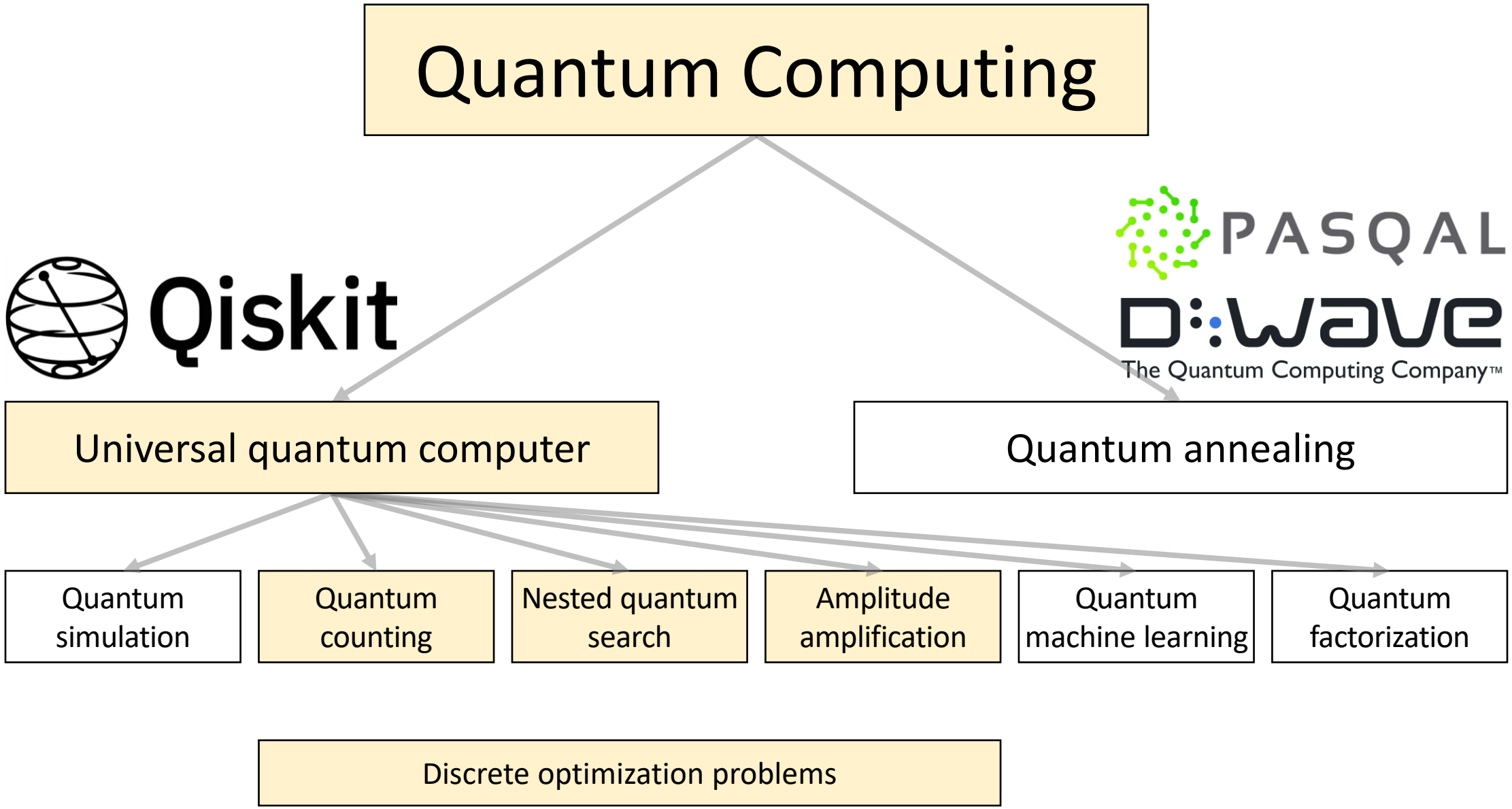
Discrete Optimization

Limitations of Existing Quantum Algorithms

Stefan Creemers
Luis Fernando Pérez

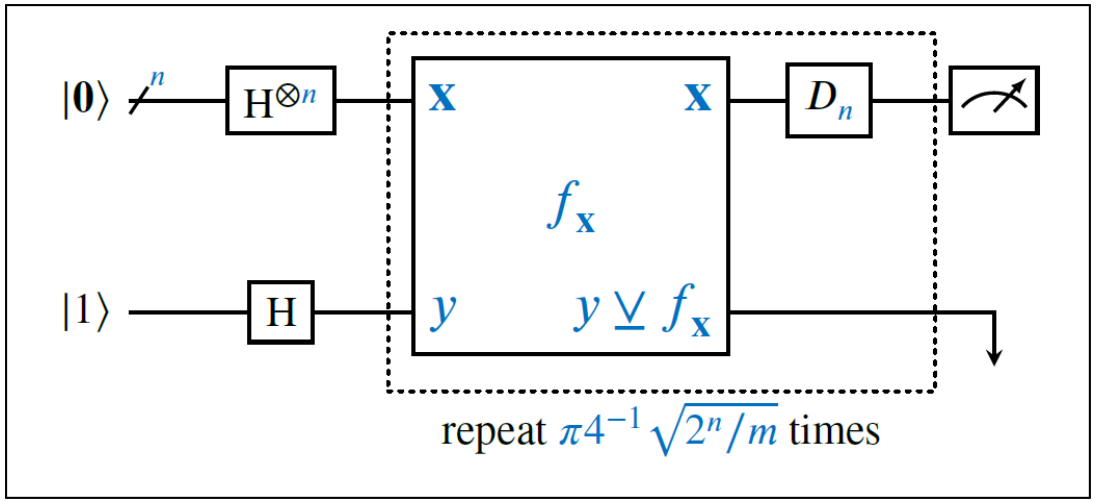
IÉSEG SCHOOL OF MANAGEMENT

KATHOLIEKE UNIVERSITEIT LEUVEN


Quantum Computing
Qiskit
PASQAL
D:wave
The Quantum Computing Company™
Universal quantum computer
Quantum annealing
Quantum simulation
Quantum counting
Nested quantum search
Amplitude amplification
Quantum machine learning
Quantum factorization
Discrete optimization problems

First however…

Grover's algorithm



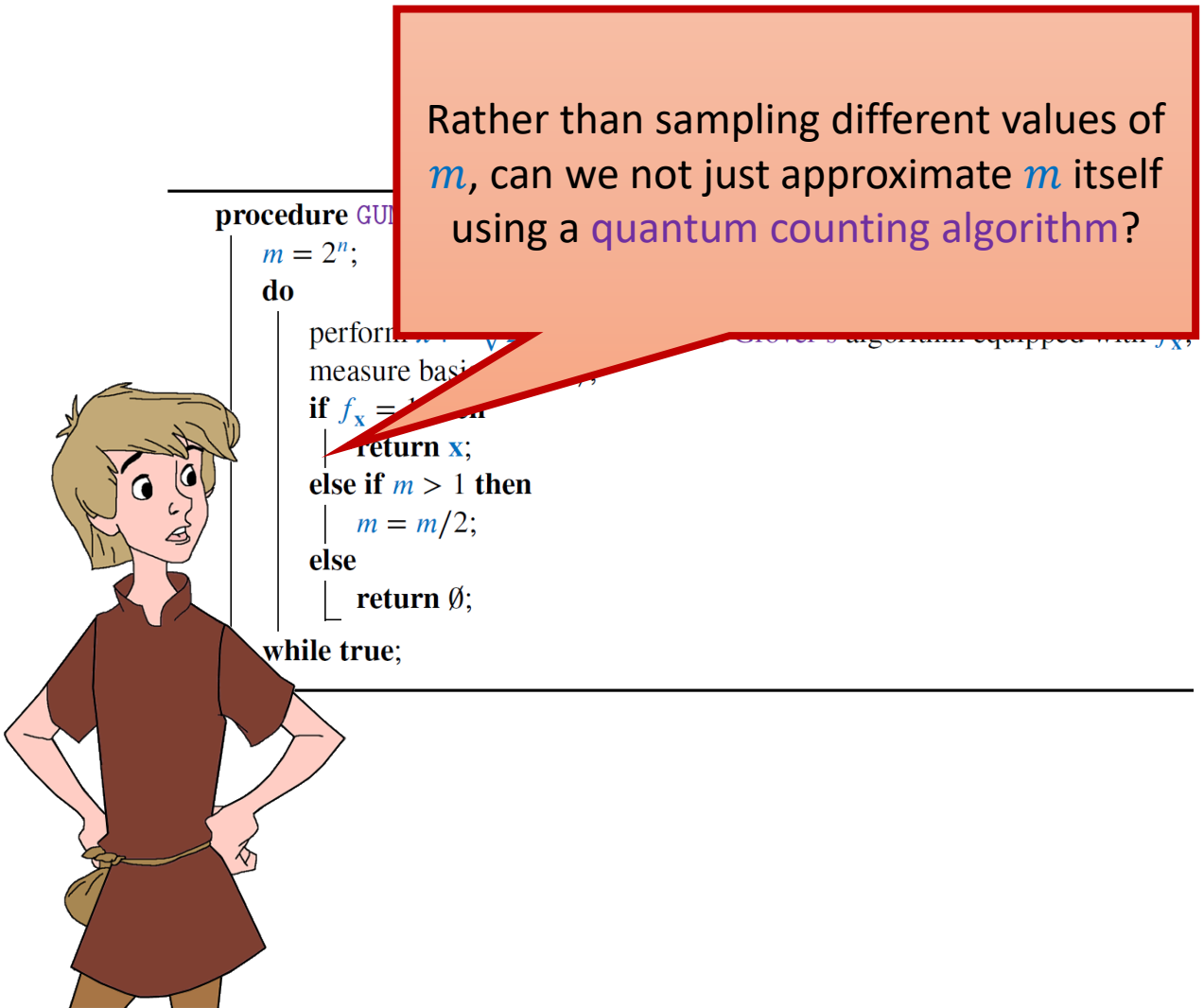
- Imagine that we want to solve a discrete optimization problem that has n decision variables.
- If decision variables are binary, there are 2^n solutions.
- Our goal is to find a valid solution $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ that respects all constraints of the optimization problem.
- To find a valid solution $\boldsymbol{x}$, we can use Grover's algorithm equipped with a function $f_{\boldsymbol{x}}$ that returns 1 if solution $\boldsymbol{x}$ is valid (and $f_{\boldsymbol{x}} = 0$ otherwise; i.e., $f_{\boldsymbol{x}}$ checks whether $\boldsymbol{x}$ respects all problem constraints).
- To find one of the m valid solutions, Grover's algorithm requires $\pi 4^{-1}\sqrt{2^n/m}$ iterations (and hence calls to function $f_{\boldsymbol{x}}$).
- Compared to a classical algorithm that requires up to 2^n calls to $f_{\boldsymbol{x}}$, Grover's algorithm achieves a quadratic speedup (this is also optimal).
- In general, however, we don't know m ➔ we need a procedure that finds a valid solution if m is unknown!

Grover's algorithm for unknown m (**GUM**)

procedure GUM(n)
  $m = 2^n$;
  do
    perform $\pi 4^{-1}\sqrt{2^n/m}$ iterations of Grover's algorithm equipped with $f_{\mathbf{x}}$;
    measure basis state $|\mathbf{x}\rangle$;
    if $f_{\mathbf{x}} = 1$ **then**
      return $\mathbf{x}$;
    else if $m > 1$ **then**
      $m = m/2$;
    else
      return $\emptyset$;
  while true;

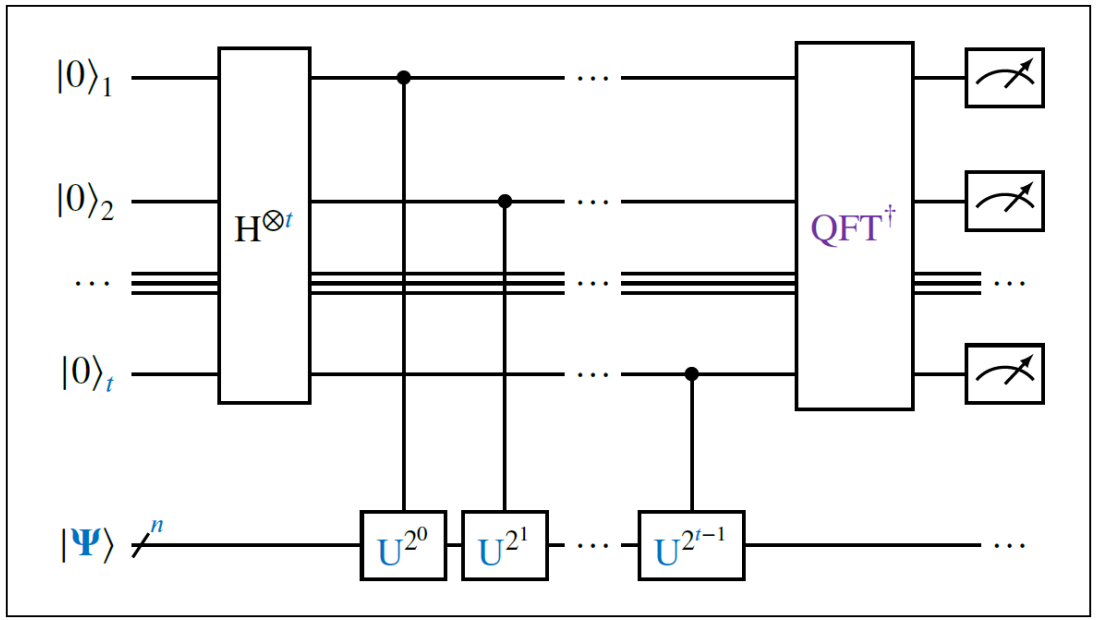
- If m is unknown, we can use Grover's algorithm for different values of m until we find a valid solution $\boldsymbol{x}$ (or until we are sufficiently certain that no valid solution can be found).
- If there are n decision variables, **GUM** tries up to $(n+1)$ values of m.
- We show that **GUM** requires $O(\sqrt{2^n})$ calls to function $f_{\boldsymbol{x}}$ in order to return one of the m valid solutions (but often has better than worst-case performance because a valid solution may already be found after a few iterations).
- Note that other approaches exist to sample different values of m (see e.g., Boyer et al. 1996).

Grover's algorithm for unknown m (**GUM**)



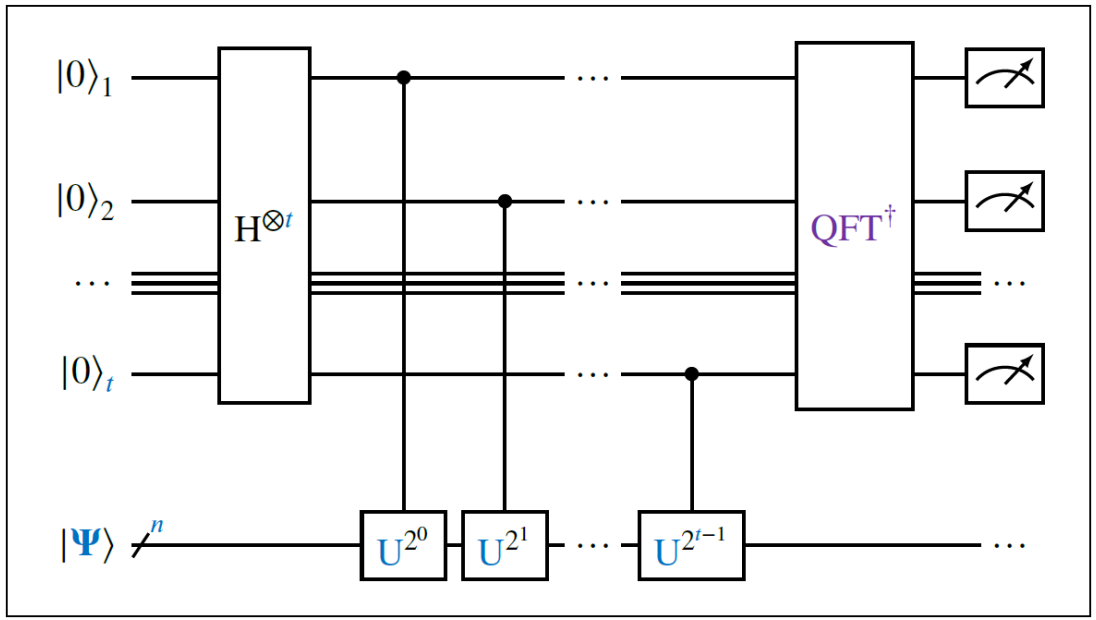
- If m is unknown, we can use Grover's algorithm for different values of m until we find a valid solution x (or until we are sufficiently certain that no valid solution can be found).
- If there are n decision variables, **GUM** tries up to $(n + 1)$ values of m.
- We show that **GUM** requires $O(\sqrt{2^n})$ calls to function f_x in order to return one of the m valid solutions (but often has better than worst-case performance because a valid solution may already be found after a few iterations).
- Note that other approaches exist to sample different values of m (see e.g., Boyer et al. 1996).

Quantum counting: Approximating m



- Let Ψ be a quantum system with n qubits that corresponds to a discrete optimization problem that has n binary decision variables.
- If we want to count the number of valid solutions m (i.e., the number of solutions x for which $f_x = 1$), we can use the quantum counting algorithm of Brassard et al. (1998).
- In this algorithm, we use Quantum Phase Estimation (QPE) to assess the change in the phase of system Ψ after applying unitary operator U on Ψ. The resulting estimate is stored in a set of t counting qubits and the precision of the estimate of m depends on the size of t. After applying an inverse quantum Fourier transform ($QFT^{\dagger}$) on the t counting qubits, we obtain m.

Quantum counting: Approximating m

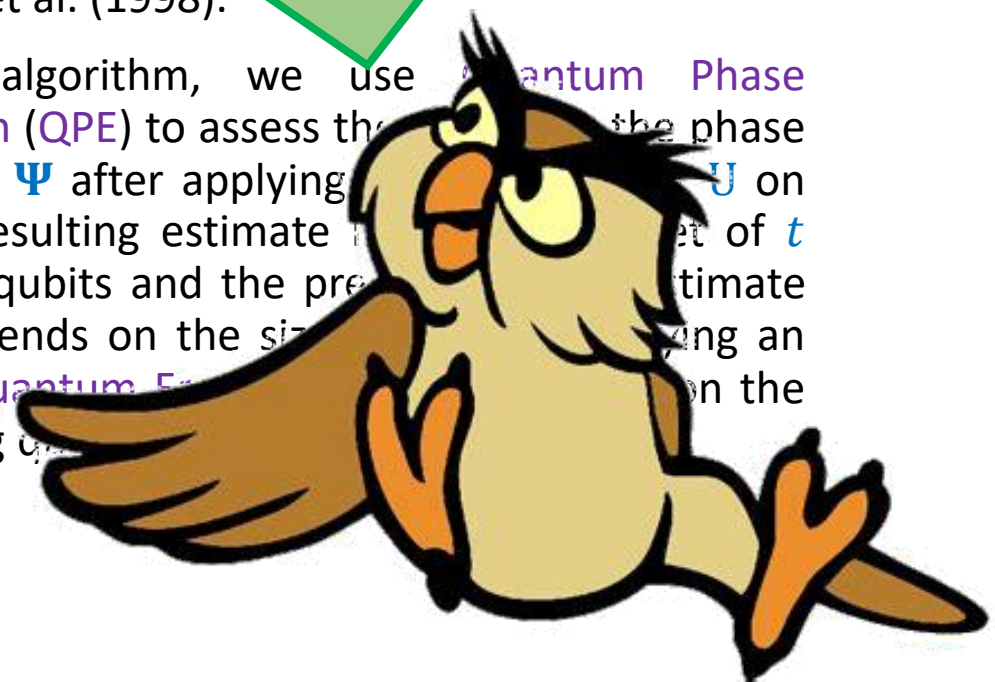


Note that QPE is also used by, for instance, Shor. In the case of Shor, U represents a modular operation, and U^i can be obtained efficiently by modular exponentiation.

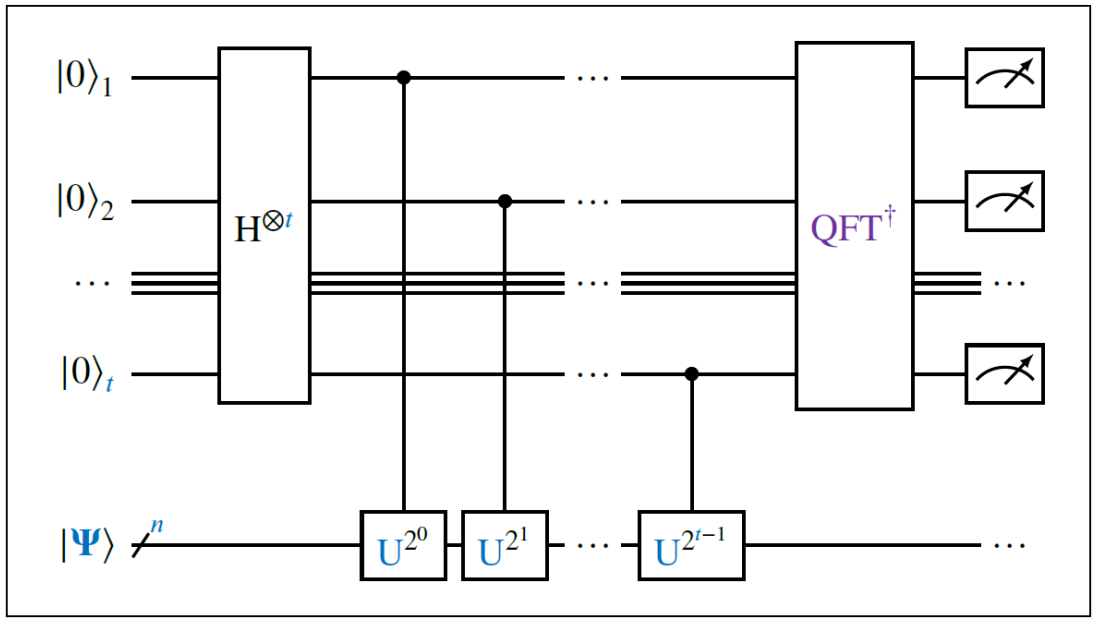
qubits that
n problem

d solutions
which $f_x =$
lgorithm of
Brassard et al. (1998).

- In this algorithm, we use antum Phase Estimation (QPE) to assess th the phase of system Ψ after applying U on Ψ. The resulting estimate et of t counting qubits and the pr timate of m depends on the si ng an inverse quantum F n the t counting q

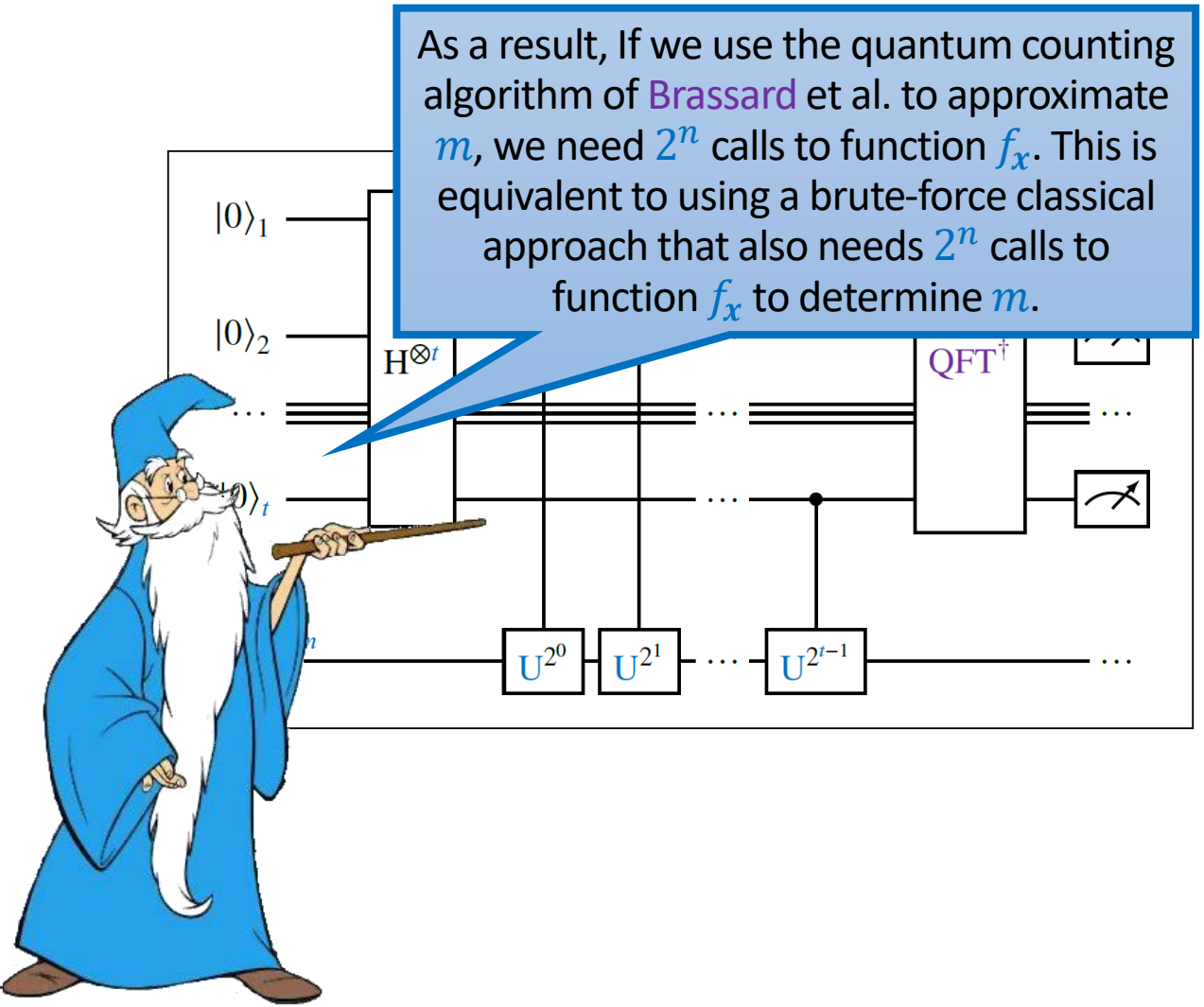


Quantum counting: Approximating m



- In our case, however, U corresponds to a single iteration of Grover's algorithm and exponentiation by squaring cannot be used to obtain U^i (if exponentiation by squaring could be used to efficiently obtain U^i, we could perform unstructured search in polynomial time).
- Instead, in order to obtain U^i, we effectively need to perform i Grover iterations.
- As a result, the number of Grover iterations required by the quantum counting algorithm of Brassard et al. is: $\sum_{i=0}^{t-1} 2^i \approx 2^t$.
- Unfortunately, to accurately approximate m, we roughly need $t = n$ counting qubits and, therefore, $2^t = 2^n$ Grover iterations (and hence calls to function f_x).

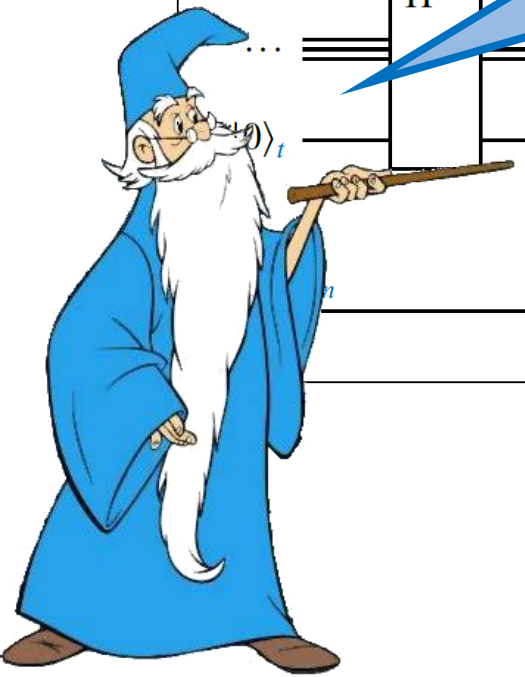
Quantum counting: Approximating m

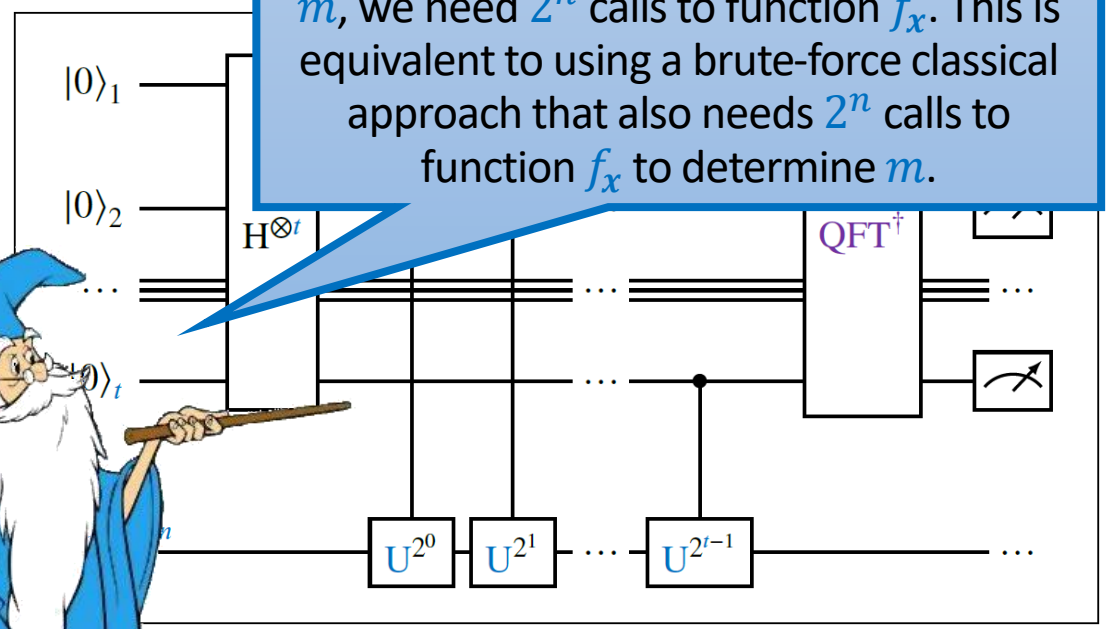


- In our case, however, U corresponds to a single iteration of Grover's algorithm and exponentiation by squaring cannot be used to obtain U^i (if exponentiation by squaring could be used to efficiently obtain U^i, we could perform unstructured search in polynomial time).
- Instead, in order to obtain U^i, we effectively need to perform i Grover iterations.
- As a result, the number of Grover iterations required by the quantum counting algorithm of Brassard et al. is: $\sum_{i=0}^{t-1} 2^i \approx 2^t$.
- Unfortunately, to accurately approximate m, we roughly need $t = n$ counting qubits and, therefore, $2^t = 2^n$ Grover iterations (and hence calls to function f_x).

Quantum counting: Approximating m

As a result, If we use the quantum counting algorithm of Brassard et al. to approximate m, we need 2^n calls to function f_x. This is equivalent to using a brute-force classical approach that also needs 2^n calls to function f_x to determine m.



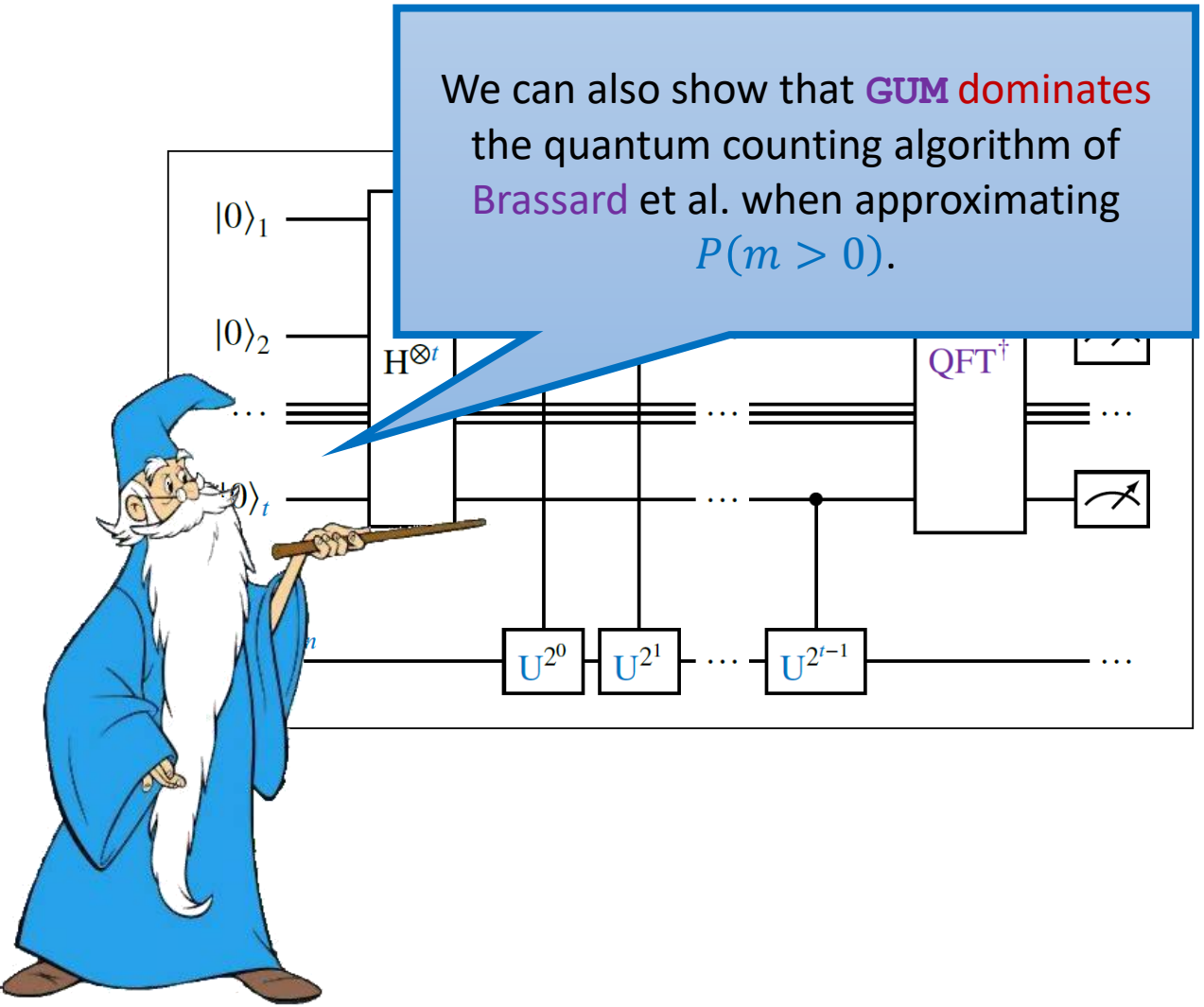


Is it really required, however, to accurately predict m? Perhaps it suffices to measure $P(m > 0)$.

to a single
onentiation
ain U^i (if
e used to
perform
.

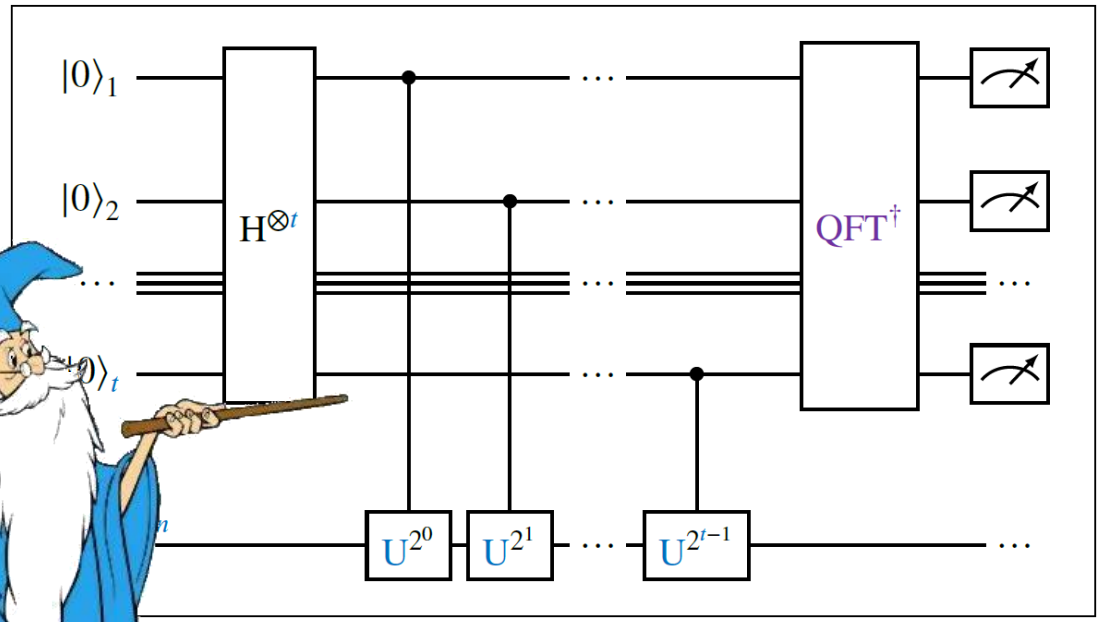
- Instead, in order to obtain effectively need to perform i Grover iterations.
- As a result, the number of Grover rations required by the quantum counting a hm of Brassard et al. is: $\sum_{i=0}^{t-1} 2^i \approx 2^t$.
- Unfortunately, to accurately approx we roughly need $t = n$ counting qubits a re, $2^t = 2^n$ Grover iterations (and h ls function f_x).

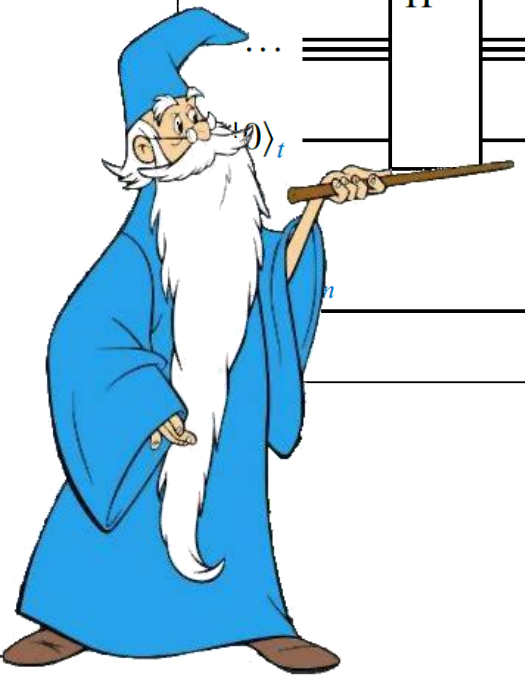
Quantum counting: Approximating m



- In our case, however, U corresponds to a single iteration of Grover's algorithm and exponentiation by squaring cannot be used to obtain U^i (if exponentiation by squaring could be used to efficiently obtain U^i, we could perform unstructured search in polynomial time).
- Instead, in order to obtain U^i, we effectively need to perform i Grover iterations.
- As a result, the number of Grover iterations required by the quantum counting algorithm of Brassard et al. is: $\sum_{i=0}^{t-1} 2^i \approx 2^t$.
- Unfortunately, to accurately approximate m, we roughly need $t = n$ counting qubits and, therefore, $2^t = 2^n$ Grover iterations (and hence calls to function f_x).

Quantum counting: Approximating m

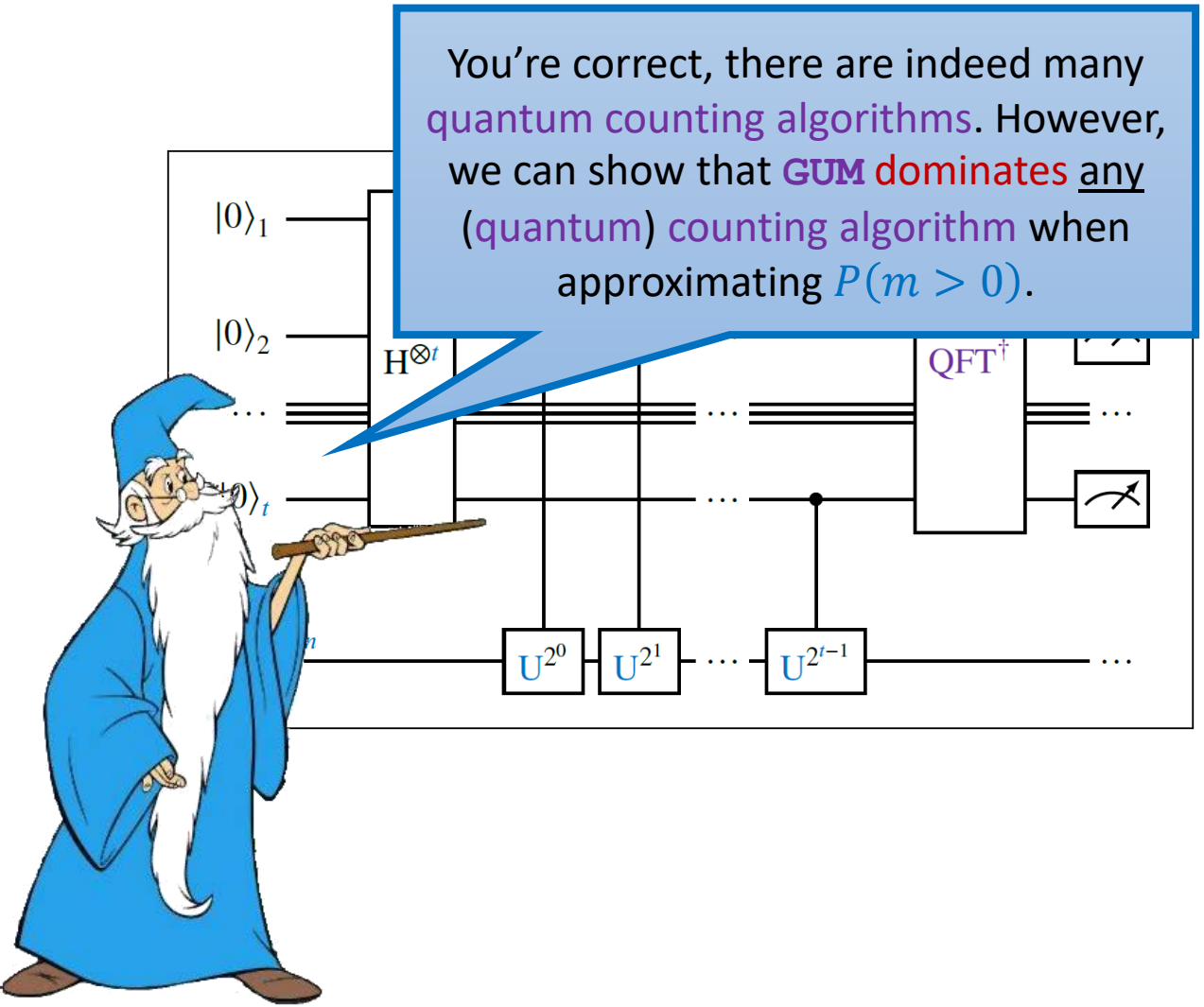




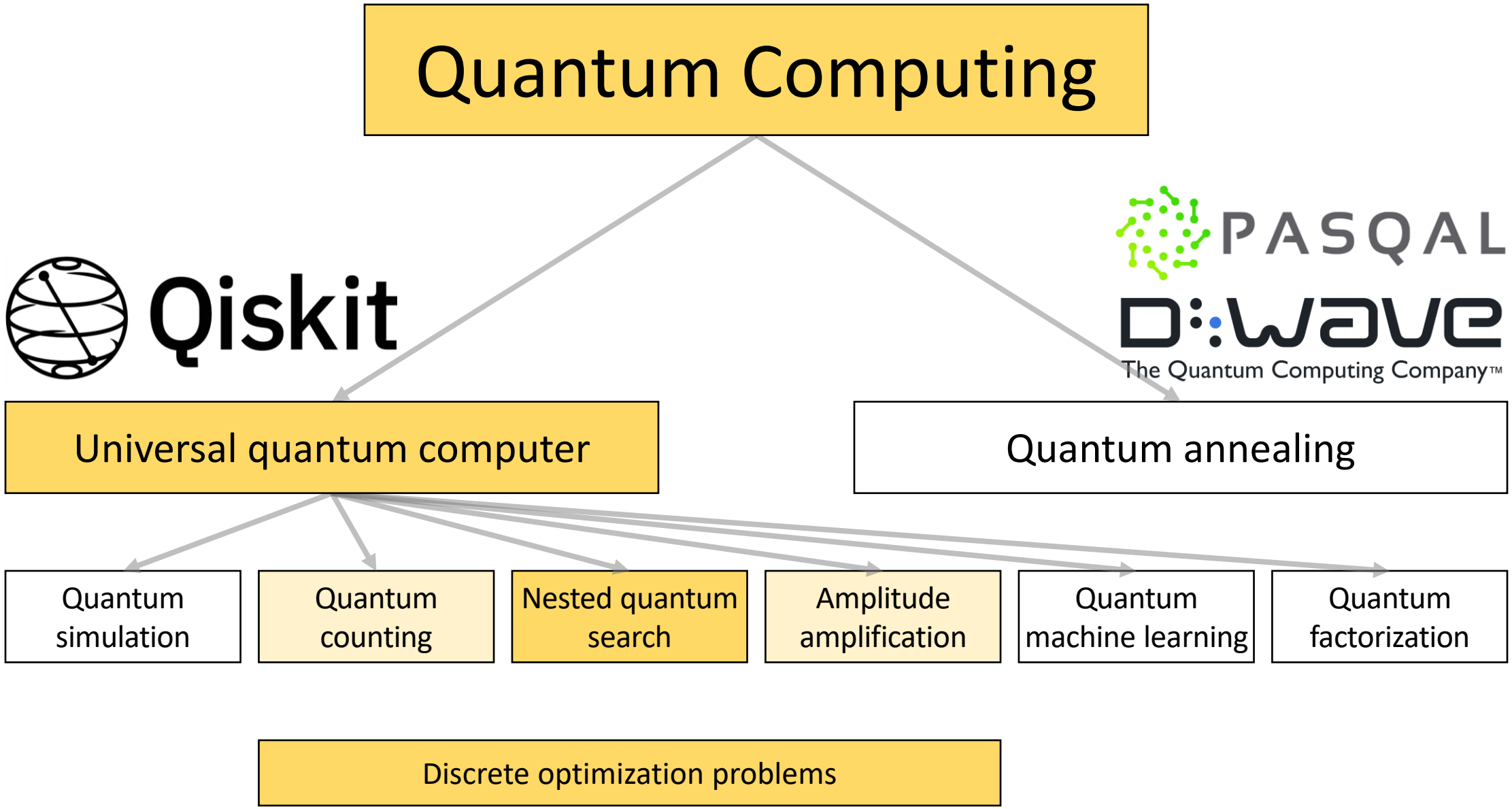
There are many quantum counting algorithms. For instance, the algorithm of Aaronson and Rall (2020), Suzuki et al. (2020), and Grinko et al. (2021). Do your results also hold for these algorithms?

- to a single onentiation tain U^i (if e used to perform .
- Instead, in order to obta… ve effectively need to perform i Grover iterations.
- As a result, the number of Gr… required by the quantum countin… Brassard et al. is: $\sum_{i=0}^{t-1} 2^i \approx 2^t$.
- Unfortunately, to accurately approx… roughly need $t = n$ counting qubits … $2^t = 2^n$ Grover iterations (and … function f_x).

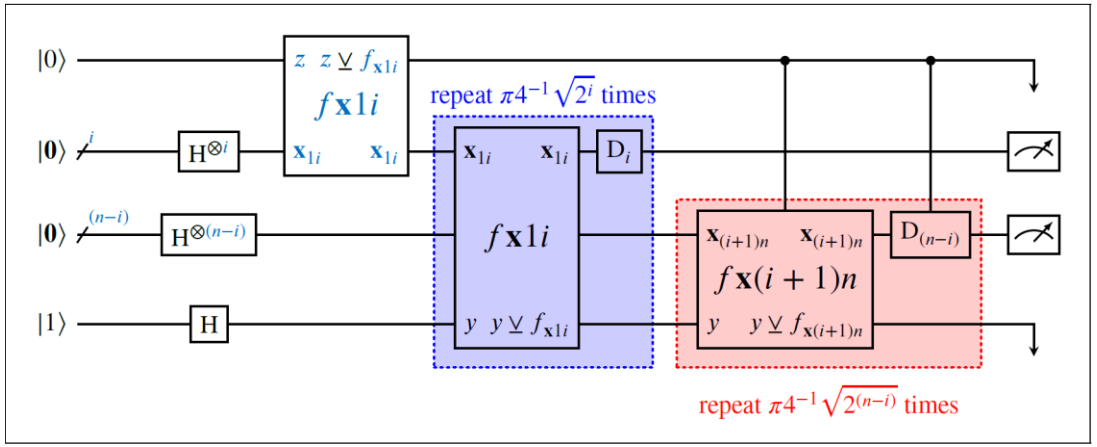
Quantum counting: Approximating m



- In our case, however, U corresponds to a single iteration of Grover's algorithm and exponentiation by squaring cannot be used to obtain U^i (if exponentiation by squaring could be used to efficiently obtain U^i, we could perform unstructured search in polynomial time).
- Instead, in order to obtain U^i, we effectively need to perform i Grover iterations.
- As a result, the number of Gr required by the quantum countin Brassard et al. is: $\sum_{i=0}^{t-1} 2^i \approx 2^t$.
- Unfortunately, to accurately approx roughly need $t = n$ counting qubits $2^t = 2^n$ Grover iterations (and function f_x).

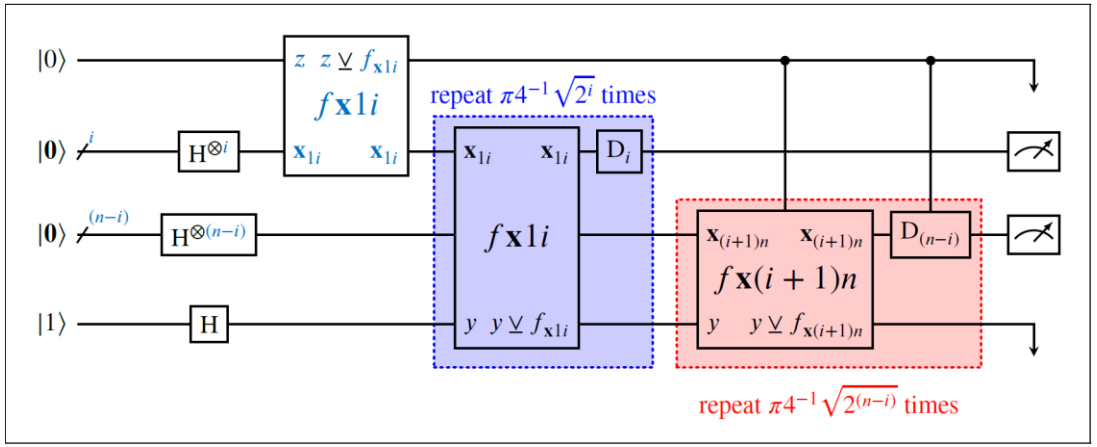

Quantum Computing
Qiskit
PASQAL
D:wave
The Quantum Computing Company™
Universal quantum computer
Quantum annealing
Quantum simulation
Quantum counting
Nested quantum search
Amplitude amplification
Quantum machine learning
Quantum factorization
Discrete optimization problems

Nested quantum search



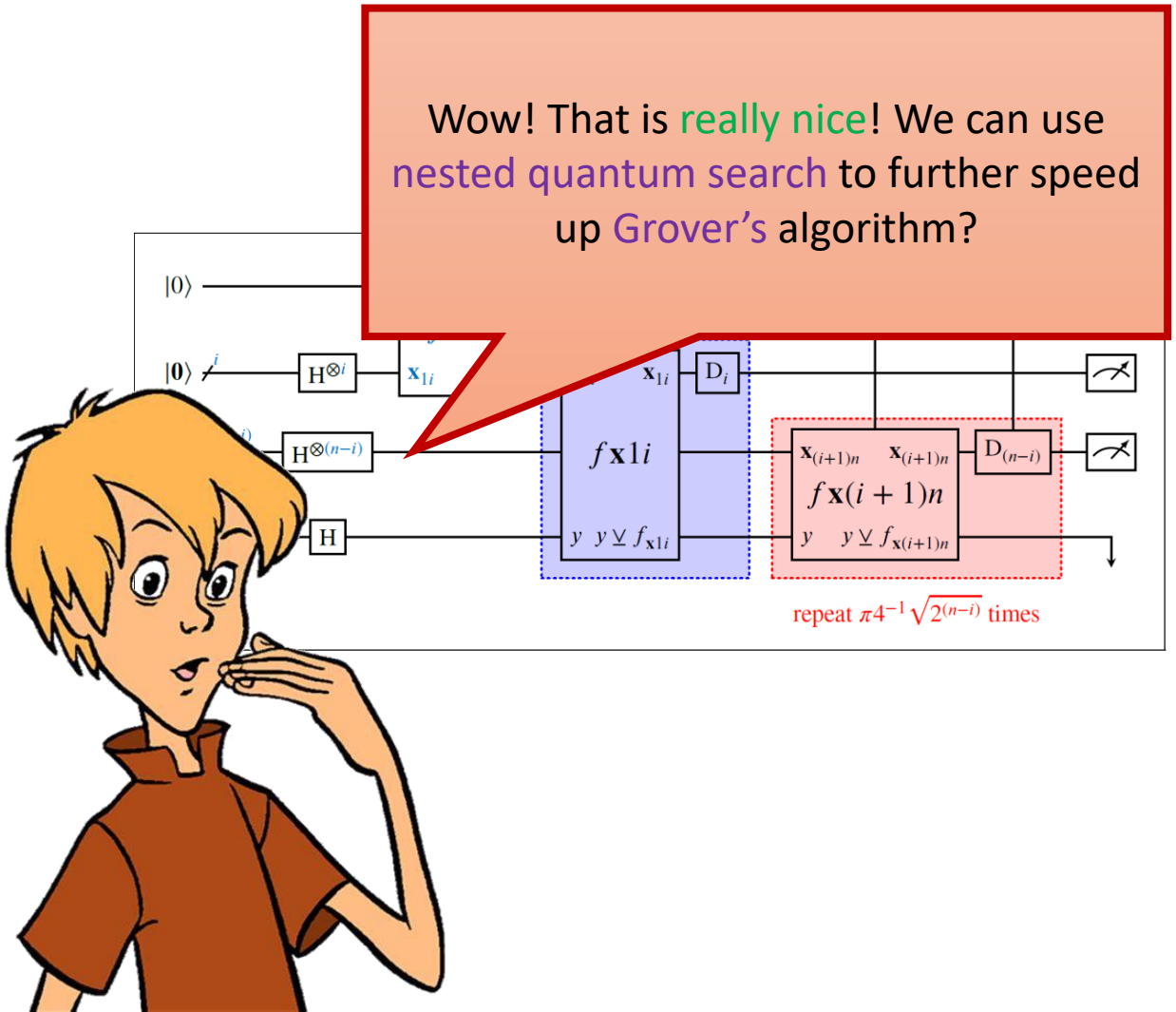
- Rather than searching the entire solution space, a nested search ``nests'' one search within another. This way, partial solutions (that are obtained efficiently) are used to build other partial solutions that in turn are used to build an optimal solution.
- Classical nested search requires $O(2^{\gamma n})$ calls to a function f_x, where γ is some number less than 1 that depends on the level of nesting as well as the characteristics of the problem instance.
- Cerf et al. (2000) proposed the idea of a nested quantum search that has complexity $O\left(\sqrt{2^{\gamma n}}\right)$.
- To illustrate the quantum nesting algorithm of Cerf et al., let's try to find a single valid solution (i.e., $m = 1$) in a set of 2^n solutions. To keep things simple, we assume a single level of nesting.

Nested quantum search



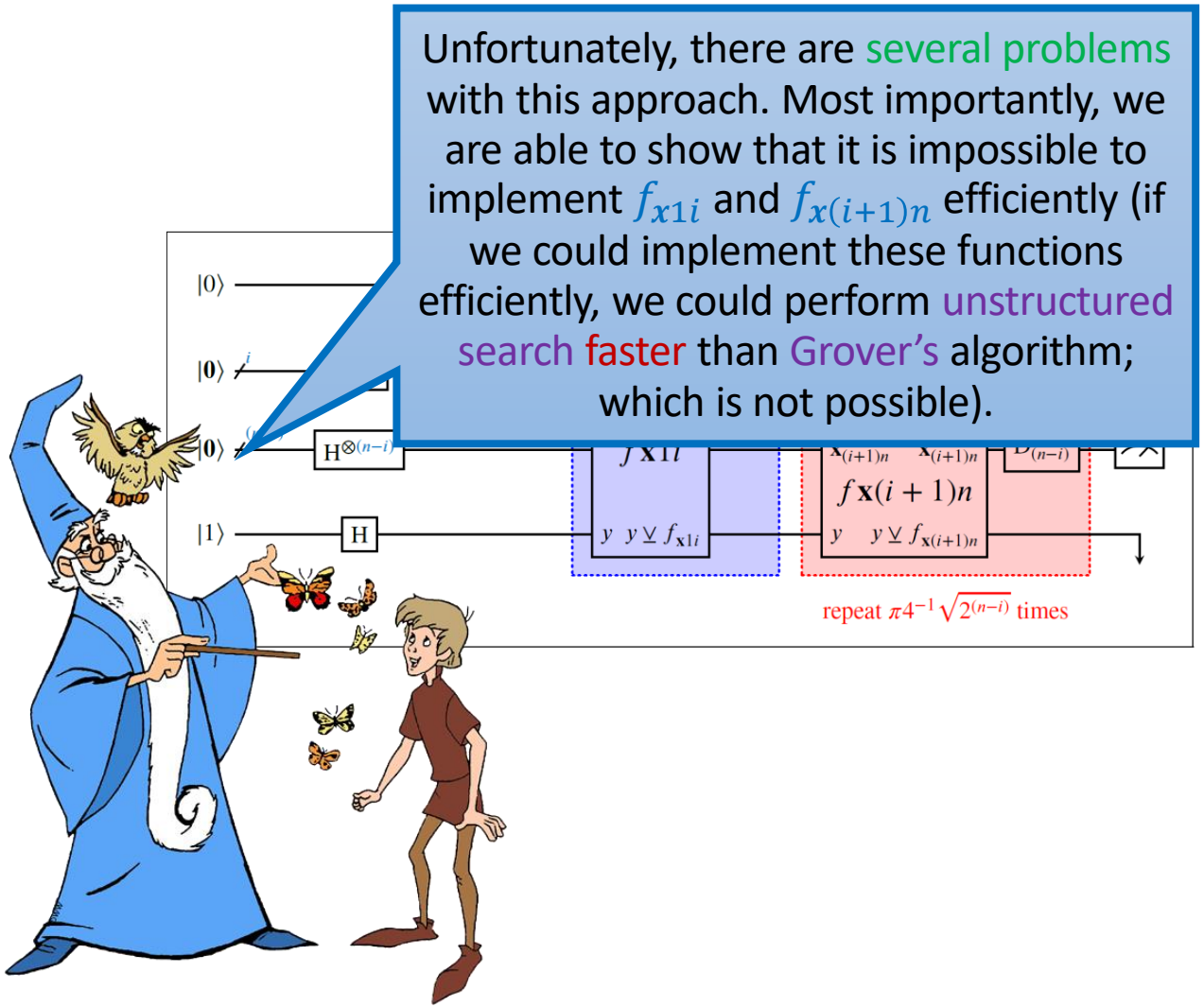
- The required circuit is given on the left. In this circuit:
  - $f_{x1i} = 1$ if $\boldsymbol{x}_{1i} = \{x_1, \dots, x_i\}$ corresponds to the first i decision variables of the optimal solution.
  - $f_{x(i+1)n} = 1$ if $\boldsymbol{x}_{(i+1)n} = \{x_{(i+1)}, \dots, x_n\}$ corresponds to the last $(n-i)$ decision variables of the optimal solution.
  - The second part of the circuit (in red) is only executed for those basis states for which $f_{x1i} = 1$.
  - The proposed speedup originates from:
    - Less Grover iterations are needed in total (i.e., $\sqrt{2^i} + \sqrt{2^{(n-i)}} \leq \sqrt{2^n}$).
    - Grover iterations are performed on smaller systems (with less qubits; i.e., systems with i and $(n-i)$ qubits rather than one big system that has n qubits).

Nested quantum search



- The required circuit is given on the left. In this circuit:
  - $f_{x1i} = 1$ if $\boldsymbol{x}_{1i} = \{x_1, \dots, x_i\}$ corresponds to the first i decision variables of the optimal solution.
  - $f_{x(i+1)n} = 1$ if $\boldsymbol{x}_{(i+1)n} = \{x_{(i+1)}, \dots, x_n\}$ corresponds to the last $(n-i)$ decision variables of the optimal solution.
  - The second part of the circuit (in red) is only executed for those basis states for which $f_{x1i} = 1$.
  - The proposed speedup originates from:
    - Less Grover iterations are needed in total (i.e., $\sqrt{2^i} + \sqrt{2^{(n-i)}} \leq \sqrt{2^n}$).
    - Grover iterations are performed on smaller systems (with less qubits; i.e., systems with i and $(n-i)$ qubits rather than one big system that has n qubits).

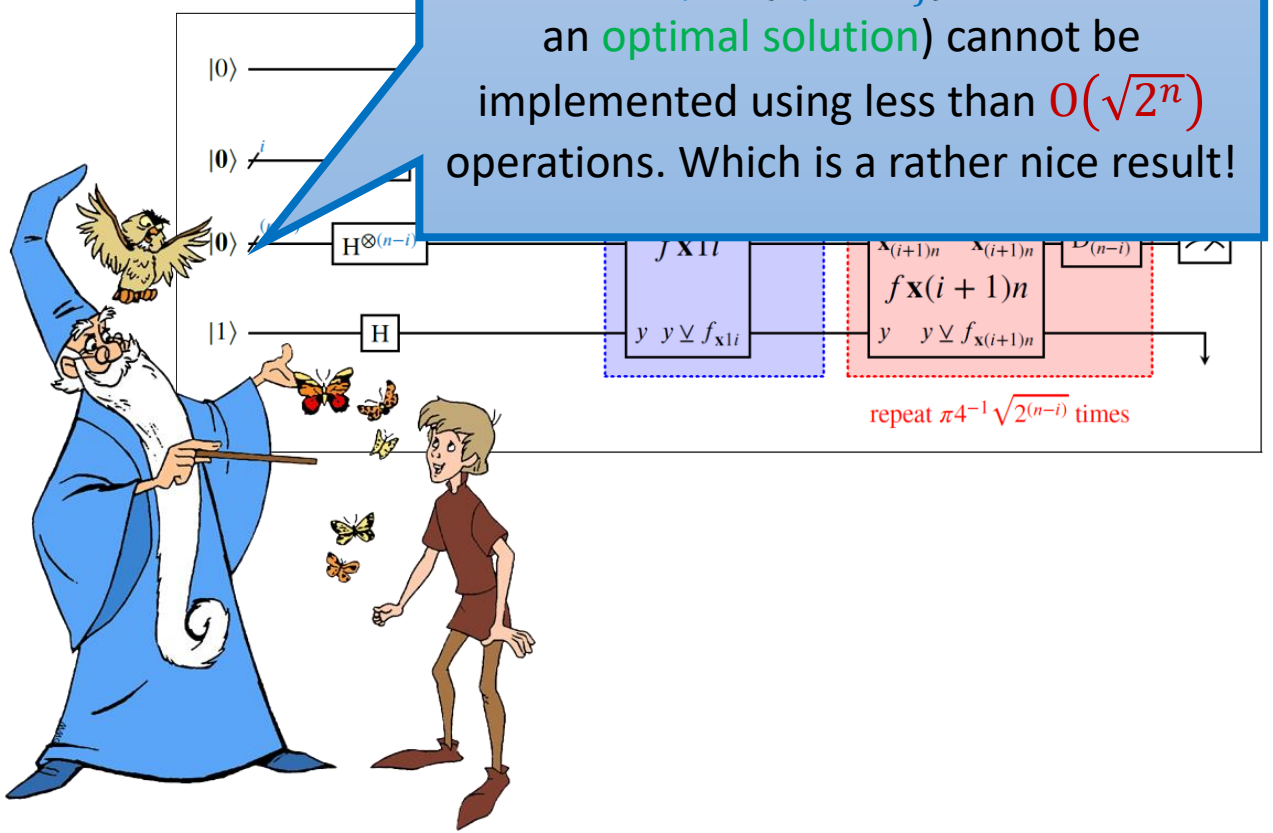
Nested quantum search



- The required circuit is given on the left. In this circuit:
  - $f_{x1i} = 1$ if $\boldsymbol{x}_{1i} = \{x_1, \dots, x_i\}$ corresponds to the first i decision variables of the optimal solution.
  - $f_{x(i+1)n} = 1$ if $\boldsymbol{x}_{(i+1)n} = \{x_{(i+1)}, \dots, x_n\}$ corresponds to the last $(n-i)$ decision variables of the optimal solution.
  - The second part of the circuit (in red) is only executed for those basis states for which $f_{x1i} = 1$.
  - The proposed speedup originates from:
    - Less Grover iterations are needed in total (i.e., $\sqrt{2^i} + \sqrt{2^{(n-i)}} \leq \sqrt{2^n}$).
    - Grover iterations are performed on smaller systems (with less qubits; i.e., systems with i and $(n-i)$ qubits rather than one big system that has n qubits).

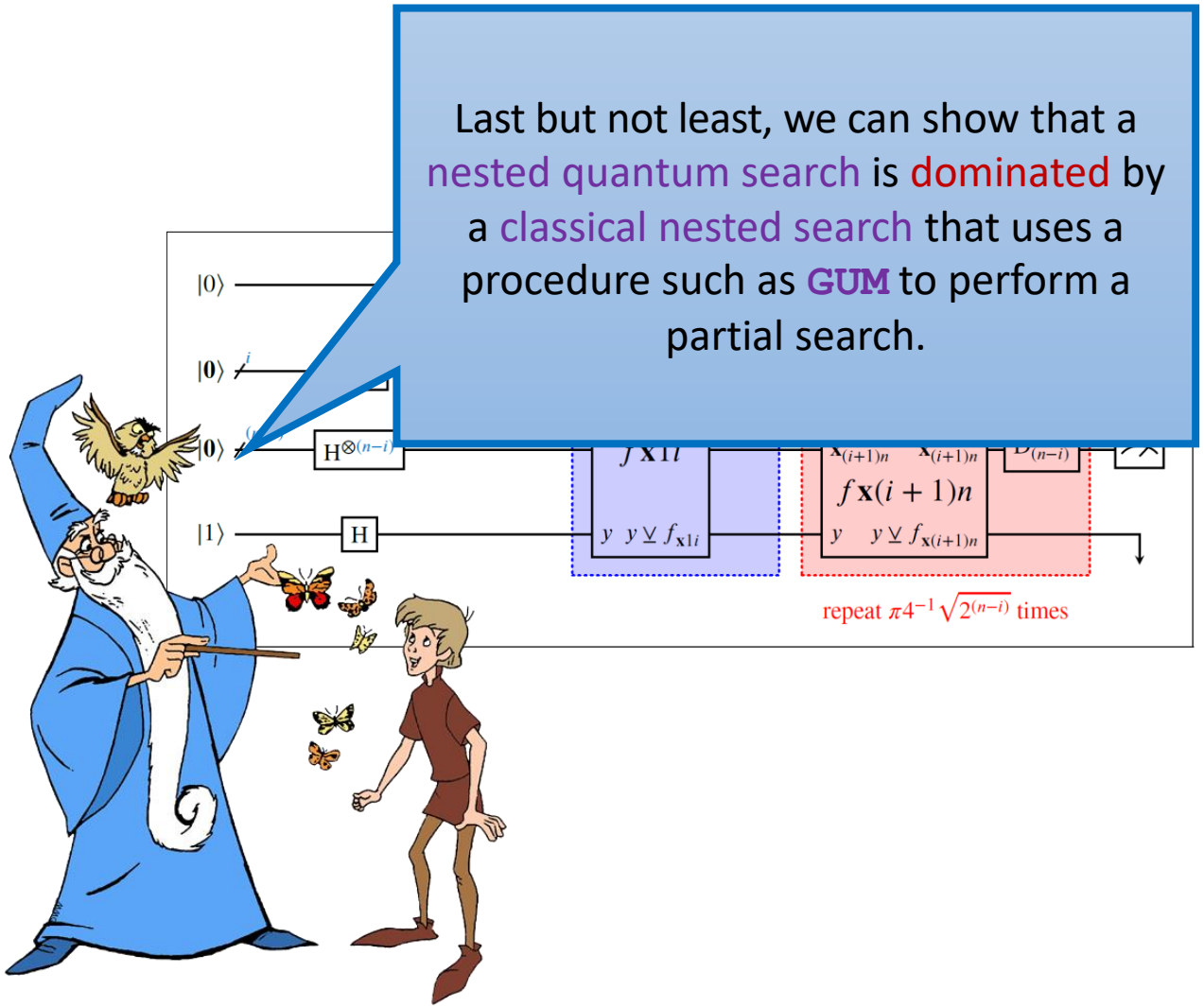
Nested quantum search

In fact, we show that a general function f_{xij} (that evaluates whether decision variables $\boldsymbol{x}_{1i} = \{x_i, \ldots, x_j\}$ correspond to an optimal solution) cannot be implemented using less than $O(\sqrt{2^n})$ operations. Which is a rather nice result!

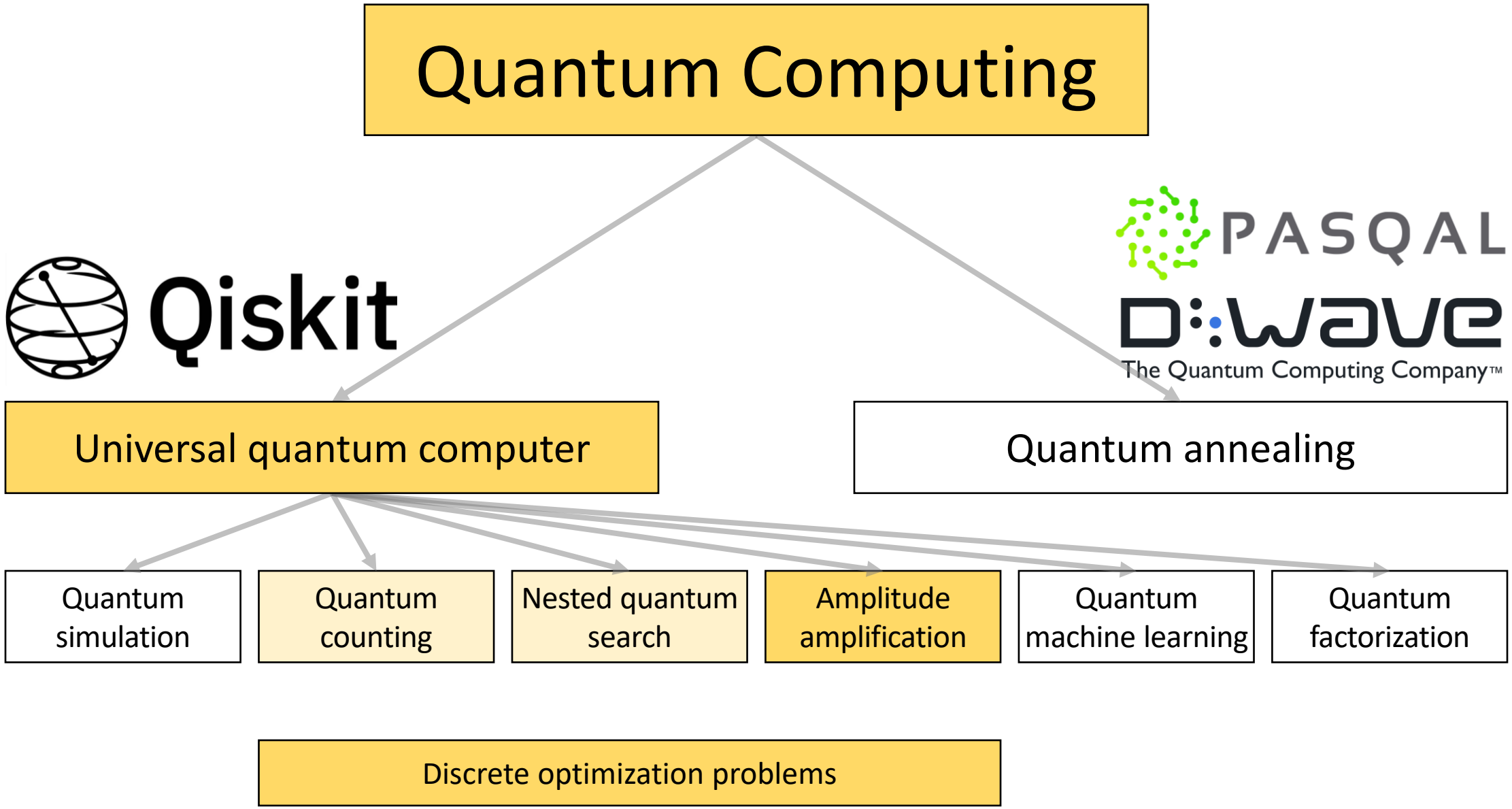


- The required circuit is given on the left. In this circuit:
  - $f_{x1i} = 1$ if $\boldsymbol{x}_{1i} = \{x_1, \ldots, x_i\}$ corresponds to the first i decision variables of the optimal solution.
  - $f_{x(i+1)n} = 1$ if $\boldsymbol{x}_{(i+1)n} = \{x_{(i+1)}, \ldots, x_n\}$ corresponds to the last $(n-i)$ decision variables of the optimal solution.
  - The second part of the circuit (in red) is only executed for those basis states for which $f_{x1i} = 1$.
  - The proposed speedup originates from:
    - Less Grover iterations are needed in total (i.e., $\sqrt{2^i} + \sqrt{2^{(n-i)}} \leq \sqrt{2^n}$).
    - Grover iterations are performed on smaller systems (with less qubits; i.e., systems with i and $(n-i)$ qubits rather than one big system that has n qubits).

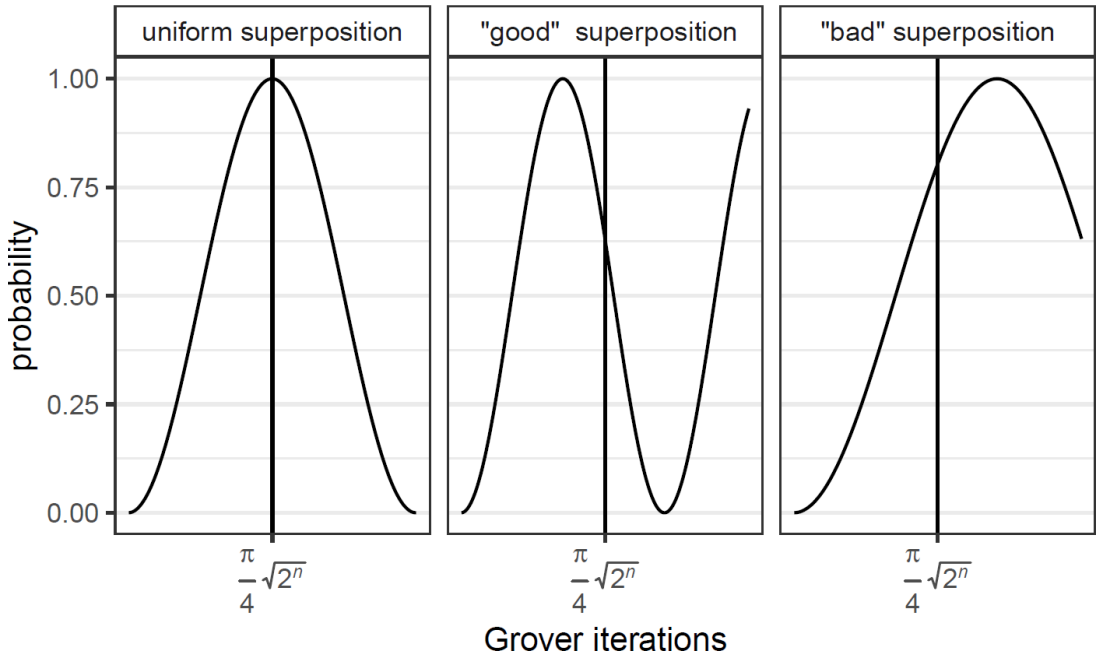
Nested quantum search



- The required circuit is given on the left. In this circuit:
  - $f_{x1i} = 1$ if $\boldsymbol{x}_{1i} = \{x_1, \dots, x_i\}$ corresponds to the first i decision variables of the optimal solution.
  - $f_{x(i+1)n} = 1$ if $\boldsymbol{x}_{(i+1)n} = \{x_{(i+1)}, \dots, x_n\}$ corresponds to the last $(n-i)$ decision variables of the optimal solution.
  - The second part of the circuit (in red) is only executed for those basis states for which $f_{x1i} = 1$.
  - The proposed speedup originates from:
    - Less Grover iterations are needed in total (i.e., $\sqrt{2^i} + \sqrt{2^{(n-i)}} \leq \sqrt{2^n}$).
    - Grover iterations are performed on smaller systems (with less qubits; i.e., systems with i and $(n-i)$ qubits rather than one big system that has n qubits).


Quantum Computing
Qiskit
PASQAL
D:wave
The Quantum Computing Company™
Universal quantum computer
Quantum annealing
Quantum simulation
Quantum counting
Nested quantum search
Amplitude amplification
Quantum machine learning
Quantum factorization
Discrete optimization problems

Amplitude amplification: Concept



- Grover's algorithm initializes a system of n qubits using a uniform superposition where all 2^n solutions have an equal probability of being measured.
- Given this initial uniform superposition, Grover's algorithm needs $\pi 4^{-1}\sqrt{2^n/m}$ iterations to find one of the m valid solutions.
- Amplitude amplification tries to look for better initial superpositions such that less iterations (and hence less calls to function f_x) are required to find a valid solution.
- Good news: these initial superpositions do exist!
- Bad news: we perform a number of experiments to show that it may not be that easy to identify these "good" superpositions.

Conclusions

- Quantum computing may perhaps cause a revolution in the field of discrete optimization. However, this revolution will probably not involve:
  - Quantum counting algorithms.
  - Nested quantum search algorithms.
  - Amplitude amplification.
- The detailed results of this study are available on SSRN and on my personal website (www.cromso.com).
- If you have any further questions, contact us:
  - sc@cromso.com
  - l.fernando@ieseg.fr

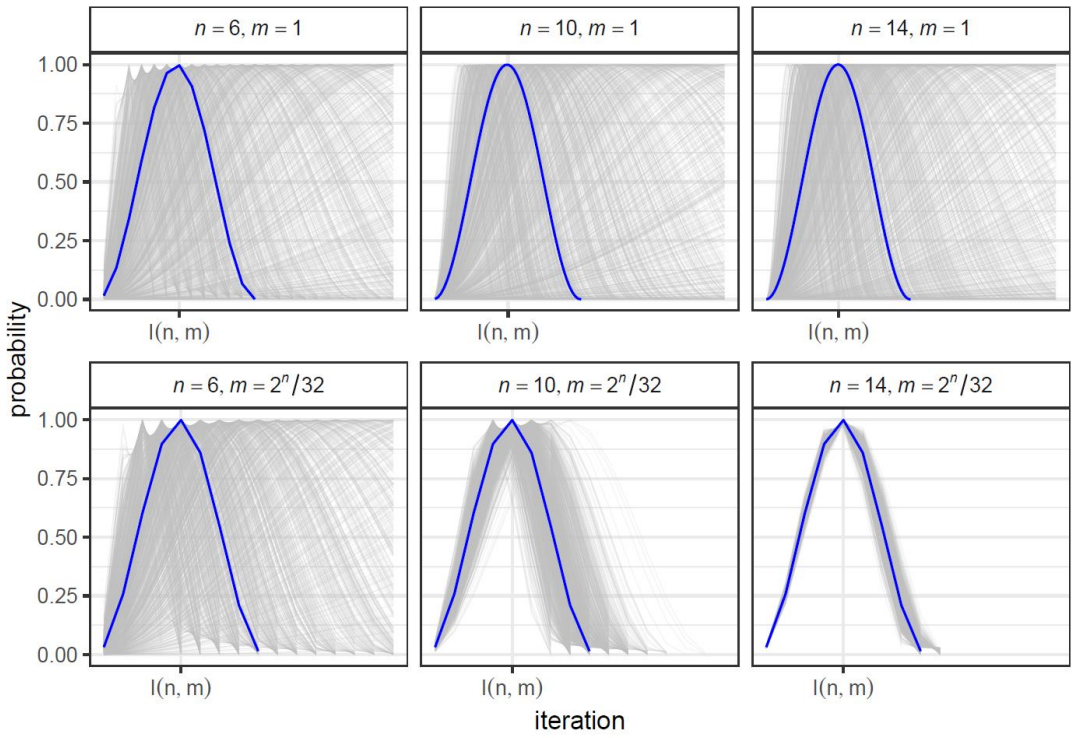
EURO 2024 Copenhagen:
Session on quantum computing

Invitation code:
7586e1c4

Stream:
Quantum Computing
Optimization

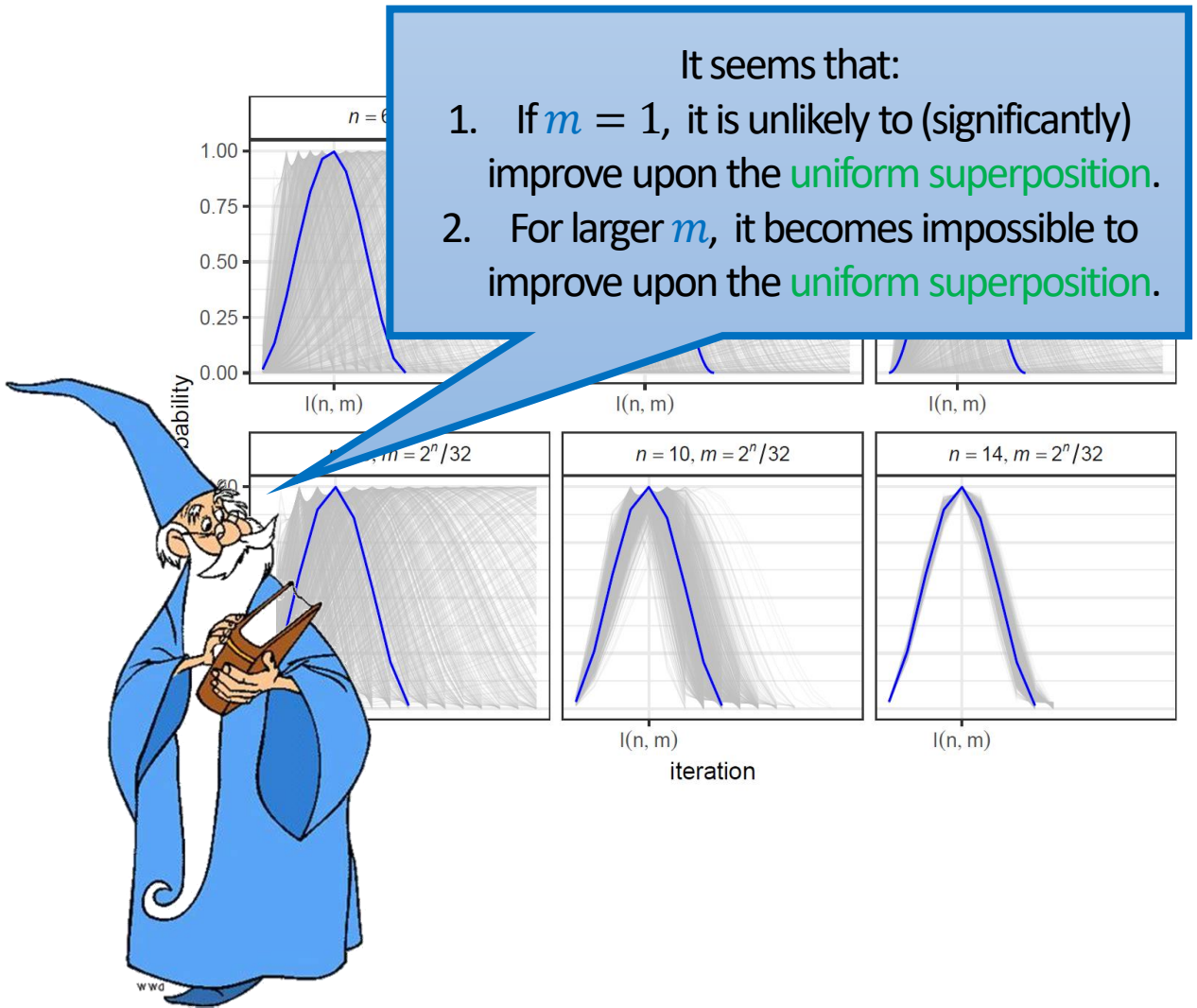
Session:
Quantum Computing &
Optimization III

Amplitude amplification: Experiment 1



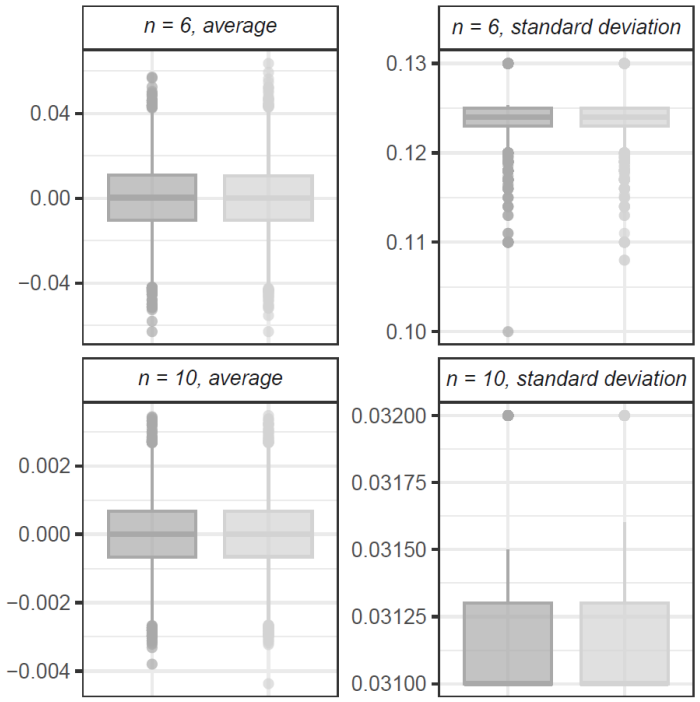
- In a first experiment, we evaluate 1000 random superpositions for $n \in \{6,10,14\}$ and verify how many of them require less iterations than a uniform superposition to measure one of the $m \in \{1, 2^n/32\}$ valid solutions.
- The results of experiment 1 are presented in the figure on the left (the blue line represents the performance of Grover's algorithm). From this figure, we can conclude:
  - If $m = 1$, roughly 30% of the superpositions require less iterations than a uniform superposition.
  - If the proportion of valid solutions increases, it becomes more difficult to find superpositions that outperform the uniform superposition.
  - The downside risk is far bigger than the upside potential (i.e., the potential increase in number of iterations is far bigger than the potential decrease).

Amplitude amplification: Experiment 1



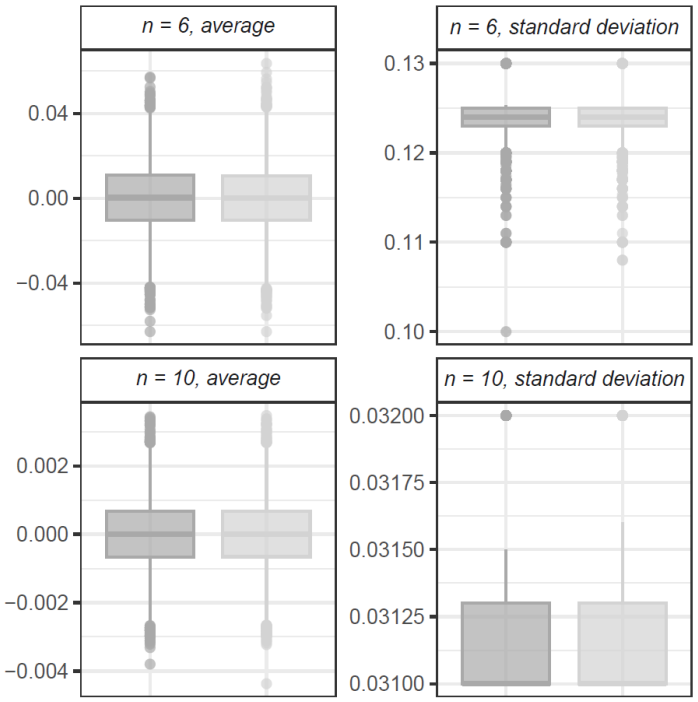
- In a first experiment, we evaluate 1000 random superpositions for $n \in \{6,10,14\}$ and verify how many of them require less iterations than a uniform superposition to measure one of the $m \in \{1, 2^n/32\}$ valid solutions.
- The results of experiment 1 are presented in the figure on the left (the blue line represents the performance of Grover's algorithm). From this figure, we can conclude:
  - If $m = 1$, roughly 30% of the superpositions require less iterations than a uniform superposition.
  - If the proportion of valid solutions increases, it becomes more difficult to find superpositions that outperform the uniform superposition.
  - The downside risk is far bigger than the upside potential (i.e., the potential increase in number of iterations is far bigger than the potential decrease).

Amplitude amplification: Experiment 2



- In a second experiment, for $n \in \{6,10\}$, we compare 1000 random superpositions and 1000 superpositions that improve upon the uniform superposition. We compare:
  - The average probability amplitude (expected to be zero in case of random superpositions).
  - The standard deviation of the probability amplitudes.
- The results of the experiment are presented on the figure on the left (random superpositions are indicated in dark grey and superpositions that improve are indicated in light grey).

Amplitude amplification: Experiment 2



$\{6,10\}$, we
and 1000
ne uniform

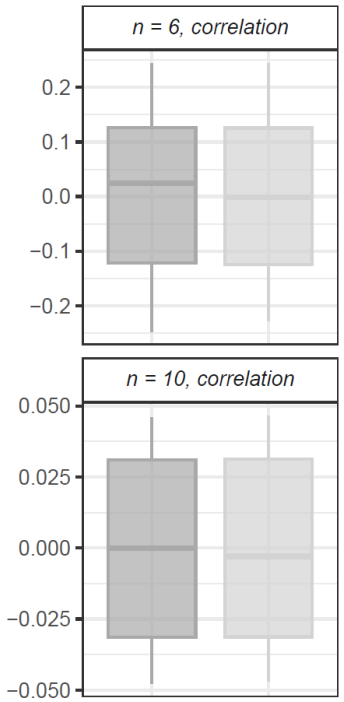
(expected
positions).

- The standard deviat… the probability amplitudes.

It seems there is no statistical difference between random superpositions and superpositions that improve the uniform superposition ➔ it may be very difficult to find these "good" superpositions!

- The results of the experiment are presented on the figure on the left (random superpositions indicated in dark grey and superposition… improve are indicated in light grey).

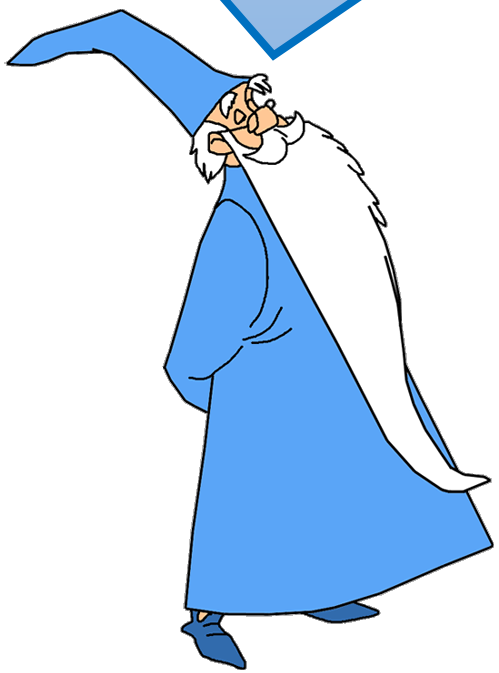
Amplitude amplification: Experiment 3

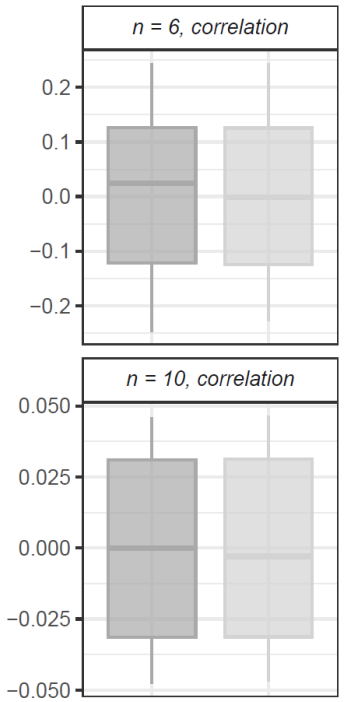


- In a third experiment, for $n \in \{6,10\}$, we compare 1000 random superpositions and 1000 superpositions that improve upon the uniform superposition.
- We compare the correlation with the optimal superposition (i.e., the superposition where the probability amplitude of qubit i is 1 if decision variable i is 1 and 0 otherwise).
- For random superpositions, we expect this correlation to be 0. For superpositions that improve upon the uniform superposition, on the other hand, we might expect that there is a positive correlation.
- The results of the experiment are presented on the figure on the left (random superpositions are indicated in dark grey and superpositions that improve are indicated in light grey).

Amplitude amplification: Experiment 3

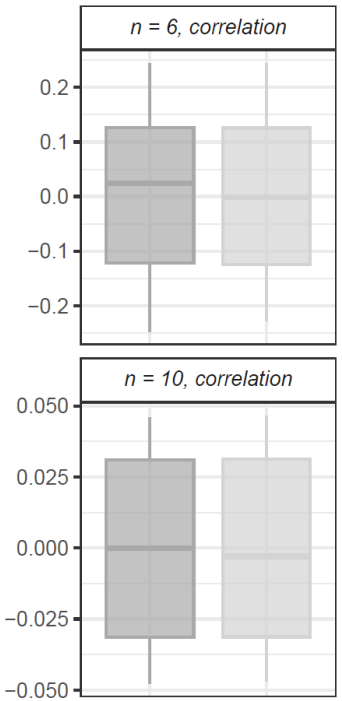
It seems there is no correlation between random superpositions and the optimal superposition (as expected). However, there also is no correlation between superpositions that improve the uniform superposition and the optimal superposition.





- In a third experiment, for $n \in \{6,10\}$, we compare 1000 random superpositions and 1000 superpositions that improve upon the uniform superposition.
- We compare the correlation with the optimal superposition (i.e., the superposition where the probability amplitude of qubit i is 1 if decision variable i is 1 and 0 otherwise).
- For random superpositions, we expect this correlation to be 0. For superpositions that improve upon the uniform superposition, on the other hand, we might expect that there is a positive correlation.
- The results of the experiment are presented on the figure on the left (random superpositions are indicated in dark grey and superpositions that improve are indicated in light grey).

Amplitude amplification: Experiment 3



Last but not least, note that for unstructured search (i.e., Grover's problem) it is not possible to outperform the uniform superposition (i.e., amplitude amplification cannot be used to solve this problem more efficiently).

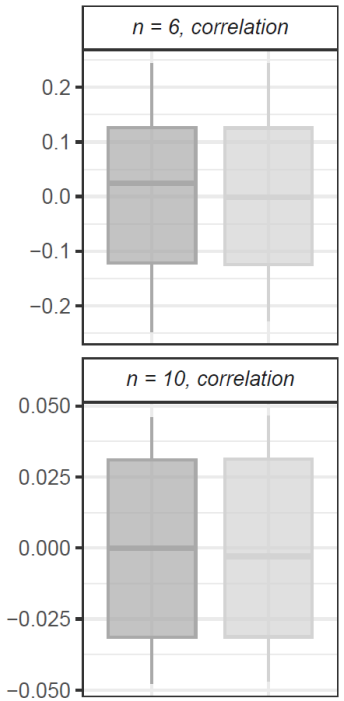
…ve compare
…nd 1000
…he uniform

…he optimal
… where the
probability amplitude of … i is 1 if decision variable i is 1 and 0 otherwise).

- For random superpositions, we … this correlation to be 0. For superpositions … upon the uniform superposition, on the o…, we might expect that there is a positive co…
- The results of the experiment are pres… figure on the left (random supe… indicated in dark grey and supe… improve are indicated in light grey).

Amplitude amplification: Experiment 3

Interesting! This implies that amplitude amplification may not be able to outperform a simple procedure such as **GUM** (that relies on Grover's algorithm)!



Last but not least, note that for unstructured search (i.e., Grover's problem) it is not possible to outperform the uniform superposition (i.e., amplitude amplification cannot be used to solve this problem more efficiently).

…ve compare
…nd 1000
…e uniform

…e optimal
… where the
probability amplitude of … i is 1 if decision variable i is 1 and 0 otherwise).

- For random superpositions, we …his correlation to be 0. For superpositions … upon the uniform superposition, on the o…, we might expect that there is a positive co…
- The results of the experiment are pres… figure on the left (random supe… indicated in dark grey and supe… improve are indicated in light grey).