# STUDY OF QUANTUM RANDOM WALK SEARCH ALGORITHM WITH QUDIT HOUSEHOLDER TRAVERSING COIN

*P. Danev* *H. Tonchev*

18/07/2022

# Contents

# INTRODUCTION

# Discrete time quantum random walk search

Discrete time quantum random walk search algorithm (DTQRWS)

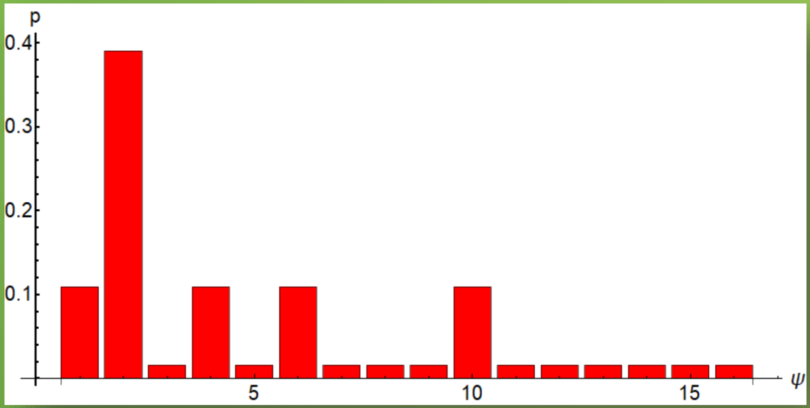
- Uses quantum walk to find searched element in unordered database;
- Quadratically faster than the corresponding classical search algorithms.

Quantum random walk algorithm is large category of quantum algorithms. It is used in variety of quantum information topics:

- quantum simulations;
- quantum algorithms;
- quantum cryptography.

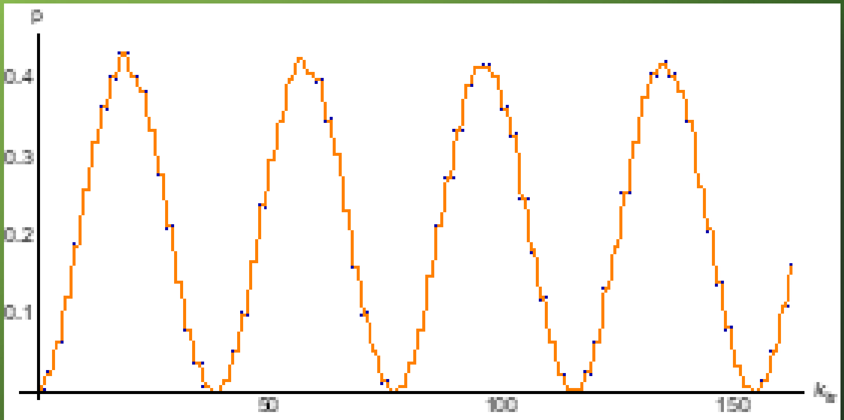
| DTQRWS | Grover search algorithm |
|---|---|
| Search in arbitrary topology | Search only in linear database |
| Needs more qubits | Needs less qubits |
| Double Oracle calls | Less Oracle calls |

1. QRWS is probabilistic algorithm with probability of finding the searched element $p = 1/2 - O(1/m_n)$.




Result of QRWS for hypercube with 16 vertices after $k_{itr} = 5$ iterations.

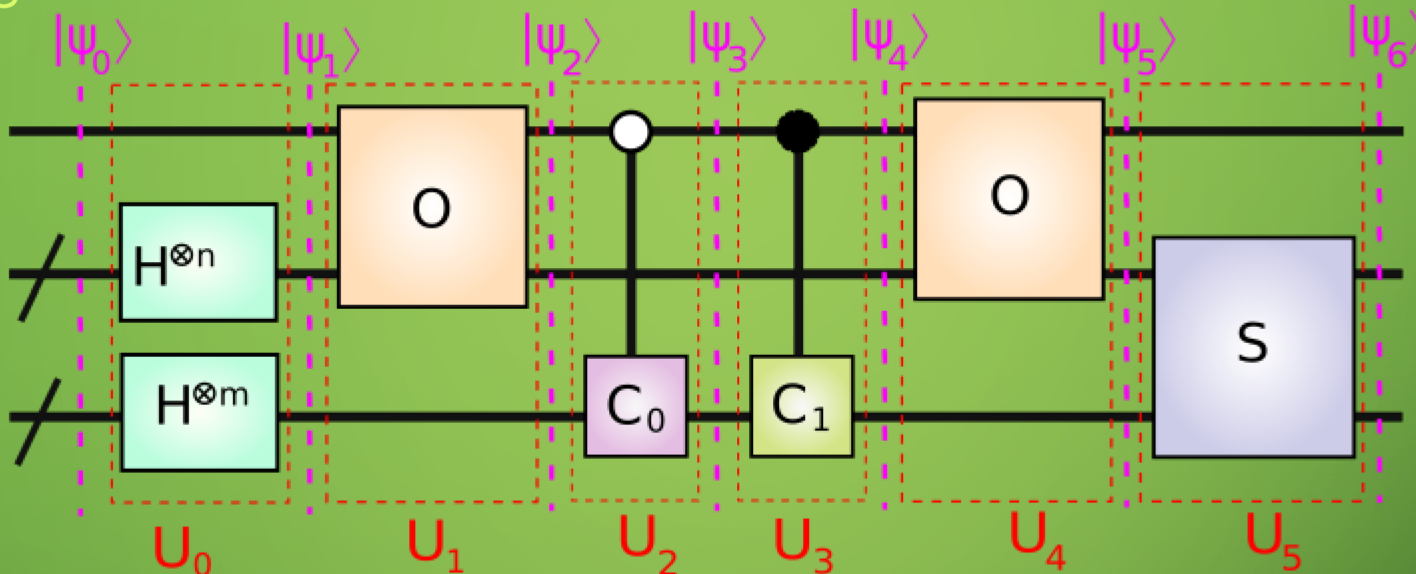
2. The probability to find searched element for QRWS is periodic function of number of iterations.




Probability p for hypercube with 256 vertices after $k_{itr}$ iterations.

When the node register state is measured, if the result is the searched element algorithm ends, otherwise it is repeated.

# QUANTUM CIRCUIT COMPONENTS




$$|\psi_k\rangle = |con_k, x_k, c_k\rangle = |con_k\rangle \otimes |\, x_k\rangle \otimes |c_k\rangle$$

Dim[$|con_k\rangle$] = 2 Dim[$|x_k\rangle$] = 2^m Dim[$|c_k\rangle$] = m

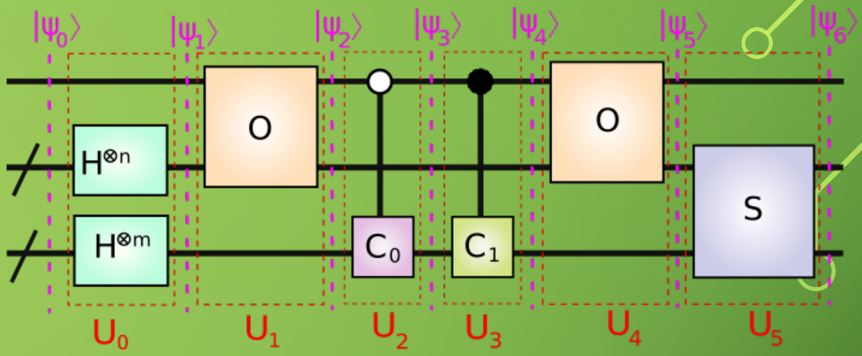
$$|\psi_{k+1}\rangle = U_k|\psi_k\rangle \quad \xrightarrow{\text{yields}} \quad |\psi_6\rangle = U_5U_4U_3U_2U_1U_0|\psi_0\rangle$$

Initial State:

$$|\psi_0\rangle = |0,0,0\rangle$$

$$U_0 = \begin{cases} \hat{I}_2 \otimes \mathrm{H}^{\otimes n} \otimes \mathrm{H}^{\otimes m} & \textit{works on when coin is power of } 2 \\ \hat{I}_2 \otimes \mathrm{F}_{2\wedge m} \otimes \mathrm{F}_m & \textit{works with arbitrary coin size} \end{cases}$$

## 1) Applying Hadamard Gates:




$$|\psi_0\rangle = |0,0,0\rangle$$

$$w = e^{-2\pi i/m}$$

$$F = \frac{1}{\sqrt{m}}\begin{pmatrix} w^{0*0} & w^{0*1} & \cdots & w^{0*(m-1)} \\ w^{1*0} & w^{1*1} & \cdots & w^{1*(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ w^{(m-1)*0} & w^{(m-1)*1} & \cdots & w^{(m-1)*(m-1)} \end{pmatrix}$$

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\hat{I}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$|\psi_1\rangle = U_0|\psi_0\rangle$$

$$|\psi_1\rangle = \left|0, \frac{1}{\sqrt{2^m}}\sum_{j=0}^{2^m-1}|j\rangle, \frac{1}{\sqrt{m}}\sum_{j=0}^{m-1}|j\rangle\right\rangle = \frac{1}{\sqrt{d}}|0\rangle \otimes \sum_{j=0}^{m2^m-1}|j\rangle$$

## 2) Applying First Oracle

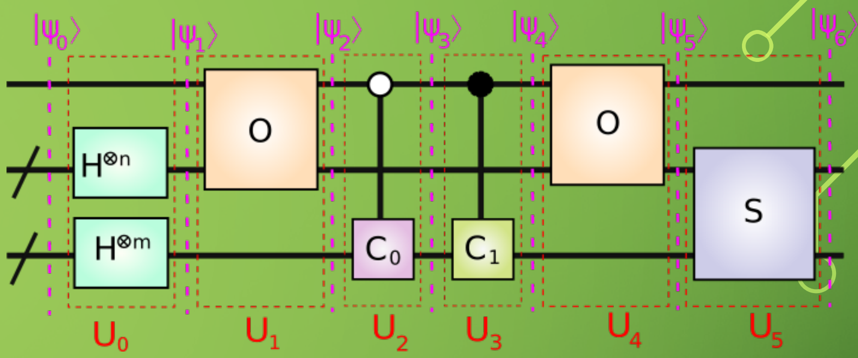
The Oracle marks all solutions, if solutions are $\{h_1, \ldots, h_\lambda\}$:

$$\hat{O} = \hat{I}_{2^{m+1}} - \sum_{i=1}^{\lambda}(|h_i\rangle\langle h_i| + |h_i + 2^m\rangle\langle h_i + 2^m|)$$
$$+ \sum_{i=1}^{\lambda}(|h_i + 2^m\rangle\langle h_i| + |h_i\rangle\langle h_i + 2^m|)$$

$$|\psi_1\rangle = \frac{1}{\sqrt{d}}(0{,}1) \otimes \sum_{j=0}^{m2^m-1}|j\rangle$$

$$U_1 = \hat{O} \otimes \hat{I}_m$$

$$|\psi_2\rangle = U_1|\psi_1\rangle$$




$$|\psi_2\rangle = \frac{1}{\sqrt{m2^m}}\Big((0{,}1) \otimes \ \big(\textstyle\sum_{j=0}^{d*2^\wedge d-1}|j\rangle - \sum_{i=1}^{\lambda}|h_i\rangle\big) + (1{,}0) \otimes \ \big(\sum_{i=1}^{\lambda}|h_i\rangle\big)\Big) \otimes \big(\sum_{j=0}^{m-1}|j\rangle\big)$$

## 3) Applying Traversing Coin

$$U_2 \ = \begin{pmatrix} \hat{I}_{m2^m} & \hat{0}_{m2^m} \\ \hat{0}_{m2^m} & \hat{I}_{2^m} \otimes C_0 \end{pmatrix}$$

$$|\psi_3\rangle = U_2|\psi_2\rangle$$

$$C_0(\phi, \chi, \zeta) = e^{i\zeta}\big(I - (1 - e^{i\phi})|\chi\rangle\langle\chi|\big)$$

$$|\chi\rangle = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle$$

$$\zeta = -2\,\phi + 3\pi + \alpha\sin(2\phi)$$

## 4) Applying Marking Coin

$$U_3 \ = \begin{pmatrix} \hat{I}_{2^m} \otimes C_1 & \hat{0}_{m2^m} \\ \hat{0}_{m2^m} & \hat{I}_{m2^m} \end{pmatrix}$$

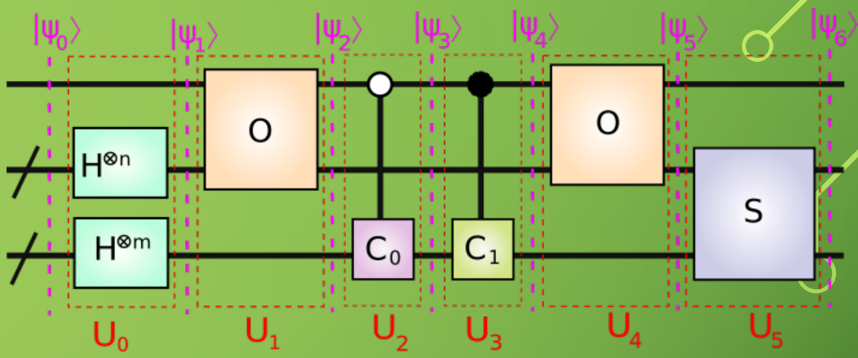
$$|\psi_3\rangle = U_2|\psi_2\rangle$$

$$C_1 \ = -\hat{I}_2$$

## 5) Applying Second Oracle

$U_4 = U_1$

$|\psi_5\rangle = U_4|\psi_4\rangle$




## 6) Applying Shift Operator

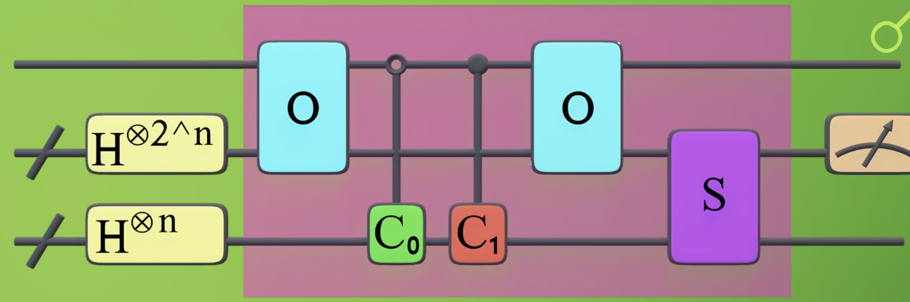
$$S = \sum_{d=0}^{m-1} \sum_{x=0}^{2^m-1} |x^d, d\rangle \langle x, d|$$

Where $x^d$ is the vector $x$ with $d^{\text{th}}$ bit flipped

$U_5 = \hat{I}_2 \otimes S$

$|\psi_6\rangle = U_5|\psi_5\rangle$

## 7) Measurement of the node register




$$\rho_{\psi_k} = |\psi_k\rangle\langle\psi_k| = |con_k, x_k, c_k\rangle\langle con_k, x_k, c_k|$$

If $\rho_{\psi_k}$ is separatable:

$$\rho_{\psi_k} = |con_k, x_k, c_k\rangle\langle con_k, x_k, c_k|$$
$$= |con_k\rangle\langle con_k| \otimes |x_k\rangle\langle x_k| \otimes |c_k\rangle\langle c_k| = \rho_{con_k} \otimes \rho_{x_k} \otimes \rho_{c_k}$$

$$Tr_{\rho_{con_k}} Tr_{\rho_{c_k}} [\rho_{\psi_k}] = \sum_i \langle i|\rho_{con_k}|i\rangle \otimes \rho_{x_k} \otimes \sum_j \langle j|\rho_{c_k}|j\rangle = \rho_{x_k}$$

When $\rho_{\psi_k}$ is not separatable:

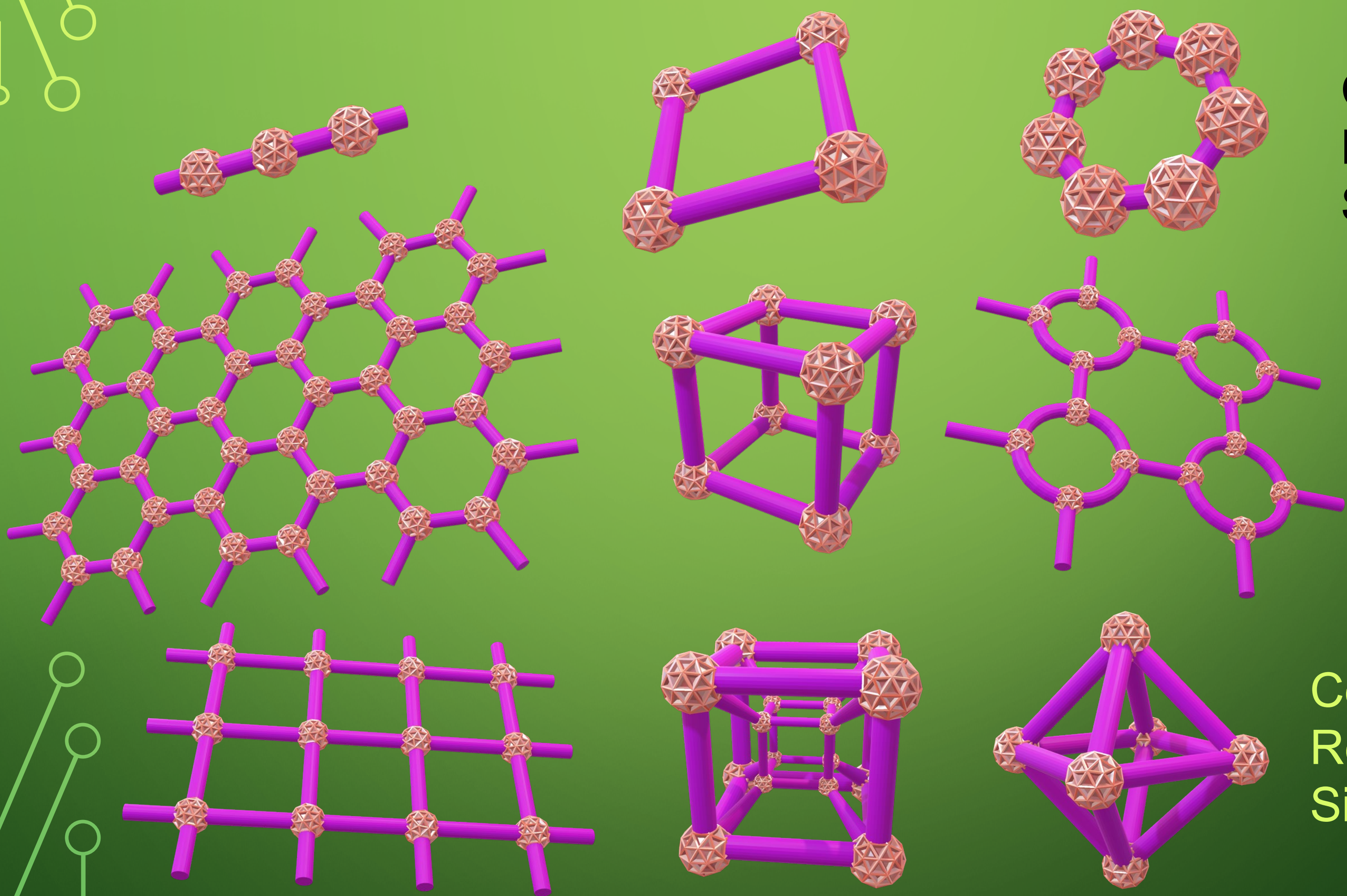
$$Tr_{\rho_{con_k}}[\rho_{\psi_k}] = \sum_j \left(\langle j|_{\rho_{con_k}} \otimes \hat{I}_{\rho_{x_k} \otimes \rho_{c_k}}\right) \rho_{\psi_k} \left(|j\rangle_{\rho_{con_k}} \otimes \hat{I}_{\rho_{x_k} \otimes \rho_{c_k}}\right)$$

$$[\rho_{x_k}] = Tr_{\rho_{c_k}}\left[Tr_{\rho_{con_k}}[\rho_{\psi_k}]\right]$$
$$= \sum_j \left(\hat{I}_{\rho_{x_k}} \otimes \langle j|_{\rho_{c_k}}\right) Tr_{\rho_{con_k}}[\rho_{\psi_k}] \left(\hat{I}_{\rho_{x_k}} \otimes |j\rangle_{\rho_{c_k}}\right)$$

М$[\rho_{x_k}]$ - измерване на регистъра на върховете

# Shift Operator

Shift operator S defines the topology of the walked object



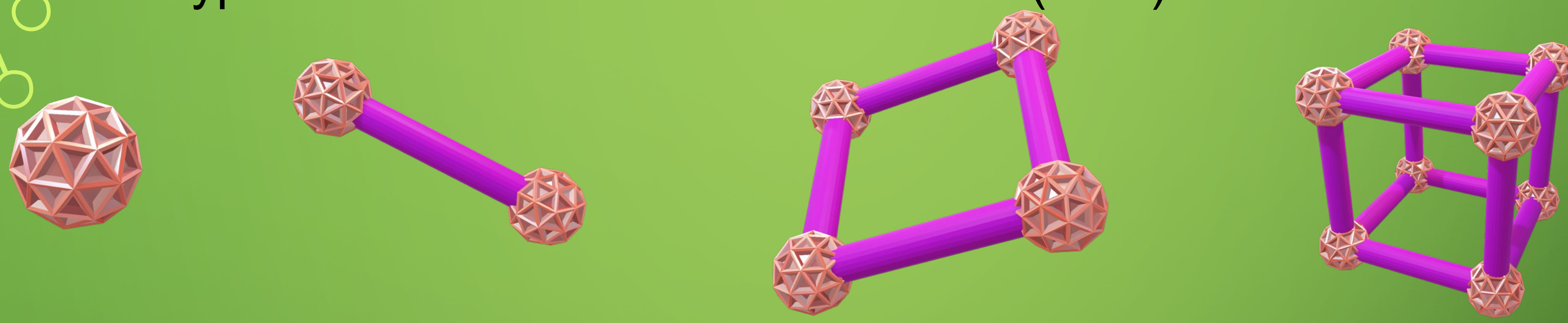
Coin Register Size is 2

Coin Register Size is 3

Coin Register Size is 4

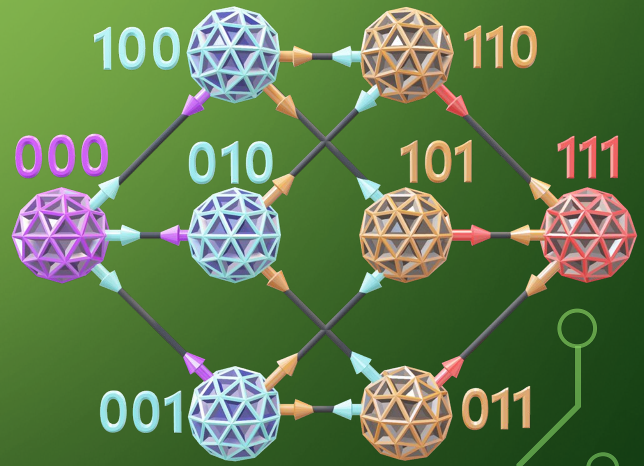
# Hypercube and Node Numbering

Hypercubes with different dimensions (0 - 3):



Number of nodes and edges of such Hypercube are:

$E_{0,d} = 2^d$ $E_{1,d} = R2^{d-1}$



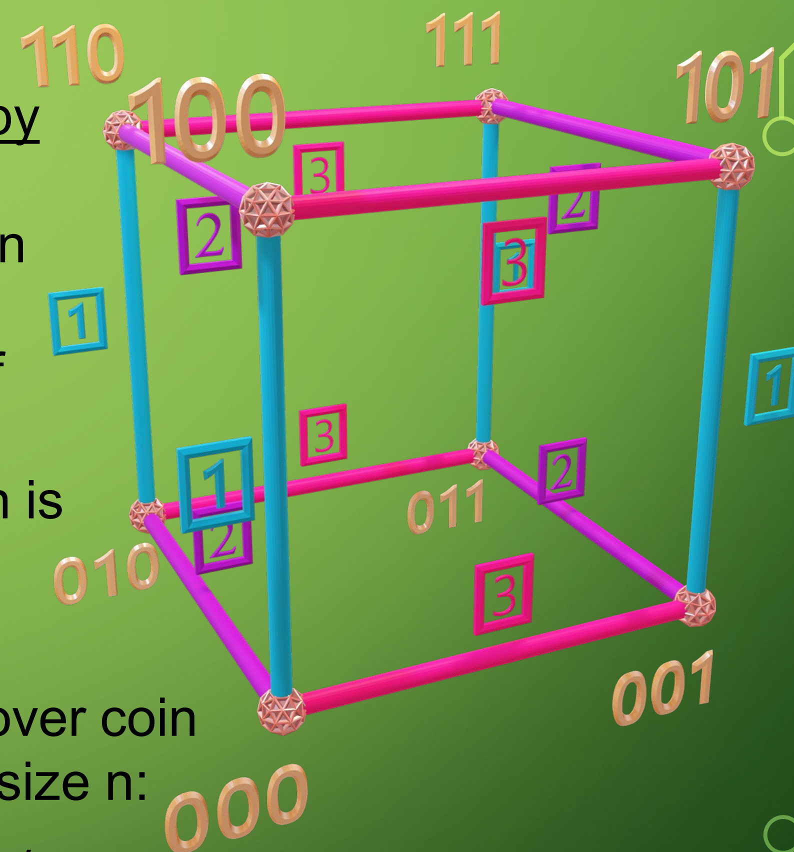

Each node (and also edges) can be numbered with binary string label. Zeroth node can be arbitrary chosen.

Two nodes in a hypercube are neighbors, if they differ by only one symbol (their Hamming distance is 1).

# Walk Coin

<u>Walk coin gives probabilities for transition between nodes connected by an edge</u>.

- The system can be in superposition of nodes, so during the evolution it can go to different superposition of states.
- If probability to go in each direction is the same, then off diagonal matrix elements should be the same.




Original QRWS algorithm uses Grover coin G for traversing a graph. For coin with size n:

$$G = \begin{pmatrix} -1+2/n & 2/n & \dots & 2/n \\ 2/n & -1+2/n & \dots & 2/n \\ \vdots & \vdots & \ddots & \vdots \\ 2/n & 2/n & \dots & -1+2/n \end{pmatrix}$$

# ROBUSTNESS OF QRWS WITH MODIFIED COIN

# Modification of the walk coin

We study the following walk coin:

$$C_0(\phi, \chi, \zeta) = \underbrace{e^{i\zeta}}_{Phase\ gate} \times \underbrace{\left(I - (1 - e^{i\phi})|\chi\rangle\langle\chi|\right)}_{Generalized\ Householder\ reflection}$$

Both Generalized Householder reflection and phase gate can be done efficiently in some physical Quantum circuit implementations like the ion traps.

To have equal probability to go at each direction $\chi$ must be equal weight superposition of the basis vectors $|j\rangle$

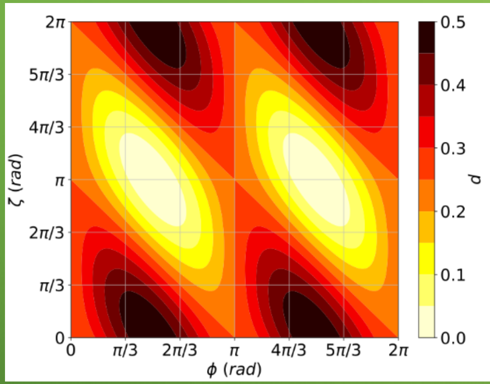
$$|\chi\rangle = \frac{1}{\sqrt{m_n}} \sum_{j=0}^{m_n - 1} |j\rangle$$

The probability to find solution $p = p(\zeta, \phi, \mathrm{n})$ depends on $\zeta$, $\phi$ and coin register size n.

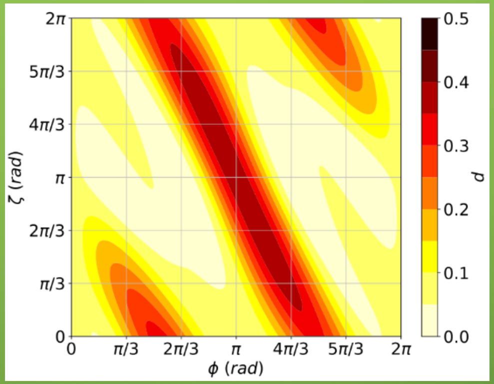
# Monte Carlo simulations of the algorithm

MC simulations of p($\zeta, \phi$) of QRWS for Hypercube
In each run n is fixed and for $\zeta, \phi \in [0, 2\pi)$ are taken random values

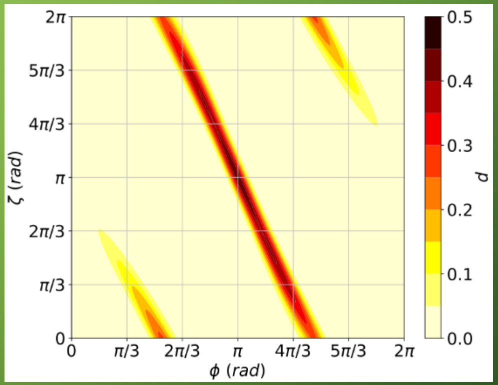
One qubit coin




Two qubit coin




Three qubit coin




There exist connected areas in $(\phi, \zeta)$ plane with high probability to find solution!

# Robustness of $p(\zeta,\phi)$

In order to make QRWS more robust to change in the phases, we search for areas in the plane defined by $(\phi,\zeta)$ that give high probability to find solution when one or both of the parameters vary:

$$p(\phi \in (\phi_{max} - \varepsilon, \phi_{max} + \varepsilon)) \cong p_{max} = p\ (\phi_{max})$$

In our case p can be expressed as function of just one of the phases:

$$\zeta = \zeta(\phi) \Longrightarrow p(\zeta(\phi), \phi\ , \mathrm{n} = \mathrm{const}) \longrightarrow p(\phi)$$

Different functions $\zeta(\phi)$ were fitted to MC data points, to find the one that makes the algorithm as robust as possible

Note: For Grover coin both phases $\phi$ and $\zeta$ are equal to $\pi$, and $|\chi\rangle$ is equal weight superposition

# Improvement of algorithm's stability

Examples for linear functions:

Best linear approximation: $\zeta = -2\,\phi + 3\pi$

$\zeta$ =const
$\zeta = \pi$

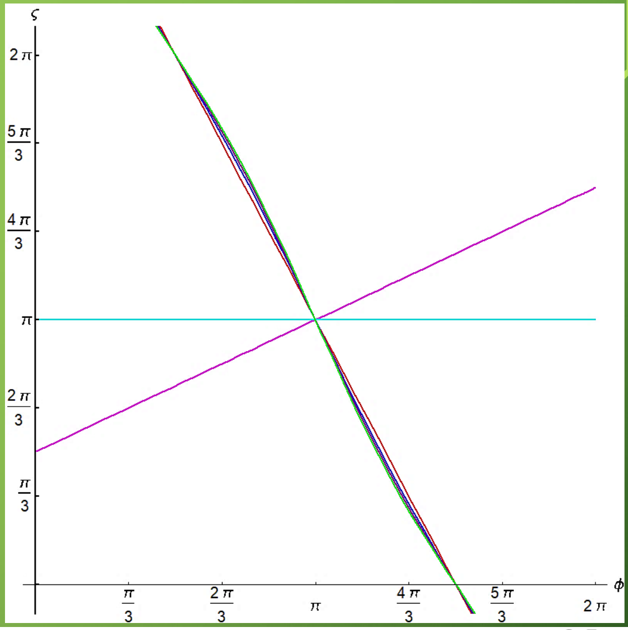
Nonlinear functions:

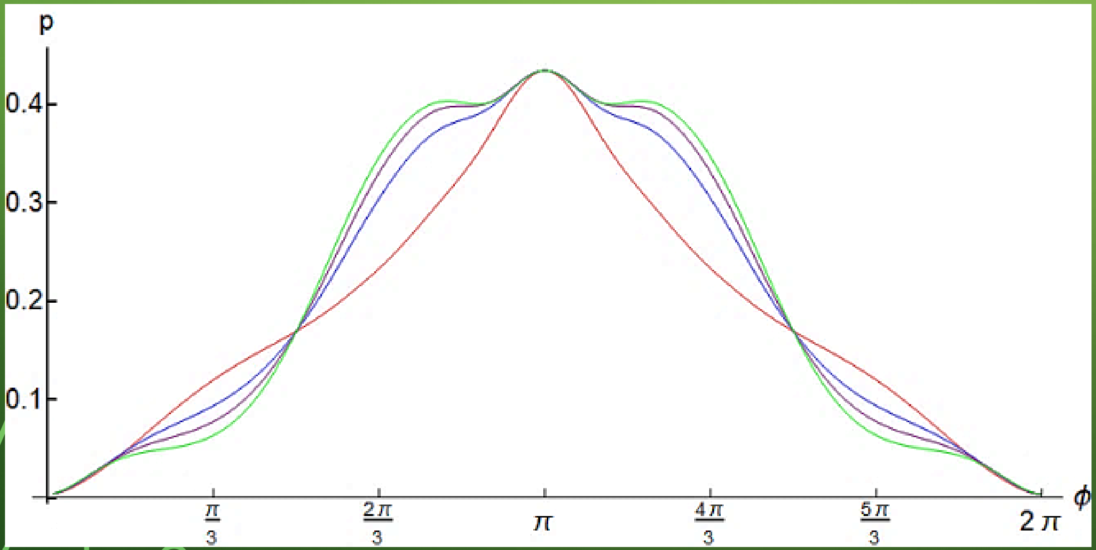
Almost linear:
$\zeta = -2\,\phi + 3\pi + \alpha\sin(2\phi)$

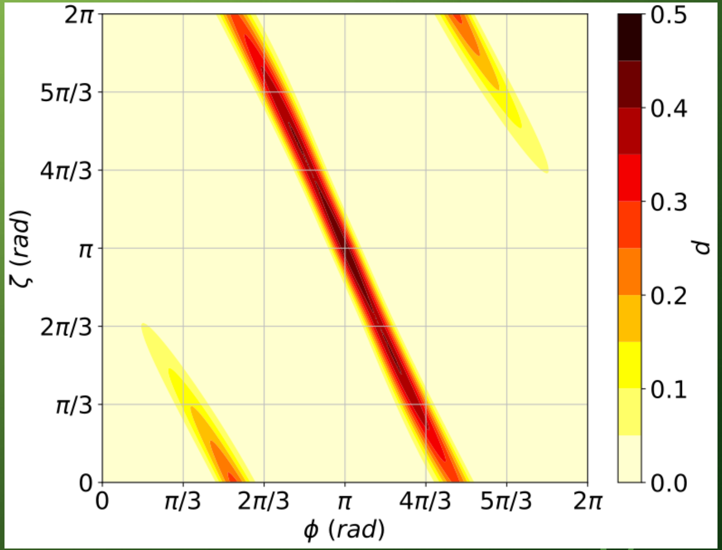
$\alpha = 0.204$

$\alpha = 1/(2\pi)$

$\alpha = 1/(3\pi)$







*Coin size* $n = 3$

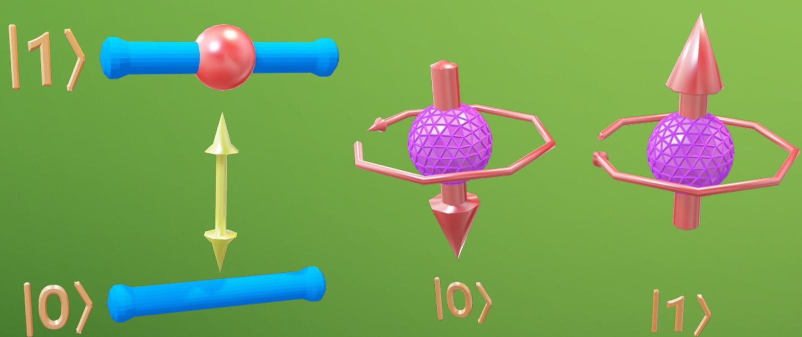
# ROBUSTNESS OF QRWS WITH QUDIT COIN

# Qubits vs Qudits

One qubit can be any two level quantum system:

1. Levels of electron in ions;
2. Spins of quantum dots;
3. Others.

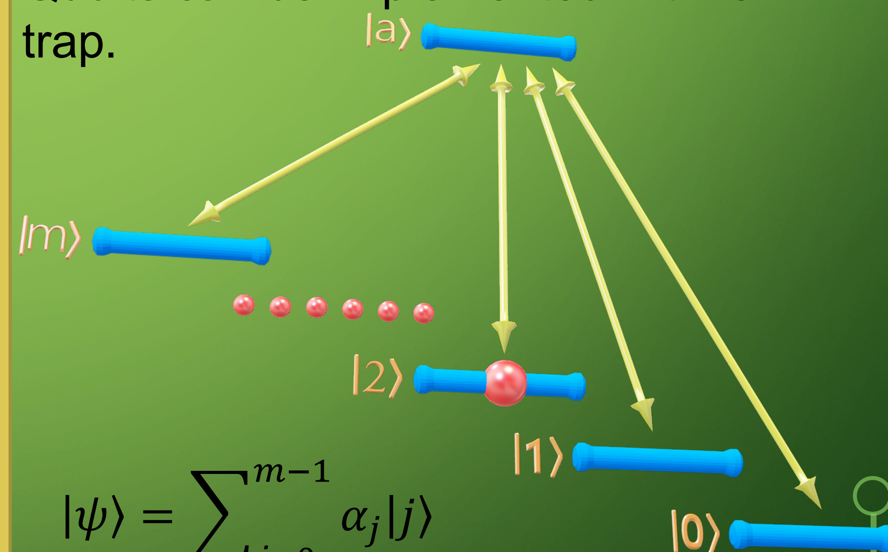
Qubit states $|\psi\rangle = \alpha_0\ |0\rangle + \alpha_1\ |1\rangle$




Qubit coin register can have only power of two number of states – 2; 4; 8; 16; 32…

$m = 2^k$ k= $Integer$

Qudits can be implemented by using any system with d levels, e.g. hyperfine states of one split level due to external electric or magnetic field. All those levels should be metastable. Qudits can be implemented with ion trap.




$$|\psi\rangle = \sum_{j=0}^{m-1} \alpha_j |j\rangle$$

Often transitions between levels are made by using ancilla state.

$m = Integer$

# Advantages of using Qudits

Using qudits instead of qubits gives various advantages for the quantum algorithms:

- They are more robust against noise and give more dependable quantum computations;
- The coin can have arbitrary dimension not only power of 2;
  - Allow us to make much more reliable extrapolations for quantum random walk search algorithm’s stability for larger coin sizes;
- Increasing the size of the coin state space;
- More efficient construction of various quantum gates;
- New quantum error correction protocols.

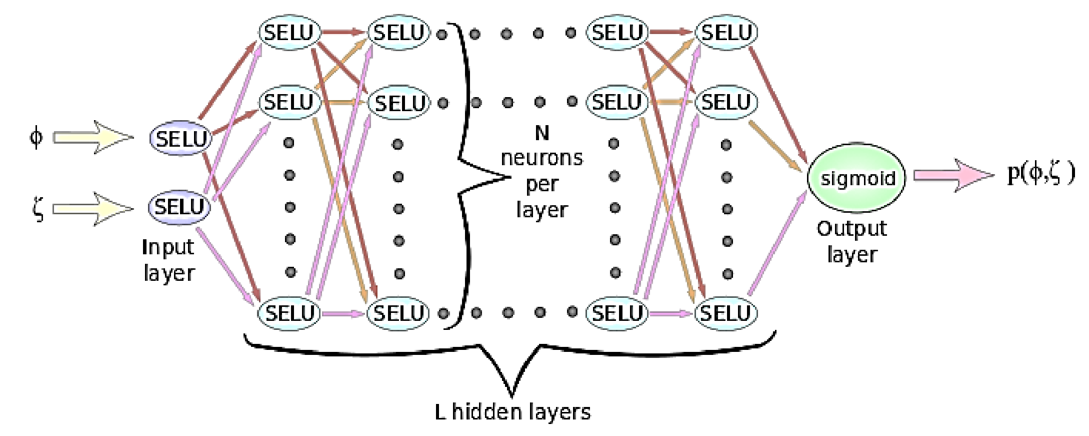
# Conclusion

- The discrete time quantum random walk search is quantum algorithm able to search in unordered database with arbitrary topology. It is quadratically faster than the corresponding classical search algorithms;
- A modification of the algorithm with walk coin constructed by Generalized Householder reflections and a phase gate could be made extremely robust to deviations in the coin parameters if a proper relations between the parameters is maintained;
- Using qudits for walk coin register give the possibility to increase even more algorithm's stability;

# THANK YOU FOR YOUR ATTENTION!

This work was supported by the Bulgarian Science Fund
under contract KP-06-M48/2 /26.11.2020.

# Machine learning and optimization




This neural network is used to fit training examples to obtain a ML model that best approximate quantum random walk search algorithm. This model is used to optimize the quantum algorithm

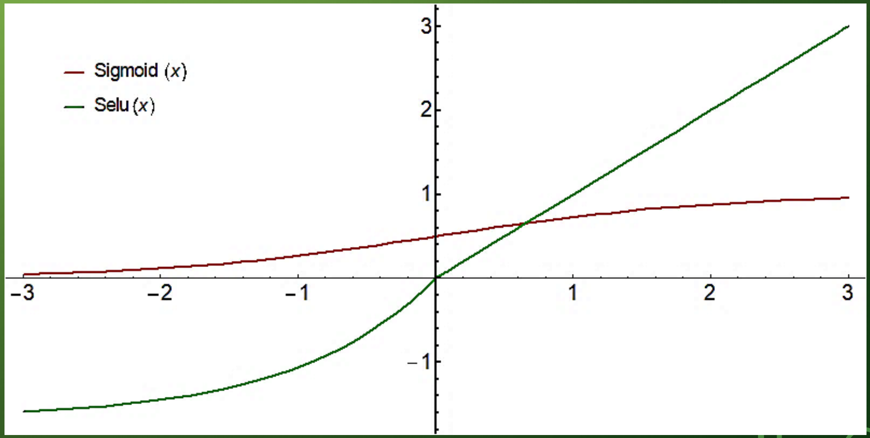
Feed Forward NN – network were information flows from k-th to (k+1)-th layer. No information flows between neurons on the same layer or from (k+1)-th layer to the k-th.

Activation Function of neuron – non-linear function, witch result depends on the input

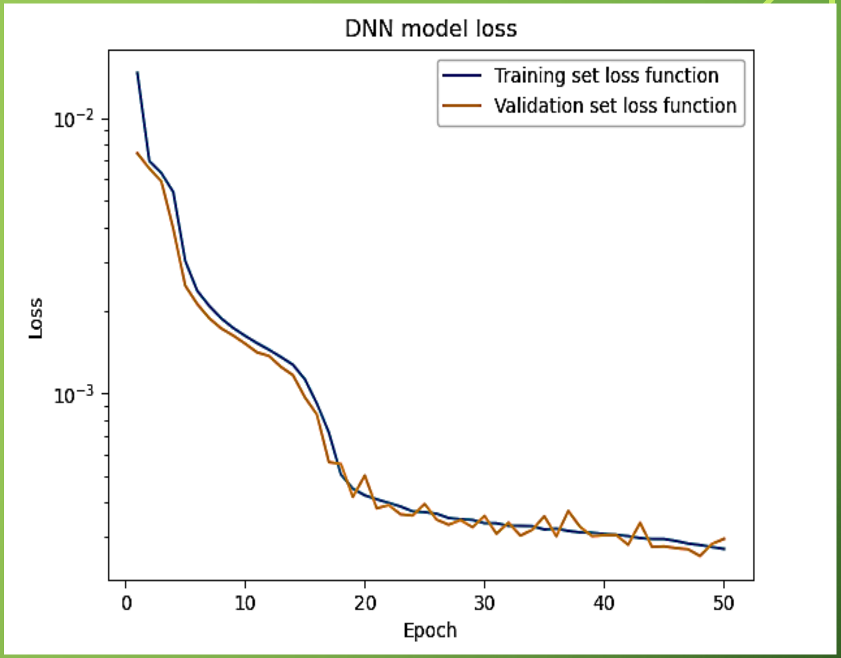
$$SELU(x) = \begin{cases} x & if\ x > 0 \\ 1.673(e^x - 1) & if\ x \leq 0 \end{cases}$$

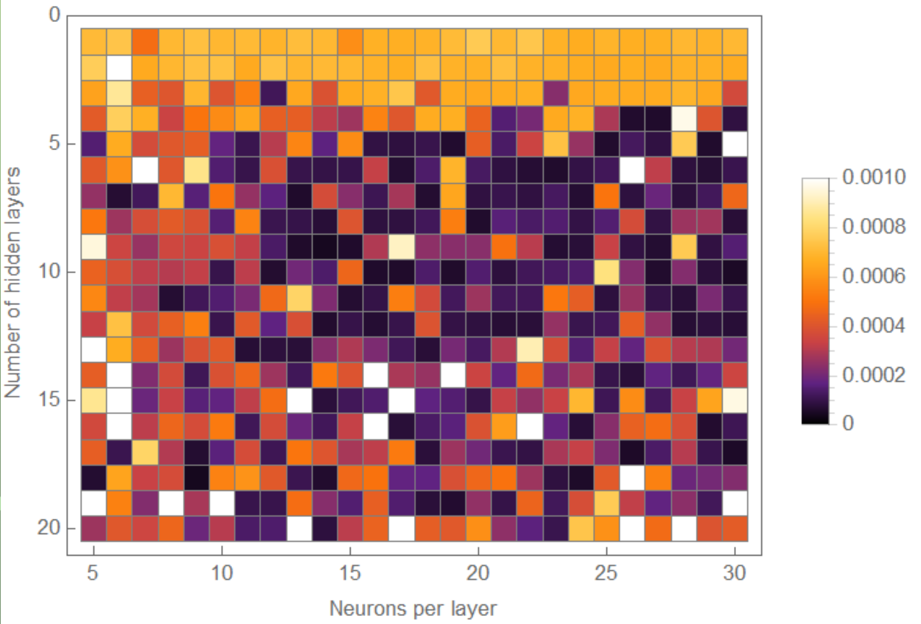
$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Gradient descent is an iterative optimization algorithm for finding a local minimum of a function by making steps at direction of the steepest descent.



Epoch – one run of the neuron network trough training examples. Network update its parameters at end of each epoch.
Batch Gradient Descent – At each epoch, lost function is calculated by smaller portion of training examples (batch). This batch is taken by random training examples. Therefore there is larger uncertainty at the end of the training, lost function oscillate around the minimum.




Training Set – set used to train NN
Validation Set – set used to evaluate NN
Lost function – measure how well ML model fit training examples (depend of network parameters)
Early stopping – stop training of NN when LF on VS starts to worsen trough the epochs
On the left figure darker colors correspond to smaller lost function => better ML model
Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of expected value and variance

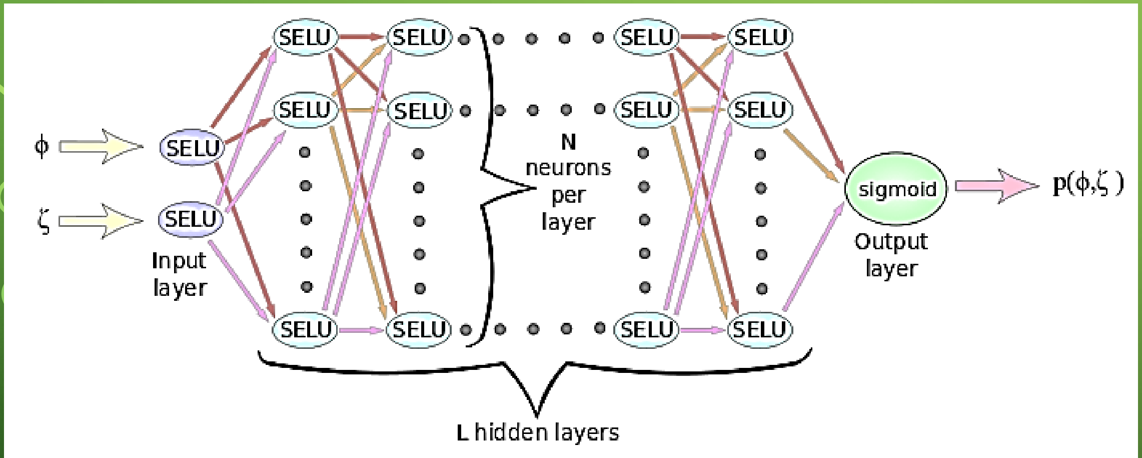
# Optimizing walk coin by machine learning

Different functions were fitted to data points, to find the function that makes the algorithm as robust as possible. So for largest possible $\varepsilon$ to be fulfilled:

$$p(\phi \in (\phi_{max} - \varepsilon, \phi_{max} + \varepsilon)) \cong p_{max} = p\ (\phi_{max})$$

Best results were obtained with the function: $\zeta = -2\ \phi + 3\pi + \alpha \sin(2\phi)$ where $\phi, \zeta\ \epsilon [0,2\pi]$

For finding the best value of $\alpha$ a feed forward Neural Network is used.




N and L are varied to find the best model. The model is used to fit the above function to the points in the $\zeta, \phi$ plane with highest $p(\zeta, \phi)$. Then extract the corresponding value of $\alpha$

Neural network Parameters:

1) $N \in [1,20]$
2) $L \in [5,30]$
3) Training Examples 300000 for one and two qubits and 15000 for 3-qubits
4) Batch size is 256 examples
5) Early stopping is used.
6) Training set 80%
7) Validation set 20%
8) Adam optimization