# Quantum walk search: from grids to networks

## Thesis defense

Mathieu Roget

Supervised by Giuseppe Di Molfetta

Laboratoire d'informatique et des systèmes (LIS)
Aix-Marseille Université (AMU)

July 11, 2025

# Overview

# Discrete time quantum walks

# Table of Contents

# Memory and state

**Classical memory :**

| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|
| | | | | ★ | | | |

# Memory and state

**Classical memory :**

| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

# Memory and state

**Classical memory :**

| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

**Quantum memory :**

| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|
| $z_0$ | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ | $z_7$ |

# Memory and state

**Classical memory :**

| **000** | **001** | **010** | **011** | **100** | **101** | **110** | **111** |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

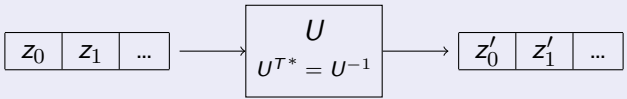
**Quantum memory :**

| $\|000\rangle$ | $\|001\rangle$ | $\|010\rangle$ | $\|011\rangle$ | $\|100\rangle$ | $\|101\rangle$ | $\|110\rangle$ | $\|111\rangle$ |
|---|---|---|---|---|---|---|---|
| $z_0$ | $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ | $z_7$ |

$|\bullet\rangle$: Vector of the canonical basis.

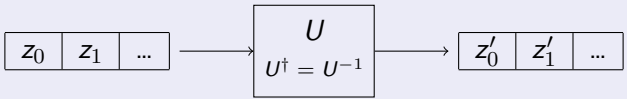
# Operations

## Unitary operation



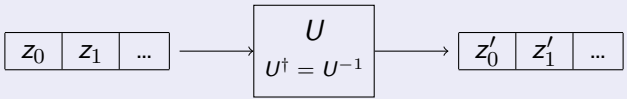
# Operations

## Unitary operation

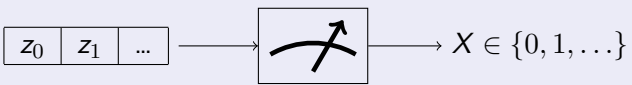


# Operations

## Unitary operation

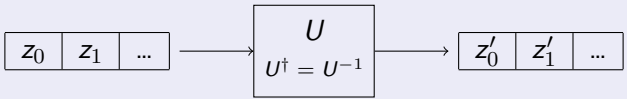



## Measure

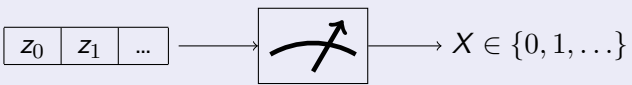


# Operations

## Unitary operation




## Measure




$\mathbb{P}(X = k) = |z_k|^2$.

# Quantum computing : rules

We assume the memory has $n$ qubits

# Quantum computing : rules

We assume the memory has $n$ qubits

Rule 1 : A state $z$ is a complex vector of norm $1$ :

$$z = \begin{array}{|c|c|c|c|c|c|c|c|} \hline z_0 & z_1 & z_2 & z_3 & z_4 & z_5 & \dots & z_{2^n-1} \\ \hline \end{array} \in \mathbb{C}^{2^n}$$

# Quantum computing : rules

We assume the memory has $n$ qubits

Rule 1 : A state $z$ is a complex vector of norm $1$ :

$$z = \boxed{z_0 \mid z_1 \mid z_2 \mid z_3 \mid z_4 \mid z_5 \mid \dots \mid z_{2^n-1}} \in \mathbb{C}^{2^n}$$

Rule 2 : An valid operation is a linear operator $U$ such that $U^\dagger = U^{-1}$ (noted unitary operator).

# Quantum computing : rules

We assume the memory has $n$ qubits

Rule 1 : A state $z$ is a complex vector of norm $1$ :

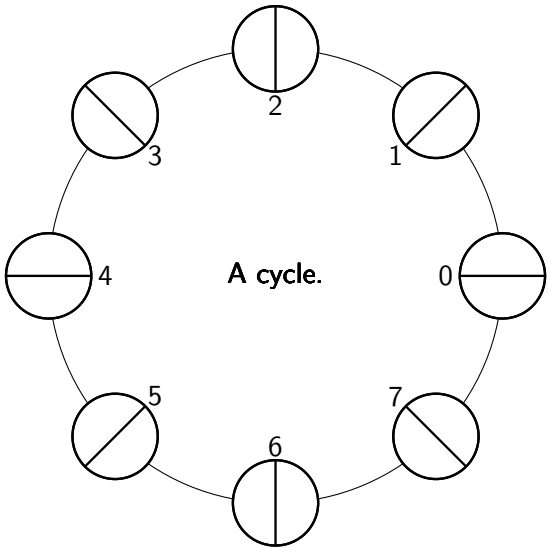
$$z = \boxed{z_0 \mid z_1 \mid z_2 \mid z_3 \mid z_4 \mid z_5 \mid \dots \mid z_{2^n-1}} \in \mathbb{C}^{2^n}$$

Rule 2 : An valid operation is a linear operator $U$ such that $U^\dagger = U^{-1}$ (noted unitary operator).

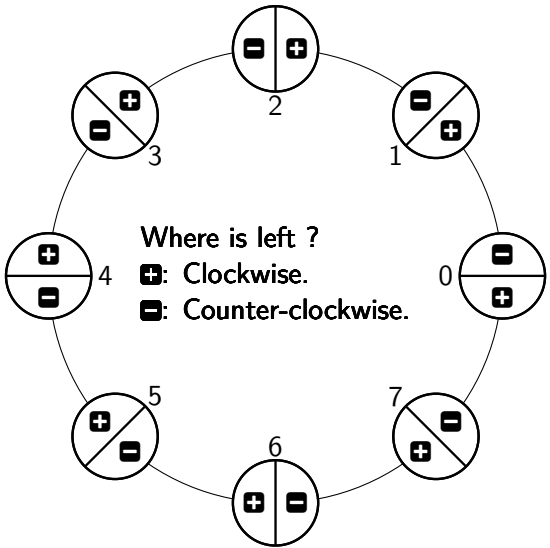
Rule 3 : At any time, you can transform a quantum state into a classical one by measuring. The classical state $X$ of the quantum state $z$ after measuring is random. $\mathbb{P}(X = i) = |z_i|^2$.

# Table of Contents

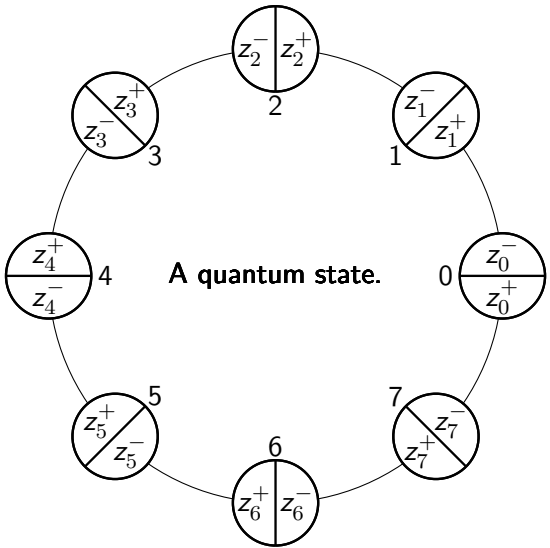
# Example on the cycle with real numbers



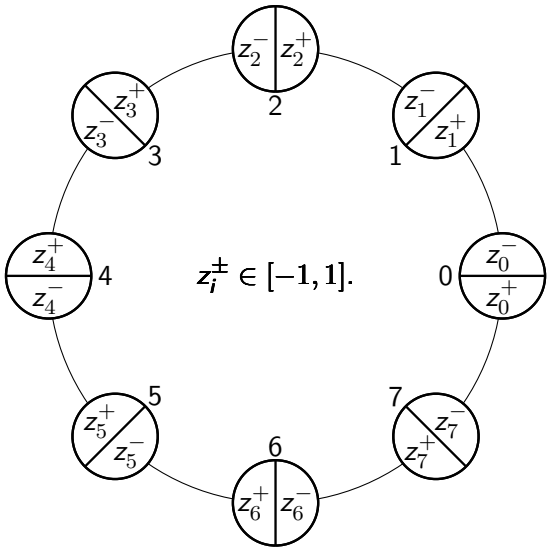
# Example on the cycle with real numbers



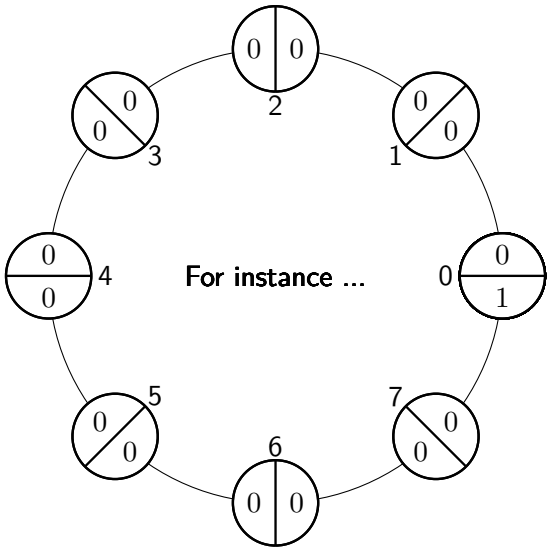
# Example on the cycle with real numbers



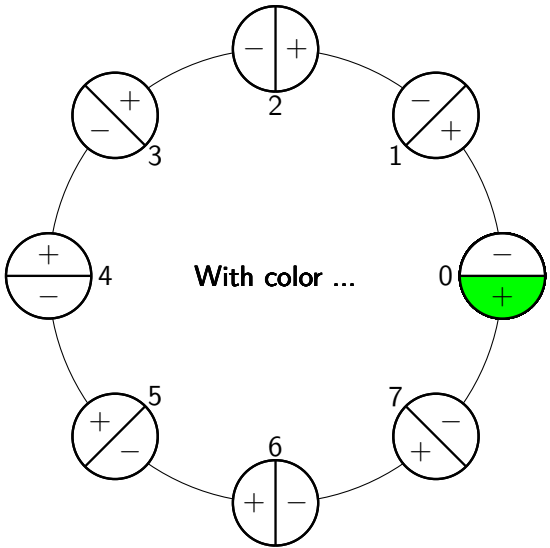
# Example on the cycle with real numbers

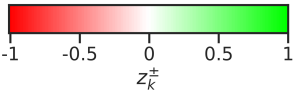


# Example on the cycle with real numbers

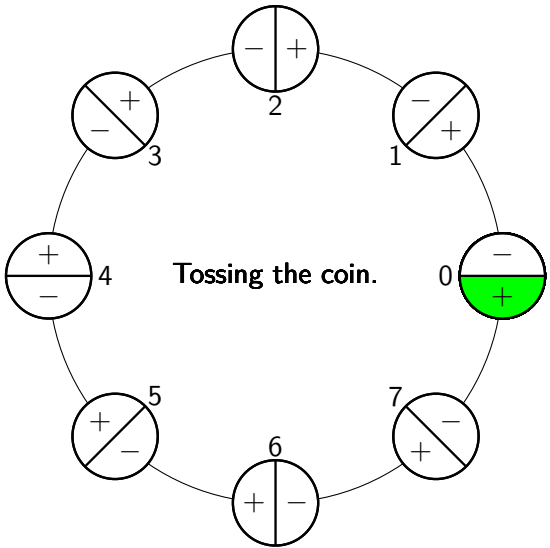


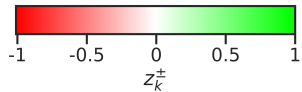
# Example on the cycle with real numbers



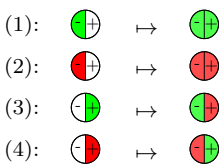



# Example on the cycle with real numbers

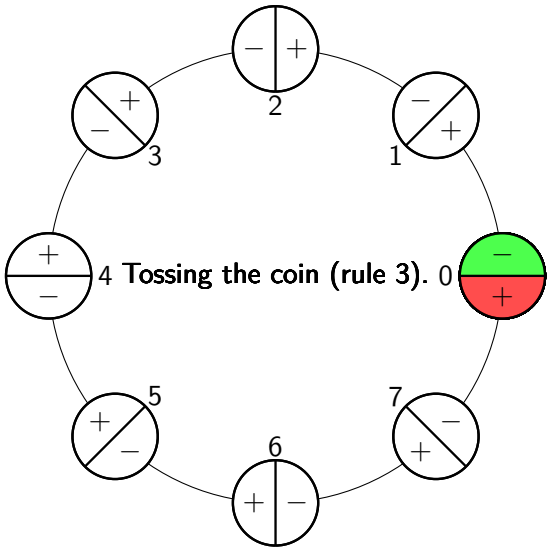


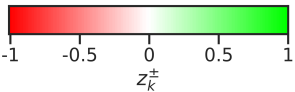




**Hadamard coin toss:**

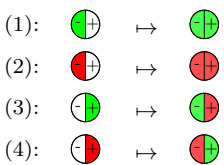


# Example on the cycle with real numbers

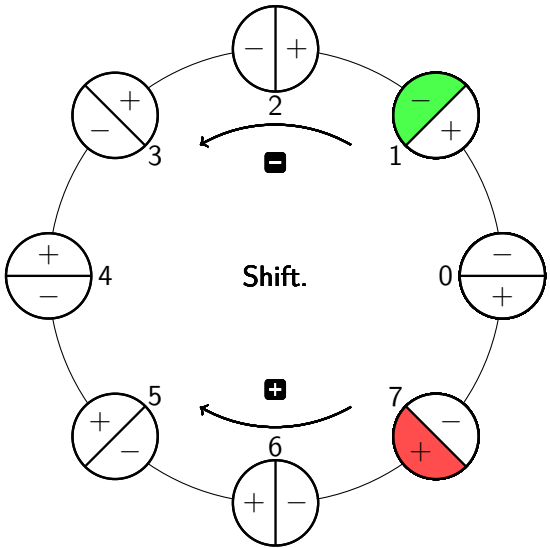


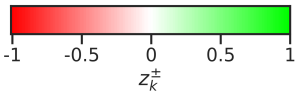




**Hadamard coin toss:**

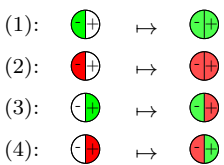


# Example on the cycle with real numbers

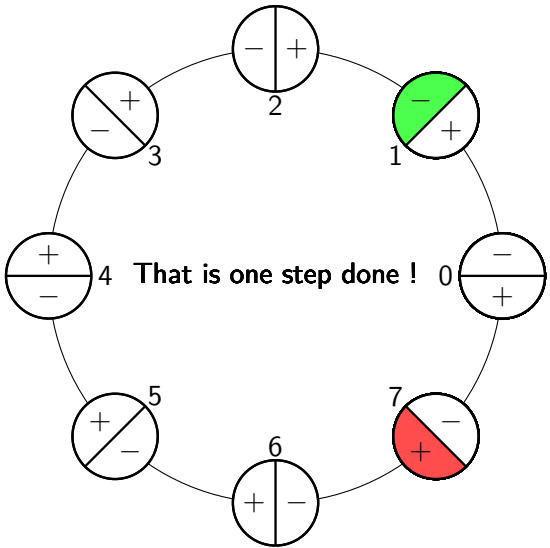


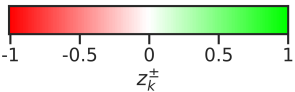




**Hadamard coin toss:**

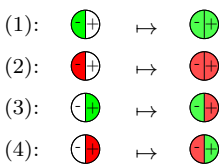


# Example on the cycle with real numbers

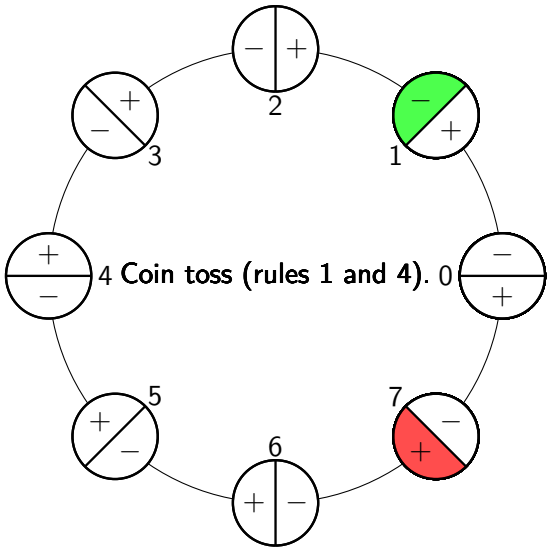


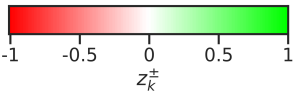




**Hadamard coin toss:**

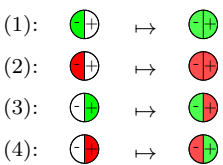


# Example on the cycle with real numbers

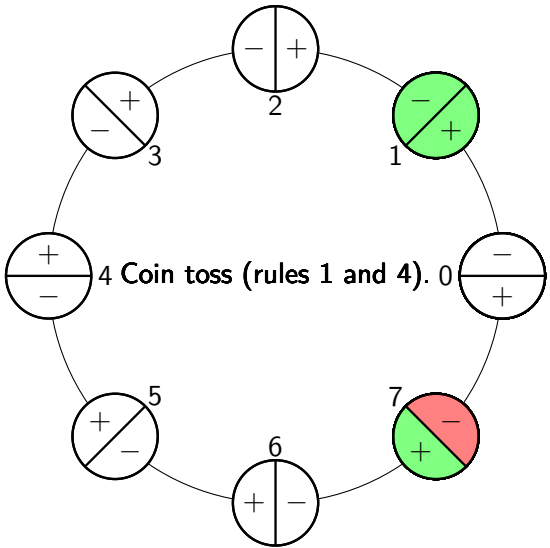


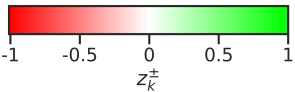




**Hadamard coin toss:**

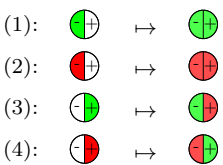


# Example on the cycle with real numbers

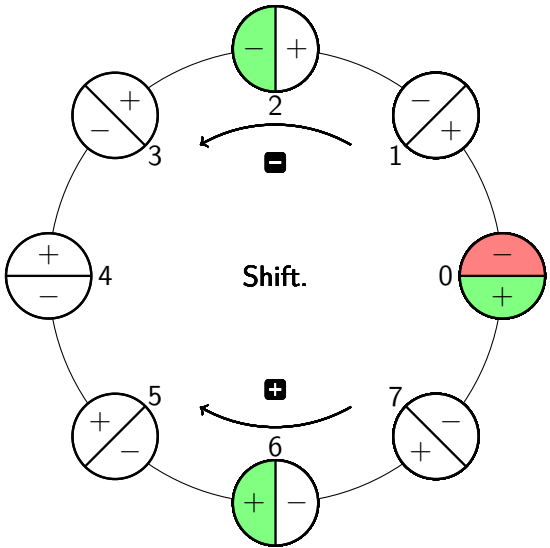


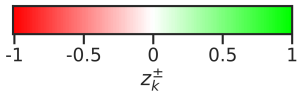




**Hadamard coin toss:**

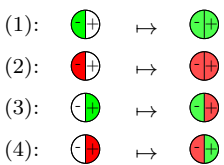


# Example on the cycle with real numbers

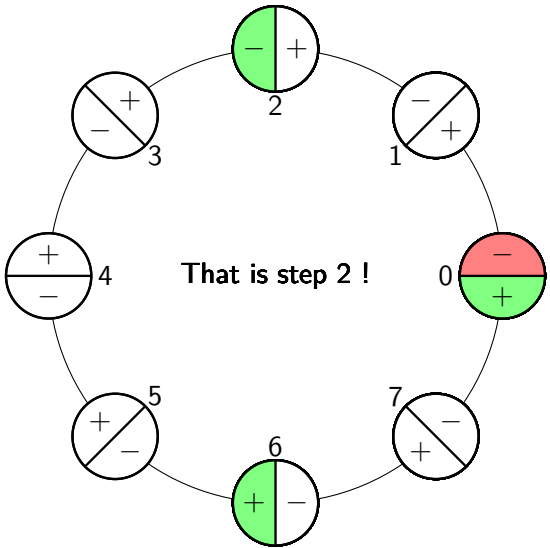


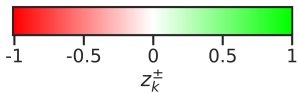




**Hadamard coin toss:**

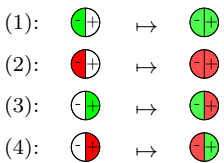


# Example on the cycle with real numbers

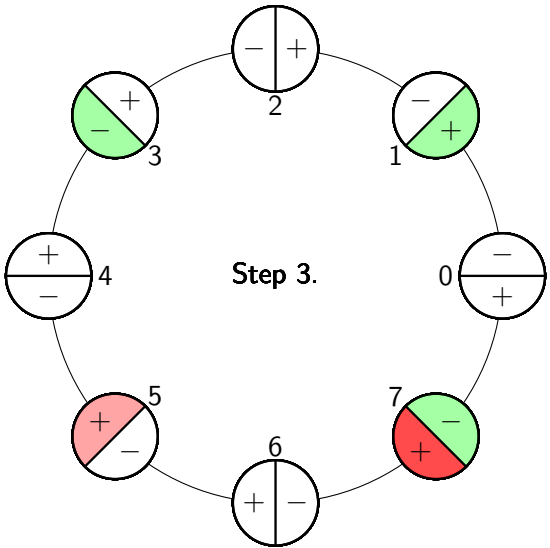


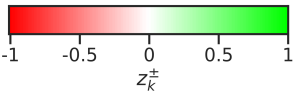




**Hadamard coin toss:**



# Example on the cycle with real numbers




**Hadamard coin toss:**

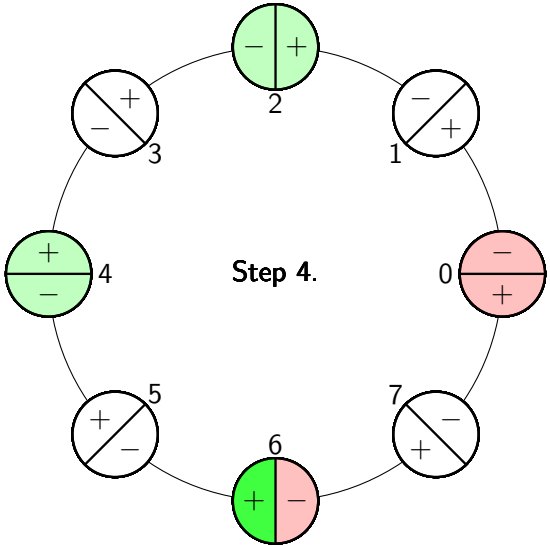
$$(1): \quad \mapsto$$

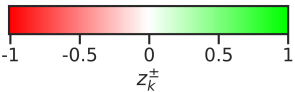
$$(2): \quad \mapsto$$

$$(3): \quad \mapsto$$

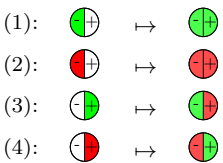
$$(4): \quad \mapsto$$

# Example on the cycle with real numbers

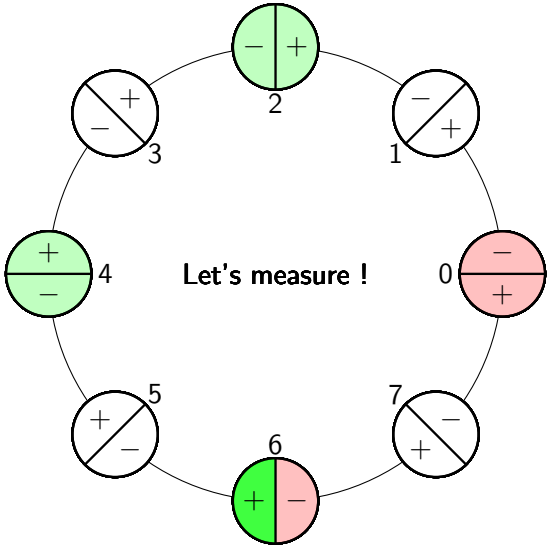


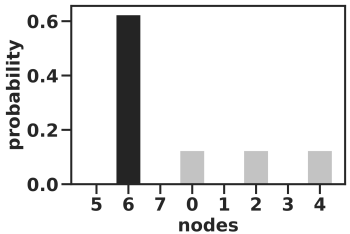

**Hadamard coin toss:**



# Example on the cycle with real numbers




$$p_t(i) = |z_i^+(t)|^2 + |z_i^-(t)|^2$$



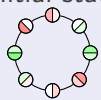
# Outline

## Scheme

# Outline

## Scheme

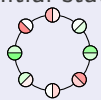
Initial state

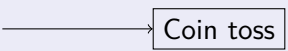


# Outline

## Scheme

Initial state

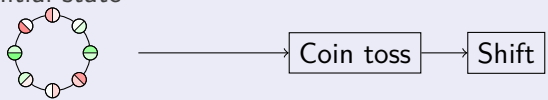




# Outline

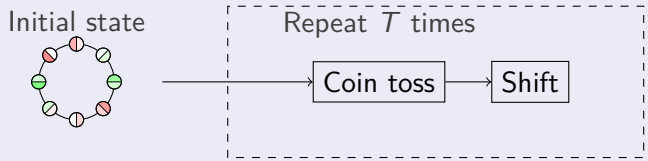
## Scheme

Initial state



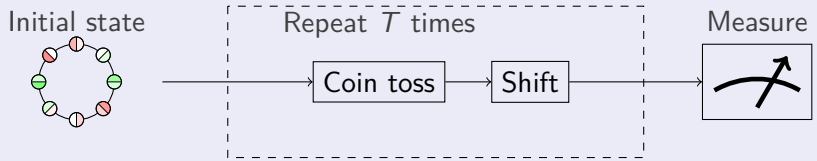
# Outline

## Scheme



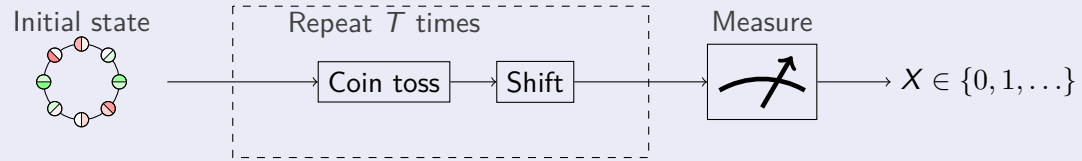
# Outline

## Scheme



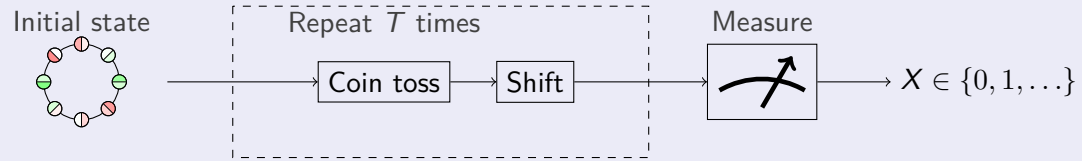
# Outline

## Scheme



# Outline

## Scheme

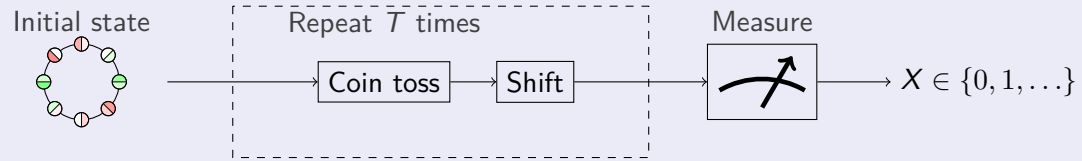



## Outcome

The outcome is the position $X$ measured after $T$ steps.

# Outline

## Scheme



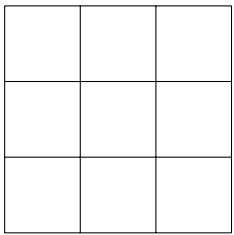

## Outcome

The outcome is the position $X$ measured after $T$ steps.

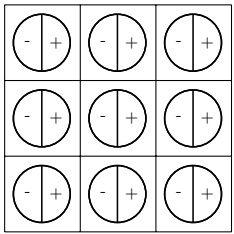
## Remark

In practice, we can only measure the position once. However, for the sake of studying this model, we consider the probability $p_t(i)$ of measuring position $i$ at step $t$.

# DTQW on a grid

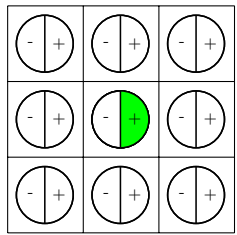


# DTQW on a grid



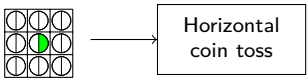
# DTQW on a grid

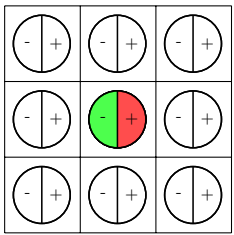
Initial state



# DTQW on a grid

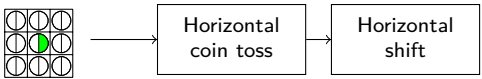
Initial state

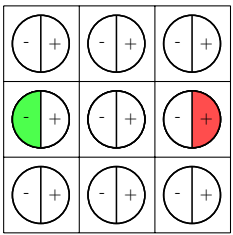


# DTQW on a grid

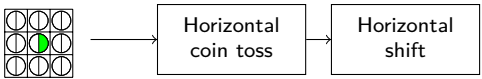
Initial state

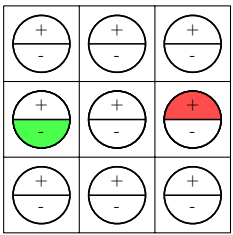


# DTQW on a grid

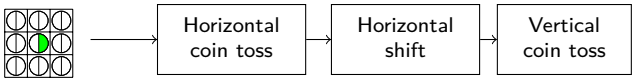
Initial state

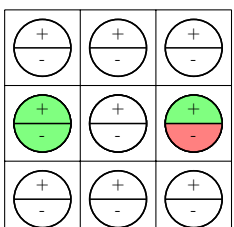


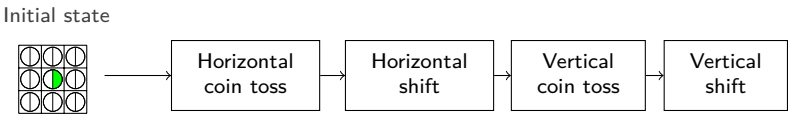
# DTQW on a grid

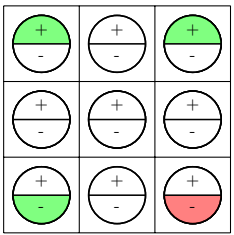
Initial state



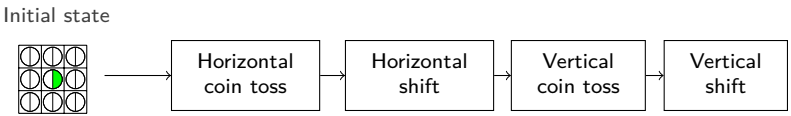



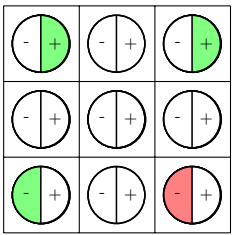
# DTQW on a grid



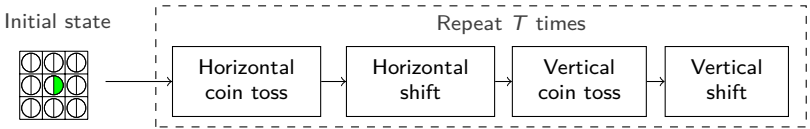



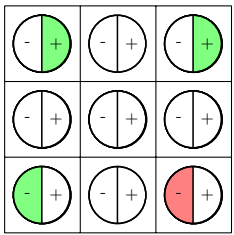
# DTQW on a grid



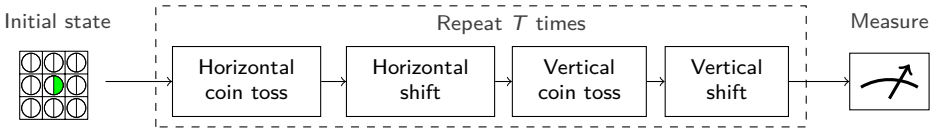



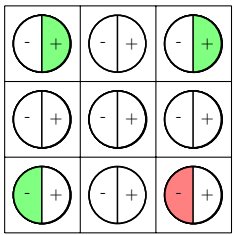
# DTQW on a grid



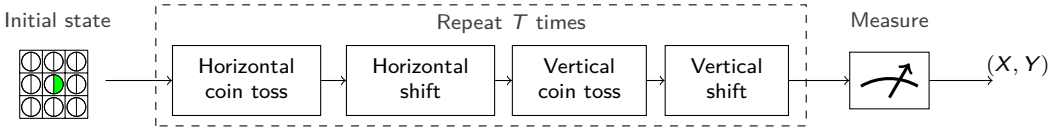



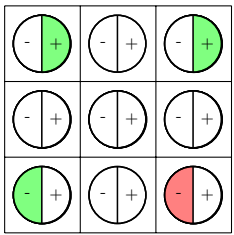
# DTQW on a grid



# DTQW on a grid

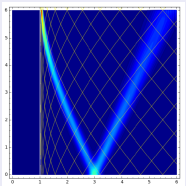


# Applications

# Applications

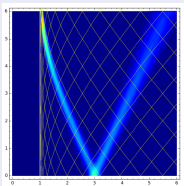
## Simulation



Di Molfetta, Brachet, Debbasch, 2013, PRA
Arrighi, Facchini, Forets, 2016, QIP

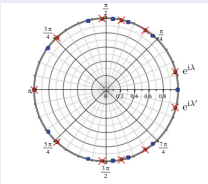
# Applications

## Simulation



Di Molfetta, Brachet, Debbasch, 2013, PRA
Arrighi, Facchini, Forets, 2016, QIP

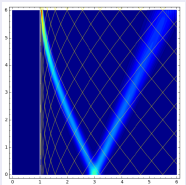
## Searching




Magniez, Nayak, Roland, Santha, 2010, ACM
**MR**, Guillet, Arrighi, Di Molfetta, 2020, PRL

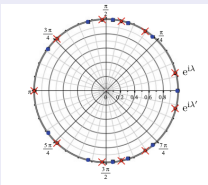
# Applications

## Simulation



Di Molfetta, Brachet, Debbasch, 2013, PRA
Arrighi, Facchini, Forets, 2016, QIP

## Searching

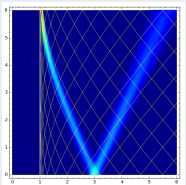


Magniez, Nayak, Roland, Santha, 2010, ACM
**MR**, Guillet, Arrighi, Di Molfetta, 2020, PRL

## Variations of the model

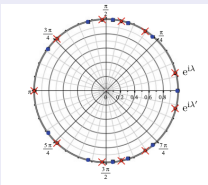
# Applications

## Simulation



Di Molfetta, Brachet, Debbasch, 2013, PRA
Arrighi, Facchini, Forets, 2016, QIP

## Searching



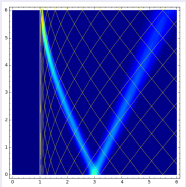
Magniez, Nayak, Roland, Santha, 2010, ACM
**MR**, Guillet, Arrighi, Di Molfetta, 2020, PRL

## Variations of the model

- The coin toss operator can be different.

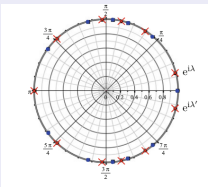
# Applications

## Simulation



Di Molfetta, Brachet, Debbasch, 2013, PRA
Arrighi, Facchini, Forets, 2016, QIP

## Searching



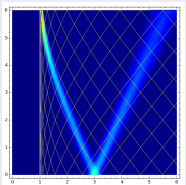
Magniez, Nayak, Roland, Santha, 2010, ACM
**MR**, Guillet, Arrighi, Di Molfetta, 2020, PRL

## Variations of the model

- The coin toss operator can be different.
- The topology can be different.

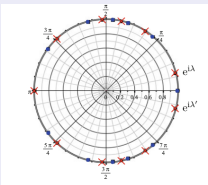
# Applications

## Simulation



Di Molfetta, Brachet, Debbasch, 2013, PRA
Arrighi, Facchini, Forets, 2016, QIP

## Searching



Magniez, Nayak, Roland, Santha, 2010, ACM
**MR**, Guillet, Arrighi, Di Molfetta, 2020, PRL

## Variations of the model

- The coin toss operator can be different.
- The topology can be different.
- The coin space can be different.

# Searching discrete time quantum walk on grids

# Table of Contents

# Searching

**Goal**

We want to find a marked position $(x_m, y_m)$ over a cyclic grid with high probability.

# Searching

## Goal

We want to find a marked position $(x_m, y_m)$ over a cyclic grid with high probability.

**Classical oracle:**

$$f : \begin{cases} (x_m, y_m) & \mapsto \quad 1 \end{cases}$$

# Searching

## Goal

We want to find a marked position $(x_m, y_m)$ over a cyclic grid with high probability.

**Classical oracle:**

$$f : \begin{cases} (x_m, y_m) & \mapsto \quad 1 \\ (x, y) \neq (x_m, y_m) & \mapsto \quad 0. \end{cases}$$

# Searching

## Goal

We want to find a marked position $(x_m, y_m)$ over a cyclic grid with high probability.

**Classical oracle:**

$$f : \begin{cases} (x_m, y_m) & \mapsto \quad 1 \\ (x, y) \neq (x_m, y_m) & \mapsto \quad 0. \end{cases}$$

**Quantum oracle:**

$$R : \begin{cases} \begin{pmatrix} z^+_{x_m,y_m} \\ z^-_{x_m,y_m} \end{pmatrix} \mapsto \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} z^+_{x_m,y_m} \\ z^-_{x_m,y_m} \end{pmatrix} & \text{(marked)} \end{cases}$$

# Searching

## Goal

We want to find a marked position $(x_m, y_m)$ over a cyclic grid with high probability.

**Classical oracle:**

$$f : \begin{cases} (x_m, y_m) & \mapsto \quad 1 \\ (x, y) \neq (x_m, y_m) & \mapsto \quad 0. \end{cases}$$

**Quantum oracle:**

$$R : \begin{cases} \begin{pmatrix} z^+_{x_m,y_m} \\ z^-_{x_m,y_m} \end{pmatrix} \mapsto \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} z^+_{x_m,y_m} \\ z^-_{x_m,y_m} \end{pmatrix} & \text{(marked)} \\ \begin{pmatrix} z^+_{x,y} \\ z^-_{x,y} \end{pmatrix} \mapsto \begin{pmatrix} z^+_{x,y} \\ z^-_{x,y} \end{pmatrix} & \text{(non-marked).} \end{cases}$$

# Searching

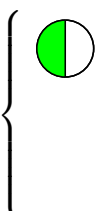
## Goal

We want to find a marked position $(x_m, y_m)$ over a cyclic grid with high probability.

**Classical oracle:**

$$f : \begin{cases} (x_m, y_m) & \mapsto \quad 1 \\ (x, y) \neq (x_m, y_m) & \mapsto \quad 0. \end{cases}$$

**Quantum oracle:**

$R :$ 

(marked)

# Searching

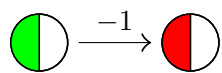
## Goal

We want to find a marked position $(x_m, y_m)$ over a cyclic grid with high probability.

**Classical oracle:**

$$f : \begin{cases} (x_m, y_m) & \mapsto \quad 1 \\ (x, y) \neq (x_m, y_m) & \mapsto \quad 0. \end{cases}$$

**Quantum oracle:**

$$R : \begin{cases} \text{  } & \text{(marked)} \\ \\ \\ \\ \end{cases}$$

# Searching

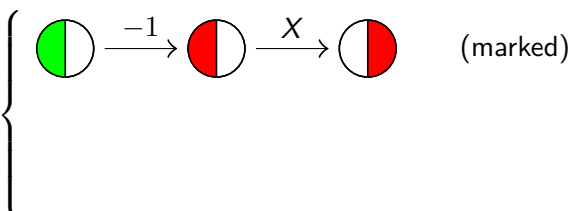
## Goal

We want to find a marked position $(x_m, y_m)$ over a cyclic grid with high probability.

**Classical oracle:**

$$f : \begin{cases} (x_m, y_m) & \mapsto \quad 1 \\ (x, y) \neq (x_m, y_m) & \mapsto \quad 0. \end{cases}$$

**Quantum oracle:**

$R :$ 

# Searching

## Goal

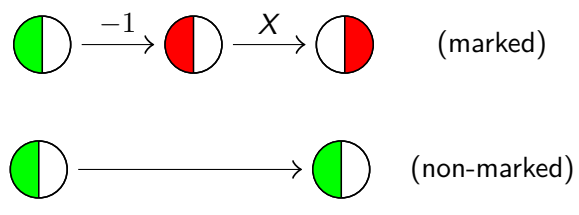
We want to find a marked position $(x_m, y_m)$ over a cyclic grid with high probability.

**Classical oracle:**

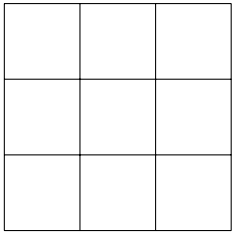
$$f : \begin{cases} (x_m, y_m) & \mapsto \quad 1 \\ (x, y) \neq (x_m, y_m) & \mapsto \quad 0. \end{cases}$$

**Quantum oracle:**

$R :$

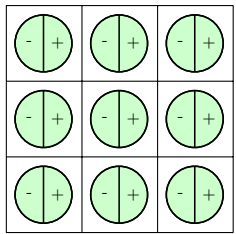


# Searching scheme

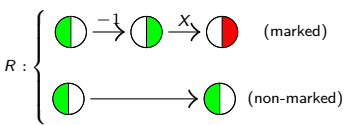


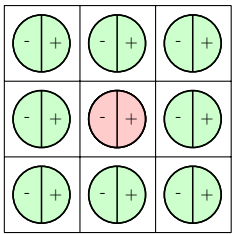
# Searching scheme

Diagonal state

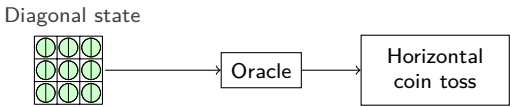


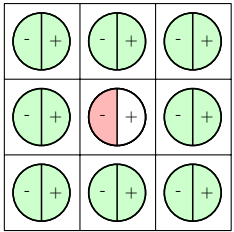
# Searching scheme

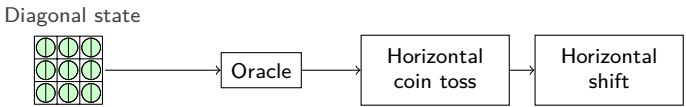



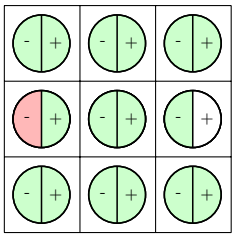
# Searching scheme



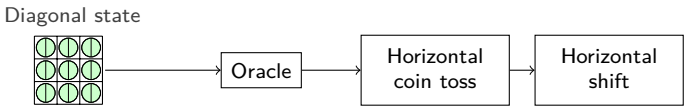



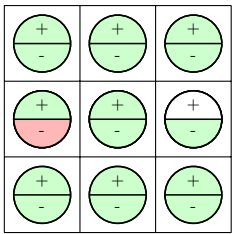
# Searching scheme



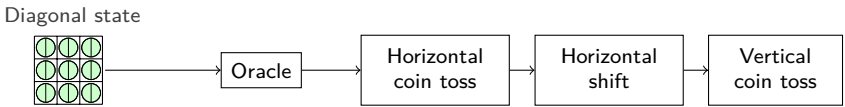



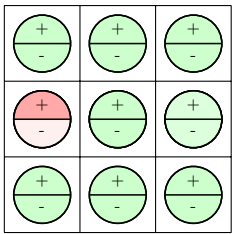
# Searching scheme



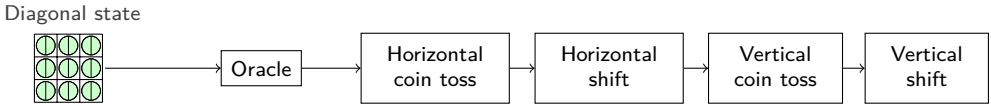



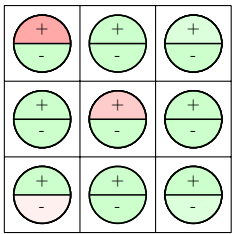
# Searching scheme



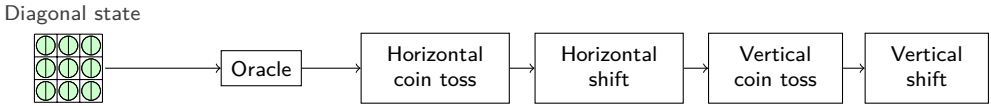



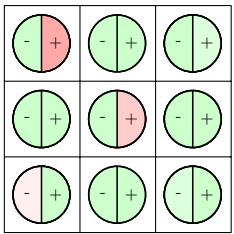
# Searching scheme



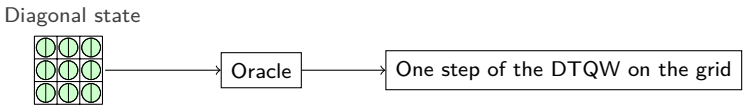



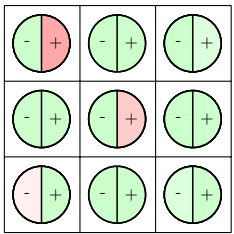
# Searching scheme



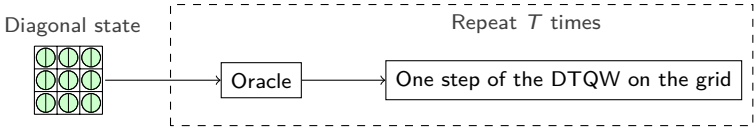



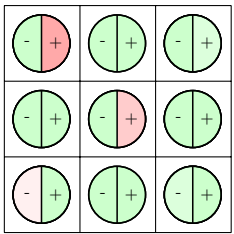
# Searching scheme



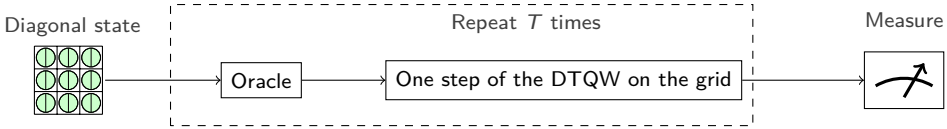



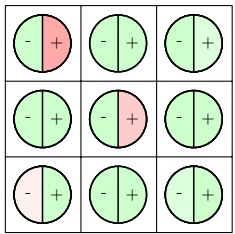
# Searching scheme



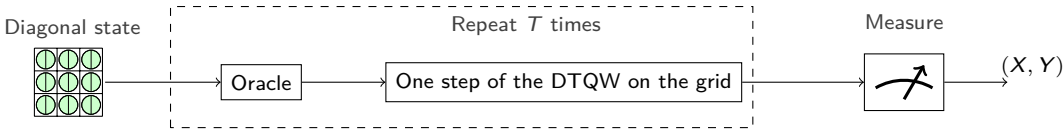



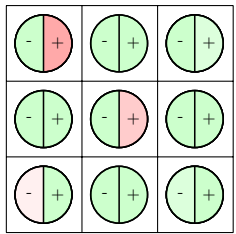
# Searching scheme



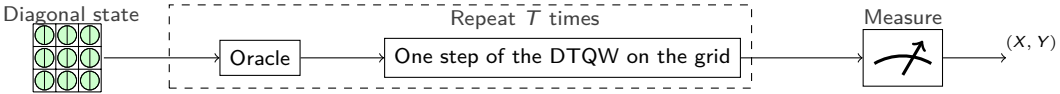



# Searching scheme



# Searching example



# Complexity

For a $\sqrt{N} \times \sqrt{N}$ grid,

Method from Portugal, 2018, Springer (Book)

# Complexity

For a $\sqrt{N} \times \sqrt{N}$ grid,

- Signal : $p(t) \sim p_s \sin^2(\lambda t)$

Method from Portugal, 2018, Springer (Book)

# Complexity

For a $\sqrt{N} \times \sqrt{N}$ grid,

- Signal : $p(t) \sim p_s \sin^2(\lambda t)$
- Hitting time : $t_{opt} = \frac{\pi}{2\lambda} \sim \frac{\sqrt{\pi N \ln N}}{4}$

Method from Portugal, 2018, Springer (Book)

# Complexity

For a $\sqrt{N} \times \sqrt{N}$ grid,

- Signal : $p(t) \sim p_s \sin^2(\lambda t)$
- Hitting time : $t_{opt} = \frac{\pi}{2\lambda} \sim \frac{\sqrt{\pi N \ln N}}{4}$
- Probability of success : $p_s \sim \frac{\pi}{4 \ln N}$

Method from Portugal, 2018, Springer (Book)

# Complexity

For a $\sqrt{N} \times \sqrt{N}$ grid,

- Signal : $p(t) \sim p_s \sin^2(\lambda t)$
- Hitting time : $t_{opt} = \frac{\pi}{2\lambda} \sim \frac{\sqrt{\pi N \ln N}}{4}$
- Probability of success : $p_s \sim \frac{\pi}{4 \ln N}$

## Complexity

After repeating a logarithmic number of times, the algorithm succeeds with probability $1 - \epsilon$ and has a complexity of

$$O\left(\sqrt{N}\ \ln^{3/2} N\ \ln 1/\epsilon\right)$$

Method from Portugal, 2018, Springer (Book)

# Complexity

For a $\sqrt{N} \times \sqrt{N}$ grid,

- Signal : $p(t) \sim p_s \sin^2(\lambda t)$
- Hitting time : $t_{opt} = \frac{\pi}{2\lambda} \sim \frac{\sqrt{\pi N \ln N}}{4}$
- Probability of success : $p_s \sim \frac{\pi}{4 \ln N}$

## Complexity

After repeating a logarithmic number of times, the algorithm succeeds with probability $1 - \epsilon$ and has a complexity of

$$O\left(\sqrt{N}\ \ln^{3/2} N\ \ln 1/\epsilon\right) \qquad \underset{\text{classical algorithm}}{\subseteq} \qquad O(N).$$

Method from Portugal, 2018, Springer (Book)

# Complexity

For a $\sqrt{N} \times \sqrt{N}$ grid,

- Signal : $p(t) \sim p_s \sin^2(\lambda t)$
- Hitting time : $t_{opt} = \frac{\pi}{2\lambda} \sim \frac{\sqrt{\pi N \ln N}}{4}$
- Probability of success : $p_s \sim \frac{\pi}{4 \ln N}$

## Complexity

After repeating a logarithmic number of times, the algorithm succeeds with probability $1 - \epsilon$ and has a complexity of

$$\underbrace{O(\sqrt{N})}_{\text{Grover algorithm}} \subseteq O\left(\sqrt{N}\ \ln^{3/2} N\ \ln 1/\epsilon\right) \subseteq \underbrace{O(N).}_{\text{classical algorithm}}$$

Method from Portugal, 2018, Springer (Book)

# Table of Contents

# Searching two positions

An instance is composed of two marked position $m_0$ and $m_1$ on a $\sqrt{N} \times \sqrt{N}$ grid. The goal is to measure one of them.

# Searching two positions

An instance is composed of two marked position $m_0$ and $m_1$ on a $\sqrt{N} \times \sqrt{N}$ grid. The goal is to measure one of them.

**Theorem (MR, Kadri, Di Molfetta, 2023, PRR)**

*The complexity for instances of two marked positions is* $O(\sqrt{N} \ln^{\alpha} N \ln 1/\epsilon)$ *except for at most* $O\left(\sqrt{N \ln N}\right)$ *instances.*

## There exist an *optimal* instance (**MR**, Kadri, Di Molfetta, 2023, PRR)

The complexity of searching the two elements $m_0 = (0, 0)$ and $m_1 = (0, 2)$ on a $\sqrt{N} \times \sqrt{N}$ is

$$O\left(\sqrt{N} \ln^{3/2} N \ln 1/\epsilon\right).$$

## There exist an *optimal* instance (**MR**, Kadri, Di Molfetta, 2023, PRR)

The complexity of searching the two elements $m_0 = (0, 0)$ and $m_1 = (0, 2)$ on a $\sqrt{N} \times \sqrt{N}$ is

$$O\left(\sqrt{N} \ln^{3/2} N \ln 1/\epsilon\right).$$

## There exist a *suboptimal* instance (**MR**, Kadri, Di Molfetta, 2023, PRR)

The complexity of searching the two elements $m_0 = (0, 0)$ and $m_1 = (1, 1)$ on a $\sqrt{N} \times \sqrt{N}$ is

$$O\left(N \ln N \ln 1/\epsilon\right).$$

**There exist an *optimal* instance (MR, Kadri, Di Molfetta, 2023, PRR)**

The complexity of searching the two elements $m_0 = (0, 0)$ and $m_1 = (0, 2)$ on a $\sqrt{N} \times \sqrt{N}$ is

$$O\left(\sqrt{N} \ln^{3/2} N \ln 1/\epsilon\right).$$

**There exist a *suboptimal* instance (MR, Kadri, Di Molfetta, 2023, PRR)**

The complexity of searching the two elements $m_0 = (0, 0)$ and $m_1 = (1, 1)$ on a $\sqrt{N} \times \sqrt{N}$ is

$$O\left(N \ln N \ln 1/\epsilon\right).$$

Worst case complexity:

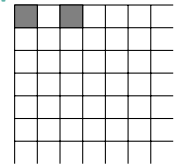
$$O\left(N \ln N \ln 1/\epsilon\right).$$

Average complexity:

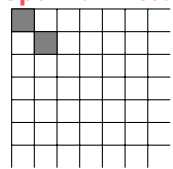
$$O\left(\sqrt{N} \log^{\alpha} N \ln 1/\epsilon\right).$$

# Example for a $128 \times 128$ grid

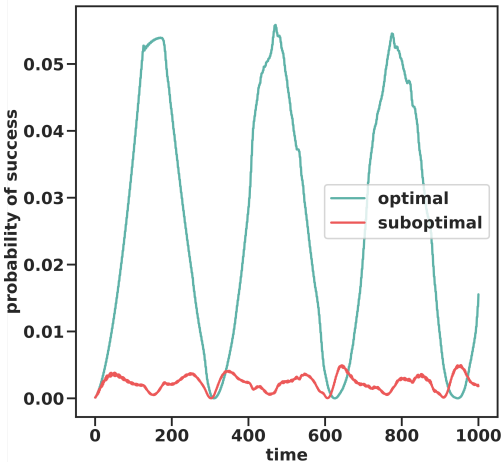
**Optimal instance:**



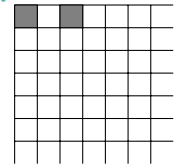
**Suboptimal instance:**



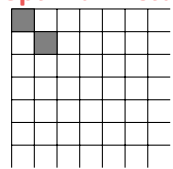
# Example for a $128 \times 128$ grid




**Optimal instance:**



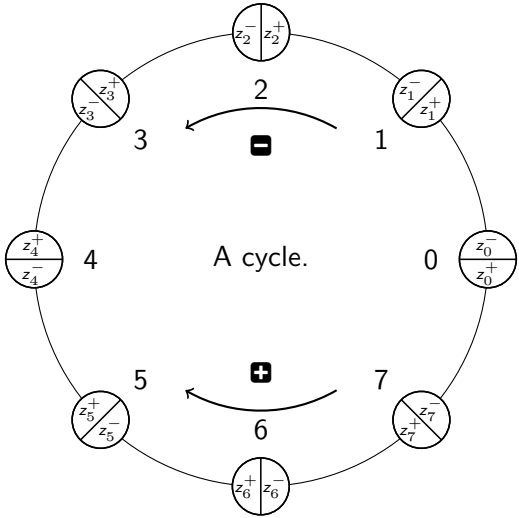
**Suboptimal instance:**



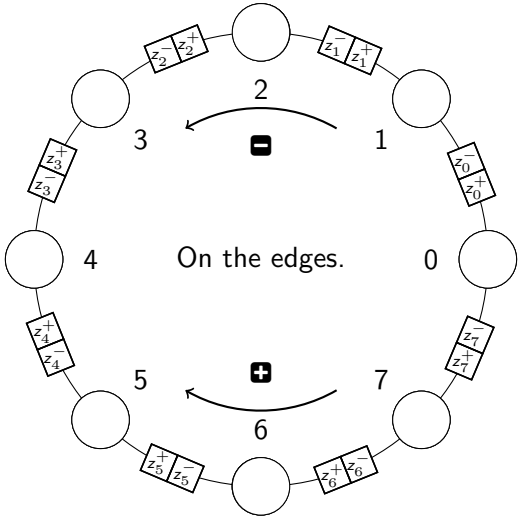
# Quantum Walk on graphs

# Table of Contents

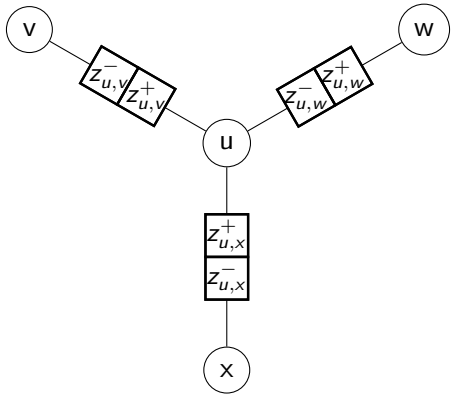
# Quantum walk on graphs



# Quantum walk on graphs

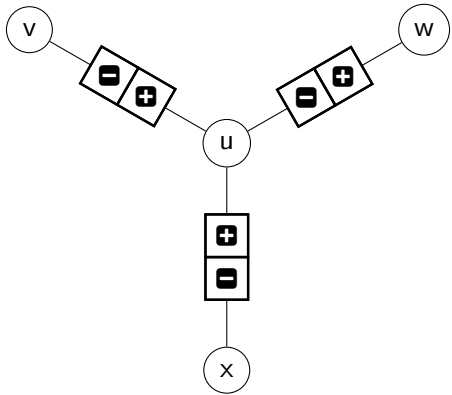


# Quantum walk on graphs



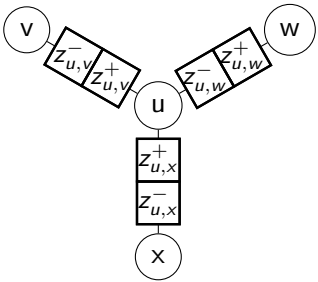

On the edges of a graph.

# Quantum walk on graphs



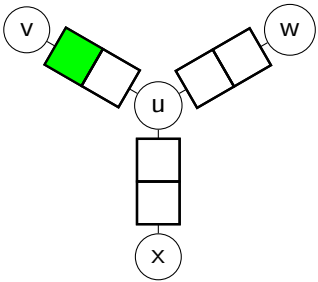

Require a polarity on the edges !

# Operations



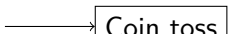
# Operations

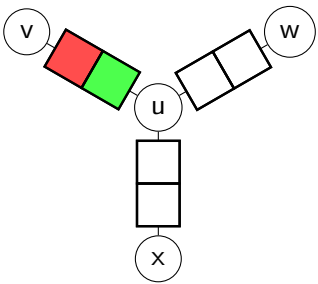
Initial state



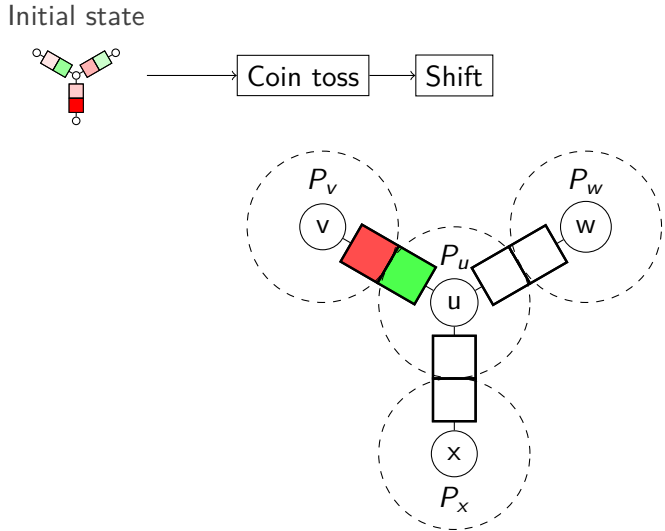
# Operations

Initial state

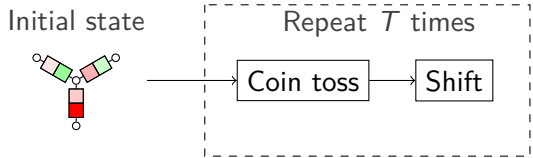


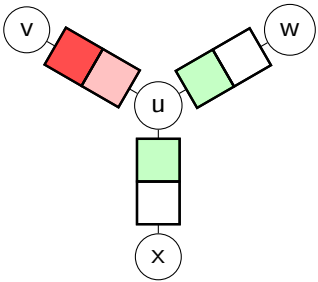



# Operations

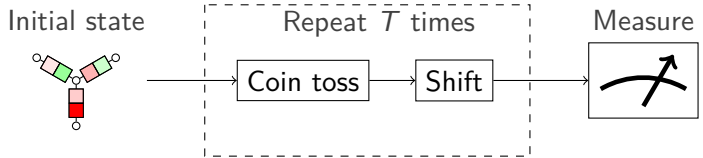


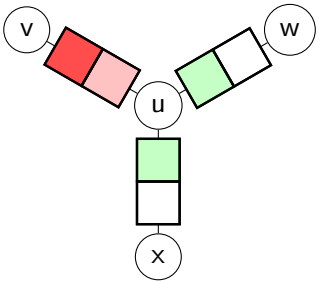
# Operations



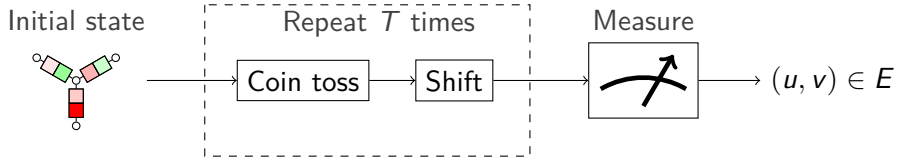



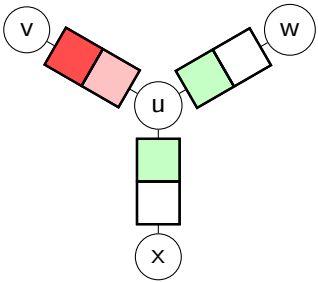
# Operations



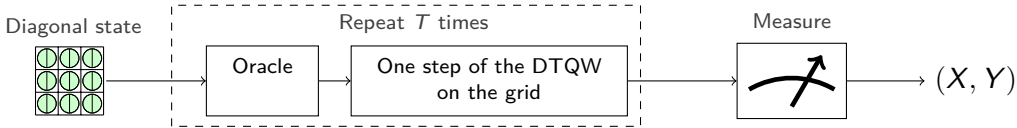



# Operations

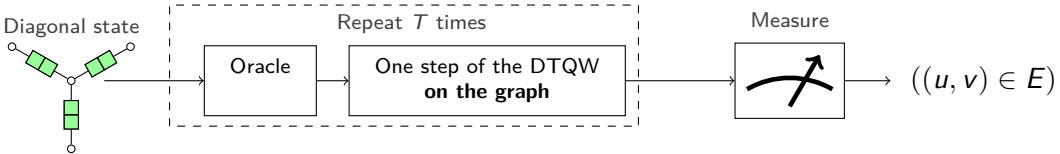

Initial state
Repeat $T$ times
Coin toss
Shift
Measure
$(u, v) \in E$


v
w
u
x

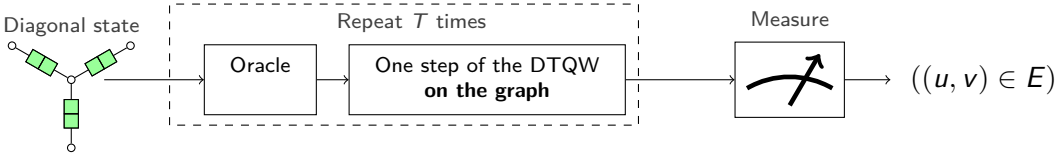
# Searching on the grid



# Searching on the graph

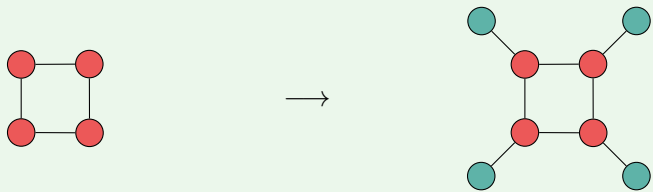


# Searching on the graph




## Remark

The position measured are edges. For searching nodes, we can use the following transformation:



# Complexities

| 2D-Grid | Hypercube | Complete Graph | Scale-Free Graph |
|---|---|---|---|
| $O\left(\sqrt{M \log^{\alpha} M}\right)$ | $O\left(\sqrt{M}\right)$ | $O\left(\sqrt{M}\right)$ | $O\left(\sqrt{M}\right)$ |

**MR**, 2024, PYPI (Python package)
**MR**, Di Molfetta, 2024, ASCAT
**MR**, Di Molfetta, 2025, Preprint

# Complexities

| 2D-Grid | Hypercube | Complete Graph | Scale-Free Graph |
|---|---|---|---|
| $O\left(\sqrt{M \log^{\alpha} M}\right)$ | $O\left(\sqrt{M}\right)$ | $O\left(\sqrt{M}\right)$ | $O\left(\sqrt{M}\right)$ |

**Better than the classical Deep First Search which is in $O(M)$ !**

**MR**, 2024, PYPI (Python package)
**MR**, Di Molfetta, 2024, ASCAT
**MR**, Di Molfetta, 2025, Preprint

# Complexities

| 2D-Grid | Hypercube | Complete Graph | Scale-Free Graph |
|---|---|---|---|
| $O\left(\sqrt{M \log^{\alpha} M}\right)$ | $O\left(\sqrt{M}\right)$ | $O\left(\sqrt{M}\right)$ | $O\left(\sqrt{M}\right)$ |

**Better than the classical Deep First Search which is in $O(M)$ !**

Implementation of a Python package in rust to obtain the numerical results.

**MR**, 2024, PYPI (Python package)
**MR**, Di Molfetta, 2024, ASCAT
**MR**, Di Molfetta, 2025, Preprint

# Table of Contents

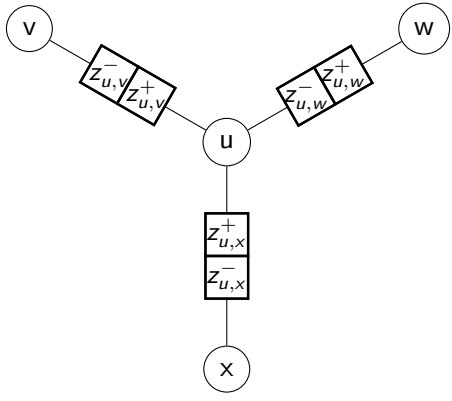
# Central idea

# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**

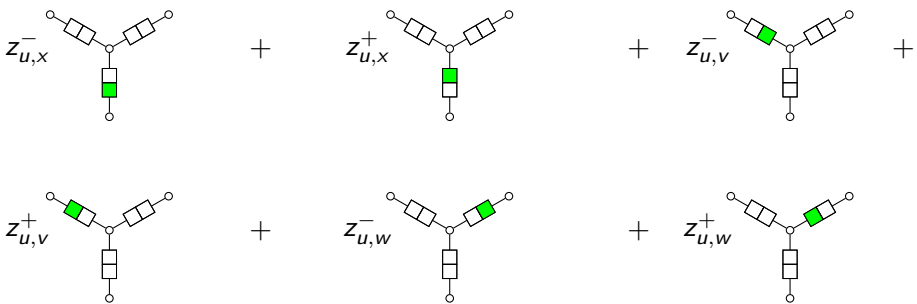
# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**



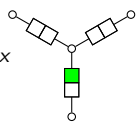
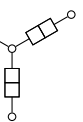
# Central idea

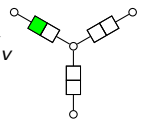
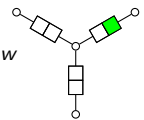
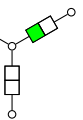
**We replace each coordinates by a qubit and work on the unary subspace.**



# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**

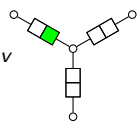
$z^-_{u,x} |100000\rangle + z^+_{u,x}$  $+ z^-_{u,v}$  $+$

$z^+_{u,v}$  $+ z^-_{u,w}$  $+ z^+_{u,w}$ 

# Central idea

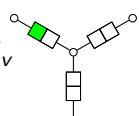
**We replace each coordinates by a qubit and work on the unary subspace.**

$$z^-_{u,x} \left|100000\right\rangle + z^+_{u,x} \left|010000\right\rangle + z^-_{u,v}$$

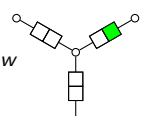


$$+$$

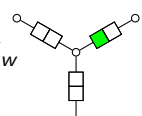
$$z^+_{u,v}$$



$$+ \; z^-_{u,w}$$



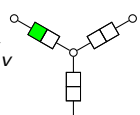
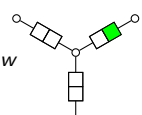
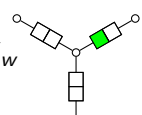
$$+ \; z^+_{u,w}$$



# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**

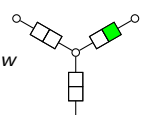
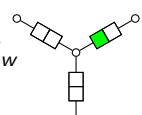
$$z^-_{u,x} \left|100000\right\rangle + z^+_{u,x} \left|010000\right\rangle + z^-_{u,v} \left|001000\right\rangle +$$

$$z^+_{u,v}$$  $$+ \quad z^-_{u,w}$$  $$+ \quad z^+_{u,w}$$ 

# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**

$$z^-_{u,x} \left|100000\right\rangle + z^+_{u,x} \left|010000\right\rangle + z^-_{u,v} \left|001000\right\rangle +$$

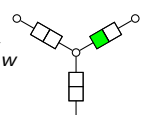
$$z^+_{u,v} \left|000100\right\rangle + z^-_{u,w}$$  $$+ z^+_{u,w}$$ 

# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**

$$z^-_{u,x} \left|100000\right\rangle + z^+_{u,x} \left|010000\right\rangle + z^-_{u,v} \left|001000\right\rangle +$$

$$z^+_{u,v} \left|000100\right\rangle + z^-_{u,w} \left|000010\right\rangle + z^+_{u,w}$$



# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**

$$z^-_{u,x} |100000\rangle + z^+_{u,x} |010000\rangle + z^-_{u,v} |001000\rangle +$$

$$z^+_{u,v} |000100\rangle + z^-_{u,w} |000010\rangle + z^+_{u,w} |000001\rangle$$

# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**

| Centralized computing (Binary) | Distributed computing (Unary) |
| --- | --- |
| | |

# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**

**Centralized computing (Binary)**

| $\|00\rangle$ | $\|01\rangle$ | $\|10\rangle$ | $\|11\rangle$ |
|---|---|---|---|
| $z_0$ | $z_1$ | $z_2$ | $z_3$ |

**Distributed computing (Unary)**

| $\|0001\rangle$ | $\|0010\rangle$ | $\|0100\rangle$ | $\|1000\rangle$ | ... |
|---|---|---|---|---|
| $z_0$ | $z_1$ | $z_2$ | $z_3$ | $0$ |

# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**

**Centralized computing (Binary)**

| $\|00\rangle$ | $\|01\rangle$ | $\|10\rangle$ | $\|11\rangle$ |
|---|---|---|---|
| $z_0$ | $z_1$ | $z_2$ | $z_3$ |

$$C = \begin{pmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{pmatrix}$$

**Distributed computing (Unary)**

| $\|0001\rangle$ | $\|0010\rangle$ | $\|0100\rangle$ | $\|1000\rangle$ | ... |
|---|---|---|---|---|
| $z_0$ | $z_1$ | $z_2$ | $z_3$ | $0$ |

$$\Lambda(C) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C_{00} & C_{01} & 0 \\ 0 & C_{10} & C_{11} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# Central idea

**We replace each coordinates by a qubit and work on the unary subspace.**

**Centralized computing (Binary)**

| $\|00\rangle$ | $\|01\rangle$ | $\|10\rangle$ | $\|11\rangle$ |
|---|---|---|---|
| $z_0$ | $z_1$ | $z_2$ | $z_3$ |

$$C = \begin{pmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{pmatrix}$$

Apply $P_u$ around node $u$

**Distributed computing (Unary)**

| $\|0001\rangle$ | $\|0010\rangle$ | $\|0100\rangle$ | $\|1000\rangle$ | ... |
|---|---|---|---|---|
| $z_0$ | $z_1$ | $z_2$ | $z_3$ | $0$ |

$$\Lambda(C) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C_{00} & C_{01} & 0 \\ 0 & C_{10} & C_{11} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Apply $P_u$ on the unary subspace around node $u$ and identity everywhere else.

# Rules

# Rules

Rule 1 (local operation): We can apply any $2 \times 2$ unitary on a qubit.

# Rules

Rule 1 (local operation): We can apply any $2 \times 2$ unitary on a qubit.

Rule 2 (communication): We can apply any $4 \times 4$ unitary on two **connected** qubits.

# Rules

Rule 1 (local operation): We can apply any $2 \times 2$ unitary on a qubit.

Rule 2 (communication): We can apply any $4 \times 4$ unitary on two **connected** qubits.

Rule 3 (connectivity): We cannot apply unitary on non-connected qubits.

# Rules

Rule 1 (local operation): We can apply any $2 \times 2$ unitary on a qubit.

Rule 2 (communication): We can apply any $4 \times 4$ unitary on two **connected** qubits.

Rule 3 (connectivity): We cannot apply unitary on non-connected qubits.

# Rules

Rule 1 (local operation): We can apply any $2 \times 2$ unitary on a qubit.

Rule 2 (communication): We can apply any $4 \times 4$ unitary on two **connected** qubits.

Rule 3 (connectivity): We cannot apply unitary on non-connected qubits.

**Goal**

Minimize the number of communications (i.e. two-qubits gates).

# Rules

Rule 1 (local operation): We can apply any $2 \times 2$ unitary on a qubit.

Rule 2 (communication): We can apply any $4 \times 4$ unitary on two **connected** qubits.

Rule 3 (connectivity): We cannot apply unitary on non-connected qubits.

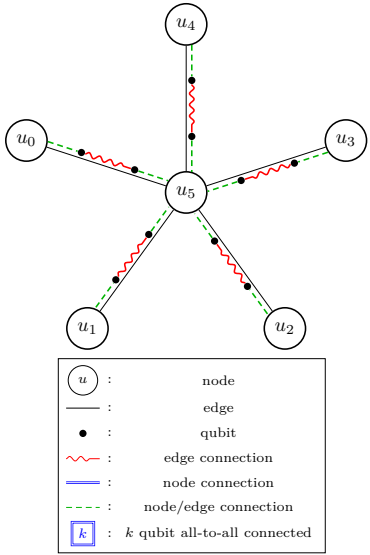
**Goal**

Minimize the number of communications (i.e. two-qubits gates).

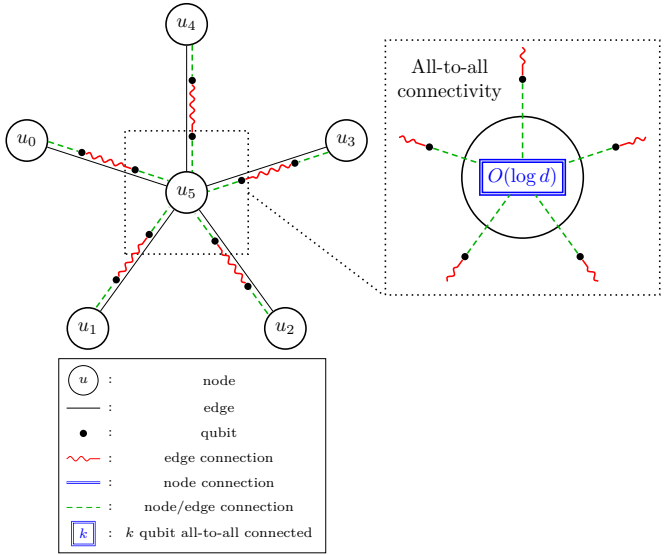
**Remark**

The coin toss uses $1$ two-qubits gate

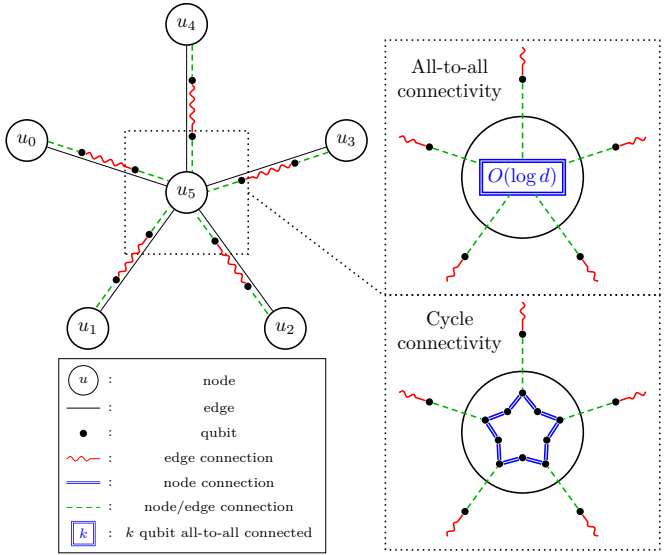
# Connectivity



# Connectivity



# Connectivity



# Shift with all-to-all connectivity

| Communication cost | Edges around $u$ | Node $u$[1] |
|---|---|---|

[1] Add one qubit to make it normalized

[2] $d = \deg(u)$

# Shift with all-to-all connectivity

| Communication cost | Edges around $u$ | | | | | Node $u$[1] | | | |
|---|---|---|---|---|---|---|---|---|---|
| State | $\|0001\rangle$ | $\|0010\rangle$ | $\|0100\rangle$ | $\|1000\rangle$ | ... | $\|00\rangle$ | $\|01\rangle$ | $\|10\rangle$ | $\|11\rangle$ |
| | $z_0$ | $z_1$ | $z_2$ | $z_3$ | | 0 | 0 | 0 | 0 |

[1] Add one qubit to make it normalized

[2] $d = \deg(u)$

# Shift with all-to-all connectivity

| Communication cost | Edges around $u$ | Node $u$[1] |
|---|---|---|
| State | $\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$ ... | $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ |
| | $z_0$ $z_1$ $z_2$ $z_3$ | 0 0 0 0 |
| Edge/node: $O(\ln d)$[2] | Send unary state on edges as binary to $u$ | |

[1] Add one qubit to make it normalized

[2] $d = \deg(u)$

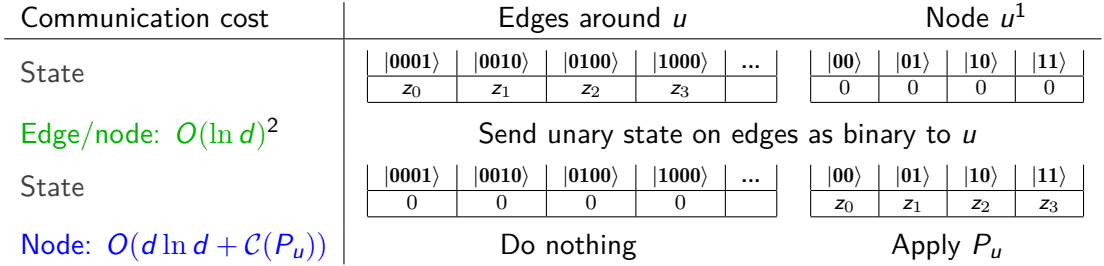
# Shift with all-to-all connectivity

| Communication cost | Edges around $u$ | Node $u$[1] |
|---|---|---|
| State | $\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$ ... | $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ |
| | $z_0$ $z_1$ $z_2$ $z_3$ | 0 0 0 0 |
| Edge/node: $O(\ln d)$[2] | Send unary state on edges as binary to $u$ | |
| State | $\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$ ... | $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ |
| | 0 0 0 0 | $z_0$ $z_1$ $z_2$ $z_3$ |

[1] Add one qubit to make it normalized

[2] $d = \deg(u)$

# Shift with all-to-all connectivity




[1] Add one qubit to make it normalized

[2] $d = \deg(u)$

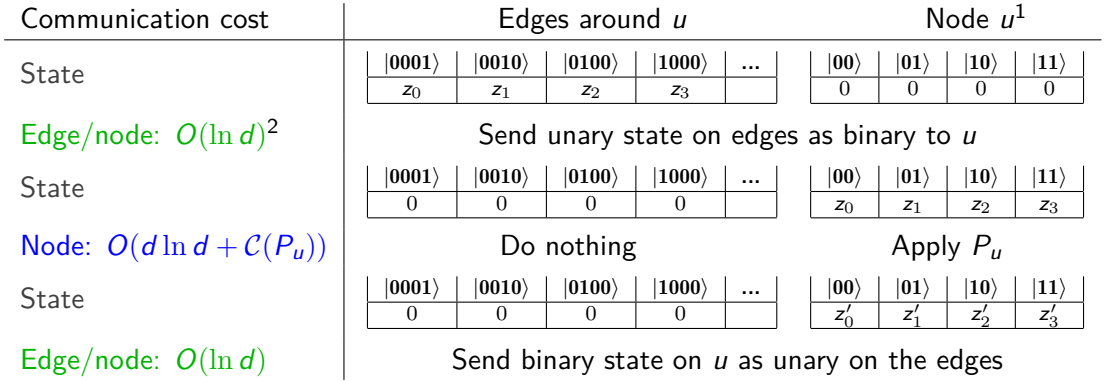
# Shift with all-to-all connectivity

| Communication cost | Edges around $u$ | Node $u$[1] |
|---|---|---|
| State | $\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$ ... <br> $z_0$ $z_1$ $z_2$ $z_3$ | $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ <br> 0 0 0 0 |
| Edge/node: $O(\ln d)$[2] | Send unary state on edges as binary to $u$ | |
| State | $\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$ ... <br> 0 0 0 0 | $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ <br> $z_0$ $z_1$ $z_2$ $z_3$ |
| Node: $O(d \ln d + \mathcal{C}(P_u))$ | Do nothing | Apply $P_u$ |
| State | $\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$ ... <br> 0 0 0 0 | $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ <br> $z'_0$ $z'_1$ $z'_2$ $z'_3$ |

[1] Add one qubit to make it normalized

[2] $d = \deg(u)$

# Shift with all-to-all connectivity




[1] Add one qubit to make it normalized

[2] $d = \deg(u)$

# Shift with all-to-all connectivity

| Communication cost | Edges around $u$ | Node $u$[1] |
|---|---|---|
| State | $\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$ ... <br> $z_0$ $z_1$ $z_2$ $z_3$ | $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ <br> 0 0 0 0 |
| Edge/node: $O(\ln d)$[2] | Send unary state on edges as binary to $u$ | |
| State | $\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$ ... <br> 0 0 0 0 | $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ <br> $z_0$ $z_1$ $z_2$ $z_3$ |
| Node: $O(d \ln d + \mathcal{C}(P_u))$ | Do nothing | Apply $P_u$ |
| State | $\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$ ... <br> 0 0 0 0 | $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ <br> $z'_0$ $z'_1$ $z'_2$ $z'_3$ |
| Edge/node: $O(\ln d)$ | Send binary state on $u$ as unary on the edges | |
| State | $\|0001\rangle$ $\|0010\rangle$ $\|0100\rangle$ $\|1000\rangle$ ... <br> $z'_0$ $z'_1$ $z'_2$ $z'_3$ | $\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$ <br> 0 0 0 0 |

[1] Add one qubit to make it normalized

[2] $d = \deg(u)$

# Shift with cyclic connectivity

# Shift with cyclic connectivity

**(MR, Di Molfetta, 2025, Preprint)**

A DTQW on the cycle with well-chosen space-time dependent coins can realize any $d \times d$ unitary in $O(d^2)$ steps.

# Shift with cyclic connectivity

**(MR, Di Molfetta, 2025, Preprint)**

A DTQW on the cycle with well-chosen space-time dependent coins can realize any $d \times d$ unitary in $O(d^2)$ steps.

$O(d^2)$ steps of a DTQW on a cycle $=$ Unitary $P_u$

# Shift with cyclic connectivity

**(MR, Di Molfetta, 2025, Preprint)**

A DTQW on the cycle with well-chosen space-time dependent coins can realize any $d \times d$ unitary in $O(d^2)$ steps.

$O(d^2)$ steps of a DTQW on a cycle $=$ Unitary $P_u$



Single particle Quantum Cellular Automata

# Shift with cyclic connectivity

**(MR, Di Molfetta, 2025, Preprint)**

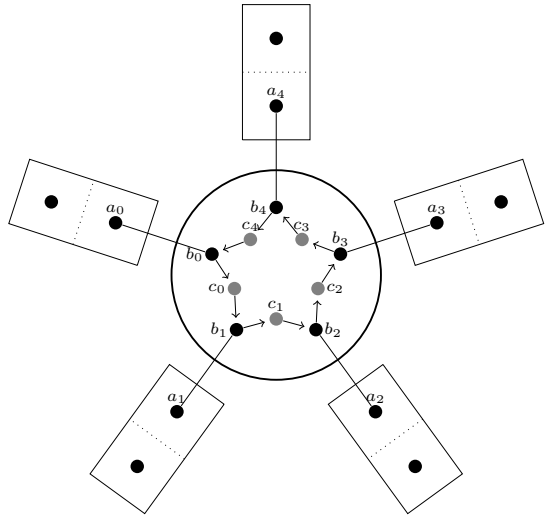
A DTQW on the cycle with well-chosen space-time dependent coins can realize any $d \times d$ unitary in $O(d^2)$ steps.

$O(d^2)$ steps of a DTQW on a cycle $=$ Unitary $P_u$

$\downarrow$

Single particle Quantum Cellular Automata $=$ Unitary $\Lambda(P_u)$

# Shift with cyclic connectivity

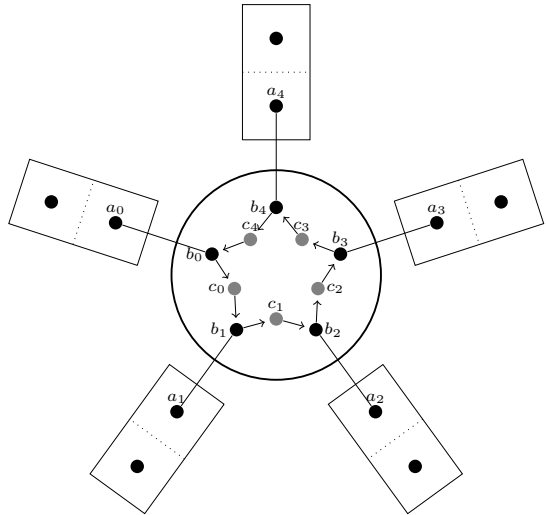


# Shift with cyclic connectivity

$a_k$: "Stay in place" coin state.

$b_k$: "Counter-clockwise" coin state.

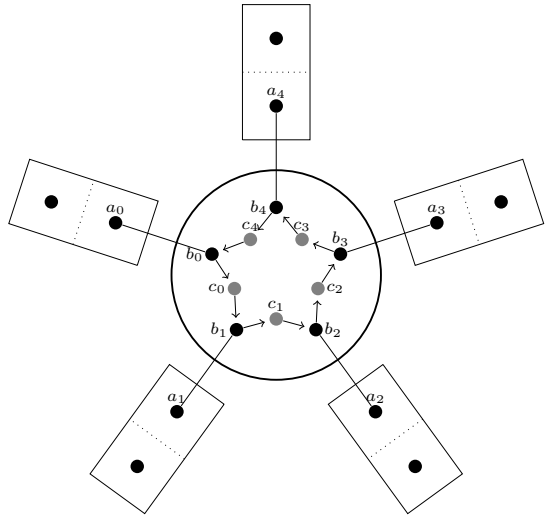
$c_k$: Ancillaries.



# Shift with cyclic connectivity

- $a_k$: "Stay in place" coin state.
- $b_k$: "Counter-clockwise" coin state.
- $c_k$: Ancillaries.

$O(d^2)$ steps $\Rightarrow$ $O(d^2)$ node communications.



# Shift with cyclic connectivity

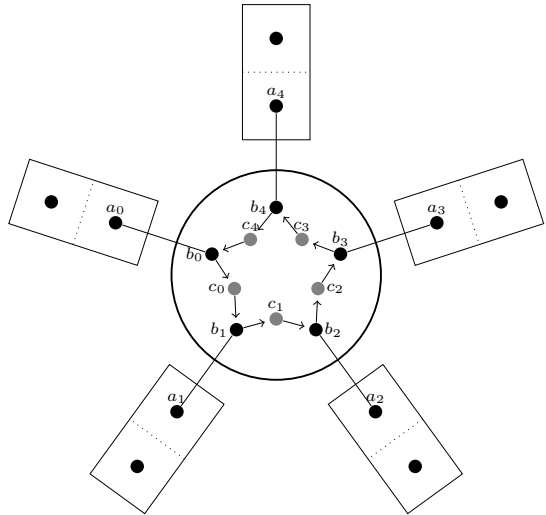
$a_k$: "Stay in place" coin state.

$b_k$: "Counter-clockwise" coin state.

$c_k$: Ancillaries.

$O(d^2)$ steps $\Rightarrow$ $O(d^2)$ node communications.

$O(d)$ edge/node communications.



# Communication cost for one step

| | **all-to-all** | **cyclic** |
|---|---|---|
| Node register | $O(\log d)$ qubits | $O(d)$ qubits |
| Edge communications | $O(1)$ | $O(1)$ |
| Node communications | $O(d \ln d + \mathcal{C}(P_u))$ | $O(d^2)$ |
| Edge/node communications | $O(\ln d)$ | $O(d)$ |

**MR**, Di Molfetta, 2024, ASCAT
**MR**, Di Molfetta, 2025, Preprint

# Conclusion and perspectives

# Conclusion and perspectives

## Searching two elements on grids

**Takeaway message:** Quantum advantage only in average.

# Conclusion and perspectives

## Searching two elements on grids

**Takeaway message:** Quantum advantage only in average.

- Figure out a way to avoid losing the quantum advantage (by changing the model, the coin).

# Conclusion and perspectives

## Searching two elements on grids

**Takeaway message:** Quantum advantage only in average.

- Figure out a way to avoid losing the quantum advantage (by changing the model, the coin).
- Check if other lattices show the same property (3d grid, hypercube, ...).

# Conclusion and perspectives

## Searching two elements on grids

**Takeaway message:** Quantum advantage only in average.

- Figure out a way to avoid losing the quantum advantage (by changing the model, the coin).
- Check if other lattices show the same property (3d grid, hypercube, ...).

## Searching on graphs

**Takeaway message:** A new model of DTQW on the edges of a graph with good searching properties.

# Conclusion and perspectives

## Searching two elements on grids

**Takeaway message:** Quantum advantage only in average.

- Figure out a way to avoid losing the quantum advantage (by changing the model, the coin).
- Check if other lattices show the same property (3d grid, hypercube, ...).

## Searching on graphs

**Takeaway message:** A new model of DTQW on the edges of a graph with good searching properties.

- Get more analytical results about searching.

# Conclusion and perspectives

## Searching two elements on grids

**Takeaway message:** Quantum advantage only in average.

- Figure out a way to avoid losing the quantum advantage (by changing the model, the coin).
- Check if other lattices show the same property (3d grid, hypercube, ...).

## Searching on graphs

**Takeaway message:** A new model of DTQW on the edges of a graph with good searching properties.

- Get more analytical results about searching.
- Find algorithmic applications linked to the polarity.

# Conclusion and perspectives

## Searching two elements on grids

**Takeaway message:** Quantum advantage only in average.

- Figure out a way to avoid losing the quantum advantage (by changing the model, the coin).
- Check if other lattices show the same property (3d grid, hypercube, ...).

## Searching on graphs

**Takeaway message:** A new model of DTQW on the edges of a graph with good searching properties.

- Get more analytical results about searching.
- Find algorithmic applications linked to the polarity.

## Distributed implementation

**Takeaway message:** We provided a distributed scheme to reproduce a DTQW on a graph using a network of qubits following the same graph.

# Conclusion and perspectives

## Searching two elements on grids

**Takeaway message:** Quantum advantage only in average.

- Figure out a way to avoid losing the quantum advantage (by changing the model, the coin).
- Check if other lattices show the same property (3d grid, hypercube, ...).

## Searching on graphs

**Takeaway message:** A new model of DTQW on the edges of a graph with good searching properties.

- Get more analytical results about searching.
- Find algorithmic applications linked to the polarity.

## Distributed implementation

**Takeaway message:** We provided a distributed scheme to reproduce a DTQW on a graph using a network of qubits following the same graph.

- Use the network of qubits for more general computation (a QCA-like extension of DTQW).

# Conclusion and perspectives

## Searching two elements on grids

**Takeaway message:** Quantum advantage only in average.

- Figure out a way to avoid losing the quantum advantage (by changing the model, the coin).
- Check if other lattices show the same property (3d grid, hypercube, ...).

## Searching on graphs

**Takeaway message:** A new model of DTQW on the edges of a graph with good searching properties.

- Get more analytical results about searching.
- Find algorithmic applications linked to the polarity.

## Distributed implementation

**Takeaway message:** We provided a distributed scheme to reproduce a DTQW on a graph using a network of qubits following the same graph.

- Use the network of qubits for more general computation (a QCA-like extension of DTQW).
- Seek distributed quantum computing applications[a]

[a] Collaboration JSPS Summer 2024 with François Le Gall

# The End

# Other models of QW on graphs

## Arc-Reversal Walk

Position: on the nodes

Evolution operator : $R \times (I \otimes C)$

where $R$ is the flip-flop operator and $C$ the coin.

The coin depends of the degree !

## Szegedy's Walk

Position: on the nodes

Evolution operator : $(2\Pi_B - I)(2\Pi_A - I)$

where $\Pi_A$ and $\Pi_B$ are computed from a classical Markov chain.

No coin in this model !

## Link between the models for the complete graph

**Our model**
$X$ coin
Grover operator for diffusion.

$=$

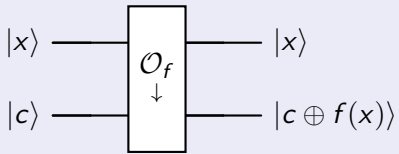
**Arc-Reversal Walk**
Grover coin.

$=$

**Szegedy's Walk**
Markov chain $\forall(u, v)$, $P_{u\to v} = \frac{1}{\sqrt{n-1}}$.

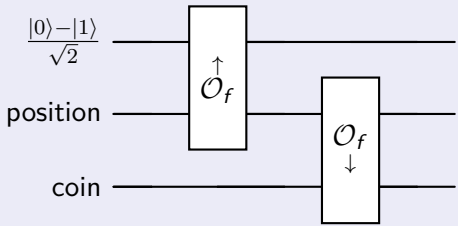
# Oracle construction

## Basic quantum oracle $\mathcal{O}_f$

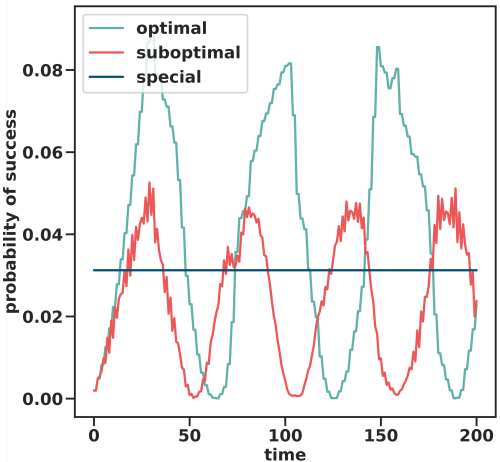



where $x \in \{0, 1, \ldots, N-1\}$ and $c \in \{0, 1\}$.

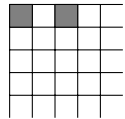
## Oracle for the quantum walk



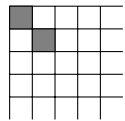
# Example for a $32 \times 32$ grid




**Optimal instance:**



**Suboptimal instance:**



**Special instance:**