Università di Genova
DIBRIS DIPARTIMENTO
DI INFORMATICA, BIOINGEGNERIA,
ROBOTICA E INGEGNERIA DEI SISTEMI

Algoritmo di Grover

Un approccio quantistico alla ricerca in un database

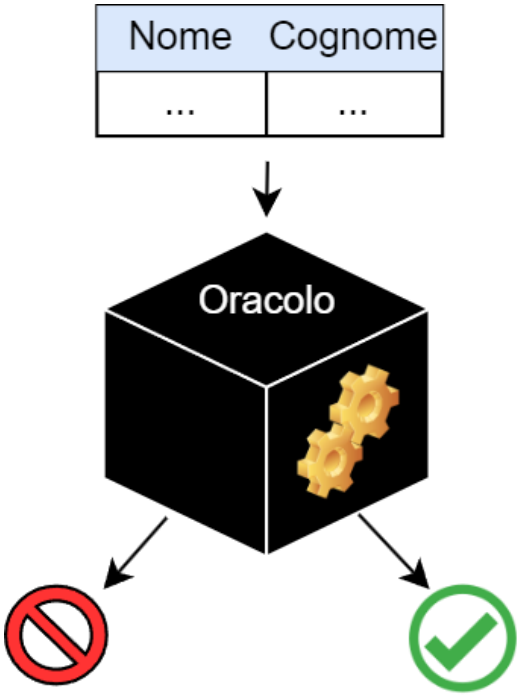
Candidato: Ninivaggi Luca
Relatore: Prof. Solinas Paolo

Ricerca in un Database non strutturato

Database Non Strutturato

Descrive un insieme di tuple in cui l'ordinamento è irrilevante, ovvero non fornisce indizi sul contenuto delle stesse.

Nome	Cognome
Angelo	Bianchi
Filippo	Verdi
Elisa	Marrone
…	…
Mario	Rossi
…	…
Giacomo	Neri

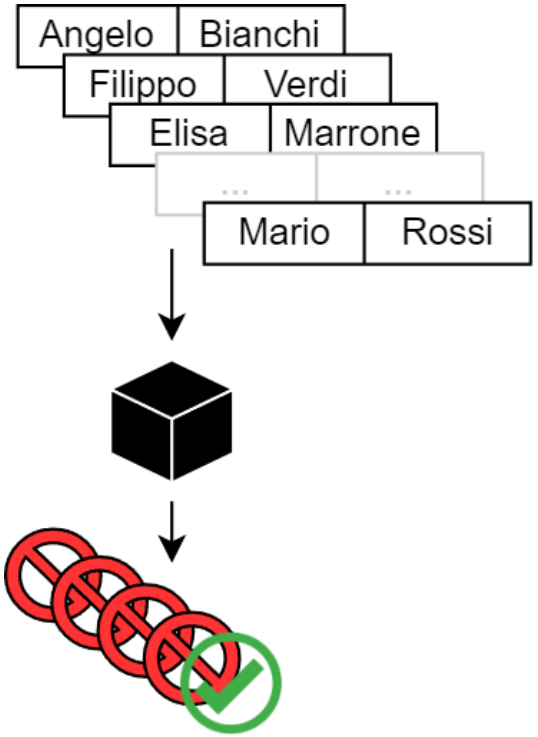


Oracolo

Risolve un **problema decisionale**:
dando in input una riga, verifica se soddisfa una condizione.
Formalmente, descrive una query sul DB.

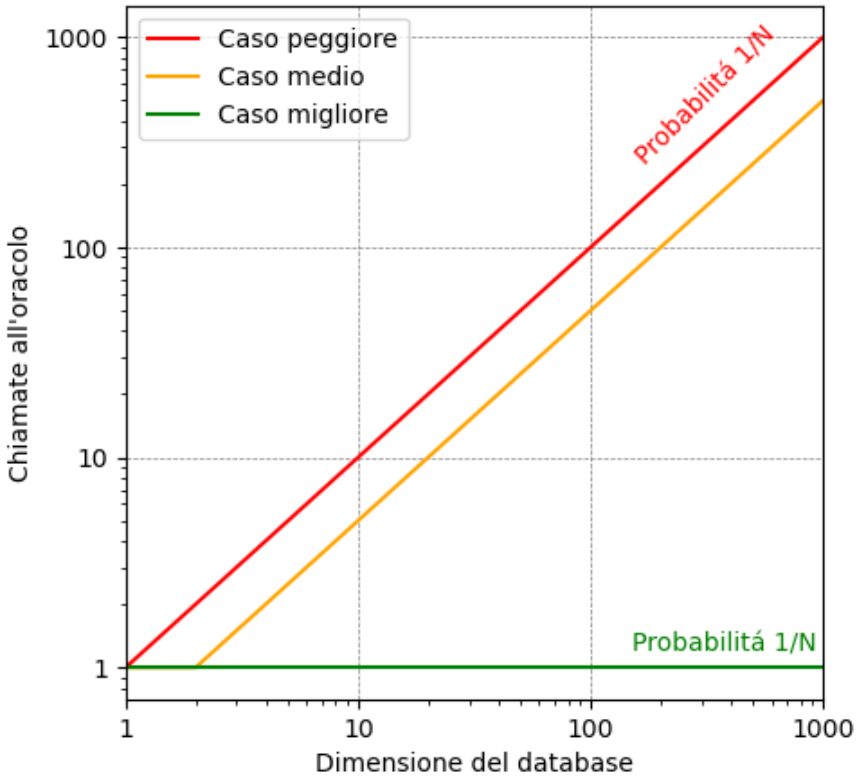
Es: "il nome è Mario?"

Soluzione Classica



Algoritmo

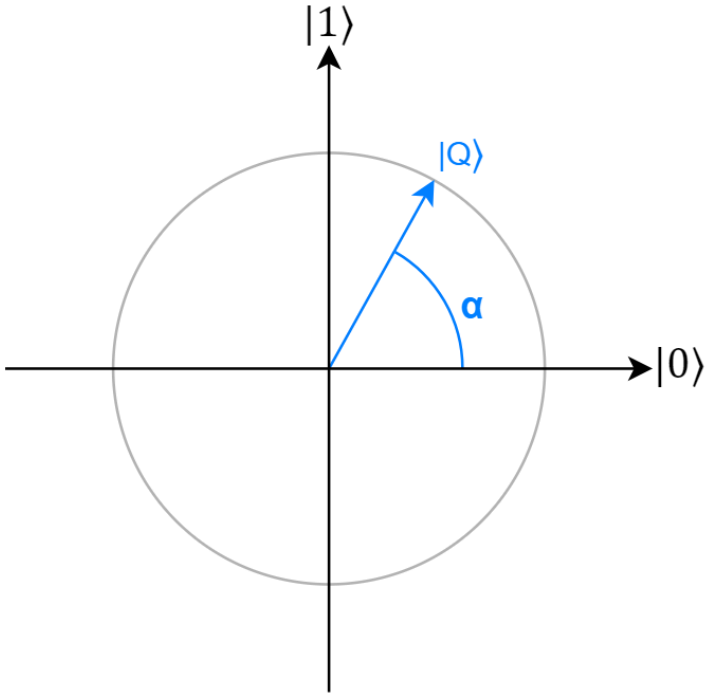
1. Estrai casualmente una tupla.
2. Usa l'oracolo e verificane la validità.
3. Se non è valida, torna al passo 1.



Complessità temporale

- **Caso peggiore:** N
- **Caso medio:** $\frac{N}{2}$
- **Caso migliore:** 1

Qubit & sovrapposizione di stati Quantistici



Come descriviamo un Quantum bit?

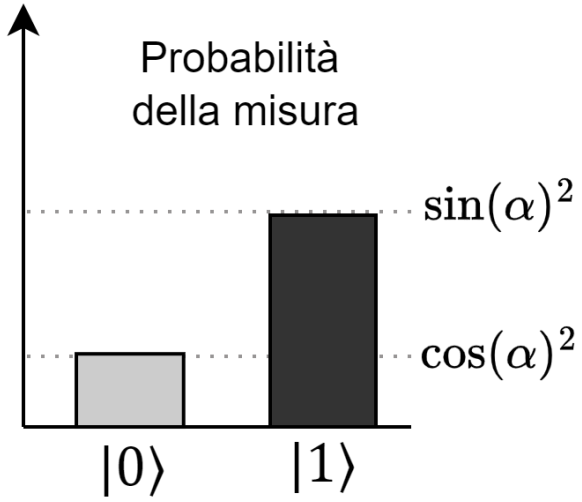
Rispetto a un bit classico (**deterministico** e con valore 0 **oppure** 1), un qubit possiede le proprietà della **sovrapposizione quantistica**.

Ovvero lo possiamo descrivere come una sovrapposizione di stati $|0\rangle$ e $|1\rangle$:

$$|\mathrm{Q}\rangle = \sin(\alpha) \cdot |1\rangle + \cos(\alpha) \cdot |0\rangle$$

Come si misura un Qubit?

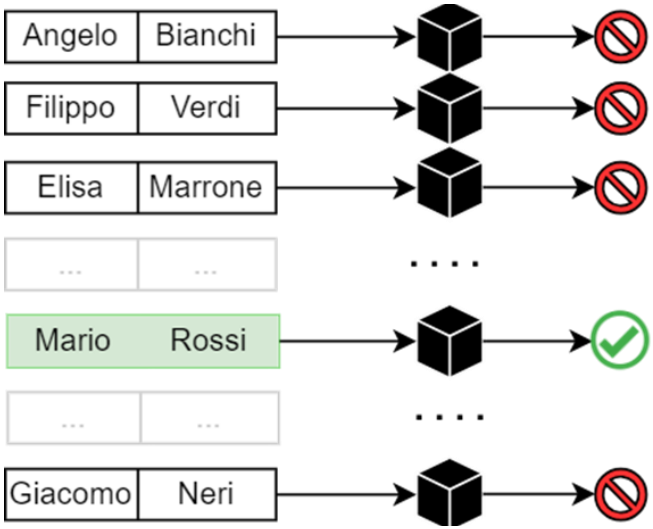
"**Misurare**" un Qubit, cioè "estrarne" il valore 0 o 1, è un processo in cui la **probabilità** di ottenere ciascun stato è proporzionale al quadrato della lunghezza della sua componente nel vettore di sovrapposizione.



Il vantaggio della sovrapposizione

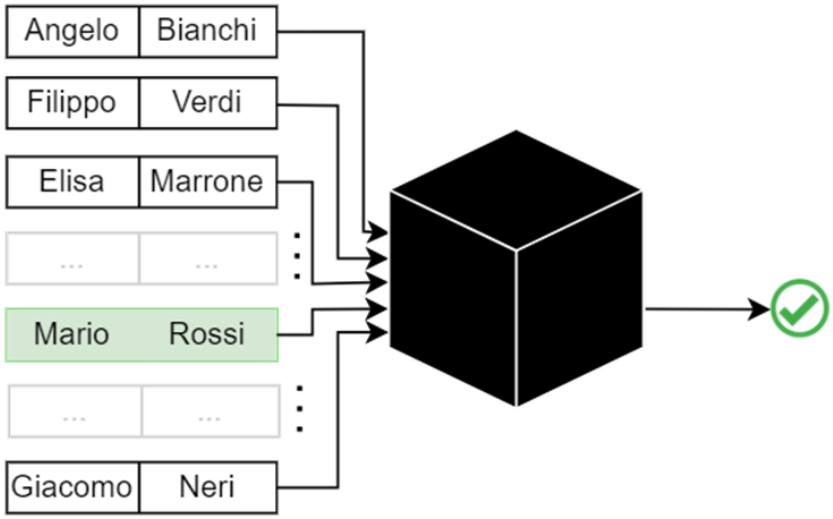
Quanto è costoso studiare tutte le tuple?

Oracolo Classico



In algoritmi classici utilizziamo **N** volte l'oracolo, dando in input ogni tupla.

Oracolo quantistico



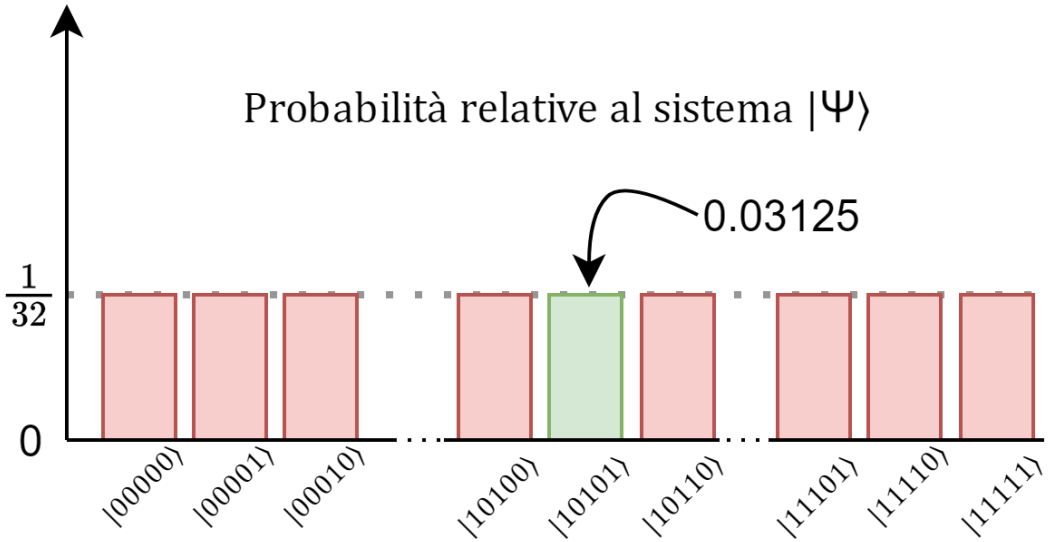
Nella controparte quantistica utilizziamo **1** chiamata all' oracolo (utilizza in input il DB) per '**marcare**' tutte le tuple accettate.

Preparazione dell' Algoritmo

Mappiamo in modo univoco ognuna delle **N** tuple con uno stato quantistico, utilizzando $n = log_2(N)$ qubit.

Nel nostro caso, utilizziamo **5 qubit** per mappare 32 tuple.

Stato	Nome	Cognome
|00000⟩	Angelo	Bianchi
|00001⟩	Filippo	Verdi
|00010⟩	Elisa	Marrone
…	…	…
|10101⟩	Mario	Rossi
…	…	…
|11111⟩	Giacomo	Neri



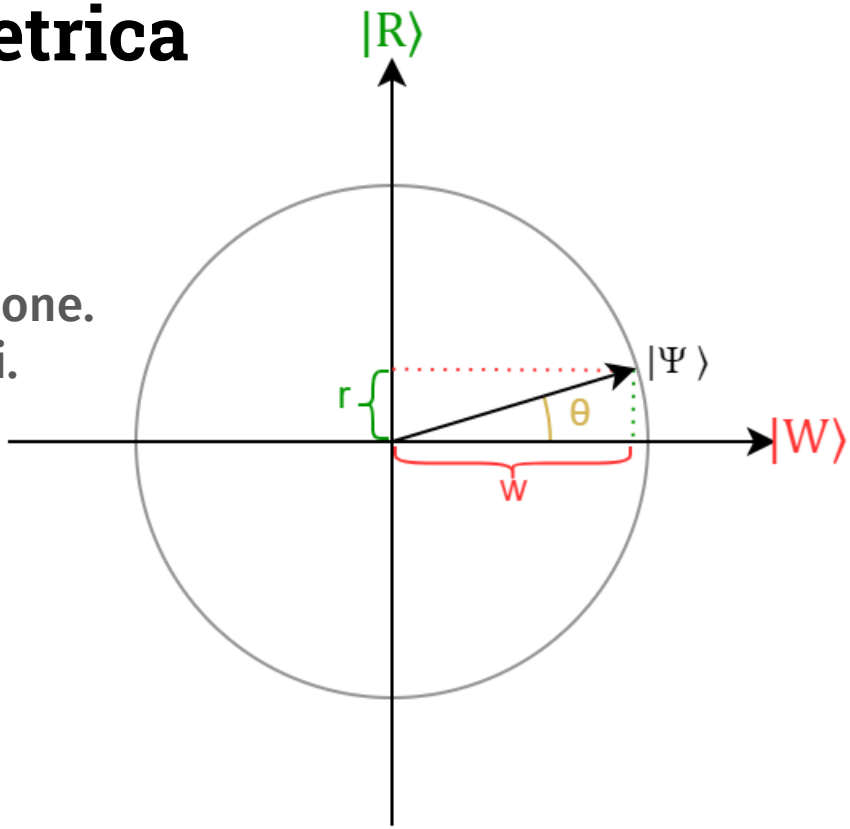
Inizializziamo il sistema con una **perfetta sovrapposizione** degli stati, denominata |Ψ⟩.

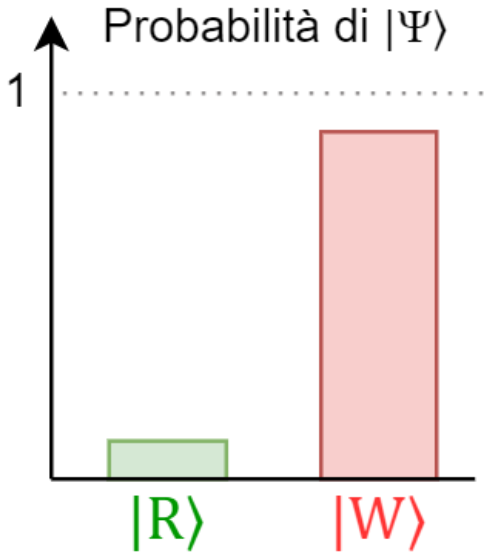
Un osservazione geometrica

Consideriamo il nostro stato come sovrapposizione di :

- $|R\rangle$, somma diretta degli stati soluzione.
- $|W\rangle$, somma diretta degli stati errati.

$$|\Psi\rangle = \sin(\theta) \cdot |R\rangle + \cos(\theta) \cdot |W\rangle$$





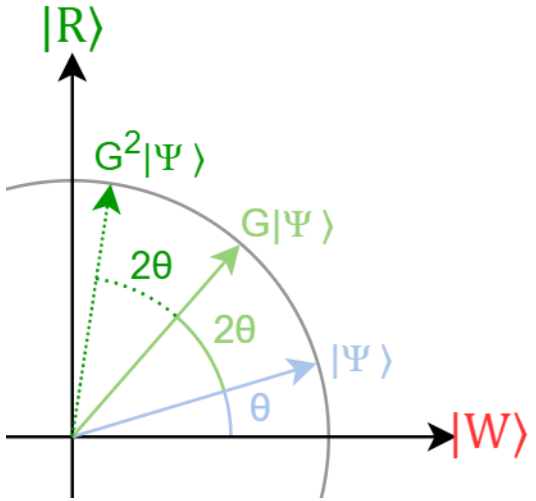
Similmente alla misurazione di un singolo qubit, misurare l'intero sistema significa ottenere uno stato che può essere componente di $|R\rangle$ o $|W\rangle$.

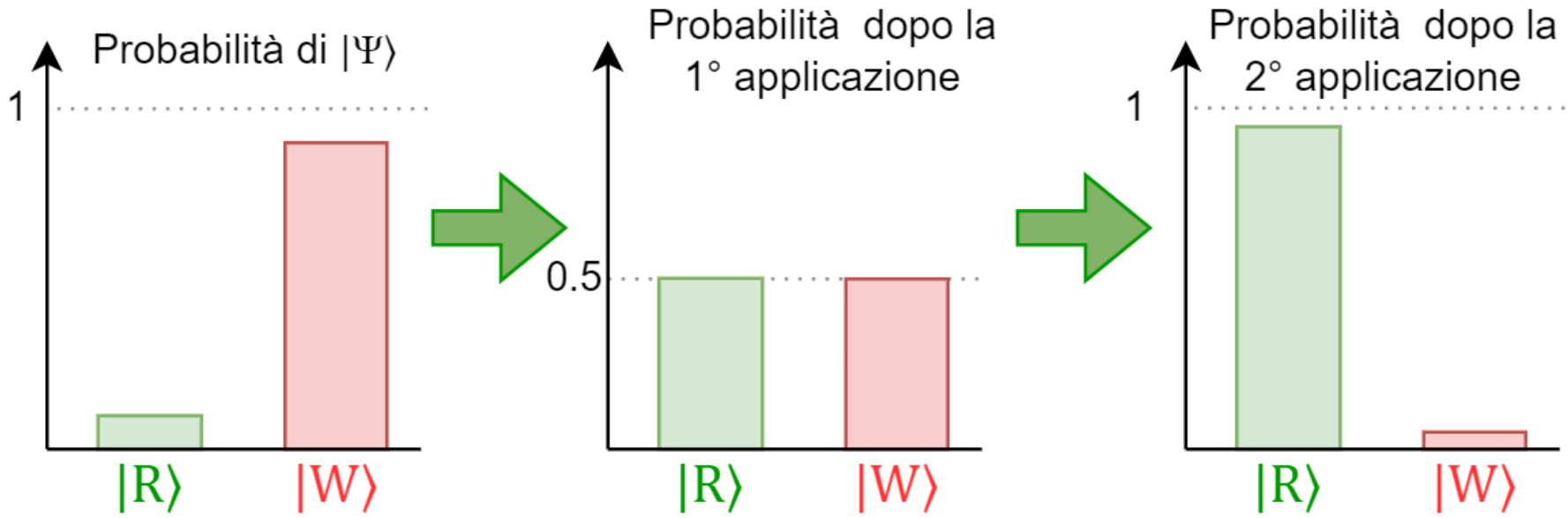
Ottenere $|R\rangle$ dalla misurazione indica che abbiamo trovato uno **stato soluzione**, mentre $|W\rangle$ ha come componenti esclusivamente **stati non-soluzione**.

Applicare l' operatore G

L' algoritmo utilizza un "**operatore di Grover**", il quale modifica i coefficienti del sistema.

Osserviamo come a ogni applicazione dell' operatore, il sistema aumenti progressivamente le probabilità di misurare uno stato soluzione.



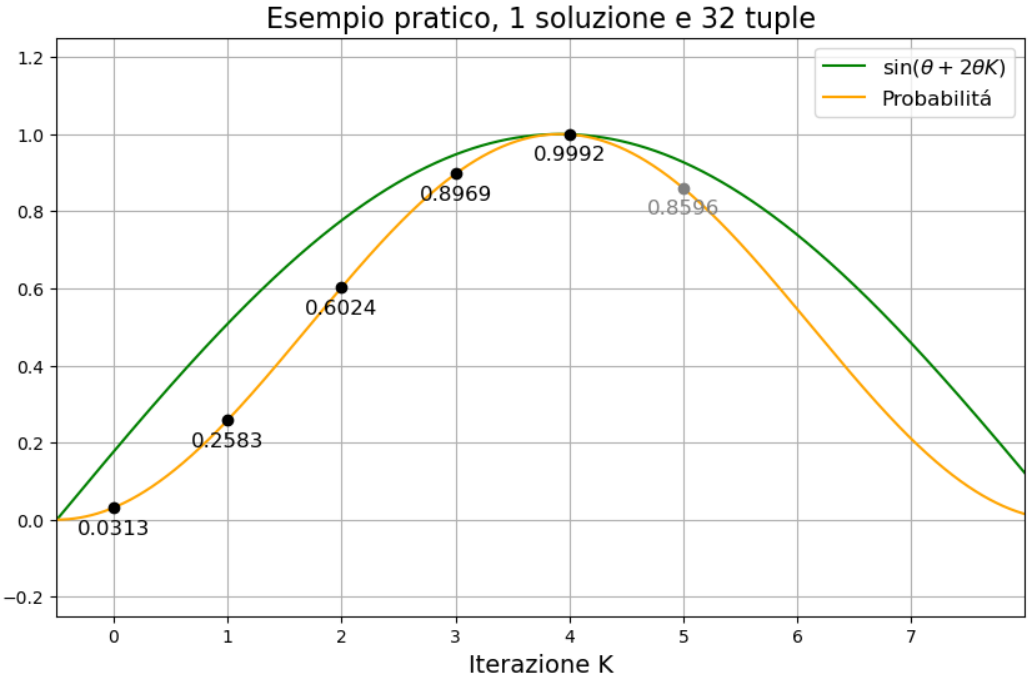


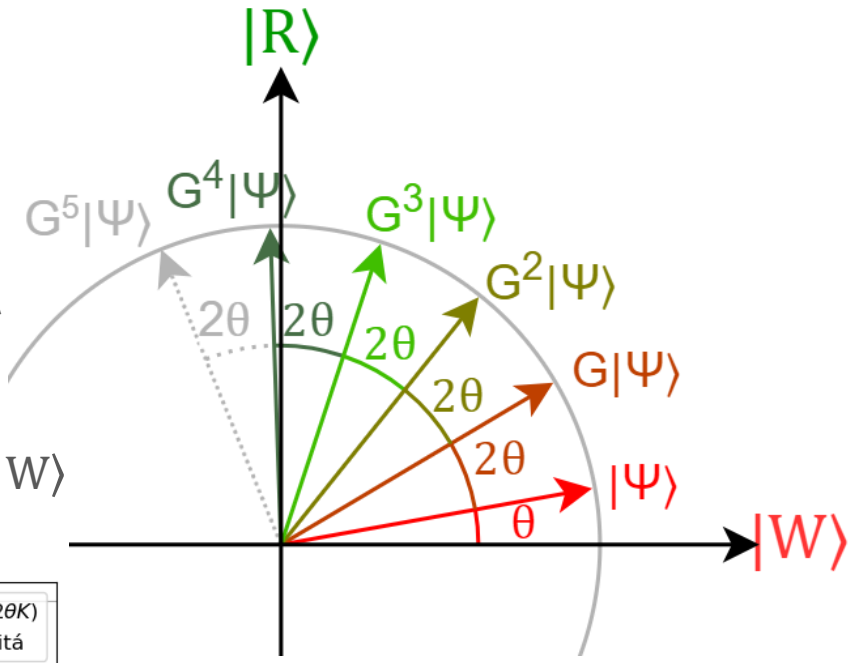
Quando fermarsi ?

Si noti come il sistema "ruoti" di 2θ a ogni applicazione dell' operatore G.

Possiamo quindi descriverlo in funzione del numero di iterazioni eseguite K:

$$G^K|\Psi\rangle = \sin(\theta + 2\theta K) \cdot |\mathrm{R}\rangle + \cos(\theta + 2\theta K) \cdot |\mathrm{W}\rangle$$

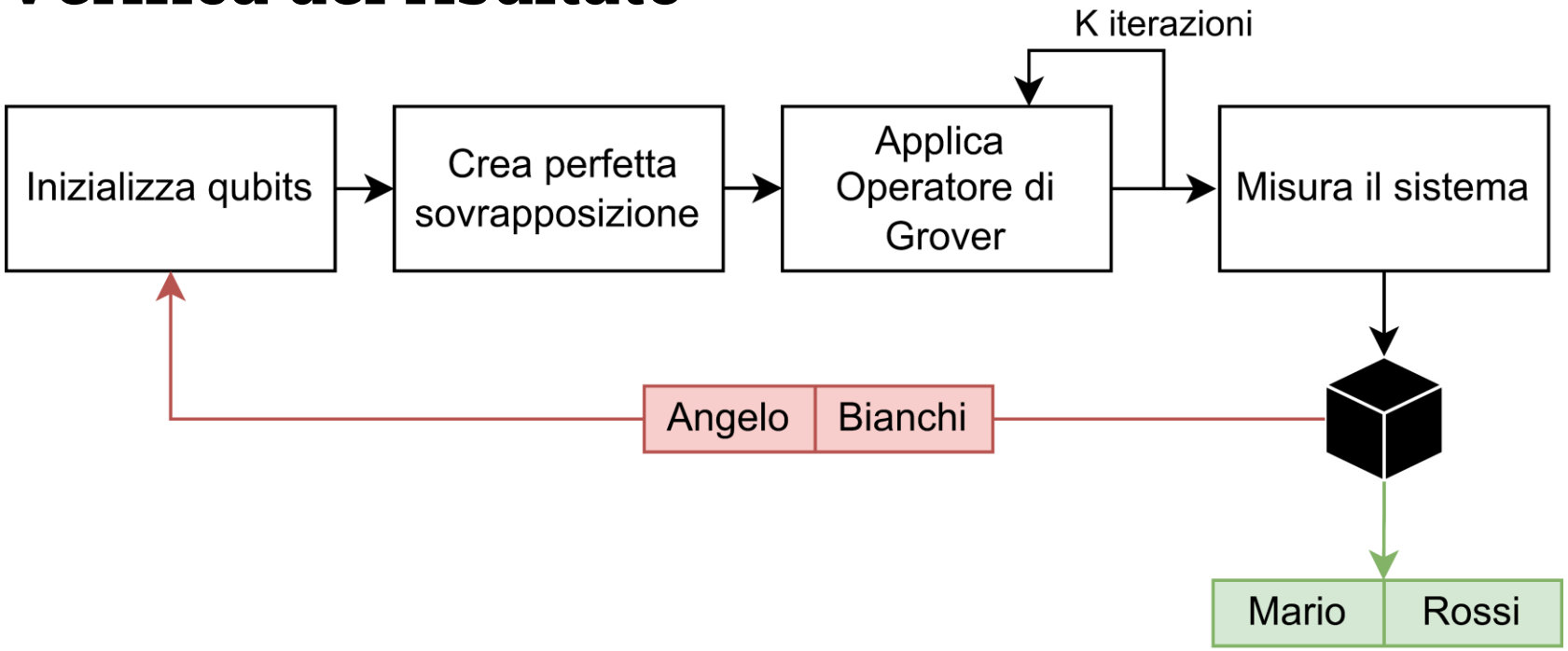




Il valore ottimale di K, che massimizza il seno e quindi la probabilità di trovare la soluzione, si può ricavare:

$$K = \frac{\pi}{4}\sqrt{N} - \frac{1}{2}$$

Verifica del risultato



Dopo k iterazioni, nel nostro esempio l'algoritmo di Grover restituisce uno stato soluzione nel 99.9% dei casi.

Indipendentemente dal DB di partenza, esiste però una piccola possibilità di errore, quindi la tupla ottenuta va verificata con un oracolo classico : se necessario, l'algoritmo va ripetuto.

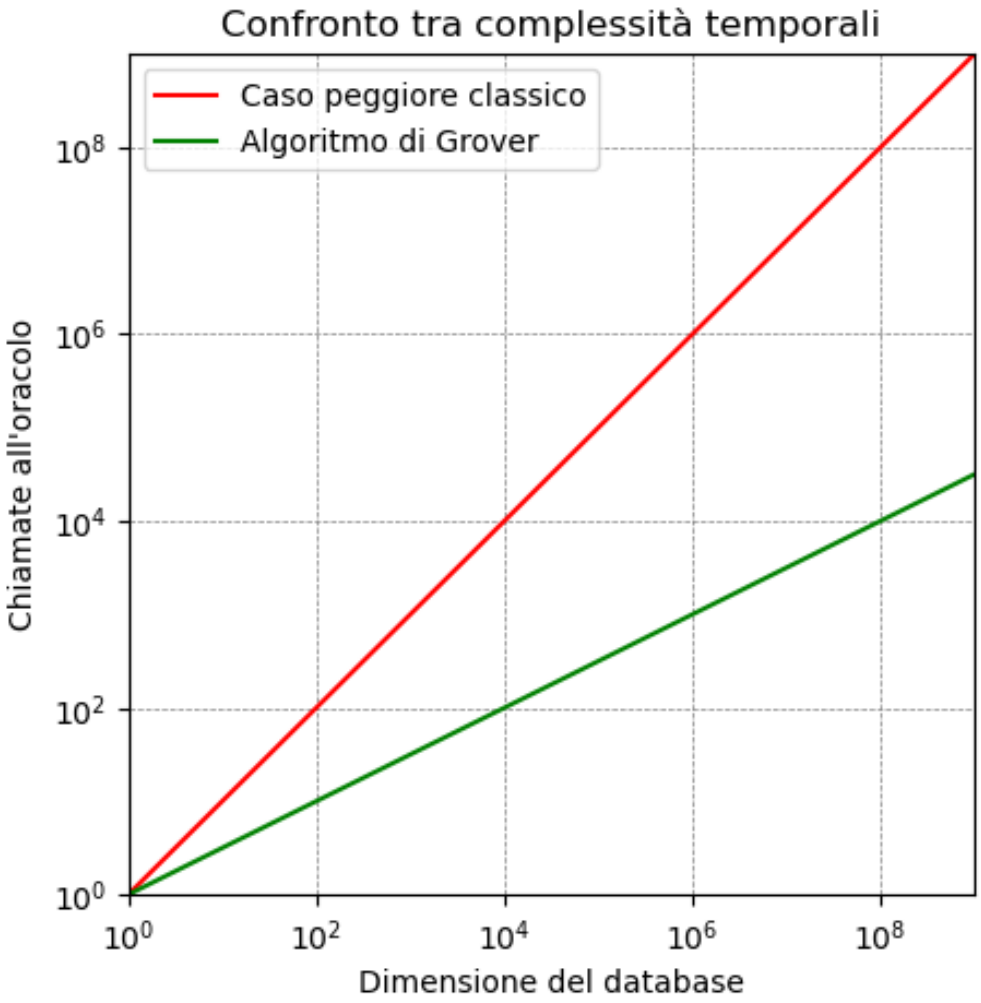
Implementazione – Python & Qiskit

```python
#La tupla di cui voglio mostrare le probabilità
marked_state = 21 #|10101>

####### INIZIALIZZAZIONE #######
#Inizializzo i qubit tutti in |0⟩
state = Statevector.from_label('00000')
#rendo tutti gli stati equiprobabili
state = state.evolve(hadamards)

#stampo la probabilità iniziale
print_probability(state,marked_state)

####### ITERAZIONI #######
k = round((np.pi / 4) * np.sqrt(N) - 1/2)
for i in range(k):  #applico l' Operatore di Grover
    #marco gli stati soluzione
    state = state.evolve(mark)
    #amplifico la loro probabilità
    state = state.evolve(diffuse)

    #stampo la probabilità dello stato
    print_probability(state,marked_state,i)

#mostro la tupla con maggiore probabilitá
print_best_row(state)
```

Stato	Nome	Cognome
|00000⟩	Angelo	Bianchi
|00001⟩	Filippo	Verdi
|00010⟩	Elisa	Marrone
…	…	…
|10101⟩	Mario	Rossi
…	…	…
|11111⟩	Giacomo	Neri

```
Dopo inizializzazione:
        P(|10101⟩) = 0.0312
Dopo 1 applicazioni di G:
        P(|10101⟩) = 0.2583
Dopo 2 applicazioni di G:
        P(|10101⟩) = 0.6024
Dopo 3 applicazioni di G:
        P(|10101⟩) = 0.8969
Dopo 4 applicazioni di G:
        P(|10101⟩) = 0.9992

Tupla |10101⟩ ottenuta con P = 0.9992:
        {'Nome':'Mario','Cognome':'Rossi'}
Oracolo utilizzato 4 volte
```

Complessità e Speed-up



- La soluzione classica ha complessità media $O(N)$.
- L'algoritmo di Grover è di complessità paragonabile a $O(\sqrt{N})$.

Otteniamo quindi uno **speed-up** quadratico rispetto all' algoritmo classico.

Applicazioni pratiche - SAT

Cosa è il SAT?

Per SAT intendiamo il problema (NP-Completo) di determinare se una formula booleana sia **soddisfacibile**.

Es. " $(A\ or\ B)\ and\ not(C\ and\ A)$ " **è soddisfatta da** $(\mathrm{A}, \mathrm{B}, C) = (1,0,0)$

In che modo si risolve con l' algoritmo di Grover?

Utilizzando un oracolo che ne descrive la formula, ovvero che marca tutte le **configurazioni** (se esistono) che la soddisfano.

Nella pratica, che vantaggi produce?

Supponiamo una formula booleana con **70 variabili** e un frequenza di chiamate all' oracolo di **1GHz** (1 miliardo di volte al secondo).

- Algoritmo classico (Brute-force con 2^{70} chiamate):

  ~12,500 anni

- Algoritmo di Grover (2^{35} chiamate):

  ~12 secondi

Università di Genova
DIBRIS DIPARTIMENTO
DI INFORMATICA, BIOINGEGNERIA,
ROBOTICA E INGEGNERIA DEI SISTEMI

Grazie per l'attenzione