Integer Factorization using lattices

Antonio Vera

INRIA Nancy/CARAMEL team/ANR CADO/ANR LAREDA

Workshop Lattice Algorithmics - CIRM - February 2010
Plan

Introduction
Plan

Introduction

Outline of the algorithm
Plan

Introduction

Outline of the algorithm

The search for "relations"
Plan

Introduction

Outline of the algorithm

The search for “relations”

Termination considerations
Plan

Introduction

Outline of the algorithm

The search for “relations”

Termination considerations

Conclusions
Outline

Introduction

Outline of the algorithm

The search for “relations”

Termination considerations

Conclusions
The Integer Factorization Problem

Given a composite integer $N$, compute a proper divisor.
The Integer Factorization Problem

- Given a composite integer \( N \), compute a proper divisor.
- Example: given \( N = 6 \), return 2 or 3.

The decisional problem belongs to \( \text{NP} \cap \text{coNP} \) (Pratt, 1975). Hence, it is unlikely to be \( \text{NP-complete} \).

Practical intractability is at the base of the RSA public key cryptosystem.

Current record for factorization of RSA numbers: RSA 768 (232 decimal digits), in December 2009.
The Integer Factorization Problem

- Given a composite integer $N$, compute a proper divisor.
- Example: given $N = 6$, return 2 or 3.
- The decisional problem belongs to $NP \cap coNP$ (Pratt, 1975).
The Integer Factorization Problem

- Given a composite integer $N$, compute a proper divisor.
- Example: given $N = 6$, return 2 or 3.
- The decisional problem belongs to $\text{NP} \cap \text{coNP}$ (Pratt, 1975).
- Hence, it is unlikely to be NP-complete.
The Integer Factorization Problem

- Given a composite integer \( N \), compute a proper divisor.
- Example: given \( N = 6 \), return 2 or 3.
- The decisional problem belongs to \( \text{NP} \cap \text{coNP} \) (Pratt, 1975).
- Hence, it is unlikely to be \( \text{NP}\)-complete.
- Practical intractability is at the base of the RSA public key cryptosystem.

Current record for factorization of RSA numbers: RSA 768 (232 decimal digits), in December 2009.
The Integer Factorization Problem

- Given a composite integer $N$, compute a proper divisor.
- Example: given $N = 6$, return 2 or 3.
- The decisional problem belongs to $\text{NP} \cap \text{coNP}$ (Pratt, 1975).
- Hence, it is unlikely to be $\text{NP}$-complete.
- Practical intractability is at the base of the RSA public key cryptosystem.
De Weger (1987) used lattice reduction and enumeration to effectively solve Diophantine equations.
De Weger (1987) used lattice reduction and enumeration to effectively solve Diophantine equations. Schnorr seems to be the first which applied this to integer factorization, in 1993.
De Weger (1987) used lattice reduction and enumeration to effectively solve Diophantine equations.

Schnorr seems to be the first which applied this to integer factorization, in 1993.

In the beginning, completely theoretical.
De Weger (1987) used lattice reduction and enumeration to effectively solve Diophantine equations.

Schnorr seems to be the first which applied this to integer factorization, in 1993.

In the beginning, completely theoretical.

First working implementation: Ritter and Rössner, 1997. 60-bit integer factored in 3 hours. Highly optimized enumeration.
De Weger (1987) used lattice reduction and enumeration to effectively solve Diophantine equations.

Schnorr seems to be the first which applied this to integer factorization, in 1993.

In the beginning, completely theoretical.

First working implementation: Ritter and Rössner, 1997. 60-bit integer factored in 3 hours. Highly optimized enumeration.

Personal toy implementation in Magma, using the built-in enumeration routine. 40-bits integer in about 40 minutes.
Preliminary goals

- Improve the results of the original paper to make them effective.

- Understand the complexity of the algorithm.

- Implement an optimized version, a priori aimed at fixed-size integers.
Preliminary goals

- Improve the results of the original paper to make them effective.
- Understand the complexity of the algorithm.
Preliminary goals

- Improve the results of the original paper to make them effective.
- Understand the complexity of the algorithm.
- Implement an optimized version, à priori aimed at fixed-size integers.
Outline

Introduction

Outline of the algorithm

The search for “relations”

Termination considerations

Conclusions
As in many factorization methods, we want to find integers \( x, y \in \mathbb{Z} \), \( x \neq \pm y \) such that

\[
x^2 \equiv y^2 \pmod{N}.
\]
Philosophy of the algorithm
Congruence of squares method

- As in many factorization methods, we want to find integers $x, y \in \mathbb{Z}$, $x \neq \pm y$ such that
  
  $$x^2 \equiv y^2 \mod N.$$ 

- Then, whenever $x \not\equiv \pm y \mod N$, we can factor $N$ by computing $\text{gcd}(x \pm y, N)$. 

Philosophy of the algorithm
Congruence of squares method

- As in many factorization methods, we want to find integers $x, y \in \mathbb{Z}$, $x \neq \pm y$ such that

$$x^2 \equiv y^2 \mod N.$$ 

- Then, whenever $x \not\equiv \pm y \mod N$, we can factor $N$ by computing $\gcd(x \pm y, N)$. 

Definition
An integer is $B$-smooth when all of its prime factors are $\leq B$.

- 100 is 5-smooth since $100 = 2^2 \cdot 5^2$.

- All integers $\leq B$ are $B$-smooth.
Philosophy of the algorithm

Congruence of squares method

- As in many factorization methods, we want to find integers $x, y \in \mathbb{Z}, x \neq \pm y$ such that

$$x^2 \equiv y^2 \mod N.$$  

- Then, whenever $x \not\equiv \pm y \mod N$, we can factor $N$ by computing $\gcd(x \pm y, N)$.

Definition

An integer is $B$-smooth when all of its prime factors are $\leq B$. 
Philosophy of the algorithm
Congruence of squares method

As in many factorization methods, we want to find integers \( x, y \in \mathbb{Z}, \ x \neq \pm y \) such that

\[ x^2 \equiv y^2 \pmod{N}. \]

Then, whenever \( x \ncong \pm y \pmod{N} \), we can factor \( N \) by computing \( \gcd(x \pm y, N) \).

Definition
An integer is \( B \)-smooth when all of its prime factors are \( \leq B \).

100 is 5-smooth since \( 100 = 2^2 \cdot 5^2 \).
Philosophy of the algorithm

Congruence of squares method

- As in many factorization methods, we want to find integers $x, y \in \mathbb{Z}$, $x \neq \pm y$ such that
  
  $x^2 \equiv y^2 \mod N$.

- Then, whenever $x \not\equiv \pm y \mod N$, we can factor $N$ by computing $\gcd(x \pm y, N)$.

**Definition**

An integer is $B$-smooth when all of its prime factors are $\leq B$.

- 100 is 5-smooth since $100 = 2^2 \cdot 5^2$.
- All integers $\leq B$ are $B$-smooth.
Factorization algorithm principle

Given an integer $N$ to be factored, do the following:
Factorization algorithm principle

Given an integer $N$ to be factored, do the following:

- Fix $d \geq 1$ and build the factor base

\[ \{p_0, \ldots, p_d\}, \quad \text{where } p_0 = -1, \ p_1 = 2, \ p_2 = 3, \ p_3 = 5, \ldots. \]
Given an integer $N$ to be factored, do the following:

- Fix $d \geq 1$ and build the factor base

\[ \{p_0, \ldots, p_d\}, \quad \text{where } p_0 = -1, \ p_1 = 2, \ p_2 = 3, \ p_3 = 5, \ldots. \]

- Find $d + 2$ solutions of

\[ u = v + kN \]

where $k \in \mathbb{Z}$, and $u$, $v$ are $p_d$-smooth.
Factorization algorithm principle

Given an integer $N$ to be factored, do the following:

- Fix $d \geq 1$ and build the factor base

$$\{p_0, \ldots, p_d\}, \quad \text{where } p_0 = -1, \ p_1 = 2, \ p_2 = 3, \ p_3 = 5, \ldots.$$

- Find $d + 2$ solutions of

$$u = v + kN$$

where $k \in \mathbb{Z}$, and $u, \ v$ are $p_d$-smooth.
Factorization algorithm principle

Given an integer \( N \) to be factored, do the following:

▶ Fix \( d \geq 1 \) and build the factor base

\[
\{p_0, \ldots, p_d\}, \quad \text{where } p_0 = -1, \ p_1 = 2, \ p_2 = 3, \ p_3 = 5, \ldots.
\]

▶ Find \( d + 2 \) solutions of

\[
u = v + kN
\]

where \( k \in \mathbb{Z} \), and \( u, v \) are \( p_d \)-smooth. Write the identities in the following form

\[
\prod_{j=0}^{d} p_j^{a_{i,j}} \equiv \prod_{j=0}^{d} p_j^{b_{i,j}} \mod N \quad i \in [1, d + 2].
\]

These last equalities are called \textit{relations}. 
Factorization algorithm principle

- Define the *exponent vectors*

\[
a_i = (a_{i,0}, \ldots, a_{i,d}), \quad b_i = (b_{i,0}, \ldots, b_{i,d}),
\]

and compute a vector \( c = (c_1, \ldots, c_{d+2}) \in \{0, 1\}^{d+2} \) such that

\[
\sum_{i=1}^{d+2} c_i (a_i + b_i) = 0 \mod 2.
\]
Factorization algorithm principle

- Define the exponent vectors

\[ a_i = (a_{i,0}, \ldots, a_{i,d}), \quad b_i = (b_{i,0}, \ldots, b_{i,d}), \]

and compute a vector \( c = (c_1, \ldots, c_{d+2}) \in \{0, 1\}^{d+2} \) such that

\[ \sum_{i=1}^{d+2} c_i (a_i + b_i) = 0 \mod 2. \]

- Define

\[
    x = \prod_{j=0}^{d} p_j^{\sum_{i=1}^{d+2} c_i (a_{i,j}+b_{i,j})/2} \mod N,
\]

and

\[
    y = \prod_{j=0}^{d} p_j^{\sum_{i=1}^{d+2} c_i a_{i,j}} = \prod_{j=0}^{d} p_j^{\sum_{i=1}^{d+2} c_i b_{i,j}} \mod N.
\]

If \( x \neq y \mod N \) then return \( \gcd(x+y, N) \). Else, restart with another vector \( c \). If there are not more of them, fail.
Factorization algorithm principle

- Define the exponent vectors

\[ a_i = (a_{i,0}, \ldots, a_{i,d}), \quad b_i = (b_{i,0}, \ldots, b_{i,d}), \]

and compute a vector \( c = (c_1, \ldots, c_{d+2}) \in \{0, 1\}^{d+2} \) such that

\[
\sum_{i=1}^{d+2} c_i (a_i + b_i) = 0 \mod 2.
\]

- Define

\[
x = \prod_{j=0}^{d} \prod_{i=1}^{d+2} p_j^{c_i(a_{i,j}+b_{i,j})/2} \mod N,
\]

and

\[
y = \prod_{j=0}^{d} \prod_{i=1}^{d+2} p_j^{c_i a_{i,j}} = \prod_{j=0}^{d} \prod_{i=1}^{d+2} p_j^{c_i b_{i,j}} \mod N.
\]

- If \( x \neq y \mod N \) then return \( \gcd(x + y, N) \). Else, restart with another vector \( c \). If there are not more of them, fail.
Outline

Introduction

Outline of the algorithm

The search for “relations”

Termination considerations

Conclusions
Finding relations

- The core of the algorithm is the search for relations.
Finding relations

- The core of the algorithm is the search for relations.
- We will see that lattices help in this search.
Finding relations

- The core of the algorithm is the search for relations.
- We will see that lattices help in this search.
- Restate the equation we want to solve as

\[ u - kN = v, \]

for \( u, v \) \( p_d \)-smooth and \( k \in \mathbb{Z} \).
Finding relations

- The core of the algorithm is the search for **relations**.
- We will see that lattices help in this search.
- Restate the equation we want to solve as

  \[ u - kN = v, \]

  for \( u, v \) \( p_d \)-smooth and \( k \in \mathbb{Z} \).

- We can forget the constraint of \( p_d \)-smoothness on \( v \) by solving

  \[ |u - kN| \leq p_d \]

  for \( u \) \( p_d \)-smooth and \( k \in \mathbb{Z} \).
Finding relations

This kind of equation can be solved by working on the exponents

\[ |u - kN| = |e^{\ln u} - e^{\ln kN}|, \]

trying to make

\[ |\ln u - \ln kN| \]

as small as possible, without increasing \( \ln u \) and \( \ln k \) too much.
Finding relations

- This kind of equation can be solved by working on the exponents

\[ |u - kN| = |e^{\ln u} - e^{\ln kN}|, \]

trying to make

\[ |\ln u - \ln kN| \]

as small as possible, without increasing \( \ln u \) and \( \ln k \) too much.

- Schnorr codes a \((u, k)\) pair in a vector

\[ z = (z_1, \ldots, z_d) \in \mathbb{Z}^d, \]

from which one recovers \( u \) and \( k \) by letting

\[ u = \prod_{z_i > 0} p_i^{z_i} \quad \text{and} \quad k = \prod_{z_i < 0} p_i^{-z_i}. \]
Finding relations

With this notation,

\[
\ln u = \sum_{z_i > 0} z_i \ln p_i \quad \text{and} \quad \ln k = \sum_{z_i < 0} -z_i \ln p_i.
\]
Finding relations

- With this notation,
  \[ \ln u = \sum_{z_i > 0} z_i \ln p_i \quad \text{and} \quad \ln k = \sum_{z_i < 0} -z_i \ln p_i. \]

- We want the two quantities
  \[ | \ln u - \ln kN | = \left| \sum_{i=1}^{d} z_i \ln p_i - \ln N \right| \]
  \[ \ln u + \ln k = \sum_{i=1}^{d} |z_i| \ln p_i. \]

  to be small.
Finding relations

- With this notation,

\[ \ln u = \sum_{z_i > 0} z_i \ln p_i \quad \text{and} \quad \ln k = \sum_{z_i < 0} -z_i \ln p_i. \]

- We want the two quantities

\[ |\ln u - \ln kN| = \left| \sum_{i=1}^{d} z_i \ln p_i - \ln N \right| \]

\[ \ln u + \ln k = \sum_{i=1}^{d} |z_i| \ln p_i. \]

to be small.

- This is where lattices enter the scene.
Schnorr’s approach to factorization
The Prime Number Lattice

Define the *Prime Number Basis* and the *target vector* by

\[ S_p = \begin{bmatrix} \sqrt{\ln p_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sqrt{\ln p_d} \\ C \ln p_1 & \cdots & C \ln p_d \end{bmatrix} \quad \text{and} \quad t = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ C \ln N \end{bmatrix} \]
Schnorr’s approach to factorization
The Prime Number Lattice

Define the *Prime Number Basis* and the *target vector* by

\[
S_p = \begin{bmatrix}
\sqrt{\ln p_1} & 0 & 0 \\
0 & \ddots & 0 \\
0 & 0 & \sqrt{\ln p_d}
\end{bmatrix}
\quad \text{and} \quad
\mathbf{t} = \begin{bmatrix}
0 \\
\vdots \\
0 \\
C \ln N
\end{bmatrix}.
\]

Let \( \mathbf{z} \in \mathbb{Z}^d \) be a *coordinate vector*. We have

\[
S_p \mathbf{z} - \mathbf{t} = \begin{bmatrix}
z_1 \sqrt{\ln p_1} \\
\vdots \\
z_d \sqrt{\ln p_d} \\
C(\sum_{i=1}^d z_i \sqrt{\ln p_i \ln N})
\end{bmatrix}.
\]
Schnorr’s approach to factorization
Computing a relation using lattices.

- The norm is given by

\[ \| \mathbf{S}_p \mathbf{z} - \mathbf{t} \|^p = \sum_{i=1}^{d} |z_i|^p \ln p_i + C^p \left| \sum_{i=1}^{d} z_i \ln p_i - \ln N \right|^p. \]
Schnorr’s approach to factorization

Computing a relation using lattices.

- The norm is given by

\[
\|S_pz - t\|_p^p = \sum_{i=1}^{d} |z_i|^p \ln p_i + C^p \left| \sum_{i=1}^{d} z_i \ln p_i - \ln N \right|^p.
\]

- By recycling a result of Micciancio [2, Prop. 5.10, page 105], one proves:
Schnorr’s approach to factorization
Computing a relation using lattices.

- The norm is given by

\[ \|S_p z - t\|^p_p = \sum_{i=1}^{d} |z_i|^p \ln p_i + Cp \left| \sum_{i=1}^{d} z_i \ln p_i - \ln N \right|^p \].

- By recycling a result of Micciancio [2, Prop. 5.10, page 105], one proves:

**Theorem**

Let \( C \geq 3 \). Then, for any nonzero integer vector \( z \) such that

\[ \|S_1 z - t\|_1 \leq 2 \ln C + 2 \ln p_d - \ln N, \]

we have

\[ |u - kN| \leq p_d. \]
Schnorr’s approach to factorization

- We have a polyhedron where every element of the lattice inside this polyhedron yields a relation.
Schnorr’s approach to factorization

- We have a polyhedron where every element of the lattice inside this polyhedron yields a relation.
- Schnorr shows that for some choice of the parameters this polyhedron is expected to contain an exponential number of relations.
Schnorr’s approach to factorization

- We have a polyhedron where every element of the lattice inside this polyhedron yields a relation.
- Schnorr shows that for some choice of the parameters this polyhedron is expected to contain an exponential number of relations.
- Why not trying to solve the inequality

$$|u - kN^\gamma| \leq p_d$$

for $\gamma \geq 1$?
Schnorr’s approach to factorization

- We have a polyhedron where every element of the lattice inside this polyhedron yields a relation.
- Schnorr shows that for some choice of the parameters this polyhedron is expected to contain an exponential number of relations.
- Why not trying to solve the inequality
  \[ |u - kN^\gamma| \leq p_d \]
  for \( \gamma \geq 1? \)
- Adleman used an approach similar to that of Schnorr, but where the search was of short vectors rather than close vectors.
Adleman’s approach to factorization

The Prime Number Lattice of Adleman includes the previously called target vector:

\[
A_p = \begin{bmatrix}
\sqrt[p]{\ln p_1} & 0 & 0 & 0 \\
0 & \ddots & 0 & 0 \\
0 & 0 & \sqrt[p]{\ln p_d} & 0 \\
C \ln p_1 & \cdots & C \ln p_d & C \ln N
\end{bmatrix}.
\]
Adleman’s approach to factorization

The Prime Number Lattice of Adleman includes the previously called target vector:

\[ A_p = \begin{bmatrix} \sqrt[\varphi]{\ln p_1} & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & \sqrt[\varphi]{\ln p_d} & 0 \\ C \ln p_1 & \cdots & C \ln p_d & C \ln N \end{bmatrix}. \]

We have, for \( z \in \mathbb{Z}^{d+1} \),

\[ A_p z = \begin{bmatrix} z_1 \sqrt[\varphi]{\ln p_1} \\ \vdots \\ z_d \sqrt[\varphi]{\ln p_d} \\ C \left( \sum_{i=1}^{d} z_i \ln p_i + z_{d+1} \ln N \right) \end{bmatrix}. \]
Adleman’s approach to factorization

The norm is given by

\[ |A_p z|^p_p = \sum_{i=1}^{d} |z_i|^p \ln p_i + C^p \sum_{i=1}^{d} z_i \ln p_i + z_{d+1} \ln N|^p. \]
Adleman’s approach to factorization

The norm is given by

\[ \| A_p z \|_p^p = \sum_{i=1}^{d} |z_i|^p \ln p_i + C^p \sum_{i=1}^{d} z_i \ln p_i + z_{d+1} \ln N|_p. \]

**Theorem**

Let \( C \geq 3 \) and \( z \in \mathbb{Z}^{d+1} \), with \( \delta = |z_{d+1}| \) and \( z_{d+1} \leq 0 \). Then, if for \( \gamma \in \mathbb{N} \setminus \{0\} \)

\[ \| A_1 z \|_1 \leq 2 \ln C + 2 \ln p_d - (C(\gamma - \delta) + \delta) \cdot \ln N, \]

then

\[ |u - kN^\gamma| \leq p_d. \]
Outline

Introduction

Outline of the algorithm

The search for “relations”

Termination considerations

Conclusions
Are there enough relations?

- We gave a sufficient condition for a vector to give a relation.
Are there enough relations?

- We gave a sufficient condition for a vector to give a relation.
- Are there at least $d + 2$ of these vectors?

Schnorr shows, making some probabilistic independency and uniform distribution hypotheses, that for the choice of parameters $p_d = (\ln N)^{\alpha} C = N^c$ with $c > 1$ and $\alpha > (2c - 1)/(c - 1)$ constants, there are at least $N_{\epsilon + o(1)}$ solution vectors, $\epsilon \equiv \epsilon(\alpha, c) > 0$.

The analysis of the algorithm is not rigorous by now.
Are there enough relations?

▶ We gave a sufficient condition for a vector to give a relation.
▶ Are there at least \( d + 2 \) of these vectors?
▶ Schnorr shows, making some probabilistic independency and uniform distribution hypotheses, that for the choice of parameters

\[
p_d = (\ln N)^\alpha \quad C = N^c
\]

with \( c > 1 \) and \( \alpha > (2c - 1)/(c - 1) \) constants, there are at least \( N^{\epsilon + o(1)} \) solution vectors, \( \epsilon \equiv \epsilon(\alpha, c) > 0 \).
Are there enough relations?

- We gave a sufficient condition for a vector to give a relation.
- Are there at least \( d + 2 \) of these vectors?
- Schnorr shows, making some probabilistic independency and uniform distribution hypotheses, that for the choice of parameters
  \[ p_d = (\ln N)^\alpha \quad C = N^c \]
  with \( c > 1 \) and \( \alpha > (2c - 1)/(c - 1) \) constants, there are at least \( N^{\epsilon + o(1)} \) solution vectors, \( \epsilon \equiv \epsilon(\alpha, c) > 0 \).
- The analysis of the algorithm is not rigorous by now.
A lower bound for the number of relations

Two approaches by now:

- Geometrical: show that the 1-norm ball of the theorem has at least \( d + 2 \) points. Tools:
  - Minkowski's Theorem.
  - Gaussian heuristic and the Gram-Schmidt lengths.

- Number Theoretic: define a particular set of relations which can be detected by the algorithm and show it has at least \( d + 2 \) elements. Tools:
  - Analytic and Probabilistic Number Theory.
  - Independence hypotheses on arithmetic properties.
A lower bound for the number of relations

Two approaches by now:
- Geometrical: show that the 1-norm ball of the theorem has at least $d + 2$ points. Tools:
  - Minkowski’s Theorem.
A lower bound for the number of relations

Two approaches by now:

- Geometrical: show that the 1-norm ball of the theorem has at least \( d + 2 \) points. Tools:
  - Minkowski’s Theorem.
  - Gaussian heuristic and the Gram-Schmidt lengths.
A lower bound for the number of relations

Two approaches by now:

- Geometrical: show that the 1-norm ball of the theorem has at least \( d + 2 \) points. Tools:
  - Minkowski’s Theorem.
  - Gaussian heuristic and the Gram-Schmidt lengths.

- Number Theoretic: define a particular set of relations which can be detected by the algorithm and show it has at least \( d + 2 \) elements. Tools:
A lower bound for the number of relations

Two approaches by now:

- Geometrical: show that the 1-norm ball of the theorem has at least $d + 2$ points. Tools:
  - Minkowski’s Theorem.
  - Gaussian heuristic and the Gram-Schmidt lengths.

- Number Theoretic: define a particular set of relations which can be detected by the algorithm and show it has at least $d + 2$ elements. Tools:
  - Analytic and Probabilistic Number Theory.
A lower bound for the number of relations

Two approaches by now:

▶ Geometrical: show that the 1-norm ball of the theorem has at least $d + 2$ points. Tools:
  ▶ Minkowski’s Theorem.
  ▶ Gaussian heuristic and the Gram-Schmidt lengths.

▶ Number Theoretic: define a particular set of relations which can be detected by the algorithm and show it has at least $d + 2$ elements. Tools:
  ▶ Analytic and Probabilistic Number Theory.
  ▶ Independence hypotheses on arithmetic properties.
Outline

Introduction

Outline of the algorithm

The search for “relations”

Termination considerations

Conclusions
Open problem

Prove a lemma like the following, for the Euclidean norm:

Let $C \geq 3$. Then, for any integer vector $z$ such that $\|S_1 z - t\|_1 \leq \varepsilon$, we have $|u - kN| \leq \exp(\varepsilon^2) \cdot \sqrt{N C}$. 
Open problem

Prove a lemma like the following, for the Euclidean norm:

**Lemma**

*Let $C \geq 3$. Then, for any integer vector $z$ such that $||S_1z - t||_1 \leq \varepsilon$, we have*

$$|u - kN| \leq \exp \left( \frac{\varepsilon}{2} \right) \cdot \frac{\sqrt{N}}{C}.$$
Conclusions

- The algorithm works in practice.
Conclusions

- The algorithm works in practice.
- The CVP formulation (Schnorr’s) allows precomputations. Potentially useful for doing many factorizations in a row.
Conclusions

- The algorithm works in practice.
- The CVP formulation (Schnorr’s) allows precomputations. Potentially useful for doing many factorizations in a row.
- Enumeration versus sampling: we need to have good estimates on the number of relations, in order to choose.
Conclusions

- We provided an effective version of Schnorr’s criterion for detecting relations.
Conclusions

▶ We provided an effective version of Schnorr’s criterion for detecting relations.

▶ This provides a means of evaluating a lower bound for the number of relations (or at least an estimate). Minkowski’s theorem can provide a completely deterministic proof for the validity of the algorithm.
Conclusions

▶ We provided an effective version of Schnorr’s criterion for detecting relations.
▶ This provides a means of evaluating a lower bound for the number of relations (or at least an estimate). Minkowski’s theorem can provide a completely deterministic proof for the validity of the algorithm.
▶ To do: perform extensive experimentations, and maybe probabilistic modelling.
Bibliography

BMM De Weger.
Solving exponential diophantine equations using lattice basis reduction algorithms.

Daniele Micciancio and Shafi Goldwasser.

Claus Peter Schnorr.
Factoring integers and computing discrete logarithms via diophantine approximation.