Finding many collisions via quantum walks

Application to lattice sieving

Xavier Bonnetain André Chailloux André Schrottenloher
Yixin Shen

April 24, 2023

Collision-finding

Classical collision algorithms

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

A first algorithm

- Create a sorted list $(x_i, f(x_i))$ of size $2^{n/2}$
- Look for collisions
- Also works for collisions between L_1 and L_2

Classical collision algorithms

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

A first algorithm

- Create a sorted list $(x_i, f(x_i))$ of size $2^{n/2}$
- Look for collisions
- Also works for collisions between L_1 and L_2

Improved algorithms

- Memoryless
- Parallelizable
- Same query complexity

Other cases

Finding 2^t collisions instead of 1

List of size $2^{t/2+n/2}$ contains 2^t collisions on average

Other cases

Finding 2^t collisions instead of 1

List of size $2^{t/2+n/2}$ contains 2^t collisions on average

$f : \{0,1\}^n \to \{0,1\}^m$, $n < m \leq 2n$

List of size $2^{t/2+m/2}$

Other cases

Finding 2^t collisions instead of 1

List of size $2^{t/2+n/2}$ contains 2^t collisions on average

$f : \{0,1\}^n \to \{0,1\}^m$, $n < m \leq 2n$

List of size $2^{t/2+m/2}$

Lower bounds

Matching query lower bound in all cases $\left(\Omega\left(2^{t/2+m/2}\right)\right)$

Quantum collisions

BHT

BHT algorithm

- Take a list $L = (f(y_0), \ldots, f(y_{2^u}))$
- *Search* for an x with $f(x) = f(y_i)$ and $x \neq y_i$
- Cost 2^u memory, $2^u + \sqrt{\frac{2^n}{2^u}}$ time

BHT

BHT algorithm

- Take a list $L = (f(y_0), \dots, f(y_{2^u}))$
- *Search* for an x with $f(x) = f(y_i)$ and $x \neq y_i$
- Cost 2^u memory, $2^u + \sqrt{\frac{2^n}{2^u}}$ time

Finding 2^t collisions

- Use one larger list of size $2^{n/3+2t/3}$
- Do 2^t quantum searches $\left(\text{cost } 2^t \times \sqrt{\frac{2^n}{2^{n/3-2t/3}}}\right)$

Lower bound [LZ19]

General query lower bound $\Omega\left(2^{m/3+2t/3}\right)$

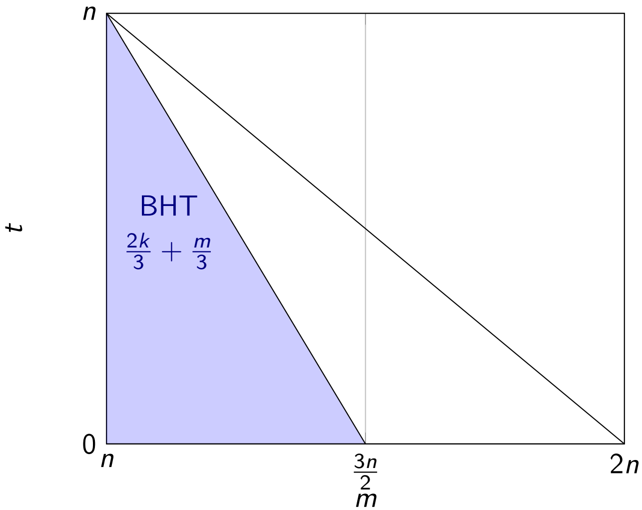
With larger m

BHT algorithm

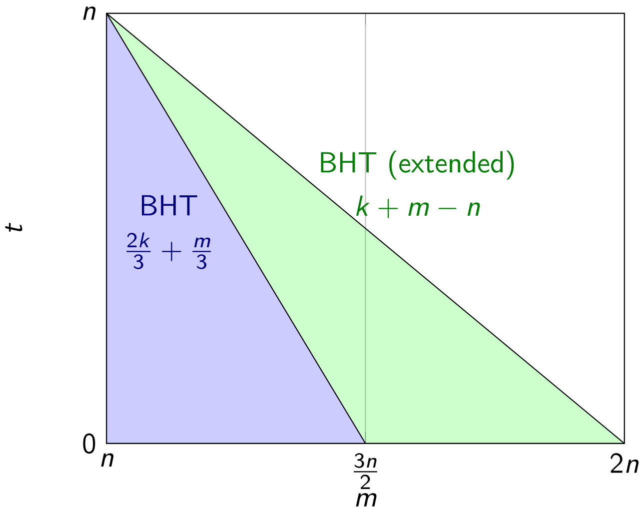
List of size $2^{m/3}$

- Only 2^{2n-m} inputs are part of a collision
- Need $m/3 \geq m - n$, otherwise the list might contain no relevant input
- $\implies m \leq 3/2n$

Summary



Summary



Quantum walks

Random walks

Idea

- Start at a random node
- Walk to adjacent nodes, stop when a good node is found

Random walks

Idea

- Start at a random node
- Walk to adjacent nodes, stop when a good node is found

Parameters

- Proportion of marked nodes p
- Number of walks steps to sample a random node $1/\varepsilon$
- Cost to construct the first random node S
- Cost to walk to an adjacent node U
- Cost to check is a node is marked T

Random walks

Idea

- Start at a random node
- Walk to adjacent nodes, stop when a good node is found

Parameters

- Proportion of marked nodes p
- Number of walks steps to sample a random node $1/\varepsilon$
- Cost to construct the first random node S
- Cost to walk to an adjacent node U
- Cost to check is a node is marked T

Total cost

$$S + \frac{1}{p}\left(\frac{1}{\varepsilon}U + T\right)$$

Example: Walk-based collision finding

Definition (Johnson graph)

- Nodes are sets of 2^r elements among 2^n
- N_1 and N_2 are adjacents is $|N_1 \cap N_2| = 2^r - 1$
- $\varepsilon = \frac{2^n}{2^r(2^n - 2^r)} \simeq 2^{-r}$

Example: Walk-based collision finding

Definition (Johnson graph)

- Nodes are sets of 2^r elements among 2^n
- N_1 and N_2 are adjacents is $|N_1 \cap N_2| = 2^r - 1$
- $\varepsilon = \frac{2^n}{2^r(2^n - 2^r)} \simeq 2^{-r}$

Collision finding with Johnson graph

- Create a random list of elements of size 2^r
- Walk 2^r times
- If the node contains a collision, stop

Example: Walk-based collision finding

Definition (Johnson graph)

- Nodes are sets of 2^r elements among 2^n
- N_1 and N_2 are adjacents is $|N_1 \cap N_2| = 2^r - 1$
- $\varepsilon = \frac{2^n}{2^r(2^n - 2^r)} \simeq 2^{-r}$

Collision finding with Johnson graph

- Create a random list of elements of size 2^r
- Walk 2^r times
- If the node contains a collision, stop

Complexity

$$2^r + \frac{1}{2^{2r-n}}(2^r \times 1 + 2^r) \simeq \max(2^r, 2^{n-r})$$

Quantum walks

Principle

Simulate a quantum search on a graph using a walk update operator

Quantum walks

Principle

Simulate a quantum search on a graph using a walk update operator

MNRS framework

- Proportion of marked nodes p
- Number of walks steps to sample a uniformly random node $1/\varepsilon$
- Cost to construct the superposition of all nodes S
- Cost to walk to an adjacent node U
- Cost to check is a node is marked T
- Total cost $S + \frac{1}{\sqrt{p}}\left(\frac{1}{\sqrt{\varepsilon}}U + T\right)$

Ambainis algorithm

Problem

$f : \{0,1\}^n \to \{0,1\}^m$, $n < m \leq 2n$, find a collision

MNRS walk in a Johnson graph

- Create a random list of elements of size 2^r
- Apply the walk operator $\sqrt{2^r}$ times
- Test if the node contains a collision

Complexity

- Setup : 2^r
- Fraction of marked nodes : 2^{2r-m}
- Assume Update and Test polynomial
- Cost $2^r + 2^{m/2-r} \times 2^{r/2} \simeq \max(2^r, 2^{m/2-r/2})$

Finding t collisions

Idea

Use more memory, amortize it to find more collisions

Finding t collisions

Idea

Use more memory, amortize it to find more collisions

Issue

At the end of Ambainis' walk, the measurement destroys the quantum state

Finding t collisions

Idea

Use more memory, amortize it to find more collisions

Issue

At the end of Ambainis' walk, the measurement destroys the quantum state

Aim

Having a procedure that allows us to extract a collision and preserve a useful quantum data structure

New algorithm

- Begin with a normal quantum walk

New algorithm

- Begin with a normal quantum walk
- Measure the number of collisions we have in the end

New algorithm

- Begin with a normal quantum walk
- Measure the number of collisions we have in the end
- Remove reversibly this number of collision from the data structure

New algorithm

- Begin with a normal quantum walk
- Measure the number of collisions we have in the end
- Remove reversibly this number of collision from the data structure
- Measure the extracted collisions

New algorithm

- Begin with a normal quantum walk
- Measure the number of collisions we have in the end
- Remove reversibly this number of collision from the data structure
- Measure the extracted collisions
- Final state is now the uniform superposition of all structures:

New algorithm

- Begin with a normal quantum walk
- Measure the number of collisions we have in the end
- Remove reversibly this number of collision from the data structure
- Measure the extracted collisions
- Final state is now the uniform superposition of all structures:
    - Without collision

New algorithm

- Begin with a normal quantum walk
- Measure the number of collisions we have in the end
- Remove reversibly this number of collision from the data structure
- Measure the extracted collisions
- Final state is now the uniform superposition of all structures:
  - Without collision
  - Without any of the extracted inputs

New algorithm

- Begin with a normal quantum walk
- Measure the number of collisions we have in the end
- Remove reversibly this number of collision from the data structure
- Measure the extracted collisions
- Final state is now the uniform superposition of all structures:
  - Without collision
  - Without any of the extracted inputs
- Do a new walk on a smaller Johnson graph:

New algorithm

- Begin with a normal quantum walk
- Measure the number of collisions we have in the end
- Remove reversibly this number of collision from the data structure
- Measure the extracted collisions
- Final state is now the uniform superposition of all structures:
  - Without collision
  - Without any of the extracted inputs
- Do a new walk on a smaller Johnson graph:
  - With smaller sets (-collisions)

New algorithm

- Begin with a normal quantum walk
- Measure the number of collisions we have in the end
- Remove reversibly this number of collision from the data structure
- Measure the extracted collisions
- Final state is now the uniform superposition of all structures:
  - Without collision
  - Without any of the extracted inputs
- Do a new walk on a smaller Johnson graph:
  - With smaller sets (-collisions)
  - In a smaller ambient set (avoid the extracted preimages)

Assumptions

Efficient history-independent operations

Assumptions

Efficient history-independent operations

- Use a data structure built upon radix trees

Assumptions

Efficient history-independent operations

- Use a data structure built upon radix trees
- Quantum memory layout: uniform superposition of all possible classical layout

Assumptions

Efficient history-independent operations

- Use a data structure built upon radix trees
- Quantum memory layout: uniform superposition of all possible classical layout

The next quantum walk needs to work

Assumptions

Efficient history-independent operations

- Use a data structure built upon radix trees
- Quantum memory layout: uniform superposition of all possible classical layout

The next quantum walk needs to work

- The quantum states after extraction must be nodes in the graph

Assumptions

Efficient history-independent operations

- Use a data structure built upon radix trees
- Quantum memory layout: uniform superposition of all possible classical layout

The next quantum walk needs to work

- The quantum states after extraction must be nodes in the graph
- It is fine to start from collision-free nodes

Assumptions

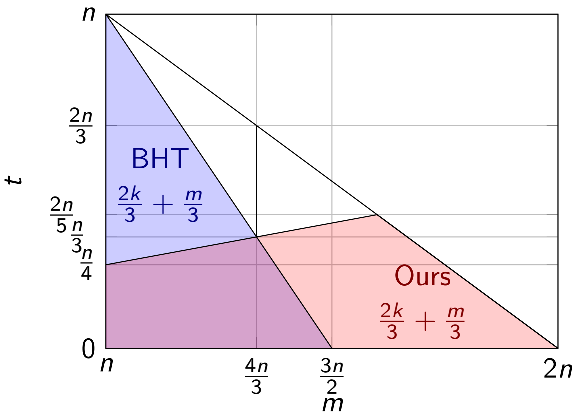
Efficient history-independent operations

- Use a data structure built upon radix trees
- Quantum memory layout: uniform superposition of all possible classical layout

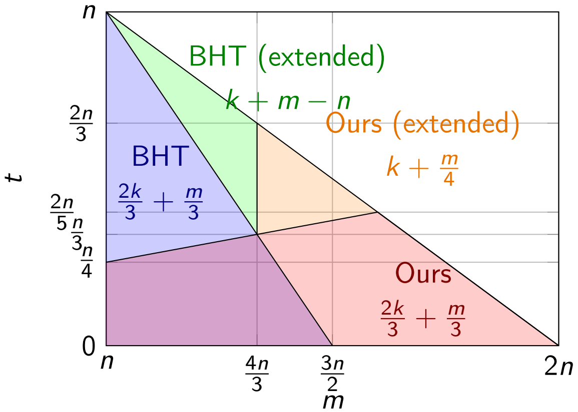
The next quantum walk needs to work

- The quantum states after extraction must be nodes in the graph
- It is fine to start from collision-free nodes
- Nodes with collisions are a small fraction of the nodes

Quantum collisions now



Quantum collisions now



With golden collisions

Golden collisions

Find (x, y) such that $f(x) = f(y)$, plus $P(x, y)$ is true.

Algorithm

The same algorithm works:

- Add the test in the walk
- Count/extract only golden collisions
- Works if a random node contains a *golden* collision with small probability.

Quantum lattice sieving

Lattice sieving

- Start with many vectors v_i
- Find many $v_i \pm v_j$ with smaller norm
- Iterate

Quantum lattice sieving

Lattice sieving

- Start with many vectors v_i
- Find many $v_i \pm v_j$ with smaller norm
- Iterate

Quantum Lattice sieving [CL21]

- Find good $v_i \pm v_j$ with a quantum walk
- Locality sensitive filtering:
  - Take a code,
  - Close v_i tend to decode to the same value

Quantum lattice sieving

Lattice sieving

- Start with many vectors v_i
- Find many $v_i \pm v_j$ with smaller norm
- Iterate

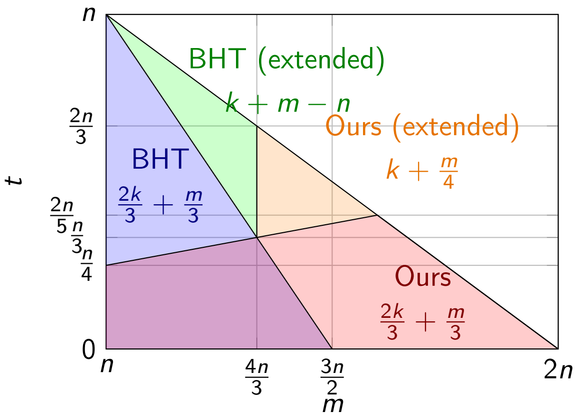
Quantum Lattice sieving [CL21]

- Find good $v_i \pm v_j$ with a quantum walk
- Locality sensitive filtering:
  - Take a code,
  - Close v_i tend to decode to the same value

Improvement

- Original quantum walk $2^{0.2570d+o(d)}$
- Improved quantum walk $2^{0.2563d+o(d)}$

Thank you!



BHT (extended)
$k+m-n$
Ours (extended)
$k+\frac{m}{4}$
BHT
$\frac{2k}{3}+\frac{m}{3}$
Ours
$\frac{2k}{3}+\frac{m}{3}$
n
$\frac{2n}{3}$
$\frac{2n}{5}$
$\frac{n}{3}$
$\frac{n}{4}$
0
t
n
$\frac{4n}{3}$
$\frac{3n}{2}$
$2n$
m