Quantum Computation

(Lecture QC-2: Quantum Algorithms A)

Luís Soares Barbosa

Quantum Computing

Universidade do Minho, 2019

A quantum machine

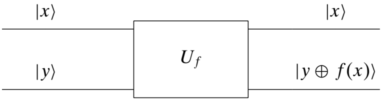
Structure of a quantum algorithm

1. State preparation (fix initial setting): typically the qubits in the initial classical state are put into a superposition of many states;
2. Transform, through unitary operators applied to the superposed state;
3. Measure, i.e. projection onto a basis vector associated with a measurement tool.

My first quantum program

Is $f : \mathbf{2} \longrightarrow \mathbf{2}$ constant, with a unique evaluation?

Oracle



where $\oplus$ stands for exclusive disjunction.

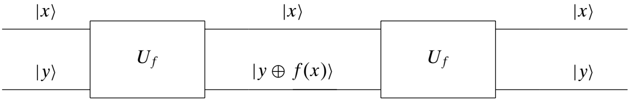
- The oracle takes input $|x, y\rangle$ to $|x, y \oplus f(x)\rangle$
- for $y = 0$ the output is $|x, f(x)\rangle$

My first quantum program

Is $f : \mathbf{2} \longrightarrow \mathbf{2}$ constant, with a unique evaluation?

Oracle

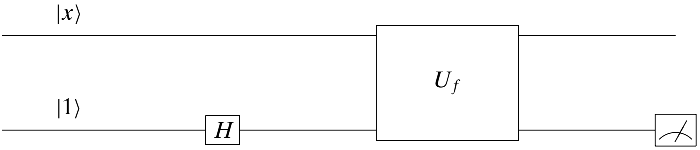
- The oracle is a unitary, i.e. reversible gate



$$|x, (y \oplus f(x)) \oplus f(x)\rangle \;=\; |x, y \oplus (f(x) \oplus f(x))\rangle \;=\; |x, y \oplus 0\rangle \;=\; |x, y\rangle$$

My first quantum program

Idea: Avoid double evaluation by superposition

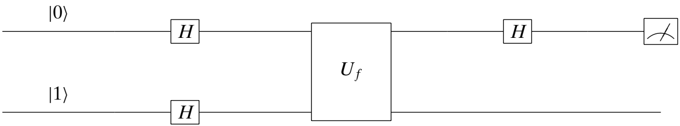


The circuit computes:

$$
\begin{aligned}
\text{output} &= |x\rangle \frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}} \\
&= \begin{cases} |x\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} & \Leftarrow f(x) = 0 \\ |x\rangle \frac{|1\rangle - |2\rangle}{\sqrt{2}} & \Leftarrow f(x) = 1 \end{cases} \\
&= (-1)^{f(x)} |x\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}
\end{aligned}
$$

My first quantum program

Idea: Avoid double evaluation by superposition



$$(H \otimes I)\; U_f\; (H \otimes H)(|01\rangle)$$

Input in superposition

$$|\sigma_1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{|00\rangle - |01\rangle + |10\rangle - |11\rangle}{2}$$

My first quantum program

$$
\begin{aligned}
|\sigma_2\rangle &= \left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \\
&= \begin{cases} (\pm 1)\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) & \Leftarrow f \text{ constant} \\ (\pm 1)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) & \Leftarrow f \text{ not constant} \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
|\sigma_3\rangle &= H|\sigma_2\rangle \\
&= \begin{cases} (\pm 1)\,|0\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) & \Leftarrow f \text{ constant} \\ (\pm 1)\,|1\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) & \Leftarrow f \text{ not constant} \end{cases}
\end{aligned}
$$

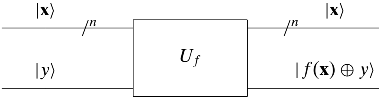
To answer the original problem is now enough to measure the first qubit: if it is in state $|0\rangle$, then f is constant.

The Deutsch-Jozsa Algorithm

Generalizing Deutsch's algorithm to functions whose domain is an initial segment n of $\mathbb{N}$, encoded into a binary string (i.e. the set of natural numbers from 0 to $\mathbf{2}^n - 1$.

Assuming $f : \mathbf{2}^n \longrightarrow \mathbf{2}$ is either balanced or constant, determine which is the case with a unique evaluation

Oracle



Using $H^{\otimes n}$ to put n qubits superposed

Computing $H^{\otimes n}$

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} (-1)^{0 \wedge 0} & (-1)^{0 \wedge 1} \\ (-1)^{1 \wedge 0} & (-1)^{1 \wedge 1} \end{bmatrix}$$

$$H^{\otimes 2} = \frac{1}{\sqrt{2}} \begin{bmatrix} (-1)^{0 \wedge 0} & (-1)^{0 \wedge 1} \\ (-1)^{1 \wedge 0} & (-1)^{1 \wedge 1} \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} (-1)^{0 \wedge 0} & (-1)^{0 \wedge 1} \\ (-1)^{1 \wedge 0} & (-1)^{1 \wedge 1} \end{bmatrix}$$

Using $H^{\otimes n}$ to put n qubits superposed

Computing $H^{\otimes n}$

$$H^{\otimes 2} = \frac{1}{\sqrt{2}}\begin{bmatrix}(-1)^{0\wedge 0} & (-1)^{0\wedge 1}\\ (-1)^{1\wedge 0} & (-1)^{1\wedge 1}\end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix}(-1)^{0\wedge 0} & (-1)^{0\wedge 1}\\ (-1)^{1\wedge 0} & (-1)^{1\wedge 1}\end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix}(-1)^{\langle 00,00\rangle} & (-1)^{\langle 00,01\rangle} & (-1)^{\langle 01,00\rangle} & (-1)^{\langle 01,01\rangle}\\ (-1)^{\langle 00,10\rangle} & (-1)^{\langle 00,11\rangle} & (-1)^{\langle 01,10\rangle} & (-1)^{\langle 01,11\rangle}\\ (-1)^{\langle 10,00\rangle} & (-1)^{\langle 10,01\rangle} & (-1)^{\langle 11,00\rangle} & (-1)^{\langle 11,01\rangle}\\ (-1)^{\langle 10,10\rangle} & (-1)^{\langle 10,11\rangle} & (-1)^{\langle 11,10\rangle} & (-1)^{\langle 11,11\rangle}\end{bmatrix}$$

where $\langle x, y\rangle = (x_0 \wedge y_0) \oplus (x_1 \wedge y_1) \oplus \cdots \oplus (x_n \wedge y_n)$
Note that

$$(-1)^{a\wedge b} \otimes (-1)^{a'\wedge b'} = (-1)^{a\wedge a' \oplus b\wedge b'} = (-1)^{\langle aa', bb'\rangle}$$

Using $H^{\otimes n}$ to put n qubits superposed

Computing $H^{\otimes n}$

In general, the value of $H^{\otimes n}$ at coordinates $\mathbf{i}, \mathbf{j}$ (row and column numbers as binary strings) is given by

$$H^{\otimes n}_{\mathbf{i},\mathbf{j}} = \frac{1}{\sqrt{2^n}}(-1)^{\langle \mathbf{i},\mathbf{j} \rangle}$$

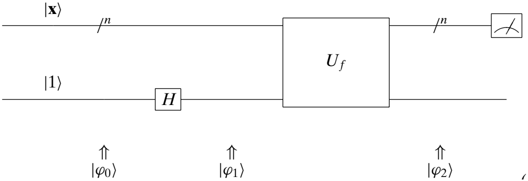
Applying $H^{\otimes n}$ to an arbitrary basic state $|\mathbf{i}\rangle$ (which is a column vector with 1 in line $\mathbf{i}$ and 0 everywhere else), extracts the $\mathbf{i}$-column of $H^{\otimes n}$:

$$H^{\otimes n}|\mathbf{i}\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x} \in \{0,1\}^n} (-1)^{\langle \mathbf{x},\mathbf{i} \rangle} |\mathbf{x}\rangle$$

e.g.

$$H^{\otimes 2}|\mathbf{0}\rangle = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \frac{1}{2}\sum_{\mathbf{x} \in \{0,1\}^n} |\mathbf{x}\rangle$$

First move: $U_f(I \otimes H)|\mathbf{x}, 1\rangle$

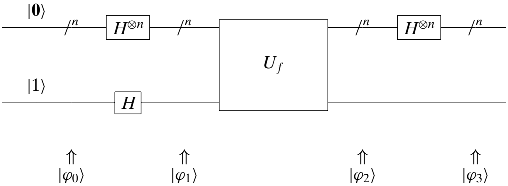


$$|\varphi_1\rangle \;=\; |\mathbf{x}\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \;=\; \frac{|\mathbf{x}, 0\rangle - |\mathbf{x}1\rangle}{\sqrt{2}}$$

$$|\varphi_2\rangle \;=\; |\mathbf{x}\rangle \frac{|f(\mathbf{x}) \oplus 0\rangle - |f(\mathbf{x}) \oplus 1\rangle}{\sqrt{2}} \;=\; (-1)^{f(\mathbf{x})}|\mathbf{x}\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Second move: $(H^{\otimes n} \otimes I)U_f(H^{\otimes n} \otimes H)|\mathbf{0}, 1\rangle$

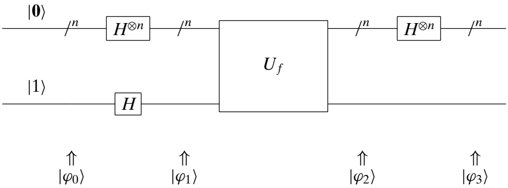
Put input $|\mathbf{x}\rangle$ into a superposition in which all 2^n possible strings have equal probability: $H^{\otimes n}|\mathbf{0}\rangle$.



$$|\varphi_1\rangle = \frac{\sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle}{\sqrt{2^n}} \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$$|\varphi_2\rangle = \frac{\sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})}|\mathbf{x}\rangle}{\sqrt{2^n}} \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Second move: $(H^{\otimes n} \otimes I)U_f(H^{\otimes n} \otimes H)|0,1\rangle$



$$
\begin{aligned}
|\varphi_3\rangle &= \frac{\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}\sum_{\mathbf{z}\in\{0,1\}^n}(-1)^{\langle \mathbf{z},\mathbf{x}\rangle}|\mathbf{z}\rangle}{\sqrt{2^n}}\ \frac{|0\rangle-|1\rangle}{\sqrt{2}} \\
&= \frac{\sum_{\mathbf{x},\mathbf{z}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}(-1)^{\langle \mathbf{z},\mathbf{x}\rangle}|\mathbf{z}\rangle}{\sqrt{2^n}}\ \frac{|0\rangle-|1\rangle}{\sqrt{2}} \\
&= \frac{\sum_{\mathbf{x},\mathbf{z}\in\{0,1\}^n}(-1)^{f(\mathbf{x})\oplus\langle \mathbf{z},\mathbf{x}\rangle}|\mathbf{z}\rangle}{\sqrt{2^n}}\ \frac{|0\rangle-|1\rangle}{\sqrt{2}}
\end{aligned}
$$

Finally: observe!

When do the top qubits of $|\varphi_3\rangle$ collapse to $|\mathbf{0}\rangle$?

Making $|\mathbf{z}\rangle = |\mathbf{0}\rangle$ (and thus $\langle \mathbf{z}, \mathbf{x}\rangle = 0$ for all $\mathbf{x}$) leads to

$$|\varphi_3\rangle \;=\; \frac{\sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})}|\mathbf{0}\rangle}{\sqrt{2^n}}\;\frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

i.e.

$\boxed{\text{the probability of collapsing to } |\mathbf{0}\rangle \text{ depends only on } f(\mathbf{x})}$

Finally: observe!

Analyse the top qubits

$\boxed{f \text{ is constant at } 1} \quad \rightsquigarrow \quad \frac{\sum_{\mathbf{x}\in\{0,1\}^n} (-1)|\mathbf{0}\rangle}{\sqrt{2^n}} = \frac{-(2^n)|\mathbf{0}\rangle}{2^n} = -|\mathbf{0}\rangle$

$\boxed{f \text{ is constant at } 0} \quad \rightsquigarrow \quad \frac{\sum_{\mathbf{x}\in\{0,1\}^n} 1|\mathbf{0}\rangle}{\sqrt{2^n}} = \frac{(2^n)|\mathbf{0}\rangle}{2^n} = |\mathbf{0}\rangle$

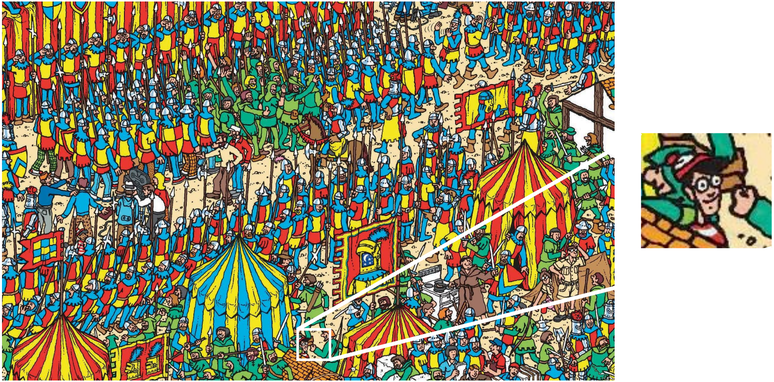
$\boxed{f \text{ is balanced}} \quad \rightsquigarrow \quad \frac{\sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})}|\mathbf{0}\rangle}{\sqrt{2^n}} = \frac{0|\mathbf{0}\rangle}{2^n} = 0|\mathbf{0}\rangle$

because half of the $\mathbf{x}$ will cancel the other half

$\boxed{\text{The top qubits collapse to } |\mathbf{0}\rangle \text{ only if } f \text{ is constant}}$

Exponential speed up: f was evaluated once rather than $2^n - 1$ times

Search problems



Search problems

A more precise formulation

Given a function $f : 2^n \longrightarrow 2$ such that there exsits a unique binary string $\mathbf{x}^*$ st

$$f(\mathbf{x}) = \begin{cases} 1 & \Leftarrow \mathbf{x} = \mathbf{x}^* \\ 0 & \Leftarrow \mathbf{x} \neq \mathbf{x}^* \end{cases}$$

determine $\mathbf{e}$.

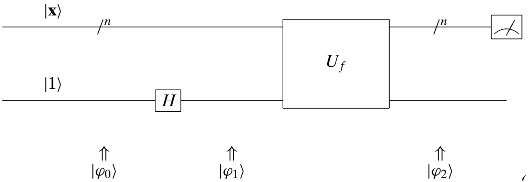
A quadratic speed up

- Worst case for a classic algorithm: 2^n evaluations of f
- Worst case for Grover's algorithm: $\sqrt{2^n}$ evaluations of f

Grover's algorithm

Oracle U_f inverts the phase at $|\mathbf{x}^*\rangle$

Recall from Deutsch-Josza:

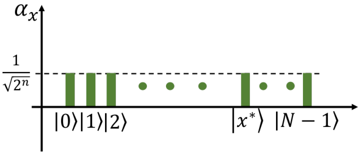


$$|\varphi_2\rangle = (-1)^{f(\mathbf{x})}|\mathbf{x}\rangle\frac{|0\rangle - |1\rangle}{\sqrt{2}} = \begin{cases} -|\mathbf{x}\rangle\frac{|0\rangle - |1\rangle}{\sqrt{2}} & \Leftarrow \mathbf{x} = \mathbf{x}^* \\ +|\mathbf{x}\rangle\frac{|0\rangle - |1\rangle}{\sqrt{2}} & \Leftarrow \mathbf{x} \neq \mathbf{x}^* \end{cases}$$

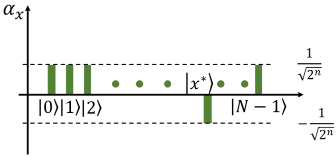
Grover's algorithm

Oracle U_f inverts the phase at $|\mathbf{x}^*\rangle$

Thus, providing as input a balanced superposition of all possible states, via $H^{\otimes n}|\mathbf{0}\rangle$, the oracle is able to detect the solution and shift its phase:



$\rightsquigarrow$

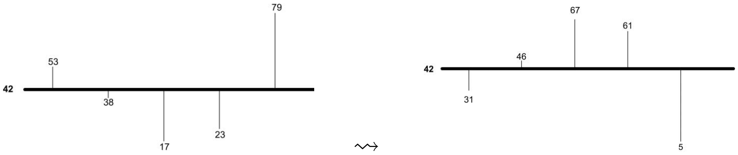


However, the probability of collapsing to $|\mathbf{x}^*\rangle$ is equal to the one of collapsing to any other basic state becase

$$|-\frac{1}{\sqrt{2^n}}|^2 = |-\frac{1}{\sqrt{2^n}}|^2$$

Boosting the phase separation

The trick: Inversion around the mean



$$e' = \text{mean} + (\text{mean} - e) \Leftrightarrow e' = -e + 2\text{mean}$$

Computing the mean (example)

$$\begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix} \begin{bmatrix} 53 \\ 38 \\ 17 \\ 23 \\ 79 \end{bmatrix} = \begin{bmatrix} 42 \\ 42 \\ 42 \\ 42 \\ 42 \end{bmatrix}$$

Boosting the phase separation

The trick: Inversion around the mean

For A the *grid matrix*,

$$V' = -V + 2AV = (-I + 2A)V$$

multiplying any state by $(-I + 2A)$ inverts amplitudes around the mean.

Healthiness test

Operator $(-I + 2A)$ is unitary, because

- $(-I + 2A)^\dagger = (-I + 2A)$
- $(-I + 2A)(-I + 2A) = I - 2A - 2A + 4A^2 = I - 4A + 4A = I$

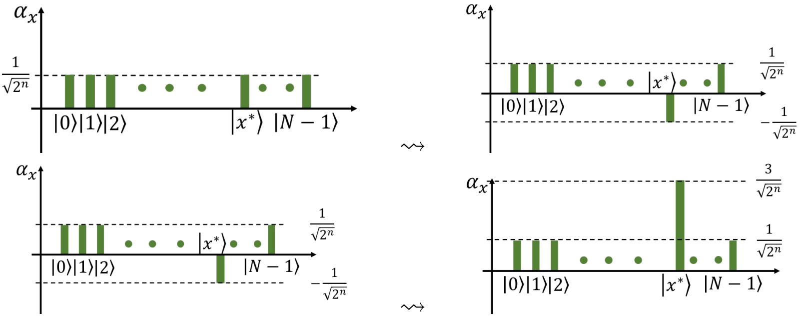
Combining effects over time to amplify the right phase

Example

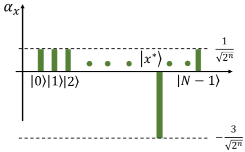
- Start with $[10, 10, 10, 10, 10]^T$
- Invert the fourth entry: $[10, 10, 10, -10, 10]^T$
- Invert around mean (6): $[2, 2, 2, 22, 2]^T$
  Note $22 - 2 = 20$
- Invert the fourth entry again: $[2, 2, 2, -22, 2]^T$
- Invert around mean (-2.8): $[-7.6, -7.6, -7.6, 16.4, -7.6]^T$
  Note $16.4 + 7.6 = 24$.
- ...

The right phase is amplified in successive iterations

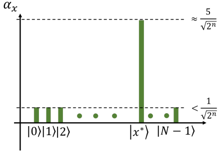
Combining effects to amplify the right phase

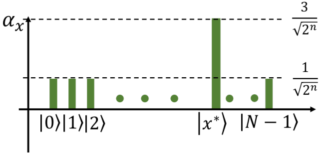


Combining effects to amplify the right phase

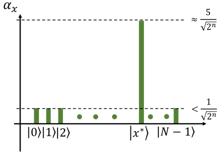


$\rightsquigarrow$

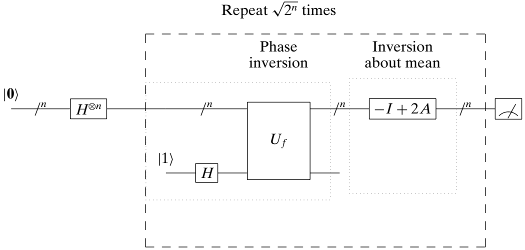




$\rightsquigarrow$



Grover's algorithm



Questions

- Why $\sqrt{2^n}$ iterations?
- How to implement the oracle?
- Generalizations?
  e.g. multiple search requires $\sqrt{\frac{2^n}{t}}$ iterations for t the multiplicity

Grover's algorithm is everywhere

SAT (= Boolean satisfiability) problems

Determining values for Boolean variables so that a given Boolean expression evaluates to true

- NP-complete
- Many problems, like scheduling, can be converted into a SAT
- Can be seen as a search problem whose goal is to find a precise combination of Boolean values that yields true

Grover's algorithm is everywhere

Mini project

Implement Grover's Algorithm in Qiskit to find a satisfying assignment containing one true literal per clause.

- INPUT: SAT formula in conjunctive normal form, i.e. a conjunction of disjunctive clauses $\bigvee_{k=1..m} \phi_k$ over n Boolean variables with 3 literals per clause.
- OUTPUT: Is there an assignment to the n Boolean variables such that every clause has exactly one true literal?

Second thoughts

Creating a uniform superposition of all basis states does not allow to satisfactorily solve NP-complete problems

Let U_f encode a SAT formula on n Boolean variables:

$$U_f(|\mathbf{i}\rangle \otimes |0\rangle) = |\mathbf{i}\rangle \otimes |f(\mathbf{i})\rangle$$

Applying U_f to a superposition obtained via $H^{\otimes n}|\mathbf{0}\rangle$, which evaluates the truth assignment of all possible binary strings, will return a binary string that satisfies the formula iff the last qubit has value 1 after the measurement, and this happens with a probability that depends on the number of binary assignments that satisfy the formula (e.g. $\frac{\tau}{2^n}$, for τ such assignments).

Second thoughts

Although, in general, solving NP-hard problems in polynomial time with quantum computers is probably not possible (cf $P = NP$?), there is a recipe to produce faster equivalent quantum algorithms:

- Create a uniform superposition of basis states
- Make the basis states interact with each other so that the modulus of the coefficients for some (desirable) basis states increase, which implies that the other coefficients decrease.
- How to do it ... depends on the problem