QMexico Summer School 2026

Max-Cut
y sumadores reversibles

Optimización en grafos, aritmética reversible y búsqueda cuántica

Ruta conceptual del módulo

Arquitectura de ideas

El problema Max-Cut se convierte en una búsqueda sobre coloraciones binarias. Cada arista aporta una prueba local: sus extremos están en lados distintos o no. Los sumadores reversibles agregan esas pruebas sin destruir información, los comparadores deciden si el conteo alcanza un valor objetivo y Grover amplifica las coloraciones marcadas.

Criterio de exposición

La secuencia mantiene juntas la intuición combinatoria, la implementación reversible y el significado computacional. Los desarrollos paso a paso aparecen cuando un ejemplo o un ejercicio los necesita para sostener la interpretación.

Reconstrucción documental del núcleo técnico

Tipo: Resumen

Evidencia conceptual usada

El material disponible apunta a una unidad sobre solución de Max-Cut mediante búsqueda de Grover y ejercicios en Cirq. Los elementos visibles permiten reconstruir una ruta que pasa por coloraciones binarias, detección de aristas cortadas, circuitos de suma y comparación, y marcado de soluciones.

Contenido inferido necesario

Para que el tema sea universitario, no basta con completar fragmentos de código. Es necesario explicar por qué Max-Cut es un problema de optimización, por qué la aritmética debe ser reversible y cómo el resultado de los sumadores se transforma en un criterio de marcado compatible con Grover.

Complementación académica

Se incorporan definiciones formales de grafos, cortes, suma reversible, half adder, full adder, ripple carry adder, comparadores y oráculos de fase. La matemática se mantiene al nivel necesario para justificar el mecanismo, sin reemplazar la comprensión conceptual.

Qué problema resuelve Max-Cut

Tipo: Concepto

Problema computacional

Dado un grafo, queremos dividir sus vértices en dos conjuntos de forma que el número de aristas que cruzan entre ambos conjuntos sea máximo. La dificultad no está en verificar una partición específica, sino en encontrar la mejor entre una cantidad exponencial de particiones posibles.

Por qué fue necesario estudiarlo

Max-Cut es un modelo canónico de optimización combinatoria. Resume una tensión frecuente: decisiones locales sencillas, como separar o no dos extremos de una arista, producen un objetivo global difícil cuando se consideran todas las aristas simultáneamente.

Papel dentro de la computación cuántica

El problema sirve como puente entre algoritmos de búsqueda y optimización. Permite ver cómo un predicado computable de manera reversible puede convertirse en un oráculo para Grover.

Definición formal de Max-Cut

Tipo: Concepto

Grafo y partición

Sea $G = (V, E)$ un grafo no dirigido. Un corte es una partición $V = S \cup \bar{S}$ con $S \cap \bar{S} = \varnothing$. Una arista $\{u, v\} \in E$ cruza el corte si u y v quedan en conjuntos distintos.

Función objetivo

El tamaño del corte se define como

$$\mathrm{Cut}(S) = \left|\{\{u, v\} \in E : u \in S,\ v \in \bar{S}\}\right|.$$

Max-Cut pide encontrar S que maximiza esa cantidad.

Interpretación combinatoria

Cada arista expresa una preferencia local por separar sus extremos. El problema global consiste en satisfacer la mayor cantidad posible de esas preferencias al mismo tiempo.

Grafos, particiones y cortes

Tipo: Concepto

Visualización mental

Una partición puede pensarse como colorear cada vértice con uno de dos colores. Una arista contribuye al corte si conecta vértices de colores distintos. Esta lectura evita pensar en conjuntos abstractos y conecta de inmediato con qubits binarios.

Por qué esta representación es útil

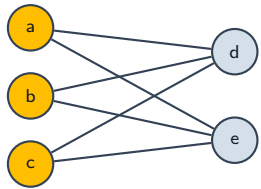
La coloración binaria permite transformar una pregunta combinatoria en una función booleana sobre cadenas de bits. Cada cadena representa una partición; el valor de la función objetivo se obtiene contando aristas cuyos extremos difieren.

Implicación computacional

Al pasar de cortes a cadenas binarias, el problema queda listo para ser explorado por un algoritmo de búsqueda. La dificultad se desplaza hacia construir un circuito que compute el tamaño del corte sin perder reversibilidad.

Ejemplo visual: un grafo bipartito pequeño

Tipo: Ejemplo



Problema que intenta resolver

El grafo tiene tres vértices a la izquierda y dos a la derecha. Todas las aristas conectan un vértice izquierdo con uno derecho. Queremos saber cuántas aristas pueden quedar cruzando un corte.

Idea ilustrada

Si ponemos $\{a, b, c\}$ de un lado y $\{d, e\}$ del otro, todas las aristas cruzan. El corte alcanza tamaño 6.

Conclusión conceptual

Un grafo bipartito ya trae una partición natural que corta todas sus aristas. Este caso sirve como referencia antes de estudiar grafos donde las restricciones compiten entre sí.

Ejercicio integrado: tamaño máximo del corte

Tipo: Ejercicio

Enunciado

Considere el grafo con vértices $\{a, b, c, d, e\}$ y aristas entre cada vértice de $\{a, b, c\}$ y cada vértice de $\{d, e\}$. Determine el tamaño máximo del corte.

Desarrollo paso a paso

Primero contamos las aristas: cada uno de los tres vértices izquierdos se conecta con dos vértices derechos, por lo que $|E| = 3 \cdot 2 = 6$. Después elegimos el corte $S = \{a, b, c\}$ y $\bar{S} = \{d, e\}$. Cada arista tiene un extremo en S y otro en $\bar{S}$, así que las seis aristas cruzan.

Interpretación

Ningún corte puede superar el número total de aristas. Como ya encontramos un corte de tamaño 6, el máximo es 6.

Variación académica: cuando no todo puede cortarse

Tipo: Ejercicio

Enunciado

Considere un triángulo con vértices $0, 1, 2$ y aristas $\{0,1\}$, $\{1,2\}$, $\{0,2\}$. ¿Puede un corte contener las tres aristas?

Desarrollo paso a paso

Si una arista cruza, sus extremos tienen colores distintos. Para cortar $\{0,1\}$ tomamos colores distintos en 0 y 1. Para cortar $\{1,2\}$, el vértice 2 debe tener color distinto de 1, por lo que queda con el mismo color que 0. Entonces $\{0,2\}$ no cruza.

Interpretación

El máximo es 2. Esta variación muestra por qué Max-Cut puede ser no trivial: las aristas imponen restricciones locales que no siempre son compatibles globalmente.

Complejidad computacional de Max-Cut

Tipo: Concepto

Limitación clásica

Para n vértices existen 2^n coloraciones binarias. Como intercambiar todos los colores no cambia el corte, hay simetría global, pero el crecimiento sigue siendo exponencial. La verificación de una coloración es fácil; la búsqueda de la óptima es difícil.

Por qué importa

Max-Cut pertenece a la familia de problemas de optimización combinatoria que sirven para modelar decisiones discretas con restricciones. En general, no se espera un algoritmo clásico exacto eficiente para todos los grafos.

Implicación cuántica

Grover no elimina la dificultad de fondo. Proporciona una aceleración cuadrática sobre la búsqueda no estructurada, siempre que podamos construir un oráculo reversible que reconozca las coloraciones deseadas.

Max-Cut como optimización binaria

Tipo: Concepto

Codificación

Asociamos a cada vértice i un bit $x_i \in \{0,1\}$. Dos vértices están en lados distintos del corte cuando $x_i \oplus x_j = 1$. Por tanto, el tamaño del corte para una cadena x es

$$C(x) = \sum_{\{i,j\} \in E} (x_i \oplus x_j).$$

Qué problema resuelve esta fórmula

Convierte un objeto gráfico en una función aritmética sobre bits. Esta traducción es el punto de entrada para circuitos: CNOT calcula XOR, Toffoli calcula productos lógicos y los sumadores agregan indicadores.

Significado computacional

El máximo corte es $\max_x C(x)$. Grover puede buscar cadenas x cuyo conteo sea máximo o supere un umbral.

Codificación binaria de colores

Tipo: Concepto

Convención

Usaremos 0 para un color y 1 para el otro. La elección de nombres de colores es arbitraria; lo importante es la diferencia entre bits. Si se intercambian todos los colores, el tamaño del corte no cambia.

Papel dentro del algoritmo

La coloración se almacena en el registro de búsqueda. Cada qubit representa la decisión de color de un vértice. La cadena completa representa una partición del grafo.

Implicación

El problema combinatorio se vuelve una búsqueda sobre estados base. Un oráculo de Max-Cut debe leer esa cadena, calcular cuántas aristas cruzan y marcar las cadenas que cumplen el criterio.

Ejercicio: estado que representa una coloración

Tipo: Ejercicio

Enunciado

Tres vértices se codifican con tres qubits. El vértice 0 tiene color 1, el vértice 1 tiene color 0 y el vértice 2 tiene color 0. ¿Qué estado base representa la coloración?

Desarrollo paso a paso

La convención indica que el qubit i almacena el color del vértice i. Por tanto, el primer bit es $x_0 = 1$, el segundo es $x_1 = 0$ y el tercero es $x_2 = 0$.

$$x = x_0 x_1 x_2 = 100.$$

Interpretación

El estado es $|100\rangle$. Esta notación no describe amplitudes todavía; describe una coloración clásica embebida como estado base de un registro cuántico.

Variación: codificación con orden explícito

Tipo: Ejercicio

Enunciado

Para cuatro vértices, use la convención $x = x_0x_1x_2x_3$. Si los colores son $x_0 = 0$, $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, determine el estado base.

Desarrollo

Se escribe la cadena en el mismo orden en que se etiquetan los vértices:

$$x = 0110.$$

Por tanto, la coloración se representa como $|0110\rangle$.

Interpretación

La variación separa dos ideas que suelen confundirse: la etiqueta matemática del vértice y el orden de lectura de bits en un marco de programación. En circuitos reales se debe fijar una convención y mantenerla.

Vértices como qubits

Tipo: Concepto

Qué problema resuelve

Representar vértices mediante qubits permite consultar muchas coloraciones de manera coherente. El registro de n qubits contiene las 2^n etiquetas de coloración como estados base posibles.

Por qué fue necesario introducirlo

Grover necesita un espacio de búsqueda sobre el cual aplicar el oráculo y la difusión. Para Max-Cut, ese espacio no son vértices individuales, sino coloraciones completas del grafo.

Papel algorítmico

Los qubits de color no deben destruirse durante el cálculo del corte. Se usan como controles para calcular indicadores, sumas y comparaciones en registros auxiliares reversibles.

Indicador de arista cortada

Tipo: Concepto

Definición mínima

Para una arista $\{i, j\}$, definimos

$$e_{ij} = x_i \oplus x_j.$$

Si $e_{ij} = 1$, los extremos tienen colores distintos y la arista cruza el corte. Si $e_{ij} = 0$, ambos extremos están del mismo lado.

Qué problema resuelve

Reduce una condición gráfica a una operación lógica elemental. El grafo completo se analiza calculando un indicador por arista y sumando esos indicadores.

Implicación para circuitos

Como XOR se implementa naturalmente con CNOT, cada arista puede computarse en un qubit auxiliar usando dos CNOTs desde los qubits de sus extremos.

CNOT como cálculo de XOR

Tipo: Concepto

Acción lógica

La compuerta CNOT transforma $|a\rangle\,|b\rangle$ en $|a\rangle\,|b \oplus a\rangle$. Si el segundo qubit inicia en 0, el resultado almacena a.

Cálculo de una arista

Para calcular $x_i \oplus x_j$ en un auxiliar t inicializado en 0, se aplican dos CNOTs:

$$t \leftarrow t \oplus x_i, \qquad t \leftarrow t \oplus x_j.$$

Al final, $t = x_i \oplus x_j$.

Significado computacional

El auxiliar no es una copia permanente de la solución; es una memoria reversible intermedia que permite construir el conteo de aristas cortadas.

Ejercicio integrado: detectar colores distintos

Tipo: Ejercicio

Enunciado

Los qubits 1 y 2 representan colores de dos vértices, y el qubit 0 empieza en 0. Queremos que el qubit 0 termine en 1 si los colores son distintos y en 0 si son iguales.

Desarrollo paso a paso

Se necesita almacenar $x_1 \oplus x_2$ en el qubit 0. Primero aplicamos CNOT desde el qubit 1 hacia el qubit 0, con lo cual $q_0 = x_1$. Después aplicamos CNOT desde el qubit 2 hacia el qubit 0, con lo cual $q_0 = x_1 \oplus x_2$.

Interpretación

El procedimiento no decide todavía si la coloración es óptima. Solo traduce una arista del grafo en un bit lógico que podrá sumarse con los demás indicadores.

Variación: indicador de arista con otro auxiliar

Tipo: Ejercicio

Enunciado

Los colores de los vértices 0 y 3 están en q_0 y q_3. El auxiliar q_6 inicia en 0. Construya el cálculo reversible de $e_{03} = x_0 \oplus x_3$.

Desarrollo

El auxiliar debe recibir la paridad de ambos colores:

$$q_6 \leftarrow q_6 \oplus q_0, \qquad q_6 \leftarrow q_6 \oplus q_3.$$

En código de circuito esto corresponde a dos CNOTs con controles q_0 y q_3, ambas dirigidas a q_6.

Interpretación

La variación muestra que el patrón es independiente de los índices: una arista siempre se representa como una XOR entre colores.

Del indicador local al tamaño del corte

Tipo: Concepto

Paso global

Una vez calculado e_{ij} para cada arista, el tamaño del corte es la suma de esos bits:

$$C(x) = \sum_{\{i,j\} \in E} e_{ij}.$$

Por qué aparece aritmética

El oráculo de Grover necesita saber si una coloración cumple un criterio global, como $C(x) = C_{\text{máx}}$ o $C(x) \geq T$. Para decidirlo, el circuito debe sumar muchos bits de arista y comparar el resultado.

Implicación

Los sumadores no son un detalle auxiliar. Son el mecanismo que convierte información local de aristas en una condición global de optimización.

Oráculo local de aristas en Cirq

Tipo: Método

Idea de implementación

Un fragmento de circuito puede calcular indicadores de varias aristas en qubits auxiliares. Cada indicador recibe dos CNOTs, uno por extremo de la arista. Si todos los indicadores relevantes valen 1, una compuerta controlada marca un qubit de decisión.

```
# Estructura conceptual en Cirq
# q0..q4: colores; q5..q8: indicadores; q9: marca
yield CX(qq[0], qq[5]); yield CX(qq[1], qq[5])
yield CX(qq[1], qq[6]); yield CX(qq[3], qq[6])
yield CX(qq[2], qq[7]); yield CX(qq[3], qq[7])
yield CX(qq[3], qq[8]); yield CX(qq[4], qq[8])
yield X(qq[9]).controlled_by(*(qq[5:9]))
```

Interpretación

El qubit q_9 no conoce el grafo por sí mismo. Solo se activa porque los auxiliares $q_5, \ldots, q_8$ ya almacenan condiciones locales.

Ejemplo: cuatro indicadores y una marca

Tipo: Ejemplo

Problema que intenta resolver

Supongamos que ya calculamos cuatro indicadores de arista en q_5, q_6, q_7, q_8. Queremos activar q_9 únicamente cuando los cuatro indicadores son 1.

Idea cuántica ilustrada

La compuerta multi-controlada implementa una conjunción reversible. No pregunta por una arista aislada; combina varias condiciones locales en una sola decisión lógica.

Significado computacional

Cuando se usa $X(q_9)$ controlada por $q_5 : q_8$, el qubit q_9 almacena que todas las aristas consideradas cruzan. En una versión completa, esa marca puede convertirse en una fase para Grover.

Ejercicio integrado: completar la marca global

Tipo: Ejercicio

Enunciado

Los qubits q_5, q_6, q_7, q_8 almacenan indicadores de cuatro aristas. El qubit q_9 inicia en 0. Se desea que q_9 cambie a 1 solo si los cuatro indicadores son 1.

Desarrollo paso a paso

La condición lógica es $q_5 \wedge q_6 \wedge q_7 \wedge q_8$. En un circuito reversible se implementa como una compuerta X multi-controlada sobre q_9:

$$q_9 \leftarrow q_9 \oplus (q_5 q_6 q_7 q_8).$$

Interpretación

El resultado no calcula todavía el máximo corte; reconoce una propiedad particular de la coloración. Esta idea se generaliza reemplazando la conjunción simple por suma y comparación.

Oráculo de Max-Cut: forma conceptual

Tipo: Concepto

Qué problema resuelve

El oráculo de Max-Cut debe distinguir coloraciones óptimas de no óptimas sin medir el registro de búsqueda. Para Grover, esa distinción debe convertirse en una fase o en una marca reversible que luego se use como fase.

Estructura

El flujo conceptual es

$$\text{colores} \to \text{indicadores de arista} \to \text{conteo} \to \text{comparación} \to \text{marca}.$$

Implicación

La dificultad práctica está en construir este proceso de manera reversible y luego deshacer los registros auxiliares. Sin esa limpieza, la difusión de Grover actúa sobre un estado contaminado por información residual.

Grover como motor de búsqueda

Tipo: Concepto

Papel de Grover

Una vez que existe un oráculo que marca coloraciones con un corte deseado, Grover amplifica esas coloraciones dentro del espacio de todas las particiones. La búsqueda no explora el grafo arista por arista; explora coloraciones completas.

Limitación clásica que intenta superar

Clásicamente, si no se aprovecha estructura adicional, se revisan muchas coloraciones. Con n vértices hay 2^n cadenas posibles. Grover reduce el número de consultas al orden de la raíz cuadrada del espacio relevante.

Significado

La ventaja cuántica no aparece por conocer el máximo de antemano, sino por poder reconocer candidatos válidos y amplificarlos coherentemente.

Espacio de búsqueda de coloraciones

Tipo: Concepto

Superposición inicial

El registro de color se prepara como

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \, .$$

Cada x representa una coloración completa del grafo.

Por qué se utiliza

Grover necesita comenzar con amplitudes distribuidas de forma uniforme, porque el oráculo solo introduce una diferencia de fase entre candidatos marcados y no marcados. La difusión convierte esa diferencia en mayor probabilidad de medición.

Interpretación física

El algoritmo manipula amplitudes asociadas a particiones, no a vértices aislados. La unidad de búsqueda es la coloración completa.

Simetría de cortes complementarios

Tipo: Concepto

Observación

Si una coloración x produce un corte, la coloración complementaria $\bar{x}$ produce exactamente el mismo corte. Todas las aristas que cruzaban siguen cruzando, porque invertir ambos extremos conserva la diferencia de colores.

Por qué importa

Esta simetría duplica soluciones en el espacio binario. Para análisis combinatorio puede fijarse un vértice como referencia; para Grover puede dejarse la simetría y amplificar ambas coloraciones óptimas.

Implicación computacional

El número de soluciones marcadas k afecta el número de iteraciones de Grover. La simetría de complemento debe considerarse al estimar k.

Marcar óptimos frente a conocer óptimos

Tipo: Concepto

Distinción central

Un oráculo no debe entenderse como una lista de soluciones precargadas. En una formulación algorítmica, el oráculo implementa un criterio verificable: calcula $C(x)$ y decide si cumple una condición.

Por qué fue necesario aclararlo

Si ya conociéramos las coloraciones óptimas como lista, la búsqueda habría terminado. El valor de Grover está en amplificar estados reconocibles por un predicado, no en usar conocimiento externo de la solución.

Papel de los sumadores

Los sumadores y comparadores permiten implementar ese predicado sin conocer de antemano qué cadenas serán óptimas.

Del umbral al máximo

Tipo: Concepto

Problema práctico

Para marcar cortes máximos se necesita saber el valor máximo $C_{\text{máx}}$ o encontrarlo progresivamente. Un enfoque pedagógico consiste en usar un umbral T y marcar coloraciones con $C(x) \geq T$.

Por qué se utiliza

Los comparadores binarios implementan naturalmente condiciones como igualdad o superación de un umbral. Esto permite convertir una tarea de optimización en una familia de tareas de búsqueda.

Interpretación

La optimización se transforma en reconocimiento. Grover no maximiza por sí mismo; amplifica las coloraciones que el oráculo declara aceptables.

Limitaciones del enfoque cuántico

Tipo: Concepto

Ventaja y alcance

Grover ofrece una mejora cuadrática en el número de consultas para búsqueda no estructurada. Esa mejora es importante, pero no convierte Max-Cut general en un problema fácil.

Costo del oráculo

El circuito que calcula indicadores, suma aristas, compara y descomputa puede ser costoso en compuertas y qubits auxiliares. En problemas reales, ese costo influye tanto como el número de iteraciones.

Implicación académica

El módulo no debe leerse como "Grover resuelve Max-Cut eficientemente". La lectura correcta es que Max-Cut ilustra cómo una función objetivo combinatoria puede convertirse en un oráculo reversible para búsqueda cuántica.

Transición: por qué aparecen los sumadores

Tipo: Resumen

Necesidad conceptual

Max-Cut pide contar cuántas aristas cruzan. El cálculo de una arista usa XOR; el cálculo del corte requiere sumar muchos bits. Por eso los sumadores son parte central del algoritmo y no un tema secundario.

Necesidad física

Los circuitos cuánticos cerrados son reversibles. No basta con escribir una suma clásica y descartar información intermedia; hay que preservar o descomputar los registros auxiliares.

Consecuencia

Los adders son el puente entre una función objetivo clásica y un oráculo cuántico utilizable por Grover.

Computación reversible y aritmética

Tipo: Concepto

Qué problema resuelve

La aritmética reversible permite calcular funciones clásicas dentro de circuitos unitarios. Si una función no es reversible, se embebe usando registros auxiliares que conservan la entrada y acumulan la salida.

Por qué fue necesario introducirla

Una evolución cuántica ideal no puede borrar información arbitrariamente. Operaciones como "calcular y desechar" no son válidas como transformaciones unitarias cerradas.

Papel dentro de Max-Cut

Para construir el oráculo se computa el tamaño del corte, se usa el resultado para marcar, y después se deshace el cálculo. Esta secuencia exige aritmética reversible.

XOR como suma módulo 2

Tipo: Concepto

Definición

La suma módulo 2 coincide con XOR:

$$a \oplus b = (a + b) \bmod 2.$$

El resultado es 1 cuando exactamente uno de los bits vale 1.

Por qué es fundamental

Las coloraciones usan bits, las aristas cruzadas se detectan con diferencia de bits y las CNOTs implementan precisamente XOR sobre un objetivo. Por eso XOR aparece en todos los niveles del módulo.

Implicación

La misma operación lógica sirve para detectar aristas cortadas y para construir el bit de suma de un half adder.

CNOT como operador XOR reversible

Tipo: Concepto

Acción

$$\mathrm{CNOT}\,|a\rangle\,|b\rangle = |a\rangle\,|b \oplus a\rangle\,.$$

La entrada de control queda intacta y el objetivo cambia por XOR.

Limitación clásica superada

La operación clásica $b \leftarrow b \oplus a$ es reversible si se conserva a. CNOT realiza exactamente esa actualización sin destruir la información de control.

Papel en adders

CNOT permite propagar bits hacia registros de suma. En un circuito de Max-Cut, el mismo mecanismo calcula indicadores de aristas y bits de suma módulo 2.

Ejercicio integrado: suma módulo 2

Tipo: Ejercicio

Enunciado

Los qubits q_0 y q_1 almacenan bits a y b. El qubit q_2 inicia en 0. Queremos almacenar $a \oplus b$ en q_2.

Desarrollo paso a paso

Primero aplicamos CNOT de q_0 a q_2, así $q_2 = a$. Luego aplicamos CNOT de q_1 a q_2, así $q_2 = a \oplus b$.

$$q_2 : 0 \longmapsto a \longmapsto a \oplus b.$$

Interpretación

El resultado es la suma módulo 2. Esta operación es la parte de suma de un half adder, pero todavía no calcula el acarreo.

Variación: suma módulo 2 con objetivo no nulo

Tipo: Ejercicio

Enunciado

Si el objetivo q_2 inicia en c en lugar de 0, ¿qué almacenan las dos CNOTs controladas por q_0 y q_1?

Desarrollo

La primera CNOT actualiza $q_2 \leftarrow c \oplus a$. La segunda actualiza $q_2 \leftarrow c \oplus a \oplus b$.

Interpretación

La operación no "sobrescribe" el objetivo; acumula por XOR. Esta propiedad es útil para circuitos reversibles, pero también exige inicializar y limpiar auxiliares con cuidado.

Toffoli y conjunción reversible

Tipo: Concepto

Acción lógica

La compuerta Toffoli, o CCNOT, realiza

$$|a\rangle\,|b\rangle\,|c\rangle \mapsto |a\rangle\,|b\rangle\,|c \oplus ab\rangle\,.$$

El objetivo cambia solo cuando ambos controles son 1.

Por qué es necesaria

Los acarreos de suma requieren productos lógicos como ab. CNOT basta para XOR, pero no para detectar simultáneamente dos bits activos.

Papel dentro de adders

Toffoli permite construir el bit de acarreo del half adder y las condiciones de mayoría del full adder.

Half adder: suma de dos bits

Tipo: Concepto

Problema que resuelve

El half adder suma dos bits a y b y produce dos salidas: suma y acarreo. Es el bloque mínimo que separa la suma módulo 2 del desbordamiento hacia el siguiente bit.

Definición formal

$$s = a \oplus b, \qquad c = ab.$$

El bit s contiene la paridad, mientras que c indica si la suma $a + b$ alcanzó el valor 2.

Implicación para Max-Cut

Contar aristas cortadas exige sumar muchos indicadores. El half adder es el primer paso para convertir bits de arista en números binarios.

Half adder reversible

Tipo: Método

Construcción conceptual

Con auxiliares $s = 0$ y $c = 0$, se calcula $s = a \oplus b$ con dos CNOTs y $c = ab$ con una Toffoli. Las entradas a, b permanecen disponibles.

```
# Qiskit: half adder conceptual
# a, b: entradas; s, c: auxiliares inicializados en 0
qc.cx(a, s)
qc.cx(b, s)
qc.ccx(a, b, c)
```

Interpretación

El circuito no destruye los sumandos. Esa preservación es lo que permite descomputar después de usar el resultado para marcar una solución.

Ejercicio integrado: tabla del half adder

Tipo: Ejercicio

Enunciado

Construya la salida (c, s) del half adder para las entradas $a, b \in \{0, 1\}$.

Desarrollo paso a paso

Si $(a, b) = (0, 0)$, entonces $s = 0$ y $c = 0$. Si $(0, 1)$ o $(1, 0)$, entonces $s = 1$ y $c = 0$. Si $(1, 1)$, entonces $s = 0$ y $c = 1$, porque $1 + 1 = 2$ en binario.

$$00 \mapsto 00, \quad 01 \mapsto 01,$$
$$10 \mapsto 01, \quad 11 \mapsto 10.$$

Interpretación

El half adder muestra por qué contar aristas requiere más que XOR: cuando dos aristas cruzan, aparece un acarreo.

Full adder: incluir un acarreo de entrada

Tipo: Concepto

Problema que resuelve

Al sumar más de dos bits, aparece un acarreo proveniente de una posición anterior. El full adder suma a, b y c_{in}, y produce un bit de suma y un nuevo acarreo.

Definición útil

$$s = a \oplus b \oplus c_{in}, \qquad c_{out} = ab \oplus c_{in}(a \oplus b).$$

La segunda expresión indica que hay acarreo si al menos dos de los tres bits de entrada valen 1.

Papel en circuitos grandes

El full adder es el componente que permite encadenar sumas y construir contadores binarios de muchos indicadores.

Full adder reversible: interpretación

Tipo: Método

Estructura

Una implementación reversible usa un auxiliar temporal para $a \oplus b$, un objetivo para la suma y otro para el acarreo. La clave es que los temporales pueden descomputarse al final.

```
# t inicia en 0, s inicia en 0, cout inicia en 0
qc.cx(a, t); qc.cx(b, t)        # t = a xor b
qc.cx(t, s); qc.cx(cin, s)      # s = a xor b xor cin
qc.ccx(a, b, cout)              # contribucion ab
qc.ccx(cin, t, cout)            # contribucion cin(a xor b)
qc.cx(b, t); qc.cx(a, t)        # limpia t
```

Interpretación

La limpieza del temporal no es estética. Si t queda entrelazado con la entrada, el oráculo deja residuos que interfieren con la difusión de Grover.

Ripple carry adder

Tipo: Concepto

Qué problema resuelve

Un ripple carry adder suma números de varios bits propagando el acarreo desde el bit menos significativo hasta el más significativo. Es una forma modular de construir aritmética binaria reversible.

Por qué aparece en Max-Cut

El conteo de aristas cortadas puede requerir sumar muchos bits. Una estrategia consiste en ir acumulando indicadores dentro de un registro binario de conteo, propagando acarreos cuando sea necesario.

Implicación de costo

La profundidad del circuito crece porque el acarreo debe propagarse. Existen diseños más avanzados, pero el ripple carry es pedagógicamente claro y suficiente para entender el mecanismo.

Conteo de aristas cortadas

Tipo: Concepto

De bits locales a número global

Cada indicador e_{ij} es un bit. El tamaño del corte es un número binario que puede tomar valores entre 0 y $|E|$. Por tanto, el registro de conteo debe tener suficientes bits para representar ese rango.

Por qué fue necesario

El comparador no puede decidir si una coloración es óptima si el circuito solo conoce aristas aisladas. Necesita un número que resuma el comportamiento global de la coloración.

Papel algorítmico

El conteo es el puente entre el grafo y el criterio de marcado. Sin conteo reversible, el oráculo no puede distinguir cortes pequeños de cortes grandes.

Ejercicio integrado: bits para sumar diez bits

Tipo: Ejercicio

Enunciado

¿Cuántos bits se necesitan para almacenar el resultado de sumar diez bits individuales $q_0 + q_1 + \cdots + q_9$?

Desarrollo paso a paso

La suma mínima es 0 y la máxima es 10. El registro debe representar todos los valores de 0 a 10, es decir, 11 valores posibles. Con m bits hay 2^m valores. Como $2^3 = 8 < 11$ y $2^4 = 16 \geq 11$, se necesitan $m = 4$ bits.

Interpretación

El número de bits no depende de cuántos sumandos haya únicamente, sino del valor máximo que la suma puede alcanzar.

Variación: bits para contar aristas de otro grafo

Tipo: Ejercicio

Enunciado

Un grafo tiene 6 aristas. ¿Cuántos bits necesita un registro para almacenar el tamaño del corte?

Desarrollo

El conteo puede tomar valores de 0 a 6, es decir, 7 valores. Con 2 bits solo hay 4 valores; con 3 bits hay 8 valores. Por tanto, se necesitan 3 bits.

Interpretación

Este cálculo determina el tamaño mínimo del registro de conteo para el oráculo. Si el registro es demasiado pequeño, el conteo se desborda y el comparador marca incorrectamente.

Costo en qubits auxiliares

Tipo: Concepto

Necesidad

Los auxiliares almacenan indicadores de arista, bits temporales de suma, acarreos y resultados de comparación. Sin ellos, no se puede construir una versión reversible clara del cálculo clásico.

Limitación práctica

Cada auxiliar incrementa el tamaño del circuito y la demanda de hardware. Además, los auxiliares deben limpiarse si el circuito se usará dentro de una iteración de Grover.

Implicación

Un oráculo conceptualmente simple puede tener un costo físico considerable. Analizar Max-Cut cuántico requiere mirar tanto el número de iteraciones de Grover como el costo del circuito que implementa el predicado.

Comparadores binarios

Tipo: Concepto

Qué problema resuelven

Un comparador decide si un registro binario coincide con un valor objetivo o supera un umbral. En Max-Cut, permite convertir el conteo de aristas cortadas en una marca lógica.

Igualdad

Para comprobar si un registro es igual a una cadena, se usan controles positivos sobre bits que deben ser 1 y controles negativos sobre bits que deben ser 0. Los controles negativos se implementan rodeando el control con compuertas X.

Papel dentro del oráculo

El comparador produce el bit que luego se transforma en fase. Sin comparador, el circuito podría contar aristas, pero no decidir qué coloraciones amplificar.

Comparator para el valor 1011_2

Tipo: Concepto

Problema

Un registro de cuatro qubits almacena $b_3b_2b_1b_0$, con b_i en el qubit correspondiente. Queremos activar un quinto qubit si el número es 1011_2.

Idea de controles negativos

El patrón 1011 tiene un cero en la posición b_2. Para usar una compuerta multi-controlada que se active sobre todos unos, se aplica X sobre el qubit que representa el cero, se aplica la compuerta controlada y luego se deshace el X.

Interpretación

El comparador no suma ni optimiza; reconoce una condición exacta. En Max-Cut, esa condición puede ser "el conteo es igual al máximo conocido".

Ejercicio integrado: detectar 1011_2

Tipo: Ejercicio

Enunciado

El registro q_0, q_1, q_2, q_3 representa b_0, b_1, b_2, b_3 respectivamente. El objetivo q_4 inicia en 0. Construya la marca para el número 1011_2.

Desarrollo paso a paso

El patrón 1011_2 significa $b_3 = 1$, $b_2 = 0$, $b_1 = 1$, $b_0 = 1$. Como b_2 está en q_2, aplicamos $X(q_2)$ para convertir el cero esperado en uno. Luego aplicamos $X(q_4)$ controlada por q_0, q_1, q_2, q_3 y finalmente deshacemos $X(q_2)$.

Interpretación

La secuencia activa q_4 solo para el patrón deseado y restaura el registro original. Esa restauración es esencial para mantener reversibilidad.

Variación: detectar 0110_2

Tipo: Ejercicio

Enunciado

Con la misma convención, diseñe la idea para marcar el valor 0110_2.

Desarrollo

El patrón 0110_2 tiene $b_3 = 0$, $b_2 = 1$, $b_1 = 1$, $b_0 = 0$. Por tanto, se aplican X sobre q_3 y q_0 antes de la multi-controlada, se activa el objetivo con todos los controles, y luego se deshacen esas dos compuertas X.

Interpretación

La técnica de controles negativos es general: cualquier patrón binario se convierte temporalmente en el patrón de todos unos.

Igualdad frente a umbral

Tipo: Concepto

Comparador de igualdad

Marca exactamente un valor, por ejemplo $C(x) = 6$. Es adecuado cuando se conoce el máximo o cuando se estudia un caso pequeño de forma controlada.

Comparador de umbral

Marca valores que satisfacen $C(x) \geq T$. Es útil cuando se busca el máximo probando umbrales o cuando se desea aceptar soluciones suficientemente buenas.

Implicación para Grover

El número de estados marcados cambia según el criterio. Un criterio más amplio produce más soluciones marcadas y modifica el número de iteraciones necesario.

Detección de valores máximos

Tipo: Concepto

Qué problema resuelve

Para usar Grover en optimización, el oráculo debe traducir "ser óptimo" en una condición booleana. Cuando se conoce $C_{\text{máx}}$, se marca $C(x) = C_{\text{máx}}$. Cuando no se conoce, se puede explorar con umbrales.

Por qué fue necesario

Grover resuelve búsqueda de estados marcados, no optimización directamente. El comparador es la pieza que convierte una función objetivo en una familia de predicados de búsqueda.

Significado computacional

La optimización se implementa como reconocimiento reversible de buenos candidatos. La calidad del oráculo determina qué amplitudes se amplifican.

De marca lógica a marca de fase

Tipo: Concepto

Problema que resuelve

Grover necesita que las soluciones tengan una fase distinta. Si el comparador produce un bit de marca $m(x)$, ese bit puede usarse para aplicar un cambio de fase condicionado.

Procedimiento conceptual

Se calcula $m(x)$, se aplica una fase $(-1)^{m(x)}$ al registro de búsqueda, y después se deshace el cálculo de $m(x)$. El estado final conserva solo la fase útil.

Interpretación física

La información de conteo no queda como residuo observable en auxiliares. Se transfiere a una fase relativa entre coloraciones marcadas y no marcadas.

Descomputar: cerrar el oráculo

Tipo: Concepto

Qué problema resuelve

Después de calcular indicadores, conteos y marcas, los registros auxiliares pueden quedar correlacionados con la coloración. Si no se limpian, la difusión de Grover no actúa sobre un espacio de coloraciones puro.

Procedimiento

Una vez aplicada la fase, se invierten las operaciones de comparación, suma e indicadores en orden inverso. Esto devuelve los auxiliares a $|0\rangle$ y conserva la fase sobre la coloración.

Implicación

Un oráculo correcto no solo calcula; también descalcula. Esta es una diferencia fundamental entre programar un predicado clásico y programarlo para un algoritmo cuántico.

Los adders no son un tema secundario

Tipo: Concepto

Razón conceptual

Max-Cut exige una cantidad global: el número de aristas cortadas. Esa cantidad no está disponible con una sola operación lógica local. Los adders son los circuitos que agregan evidencia local hasta producir un número comparable.

Razón física

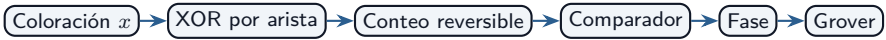
La suma debe ser reversible para poder usarse dentro de un oráculo unitario. Esto obliga a pensar en acarreos, auxiliares y descomputación, no solo en fórmulas clásicas.

Implicación para optimización cuántica

Cualquier problema que requiera computar una función objetivo discreta enfrentará el mismo patrón: calcular características locales, sumarlas, comparar y marcar.

Mapa integrado extremo a extremo

Tipo: Resumen



Lectura global

El flujo muestra cómo un problema de grafos se transforma en búsqueda cuántica. La parte combinatoria define qué se quiere optimizar; la parte aritmética reversible permite reconocerlo; Grover amplifica los estados reconocidos.

Implicación

La integración es el punto central del módulo: Max-Cut proporciona el problema, los adders proporcionan el mecanismo de cálculo y Grover proporciona la amplificación.

Del grafo al predicado reversible

Tipo: Concepto

Qué problema resuelve

Necesitamos convertir una instancia gráfica en una función booleana $g(x)$ que sea 1 cuando la coloración x debe marcarse. Esta función no es simplemente el tamaño del corte; es una decisión basada en ese tamaño.

Construcción

Primero se calcula $C(x)$. Luego se define, por ejemplo,

$$g(x) = 1 \quad \text{si y solo si} \quad C(x) = T.$$

El valor T puede ser el máximo conocido o un umbral exploratorio.

Papel cuántico

El predicado g se implementa reversiblemente y se usa para producir una fase sobre las coloraciones aceptadas.

Oráculo completo de Max-Cut

Tipo: Concepto

Secuencia funcional

El oráculo aplica cuatro etapas: calcula indicadores de arista, suma los indicadores, compara el conteo con el criterio de marcado y deshace los cálculos intermedios después de aplicar la fase.

Por qué se utiliza este procedimiento

Cada etapa resuelve una limitación distinta. Los indicadores transforman grafos en bits; los sumadores transforman bits en número; los comparadores transforman número en decisión; la fase transforma decisión en señal útil para Grover.

Significado computacional

El oráculo no entrega el valor de Max-Cut al medirlo. Prepara el espacio de amplitudes para que Grover haga más probable observar una coloración óptima.

Ejemplo: Max-Cut del grafo bipartito

Tipo: Ejemplo

Problema

En el grafo con partes $\{a, b, c\}$ y $\{d, e\}$, queremos interpretar una coloración óptima como cadena binaria. Asignemos color 0 a a, b, c y color 1 a d, e.

Idea ilustrada

La cadena conceptual es 00011 si el orden de vértices es a, b, c, d, e. Cada arista conecta un bit 0 con un bit 1, por lo que cada indicador de arista vale 1.

Conclusión

El conteo es 6. La coloración complementaria 11100 produce el mismo corte. En una búsqueda de Grover, ambas podrían ser estados marcados.

Ejercicio integrado: conteo para una coloración

Tipo: Ejercicio

Enunciado

Use el grafo bipartito con partes $\{a, b, c\}$ y $\{d, e\}$. En el orden a, b, c, d, e, calcule el tamaño del corte para $x = 00111$.

Desarrollo paso a paso

Los colores son $a = 0$, $b = 0$, $c = 1$, $d = 1$, $e = 1$. Las aristas desde a y b hacia d, e cruzan: aportan 4. Las aristas desde c hacia d, e no cruzan porque ambos extremos tienen color 1: aportan 0.

$$C(00111) = 4.$$

Interpretación

Aunque el grafo tiene un corte de tamaño 6, esta coloración no es óptima. El oráculo debería no marcarla si el criterio es $C(x) = 6$.

Variación: conteo para un ciclo de cuatro vértices

Tipo: Ejercicio

Enunciado

Considere el ciclo $0-1-2-3-0$ y la coloración $x=0101$. Determine el tamaño del corte.

Desarrollo

Cada arista conecta colores distintos: 0 con 1, 1 con 0, 0 con 1 y 1 con 0. Por tanto, los cuatro indicadores valen 1 y el conteo es 4.

Interpretación

El ciclo par admite una coloración alternante que corta todas sus aristas. Esta variación refuerza la conexión entre bipartición y cortes máximos completos.

Grover sobre coloraciones

Tipo: Concepto

Proceso

El algoritmo prepara una superposición uniforme de coloraciones, aplica el oráculo de Max-Cut para cambiar la fase de las coloraciones marcadas y usa difusión para aumentar sus amplitudes.

Número de soluciones

Si hay k coloraciones marcadas entre $N = 2^n$, el número de iteraciones escala aproximadamente como

$$r \approx \frac{\pi}{4}\sqrt{\frac{N}{k}}.$$

Interpretación

Un mayor número de coloraciones óptimas reduce el número de iteraciones necesarias. La estructura de simetrías del grafo puede cambiar k.

Ejercicio integrado: iteraciones aproximadas

Tipo: Ejercicio

Enunciado

Un grafo tiene 5 vértices, por lo que hay $N = 32$ coloraciones. Suponga que el oráculo marca dos coloraciones óptimas complementarias. Estime el número de iteraciones de Grover.

Desarrollo paso a paso

Usamos $k = 2$ y

$$r \approx \frac{\pi}{4}\sqrt{\frac{32}{2}} = \frac{\pi}{4}\sqrt{16} = \frac{\pi}{4} \cdot 4 = \pi \approx 3.14.$$

Se redondea a 3 iteraciones.

Interpretación

El cálculo no garantiza éxito absoluto, pero ubica el punto donde la amplitud marcada está cerca de su máximo antes de sobrerrotar.

Comparación clásica y cuántica

Tipo: Concepto

Enfoque clásico directo

Revisar coloraciones una por una cuesta $O(2^n)$ verificaciones en el peor caso si no se usa estructura adicional. Cada verificación calcula el tamaño del corte de una coloración.

Enfoque con Grover

Grover reduce el número de llamadas al predicado marcado a $O(\sqrt{2^n/k})$. La mejora es cuadrática, no exponencial.

Matiz esencial

La complejidad total incluye el costo del oráculo. En Max-Cut, construir y ejecutar el cálculo reversible del conteo puede ser una parte dominante del circuito.

Cirq: qué debe expresar el código

Tipo: Método

Objetivo

El código en Cirq debe expresar tres ideas: los qubits de color representan vértices, los auxiliares almacenan indicadores o marcas, y las compuertas controladas implementan condiciones lógicas reversibles.

```
import cirq
qq = cirq.LineQubit.range(10)

# Ejemplo: indicador de arista (u, v) en auxiliar t
circuit.append(cirq.CNOT(qq[u], qq[t]))
circuit.append(cirq.CNOT(qq[v], qq[t]))
```

Interpretación

Cada CNOT representa una contribución al XOR de colores. El auxiliar termina almacenando si la arista cruza el corte.

Cirq: marca controlada por indicadores

Tipo: Método

Construcción

Una vez computados varios indicadores, una compuerta multi-controlada puede activar un qubit de marca. En Cirq, la sintaxis `controlled_by` expresa controles múltiples.

```
# q5, q6, q7, q8 almacenan condiciones locales
# q9 se activa si todas son verdaderas
circuit.append(cirq.X(qq[9]).controlled_by(*qq[5:9]))
```

Interpretación

Esta operación implementa una conjunción reversible. Para Max-Cut general, la conjunción directa se reemplaza por conteo y comparación, pero la idea de marca condicionada es la misma.

Ejercicio integrado Cirq: completar controles

Tipo: Ejercicio

Enunciado

En un circuito con auxiliares q_5, q_6, q_7, q_8, escriba conceptualmente la operación que activa q_9 solo si todos esos auxiliares valen 1.

Desarrollo

La condición es una conjunción de cuatro bits. En Cirq se expresa como una compuerta X sobre q_9 controlada por la lista q_5, q_6, q_7, q_8:

$$X(q_9) \text{ controlada por } q_5, q_6, q_7, q_8.$$

Interpretación

El ejercicio enfatiza que la marca global solo es válida si los auxiliares ya fueron computados correctamente. El orden lógico del circuito importa.

Cirq: estructura de un oráculo limpio

Tipo: Método

Proceso recomendado

Primero se calculan los indicadores. Después se aplica la marca condicionada. Finalmente se aplican de nuevo las mismas operaciones de indicadores en orden inverso para limpiar auxiliares.

```
def compute_edges(circuit, qq, edges, start):
    for t, (u, v) in enumerate(edges, start=start):
        circuit.append(cirq.CNOT(qq[u], qq[t]))
        circuit.append(cirq.CNOT(qq[v], qq[t]))

# oracle = compute -> mark -> inverse compute
```

Interpretación

La estructura compute–mark–uncompute es la forma estándar de convertir cálculo clásico reversible en una fase utilizable por Grover.

Qiskit: estructura equivalente

Tipo: Método

Objetivo

En Qiskit se mantiene la misma lógica: los qubits de color controlan CNOTs hacia auxiliares, los comparadores usan controles múltiples y el circuito se deshace antes de la difusión.

```
from qiskit import QuantumCircuit

qc = QuantumCircuit(num_qubits)
qc.cx(u, t)    # t <- t xor x_u
qc.cx(v, t)    # t <- t xor x_v
qc.mcx(controls, target)  # conjuncion reversible
```

Interpretación

La sintaxis cambia respecto a Cirq, pero el modelo computacional es el mismo: operaciones reversibles sobre registros bien definidos.

Qiskit: indicador de arista

Tipo: Método

Función auxiliar

La siguiente rutina computa $x_u \oplus x_v$ en el qubit auxiliar t. Si se ejecuta de nuevo, deshace el cálculo porque CNOT es su propia inversa.

```
def edge_indicator(qc, u, v, t):
    qc.cx(u, t)
    qc.cx(v, t)
```

Interpretación

Esta simplicidad es una ventaja pedagógica del modelo binario. Una arista cortada se detecta con exactamente la operación lógica que CNOT implementa naturalmente.

Qiskit: comparador de patrón

Tipo: Método

Idea

Para marcar un patrón binario, se convierten temporalmente los ceros esperados en unos, se aplica una multi-controlada y se deshacen las compuertas X.

```
def mark_pattern(qc, bits, target, pattern):
    zeros = [q for q, b in zip(bits, pattern) if b == '0']
    for q in zeros:
        qc.x(q)
    qc.mcx(bits, target)
    for q in reversed(zeros):
        qc.x(q)
```

Interpretación

El comparador es reversible porque restaura los bits de entrada. El objetivo cambia solo cuando el patrón coincide.

Qiskit: fragmentos de sumadores

Tipo: Método

Half adder

```
def half_adder(qc, a, b, s, c):
    qc.cx(a, s)
    qc.cx(b, s)
    qc.ccx(a, b, c)
```

Interpretación

El fragmento produce paridad y acarreo sin modificar los sumandos. Encadenar bloques de este tipo permite construir contadores de aristas cortadas.

Limitación

Un oráculo completo para grafos grandes requiere administrar muchos acarreos y auxiliares. La claridad conceptual del bloque no elimina el costo de escalamiento.

Implementación demostrativa con Grover

Tipo: Método

Versión compacta

Para demostraciones pequeñas, puede construirse un oráculo de fase que marca directamente las coloraciones óptimas obtenidas por una rutina clásica auxiliar. Esta versión no reemplaza el oráculo aritmético, pero permite verificar el comportamiento de Grover en simulación.

```
def phase_mark_state(qc, qubits, bitstring):
    for q, b in zip(qubits, bitstring):
        if b == '0': qc.x(q)
    qc.h(qubits[-1])
    qc.mcx(qubits[:-1], qubits[-1])
    qc.h(qubits[-1])
    for q, b in reversed(list(zip(qubits, bitstring))):
        if b == '0': qc.x(q)
```

Interpretación

Esta forma sirve para estudiar amplificación. El diseño aritmético reversible sigue siendo necesario cuando el oráculo debe calcular el criterio desde el grafo.

Lectura de resultados

Tipo: Concepto

Qué significa el histograma

Después de Grover, las cadenas con mayor frecuencia corresponden a coloraciones marcadas por el oráculo. El histograma no muestra directamente el valor del corte; muestra qué coloraciones fueron amplificadas.

Interpretación computacional

Para confirmar el resultado, se toma una cadena medida y se calcula clásicamente $C(x)$. Esta verificación es barata comparada con buscar exhaustivamente todas las cadenas.

Cuidado práctico

El orden de bits impreso por el marco de programación puede diferir de la convención conceptual. Por eso se debe documentar cómo se mapean qubits, bits clásicos y vértices.

Convenciones de orden de bits

Tipo: Concepto

Problema práctico

Una cadena como 00111 solo tiene significado si sabemos qué vértice corresponde a cada posición. Diferentes marcos pueden mostrar los bits en orden inverso al orden de qubits.

Procedimiento

Antes de interpretar resultados, se fija una convención: el bit i representa el vértice i. En Qiskit puede medirse el qubit i en el bit clásico $n-1-i$ si se desea que la cadena impresa respete el orden conceptual.

Implicación

Muchos errores aparentes no son errores del algoritmo, sino errores de lectura de cadenas. La convención debe ser explícita en todo el material.

Ruido y hardware real

Tipo: Concepto

Modelo ideal

La explicación asume compuertas exactas, auxiliares inicializados correctamente y mediciones sin error. En ese modelo, Grover amplifica las coloraciones marcadas según la teoría.

Efecto del ruido

El ruido degrada coherencia, distorsiona fases y puede dejar residuos en auxiliares. Como Grover depende de interferencia precisa, los errores acumulados reducen la probabilidad de éxito.

Implicación

La implementación física exige balancear profundidad, número de auxiliares y fidelidad. Un oráculo aritmético muy profundo puede ser difícil de ejecutar en hardware ruidoso actual.

Error conceptual: conocer la solución de antemano

Tipo: Concepto

Aclaración

El algoritmo requiere un predicado que reconozca soluciones, no una lista de soluciones ya conocidas. En Max-Cut, el predicado se construye calculando el tamaño del corte y comparándolo con un criterio.

Por qué importa

Si las soluciones estuvieran listadas, no habría búsqueda. La contribución algorítmica está en amplificar estados que satisfacen una regla verificable dentro de una superposición.

Implicación

La construcción del oráculo es parte del problema. En optimización cuántica, saber verificar y saber encontrar no son lo mismo.

Error conceptual: no limpiar auxiliares

Tipo: Concepto

Fallo

Si indicadores, acarreos o comparadores quedan con información sobre la coloración, el estado final no es solo "coloración con fase". Es una superposición correlacionada con basura computacional.

Consecuencia

La difusión de Grover presupone que actúa sobre el registro de búsqueda con auxiliares restaurados. Si esa condición falla, la amplificación esperada se degrada o apunta a un espacio incorrecto.

Corrección

El oráculo debe estructurarse como cálculo, fase y descomputación. Esta disciplina es una de las ideas centrales de la computación reversible.

Error conceptual: duplicidad por complemento

Tipo: Concepto

Fallo

Interpretar x y $\bar{x}$ como soluciones distintas puede ser correcto en el espacio de búsqueda, pero representan el mismo corte como partición sin nombres de color.

Por qué importa

Al estimar el número de soluciones marcadas k, esta simetría modifica el número de iteraciones de Grover. También afecta la lectura combinatoria del resultado.

Corrección conceptual

Para análisis de cortes se puede identificar coloraciones complementarias. Para circuitos de Grover, ambas son estados base distintos salvo que se imponga una restricción adicional.

Aplicaciones e interpretación amplia

Tipo: Concepto

Optimización por predicado

La integración Max-Cut–adders–Grover ilustra una plantilla general: convertir una función objetivo discreta en un predicado reversible y usar búsqueda cuántica para amplificar candidatos aceptables.

Relación con problemas de restricciones

Muchos problemas pueden formularse como contar restricciones satisfechas. Las aristas cortadas de Max-Cut son un caso particular de esa lógica.

Implicación

El valor del módulo está en entender la metodología de construcción de oráculos, no solo un grafo específico.

Relación con Grover

Tipo: Resumen

Qué se conserva

El mecanismo de Grover sigue siendo el mismo: superposición inicial, marcado por fase, difusión e iteraciones cercanas al punto óptimo.

Qué cambia

En búsqueda abstracta, el oráculo se daba como caja negra. En Max-Cut, se muestra cómo construirlo a partir de lógica reversible: XORs, sumadores, comparadores y descomputación.

Significado

Este módulo desplaza la atención desde "cómo amplifica Grover" hacia "cómo diseñar el predicado que Grover necesita".

Preguntas conceptuales

Tipo: Concepto

Por qué Max-Cut se codifica en bits

Porque cada coloración es una partición binaria, y cada arista cortada se detecta con una XOR entre los colores de sus extremos.

Por qué la aritmética reversible es imprescindible

Porque el oráculo debe calcular, usar y deshacer información sin borrar registros intermedios de manera irreversible.

Por qué Grover entra al final

Porque Grover no calcula el tamaño del corte; amplifica coloraciones que un oráculo ya puede reconocer como buenas.

Ejercicios propuestos integrados

Tipo: Ejercicio

Variación Max-Cut

Para el ciclo de cinco vértices, proponga una coloración y calcule manualmente cuántas aristas cruzan. Explique por qué no pueden cruzar las cinco aristas simultáneamente.

Variación de sumadores

Determine cuántos bits se necesitan para almacenar la suma de m indicadores de arista. Aplique la regla a $m = 12$ y $m = 20$.

Variación de oráculo

Diseñe el esquema compute–mark–uncompute para marcar coloraciones con $C(x) = T$ en un grafo de cuatro aristas. Identifique qué registros auxiliares son necesarios.

Qué ocurre si el umbral cambia

Tipo: Concepto

Problema que resuelve

Cambiar el umbral T cambia el conjunto de coloraciones marcadas. Un umbral bajo acepta muchas coloraciones; un umbral alto acepta pocas y se aproxima a la búsqueda de óptimos.

Papel dentro del algoritmo

El oráculo no tiene una noción abstracta de calidad. Solo implementa la condición que se le da. Por eso el umbral define qué significa "solución" para Grover.

Implicación computacional

Si se marca un conjunto demasiado grande, la medición puede devolver cortes aceptables pero no máximos. Si se marca un conjunto demasiado pequeño o vacío, la amplificación deja de producir el efecto esperado.

Ejercicio integrado: criterio con umbral

Tipo: Ejercicio

Enunciado

Un grafo tiene 6 aristas y una coloración produce $C(x) = 4$. Analice si debe marcarse bajo los criterios $C(x) = 6$ y $C(x) \geq 4$.

Desarrollo paso a paso

Bajo igualdad con 6, la coloración no se marca porque $4 \neq 6$. Bajo el umbral $C(x) \geq 4$, sí se marca porque satisface la desigualdad.

Interpretación

El mismo conteo puede ser rechazado o aceptado según el predicado. Esta diferencia muestra que el oráculo codifica el problema de búsqueda específico, no una verdad absoluta sobre la coloración.

Variación: múltiples cortes óptimos

Tipo: Ejercicio

Enunciado

Suponga que un grafo tiene cuatro coloraciones óptimas en un espacio de 32 coloraciones. ¿Cómo cambia cualitativamente el número de iteraciones respecto al caso de dos coloraciones óptimas?

Desarrollo

La regla de Grover depende de $\sqrt{N/k}$. Al pasar de $k = 2$ a $k = 4$, el cociente N/k se reduce a la mitad. Por tanto, el número de iteraciones óptimas disminuye.

Interpretación

Tener más soluciones no dificulta la búsqueda de Grover; en el modelo ideal la facilita porque hay más amplitud total asociada al subespacio marcado.

Diseño de oráculos: decisión académica

Tipo: Concepto

Dos estrategias

Para grafos pequeños puede marcarse directamente una lista de coloraciones óptimas, lo cual es útil para simulación. Para un algoritmo genuino, el oráculo debe calcular el criterio desde el grafo mediante circuitos reversibles.

Por qué importa la distinción

La primera estrategia verifica la dinámica de Grover; la segunda representa el problema computacional real. Confundirlas lleva a pensar que el algoritmo recibe la respuesta en la entrada.

Implicación

El estándar académico es explicar ambas: una como herramienta didáctica de simulación, otra como arquitectura algorítmica completa.

Síntesis conceptual

Tipo: Resumen

Max-Cut

Aporta el problema combinatorio: buscar una partición que maximice aristas cruzadas. Su estructura local se expresa mediante XORs por arista.

Adders

Aportan el mecanismo reversible para convertir muchos bits locales en un conteo global. Half adders, full adders y ripple carry adders hacen posible computar la función objetivo dentro de un circuito cuántico.

Grover

Aporta la amplificación. Una vez que el oráculo marca las coloraciones aceptadas, la difusión aumenta su probabilidad de medición.

Cierre.

Tipo: Resumen

Idea central

El módulo muestra cómo pasar de una instancia de optimización en grafos a un algoritmo cuántico de búsqueda. La transformación no es automática: requiere codificación binaria, cálculo reversible, comparación y marcado de fase.

Resultado académico

El estudiante debe poder explicar por qué los adders son tan importantes como Max-Cut: sin ellos no hay conteo reversible y, por tanto, no hay oráculo útil para Grover.

Proyección

La misma arquitectura aparece en otras tareas de optimización: representar candidatos, computar una función objetivo, marcar estados aceptables y amplificarlos mediante interferencia controlada.