and Serbyn, M. [39] di utilizzare la Trotterized Quantum Annealing (TQA) per poter inizializzare gli angoli in un intorno del minimo reale. Anche la tecnica proposta da Egger, D. J. *et al.* [40] che consiste nel far iniziare la computazione da un risultato vicino a quello ottimale, trovato con metodi classici come l'algoritmo di Goemans-Williamson modificando lo stato di partenza e l'Hamiltoniana di mixing. Nonostante queste tecniche promettono risultati migliori del parameter fixing non saranno affrontate in questa tesi.

2.3 Implementazione su computer reali

La maggior parte dei test sono stati effettuati su simulatori di computer quantistici come quelli forniti dalla libreria Qiskit Aer, citata in precedenza, oppure sul "qasm_simulator" della libreria Qiskit Runtime che simula a tutti gli effetti un computer quantistico ideale. Le sfide per poter eseguire l'algoritmo QAOA su computer quantistici reali, come quelli forniti da IBM, si articolano su vari livelli. A causa della decoerenza quantistica, i qubit superconduttivi, pur avendo vantaggi significativi in termini di scalabilità e coerenza, tendono a perdere le loro proprietà quantistiche con il passare del tempo e con le interazioni ambientali, limitando quindi il tempo utile per l'esecuzione delle operazioni. Un ulteriore problema riguarda l'implementazione delle porte quantistiche. Sebbene la tecnologia IBM permetta l'implementazione di una vasta gamma di porte, la realizzazione di circuiti complessi può essere ostacolata da errori di gate e limitazioni di connettività tra i qubit. Nel contesto del calcolo quantistico si pone una sfida significativa nel mappare efficientemente i problemi di ottimizzazione su hardware quantistico. Questa sfida è particolarmente difficile quando si tratta di problemi densi, in cui molte variabili (o qubit) interagiscono tra loro. Una strategia per affrontare questa sfida è l'uso di operazioni di swap. Queste operazioni scambiano lo stato di due qubit, permettendo di eseguire operazioni tra qubit che altrimenti non sarebbero direttamente connessi nel layout

fisico del computer quantistico (come un gate $CNOT$). Nell'articolo proposto da Weidenfeller, J. *et al.* [41] vengono discussi vari aspetti della scalabilità dell'algoritmo QAOA su qubit superconduttivi. L'articolo esplora diverse strategie di swap per mappare problemi densi in mappe di accoppiamento lineari, griglia e heavy-hex. Queste strategie determinano come vengono effettuati gli swap per ottimizzare l'esecuzione di un algoritmo su una specifica mappa di accoppiamento:

- Mappe di accoppiamento lineari, in cui ogni qubit è connesso ai suoi due vicini immediati in una configurazione lineare o anulare.
- Mappe di accoppiamento a griglia, in cui i qubit sono disposti in una griglia bidimensionale e ogni qubit è connesso ai suoi quattro vicini.
- Mappe di accoppiamento heavy-hex, una configurazione in cui i qubit sono organizzati in un reticolo esagonale con connessioni extra tra certi qubit, formando così esagoni "pesanti".

Attraverso l'uso di queste strategie di swap, gli autori cercano di ottimizzare l'esecuzione del QAOA su hardware quantistico rumoroso.

In questo articolo vengono anche descritte le limitazioni del QAOA dovute agli errori nei gate (la cosiddetta *gate fidelity*). La performance del QAOA è fortemente influenzata dalla presenza di errori nei gate dei circuiti quantistici. Le disuguaglianze entropiche [42, 43] ci permettono di stabilire un limite alla profondità massima di un circuito QAOA in presenza di rumore depolarizzante. Il rumore depolarizzante, dovuto a una varietà di fattori come interferenze esterne, imperfezioni nel sistema quantistico e l'interazione del sistema quantistico con l'ambiente, è un tipo comune di errore nei computer quantistici che può causare la perdita di informazioni da un qubit, portandolo a uno stato di massima incertezza. La profondità di un circuito si riferisce al numero di livelli di gate quantistici che possono essere applicati. Nello specifico, un limite superiore per la profondità di un circuito QAOA che presenta una frazione f_1 di livelli di gate a singolo qubit e una frazione f_2 di livelli di gate a due qubit, con rispettive probabilità di rumore depolarizzante p_1 e p_2, può essere espressa come:

$$L_{\text{max}} \approx \frac{\ln(\epsilon^{-1})}{2(f_1 p_1 + f_2 p_2)} \tag{2.5}$$

In questa formula, ϵ rappresenta la precisione con cui si approssima l'energia. Questo valore dovrebbe variare tra 10^{-1} e 10^{-2}, poiché la maggior parte degli algoritmi di ottimizzazione richiede un numero di campionamenti (shots) che scala in modo inversamente proporzionale

a ϵ. Considerando che la porta $CNOT$ è la principale fonte di errore nei circuiti QAOA, in particolare per problemi di alta densità che sono dominati da livelli di gate a due qubit, si può semplificare ulteriormente la formula in:

$$L_{\max} \approx \frac{\ln(\epsilon^{-1})}{2p_2} \tag{2.6}$$

In questo caso, la probabilità p_2 è considerata come la probabilità di un canale depolarizzante.

Un'altra sfida deriva dalla correzione degli errori quantistici. A differenza dei computer classici, nei quali gli errori possono essere facilmente rilevati e corretti grazie alla ridondanza delle informazioni e ai codici di correzione degli errori, nei computer quantistici la situazione è molto più complessa. Questo è dovuto a due principali caratteristiche dei sistemi quantistici quali la possibilità dei qubit di trovarsi in uno stato di sovrapposizione e a fenomeni quantistici, come la decoerenza e l'interferenza quantistica, che possono introdurre errori nei calcoli. La decoerenza [44], in particolare, è un processo per cui l'informazione quantistica viene persa a causa dell'interazione con l'ambiente circostante. Questo può avvenire molto rapidamente e può causare errori significativi nei calcoli. Inoltre, a causa delle restrizioni fisiche e tecnologiche, attualmente abbiamo a disposizione un numero limitato di qubit. Questo limite impone restrizioni significative alla capacità di implementare i codici di correzione degli errori quantistici, che richiedono spesso un grande numero di qubit per codificare in modo sicuro un singolo qubit logico.

Un ulteriore problema deriva dagli errori di misurazione. Gli errori di misurazione nei calcoli quantistici si verificano quando il risultato di una misurazione di un qubit non corrisponde al vero stato del qubit. Questi errori possono derivare da vari fattori come il rumore termico, interferenze elettromagnetiche e imperfezioni nel processo di misurazione. Questi errori sono particolarmente problematici perché una volta che un qubit è misurato, il suo stato collassa in uno stato definitivo, e qualsiasi errore durante la misurazione non può essere corretto successivamente. Questo diventa particolarmente problematico in algoritmi come QAOA, che si basano sulla misurazione ripetuta dei qubit. Per affrontare questo problema, i ricercatori stanno esplorando varie strategie, tra cui il miglioramento del segnale di misurazione, l'uso di tecniche di elaborazione del segnale per ridurre l'effetto del rumore [45, 46], e l'implementazione di protocolli di correzione degli errori di misurazione [47, 48].

Infine, l'ottimizzazione delle istanze del problema è un'altra questione complessa. Mentre in un computer classico possiamo risolvere molte istanze di un problema contemporaneamente per trovare la soluzione ottimale, l'esecuzione di QAOA su un computer quantistico reale può

richiedere più tempo e risorse, data la necessità di eseguire le operazioni sequenzialmente a causa delle limitazioni fisiche e tecniche. Tutto ciò rende l'esecuzione di QAOA su un vero computer quantistico IBM un compito impegnativo, ma nonostante queste sfide, i progressi nella tecnologia quantistica continuano ad accelerare, promettendo miglioramenti futuri.

Capitolo 3

Analisi Sperimentale del QAOA su Grafi Generici e Barabási–Albert

In questo capitolo ci immergeremo nel modello Barabási–Albert, un concetto fondamentale per comprendere la struttura delle reti complesse (che può essere approfondito nel libro "Networks an introduction" [49]). Successivamente, esporremo e discuteremo i risultati sperimentali ottenuti sia su grafi non pesati che pesati. Infine, porteremo il modello Barabási–Albert in un contesto pratico, testando il Quantum Approximate Optimization Algorithm (QAOA) su grafi generati con questo modello. Questo ci darà una comprensione più profonda di come la teoria e la pratica possono incontrarsi nella scienza delle reti e nella computazione quantistica.

3.1 Modello Barabási-Albert

Il modello Barabási–Albert (BA) [50] è un algoritmo per generare reti scale-free (la cui distribuzione dei gradi segue una legge di potenza, almeno asintoticamente) utilizzando un meccanismo di crescita preferenziale. Il processo inizia con una rete di m_0 nodi isolati. I nuovi nodi vengono aggiunti alla rete uno alla volta, ognuno dei quali si collega a $m \leq m_0$ nodi esistenti con una probabilità proporzionale al numero di link che i nodi esistenti hanno già. Formalmente, la probabilità p_i che il nuovo nodo si colleghi al nodo i è:

$$p_i = \frac{k_i}{\sum_j k_j} \tag{3.1}$$

dove k_i è il grado del nodo i e la somma è fatta su tutti i nodi preesistenti j. I nodi fortemente collegati ("hubs") tendono ad accumulare rapidamente ulteriori link, mentre i nodi con pochi link difficilmente vengono scelti come destinazione per un nuovo link. I nuovi nodi hanno una "preferenza" per collegarsi ai nodi già fortemente collegati. La distribuzione dei gradi

risultante dal modello BA è senza scala, in particolare, è una legge di potenza della forma

$$P(k) \sim k^{-3} \tag{3.2}$$

Per sua natura, il modello Barabási-Albert descrive un processo dinamico che si sviluppa nel tempo. Oltre alla sua nota proprietà di rete scale-free, è possibile esaminare anche le sue proprietà di dynamic scaling [51]. Questo concetto si riferisce a come la struttura della rete cambia nel tempo, poiché nuovi nodi vengono continuamente aggiunti. Nella rete del modello BA, ogni nodo ha un 'tempo di nascita', che rappresenta il momento in cui è stato inserito nella rete. Questo tempo di nascita è importante perché i nodi inseriti prima avranno più tempo per acquisire link tramite l'attaccamento preferenziale. Per tener conto di questo, i nodi possono essere caratterizzati non solo dal loro grado k (il numero di link che hanno), ma da un *grado generalizzato* q. Il grado generalizzato è definito come il prodotto del grado k del nodo e la radice quadrata del suo tempo di nascita, ovvero

$$q = k\sqrt{t} \tag{3.3}$$

Questo permette di considerare sia il grado del nodo sia il momento in cui è stato aggiunto alla rete. La proprietà di dynamic scaling si riferisce alla maniera in cui la distribuzione del grado generalizzato q cambia nel tempo. In particolare, si osserva che la distribuzione di q segue una legge di potenza, con un esponente che varia nel tempo. Questo è un tratto distintivo delle reti che evolvono nel tempo, come quella descritta dal modello Barabási-Albert.

Esistono anche varianti del modello BA, come il Modello A che mantiene la crescita ma non include l'attaccamento preferenziale, e il Modello B che mantiene l'attaccamento preferenziale ma elimina la crescita. Nessuno di questi modelli produce una struttura scale-free, indicando che sia la crescita che l'attaccamento preferenziale sono necessari simultaneamente per riprodurre la distribuzione stazionaria di legge di potenza osservata nelle reti reali.

Il modello BA può essere pensato come un caso specifico del modello di attaccamento preferenziale non lineare più generale (NLPA) [52]. L'algoritmo NLPA è identico al modello BA con la probabilità di attaccamento sostituita dalla forma più generale

$$p_i = \frac{k_i^\alpha}{\sum_j k_j^\alpha} \tag{3.4}$$

dove α è un esponente positivo costante. Se $\alpha = 1$, NLPA si riduce al modello BA e viene definito "lineare". Se $0 < \alpha < 1$, NLPA è definito "sub-lineare" e la distribuzione di grado della rete tende a una distribuzione esponenziale estesa, cioè una famiglia di distribuzioni di probabilità che generalizzano la distribuzione esponenziale tradizionale. Mentre una distribuzione esponenziale ha la forma $p(x) \propto e^{-x}$, una distribuzione esponenziale estesa ha la forma $p(x) \propto e^{-x^{\beta}}$, dove β è un parametro che può variare da 0 a 1. Se $\alpha > 1$, NLPA è definito "super-lineare" e un piccolo numero di nodi si collega a quasi tutti gli altri nodi nella rete. Per entrambi i casi $\alpha < 1$ e $\alpha > 1$, la proprietà scale-free della rete è interrotta nel limite di dimensione infinita del sistema.

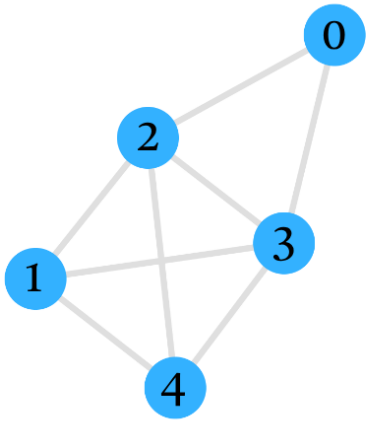

Figura 9: Grafo con 5 nodi non pesato.

3.2 Un esempio preliminare

Per valutare le prestazioni del QAOA, sono state condotte diverse simulazioni.Le simulazioni sono state eseguite utilizzando il ”qasm_simulator” offerto dalla libreria Qiskit Aer. Per ogni run del QAOA sono state effettuate 1024 iterazioni per poter costruire le distribuzioni di probabilità che vedremo. Inizialmente, l'analisi si è concentrata sulla capacità del QAOA di risolvere il problema MaxCut su un grafo non pesato di 5 nodi. Questo grafo è stato successivamente modificato per includere pesi sugli archi, aumentando così la complessità del problema. Una rappresentazione del grafo selezionato per questo primo test può essere visualizzata in Figura 9. Prima di procedere con l'implementazione del QAOA, abbiamo determinato il valore associato ad ogni possibile cut. Questo è stato ottenuto sommando i pesi degli archi che separano i due insiemi risultanti dalla suddivisione data da ogni possibile bitstring. Importante sottolineare che il valore del cut corrisponde all'opposto dell'energia calcolata su ogni bitstring. Questa fase preliminare ha fornito un punto di riferimento per valutare l'efficacia del QAOA nel risolvere il problema MaxCut. Nella Figura 10 è presentato un istogramma che mostra i cut associati a ciascuna bitstring. Le bitstring sono ordinate in base al valore del loro cut, dal più basso al più alto. Nella Figura 11 è illustrata la soluzione ottimale del problema MaxCut sul grafo non pesato a 5 nodi. I due sottoinsiemi del grafo, distinti dai colori blu e azzurro, sono separati in modo da massimizzare il numero di archi tra di loro. Questa separazione specifica corrisponde alla bitstring che ha prodotto il valore massimo del cut nell'istogramma in Figura 10. É importante notare che le due bitstring presentate come soluzione in questa figura sono interscambiabili e, pertanto, definiscono la stessa suddivisione del grafo.

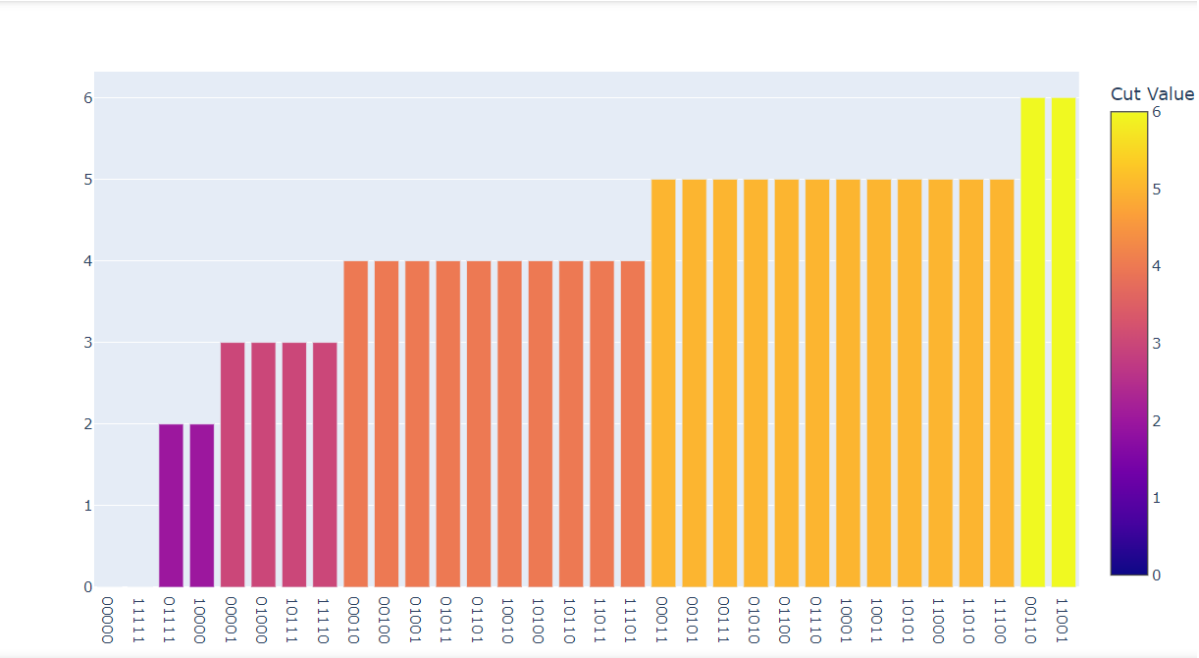

Figura 10: Istogramma contenente tutti i cut del grafo di 5 nodi non pesato, cioè dove tutti gli archi hanno peso 1.

Passiamo adesso all'utilizzare il QAOA per risolvere lo stesso problema, nella Figura 12 è possibile vedere l'esatta composizione del circuito per $p = 1$ (profondità del circuito). L'Hamiltoniana di costo H_C che descrive il problema è della forma:

$$\begin{aligned} H_C = \quad & 0.5 \cdot IIZIZ \\ & + 0.5 \cdot IIZZI \\ & + 0.5 \cdot IZIIZ \\ & + 0.5 \cdot IZIZI \\ & + 0.5 \cdot IZZII \\ & + 0.5 \cdot ZIIZI \\ & + 0.5 \cdot ZIZII \\ & + 0.5 \cdot ZZIII \\ & - 4 \end{aligned}$$

Essa rappresenta l'interazione tra cinque qubit, ordinati secondo la numerazione dei vertici

del grafo scelta in Figura 9. Ciascun termine dell'Hamiltoniana è costituito da un prodotto tensoriale di operatori di Pauli 'I' (identità) e 'Z' (operatore di Pauli Z). L'operatore 'I' agisce come l'identità su qualsiasi stato del qubit, lasciandolo inalterato. Diversamente, l'operatore 'Z' cambia il segno del qubit se il suo stato è $|1\rangle$ e lo lascia inalterato se lo stato è $|0\rangle$. In termini matematici, questo si traduce nelle seguenti trasformazioni: $Z|0\rangle = |0\rangle$ e $Z|1\rangle = -|1\rangle$. Il coefficiente 0.5 in ogni termine indica che tutte le interazioni tra i qubit contribuiscono con lo stesso peso al valore di aspettazione dell'Hamiltoniana totale. L'offset di -4.0 è una costante sommata all'energia totale calcolata dall'Hamiltoniana e serve a spostare l'intero spettro energetico di una quantità fissa, ma non influisce sull'evoluzione temporale dello stato quantistico. Il nostro obiettivo nel contesto del QAOA è trovare lo stato quantistico che minimizza il valore di aspettazione dell'Hamiltoniana totale.

La Figura 13 illustra le distribuzioni di probabilità in funzione del parametro p, con le bitstring disposte in ordine crescente di probabilità. Il colore associato a ciascuna bitstring riflette il valore del cut che essa definisce, come indicato dalla scala cromatica intitolata 'Function Value' posta a destra. In questa rappresentazione, le bitstring che definiscono la soluzione ottimale sono facilmente riconoscibili grazie al loro colore giallo chiaro. Notiamo che per $p = 3$ l'algoritmo non trova la soluzione corretta, questo è dovuto al fatto che, all'aumentare dei parametri, diventa sempre più difficile per l'ottimizzatore classico trovare il minimo globale dell'Hamiltoniana di costo. Per ovviare a questo problema è stata utilizzata la tecnica del "parameter fixing" descritta in precedenza. In Figura 14 è possibile vedere i risultati di una simulazione fatta con lo stesso grafo ma con la tecnica del parameter fixing. Notiamo che in questo caso la soluzione più probabile è sempre quella corretta ed, inoltre, a $p = 4$ il minimo è stato trovato con particolare precisione passando da una fidelity del 49% a $p = 4$ senza parameter fixing ad una di 75% allo stesso p.

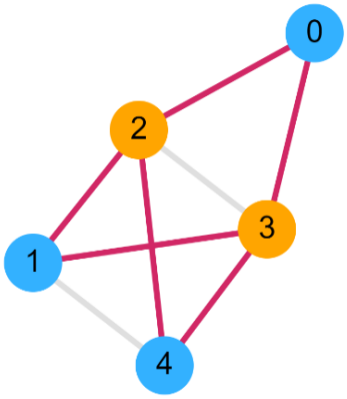

Figura 11: Soluzione del MaxCut sul grafo non pesato a 5 nodi.

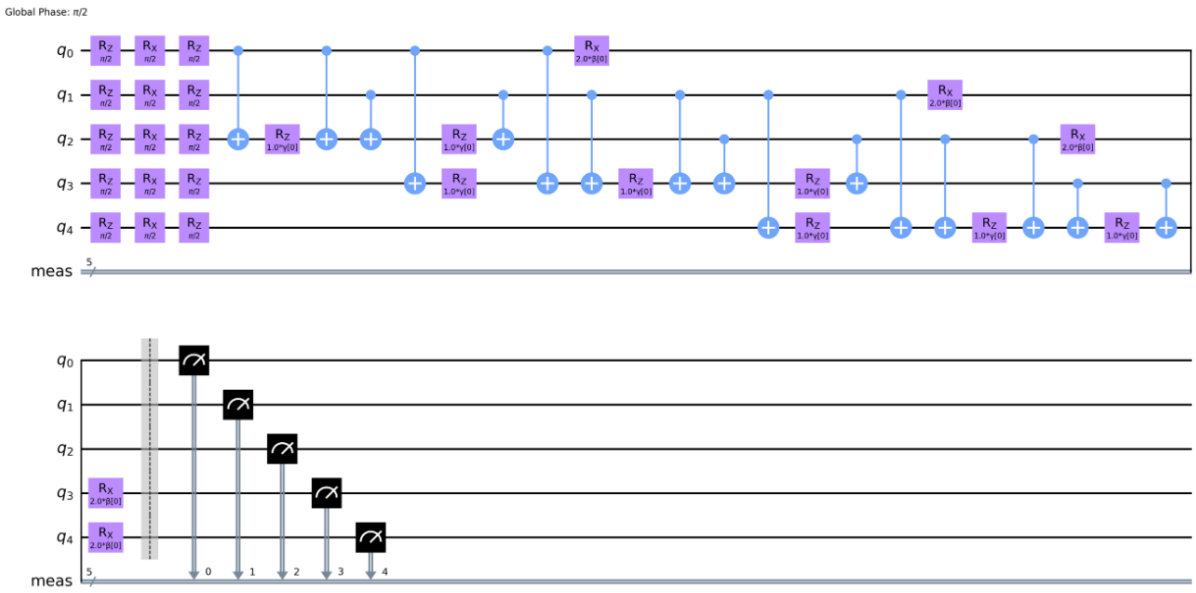

Figura 12: Circuito QAOA a $p = 1$.

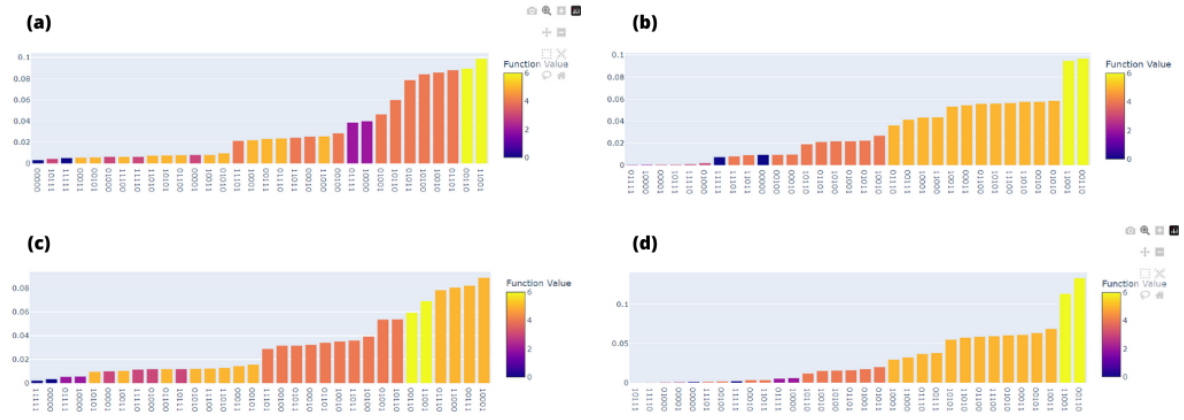

Figura 13: Istogrammi con distribuzione di probabilità ottenuta dal QAOA al variare di p per il grafo non pesato da 5 nodi. In Figura (a) risultati per $p = 1$, in Figura (b) risultati per $p = 2$, in Figura (c) risultati per $p = 3$, in Figura (d) risultati per $p = 4$.

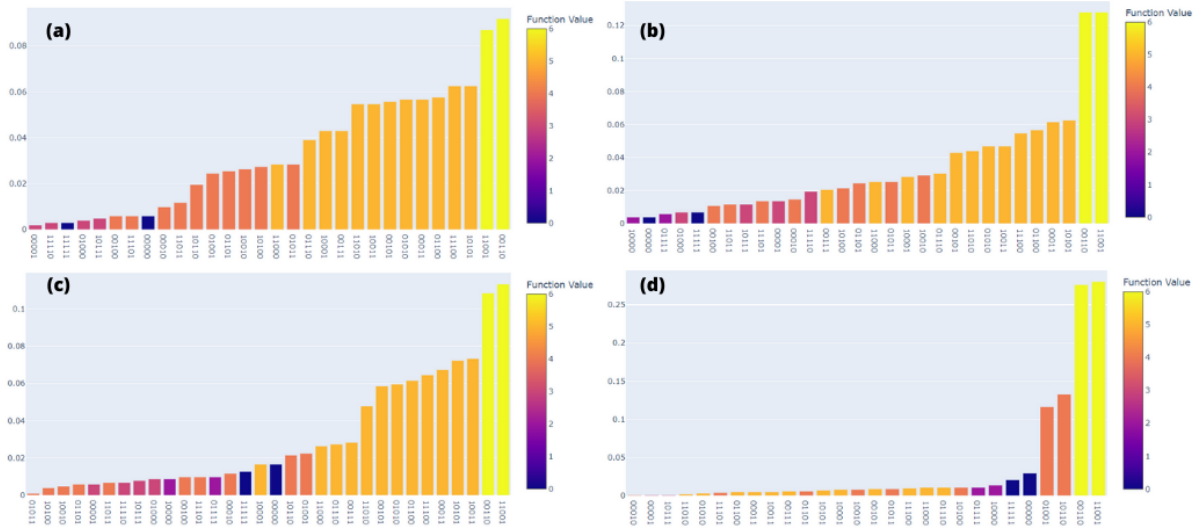

Figura 14: Istogrammi con distribuzione di probabilità ottenuta dal QAOA con aggiunta della tecnica del parameter fixing al variare di p per il grafo non pesato da 5 nodi. In Figura (a) risultati per $p = 1$, in Figura (b) risultati per $p = 2$, in Figura (c) risultati per $p = 3$, in Figura (d) risultati per $p = 4$.

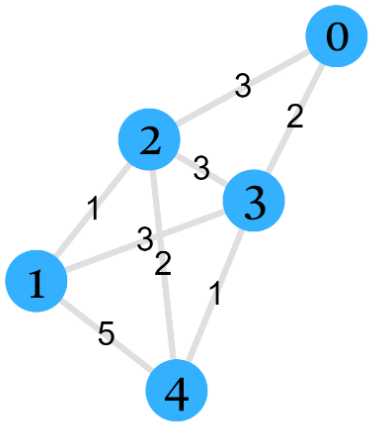

Figura 15: Grafo con 5 nodi pesato.

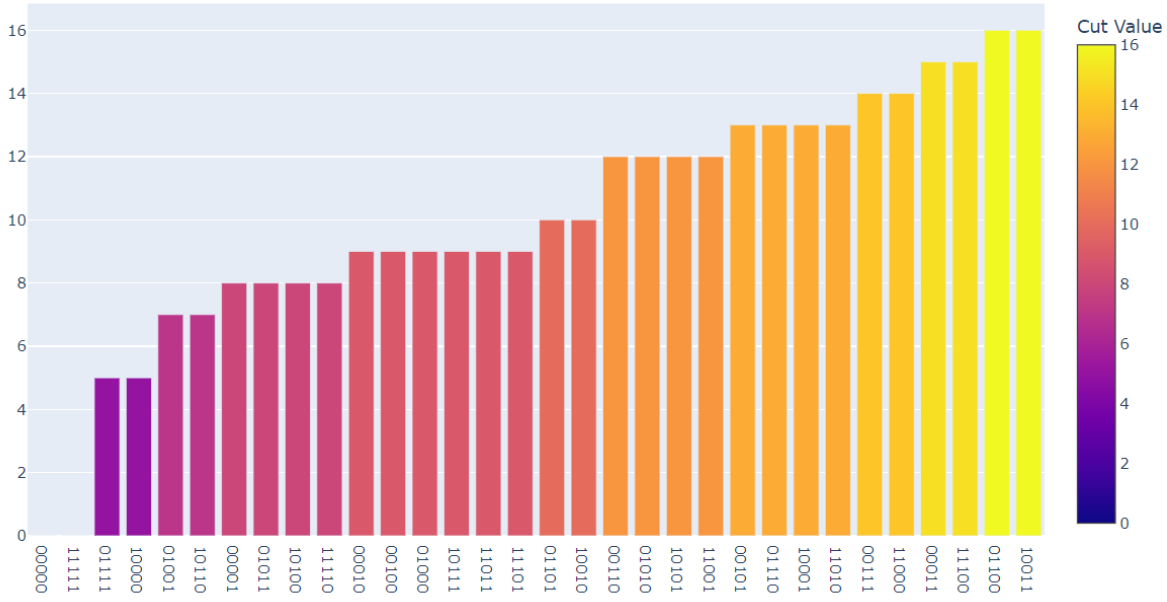

Figura 16: Istogramma contenente tutti i cut del grafo di 5 nodi pesato.

Passiamo adesso ad analizzare la versione pesata del grafo disegnata in Figura 15, i cui cut sono rappresentati nell'istogramma in Figura 16. L'Hamiltoniana di costo H_C diventa:

$$\begin{aligned} H_C = \quad & 1.5 \cdot IIZIZ \\ & + 1.0 \cdot IZIIZ \\ & + 0.5 \cdot IIZZI \\ & + 1.5 \cdot IZIZI \\ & + 2.5 \cdot ZIIZI \\ & + 1.5 \cdot IZZII \\ & + 1.0 \cdot ZIZII \\ & + 0.5 \cdot ZZIII \\ & - 10 \end{aligned}$$

I risultati del QAOA senza parameter fixing sono presentati in Figura 17 mentre quelli con parameter fixing sono in Figura 18. In questa circostanza è possibile notare una differenza ancora più grande tra le soluzioni trovate. É evidente che la strategia del parameter fixing permette di trovare risultati migliori, specialmente se il problema da risolvere è complesso. Per questo motivo da questo momento in avanti tutti i test effettuati utilizzeranno la tecnica del parameter fixing ad esclusione di quelli fatti su computer quantistici reali.

É possibile adesso simulare gli stessi problemi su un computer reale. Per questo esperimento è stato utilizzato il computer "ibmq_lima", un computer quantistico da 5 qubit, 8 QV e 2.7k CLOPS. Quantum Volume (QV) e Circuit Layer Operations Per Second (CLOPS) sono metriche per valutare le prestazioni dei computer quantistici. Il QV, introdotto da IBM, tiene conto di vari fattori come il numero di qubits, errori di gate e di misura, cross talk del dispositivo e connettività. Le CLOPS, invece, misurano la velocità con cui un computer quantistico può eseguire le sue operazioni. Queste metriche aiutano a confrontare le capacità dei diversi computer quantistici.

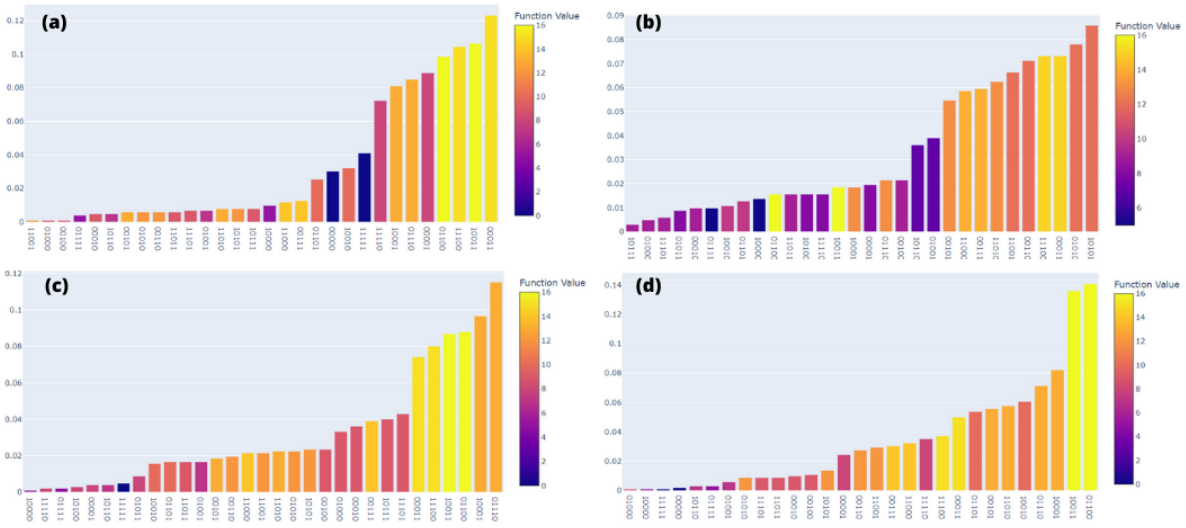

Figura 17: Istogrammi con distribuzione di probabilità ottenuta dal QAOA al variare di p per il grafo pesato da 5 nodi. In Figura (a) risultati per $p = 1$, in Figura (b) risultati per $p = 2$, in Figura (c) risultati per $p = 3$, in Figura (d) risultati per $p = 4$.

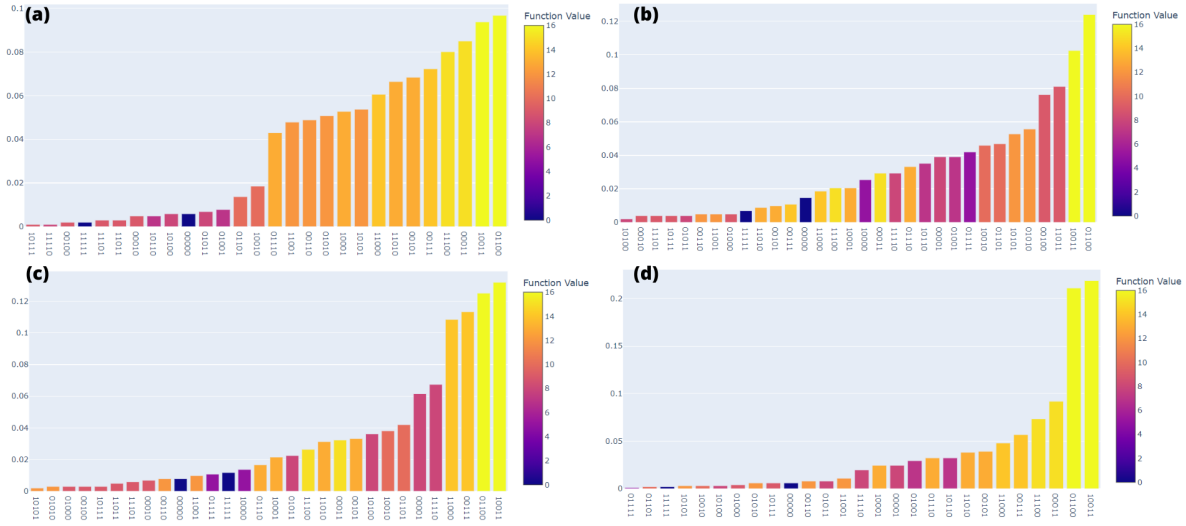

Figura 18: Istogrammi con distribuzione di probabilità ottenuta dal QAOA con aggiunta della tecnica del parameter fixing al variare di p per il grafo pesato da 5 nodi. In Figura (a) risultati per $p = 1$, in Figura (b) risultati per $p = 2$, in Figura (c) risultati per $p = 3$, in Figura (d) risultati per $p = 4$.

Figura 19: Istogrammi con distribuzione di probabilità ottenuta dal QAOA a $p = 2$ su un computer reale. A sinistra il grafo non pesato e a destra quello pesato.

Nella Figura 19 sono mostrate le distribuzioni di probabilità generate dal QAOA sul computer quantistico reale con un parametro $p = 2$ sia per il grafo non pesato che pesato. Come si può notare, gli errori, precedentemente discussi, hanno un impatto significativo sui risultati, rendendo l'ottimizzazione un compito notevolmente difficile. Quando la complessità del problema aumenta, la difficoltà nell'individuare il risultato corretto cresce di conseguenza. I pesi associati ai bordi di un grafo influenzano le operazioni sui qubit corrispondenti. Un errore o rumore in un'operazione su un qubit che rappresenta un bordo con un peso maggiore può avere un impatto più significativo sulla soluzione finale rispetto a un errore su un qubit che rappresenta un bordo con un peso minore. Questo è dovuto al fatto che gli errori in operazioni con un grande impatto sul valore di aspettazione dell'Hamiltoniana possono modificare significativamente tale valore, alterando la soluzione ottimale. D'altra parte, gli errori su operazioni che hanno un impatto minore possono avere un effetto meno significativo sulla soluzione finale. Per migliorare i risultati, si potrebbero eseguire simulazioni con valori di p più elevati. Tuttavia, ciò comporterebbe un aumento della lunghezza del circuito, e quindi un incremento degli errori dovuti alla decoerenza e alle fluttuazioni quantistiche. L'implementazione di tecniche di correzione degli errori e di tecniche di swap per adattare il circuito al layout del computer quantistico reale che permetterebbero la riduzione del numero di gate $CNOT$, potrebbe contribuire a migliorare i risultati.

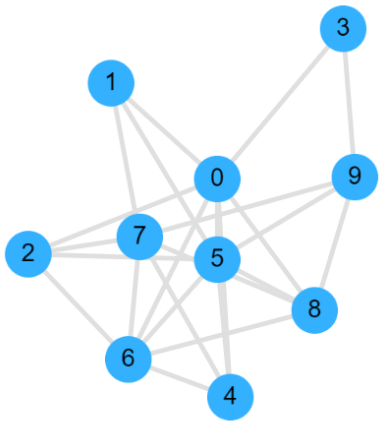

Figura 20: Grafo Barabási–Albert con 10 nodi e connettività $m = 4$ generato utilizzando la libreria networkx di python come descritto nel Capitolo 2

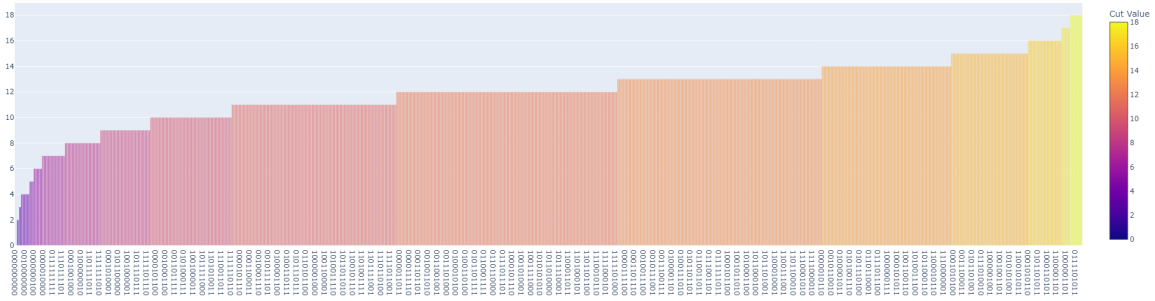

Figura 21: Istogramma contenente tutti i cut del grafo Barabási–Albert con 10 nodi e connettività $m = 4$.

3.3 Risultati su grafi Barabási–Albert

Passiamo adesso all'analisi del comportamento del QAOA con grafi Barabási–Albert. Per prima cosa vediamo, come esempio, come si comporta con il grafo di 10 nodi con connettività $m = 4$ in Figura 20. Iniziamo calcolando tutti i valori dei cut e mettiamoli in un istogramma come presentato in Figura 21, purtroppo i possibili cut per un grafo di 10 nodi iniziano ad essere molti quindi è difficile distinguere dall'istogramma il cut di ogni singola bitstring, è comunque possibile capire la loro distribuzione. Spingiamoci adesso a calcolare la distribuzione di probabilità del QAOA fino a $p = 6$, Figura 22. Possiamo vedere come, all'aumentare di p, il risultato diventa sempre più preciso. A $p = 6$ il l'istogramma mostra come tutte le possibili soluzioni hanno un'alta probabilità di essere estratte e molte bitstring non sono nemmeno incluse nel grafico perché la loro probabilità è praticamente nulla. Ricorrendo all'Equazione (2.4) del Capitolo 2, possiamo calcolare la fidelity, una misura della somiglianza tra il nostro stato ottenuto dal QAOA e lo stato soluzione. Un valore di fidelity più vicino all'unità indica una somiglianza quasi perfetta tra i due stati. Nel nostro caso, con $p = 6$, abbiamo raggiunto una fidelity del 63%. Notiamo come anche se la soluzione sembra essere molto vicina a quella corretta la fidelity è ancora abbastanza lontana dall'unità. Questo è dovuto al fatto che, con grafi così grandi e connessi, è molto difficile raggiungere valori per la fidelity vicini all'unità. Per farlo sarebbe necessario aumentare sempre di più p andando oltre le capacità computazionali che abbiamo a disposizione. É però interessante capire, al variare di m, a quale p si raggiunge un certo valore di fidelity che noi riteniamo accettabile. É importante notare però che essa dipende molto dal tipo di grafo che stiamo analizzando e in particolare dipende dal numero di soluzioni presenti nel grafico. Se sono presenti più bitstring con il cut massimo sarà più facile per l'algoritmo trovare una soluzione con una fidelity molto alta mentre se le bitstring con il cut massimo sono solo 2 potrebbe essere necessario raggiungere p più alti a parità di m per raggiungere la stessa fidelity. Per fare un'analisi affidabile, è necessario mediare sui risultati di diversi grafi, cioè eseguire l'algoritmo su una serie di grafi diversi generati in modo casuale e poi calcolare media e deviazione standard dei risultati della fidelity ottenuti per ogni valore di p. Questo però inizia ad essere particolarmente difficile con $n = 10$ perché i tempi computazionali diventano particolarmente elevati. Possiamo però effettuare quest'analisi su grafi Barabási–Albert con n=6. Nella Figura 23 vengono presentati i risultati della variazione della fidelity al variare di p mediati su 10 grafi generati in maniera casuale. Vediamo che all'aumentare di p la fidelity aumenta di molto inizialmente (fino a $p = 3$) per poi crescere in molto lentamente.

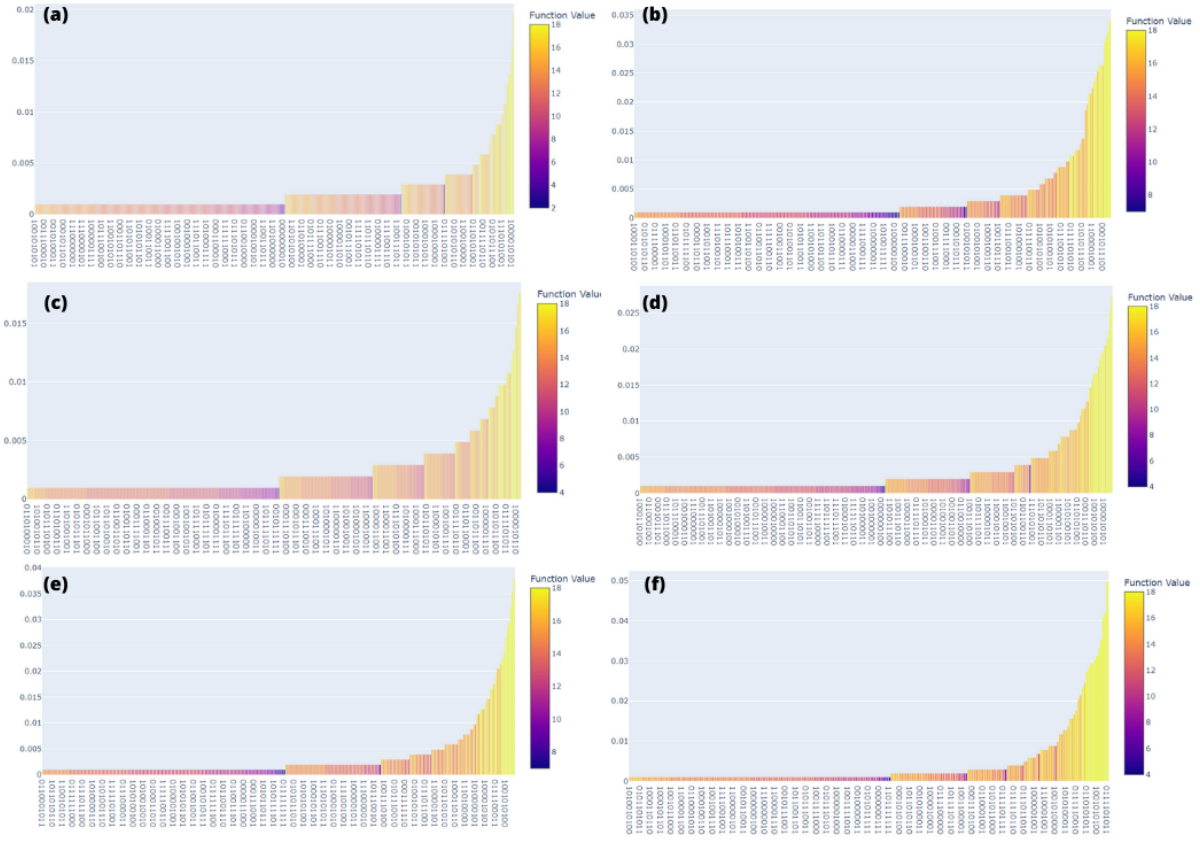

Figura 22: Istogrammi con distribuzione di probabilità ottenuta dal QAOA con aggiunta della tecnica del parameter fixing al variare di p. In Figura (a) risultati per $p = 1$, in Figura (b) risultati per $p = 2$, in Figura (c) risultati per $p = 3$, in Figura (d) risultati per $p = 4$, in Figura (e) risultati per $p = 5$, in Figura (f) risultati per $p = 6$

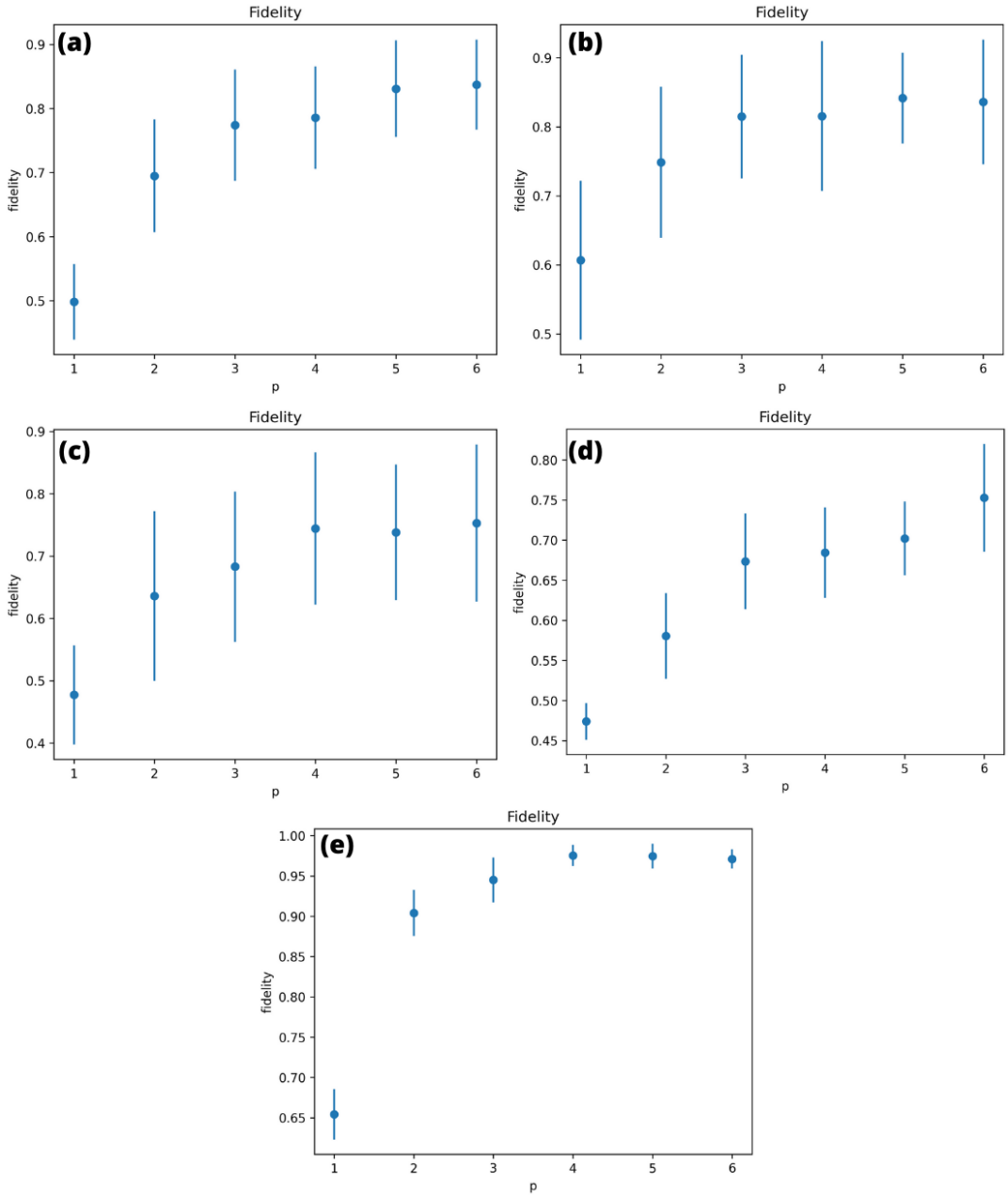

Figura 23: Nella figura è rappresentato l'andamento della fidelity in funzione del parametro p su grafi Barabási–Albert di 6 nodi a m costante. Per ogni valore di m, sono stati generati casualmente 10 grafi, e per ciascuno di questi è stata calcolata la fidelity per vari valori di p. Ogni punto del grafico rappresenta la media di tali valori, con la corrispondente incertezza data dalla deviazione standard. In particolare, la Figura (a) mostra i risultati per $m = 1$, la Figura (b) per $m = 2$, la Figura (c) per $m = 3$, la Figura (d) per $m = 4$, e infine, la Figura (e) presenta i risultati per $m = 5$.

Conclusione

La presente tesi ha affrontato una analisi completa del Quantum Approximate Optimization Algorithm (QAOA) con una particolare attenzione al problema del MaxCut su grafi, tra cui i grafi Barabási-Albert. La nostra indagine ha evidenziato l'importanza del QAOA come uno degli algoritmi quantistici più promettenti per risolvere problemi di ottimizzazione anche di grandi dimensioni facendo leva sulle proprietà dei computer quantistici.

Dai risultati sperimentali emerge con chiarezza che il QAOA è in grado di risolvere efficacemente il problema del MaxCut su grafi di piccole dimensioni. Analizzando l'andamento della fidelity su grafi Barabási-Albert, abbiamo riscontrato un comportamento coerente con le previsioni teoriche: un aumento sostenuto della fidelity nelle fasi iniziali, che poi tende a rallentare con l'incremento del parametro p.

Uno dei principali risultati del nostro studio è stata la conferma che l'efficacia del QAOA può essere fortemente influenzata dal modo in cui i suoi parametri sono inizializzati. In questo contesto, la tecnica del parameter fixing si è rivelata molto utile per mitigare le sfide associate alla scelta dei parametri. Questa osservazione non solo migliora le nostre capacità attuali con l'algoritmo QAOA, ma apre anche nuovi percorsi di ricerca per ulteriormente ottimizzare le prestazioni dell'algoritmo. In particolare, abbiamo individuato due promettenti approcci che non sono stati applicati o esplorati in profondità nel contesto di questa tesi, ma che potrebbero essere sviluppati in ulteriori ricerche. Il primo è la tecnica del warm start [40], che prevede l'inizializzazione dell'algoritmo a partire da una soluzione approssimata, calcolata in maniera classica rilassando il problema. Questa tecnica potrebbe accelerare la convergenza del QAOA a una soluzione ottimale o quasi-ottimale. Il secondo spunto riguarda l'utilizzo del Trotterized Quantum Annealing (TQA) [39] per l'inizializzazione dei parametri del QAOA. L'annealing quantistico, simulato attraverso una sequenza di gate quantistici basati sulle formule di Trotter, potrebbe offrire un metodo più sofisticato per l'inizializzazione dei parametri, che potrebbe portare a soluzioni più precise.

Nel corso del nostro studio, abbiamo applicato il QAOA al problema del MaxCut su vari tipi di grafi, inclusi i grafi Barabási-Albert. Questi ultimi, noti per le loro caratteristiche di formazione di cluster e robustezza ai guasti casuali, hanno offerto un contesto interessante per testare il QAOA. Nonostante le sfide derivanti dalla loro struttura complessa, il QAOA ha mostrato di poter essere utilizzato per affrontare problemi su tali grafi. Questo suggerisce che l'algoritmo

potrebbe essere utilizzato in future ricerche per indagare problemi di ottimizzazione in una serie di contesti reali, da reti sociali a sistemi di telecomunicazione, in cui modelli simili a grafi Barabási-Albert sono spesso riscontrati.

In conclusione, questa tesi ha approfondito l'utilizzo del Quantum Approximate Optimization Algorithm per il problema del MaxCut. Sebbene esistano ancora sfide nell'ottimizzazione dei parametri, le verifiche sperimentali condotte, inclusa l'applicazione a grafi Barabási-Albert, hanno rafforzato l'importanza del QAOA come strumento potenziale per risolvere problemi di ottimizzazione complessi nel campo della computazione quantistica.

Appendice A

L'algoritmo di Goemans-Williamson

L'algoritmo di Goemans-Williamson (GW), presentato nel paper [11], è un noto metodo di approssimazione per il problema del Max-Cut nel campo dell'ottimizzazione combinatoria.

Un problema Max-Cut può essere formalmente definito come segue. Dato un grafo non diretto $G = (V, E)$ con pesi sugli archi $w : E \to \mathbb{R}$, vogliamo trovare una suddivisione dei vertici V in due insiemi disgiunti S e $\bar{S} = V \setminus S$ che massimizza la somma dei pesi degli archi che attraversano i due insiemi.

Questo può essere espresso come il seguente problema di ottimizzazione:

$$\max \quad \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - x_i x_j)$$

Dove $x_i = 1$ se il vertice i è in S e $x_i = -1$ se il vertice i è in $\bar{S}$. La chiave dell'algoritmo di Goemans-Williamson è riformulare il problema Max-Cut come un problema di programmazione semidefinita. Invece di cercare direttamente la soluzione x, cerchiamo una matrice X tale che $X_{ij} = x_i x_j$. Il problema di ottimizzazione diventa quindi:

$$\max_{X\in S_n} \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - X_{ij}) \tag{3.5}$$

$$\text{t.c. } X_{ii} = 1, \forall i \in V \tag{3.6}$$

$$X \succeq 0 \tag{3.7}$$

Dove S_n è l'insieme delle matrici semidefinite $n \times n$, e $X \succeq 0$ indica che X è semidefinita positiva. Questo problema può essere risolto in tempo polinomiale usando algoritmi standard per la programmazione semidefinita. Una volta che abbiamo una soluzione ottimale X^* per il problema SDP, dobbiamo trovare un taglio corrispondente $x \in \{-1, 1\}^n$. Questo è fatto

attraverso un processo di arrotondamento randomizzato. Scegliamo un vettore normale casuale r da una distribuzione gaussiana standard n-dimensionale che descriverà un iperpiano casuale nello spazio vettoriale n-dimensionale. Successivamente, eseguiamo la decomposizione di Cholesky sulla matrice X^*. La decomposizione di Cholesky è un metodo per decomporre una matrice semidefinita positiva in il prodotto di una matrice triangolare inferiore e la sua trasposta. Applicando questo metodo a X^*, otteniamo una matrice L, notiamo che ogni riga di L può essere interpretata come un vettore unitario x_i nello spazio euclideo che rappresenta la posizione del vertice i. Questi vettori x_i hanno la proprietà unica che il loro prodotto scalare corrisponde agli elementi della matrice X^*, ovvero, $x_i \cdot x_j = X^*_{ij}$. Ora, possiamo utilizzare questi vettori per creare un taglio del grafo. Per ogni vertice i, calcoliamo il prodotto scalare tra x_i e r. Se il prodotto scalare è non-negativo, allora il vertice i viene inserito nel set S, altrimenti viene inserito nel set $\bar{S}$. Geometricamente, se il vettore si trova nella parte positiva dell'iperpiano $x_i = 1$ altrimenti $x_i = -1$ In altre parole:

$$x_i = \begin{cases} 1, & \text{se } X_i^* \cdot r \geq 0 \\ -1, & \text{se } X_i^* \cdot r < 0 \end{cases} \tag{3.8}$$

L'algoritmo di Goemans-Williamson garantisce un rapporto di approssimazione di almeno $0.87856 < \alpha < 0.87857$ per il problema Max-Cut, che è il miglior limite noto per un algoritmo di approssimazione polinomiale per Max-Cut. La dimostrazione di questa affermazione è particolarmente complessa, possiamo però fornire l'idea principale che ci sta dietro. Consideriamo un singolo arco alla volta, ad esempio l'arco tra i nodi i e j. Abbiamo due vettori associati a questi nodi, X_i e X_j, che sono stati ottenuti come soluzione dal nostro programma semidefinito (SDP). Questi vettori vengono arrotondati osservando su quale lato di un iperpiano generato casualmente si trovano. La probabilità che l'arrotondamento produca etichette differenti per i due nodi, e quindi che l'arco diventi parte del taglio, è legata all'angolo θ tra X_i e X_j. In particolare, se l'iperpiano casuale si trova all'interno dell'angolo formato da X_i e X_j, i due nodi avranno etichette differenti. In caso contrario, avranno la stessa etichetta. Pertanto, la probabilità che i nodi ricevano etichette diverse è data da θ/π. Calcolando la somma di queste probabilità su tutti gli archi, otteniamo l'aspettativa del valore del taglio prodotto da questo metodo di arrotondamento. Ci chiediamo ora come questa quantità si relaziona alla nostra funzione obiettivo modificata. In particolare, vogliamo capire come si relaziona θ/π a $(1-X_i^T X_j)/2$. Dal momento che X_i e X_j sono vettori unitari, il loro prodotto scalare è uguale al coseno dell'angolo θ. Quindi, ci interessa capire come si relaziona θ/π a $(1-\cos(\theta))/2$. Se tracciamo il rapporto tra queste due quantità in funzione di θ, osserviamo che ha un minimo

in un certo punto che corrisponde a circa 0.87. Questo implica che il taglio ottenuto dopo la procedura di arrotondamento è almeno l'87% del valore della funzione obiettivo del nostro SDP. Ricordiamo che la funzione obiettivo dell'SDP era una versione rilassata del problema Max-Cut originale.

Rimane un importante argomento di ricerca capire se e quando le tecniche di ottimizzazione quantistica, come il QAOA, possono superare questo limite di approssimazione.

Bibliografia

[1] Sfera di bloch (2023). URL https://it.wikipedia.org/wiki/Sfera_di_Bloch.

[2] Qiskit textbook. https://learn.qiskit.org/summer-school/2021/lab2-variational-algorithms.

[3] Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, USA, 2011), 10th edn.

[4] Rasiowa, H. *Stephen Cole Kleene. Introduction to metamathematics. North-Holland Publishing Co., Amsterdam, and P. Noordhoff, Groningen, 1952; D. van Nostrand Company, New York and Toronto 1952; X 550 pp.*, vol. 19 (Cambridge University Press, 1954).

[5] Deutsch, D. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* **400**, 97–117 (1985).

[6] Wootters, W. K. & Zurek, W. H. A single quantum cannot be cloned. *Nature* **299**, 802–803 (1982).

[7] Dieks, D. Communication by EPR devices. *Physics Letters A* **92**, 271–272 (1982).

[8] Lang, S. *Algebra* (Springer, New York, NY, 2002).

[9] Deutsch, D. & Jozsa, R. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* **439**, 553–558 (1992).

[10] Griffiths, D. J. *Introduction to Quantum Mechanics (2nd Edition)* (Pearson Prentice Hall, 2004), 2nd edn.

[11] Goemans, M. X. & Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* **42**, 1115–1145 (1995). URL https://doi.org/10.1145/227683.227684.

[12] Karloff, H. How good is the Goemans-Williamson max cut algorithm? *SIAM Journal on Computing* **29**, 336–350 (1999). URL https://doi.org/10.1137/S0097539797321481. https://doi.org/10.1137/S0097539797321481.

[13] Khot, S. On the power of unique 2-prover 1-round games 767–775 (2002). URL https://doi.org/10.1145/509907.510017.

[14] Khot, S. On the unique games conjecture. In *2010 IEEE 25th Annual Conference on Computational Complexity*, 99–121 (2010).

[15] Khot, S., Kindler, G., Mossel, E. & O'Donnell, R. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing* **37**, 319–357 (2007). URL https://doi.org/10.1137/S0097539705447372. https://doi.org/10.1137/S0097539705447372.

[16] Lin, S. Heuristic techniques for solving large combinatorial problems on a computer. *Theoretical Approaches to Non-Numerical Problem Solving* 410–418 (1970).

[17] Volpe, D., Cirillo, G. A., Zamboni, M. & Turvani, G. Integration of simulated quantum annealing in parallel tempering and population annealing for heterogeneous-profile qubo exploration. *IEEE Access* **11**, 30390–30441 (2023).

[18] Quintero, R. A. & Zuluaga, L. F. Qubo formulations of combinatorial optimization problems for quantum computing devices. *Encyclopedia of Optimization* 1–13 (2022).

[19] Qiskit textbook. https://learn.qiskit.org/summer-school/2021/lec5-2-introduction-quantum-approximate-optimization-algorithm-applications.

[20] Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm (2014). 1411.4028.

[21] Lucas, A. Ising formulations of many NP problems. *Frontiers in Physics* **2** (2014). URL https://doi.org/10.3389%2Ffphy.2014.00005.

[22] Hatano, N. & Suzuki, M. Finding exponential product formulas of higher orders. In *Quantum Annealing and Other Optimization Methods*, 37–68 (Springer Berlin Heidelberg, 2005). URL https://doi.org/10.1007%2F11526216_2.

[23] Lee, X., Saito, Y., Cai, D. & Asai, N. Parameters fixing strategy for quantum approximate optimization algorithm (2021). URL https://doi.org/10.1109%2Fqce52317.2021.00016.

[24] Qiskit 0.43.1 documentation - qiskit 0.43.1 documentation. https://qiskit.org/documentation/.

[25] Qiskit terra api reference - qiskit 0.43.1 documentation. https://qiskit.org/documentation/apidoc/terra.html.

[26] Qiskit nature overview - qiskit nature 0.6.2 documentation. https://qiskit.org/ecosystem/nature/.

[27] Qiskit finance overview - qiskit finance 0.3.4 documentation. https://qiskit.org/ecosystem/finance/.

[28] Qiskit machine learning overview - qiskit machine learning 0.6.1 documentation. https://qiskit.org/ecosystem/machine-learning/.

[29] Qiskit optimization overview - qiskit optimization 0.5.0 documentation. https://qiskit.org/ecosystem/optimization/.

[30] Qiskit metal — quantum device design & amp analysis (q-eda) 0.1.5 - qiskit metal 0.1.5 0.1.5 documentation. https://qiskit.org/ecosystem/metal/.

[31] Qiskit dynamics documentation - qiskit dynamics 0.4.1 documentation. https://qiskit.org/ecosystem/dynamics/.

[32] Qiskit experiments documentation - qiskit experiments 0.5 0.5.2 documentation. https://qiskit.org/ecosystem/experiments/.

[33] Qiskit aer documentation - qiskit aer 0.12.1 documentation. https://qiskit.org/ecosystem/aer/.

[34] Qiskit runtime overview - qiskit runtime ibm client 0.11.1 documentation. https://qiskit.org/ecosystem/ibm-runtime/.

[35] Crooks, G. E. Performance of the quantum approximate optimization algorithm on the maximum cut problem (2018). 1811.08419.

[36] Zhou, L., Wang, S.-T., Choi, S., Pichler, H. & Lukin, M. D. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X* **10**, 021067 (2020). URL https://link.aps.org/doi/10.1103/PhysRevX.10.021067.

[37] Cerezo, M., Sone, A., Volkoff, T., Cincio, L. & Coles, P. J. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications* **12** (2021).

[38] Holmes, Z., Sharma, K., Cerezo, M. & Coles, P. J. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum* **3** (2022). URL https://doi.org/10.1103%2Fprxquantum.3.010313.

[39] Sack, S. H. & Serbyn, M. Quantum annealing initialization of the quantum approximate optimization algorithm. *Quantum* **5**, 491 (2021). URL https://doi.org/10.22331%2Fq-2021-07-01-491.

[40] Egger, D. J., Mareček, J. & Woerner, S. Warm-starting quantum optimization. *Quantum* **5**, 479 (2021). URL https://doi.org/10.22331%2Fq-2021-06-17-479.

[41] Weidenfeller, J. *et al.* Scaling of the quantum approximate optimization algorithm on superconducting qubit based hardware. *Quantum* **6**, 870 (2022). URL https://doi.org/10.22331%2Fq-2022-12-07-870.

[42] Aharonov, D., Ben-Or, M., Impagliazzo, R. & Nisan, N. Limitations of noisy reversible computation (1996). quant-ph/9611028.

[43] Stilck França, D. & García-Patrón, R. Limitations of optimization algorithms on noisy quantum devices. *Nature Physics* **17**, 1221–1227 (2021).

[44] Pellizzari, T., Gardiner, S. A., Cirac, J. I. & Zoller, P. Decoherence, continuous observation, and quantum computing: A cavity qed model. *Phys. Rev. Lett.* **75**, 3788–3791 (1995). URL https://link.aps.org/doi/10.1103/PhysRevLett.75.3788.

[45] Rossi, Z. M. & Chuang, I. L. Multivariable quantum signal processing (m-QSP): prophecies of the two-headed oracle. *Quantum* **6**, 811 (2022). URL https://doi.org/10.22331%2Fq-2022-09-20-811.

[46] Ying, L. Stable factorization for phase factors of quantum signal processing. *Quantum* **6**, 842 (2022). URL https://doi.org/10.22331%2Fq-2022-10-20-842.

[47] Roffe, J. Quantum error correction: an introductory guide. *Contemporary Physics* **60**, 226–245 (2019). URL https://doi.org/10.1080/00107514.2019.1667078. https://doi.org/10.1080/00107514.2019.1667078.

[48] Holmes, A. *et al.* Nisq+: Boosting quantum computing power by approximating quantum error correction. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 556–569 (2020).

[49] J., N. M. E. *Networks an introduction* (Oxford University Press, 2018).

[50] Barabási, A.-L. & Albert, R. Emergence of scaling in random networks. *Science* **286**, 509–512 (1999). URL https://www.science.org/doi/abs/10.1126/science.286.5439.509. https://www.science.org/doi/pdf/10.1126/science.286.5439.509.

[51] Hassan, M. K., Hassan, M. Z. & Pavel, N. I. Dynamic scaling, data-collapse and self-similarity in barabási–albert networks. *Journal of Physics A: Mathematical and Theoretical* **44**, 175101 (2011). URL https://dx.doi.org/10.1088/1751-8113/44/17/175101.

[52] Krapivsky, P. L., Redner, S. & Leyvraz, F. Connectivity of growing random networks. *Physical Review Letters* **85**, 4629–4632 (2000). URL https://doi.org/10.1103%2Fphysrevlett.85.4629.