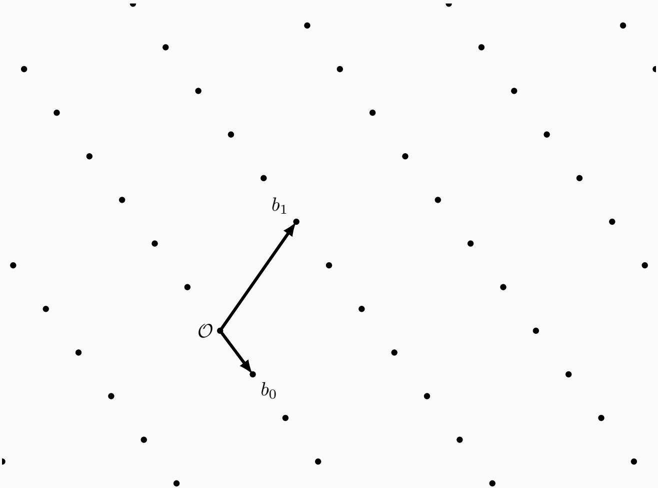
Quantum Algorithms for the Approximate k-List Problem and their Application to Lattice Sieving

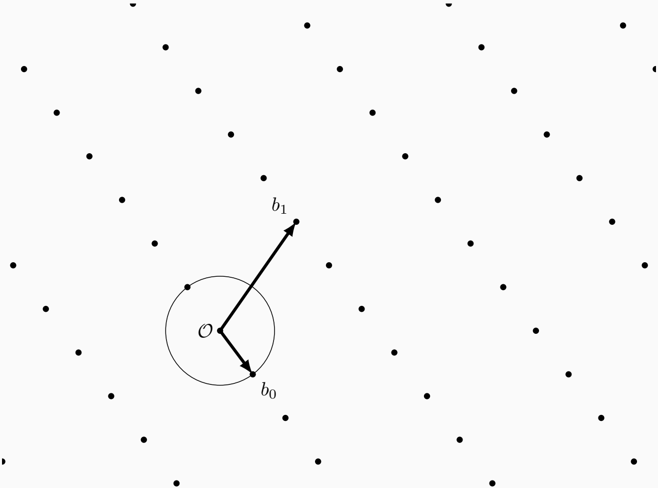
Elena Kirshanova[1], Erik Mårtensson[2], Eamonn W. Postlethwaite[3], Subhayan Roy Moulik[4]

December 6, 2019

[1]I. Kant Baltic Federal University
[2]Lund University
[3]Royal Holloway, University of London
[4]University of Oxford



$\Lambda = \mathrm{Span}_{\mathbb{Z}}(b_0, \ldots, b_{d-1})$, $B = \{b_0, \ldots, b_{d-1}\} \subset \mathbb{R}^d$ a basis



SVP: find $v \in \Lambda \setminus \{0\}$ such that $\|v\|_2 \leq \|w\|_2$ for all $w \in \Lambda \setminus \{0\}$

Some Concrete Sieves

algorithm	variant	$\log_2$time	$\log_2$memory[1]	tradeoffs
sieving	–	$\Theta(d)$	$\Theta(d)$	–
[HK17][2]	h, c	$0.396d$	$0.189d$	yes
[LMvdP15]	h, q	$0.312d$	$0.208d$	no
[HKL18]	h, c, n	$0.359d$	$0.189d$	yes
[Laa16]	h, q, n	$0.265d$	$0.265d$	yes[3]

h: heuristic, n: near neighbour, c: classical, q: quantum

[1] classical memory, requiring poly(d) width quantum circuits and assuming QRAM

[2] all table values here, e.g. $0.396d$, are missing $o(d)$ terms

[3] although we provide new tradeoffs

Some Concrete Sieves

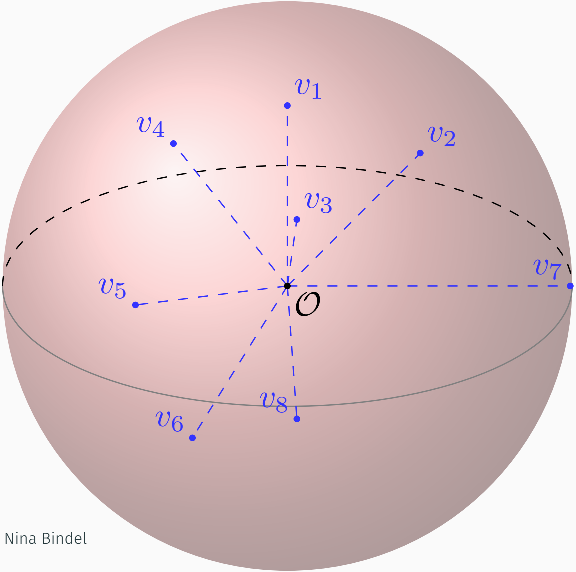
algorithm	variant	$\log_2$time	$\log_2$memory[1]	tradeoffs
sieving	–	$\Theta(d)$	$\Theta(d)$	–
[HK17][2]	h, c	$0.396d$	$0.189d$	yes
(this work)	h, q	$0.312d \rightarrow 0.299d$	$0.208d \rightarrow 0.140d$	yes!
[HKL18]	h, c, n	$0.359d$	$0.189d$	yes
[Laa16]	h, q, n	$0.265d$	$0.265d$	yes[3]

h: heuristic, n: near neighbour, c: classical, q: quantum

[1] classical memory, requiring poly(d) width quantum circuits and assuming QRAM

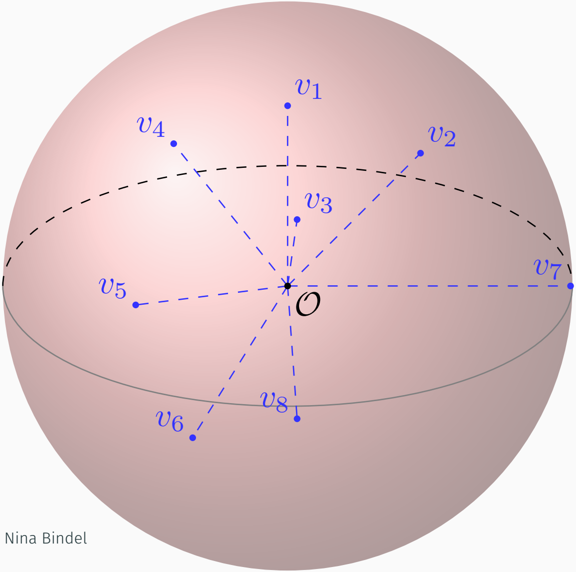
[2] all table values here, e.g. $0.396d$, are missing $o(d)$ terms

[3] although we provide new tradeoffs



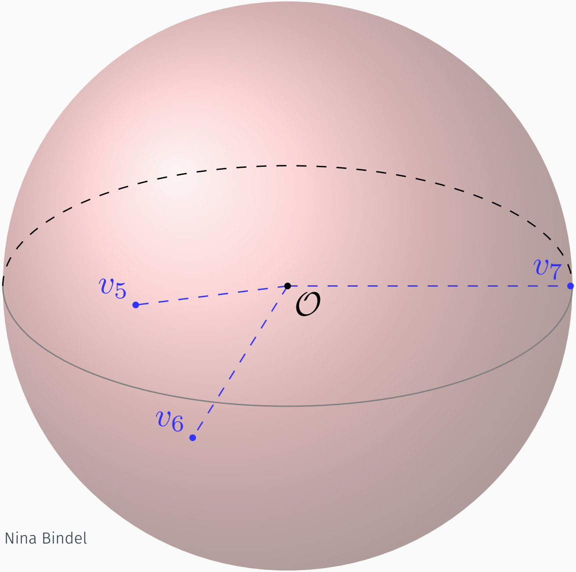
Template Credit: Nina Bindel

heuristic says (after normalisation) v_i are i.i.d. uniform on S^{d-1}



Template Credit: Nina Bindel

search for ($k = 3$ case) v_a, v_b, v_c such that $\|v_a + v_b + v_c\|_2 \leq 1$



Template Credit: Nina Bindel

search for ($k = 3$ case) v_a, v_b, v_c such that $\|v_a + v_b + v_c\|_2 \leq 1$

Configuration Method of [HK17]

If we consider $v_1, \ldots, v_k \in S^{d-1}$ and the Gram matrix

$$C = \begin{pmatrix} \langle v_1, v_1 \rangle & \cdots & \langle v_1, v_k \rangle \\ \vdots & \ddots & \vdots \\ \langle v_k, v_1 \rangle & \cdots & \langle v_k, v_k \rangle \end{pmatrix} = \begin{pmatrix} 1 & \cdots & \langle v_1, v_k \rangle \\ \vdots & \ddots & \vdots \\ \langle v_k, v_1 \rangle & \cdots & 1 \end{pmatrix}$$

then

$$\|v_1 + \cdots + v_k\|_2 \leq 1 \iff \mathbb{1}^t C \mathbb{1} \leq 1, \text{ “good”.}$$

Definition (approximate k-List problem)
Given list $L \subset S^{d-1}$ of i.i.d. uniform elements, find $|L|$ tuples $(v_1, \dots, v_k) \in L \times \cdots \times L$ such that $\|v_1 + \cdots + v_k\|_2 \leq 1$.

[4] up to some small distance ε

Definition (approximate k-List problem)
Given list $L \subset S^{d-1}$ of i.i.d. uniform elements, find $|L|$ tuples $(v_1, \ldots, v_k) \in L \times \cdots \times L$ such that $\|v_1 + \cdots + v_k\|_2 \leq 1$.

We solve the approximate k-List problem by

- picking a good C,

[4] up to some small distance ε

Definition (approximate k-List problem)
Given list $L \subset S^{d-1}$ of i.i.d. uniform elements, find $|L|$ tuples $(v_1, \ldots, v_k) \in L \times \cdots \times L$ such that $\|v_1 + \cdots + v_k\|_2 \leq 1$.

We solve the approximate k-List problem by

- picking a good C,
- generating a sufficiently large L for this C,

[4] up to some small distance ε

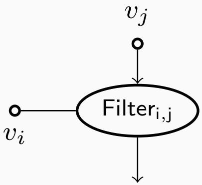
Configuration Method of [HK17]

Definition (approximate k-List problem)
Given list $L \subset S^{d-1}$ of i.i.d. uniform elements, find $|L|$ tuples $(v_1, \ldots, v_k) \in L \times \cdots \times L$ such that $\|v_1 + \cdots + v_k\|_2 \leq 1$.

We solve the approximate k-List problem by

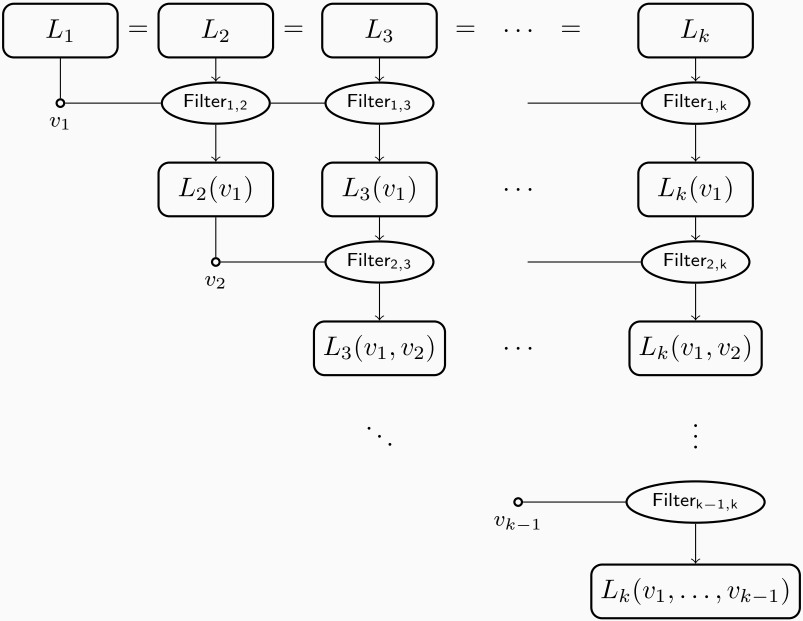
- picking a good C,
- generating a sufficiently large L for this C,
- finding a $1 - o(1)$ fraction of tuples in $L \times \cdots \times L$ with configuration C,[4]

[4] up to some small distance ε



v_j passes through Filter$_{\mathsf{i,j}}$ given v_i iff

$$|\langle v_i, v_j \rangle - C_{i,j}| < \varepsilon$$


L_1
=
L_2
=
L_3
= ⋯ =
L_k
$\mathsf{Filter}_{1,2}$
$\mathsf{Filter}_{1,3}$
$\mathsf{Filter}_{1,k}$
v_1
$L_2(v_1)$
$L_3(v_1)$
$L_k(v_1)$
$\mathsf{Filter}_{2,3}$
$\mathsf{Filter}_{2,k}$
v_2
$L_3(v_1, v_2)$
$L_k(v_1, v_2)$
$\mathsf{Filter}_{k-1,k}$
v_{k-1}
$L_k(v_1, \ldots, v_{k-1})$

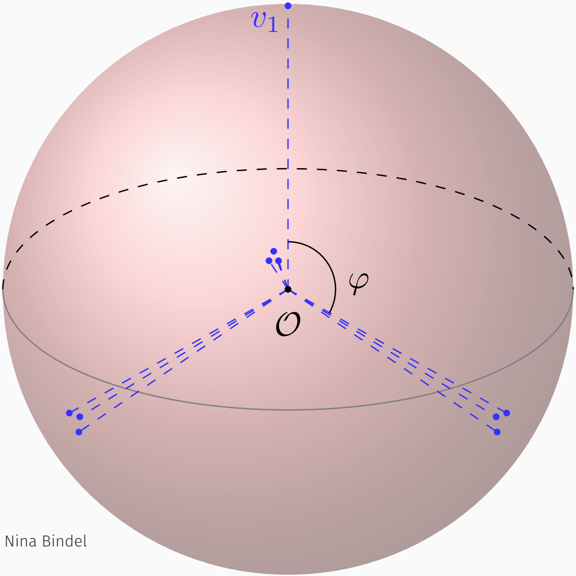
Memory optimal C; a starting point

Most solutions are concentrated around

$$C = \begin{pmatrix} 1 & -1/k & \cdots & -1/k \\ -1/k & 1 & \cdots & -1/k \\ \vdots & & \ddots & \vdots \\ -1/k & -1/k & \cdots & 1 \end{pmatrix},$$

$\Rightarrow$ using this configuration requires smallest $L \leftrightarrow$ least memory.

Memory optimal C; a starting point



$\varphi \approx \arccos(-1/3)$, the “edge central angle”

Memory optimal C; a starting point

Most solutions are concentrated around

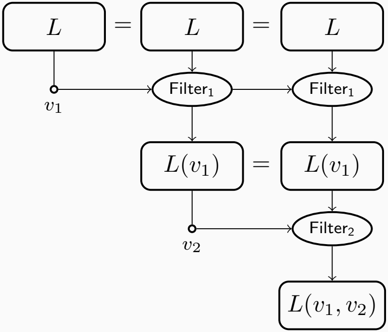
$$C = \begin{pmatrix} 1 & -1/k & \cdots & -1/k \\ -1/k & 1 & \cdots & -1/k \\ \vdots & & \ddots & \vdots \\ -1/k & -1/k & \cdots & 1 \end{pmatrix},$$

$\Rightarrow$ using this configuration requires smallest $L \leftrightarrow$ least memory.

Larger $L \leftrightarrow$ more available good $C \leftrightarrow$ more memory, but faster.

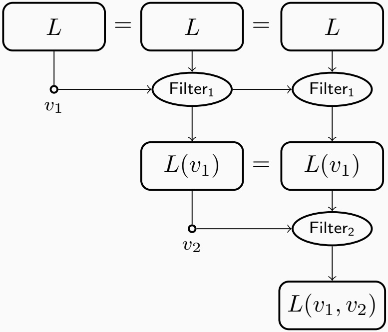
Running example, classical

procedure 3-List, memory optimal C
    for $v_1 \in L$ **do**
        $L(v_1) \leftarrow \{v_2 \colon |\langle v_1, v_2 \rangle| \approx 1/3\}$
        for $v_2 \in L(v_1)$ **do**
            $L(v_1, v_2) \leftarrow \{v_3 \colon |\langle v_2, v_3 \rangle| \approx 1/3\}$
            output all $(v_1, v_2, v_3 \in L(v_1, v_2))$
        end for
    end for
end procedure



Running example, classical

procedure 3-LIST, MEMORY OPTIMAL C
    for $v_1 \in L$ **do**
        $L(v_1) \leftarrow \{v_2 \colon |\langle v_1, v_2\rangle| \approx 1/3\}$
        for $v_2 \in L(v_1)$ **do**
            $L(v_1, v_2) \leftarrow \{v_3 \colon |\langle v_2, v_3\rangle| \approx 1/3\}$
            output all $(v_1, v_2, v_3 \in L(v_1, v_2))$
        end for
    end for
end procedure



$$T_C = |L| \cdot \left(|L| + |L(v_1)|^2\right) = 2^{0.396d + o(d)}$$

tldr; Grover square roots unstructured list search

For some v_1, to find a "good" $v_2 \in L$

Notation	Method	Complexity
$\xrightarrow[\lvert\langle v_1, v_2\rangle\rvert \approx 1/3]{\textit{Brute}}$	check $\lvert\langle v_1, v_2\rangle\rvert \approx 1/3$ for each v_2	$O(\lvert L\rvert)$
$\xrightarrow[\lvert\langle v_1, v_2\rangle\rvert \approx 1/3]{\textit{Grover}}$	run quantum circuit encoding $\lvert\langle v_1, v_2\rangle\rvert \approx 1/3$ over superposition of all v_2	$O(\sqrt{\lvert L\rvert})$

Running example, quantum

Given v_1, use Grover to find one triple, deferring measurement

1. $\displaystyle \frac{1}{|L|} \sum_{v_2, v_3 \in L} |v_2\rangle |v_3\rangle \xrightarrow[|\langle v_1, v_2 \rangle| \approx 1/3]{\textit{Grover}} \quad \xrightarrow[|\langle v_1, v_3 \rangle| \approx 1/3]{\textit{Grover}}$

Running example, quantum

Given v_1, use Grover to find one triple, deferring measurement

1. $\frac{1}{|L|}\sum_{v_2, v_3 \in L} |v_2\rangle|v_3\rangle \xrightarrow[|\langle v_1, v_2\rangle| \approx 1/3]{Grover} \quad \xrightarrow[|\langle v_1, v_3\rangle| \approx 1/3]{Grover}$
2. $\frac{1}{|L(v_1)|}\sum_{v_2, v_3 \in L(v_1)} |v_2\rangle|v_3\rangle \xrightarrow[|\langle v_2, v_3\rangle| \approx 1/3]{Grover}$

Running example, quantum

Given v_1, use Grover to find one triple, deferring measurement

1. $$\frac{1}{|L|}\sum_{v_2,v_3\in L}|v_2\rangle|v_3\rangle \xrightarrow[|\langle v_1,v_2\rangle|\approx 1/3]{\textit{Grover}} \quad \xrightarrow[|\langle v_1,v_3\rangle|\approx 1/3]{\textit{Grover}}$$
2. $$\frac{1}{|L(v_1)|}\sum_{v_2,v_3\in L(v_1)}|v_2\rangle|v_3\rangle \xrightarrow[|\langle v_2,v_3\rangle|\approx 1/3]{\textit{Grover}}$$
3. $$\frac{1}{O(1)}\sum_{v_2,v_3 \text{ good}}|v_2\rangle|v_3\rangle$$

Running example, quantum

Given v_1, use Grover to find one triple, deferring measurement

1. $$\frac{1}{|L|}\sum_{v_2,v_3\in L}|v_2\rangle|v_3\rangle \xrightarrow[|\langle v_1,v_2\rangle|\approx 1/3]{\textit{Grover}} \quad \xrightarrow[|\langle v_1,v_3\rangle|\approx 1/3]{\textit{Grover}}$$
2. $$\frac{1}{|L(v_1)|}\sum_{v_2,v_3\in L(v_1)}|v_2\rangle|v_3\rangle \xrightarrow[|\langle v_2,v_3\rangle|\approx 1/3]{\textit{Grover}}$$
3. $$\frac{1}{O(1)}\sum_{v_2,v_3 \text{ good}}|v_2\rangle|v_3\rangle$$
4. Measure the state, check it, and output (v_1, v_2, v_3)

Running example, quantum

Given v_1, use Grover to find one triple, deferring measurement

1. $\displaystyle \frac{1}{|L|}\sum_{v_2,v_3\in L}|v_2\rangle|v_3\rangle \xrightarrow[|\langle v_1,v_2\rangle|\approx 1/3]{\textit{Grover}} \quad \xrightarrow[|\langle v_1,v_3\rangle|\approx 1/3]{\textit{Grover}}$
2. $\displaystyle \frac{1}{|L(v_1)|}\sum_{v_2,v_3\in L(v_1)}|v_2\rangle|v_3\rangle \xrightarrow[|\langle v_2,v_3\rangle|\approx 1/3]{\textit{Grover}}$
3. $\displaystyle \frac{1}{O(1)}\sum_{v_2,v_3 \text{ good}}|v_2\rangle|v_3\rangle$
4. Measure the state, check it, and output (v_1, v_2, v_3)

$$T_q = |L|\cdot\left(\sqrt{\frac{|L|}{|L(v_1)|}}\cdot|L(v_1)|\right) = 2^{0.335d+o(d)} < T_c = 2^{0.396d+o(d)}$$

By generalising these arguments to varying $k = \Theta(1)$ and optimising over good C for time[5] (rather than memory)

$$\log_2 \text{time} = 0.299d \qquad \log_2 \text{memory} = 0.140d$$

[5] again, all missing $o(d)$ terms

By generalising these arguments to varying $k = \Theta(1)$ and optimising over good C for time[5] (rather than memory)

$$\log_2 \text{time} = 0.299d \qquad \log_2 \text{memory} = 0.140d$$

Compare, in the same model, the "generic" quantum 2-sieve

$$\log_2 \text{time} = 0.312d \qquad \log_2 \text{memory} = 0.208d$$

[5] again, all missing $o(d)$ terms

By generalising these arguments to varying $k = \Theta(1)$ and optimising over good C for time[5] (rather than memory)

$$\log_2 \text{time} = 0.299d \qquad \log_2 \text{memory} = 0.140d$$

Compare, in the same model, the "generic" quantum 2-sieve

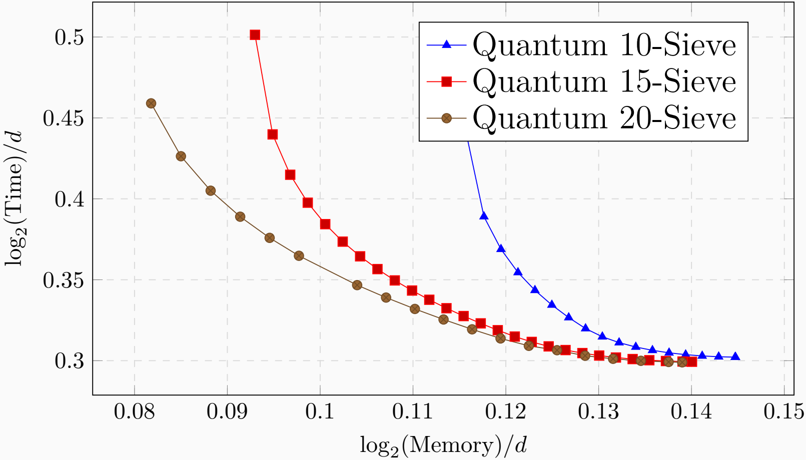
$$\log_2 \text{time} = 0.312d \qquad \log_2 \text{memory} = 0.208d$$

and the best known sieving time complexity

$$\log_2 \text{time} = 0.265d \qquad \log_2 \text{memory} = 0.265d$$

[5] again, all missing $o(d)$ terms

Explicit tradeoffs for specific k



Other contributions of the paper

- embed the problem into a graph and generalise clique listing algorithms in the *QRAM* model
- specialise to 3 cliques (triangles) and solve in the *query* model, with queries to an adjacency matrix
- give an algorithm in the *quantum circuit* model, for $k = 2$, using parallel quantum search [BBG+13] with $(\log_2 \text{depth}, \log_2 \text{width}) = (0.104d, 0.208d)$

Thanks!

- new time/memory tradeoffs = faster low memory sieves
- several approaches in different computational models
- open questions: prove time optimal limit, generalise quantum circuit algorithm to $k > 2$, move from query model to QRAM model...

`https://eprint.iacr.org/2019/1016`

Robert Beals, Stephen Brierley, Oliver Gray, Aram W. Harrow, Samuel Kutin, Noah Linden, Dan Shepherd, and Mark Stather, *Efficient distributed quantum computing*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **469** (2013), no. 2153, 20120686.

Gottfried Herold and Elena Kirshanova, *Improved algorithms for the approximate k-list problem in euclidean norm*, PKC 2017, Part I (Serge Fehr, ed.), LNCS, vol. 10174, Springer, Heidelberg, March 2017, pp. 16–40.

Gottfried Herold, Elena Kirshanova, and Thijs Laarhoven, *Speed-ups and time-memory trade-offs for tuple lattice sieving*, PKC 2018, Part I (Michel Abdalla and Ricardo Dahab, eds.), LNCS, vol. 10769, Springer, Heidelberg, March 2018, pp. 407–436.

- Thijs Laarhoven, *Search problems in cryptography: from fingerprinting to lattice sieving*, Ph.D. thesis, Department of Mathematics and Computer Science, 2 2016, Proefschrift.
- Thijs Laarhoven, Michele Mosca, and Joop van de Pol, *Finding shortest lattice vectors faster using quantum search*, Designs, Codes and Cryptography **77** (2015), no. 2, 375–400.