Lecture 4:
Quantum Tree Search

Andreas Wichert
Department of Computer Science and Engineering
Técnico Lisboa

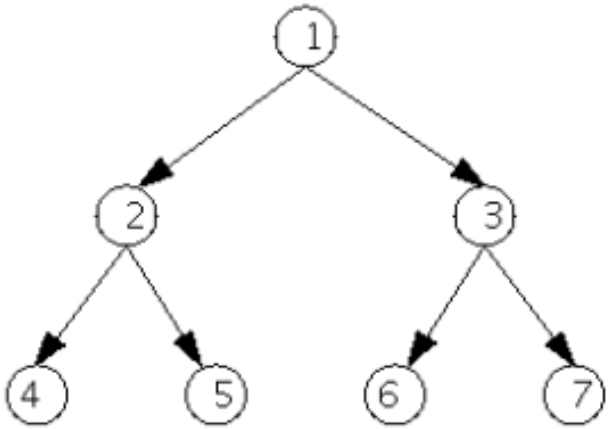
Overview

- Tree Search
- Quantum Tree Search
  - Iterative Deepening
- Pure Production System
- Quantum Production Systems
- 3-Puzzle
  - Rules and Trace
- 8 Puzzle
  - Branching Factor
- Blocks World
- Games and Quantum Tree Search

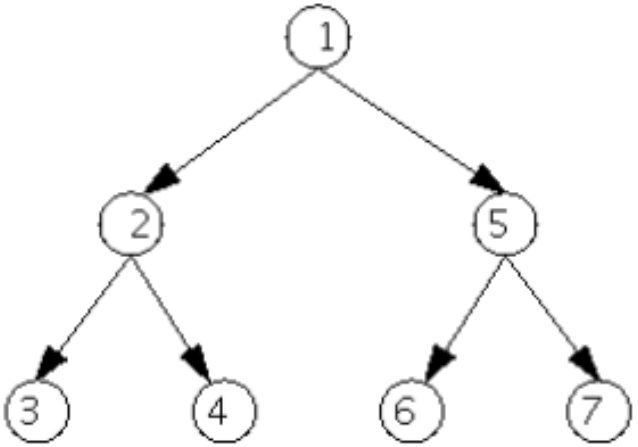
Tree Search

- Nodes and edges represent a search tree.
- Each node represents a state, and each edge represents a transition from one state to the following state.
- The initial state defines the root of the tree.
- From each state, either $B \in \boldsymbol{N}$ states can be reached, or the state is a leaf.
- From a leaf, no other state can be reached.
- B represents the branching factor of the node, the number of possible choices.
- A leaf represents either the goal of the computation or an impasse when there is no valid transition to a succeeding state.
- Every node besides the root has a unique node from which it was reached, which is called the parent.

Blind Search



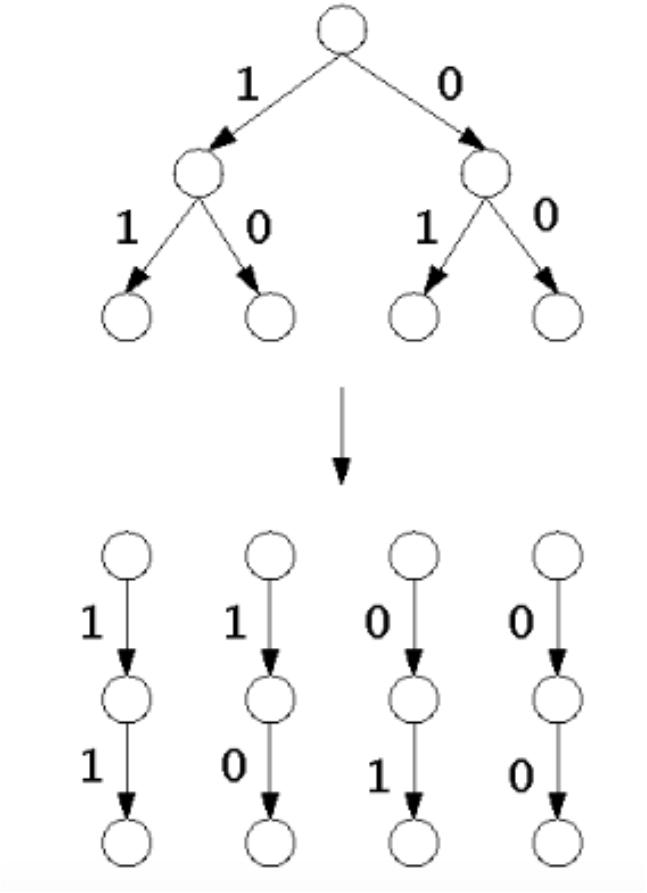
Breadth-first search



Depth-first search

Path Descriptors for *B=2*

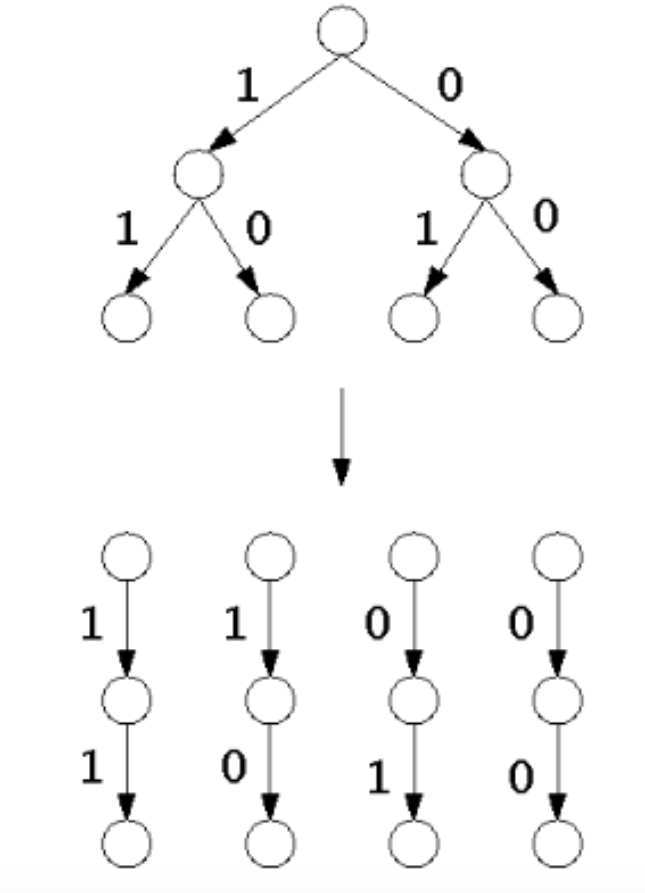
- Each parent has *B* children
- If *B* = *2*, each of the *m* questions has a reply of either “yes” or “no” and can be represented by a bit.
- The m answers are represented by a binary number of length m. There are $n = 2^m = B^m$ possible binary numbers of length *m*.



- Each binary number represents a path from the root to a leaf. For each goal, a certain binary number indicates the solution
- For a constant branching factor B > 2, each question has B possible answers
- The *m* answers can be represented by *m* digits
  - For example, with $B = 8$, the number is represented by 2^3 bits. These numbers represent all paths from the root to the leaves
- These numbers represent all paths from the root to the leaves

Quantum Tree Search

- In a quantum computation, we can simultaneously represent all possible path descriptors.
- There is one path descriptor for each leaf of the tree.
- Using Grover's algorithm, we search through all possible paths and verify whether each path leads to the goal state.
- This type of procedure is called a quantum tree search



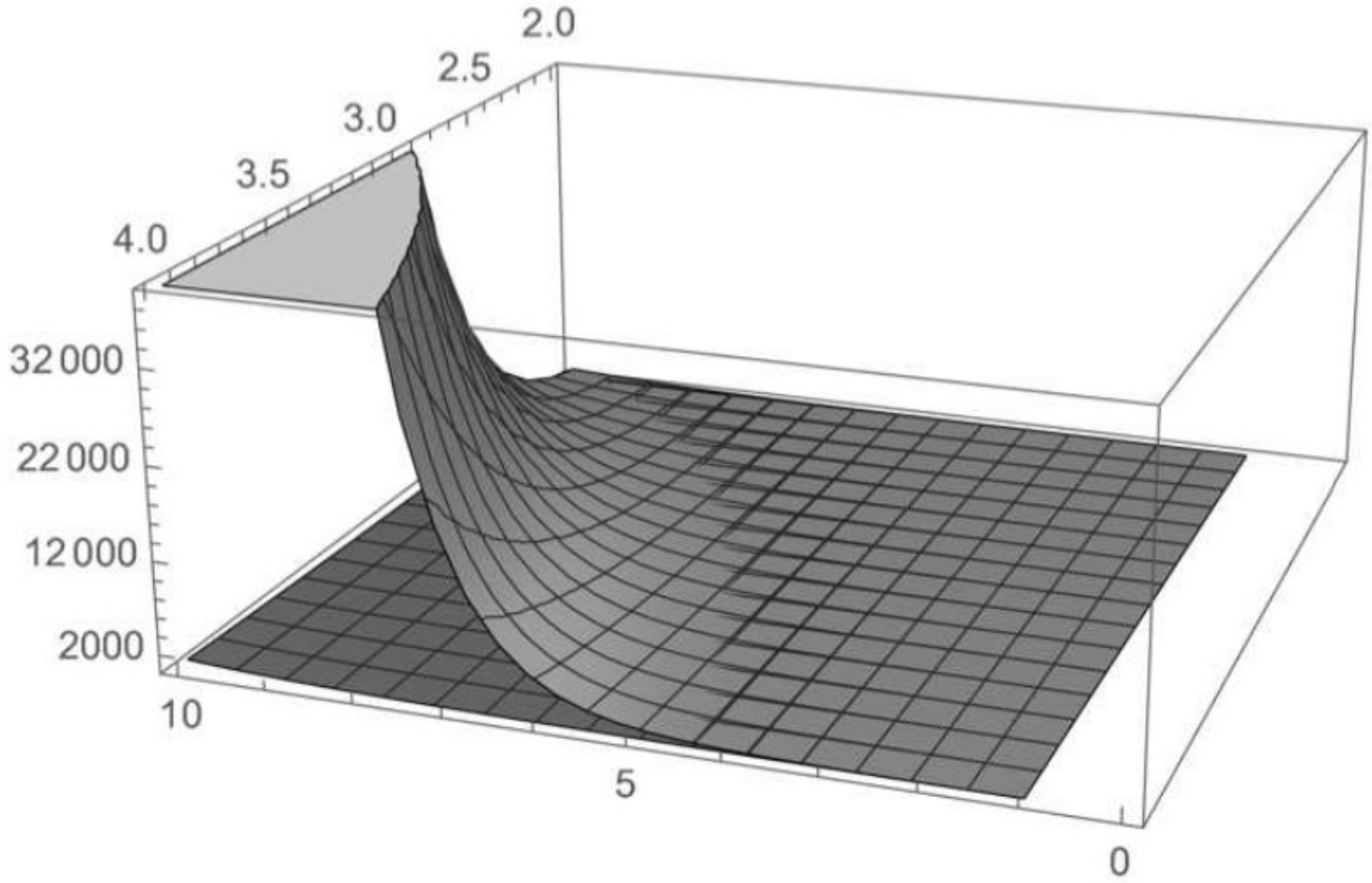


Figure 7.2 For branching factor B from 2 to 4 and the depth of the tree search m from 1 to 10. The cost on a conventional computer are $n = B^m$, upper plane. On a quantum computer wee need only $\sqrt{n} = B^{\frac{m}{2}}$ steps, plane below.

Iterative Deepening

- For $n = B^m$ possible paths, the costs are (approximately) $\sqrt{n} = B^{m/2}$
  A constraint of this approach is that we must know the depth m of the search tree in advance
- The constraint can be overcome by iterative deepening
- During the iterative deepening search, the states are generated multiple times
- The time complexity of the iterative deepening search is of the same order of magnitude as a search to the maximum depth

Iterative Deepening

- Richard E. Korf:
- *“Since the number of nodes on a given level of the tree grows exponentially with depth, almost all time is spent in the deepest level, even though shallower levels are generated an arithmetically increasing number of times”*

Professor of Computer Science, University of California, Los Angeles

The number of nodes for iterative deepening for each level starting with level zero (for simplicity) is given by

$$1$$

$$1 + B$$

$$1 + B + B^2$$

$$\dots$$

to level m is given by

$$1 + B + B^2 + B^3 + \cdots + B^m$$

the costs C_m (total number of visited nodes) are

$$C_m = (m+1) \cdot 1 + m \cdot B + (m-1) \cdot B^2 + (m-2) \cdot B^3 + \cdots + 1 \cdot B^m,$$

$$C_m = \sum_{i=0}^{m} (m + 1 - i) \cdot B^i.$$

With

$$C_m = \sum_{i=1}^{m+1} (m + 1 - (i - 1)) \cdot B^{i-1}.$$

According to Riley[Riley *et al.* (2006)]

$$C_m - C_m \cdot B = C_m \cdot (1 - B) = m + 1 - (2 \cdot m + 1) \cdot B^{m+1} + \frac{B \cdot (1 - B^m)}{(1 - B)},$$

$$C_m = \frac{m + 1}{1 - B} - \frac{(2 \cdot m + 1) \cdot B^{m+1}}{1 - B} + \frac{B \cdot (1 - B^m)}{(1 - B)^2},$$

by simplifying this equation we arrive at

$$C_m = \frac{1 + m - B \cdot m - 2 \cdot B^{1+m} \cdot (1 + m) + B^{2+m} \cdot (1 + 2 \cdot m)}{(B - 1)^2} = O(B^m).$$

For a **quantum tree search**

starting with level zero (for simplicity) is given by

$$O(1)$$

$$O(\beta) = O(B^{\frac{1}{2}})$$

$$O(\beta^2) = O(B^{\frac{2}{2}})$$

$$\cdots$$

to level m is given by

$$O(\beta^m) = O(B^{\frac{m}{2}})$$

the total costs of m_k iterations with m measurments are $O(B^{\frac{m}{2}}) = O\left(\sqrt{B^m}\right)$

$$O(1) + O(B^{\frac{1}{2}}) + O(B^{\frac{2}{2}}) + O(B^{\frac{3}{2}}) + \cdots + O(B^{\frac{m}{2}}) = O(B^{\frac{m}{2}}) = O\left(\sqrt{B^m}\right) \tag{11.22}$$

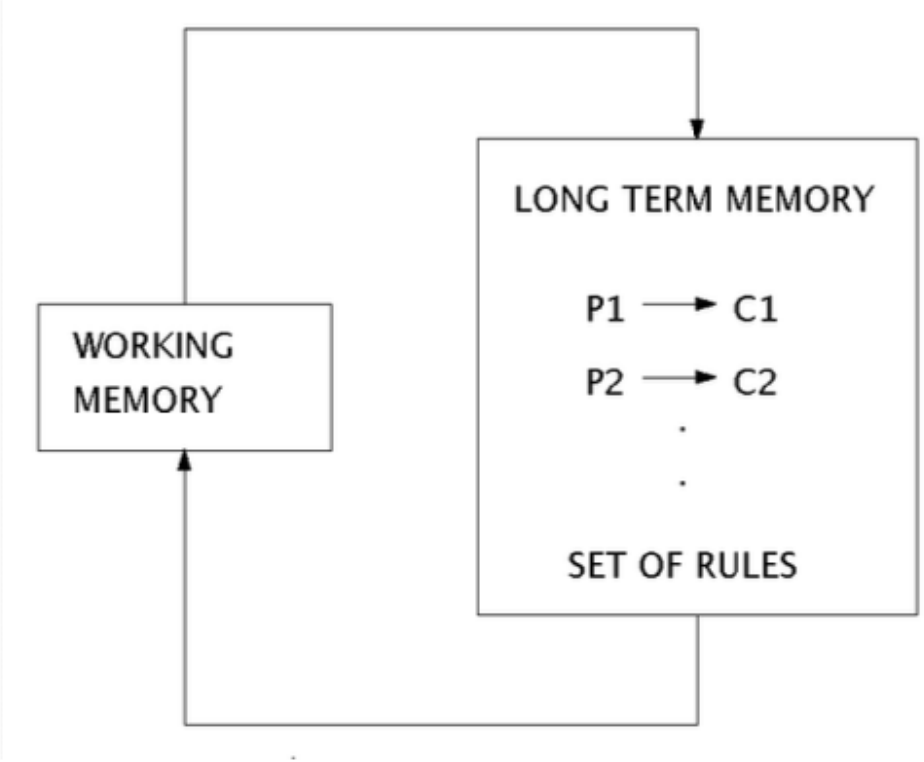
$$O\left(\sqrt{B^m}\right) = O\left(\sqrt{2^\mu}\right) = O\left(\sqrt{2^{m \cdot \lceil \log_2 B \rceil}}\right)$$

the equation is based on the geometric series

$$1+\beta+\beta^2+\beta^3+\cdots+\beta^m = \sum_{i=0}^{m} \beta^i = \frac{1-\beta^{m+1}}{1-\beta} = O(\beta^m) = O(B^{\frac{m}{2}}) \tag{11.23}$$

- A second constraint is represented by the constant branching factor.
- If the branching factor is not constant, the maximal branching factor B_{max} must be used for the quantum tree search
- For quantum tree search for a large number of instances (different initial and goal states) the effective branching factor converge to the averaged branching factor for uninformed tree search
  - This is due to equal solutions with different path descriptor (see later)

Production Systems



Pure Production System

- The pure production system model has no mechanism for recovering from an impasse .
- The system halts if no production can fire. It is composed of the set of productions *L* (the long-term memory) and control system *C*
- A pure production system is a sextuple:

$$(\Sigma, L, W, \gamma_i, \gamma_g, C)$$

$$(\Sigma, L, W, \gamma_i, \gamma_g, C)$$

with

- Σ is a finite alphabet;
- W is the working memory. It represents a state $\gamma \in \Sigma$.
- L is the long term memory. It is the set of B productions. A production p has the form $(precondition, conclusion) \in \Sigma$. The precondition is matched against the contents of the working memory. If the precondition is met then the conclusion is preformed and changes the contents of the working memory;
- $\gamma_i \in \Sigma$ is the initial state. The working memory is initialized with the initial state γ_i;
- $\gamma_g \in \Sigma$ is the goal state;
- δ is the control function of the form $\Sigma \rightarrow L \times \Sigma \times h$. It chooses a production and fires it or halts h.

If $C(\gamma) = (p, \gamma', h)$, then the working memory contains symbol γ. It is substituted by a production p by γ' or the computation halts h. The computation halts if the goal state γ_g is reached or an impasse is present, means no production can be applied.

Turing Machine

Alan Turing (1912 - 1954)

- Production systems are closely related to the approach of Markov algorithms; similar to these approaches, production systems are equivalent in power to a Turing machine
- A Turing machine can also be easily simulated by a production system; thus, a production system is a complete model of computation.
- The search represented by a search tree is performed from an initial state through the following states until a goal state is attained.

Quantum Production Systems

- Quantum production systems are related to pure production systems since the computation is not continued in the branch if an impasse is present, no backtracking to a previous state is done.
- Contrary to the pure production systems no control system C exists since all productions are executed simultaneously.
- Quantum production system can operate independently of whether the computation terminates; in the case of non-termination, the computation continues forever, and the iterations do not terminate
- The quantum production system also provides a maximal speedup of $O(\sqrt{n})$ if the Turing machine simulation allows n multiple computational branches

Example: Sorting a String

- Σ: a, b, c, d composed of four letters
- W is the working memory. It represents a state $\gamma \in \Sigma$ by a string of five letters
- The long term memory L, the set of $B = 6$ productions

  1. $ab \rightarrow ba$
  2. $ac \rightarrow ca$
  3. $ad \rightarrow da$
  4. $bc \rightarrow cb$
  5. $bd \rightarrow db$
  6. $cd \rightarrow dc$

  The production (rule) can fire if the precondition matches portion of the string in the working memory.

Iteration	Working Memory	Conflict Set	Rule Fired
0	*cdcab*	1, 6	1
1	*cdcba*	6	6
2	*dccba*		Halt

- $\gamma_i \in \Sigma$ is the initial state: *cdcab*
- $\gamma_g \in \Sigma$ is the goal state: *dccba*
- δ is the control function: random choice of a production

Quantum Production System

- To port this simple pure production systems into the quantum production system we represent first the finite alphabet Σ by two qubits and approximate a problem by reformulation of the rules without a precondition that has to be tested
  - Σ: $a = |00\rangle$, $b = |01\rangle$, $c = |10\rangle$, $d = |11\rangle$
- W is the working memory. It represents a state of teen qubits $\gamma \in \Sigma$, two qubits represent a position p_i

$$|p_5 p_4 p_3 p_2 p_1\rangle$$

- index $|11\rangle$: $p_2 p_1 \rightarrow p_1 p_2$
- index $|10\rangle$: $p_3 p_2 \rightarrow p_2 p_3$
- index $|01\rangle$: $p_4 p_3 \rightarrow p_3 p_4$
- index $|00\rangle$: $p_5 p_4 \rightarrow p_4 p_5$

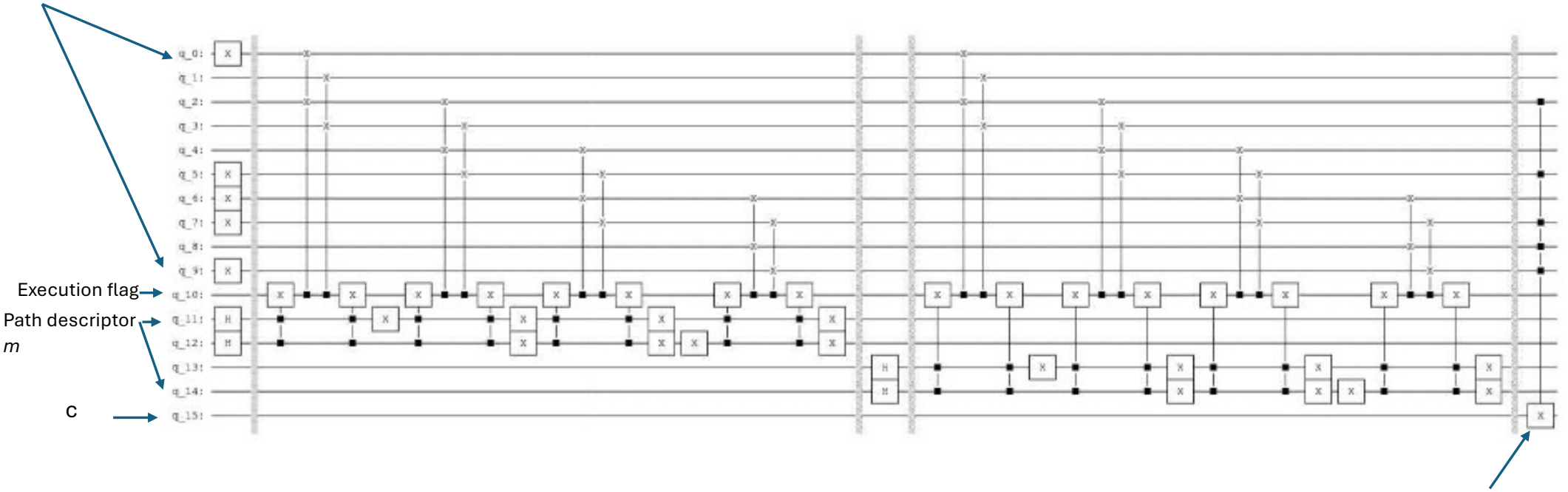
- Σ: $a = |00\rangle$, $b = |01\rangle$, $c = |10\rangle$, $d = |11\rangle$
- W is the working memory. It represents a state of teen qubits $\gamma \in \Sigma$, two qubits represent a position p_i

$$|x_7x_6x_5x_4x_3x_2x_1x_0\rangle = |p_5p_4p_3p_2p_1\rangle$$

- The long term memory L, the set of $B = 4$ productions that manipulate the position of the string, indexed by two qubits that represent the path descriptor. Each precondition determines the rule, each rule can be always applied
  - index $|11\rangle$: $p_2p_1 \rightarrow p_1p_2$
  - index $|10\rangle$: $p_3p_2 \rightarrow p_2p_3$
  - index $|01\rangle$: $p_4p_3 \rightarrow p_3p_4$
  - index $|00\rangle$: $p_5p_4 \rightarrow p_4p_5$

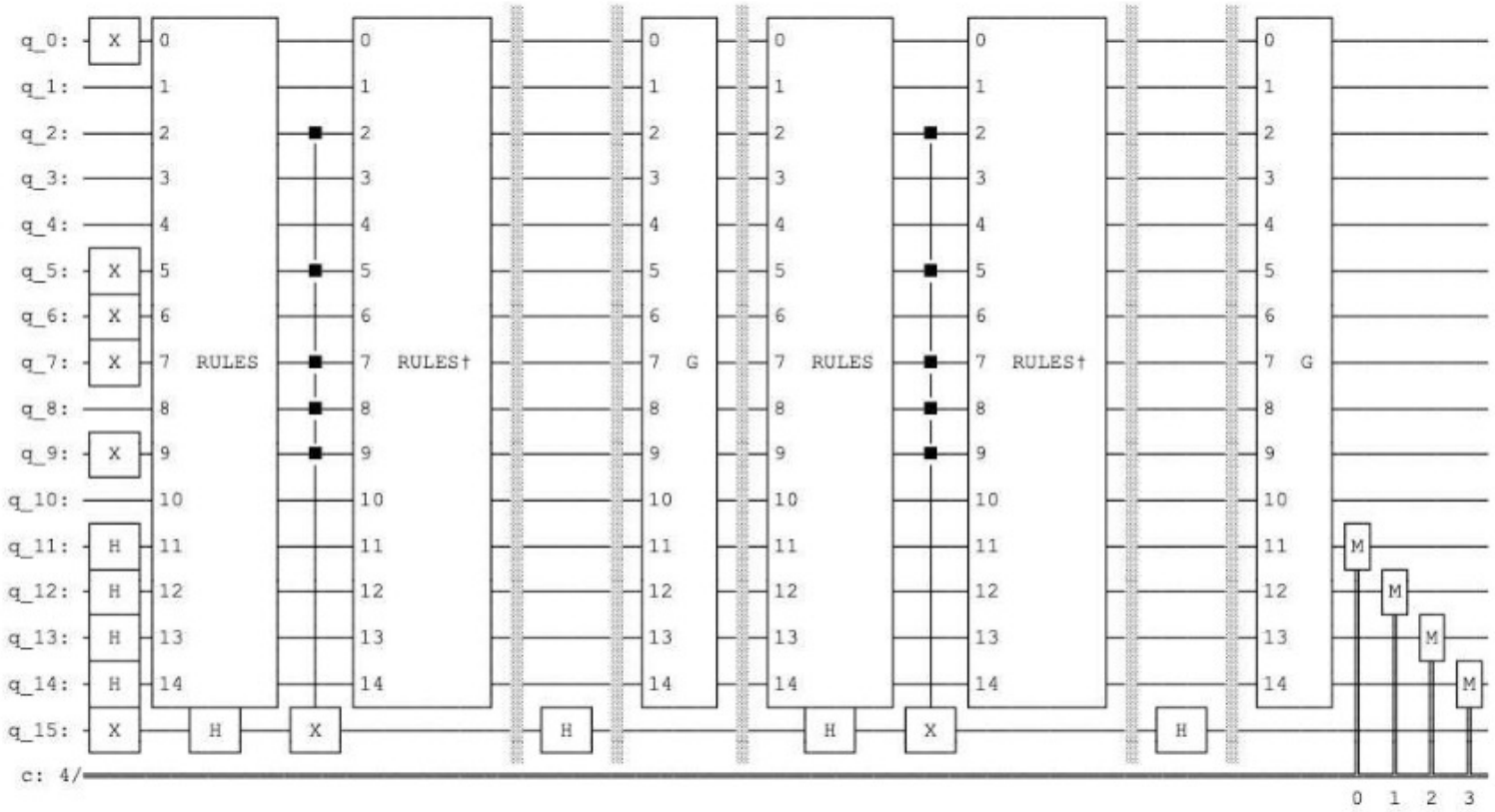
  At each search step all four rules can be applied in parallel.
- $\gamma_i \in \Sigma$ is the initial state: $cdcab = |1011100001\rangle$
- $\gamma_g \in \Sigma$ is the goal state is represented by the oracle : $dccba = |1110100100\rangle$

$\gamma_i \in \Sigma$ is the initial state: $cdcab = |1011100001\rangle$

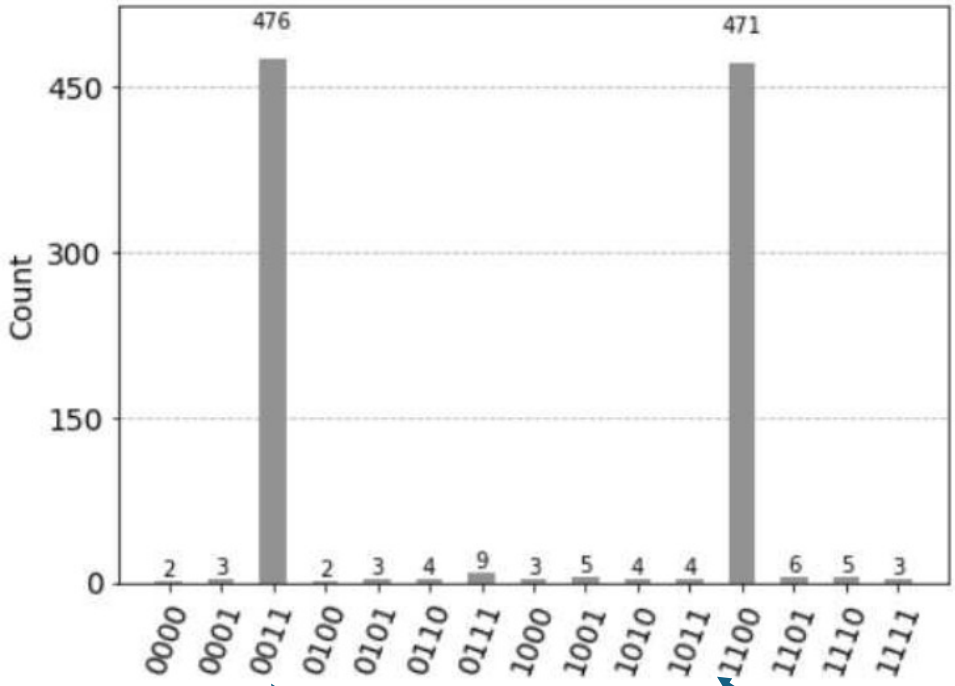


$\gamma_g \in \Sigma$ is the goal state is represented by the oracle : $dccba = |1110100100\rangle$

All 4 rules are executed at the same time with the depth 2



Quantum circuit for a simple quantum production systems for sorting of a string.
We apply the Grover's algorithm with two rotations.



The solution of the path descriptor $|0011\rangle$ corresponds to:

- index $|00\rangle$: $p_5p_4 \rightarrow p_4p_5$, working memory: $cdcab \rightarrow dccab$
- index $|11\rangle$: $p_2p_1 \rightarrow p_1p_2$, working memory: $dccab \rightarrow dccba$

The solution of the path descriptor $|1100\rangle$ corresponds to:

- index $|11\rangle$: $p_2p_1 \rightarrow p_1p_2$, working memory: $cdcab \rightarrow cdcba$
- index $|00\rangle$: $p_5p_4 \rightarrow p_4p_5$, working memory: $cdcba \rightarrow dccba$

Number of Iterations

The number of iterations r is

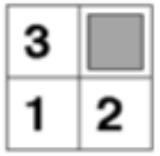
$$r = \lfloor t^* \rfloor = \left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{2^m}{k}} \right\rfloor$$

The value of r depends on the relation of n versus k, with k being the number of solutions. When r is unknown we can repeatedly chose r randomly between 1 and $\frac{\pi}{4} \cdot \sqrt{2^m}$. This simple strategy leads to success in $O(\sqrt{n}) = O(\sqrt{2^m})$ [82].

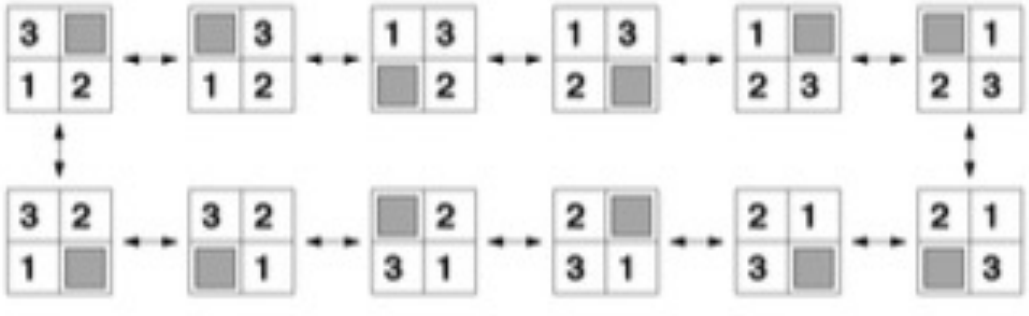
Control Function

- In many applications all productions cannot fire at the same time, instead **several instantiations** of **certain** productions can be executed.
- The control function determines if a production can fire or not
  - The preconditions are matched against the contents of the working memory.
- If the preconditions are met, then the productions can fire at the same time.
- All the productions that can fire are mapped in superposition and are instantiated
- In the next step all instantiated productions fire and changes the contents of the working memories in the superposition.

3-Puzzle



- The 3-puzzle is composed of three numbered movable tiles in a 2×2 frame
  - One cell of the frame is empty, and because of this tiles can be moved around to form different patterns

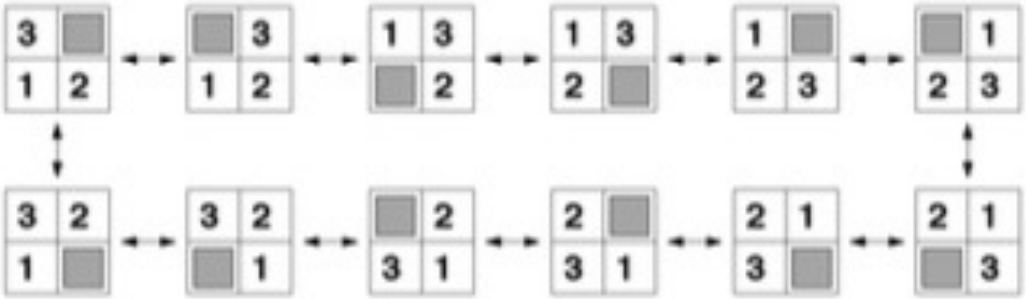


- There are twelve possible configurations
  - For any of this configuration only two movements are possible
  - The movement of the empty cell are either a clockwise or counter-clockwise movement

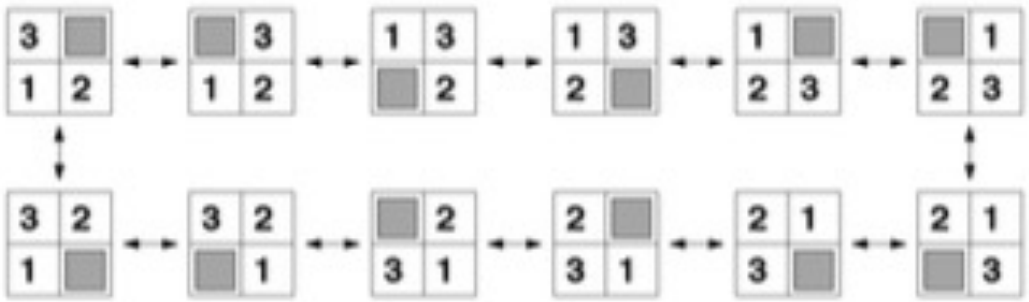
- The control function of the quantum production system needs to fulfill two requirements:

- For a given board, configuration and a production rule determine the new board configuration
- To determine if the configuration is the goal configuration.
  - The new board configuration is determined by productions that are represented by the function *p*
  - There are four possible positions of the empty cell.

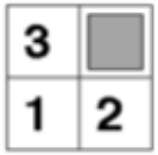
- The input of the function *p* is the current board configuration and a bit m that indicates whether the blank cell should perform a clockwise (*m = 1*) or counter-clockwise movement (*m = 0*)
- Together, there are *8* possible mappings, which are represented by *8* productions
- There are four possible positions of the empty cell times *two possible moves*



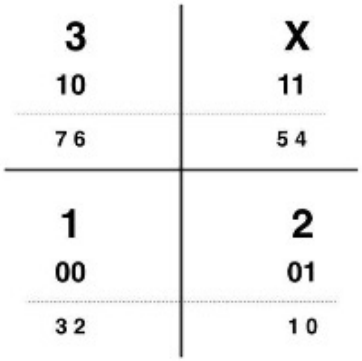
- For simplicity, we represent the mappings of the function p by a unitary permutation matrix *L(1)*
- For each mapping, the empty tile can have three different neighbors
- It follows that, in total, there are *24* = *8* × *3* instantiated rules



Representation



x=10 11 00 01



- Each object can be coded by two qubits (2^2), and a configuration of the four objects can be represented by a register of eight qubits
  - The object **1** is represented by *00*, **2** is represented by *01*, **3** is represented by *10* and empty space **x** is represented by *11*.
- The state is represented by 8 qubits $x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0$
  - *Little-endian representation in Qiskit*

- The position description (adjective) is fixed, and the class descriptors moves

- *L(1)* matrix acts on the *8 + 1* qubits with $m \in B^1$ and $x \in B^8$

$$L(1) \cdot |m\rangle|x\rangle = |m\rangle|\gamma\rangle.$$

- The *L(1)* matrix represents the long-term memory of our production system.
- The function *o(x)*, called oracle, determines if the configuration is the goal configuration

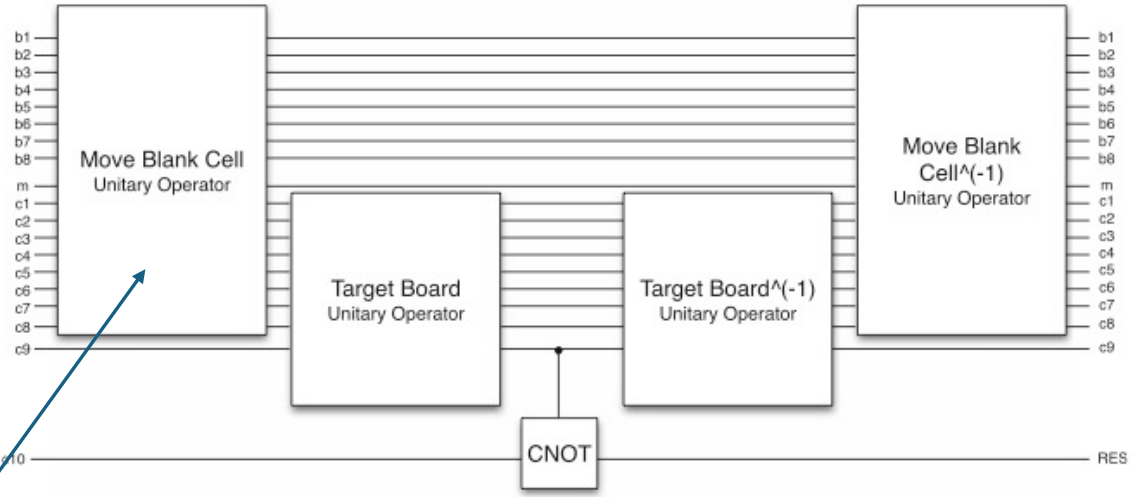
$$o(x) = o(\underbrace{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7}_{board\ \ configuration}) = \begin{cases} 1 & \text{if goal} \\ 0 & \text{otherwise.} \end{cases}$$

Big-endian representation in quantum physics books, Google CIRQ, Microsoft Q#

- Function *o(x)* oracle is represented by a unitary operator T (for target)
- T acts on the *8 + 1* qubits, with $x \in B^8$ and $c \in B^1$ being the auxiliary qubit

$$T \cdot |x\rangle|c\rangle = |x\rangle|o(x) \oplus c\rangle$$

- An important open question is whether the permutation matrix *L(1)* of dimension *512 = 29* can be decomposed?
- Yes, it can be decomposed.



- An important open question is whether the permutation matrix *L(1)* of dimension *512* = *29* can be decomposed? Yes, it can be!

Prof. Luís Domingues Tomé Jardim Tarrataca

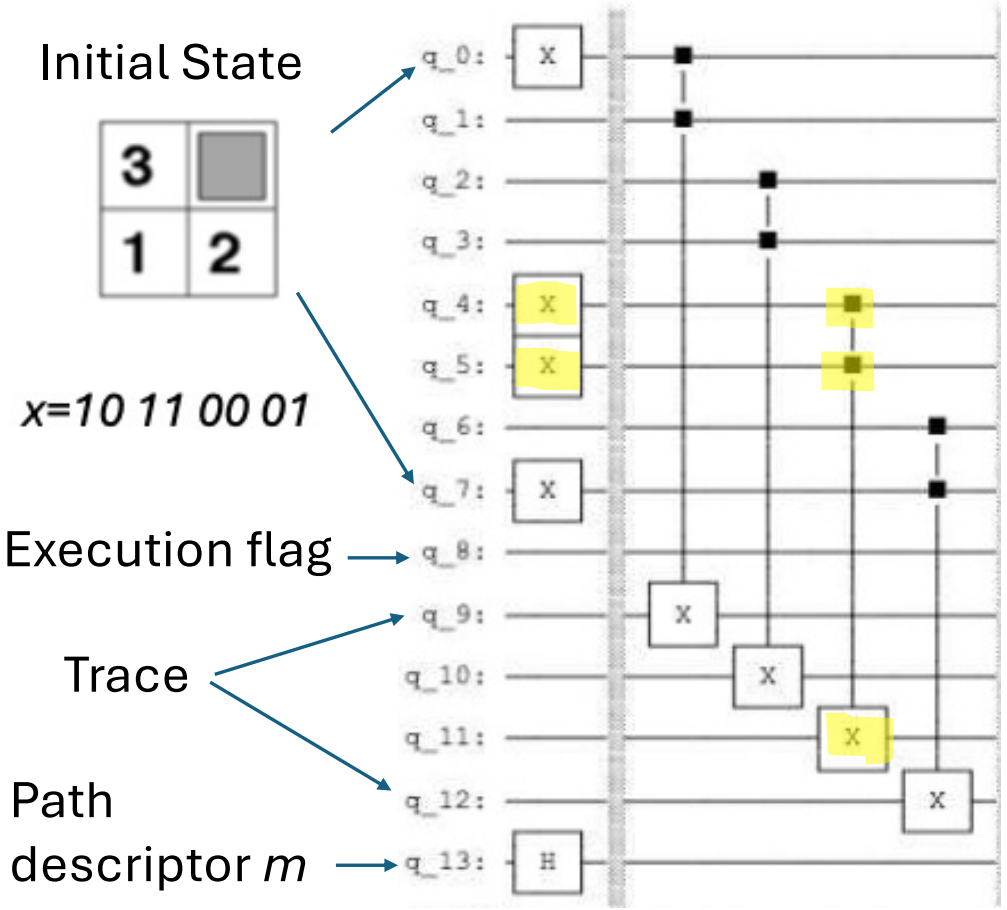
http://web.ist.utl.pt/luis.tarrataca/

PhD: **The Quantum Production System**

http://web.ist.utl.pt/luis.tarrataca/publications/theQuantumProductionSystem.pdf

- L. Tarrataca and A. Wichert: Tree Search and Quantum Computation, Quantum Information Processing, 10 (4): 475-500, 2011 doi:10.1007/s11128-010-0212-z
- L. Tarrataca and A. Wichert: A Quantum Production Model, Quantum Information Processing, 1(1): 189-209, 2012 doi:10.1007/s11128-011-0241-2
- L. Tarrataca and A. Wichert: Problem solving and quantum computation, Cognitive Computation, 3(4): 510-524, 2011 doi:10.1007/s12559-011-9103-6
- L. Tarrataca and A. Wichert: Quantum Iterative Deepening with an application to the Halting problem, PLOS ONE, 8(3) e57309, 2013, dx.plos.org/10.1371/journal.pone.0057309

Rules and Trace

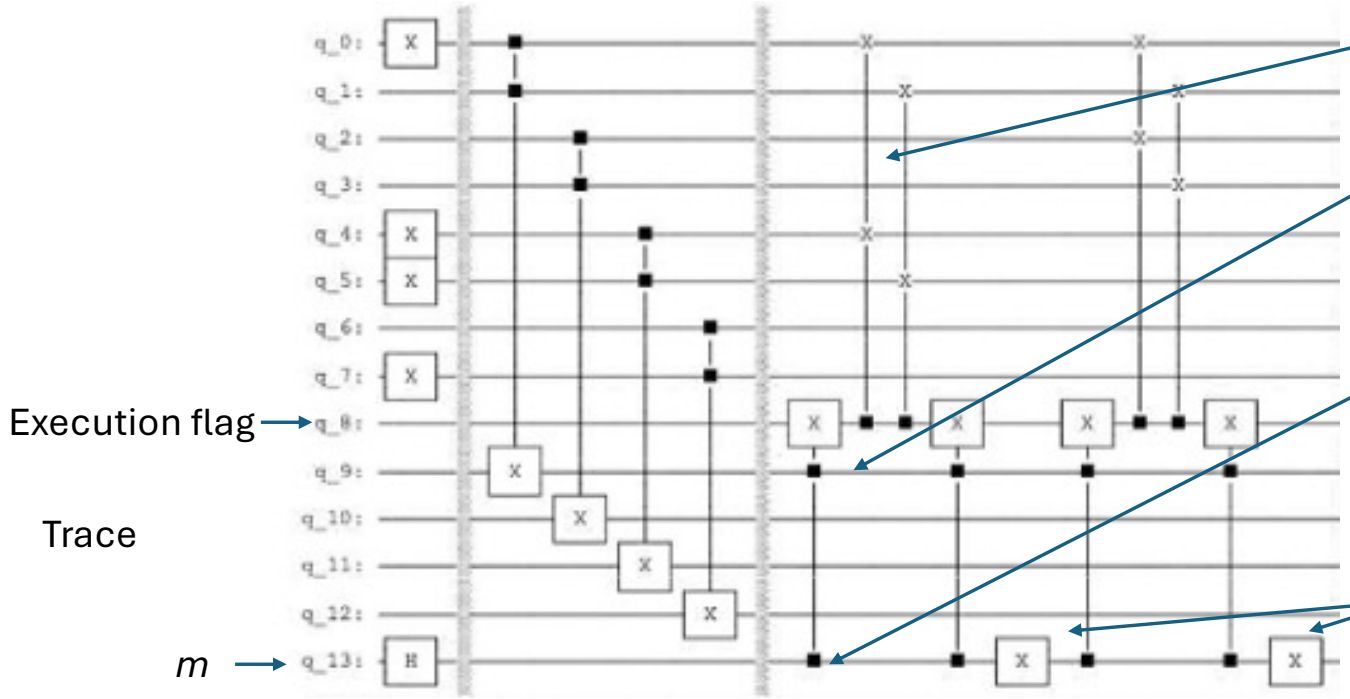


- The *if part of the rules* is implemented by the Toffoli gate, also called the ccX gate (CCNOT gate, controlled controlled not gate)
- It recognizes the position of the empty space and indicates it by setting one qubit of the four qubits *9* to *12* to one (Trace)

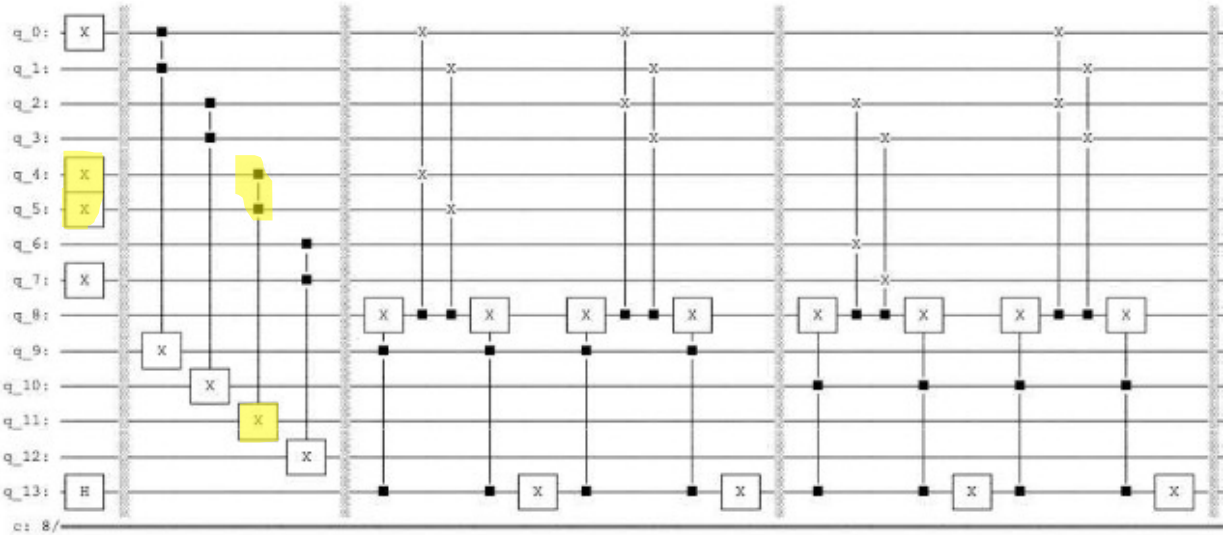
Why Trace?

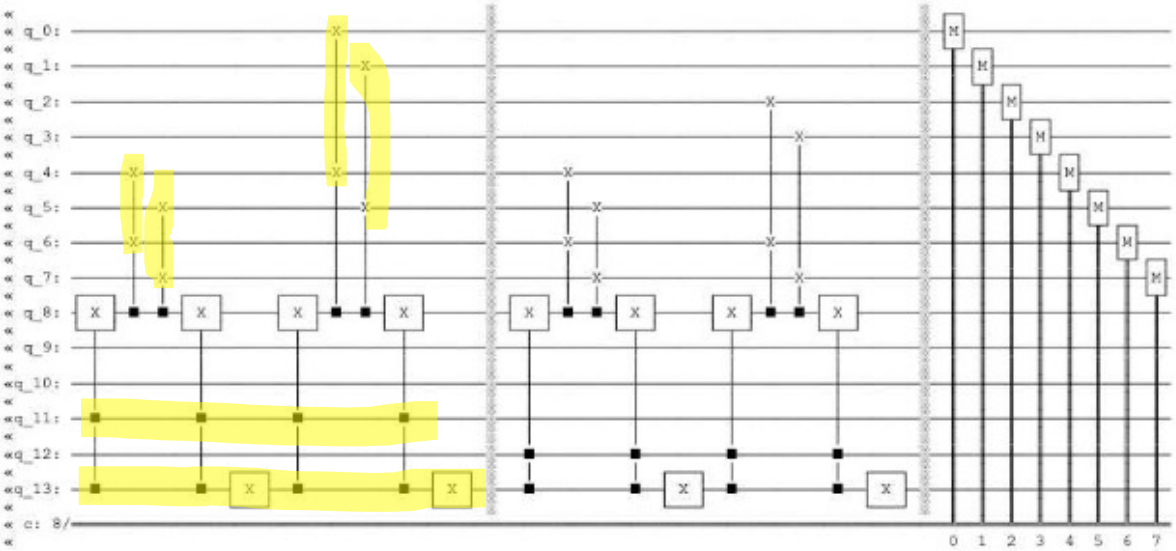
- We recognize the four rules and indicate the presence of a rule by a qubit
- We use four qubits that indicate the presence of the four rules and call them the trace
- We need the trace represented by the four qubits, since we cannot delete the information and we cannot un-compute the output back
- By un-computing, we would redo the rules.

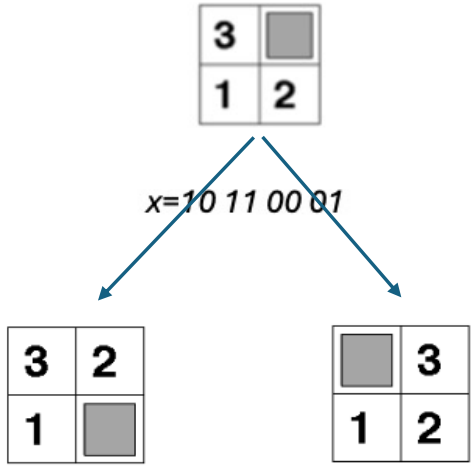
Execution of Rules

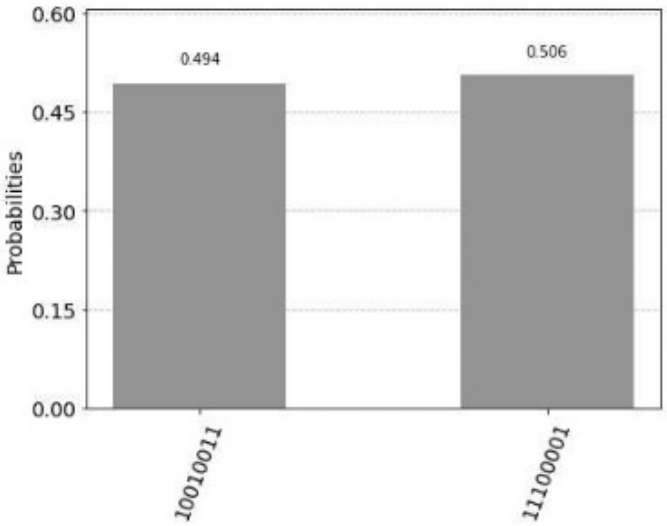


- The execution of the rules uses the Fredkin gate, also called controlled swap (CSWAP) gate, using the **trace** information and **the path descriptor** setting the flag qubit
- We change the path descriptor by the NOT gate and execute the second instantiation of the rule depending on the trace value


q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
q_12:
q_13:
c: 8/


q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
q_12:
q_13:
c: 8/
0 1 2 3 4 5 6 7


3
1 2
x=10 11 00 01
3 2
1
3
1 2


0.60
0.45
0.30
0.15
0.00
Probabilities
0.494
0.506
10010011
11100001

Search of depth Two

- Grover's amplification cannot be applied to fewer than four states. A search of depth one for the 3-puzzle results in two states and a search of depth two in four states.
- The operator L(2) that describes the search of depth two is represented as

$$L(2) \cdot |m_2, m_1\rangle|x\rangle = |m_2, m_1\rangle|\gamma\rangle,$$

- using two qubits, m_2,m_1, representing the path descriptor

- The unitary operator *T* represents the oracle function o(x) that determines if the configuration is the goal configuration

$$T \cdot |x\rangle|c\rangle = |x\rangle|o(x) \oplus c\rangle.$$

$$T \cdot H_{m+1} \cdot |0^{\otimes m}\rangle|1\rangle = \frac{1}{\sqrt{n}} \sum_{x \in B^m} (-1)^{o(x)} \cdot |x\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

  - For simplicity, we ignore the trace and the flag *qubit* and we obtain

$$(I_2 \otimes T) \cdot (L(2) \otimes I_1) \cdot (L(2) \otimes I_1) \cdot |m_2, m_1, x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, c\rangle$$
$$(I_2 \otimes T) \cdot (L(2) \otimes I_1)^2 \cdot |m_2, m_1, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, c\rangle.$$

- Depth search t, and a test condition in order to determine if the final board is a target configuration board, is represented with

$$L(t) \cdot |m_t, \cdots, m_1\rangle |x\rangle = |m_t, \cdots, m_1\rangle |\gamma\rangle$$

$$|\kappa^t\rangle = |m_t, \cdots, m_1\rangle$$

$$(I_t \otimes T) \cdot (L(t) \otimes I_1)^t \cdot |\kappa^t, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, c\rangle.$$

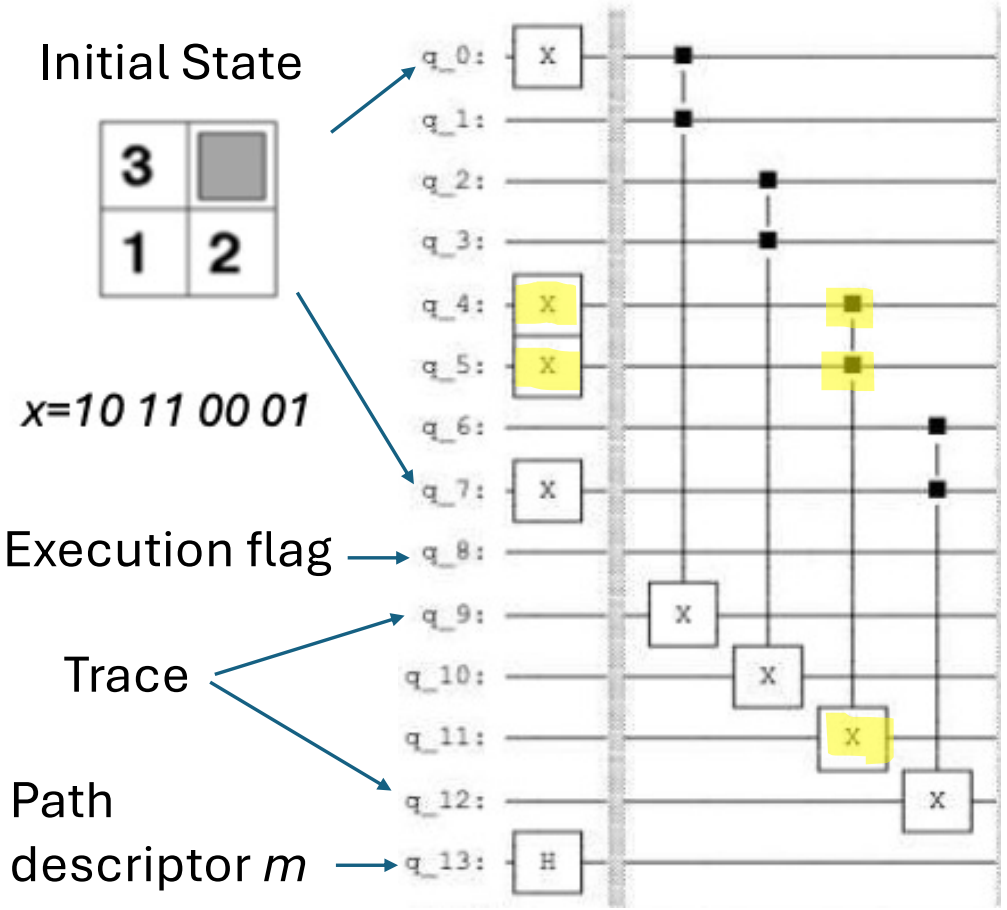
Un-Computation

- In quantum computation it is not possible to reset the information to the pattern representing the initial state
- Instead, we un-compute the output back to the input before applying the amplification step of the Grover's algorithm

$$((L(t) \otimes I_1)^*)^t \cdot (I_t \otimes T) \cdot (L(t) \otimes I_1)^t \cdot |\kappa^t, x, c\rangle$$

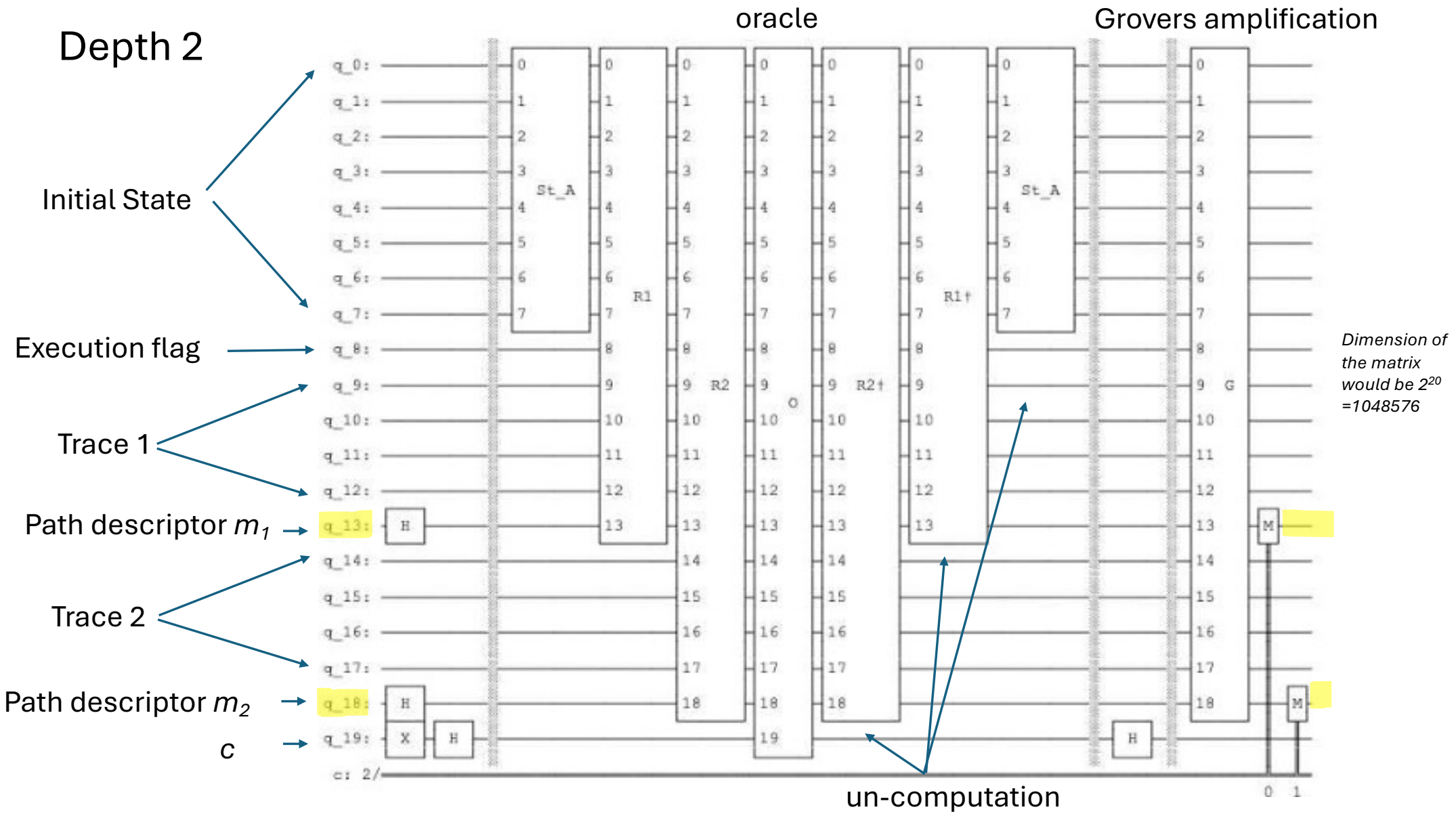
- Computation can be un-done
- The corresponding path is marked by a negative sign using the auxiliary qubit *c*

Rules and Trace

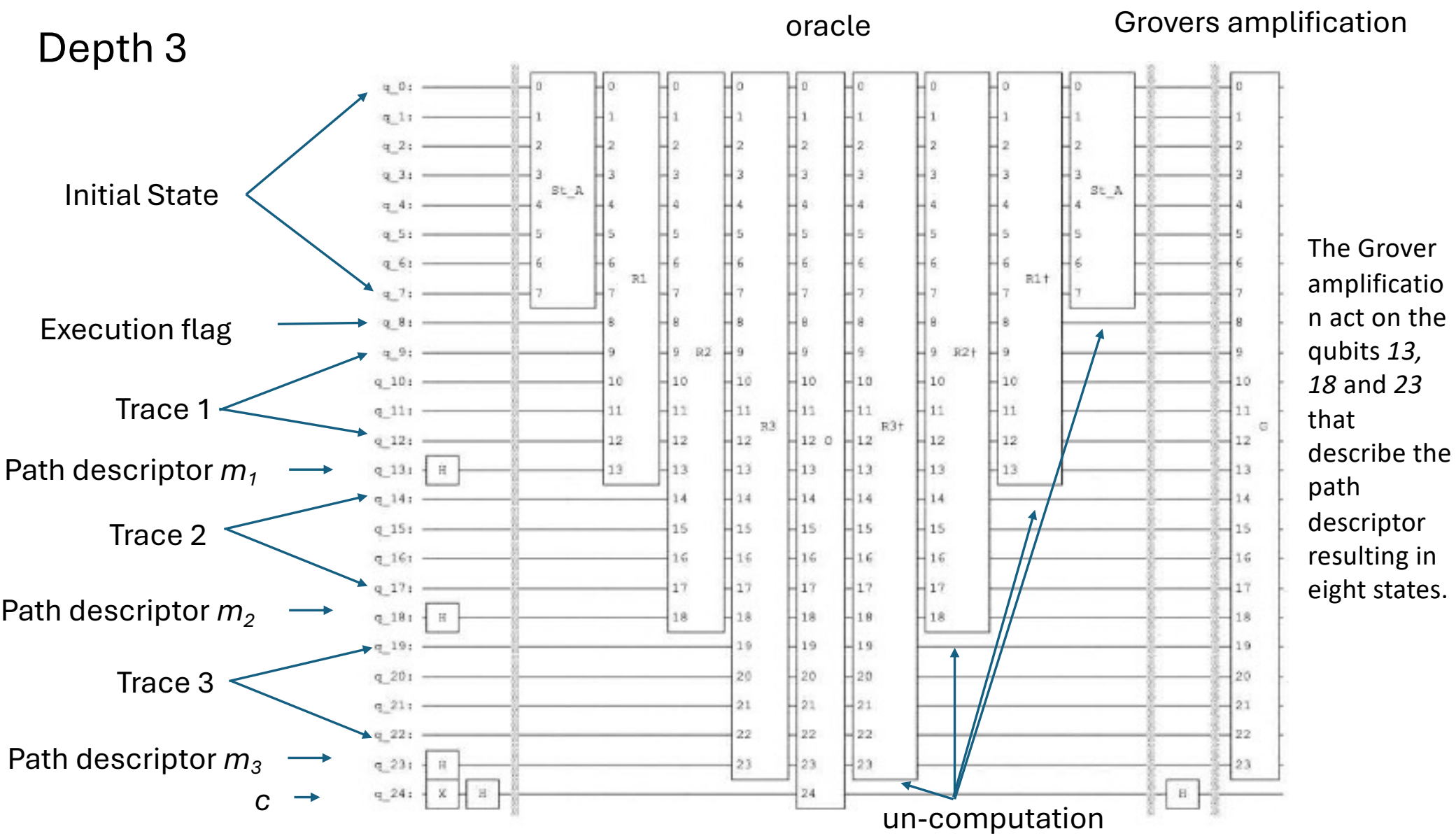


- The *if part of the rules* is implemented by the Toffoli gate, also called the ccX gate (CCNOT gate, controlled controlled not gate)
- It recognizes the position of the empty space and indicates it by setting one qubit of the four qubits *9* to *12* to one (Trace)

Depth 2

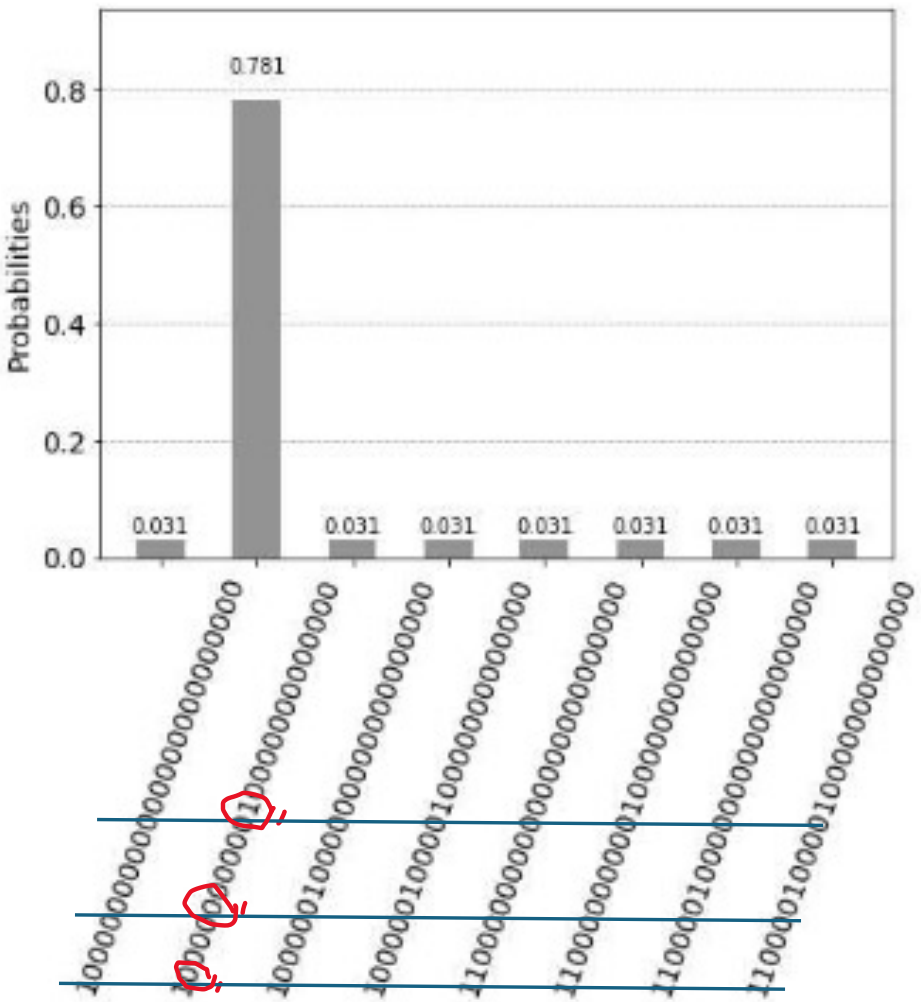

oracle
Grovers amplification
Initial State
Execution flag
Trace 1
Path descriptor m_1
Trace 2
Path descriptor m_2
c
St_A
R1
R2
O
R2†
R1†
St_A
G
H
X
M
un-computation
Dimension of the matrix would be 2^{20} =1048576

Depth 3



The Grover amplification act on the qubits *13, 18* and *23* that describe the path descriptor resulting in eight states.

```
def Grover():
    qc = QuantumCircuit(24)
    #Diffusor
    qc.h([13,18,23])
    qc.x([13,18,23])
    qc.h(13)
    qc.ccx(18,23,13)
    qc.h(13)
    qc.x([13,18,23])
    qc.h([13,18,23])
    qc.name="G"
    return qc
```



The Grover amplification act on the qubits *13, 18* and *23* that describe the path descriptor resulting in eight states.

One marked state resulted after one iteration is indicated with a probability value 0.781 and the path descriptor 001

Search with depth 3 and two iterations

We apply the $U_{3-puzzle}$ operator ignoring the trace for simplicity for the depth t resulting in 2^t states represented by the path descriptor

$$U_{3-puzzle} = ((L(t) \otimes I_1)^*)^t \cdot (I_t \otimes T) \cdot (L(t) \otimes I_1)^t$$

With Grover amplification on t qubits representing the path descriptor by the unitary operator G_t

$$\Gamma_t := (G_t \otimes I_{10}) \cdot U_{3-puzzle}.$$

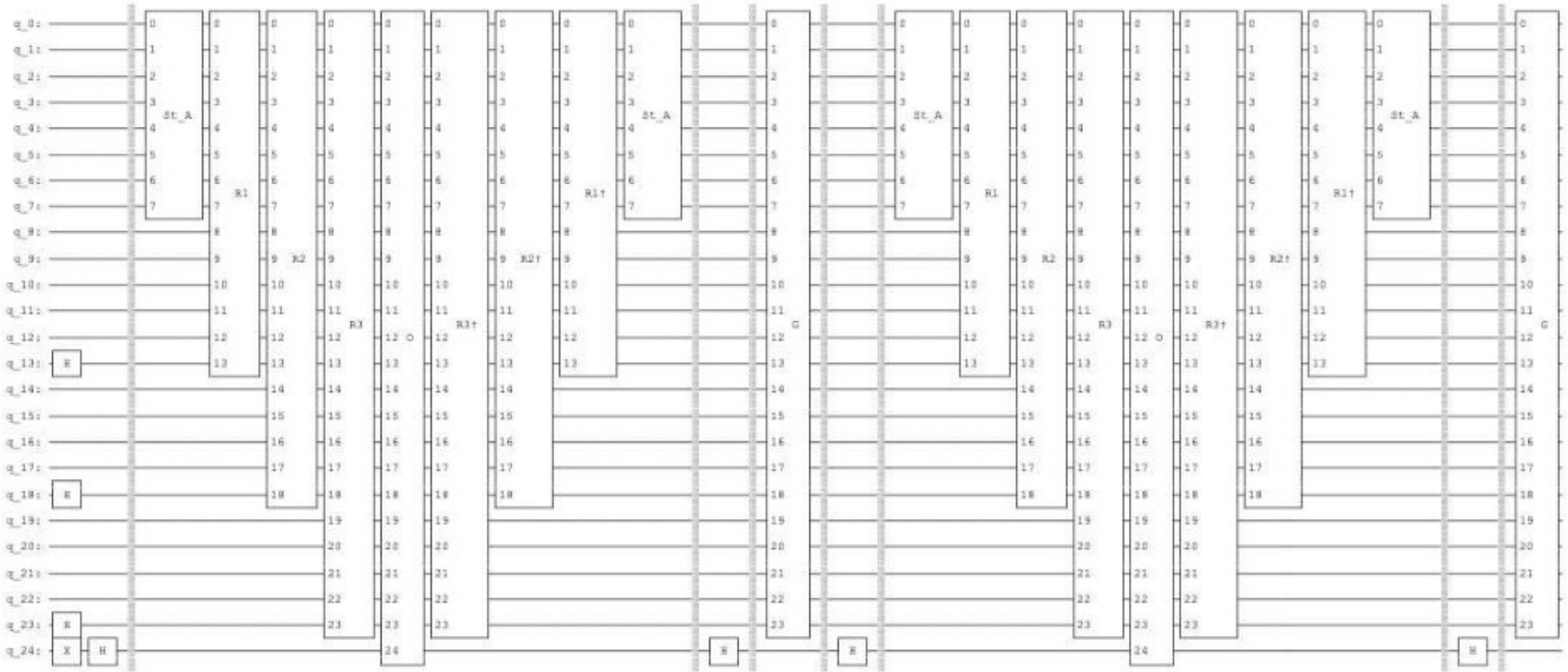
With r iterations

$$\Gamma_t^r = \prod_{t=1}^{r} \Gamma_t$$

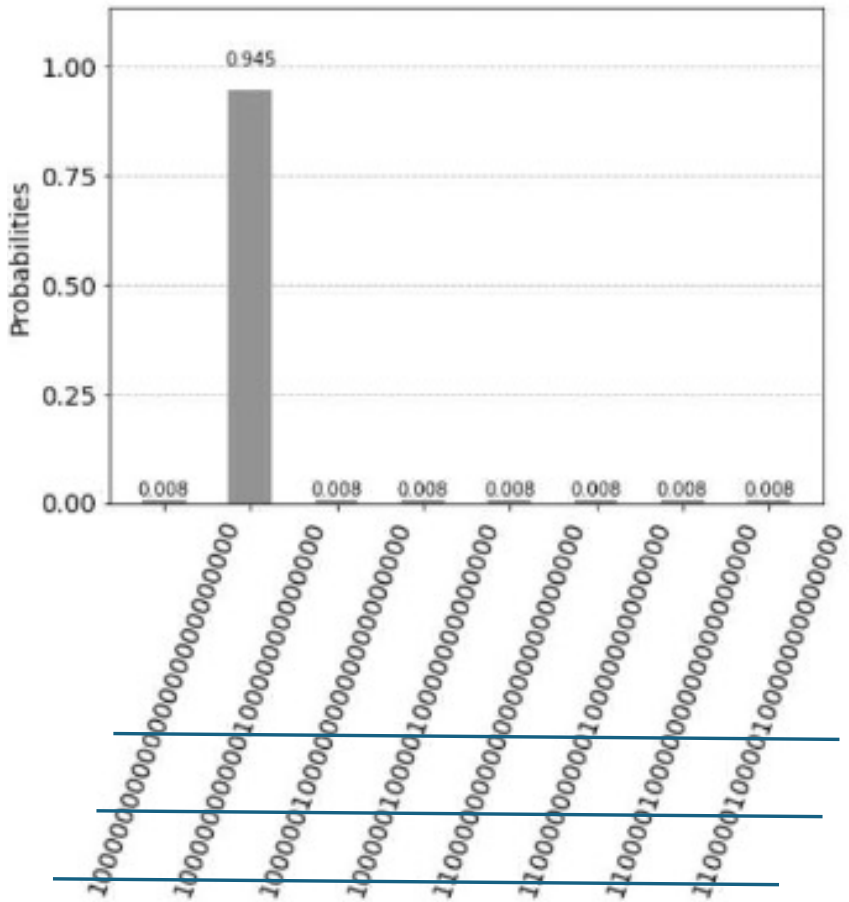
and determine the solution by the measurement of the register that represents the path descriptor.

$$\Gamma_t^r = \prod_{t=1}^{r} \Gamma_t$$

$t = 3$ and $r = 2$, with Γ_3^2

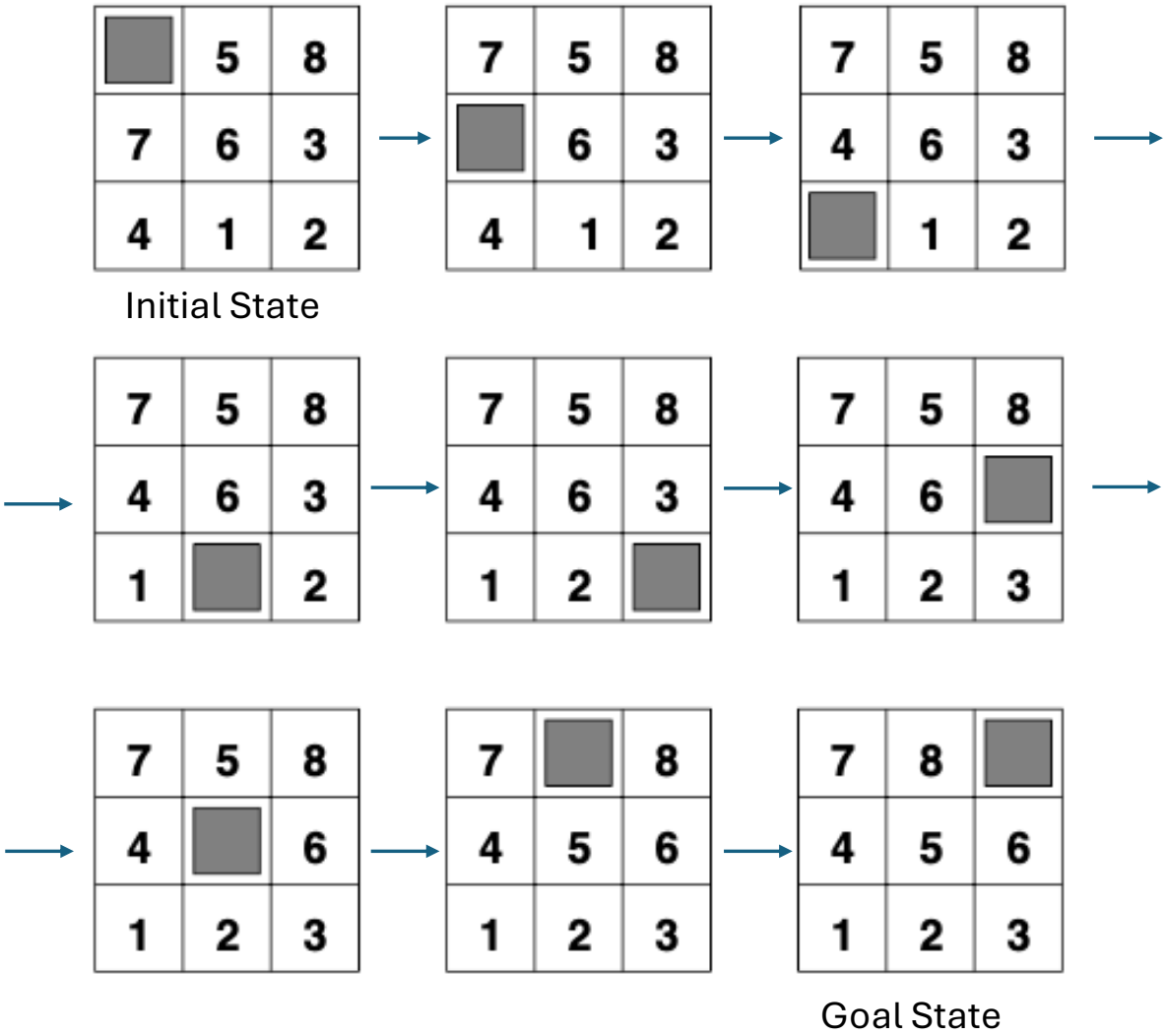


```
def Grover():
    qc = QuantumCircuit(24)
    #Diffusor
    qc.h([13,18,23])
    qc.x([13,18,23])
    qc.h(13)
    qc.ccx(18,23,13)
    qc.h(13)
    qc.x([13,18,23])
    qc.h([13,18,23])
    qc.name="G"
    return qc
```



One marked state results after two iterations is indicated with a probability value *0.945* by the *statevecto*r simulator
This is the optimum theoretical value for one marked solution using Grover's amplification of eight state
If we apply another rotation, the theoretical probability will decrease

8 Puzzle

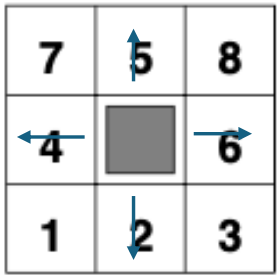


8 Puzzle: No Constant Branching Factor

For 8-puzzle $B_{max} = 4$, $B_{min} = 2$ and $B_{average}$

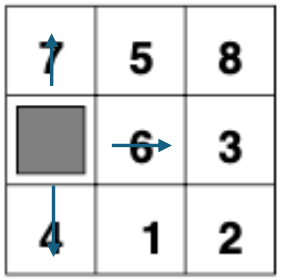
Naively, we would assume that the branching factor is reduced by Grover's amplification to

$$\sqrt{B_{max}} = \sqrt{4} = 2$$

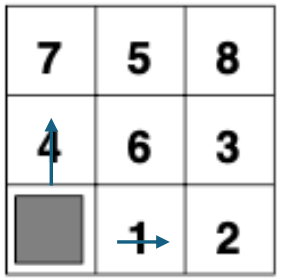


empty space in the center: 1*4 movements

$$B_{average} = \frac{4 \cdot 1 + 2 \cdot 4 + 3 \cdot 4}{9} = 2.6667.$$



empty space in the edge: 4*3 movements



empty space in the corner: 4*2 movements

Quantum Inf Process (2011) 10:475–500
DOI 10.1007/s11128-010-0212-z

Tree search and quantum computation

Luís Tarrataca · Andreas Wichert

Naively, we would assume that the branching factor is reduced by Grover's amplification to

$$= \sqrt{B_{max}}$$

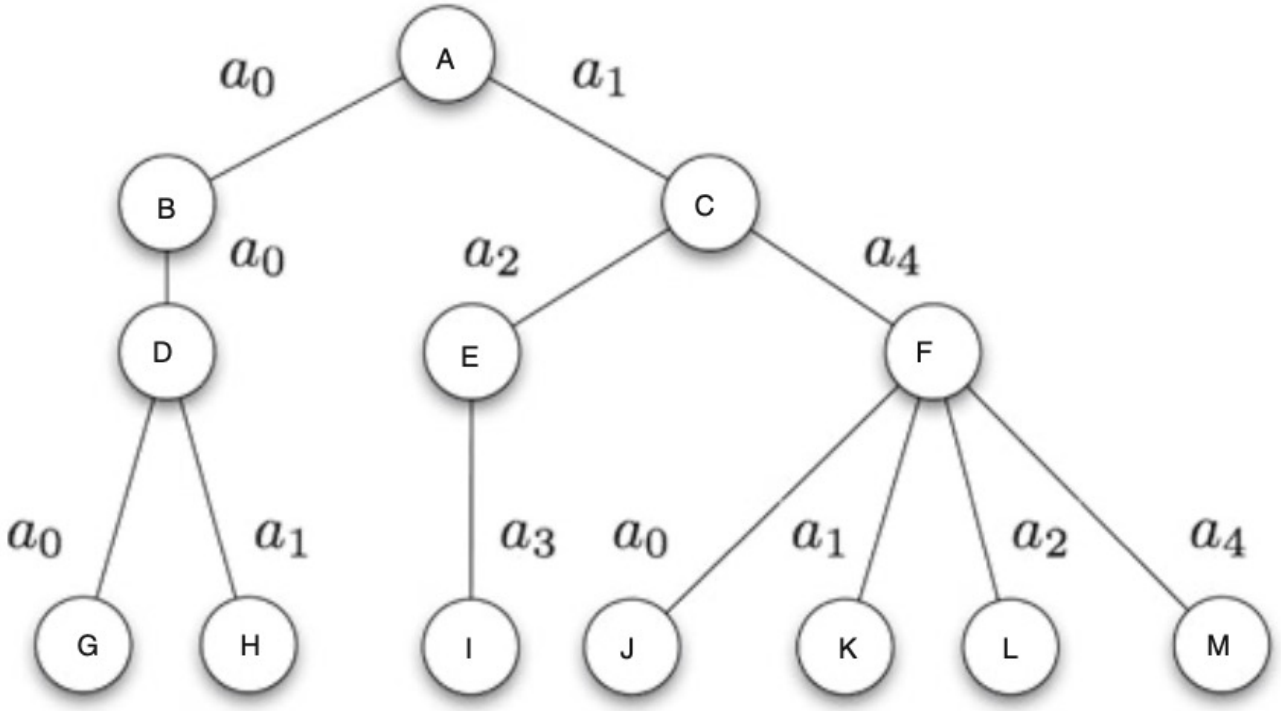
Received: 1 September 2010 / Accepted: 4 November 2010 / Published online: 21 November 2010

Abstract Traditional tree search algorithms supply a blueprint for modeling problem solving behaviour. A diverse spectrum of problems can be formulated in terms of tree search. Quantum computation, namely Grover's algorithm, has aroused a great deal of interest since it allows for a quadratic speedup to be obtained in search procedures. In this work we consider the impact of incorporating classical search concepts alongside Grover's algorithm into a hybrid quantum search system. Some of the crucial

- Suppose the branching factor is not constant, in this case the tree search can be described by the effective branching factor.
- Effective branching factor b is related to the costs C_m represented by the number of generated nodes during A^* search (≈deep search)

$$C_m = 1 + b + b^2 + b^3 + \cdots + b^m = \sum_{i=0}^{m} b^i = \frac{1 - b^{m+1}}{1 - b}.$$

- For uninformed tree search for a large number of instances (different initial and goal states) the effective branching factor converge to the averaged branching factor for uninformed tree search



A search tree with a maximum branching factor B_{max} = 5 and an average branching factor B_{avg} = *(2+1+2+2+1+4)/6 = 2*

- For a not constant branching factor the quantum tree search the maximal branching factor B_{max} has to be used for the quantum tree search
- For B_{max} the quantum algorithm using qubit representation is better then the classical tree search described by the effective branching factor b in the case

$$b > b_q = \sqrt{B_{max}}$$

$$b > b_q.$$

If B_{max} is a power of two then

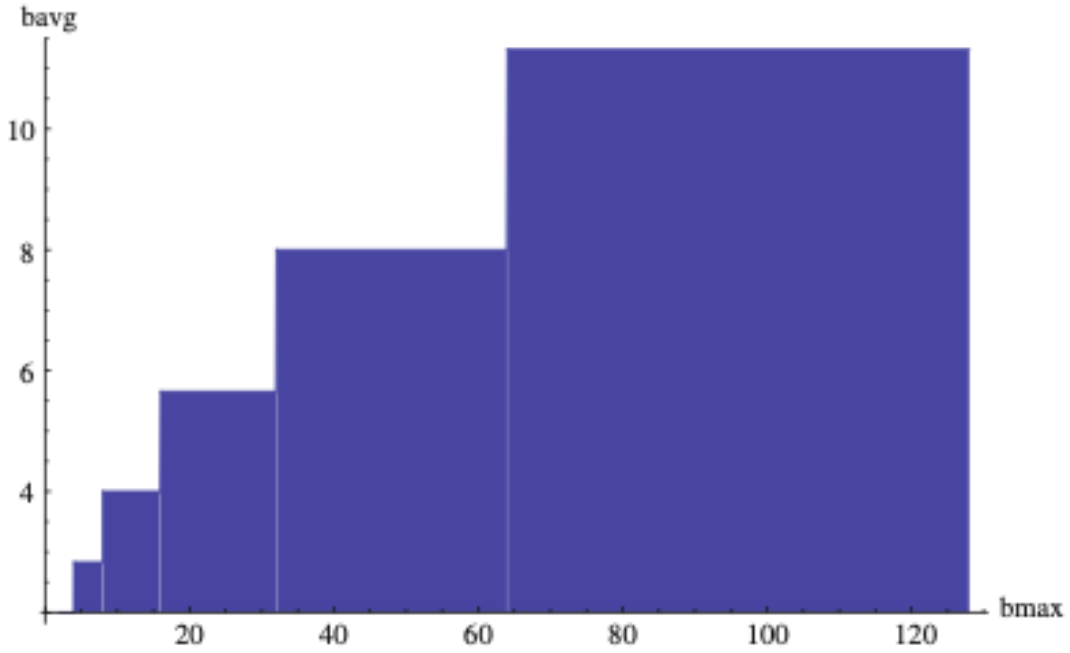
$$b > b_q = \sqrt{2^{\lceil \log_2 B_{max} \rceil}} = \sqrt{B_{max}}$$

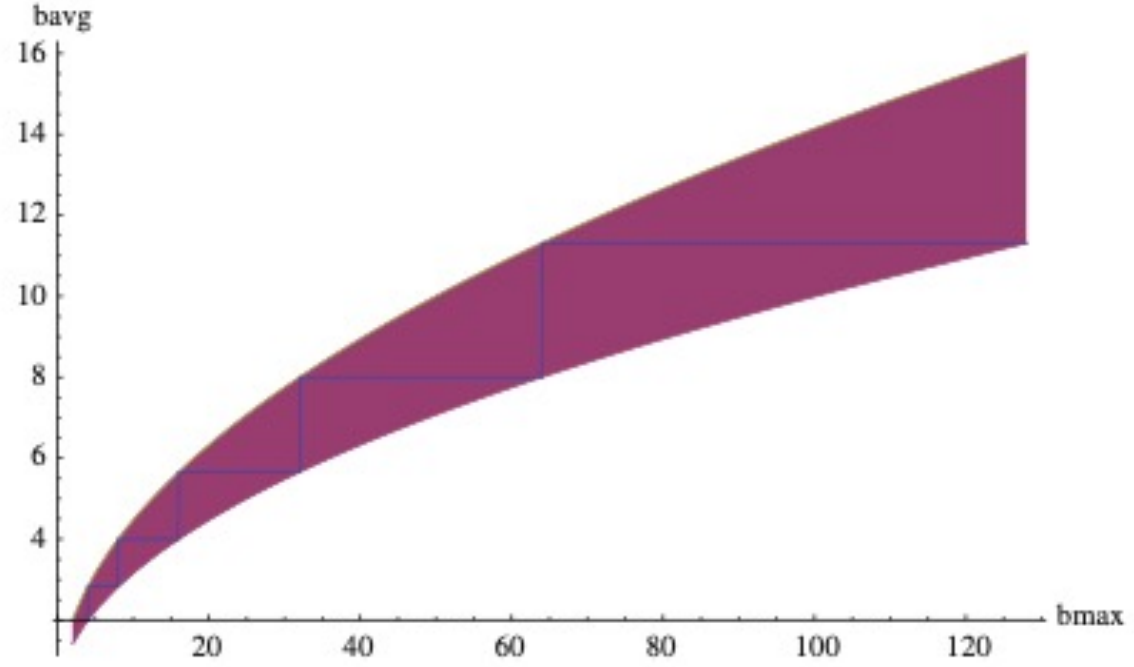
otherwise

$$b > b_q = \sqrt{2^{\lceil \log_2 B_{max} \rceil}} > \sqrt{B_{max}}$$

for example $B_{max} = 9$, then

$$b_q = 4 = \sqrt{2^4} > 3 = \sqrt{9}.$$


bavg
10
8
6
4
20
40
60
80
100
120
bmax


bavg
16
14
12
10
8
6
4
20
40
60
80
100
120
bmax

2021 IEEE/ACM 2nd International Workshop on Quantum Software Engineering (Q-SE)

Wrong?

Generalised Quantum Tree Search

André Sequeira
Department of Mathematics
University of Aveiro
Aveiro, Portugal
andresequeira401@gmail.com

Luis Paulo Santos
International Iberian Nanotechnology Lab
CSIG, INESC TEC
University of Minho
Braga, Portugal
psantos@di.uminho.pt

Luis Soares Barbosa
International Iberian Nanotechnology Lab
HASLab, INESC TEC
University of Minho
Braga, Portugal
lsb@di.uminho.pt

***Abstract*—This extended abstract reports on on-going research on quantum algorithmic approaches to the problem of generalised tree search that may exhibit effective quantum speedup, even in the presence of non-constant branching factors. Two strategies are briefly summarised and current work outlined.**

strategies can overcome the quadratic speed-up employed by Grover's algorithm. It is of great importance the study of quantum heuristics and their potential to scale quantum tree search algorithms. This abstract reports on-going work on two novel strategies for performing tree search in the quantum setting:

Wrong?

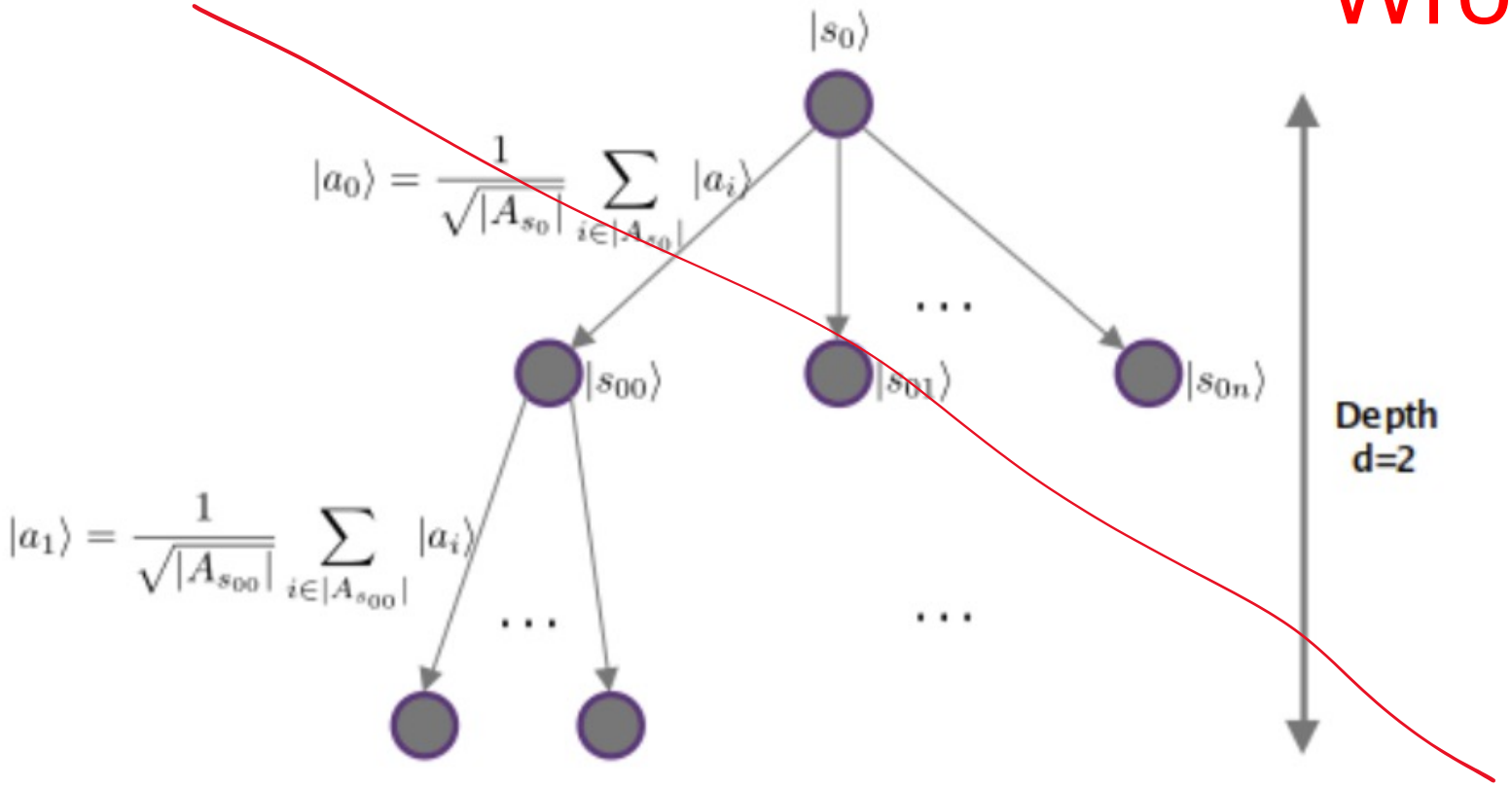


Figure 1: Superposition tree with arbitrary branching factor

Dynamically branching factor, possible?

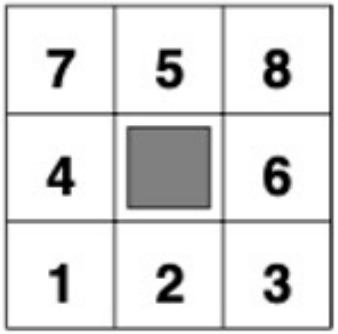
- How to know how many solutions exist at each depth?
- One could determine the branching factor dynamically in the way you can determine which rule to apply (indicating by trace)
- Generate dynamically path descriptor (Hadamard gates)
- The problem
  - The path descriptor should **be only entangled with the indicator of possible solution(s),** so we can apply Grover's amplification

- Imagine we have only a possible rule with two instantiations and another with four instantiations of branching.
- An if part would determine if apply two branching or four
- This if part would determine if we applied **one** Hadamard gate or **two** in parallel
- Problem: **the resulting qubits that describe the path descriptor would be entangled with the if part routine** and we cannot apply Grover's amplification

- By simulating the quantum tree search, it becomes clear that the branching factor is reduced by Grover's amplification to the square root of the **average branching factor** and not to the maximal branching factor as previously assumed

$$\sqrt{B_{average}} = \sqrt{2.6667} = 1.63299.$$

- For 8-puzzle, there are 9 different objects: eight cells and one empty cell
- Each object has to be represented by four *4* qubits since *3* qubits allow only to represent $2^3 = 8$ different states
- The object 1 is represented by 001, 2 is represented by 010 and 3 is represented by 011, and we continue the representation as binary numbers with 8 represented as 1000
- We represent the empty space x by 1111. The state is represented by 36 qubits $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, \cdots, x35$



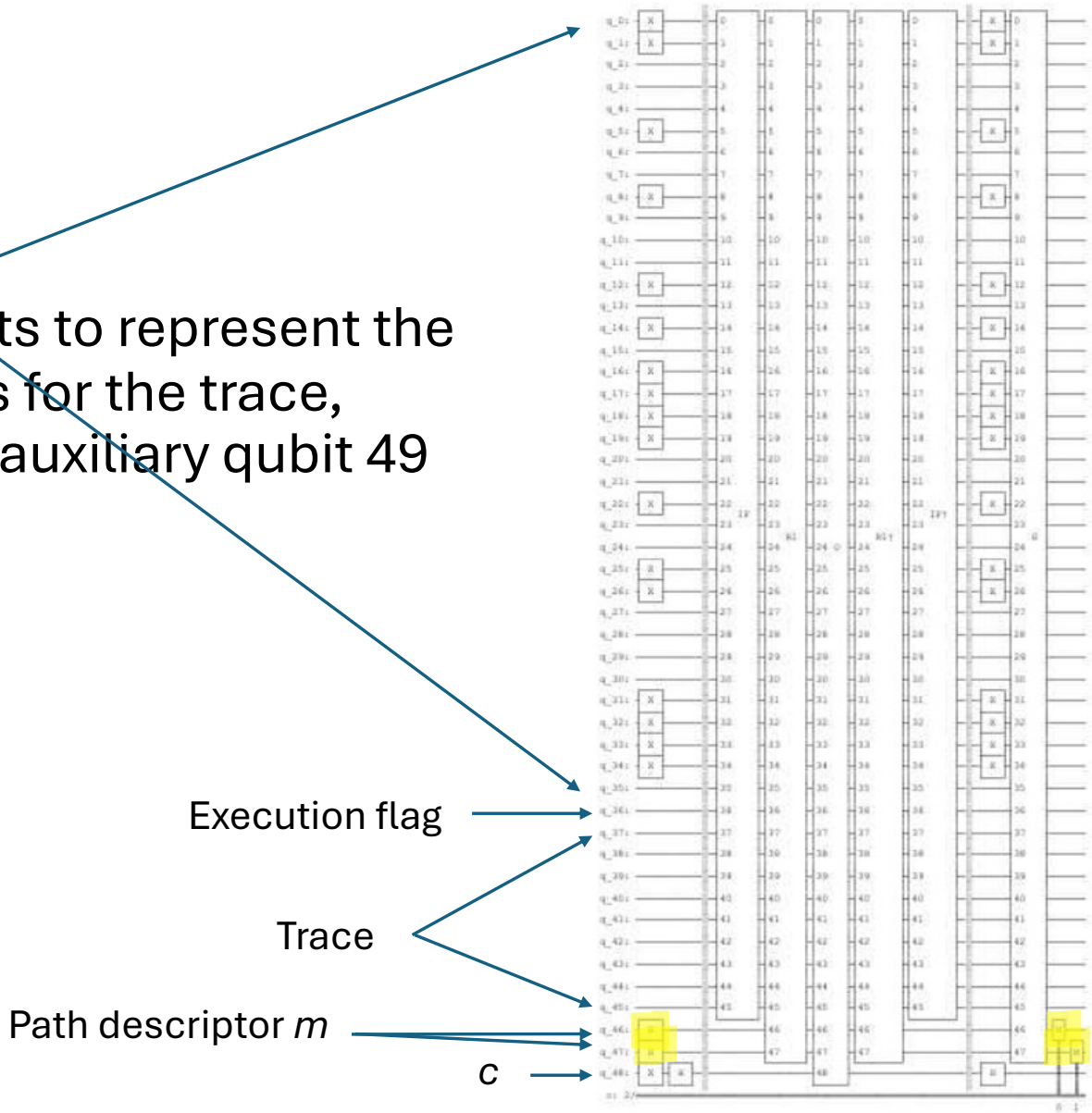
7 0111 35 34 33 32	8 1000 31 30 29 28	6 0110 27 26 25 24
4 0100 23 22 21 20	X 1111 19 18 17 16	5 0101 15 14 13 12
1 0001 11 10 9 8	2 0010 7 6 5 4	3 0011 3 2 1 0

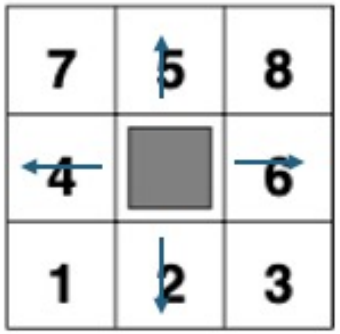
- The empty cell can be present in 9 different positions. The empty cell can move either up, down, left or right
- The new board configuration is determined by the function p. The input of the function p is the current board configuration and two bits $m = m_1, m_2$ (qubits *46* and *47*) indicating whether the blank cell should perform

  - move right (m = 0 = $|00\rangle$), left (m = 1 = $|01\rangle$), up (m = 2 = $|10\rangle$) or down (m = 3 = $|11\rangle$)

Depth 1

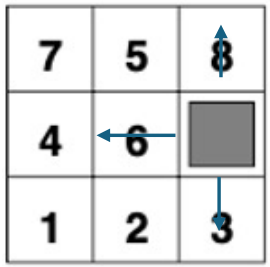
- There are 36 qubits to represent the state and 9 qubits for the trace, together with the auxiliary qubit 49





For the empty space in the center, there are four instantiations corresponding to the four movements

- For the path descriptor 00, move right $16, 17, 18, 19 \rightarrow 12, 13, 14, 15$.
- For the path descriptor 01, move left $16, 17, 18, 19 \rightarrow 20, 21, 22, 23$.
- For the path descriptor 10, move up $16, 17, 18, 19 \rightarrow 28, 29, 30, 31$.
- For the path descriptor 11, move down $16, 17, 18, 19 \rightarrow 4, 5, 6, 7$.

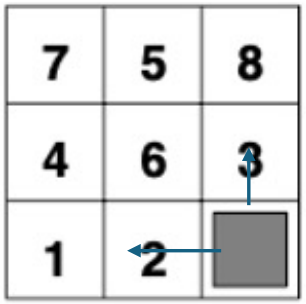


For the empty space in the edge, there are four instantiations corresponding to the three movements. The representation if performed in the same way as before, in our example, the empty space is at the position $12, 13, 14, 15$.

- For the path descriptor 00, move up $12, 13, 14, 15 \rightarrow 24, 25, 26, 27$. ← First Movement
- For the path descriptor 01, move down $12, 13, 14, 15 \rightarrow 0, 1, 2, 3$. ← Second Movement
- For the path descriptor 10, move left $12, 13, 14, 15 \rightarrow 16, 17, 18, 19$.
- For the path descriptor 11, move left $12, 13, 14, 15 \rightarrow 16, 17, 18, 19$.

Third Movement

The only difference is that the rule move left is repeated twice.



The only difference is that the rule move left is repeated twice. For the empty space in the corner, there are four instantiations corresponding to the two movements. The representation is performed in the same way as before, in our example, the empty space is at the position $0, 1, 2, 3$.

- For the path descriptor 00, move up $0, 1, 2, 3 \rightarrow 12, 13, 14, 15$.
- For the path descriptor 01, move up $0, 1, 2, 3 \rightarrow 12, 13, 14, 15$.

First Movement

- For the path descriptor 10, move left $0, 1, 2, 3 \rightarrow 4, 5, 6, 7$.
- For the path descriptor 11, move left $0, 1, 2, 3 \rightarrow 4, 5, 6, 7$.

Second Movement

The rule move up and the rule move left are repeated twice.

Number of Iterations

- For *k* solutions, the probability of measuring a state that represents one solution of k solutions is related to the number r of iterations of the Grover's operator

$$r = \left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{2^m}{k}} \right\rfloor$$

- After *r* iterations, the probability of measuring a solution is nearly one, with m being the number of qubits describing the path descriptor

$$r = \sqrt{\frac{(B_{max})^m}{k}}$$

- The value of *r* depends on the relation of *m* versus *k*, and *k* is bigger than one since we execute **same rules several** times
  - Different path descriptors represent the same solution
- For the depth *m*, there are *k* solutions with converge to

$$k = \left(\frac{B_{max}}{B_{average}} \right)^{m}$$

$$r = \sqrt{\frac{(B_{max})^{m}}{k}} \qquad r = \sqrt{(B_{average})^{m}}$$

Branching Factor

- The branching factor b_q converge to $\sqrt{B_{average}}$ since we execute **same rules several** times

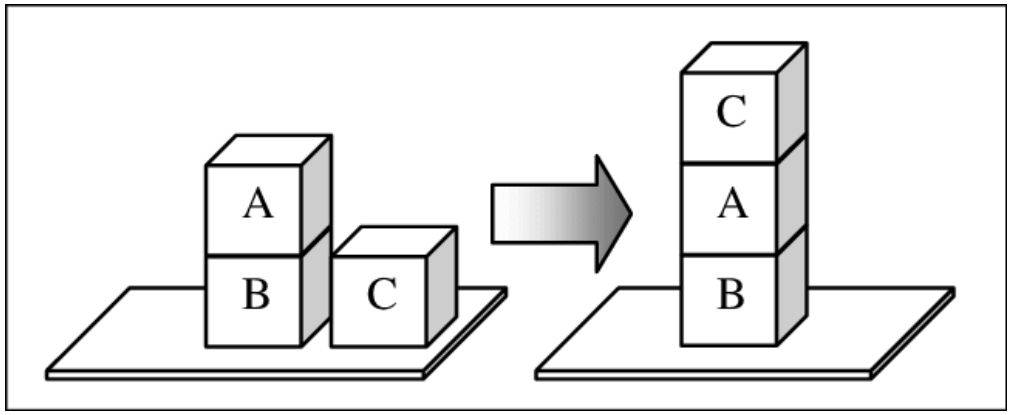
- For 8 puzzle it converges to $B_{average} = \frac{4 \cdot 1 + 2 \cdot 4 + 3 \cdot 4}{9} = 2.6667.$

$$\sqrt{B_{average}} = \sqrt{2.6667} = 1.63299.$$

- Which is much smaller compeered to $\sqrt{B_{max}} = \sqrt{4} = 2$

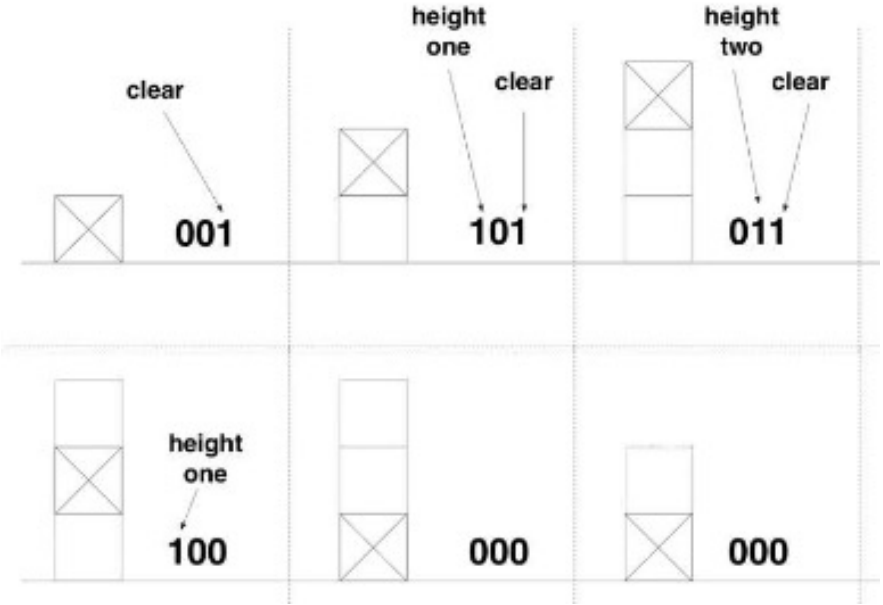
Blocks World

- The blocks world is a planning domain in artificial intelligence.
- The blocks can be placed at the table and picked up and set down on a table or another block, and the goal is to build one or more vertical stacks of blocks.
  - There are three different types of blocks, they are traditionally called A, B, C blocks

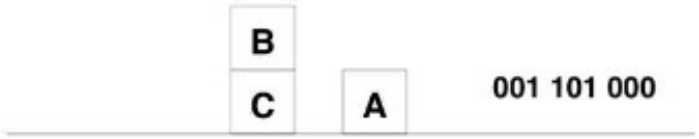


Representation

- The class descriptor is fixed and the position descriptor (adjective) moves.
  - It is reversed as in the puzzle examples, since the reverse in this case is a more economic representation requiring 9 qubits, three qubits for each

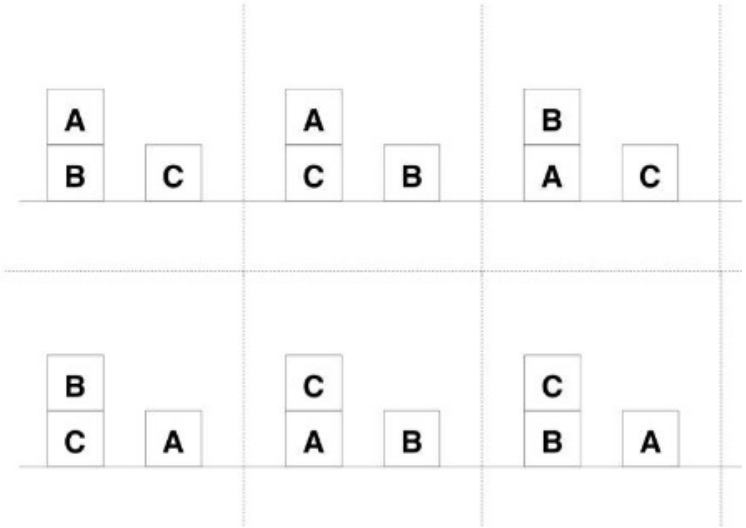


A		B		C		
x	2	x	5		8	clear
	1		4		7	height two
	0	x	3		6	height one

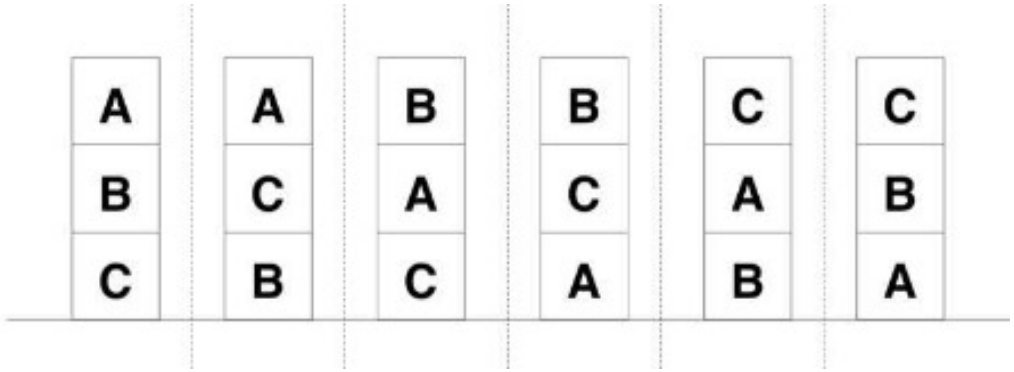


Possible States

The class all blocks on floor has one combination



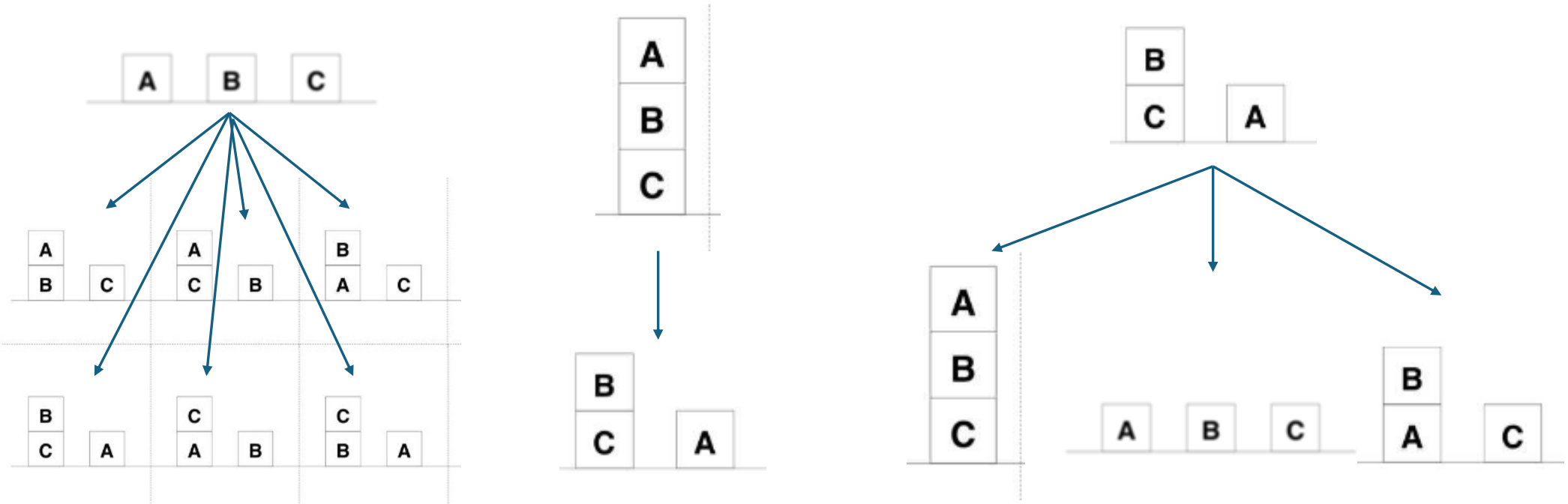
The class small tower and a block on table appears in six different combi-nations.



The class tower appears in six different combinations.

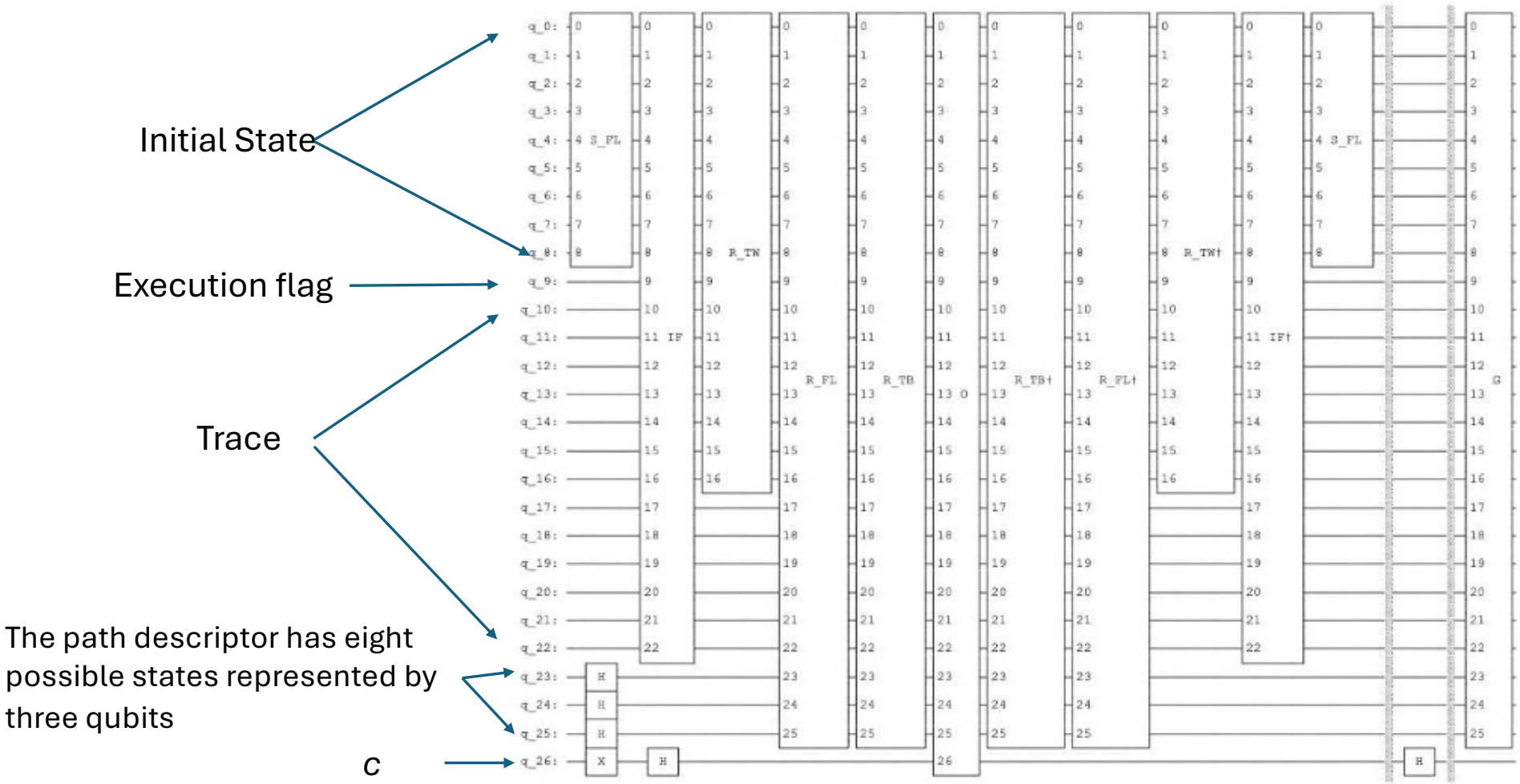
Rules

For A, B, C blocks $B_{max} = 6$, $B_{min} = 1$ and $B_{average}$

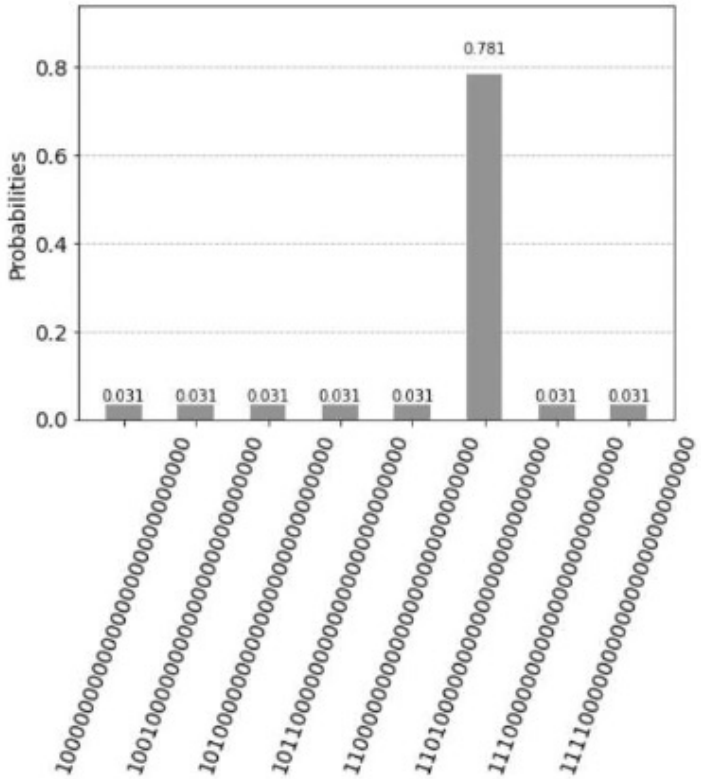


$$B_{average} = \frac{6 \cdot 1 + 1 \cdot 6 + 3 \cdot 6}{13} = 2.30769.$$

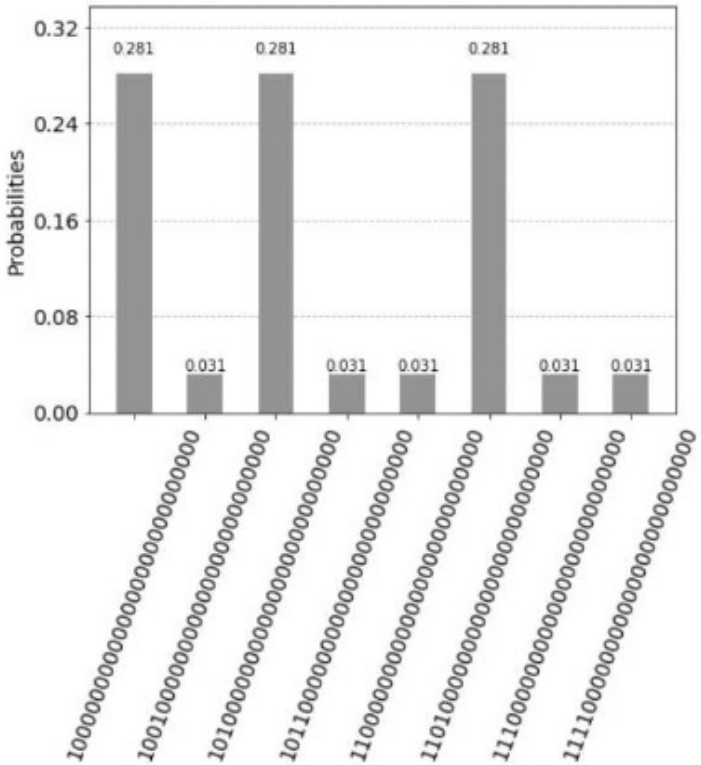
The quantum circuit for the ABC blocks task of the depth search 1




A
B
C
A
C
B


Probabilities
0.0
0.2
0.4
0.6
0.8
0.781
0.031
0.031
0.031
0.031
0.031
0.031
0.031
1000000000000000000000
1001000000000000000000
1010000000000000000000
1011000000000000000000
1100000000000000000000
1101000000000000000000
1110000000000000000000
1111000000000000000000


A
B
C
A
C
B


Probabilities
0.00
0.08
0.16
0.24
0.32
0.281
0.281
0.281
0.031
0.031
0.031
0.031
0.031
0.031
1000000000000000000000
1001000000000000000000
1010000000000000000000
1011000000000000000000
1100000000000000000000
1101000000000000000000
1110000000000000000000
1111000000000000000000

Number of Iterations

For A, B, C blocks $B_{max} = 6$, $B_{min} = 1$ and $B_{average}$

$$B_{average} = \frac{6 \cdot 1 + 1 \cdot 6 + 3 \cdot 6}{13} = 2.30769.$$

Naïvely, we would assume that the branching factor is reduced by Grover's amplification to the number 8 represented by three qubits

$$\sqrt{8} = 2.82843$$

With growing value m, the branching factor is reduced by Grover's amplification to

$$\sqrt{B_{average}} = \sqrt{2.30769} = 1.51911$$

For k solutions, the probability of measuring a state that represents one solution of k solutions is related to the number r of iterations of the Grover's operator. Simplified, we can state that

$$r = \sqrt{\frac{(8)^m}{k}}$$

The value of r depends on the relation of m versus k. For the depth m

$$k = \left(\frac{8}{B_{average}}\right)^m$$

it follows

$$r = \sqrt{(B_{average})^m}$$

and the branching factor is reduced by Grover's amplification to $\sqrt{B_{average}} = 1.5191$.

Games and Quantum Tree Search

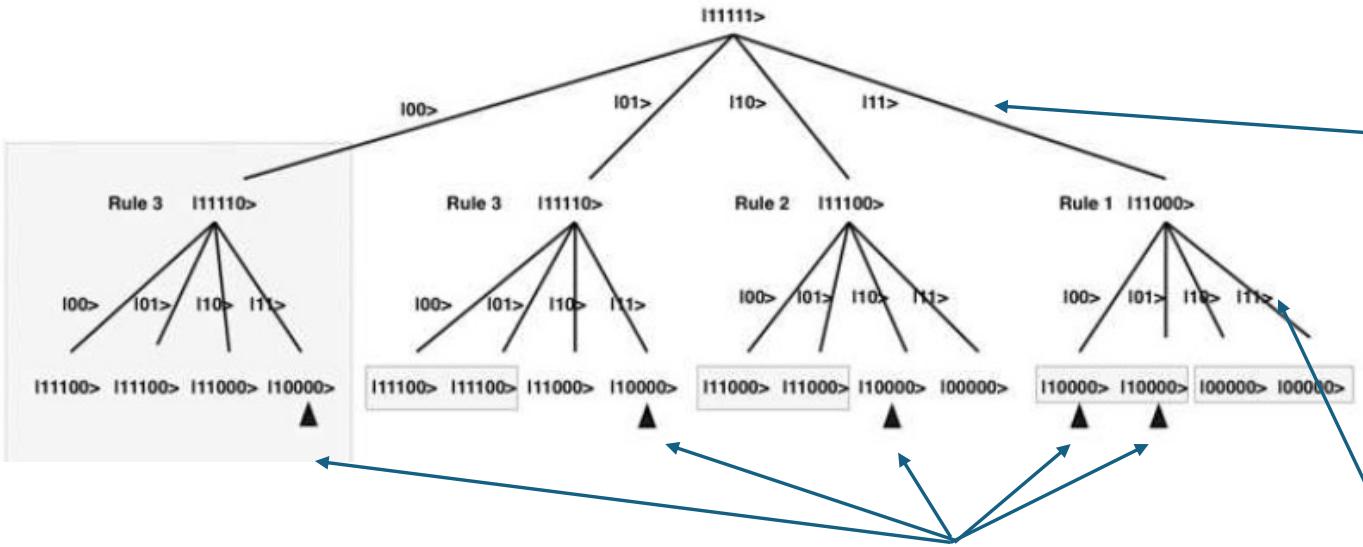
- Games involving two players represent one of the classic applications of symbolic problem solving
- Starting with some initial game position, the algorithm explores the tree of all legal moves down to the requested depth
- In quantum tree search *we cannot compare different states* that are *described by different path descriptors*
- **We cannot use the minimax-algorithm**
  - The comparison of alternative states is the basis of the minimax- algorithm.
  - The minimax-algorithm explores the tree of all legal moves down to the requested depth
  - Scores associated with leaves of the tree are calculated using an evaluation function.

Five Pennies Nim Game

- Two players alternate remove either one, two or three pennies from a stack that initially contains five pennies
- The player who pick up last penny loses
- The five pennies nim game is a zero game where neither player has any legal options
- The first player loses, and the second-player wins if correct moves are chosen.

The Nimatron was an electro-mechanical machine that played Nim. It was first exhibited in April–October 1940 by the Westinghouse Electric Corporation at the 1939-1940 New York World's Fair to entertain fair-goers

Five Pennies Nim Game of the Depth Two

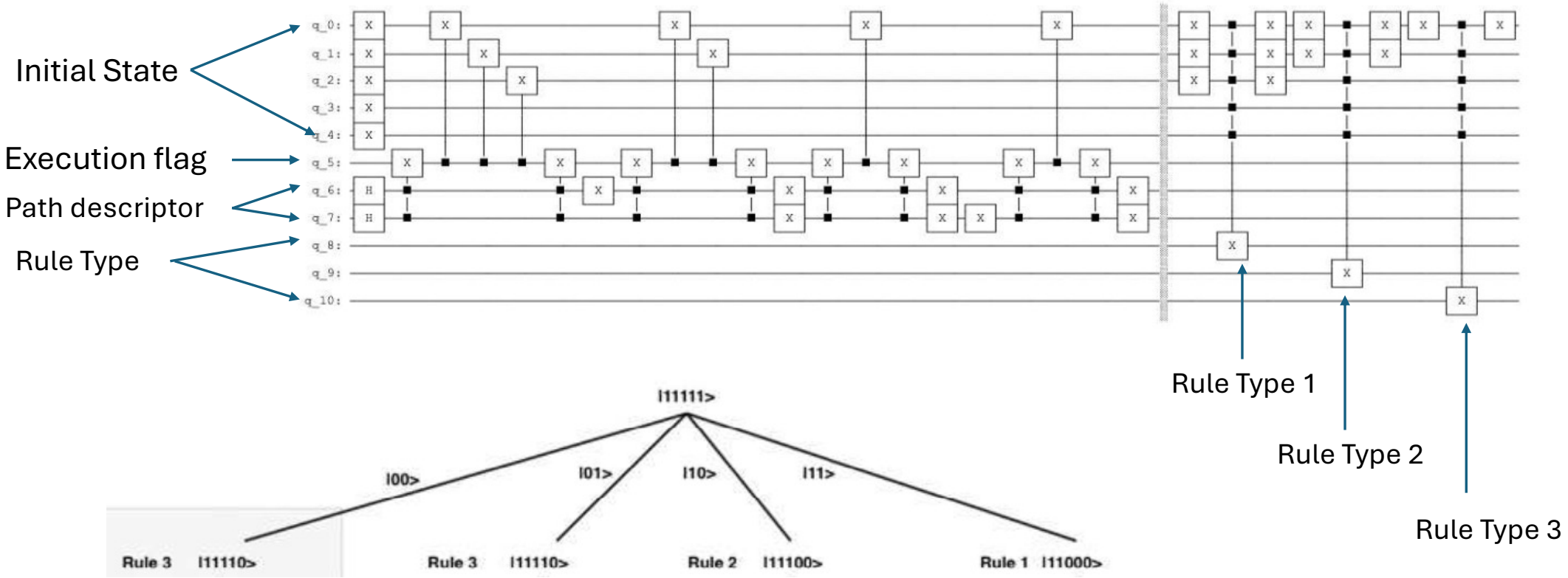


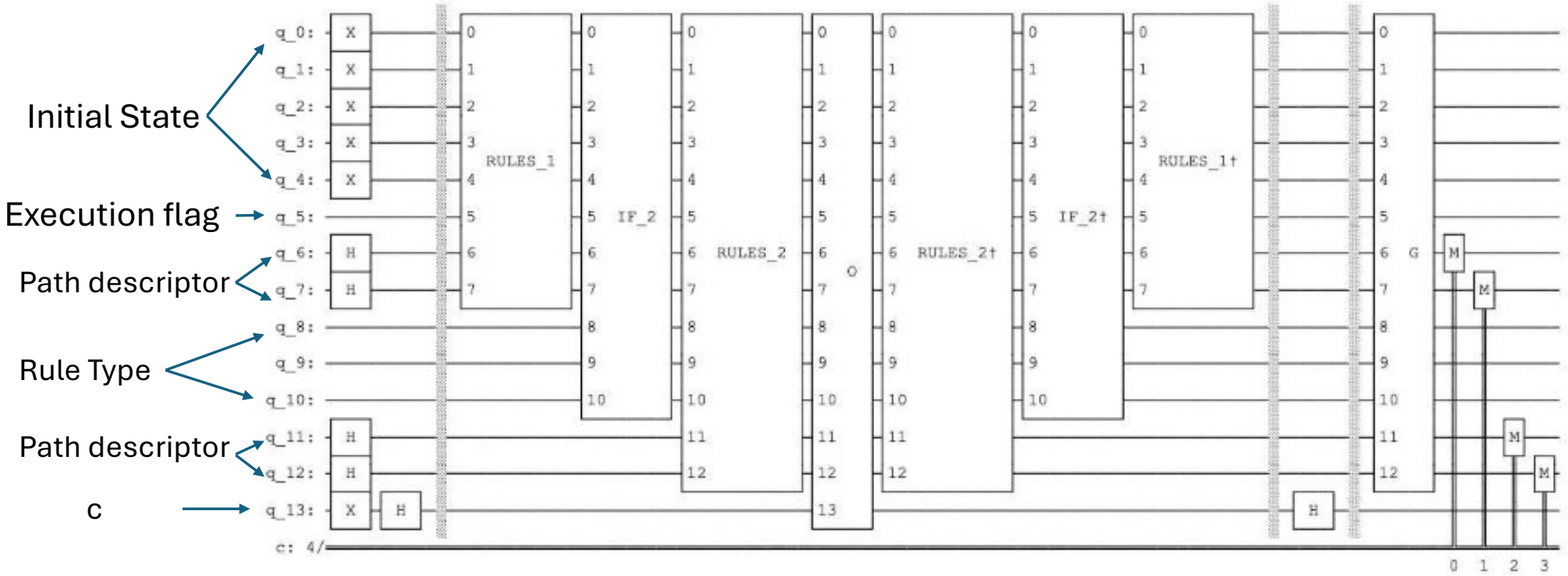
The resulting states corresponding to one penny |10000⟩ are marked by the oracle, they are indicated in the figure by a triangle.
They correspond to the loss of player A, since the player who picks up the last penny loses. Similar states and equal rules are marked by gray rectangles

- The player A starts the game, he can remove one, two or three pennies. The path descriptors are represented by two qubits |00⟩, |01⟩, |10⟩, |11⟩.
- The first rule removes three pennies, the second two and the third one penny
- Since we have four possible paths, the rule three and four are identical.

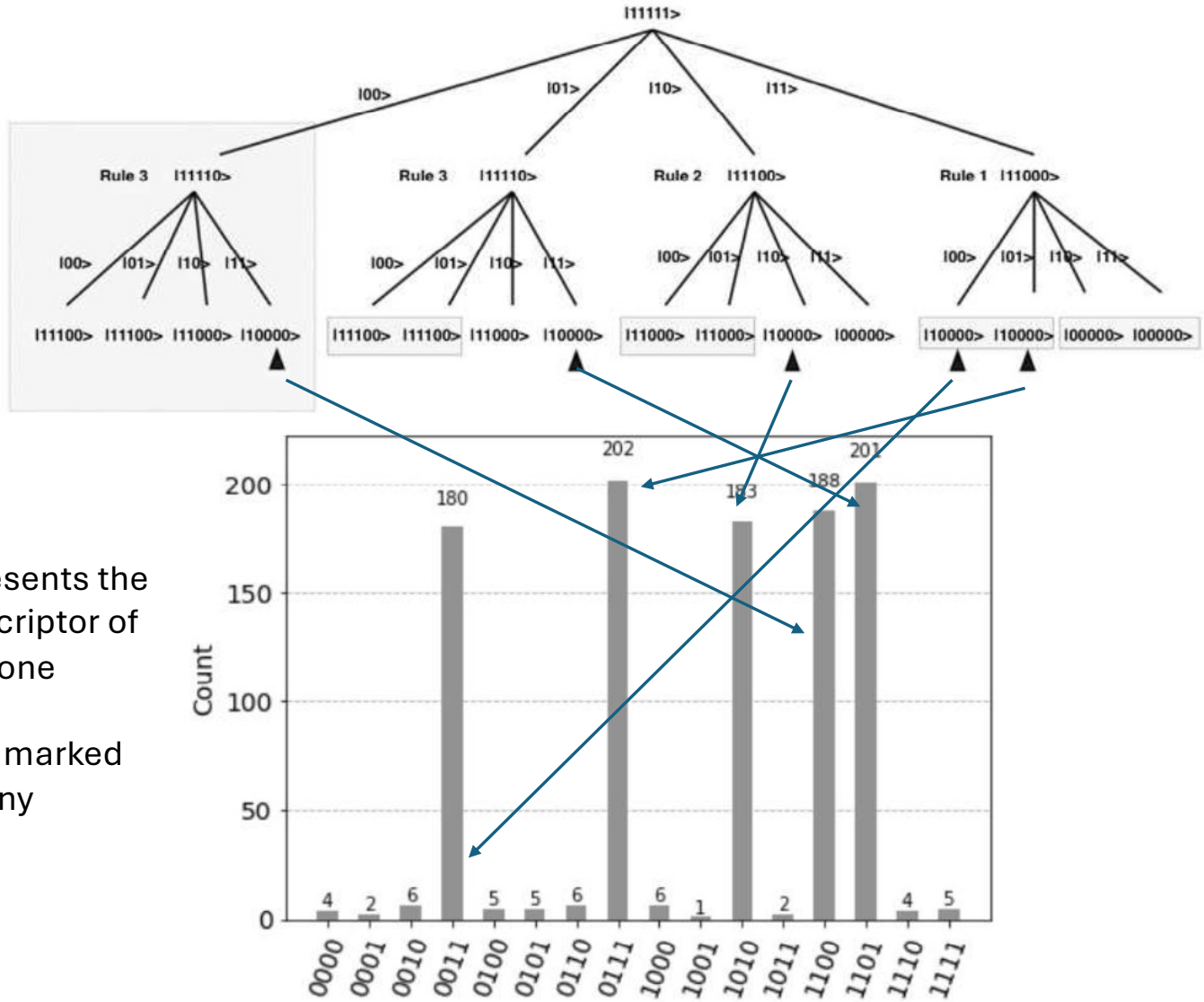
- The player B has either a stack with four, three or two pennies. Depending of the number of pennies, different rule types can be executed, either the type 1, type 2 or type 3
- Each of the three rules is instantiated with the path descriptors |00⟩, |01⟩, |10⟩, |11⟩.

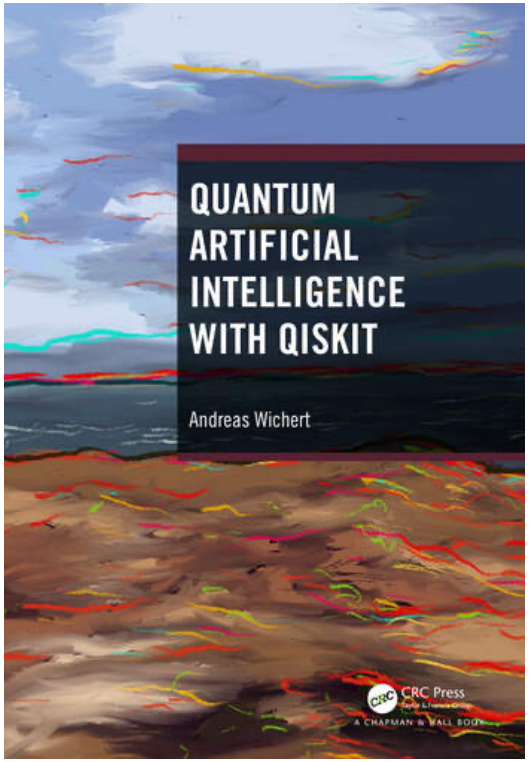
Player A


Initial State
Execution flag
Path descriptor
Rule Type
q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
Rule Type 1
Rule Type 2
Rule Type 3
|11111>
|00>
|01>
|10>
|11>
Rule 3 |11110>
Rule 3 |11110>
Rule 2 |11100>
Rule 1 |11000>


Initial State
Execution flag
Path descriptor
Rule Type
Path descriptor
c
q_0:
q_1:
q_2:
q_3:
q_4:
q_5:
q_6:
q_7:
q_8:
q_9:
q_10:
q_11:
q_12:
q_13:
c: 4/
X
H
RULES_1
IF_2
RULES_2
O
RULES_2†
IF_2†
RULES_1†
G
M
0 1 2 3

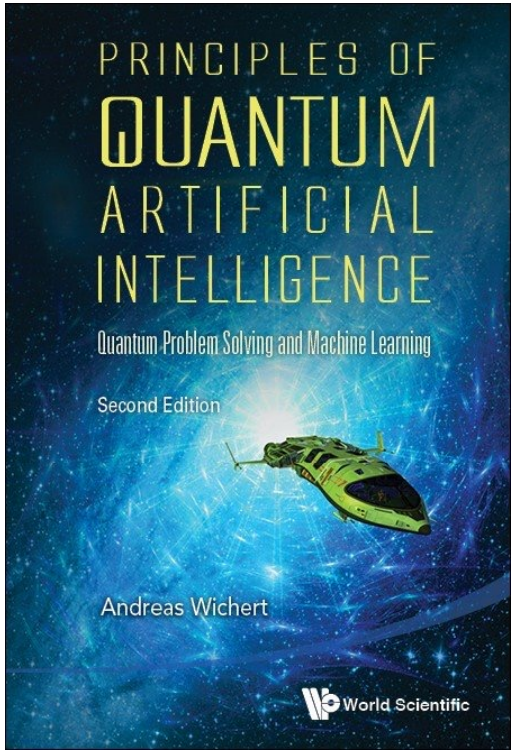
The histogram represents the measured path descriptor of the depth two after one rotation of Grover's amplification of the marked states with one penny





- Chapter 7
- Chapter 8
- Chapter 9
- Chapter 10
- Chapter 11
- Chapter 12

Quantum Artificial Intelligence with Qiskit, A. Wichert, Chapman and Hall/CRC, 2024



- Chapter 11

Principles of Quantum Artificial Intelligence: Quantum Problem Solving and Machine Learning, 2nd Edition, A. Wichert, World Scientific, 2020