Constant factor improvement of the Grover's algorithm

Aleksandrs Rivošs
Nikolajs Nahimovs

Faculty of Computing
University of Latvia

Riga 2010

Introduction

- The Grover's algorithm is a quantum search algorithm solving the unstructured search problem in about $O(\sqrt{N})$ queries.

- The algorithm is known to be optimal. For any number of queries up to $(\pi/4)\sqrt{N}$ it ensures a maximal possible probability of finding the desired element.

Introduction

- However, it is still possible to reduce the average number of steps required to find the desired element by ending the computation earlier and repeating the algorithm

- This fact is mentioned by Christof Zalka as a short remark on analysis of the Grover’s algorithm

- We give a detailed description of this simple fact

Unstructured search

- We have a function given as a black-box:

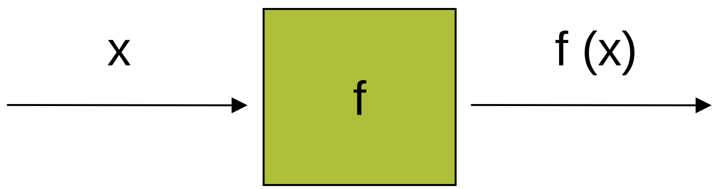
$$f : \{0,1\}^n \rightarrow \{0,1\}$$

The unstructured search problem is to find $x \in \{0,1\}^n$ such that $f(x) = 1$, or to conclude that no such x exists.

Query model : classical case

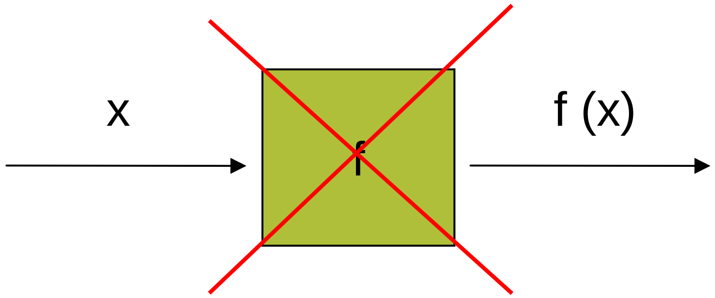
- In classical case we do not have any limitation on the behavior of the function.

- The function takes an input and returns corresponding output

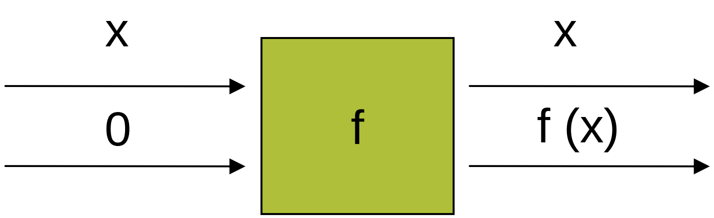


Query model : quantum case

- In quantum case the function must be reversible.
  Thus we can not use classical approach.

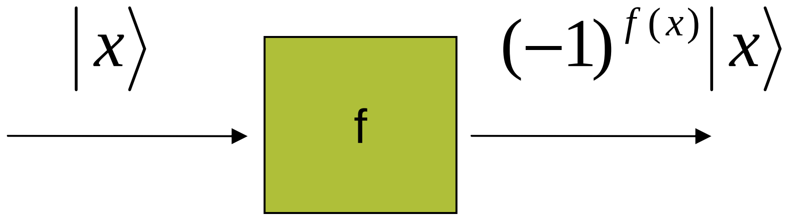


- To overcome the limitation we add an auxiliary input

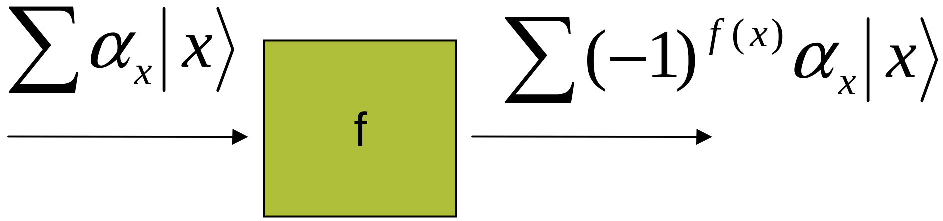


Query model : quantum case

- We can also make queries like



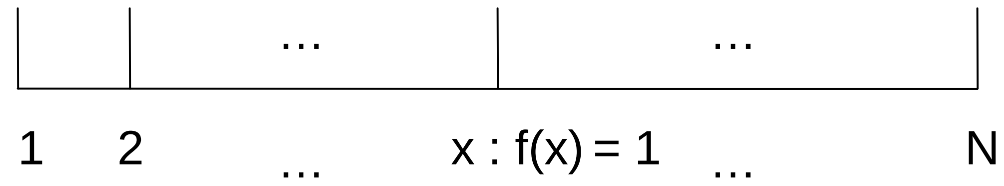
or even query a superposition all possible inputs

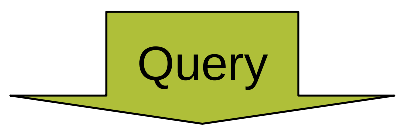


Query model : quantum case

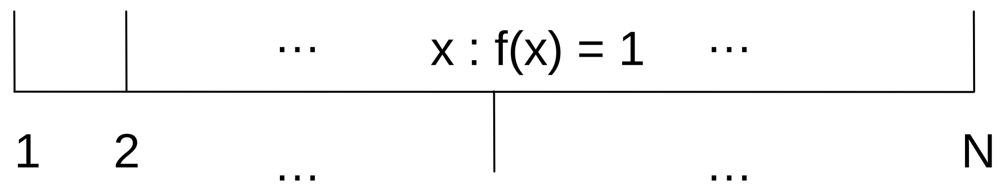
- There is a simple way to visualize a quantum query

State before the query





State after the query



Unstructured search

- We have a function given as a black-box:

$$f : \{0,1\}^n \to \{0,1\}$$

The unstructured search problem is to find $x \in \{0,1\}^n$ such that $f(x) = 1$, or to conclude that no such x exists.

- How many times do we need to calculate the function to solve the problem?

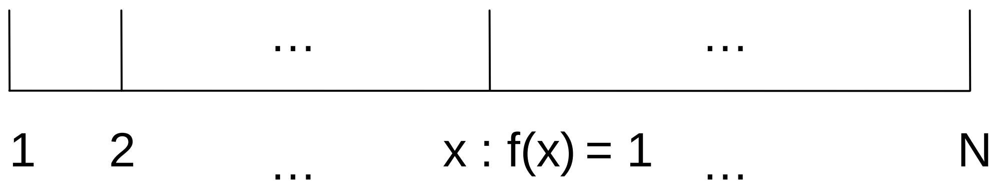
Unstructured search

- Any deterministic algorithm needs $N = 2^n$ queries to the black-box in a worst case.
- Probabilistically we also need $\Omega(N)$ queries to solve the problem.
- Grover's quantum search algorithm can solve the problem making $O(\sqrt{N})$ queries to the black-box

Grover's algorithm

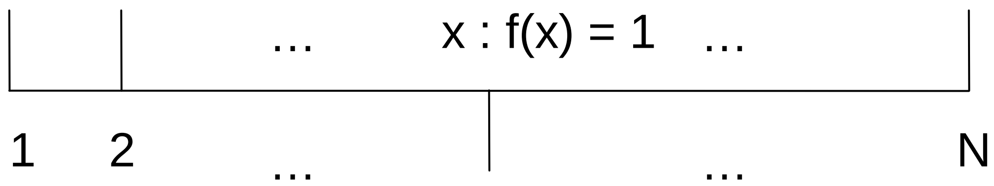
- Start at uniform superposition of all $x \in \{0,1\}^n$

- Repeat $O(\sqrt{N})$ times
  - Perform query
  - Apply the inversion about average



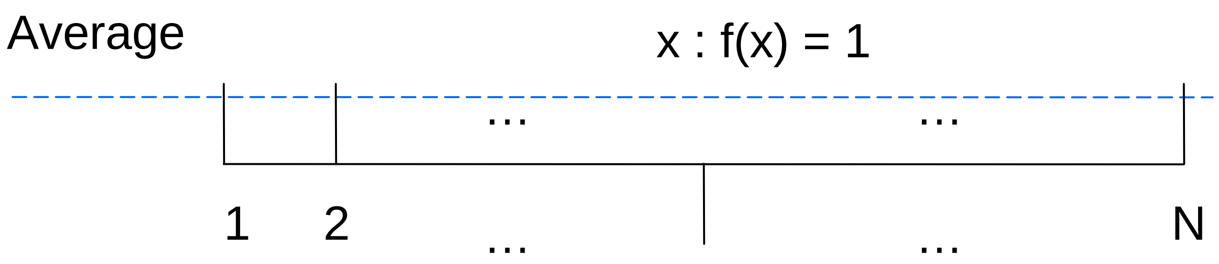
Grover's algorithm

- Start at uniform superposition of all $x \in \{0,1\}^n$
- Repeat $O(\sqrt{N})$ times
  - Perform query
  - Apply the inversion about average



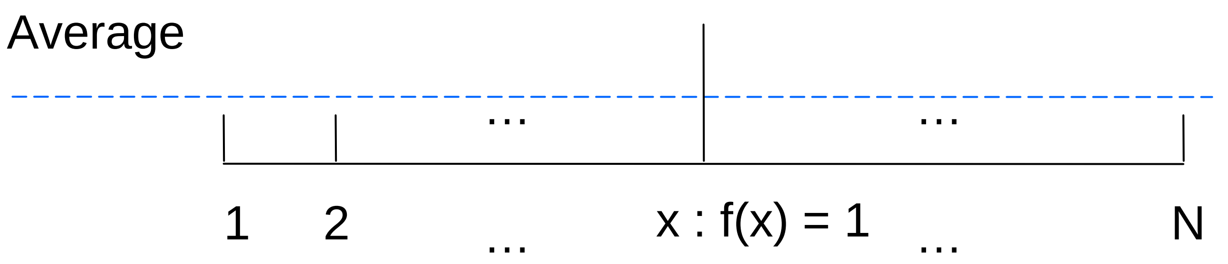
Grover's algorithm

- Start at uniform superposition of all $x \in \{0,1\}^n$
- Repeat $O(\sqrt{N})$ times
  - Perform query
  - Apply the inversion about average



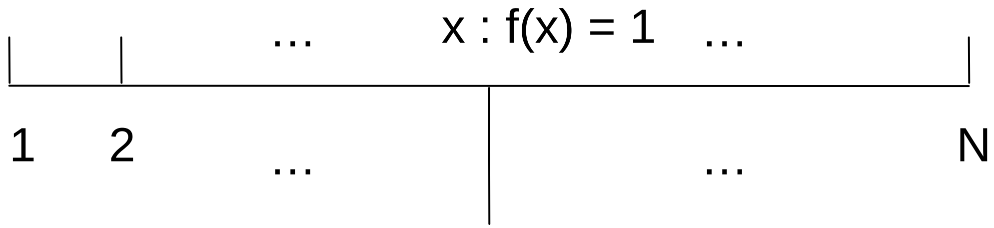
Grover's algorithm

- Start at uniform superposition of all $x \in \{0,1\}^n$
- Repeat $O(\sqrt{N})$ times
  - Perform query
  - Apply the inversion about average



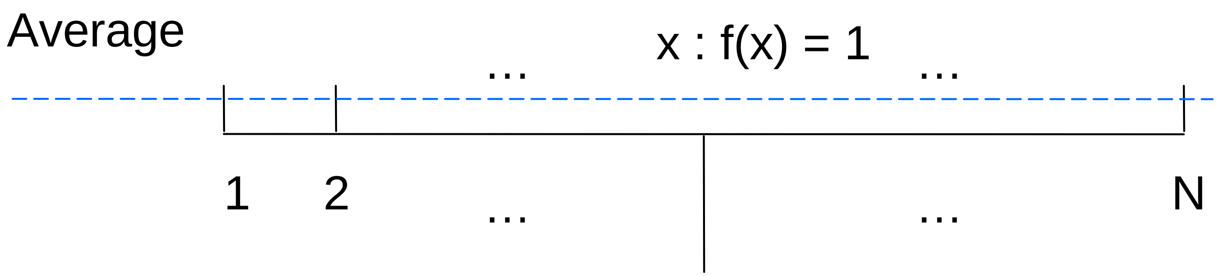
Grover’s algorithm

- Start at uniform superposition of all $x \in \{0,1\}^n$
- Repeat $O(\sqrt{N})$ times
  - Perform query
  - Apply the inversion about average



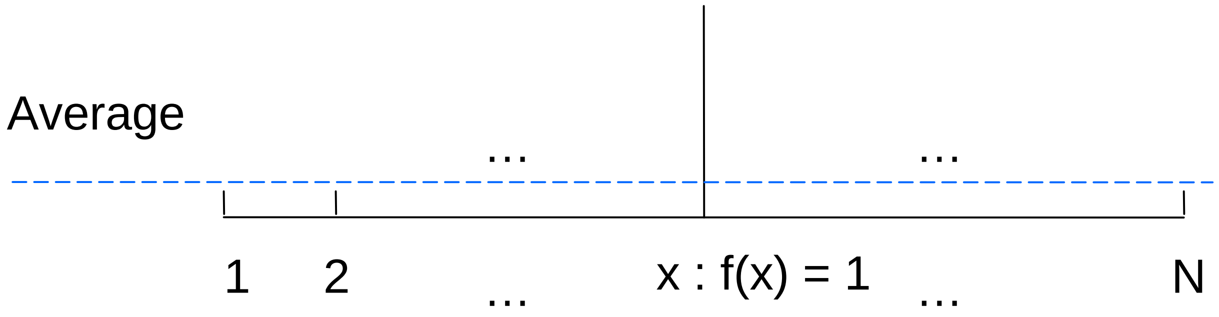
Grover's algorithm

- Start at uniform superposition of all $x \in \{0,1\}^n$
- Repeat $O(\sqrt{N})$ times
  - Perform query
  - Apply the inversion about average



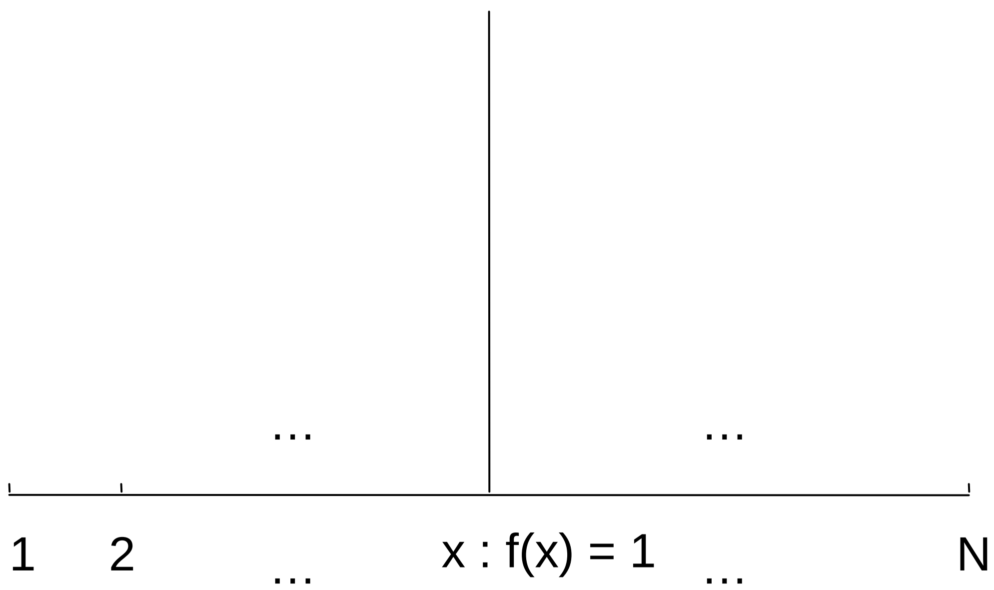
Grover's algorithm

- Start at uniform superposition of all $x \in \{0,1\}^n$
- Repeat $O(\sqrt{N})$ times
  - Perform query
  - Apply the inversion about average



Grover's algorithm

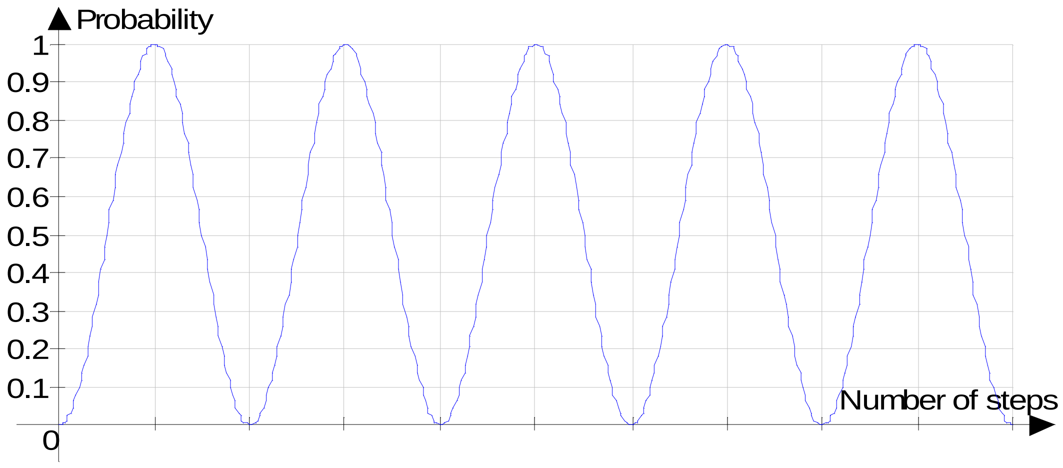
- By repeating algorithm steps $O(\sqrt{N})$ times probability to measure x with f(x) = 1 will become close to 1.



Grover's algorithm

- Grover's quantum search algorithm can solve the problem making $O(\sqrt{N})$ queries to the black-box

- The probability of finding a solution after k steps is $\sin^2 (2k / \sqrt{N})$

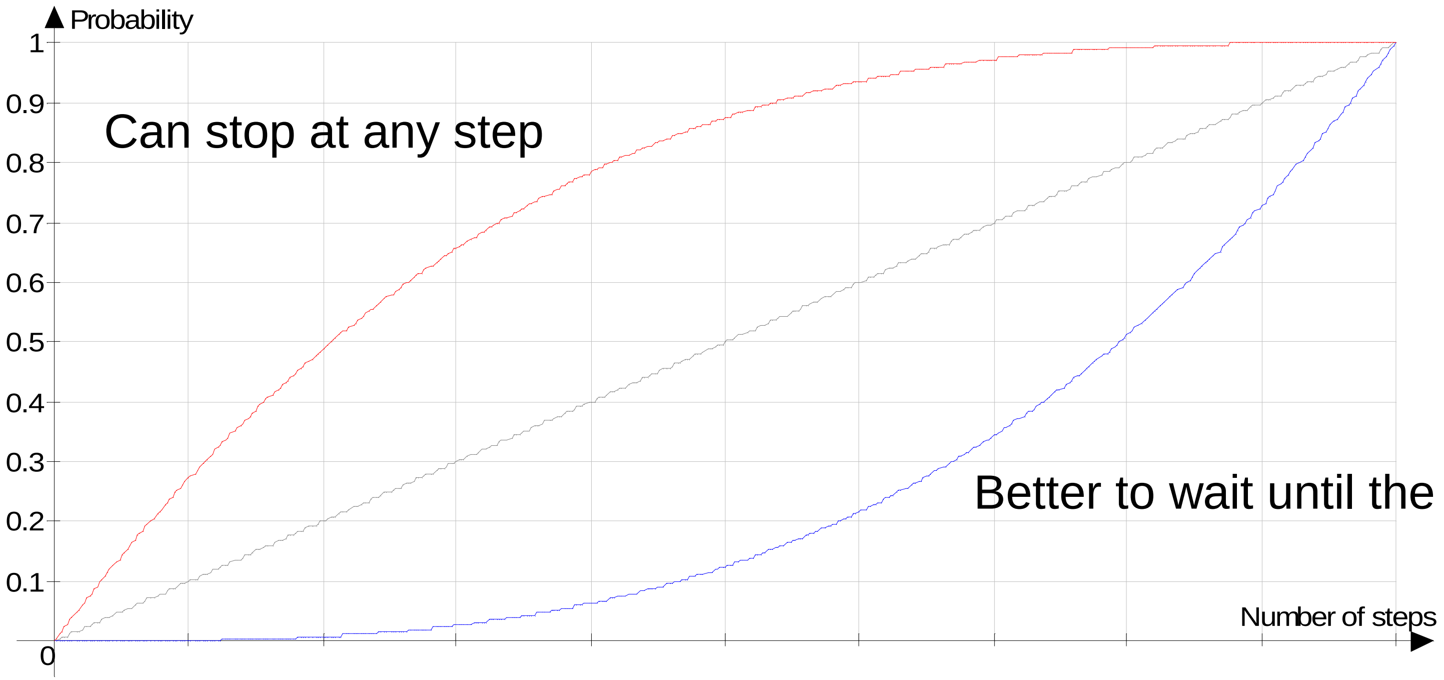


Probabilistic algorithms

- We have a probabilistic algorithm, which finds a solution with probability p.
- How many times we need to run the algorithm to find a solution with probability 1 ?
- On the average we should run the algorithm 1/p times.

Probabilistic algorithms

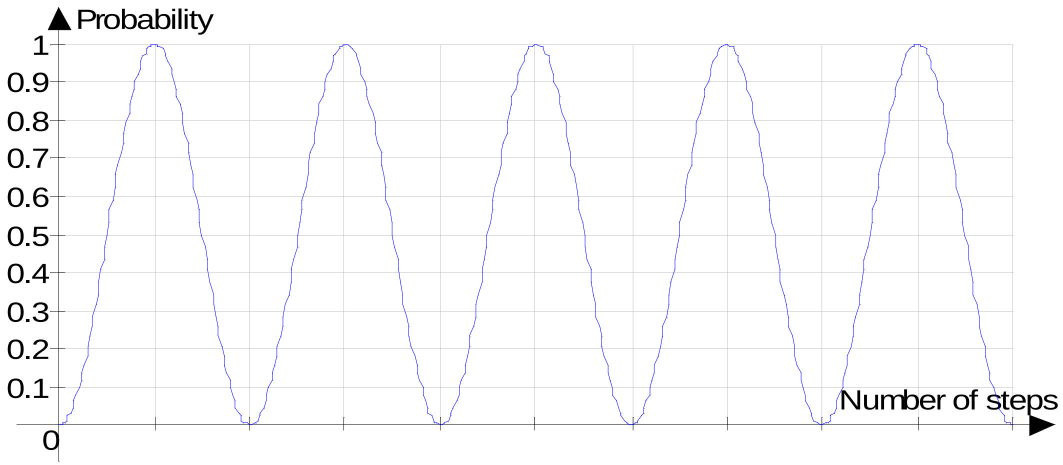
- If after k steps the probability of finding a solution is p(k), the average running time of the algorithm is k / p(k).



Grover's algorithm

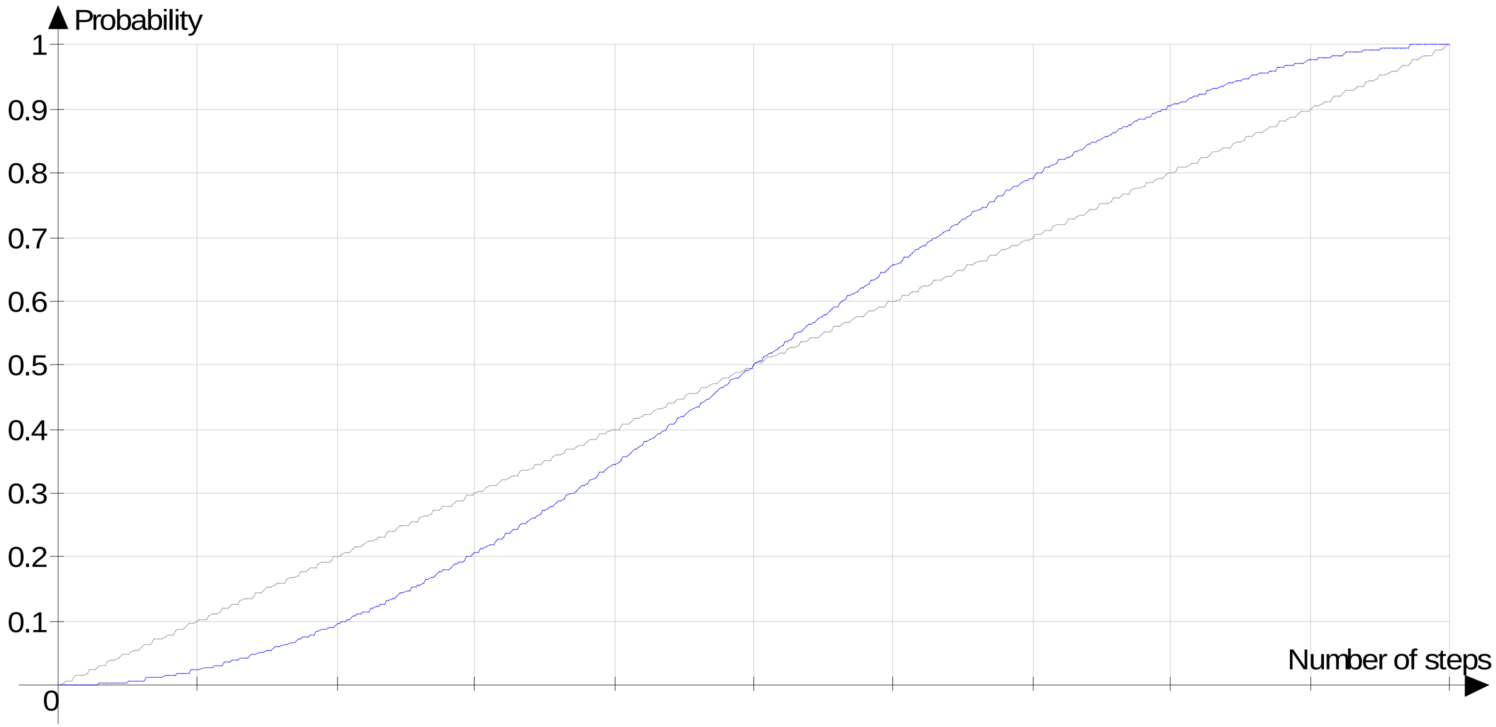
- Grover's quantum search algorithm can solve the problem making $O(\sqrt{N})$ queries to the black-box

- The probability of finding a solution after k steps is $\sin^2 (2k / \sqrt{N})$



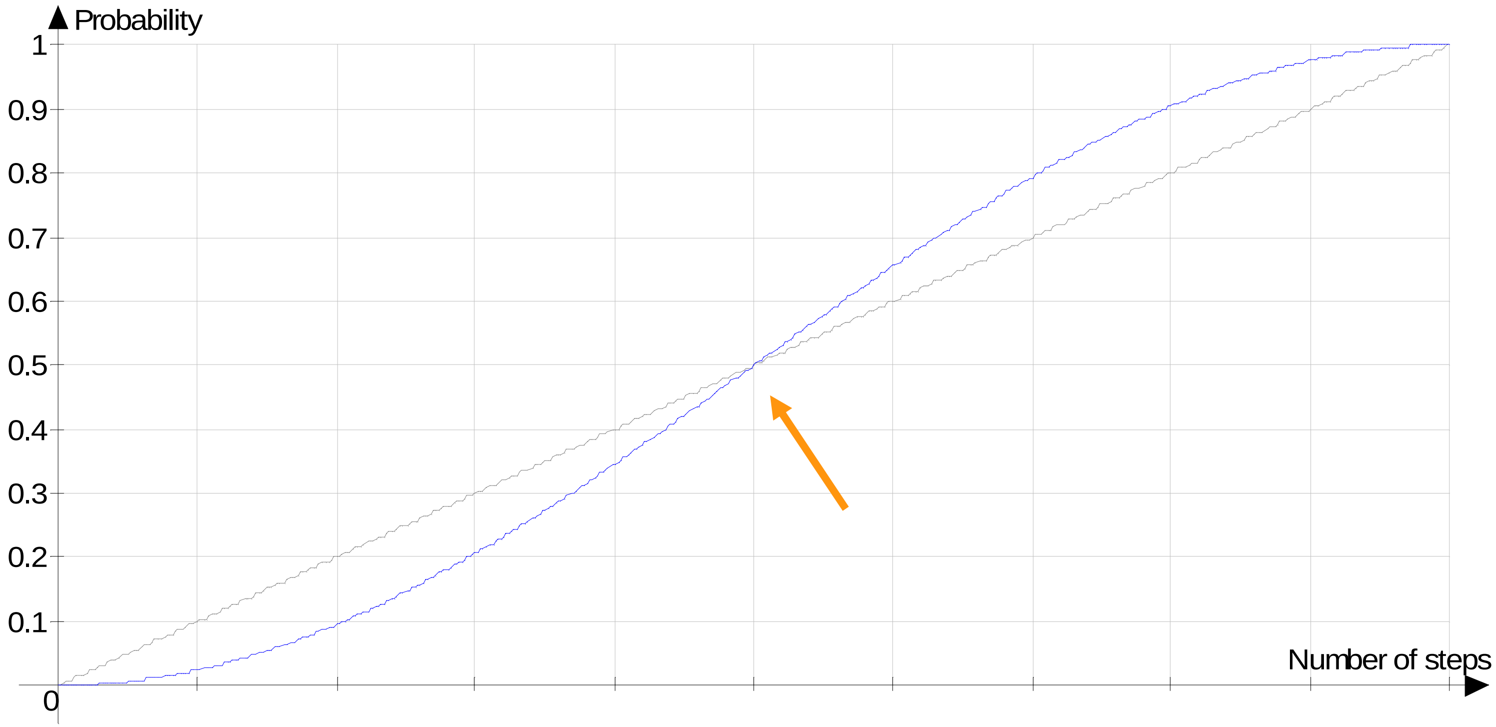
Grover's algorithm

- Let M be a number of steps of the Grover's algorithm.



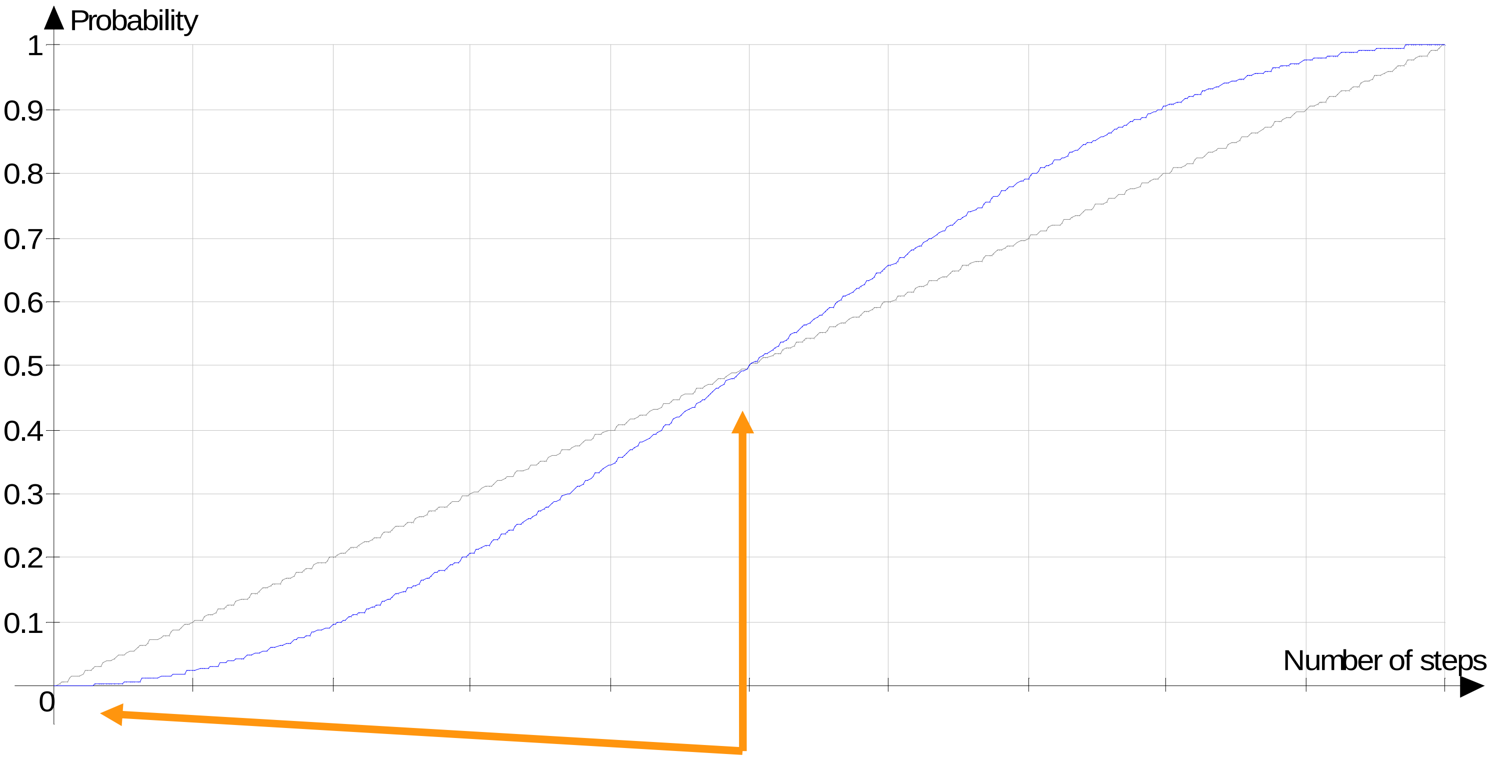
Grover's algorithm

- If p(k) = k/M, the average running time is M.



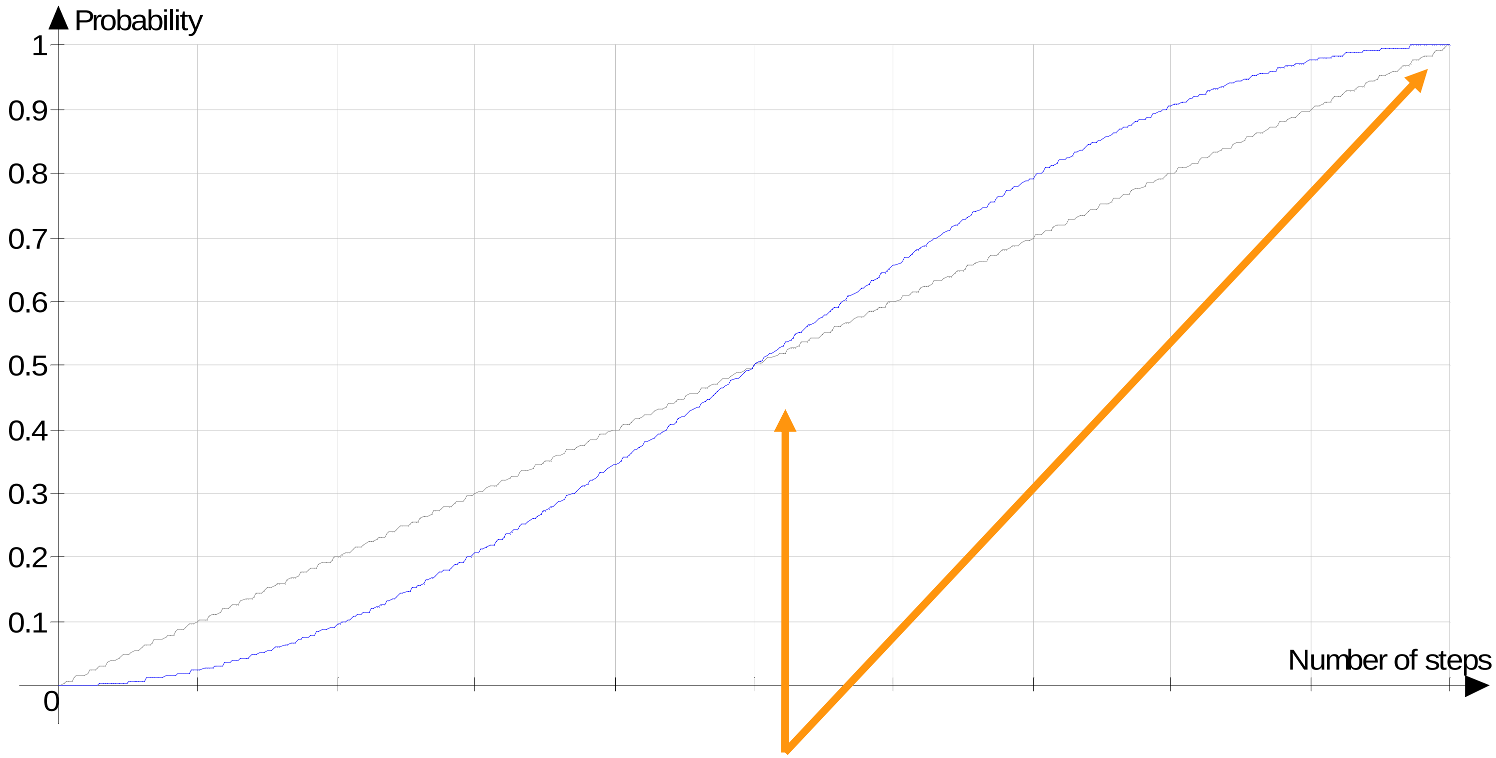
Grover's algorithm

- If p(k) < k/M, the average running time > M.



Grover's algorithm

- If p(k) > k/M, the average running time < M.



Grover's algorithm

- The optimal moment to end the computation is the minimum of the $k/p(k) = k / \sin^2 (\pi k / 2M)$ function.
- Calculation gives $k \approx 0.74202$ and the average running time $k/p(k) \approx 0.87857$.
- That is the average number of steps can be reduced by approximately 12.14%.

Conclusions

- The average number of Grover's algorithm steps can be reduced by approximately 12.14%.

- The same argument can be applied to a wide range of other quantum query algorithms, such as amplitude amplification, some variants of quantum walks and NAND formula evaluation, etc.

Thank you !