Improvements to ideal class group and regulator computation in real quadratic number fields

Jean-François Biasse[1], Michael J. Jacobson Jr[2]
[1]École polytechnique,
[2]University of Calgary

ANTS IX

Motivations

Let $\mathbb{K}$ be a quadratic number field of discrimiant Δ and maximal order $\mathcal{O}_\Delta$. We are interested in

- Computing the group structure of $\mathrm{Cl}(\Delta) := \mathrm{Cl}(\mathcal{O}_\Delta)$.

Motivations

Let $\mathbb{K}$ be a quadratic number field of discrimiant Δ and maximal order $\mathcal{O}_\Delta$. We are interested in

- Computing the group structure of $\mathrm{Cl}(\Delta) := \mathrm{Cl}(\mathcal{O}_\Delta)$.
- Computing the regulator R_Δ of $\mathbb{K}$.

Motivations

Let $\mathbb{K}$ be a quadratic number field of discrimiant Δ and maximal order $\mathcal{O}_\Delta$. We are interested in

- Computing the group structure of $\mathrm{Cl}(\Delta) := \mathrm{Cl}(\mathcal{O}_\Delta)$.
- Computing the regulator R_Δ of $\mathbb{K}$.
- Computing a compact representation of the fundamental unit ε_Δ.

Motivations

Let $\mathbb{K}$ be a quadratic number field of discriminant Δ and maximal order $\mathcal{O}_\Delta$. We are interested in

- Computing the group structure of $\mathrm{Cl}(\Delta) := \mathrm{Cl}(\mathcal{O}_\Delta)$.
- Computing the regulator R_Δ of $\mathbb{K}$.
- Computing a compact representation of the fundamental unit ε_Δ.

We provide practical improvements to the classical subexponential algorithms.

Motivations

Let $\mathbb{K}$ be a quadratic number field of discriminant Δ and maximal order $\mathcal{O}_\Delta$. We are interested in

- Computing the group structure of $\mathrm{Cl}(\Delta) := \mathrm{Cl}(\mathcal{O}_\Delta)$.
- Computing the regulator R_Δ of $\mathbb{K}$.
- Computing a compact representation of the fundamental unit ε_Δ.

We provide practical improvements to the classical subexponential algorithms.

We achieve the computation of $\mathrm{Cl}(\Delta)$ and R_Δ for a 110-digit discriminant.

1. Introduction
2. Classical Algorithms
3. Practical improvements

1. Introduction
2. Classical Algorithms
3. Practical improvements

Ideals

- We work in $\mathbb{K}$ that satisfies $[\mathbb{K} : \mathbb{Q}] = 2$.

Ideals

- We work in $\mathbb{K}$ that satisfies $[\mathbb{K} : \mathbb{Q}] = 2$.
- Let $\mathcal{O}_\Delta$ be the *ring of integers* of $\mathbb{K}$ and Δ its discriminant.

Ideals

- We work in $\mathbb{K}$ that satisfies $[\mathbb{K} : \mathbb{Q}] = 2$.
- Let $\mathcal{O}_\Delta$ be the *ring of integers* of $\mathbb{K}$ and Δ its discriminant.
- If $\Delta < 0$: **imaginary** case. If $\Delta > 0$: **real case**.

Ideals

- We work in $\mathbb{K}$ that satisfies $[\mathbb{K} : \mathbb{Q}] = 2$.
- Let $\mathcal{O}_\Delta$ be the *ring of integers* of $\mathbb{K}$ and Δ its discriminant.
- If $\Delta < 0$: **imaginary** case. If $\Delta > 0$: **real case**.
- The *fractional ideals* $\mathfrak{a}$ are the sets of the form

$$\frac{1}{d}\mathfrak{a}', \mid \; d \in \mathbb{K}, \;\; \mathfrak{a}' \text{ is an ideal of } \mathcal{O}_\Delta.$$

Ideal class group

- Let $\mathcal{I}(\Delta)$ be the invertible fractional ideals and $\mathcal{P}$ the principal ideals, then

$$\mathrm{Cl}(\Delta) := \mathcal{I}(\Delta)/\mathcal{P}.$$

Ideal class group

- Let $\mathcal{I}(\Delta)$ be the invertible fractional ideals and $\mathcal{P}$ the principal ideals, then

$$\mathrm{Cl}(\Delta) := \mathcal{I}(\Delta)/\mathcal{P}.$$

- $\mathrm{Cl}(\Delta)$ is finite of cardinality $h(\Delta)$.

Ideal class group

- Let $\mathcal{I}(\Delta)$ be the invertible fractional ideals and $\mathcal{P}$ the principal ideals, then

$$\mathrm{Cl}(\Delta) := \mathcal{I}(\Delta)/\mathcal{P}.$$

- $\mathrm{Cl}(\Delta)$ is finite of cardinality $h(\Delta)$.
- $h(\Delta)$ is essentially as “hard” to compute as $\mathrm{Cl}(\Delta)$.

Ideal class group

- Let $\mathcal{I}(\Delta)$ be the invertible fractional ideals and $\mathcal{P}$ the principal ideals, then

$$\mathrm{Cl}(\Delta) := \mathcal{I}(\Delta)/\mathcal{P}.$$

- $\mathrm{Cl}(\Delta)$ is finite of cardinality $h(\Delta)$.
- $h(\Delta)$ is essentially as "hard" to compute as $\mathrm{Cl}(\Delta)$.

Let $\mathfrak{a}, \mathfrak{b} \in \mathcal{I}(\Delta)$, then we denote by $\mathfrak{a} \sim \mathfrak{b}$:

$$[\mathfrak{a}] = [\mathfrak{b}] \in \mathrm{Cl}(\Delta) \Leftrightarrow \exists \alpha \in \mathbb{K},\ \mathfrak{a} = (\alpha)\mathfrak{b}.$$

Regulator

We assume that $\Delta > 0$.

Regulator

We assume that $\Delta > 0$.

- Elements of $\mathbb{K}$ such that $\mathcal{N}(x) = \pm 1$ are *units*.

Regulator

We assume that $\Delta > 0$.

- Elements of $\mathbb{K}$ such that $\mathcal{N}(x) = \pm 1$ are *units*.
- Every unit ε can be written as $\varepsilon = \pm\varepsilon_\Delta^n$, where ε_Δ is the *fundamental unit* of $\mathbb{K}$.

Regulator

We assume that $\Delta > 0$.

- Elements of $\mathbb{K}$ such that $\mathcal{N}(x) = \pm 1$ are *units*.
- Every unit ε can be written as $\varepsilon = \pm\varepsilon_\Delta^n$, where ε_Δ is the *fundamental unit* of $\mathbb{K}$.

The *regulator* of $\mathbb{K}$ is

$$R_\Delta = \log \varepsilon_\Delta .$$

Regulator

We assume that $\Delta > 0$.

- Elements of $\mathbb{K}$ such that $\mathcal{N}(x) = \pm 1$ are *units*.
- Every unit ε can be written as $\varepsilon = \pm\varepsilon_\Delta^n$, where ε_Δ is the *fundamental unit* of $\mathbb{K}$.

The *regulator* of $\mathbb{K}$ is

$$R_\Delta = \log \varepsilon_\Delta.$$

Every unit ε satisfies $\exists n, \log|\varepsilon| = nR_\Delta$.

1. Introduction
2. Classical Algorithms
3. Practical improvements

General strategy

The algorithms for solving our problems follow the same pattern.
Let $\mathcal{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}$ be a generating set of $\mathrm{Cl}(\Delta)$.

General strategy

The algorithms for solving our problems follow the same pattern.
Let $\mathcal{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}$ be a generating set of $\mathrm{Cl}(\Delta)$.

1. Find relations of the form

$$(\alpha) = \mathfrak{p}_1^{e_1} \cdots \mathfrak{p}_N^{e_N},$$

that is $\prod_i [\mathfrak{p}_i]^{e_i} = [1]$

General strategy

The algorithms for solving our problems follow the same pattern.
Let $\mathcal{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}$ be a generating set of $\mathrm{Cl}(\Delta)$.

1. Find relations of the form

$$(\alpha) = \mathfrak{p}_1^{e_1} \cdots \mathfrak{p}_N^{e_N},$$

   that is $\prod_i [\mathfrak{p}_i]^{e_i} = [1]$

2. Every time a relation is found, $[e_1, \ldots, e_N]$ is added as a row of the *relation matrix* M

General strategy

The algorithms for solving our problems follow the same pattern. Let $\mathcal{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}$ be a generating set of $\mathrm{Cl}(\Delta)$.

1. Find relations of the form

$$(\alpha) = \mathfrak{p}_1^{e_1} \cdots \mathfrak{p}_N^{e_N},$$

   that is $\prod_i [\mathfrak{p}_i]^{e_i} = [1]$
2. Every time a relation is found, $[e_1, \ldots, e_N]$ is added as a row of the *relation matrix* M
3. Perform a *linear algebra phase* on M.

Complexity

We define the *subexponential* function by

$$L_\Delta(\alpha, \beta) = e^{\beta \log |\Delta|^{\alpha} \log\log |\Delta|^{1-\alpha}}.$$

Complexity

We define the *subexponential* function by

$$L_\Delta(\alpha,\beta) = e^{\beta \log|\Delta|^{\alpha} \log\log|\Delta|^{1-\alpha}}.$$

For $\alpha \in [0,1]$, $L_\Delta(\alpha,\beta)$ is between exponential and polynomial in $\log|\Delta|$ since

$$L_\Delta(0,\beta) = \log|\Delta|^{\beta},$$
$$L_\Delta(1,\beta) = |\Delta|^{\beta}.$$

Complexity

We define the *subexponential* function by

$$L_\Delta(\alpha, \beta) = e^{\beta \log|\Delta|^\alpha \log\log|\Delta|^{1-\alpha}}.$$

For $\alpha \in [0,1]$, $L_\Delta(\alpha, \beta)$ is between exponential and polynomial in $\log|\Delta|$ since

$$L_\Delta(0, \beta) = \log|\Delta|^\beta,$$
$$L_\Delta(1, \beta) = |\Delta|^\beta.$$

Our problems for qudratic number fields have complexity

$$L_\Delta(1/2, c),$$

where c depends on the linear algebra phase.

The factor base

We fill the factor base with invertible **prime ideals** $\mathfrak{p}$. There is p prime such that

$$\mathfrak{p} \cap \mathbb{Z} = (p) \text{ and } \mathcal{N}(\mathfrak{p}) = p.$$

The factor base

We fill the factor base with invertible **prime ideals** $\mathfrak{p}$. There is p prime such that

$$\mathfrak{p} \cap \mathbb{Z} = (p) \text{ and } \mathcal{N}(\mathfrak{p}) = p.$$

Let B a bound, we define

$$\mathcal{B} := \{\mathfrak{p} \text{ invertible prime} \mid \mathcal{N}(\mathfrak{p}) \leq B\} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}.$$

The factor base

We fill the factor base with invertible **prime ideals** $\mathfrak{p}$. There is p prime such that

$$\mathfrak{p} \cap \mathbb{Z} = (p) \text{ and } \mathcal{N}(\mathfrak{p}) = p.$$

Let B a bound, we define

$$\mathcal{B} := \{\mathfrak{p} \text{ invertible prime} \mid \mathcal{N}(\mathfrak{p}) \leq B\} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}.$$

Under ERH, if $B > 6 \log^2 |\Delta|$, then $\mathcal{B}$ generates $\mathrm{Cl}(\Delta)$, and the lattice $\mathcal{L}$ of the relations satisfies

$$\mathrm{Cl}(\Delta) \simeq \mathbb{Z}^N / \mathcal{L}.$$

The factor base

We fill the factor base with invertible **prime ideals** $\mathfrak{p}$. There is p prime such that

$$\mathfrak{p} \cap \mathbb{Z} = (p) \text{ and } \mathcal{N}(\mathfrak{p}) = p.$$

Let B a bound, we define

$$\mathcal{B} := \{\mathfrak{p} \text{ invertible prime} \mid \mathcal{N}(\mathfrak{p}) \leq B\} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}.$$

Under ERH, if $B > 6\log^2|\Delta|$, then $\mathcal{B}$ generates $\mathrm{Cl}(\Delta)$, and the lattice $\mathcal{L}$ of the relations satisfies

$$\mathrm{Cl}(\Delta) \simeq \mathbb{Z}^N/\mathcal{L}.$$

We have ($\mathfrak{a}$ is $\mathcal{B}$-smooth)$\Leftrightarrow$($\mathcal{N}(\mathfrak{a})$ is B-smooth).

Hermite Normal Form

Invertible operations on rows lead to the **Hermite Normal Form** H of M :

$$H = \left(\begin{array}{ccc|cc} h_{1,1} & \dots & 0 & & \\ \vdots & \ddots & \vdots & \multicolumn{2}{c}{(0)} \\ * & \dots & h_{l,l} & & \\ \hline & & & 1 & (0) \\ & (*) & & & \ddots \\ & & & (0) & 1 \\ \hline \multicolumn{5}{c}{(0)} \end{array}\right),$$

where $\forall i > j : 0 \leq h_{ij} < h_{jj}$.

Hermite Normal Form

Invertible operations on rows lead to the **Hermite Normal Form** H of M :

$$H = \left(\begin{array}{ccc:cc} h_{1,1} & \dots & 0 & & \\ \vdots & \ddots & \vdots & \multicolumn{2}{c}{(0)} \\ * & \dots & h_{l,l} & & \\ \hdashline & & & 1 & (0) \\ & (*) & & & \ddots \\ & & & (0) & 1 \\ \hdashline \multicolumn{5}{c}{(0)} \end{array}\right),$$

where $\forall i > j : 0 \leq h_{ij} < h_{jj}$.

Upper left : **Essential part**

Smith Normal Form (SNF) and class group structure

Any matrix $A \in \mathbb{Z}^{n\times n}$ with non zero determinant can be written as :

$$A = V^{-1} \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & d_n \end{pmatrix} U^{-1}$$

where $\forall i$ such that $1 \leq i < n$: $d_{i+1} | d_i$.

Smith Normal Form (SNF) and class group structure

Any matrix $A \in \mathbb{Z}^{n\times n}$ with non zero determinant can be written as :

$$A = V^{-1} \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & d_n \end{pmatrix} U^{-1}$$

where $\forall i$ such that $1 \leq i < n$: $d_{i+1} | d_i$.

If (d_i) are the diagonal coefficients of the SNF of the essential part of H then

$$Cl(\Delta) = \bigoplus_{1\leq i\leq n} (\mathbb{Z}/d_i\mathbb{Z})$$

Computing the HNF in practice

Several implementation of HNF algorithm exist : Pari, Kash, Sage, Magma, NTL ... We used an NTL/Linbox-based strategy.

Computing the HNF in practice

Several implementation of HNF algorithm exist : Pari, Kash, Sage, Magma, NTL ... We used an NTL/Linbox-based strategy.

Let $M \in \mathbb{Z}^{N' \times N}$ be the relation matrix.

1. Extract two random $N \times N$ full-rank submatrices M_1 and M_2 of M.

Computing the HNF in practice

Several implementation of HNF algorithm exist : Pari, Kash, Sage, Magma, NTL ... We used an NTL/Linbox-based strategy.

Let $M \in \mathbb{Z}^{N' \times N}$ be the relation matrix.

1. Extract two random $N \times N$ full-rank submatrices M_1 and M_2 of M.
2. Compute $h_1 \leftarrow \det(M_1)$ and $h_2 \leftarrow \det(M_2)$ with function `det` of linbox.

Computing the HNF in practice

Several implementation of HNF algorithm exist : Pari, Kash, Sage, Magma, NTL ... We used an NTL/Linbox-based strategy.

Let $M \in \mathbb{Z}^{N' \times N}$ be the relation matrix.

1. Extract two random $N \times N$ full-rank submatrices M_1 and M_2 of M.
2. Compute $h_1 \leftarrow \det(M_1)$ and $h_2 \leftarrow \det(M_2)$ with function `det` of linbox.
3. Let $h := \gcd(h_1, h_2)$. It is a multiple of h_Δ.

Computing the HNF in practice

Several implementation of HNF algorithm exist : Pari, Kash, Sage, Magma, NTL ... We used an NTL/Linbox-based strategy.

Let $M \in \mathbb{Z}^{N' \times N}$ be the relation matrix.

1. Extract two random $N \times N$ full-rank submatrices M_1 and M_2 of M.
2. Compute $h_1 \leftarrow \det(M_1)$ and $h_2 \leftarrow \det(M_2)$ with function `det` of linbox.
3. Let $h := \gcd(h_1, h_2)$. It is a multiple of h_Δ.
4. Call the implementation of DomKanTro87 modular HNF algorithm with (M, h).

Computing the HNF in practice

Several implementation of HNF algorithm exist : Pari, Kash, Sage, Magma, NTL ... We used an NTL/Linbox-based strategy.

Let $M \in \mathbb{Z}^{N' \times N}$ be the relation matrix.

1. Extract two random $N \times N$ full-rank submatrices M_1 and M_2 of M.
2. Compute $h_1 \leftarrow \det(M_1)$ and $h_2 \leftarrow \det(M_2)$ with function `det` of linbox.
3. Let $h := \gcd(h_1, h_2)$. It is a multiple of h_Δ.
4. Call the implementation of DomKanTro87 modular HNF algorithm with (M, h).

In fact MAGMA is much faster :($\Rightarrow$ room for improvement.

Regulator computation

Let $M = (m_{ij}) \in \mathbb{Z}^{N' \times N}$, be the relation matrix, $\mathcal{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}$, and

$$(\alpha_i) = \mathfrak{p}_1^{m_{i1}} \cdots \mathfrak{p}_N^{m_{N1}}.$$

Regulator computation

Let $M = (m_{ij}) \in \mathbb{Z}^{N' \times N}$, be the relation matrix, $\mathcal{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}$, and

$$(\alpha_i) = \mathfrak{p}_1^{m_{i1}} \cdots \mathfrak{p}_N^{m_{N1}}.$$

Let $X = (x_i)$, $i \leq N'$ be a kernel vector of M. Then

$$\gamma := \alpha_1^{x_1} \cdots \alpha_{N'}^{x_{N'}}$$

is a unit since $(\gamma) = \prod_i \alpha_i^{x_i} = (1)$.

Regulator computation

Let $M = (m_{ij}) \in \mathbb{Z}^{N' \times N}$, be the relation matrix, $\mathcal{B} = \{\mathfrak{p}_1, \dots, \mathfrak{p}_N\}$, and

$$(\alpha_i) = \mathfrak{p}_1^{m_{i1}} \cdots \mathfrak{p}_N^{m_{N1}}.$$

Let $X = (x_i)$, $i \leq N'$ be a kernel vector of M. Then

$$\gamma := \alpha_1^{x_1} \cdots \alpha_{N'}^{x_{N'}}$$

is a unit since $(\gamma) = \prod_i \alpha_i^{x_i} = (1)$.

There is n such that

$$\log |\gamma| = nR_\Delta$$

Regulator computation

Let $M = (m_{ij}) \in \mathbb{Z}^{N' \times N}$, be the relation matrix, $\mathcal{B} = \{\mathfrak{p}_1, \ldots, \mathfrak{p}_N\}$, and

$$(\alpha_i) = \mathfrak{p}_1^{m_{i1}} \cdots \mathfrak{p}_N^{m_{N1}}.$$

Let $X = (x_i)$, $i \leq N'$ be a kernel vector of M. Then

$$\gamma := \alpha_1^{x_1} \cdots \alpha_{N'}^{x_{N'}}$$

is a unit since $(\gamma) = \prod_i \alpha_i^{x_i} = (1)$.

There is n such that

$$\log |\gamma| = nR_\Delta$$

Each kernel vector of M yields a multiple of R_Δ. We recover R_Δ by successive real-GCD computation.

Relation collection via sieving

Let $\mathfrak{a}$ be an ideal. There is $\mathfrak{a}' \sim \mathfrak{a}$ of the form $\mathfrak{a}' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$.

Relation collection via sieving

Let $\mathfrak{a}$ be an ideal. There is $\mathfrak{a}' \sim \mathfrak{a}$ of the form $\mathfrak{a}' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$. Then for each x, y we have

$$\gamma := ax + y\left(\frac{b+\sqrt{\Delta}}{2}\right) \in \mathfrak{a}'.$$

Relation collection via sieving

Let $\mathfrak{a}$ be an ideal. There is $\mathfrak{a}' \sim \mathfrak{a}$ of the form $\mathfrak{a}' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$. Then for each x, y we have

$$\gamma := ax + y\left(\frac{b+\sqrt{\Delta}}{2}\right) \in \mathfrak{a}'.$$

(JacWil09) There is an ideal $\mathfrak{b}$ such that $(\gamma) = \mathfrak{a}'\mathfrak{b}$ (that is $\mathfrak{a} \cdot \mathfrak{b} \sim 1$) and

$$\mathcal{N}(\mathfrak{b}) = ax^2 + bxy + cy^2.$$

Relation collection via sieving

Let $\mathfrak{a}$ be an ideal. There is $\mathfrak{a}' \sim \mathfrak{a}$ of the form $\mathfrak{a}' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$. Then for each x, y we have

$$\gamma := ax + y\left(\frac{b+\sqrt{\Delta}}{2}\right) \in \mathfrak{a}'.$$

(JacWil09) There is an ideal $\mathfrak{b}$ such that $(\gamma) = \mathfrak{a}'\mathfrak{b}$ (that is $\mathfrak{a} \cdot \mathfrak{b} \sim 1$) and

$$\mathcal{N}(\mathfrak{b}) = ax^2 + bxy + cy^2.$$

1. Start with $\mathfrak{a} := \prod_i \mathfrak{p}_i^{e_i}$ which is $\mathcal{B}$-smooth.

Relation collection via sieving

Let $\mathfrak{a}$ be an ideal. There is $\mathfrak{a}' \sim \mathfrak{a}$ of the form $\mathfrak{a}' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$. Then for each x, y we have

$$\gamma := ax + y\left(\frac{b+\sqrt{\Delta}}{2}\right) \in \mathfrak{a}'.$$

(JacWil09) There is an ideal $\mathfrak{b}$ such that $(\gamma) = \mathfrak{a}'\mathfrak{b}$ (that is $\mathfrak{a} \cdot \mathfrak{b} \sim 1$) and

$$\mathcal{N}(\mathfrak{b}) = ax^2 + bxy + cy^2.$$

1. Start with $\mathfrak{a} := \prod_i \mathfrak{p}_i^{e_i}$ which is $\mathcal{B}$-smooth.
2. Find x, y such that $\phi_{\mathfrak{a}}(x, y) := ax^2 + bxy + cy^2$ is B-smooth

Relation collection via sieving

Let $\mathfrak{a}$ be an ideal. There is $\mathfrak{a}' \sim \mathfrak{a}$ of the form $\mathfrak{a}' = a\mathbb{Z} + \frac{(b+\sqrt{\Delta})}{2}\mathbb{Z}$. Then for each x, y we have

$$\gamma := ax + y\left(\frac{b+\sqrt{\Delta}}{2}\right) \in \mathfrak{a}'.$$

(JacWil09) There is an ideal $\mathfrak{b}$ such that $(\gamma) = \mathfrak{a}'\mathfrak{b}$ (that is $\mathfrak{a} \cdot \mathfrak{b} \sim 1$) and

$$\mathcal{N}(\mathfrak{b}) = ax^2 + bxy + cy^2.$$

1. Start with $\mathfrak{a} := \prod_i \mathfrak{p}_i^{e_i}$ which is $\mathcal{B}$-smooth.
2. Find x, y such that $\phi_{\mathfrak{a}}(x,y) := ax^2 + bxy + cy^2$ is B-smooth
3. Deduce $\mathcal{B}$-smooth ideal $\mathfrak{b}$ such that $\mathfrak{a} \cdot \mathfrak{b} \sim 1$

The quadratic sieve

Let $\phi_{\mathfrak{a}}(X, Y) = aX^2 + bXY + cY^2$ and B defining $\mathcal{B}$. We look for B-smooth values of $\phi_{\mathfrak{a}}(X, Y)$. (Jac99) : use the **quadratic sieve**

The quadratic sieve

Let $\phi_{\mathfrak{a}}(X, Y) = aX^2 + bXY + cY^2$ and B defining $\mathcal{B}$. We look for B-smooth values of $\phi_{\mathfrak{a}}(X, Y)$. (Jac99) : use the **quadratic sieve**

We look for $x \in [-M, M]$ such that $\phi_{\mathfrak{a}}(x, 1)$ is B-smooth. We do not want to test them all.

The quadratic sieve

Let $\phi_{\mathfrak{a}}(X, Y) = aX^2 + bXY + cY^2$ and B defining $\mathcal{B}$. We look for B-smooth values of $\phi_{\mathfrak{a}}(X, Y)$. (Jac99) : use the **quadratic sieve**

We look for $x \in [-M, M]$ such that $\phi_{\mathfrak{a}}(x, 1)$ is B-smooth. We do not want to test them all.

1. We compute the roots r_p of $\phi_{\mathfrak{a}}(X, 1) \mod p$ for $p \leq B$.

The quadratic sieve

Let $\phi_{\mathfrak{a}}(X, Y) = aX^2 + bXY + cY^2$ and B defining $\mathcal{B}$. We look for B-smooth values of $\phi_{\mathfrak{a}}(X, Y)$. (Jac99) : use the **quadratic sieve**

We look for $x \in [-M, M]$ such that $\phi_{\mathfrak{a}}(x, 1)$ is B-smooth. We do not want to test them all.

1. We compute the roots r_p of $\phi_{\mathfrak{a}}(X, 1) \mod p$ for $p \leq B$.
2. We initialize S of length $2M + 1$ to 0.

The quadratic sieve

Let $\phi_{\mathfrak{a}}(X, Y) = aX^2 + bXY + cY^2$ and B defining $\mathcal{B}$. We look for B-smooth values of $\phi_{\mathfrak{a}}(X, Y)$. (Jac99) : use the **quadratic sieve**

We look for $x \in [-M, M]$ such that $\phi_{\mathfrak{a}}(x, 1)$ is B-smooth. We do not want to test them all.

1. We compute the roots r_p of $\phi_{\mathfrak{a}}(X, 1) \mod p$ for $p \leq B$.
2. We initialize S of length $2M + 1$ to 0.
3. For $x = r_p + kp \in [-M, M]$ do $S[x] \leftarrow S[x] + \log p$ because

$$\phi_{\mathfrak{a}}(x, 1) = \phi_{\mathfrak{a}}(r_p + kp, 1) \equiv \phi_{\mathfrak{a}}(r_p, 1) \equiv 0 \mod p.$$

The quadratic sieve

Let $\phi_{\mathfrak{a}}(X,Y) = aX^2 + bXY + cY^2$ and B defining $\mathcal{B}$. We look for B-smooth values of $\phi_{\mathfrak{a}}(X,Y)$. (Jac99) : use the **quadratic sieve**

We look for $x \in [-M, M]$ such that $\phi_{\mathfrak{a}}(x,1)$ is B-smooth. We do not want to test them all.

1. We compute the roots r_p of $\phi_{\mathfrak{a}}(X,1) \mod p$ for $p \leq B$.
2. We initialize S of length $2M+1$ to 0.
3. For $x = r_p + kp \in [-M, M]$ do $S[x] \leftarrow S[x] + \log p$ because

$$\phi_{\mathfrak{a}}(x,1) = \phi_{\mathfrak{a}}(r_p + kp, 1) \equiv \phi_{\mathfrak{a}}(r_p, 1) \equiv 0 \mod p.$$

4. For "large" $S[x]$, test the smoothness of $\phi_{\mathfrak{a}}(x,1)$.

Large prime variants

We speed-up the relation collection phase by considering $\mathfrak{p}$ such that $B \leq \mathcal{N}(\mathfrak{p}) \leq B_2$.

Large prime variants

We speed-up the relation collection phase by considering $\mathfrak{p}$ such that $B \leq \mathcal{N}(\mathfrak{p}) \leq B_2$.

- **Single large prime variant**. We authorize relations of the form

$$\mathfrak{a} = \underbrace{\mathfrak{p}_1 \dots \mathfrak{p}_n}_{\in \mathcal{B}} \mathfrak{p},$$

where $B \leq \mathcal{N}(\mathfrak{p}) \leq B_2$.

Large prime variants

We speed-up the relation collection phase by considering $\mathfrak{p}$ such that $B \leq \mathcal{N}(\mathfrak{p}) \leq B_2$.

- **Single large prime variant**. We authorize relations of the form
$$\mathfrak{a} = \underbrace{\mathfrak{p}_1 \dots \mathfrak{p}_n}_{\in \mathcal{B}} \mathfrak{p},$$
where $B \leq \mathcal{N}(\mathfrak{p}) \leq B_2$.
- **Double large prime variant** . We authorise relations of the form
$$\mathfrak{a} = \underbrace{\mathfrak{p}_1 \dots \mathfrak{p}_n}_{\in \mathcal{B}} \mathfrak{p}\mathfrak{p}',$$
where $B \leq \mathcal{N}(\mathfrak{p}), \mathcal{N}(\mathfrak{p}') \leq B_2$.

Batch smoothness test

Quadratic sieve : for large $S[x]$, we test the smoothness of $\phi_{\mathfrak{a}}(x, 1)$.

Batch smoothness test

Quadratic sieve : for large $S[x]$, we test the smoothness of $\phi_{\mathfrak{a}}(x, 1)$.

This can be done by trial division.

Batch smoothness test

Quadratic sieve : for large $S[x]$, we test the smoothness of $\phi_{\mathfrak{a}}(x,1)$.

This can be done by trial division.

We used an algorithm due to Berstein.

Batch smoothness test

Quadratic sieve : for large $S[x]$, we test the smoothness of $\phi_{\mathfrak{a}}(x, 1)$.

This can be done by trial division.

We used an algorithm due to Berstein.

- Takes non negative $x_1, \dots, x_K$ and primes $p_1, \dots, p_N$.

Batch smoothness test

Quadratic sieve : for large $S[x]$, we test the smoothness of $\phi_{\mathfrak{a}}(x, 1)$.

This can be done by trial division.

We used an algorithm due to Berstein.

- Takes non negative $x_1, \ldots, x_K$ and primes $p_1, \ldots, p_N$.
- returns the $\{p_1, \ldots, p_N\}$-smooth part of each x_i

Batch smoothness test

Quadratic sieve : for large $S[x]$, we test the smoothness of $\phi_{\mathfrak{a}}(x,1)$.

This can be done by trial division.

We used an algorithm due to Berstein.

- Takes non negative $x_1, \dots, x_K$ and primes $p_1, \dots, p_N$.
- returns the $\{p_1, \dots, p_N\}$-smooth part of each x_i
- Test is simultanous

Batch smoothness test

Quadratic sieve : for large $S[x]$, we test the smoothness of $\phi_{\mathfrak{a}}(x, 1)$.

This can be done by trial division.

We used an algorithm due to Berstein.

- Takes non negative $x_1, \ldots, x_K$ and primes $p_1, \ldots, p_N$.
- returns the $\{p_1, \ldots, p_N\}$-smooth part of each x_i
- Test is simultanous
- uses a tree structure.

Relation collection timings

TAB.: Comparison of the relation collection time for $\Delta = -4(10^n + 1)$

n	0LP	1LP	2LP	2LP Batch
40	0.69	0.56	0.59	0.66
45	7.25	3.77	3.83	4.41
50	18.82	9.30	9.84	6.82
55	152.28	74.78	55.99	36.49
60	333.26	166.88	140.79	83.06
65	2033.97	853.27	478.57	368.31
70	2828.92	1277.94	822.39	670.63
75	14811.70	6033.89	3324.61	2732.68

Eliminating columns

Sparse large matrix. Especially with the large primes.

Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

We can use the standard Gaussian elimination. It consists of pivoting with an arbitrary row.

Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

We can use the standard Gaussian elimination. It consists of pivoting with an arbitrary row.

Two problems encontered :

Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

We can use the standard Gaussian elimination. It consists of pivoting with an arbitrary row.

Two problems encontered :

1. R_3 can have Hamming weight $w(R_3) = w(R_1) + w(R_2)$.

Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

We can use the standard Gaussian elimination. It consists of pivoting with an arbitrary row.

Two problems encontered :

1. R_3 can have Hamming weight $w(R_3) = w(R_1) + w(R_2)$.
2. The coefficients might grow dramatically.

Eliminating columns

Sparse large matrix. Especially with the large primes.

We want to eliminate columns to reduce its dimension and apply algorithms for dense matrices.

We can use the standard Gaussian elimination. It consists of pivoting with an arbitrary row.

Two problems encontered :

1. R_3 can have Hamming weight $w(R_3) = w(R_1) + w(R_2)$.
2. The coefficients might grow dramatically.

We describe a method for managing the growth of the density and the size of the coefficients during the elimination.

Structured Elimination

Row $R \longrightarrow$ cost function $COST(R)$ taking into account :

Structured Elimination

Row $R \longrightarrow$ cost function $COST(R)$ taking into account :

1. Hamming weight of R

Structured Elimination

Row $R \longrightarrow$ cost function $COST(R)$ taking into account :

1. Hamming weight of R
2. Size of its coefficients

Structured Elimination

Row $R \longrightarrow$ cost function $COST(R)$ taking into account :

1. Hamming weight of R
2. Size of its coefficients

For a given column involving rows $R_1, ..., R_k$ we construct the complete graph $\mathcal{G}$:

Structured Elimination

Row $R \longrightarrow$ cost function $COST(R)$ taking into account :

1. Hamming weight of R
2. Size of its coefficients

For a given column involving rows $R_1, ..., R_k$ we construct the complete graph $\mathcal{G}$:

1. vertices R_i

Structured Elimination

Row $R \longrightarrow$ cost function $COST(R)$ taking into account :

1. Hamming weight of R
2. Size of its coefficients

For a given column involving rows $R_1, ..., R_k$ we construct the complete graph $\mathcal{G}$:

1. vertices R_i
2. edges labeled with the cost of the recombination $C_{ij} = COST(RECOMB(R_i, R_j))$

Structured Elimination

Row $R \longrightarrow$ cost function $COST(R)$ taking into account :

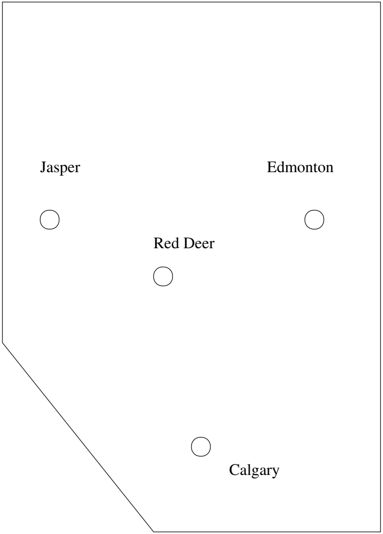
1. Hamming weight of R
2. Size of its coefficients

For a given column involving rows $R_1, ..., R_k$ we construct the complete graph $\mathcal{G}$:

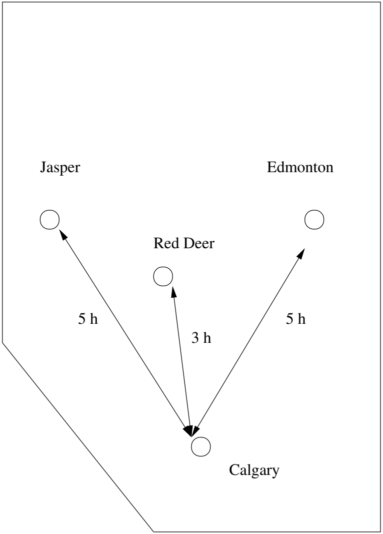
1. vertices R_i
2. edges labeled with the cost of the recombination $C_{ij} = COST(RECOMB(R_i, R_j))$

We then construct the minimum spanning tree of $\mathcal{G}$ and eliminate rows from the leaves to the root.

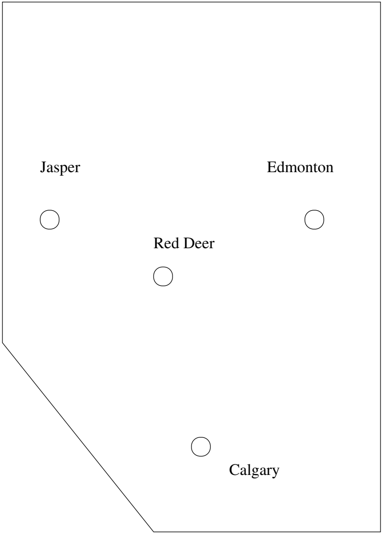
Minimum spanning tree on Alberta's map



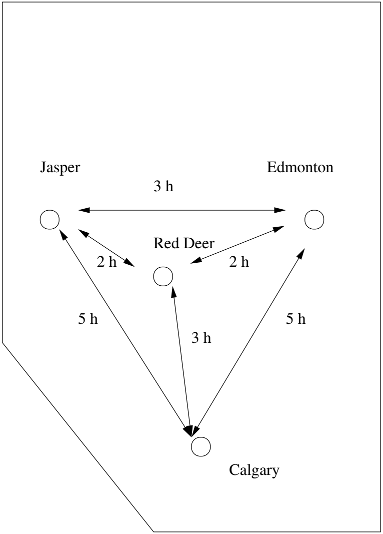
Minimum spanning tree on Alberta's map



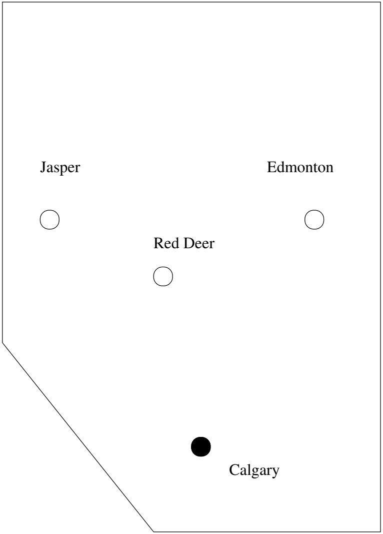
Minimum spanning tree on Alberta's map



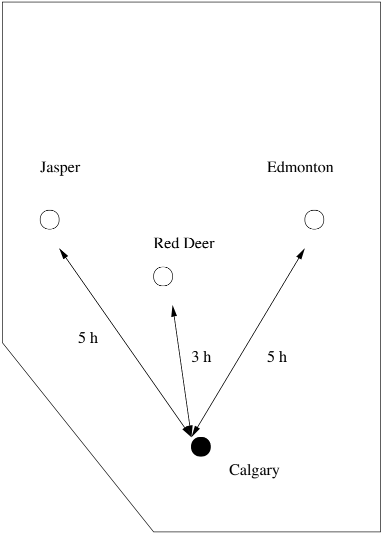
Minimum spanning tree on Alberta's map



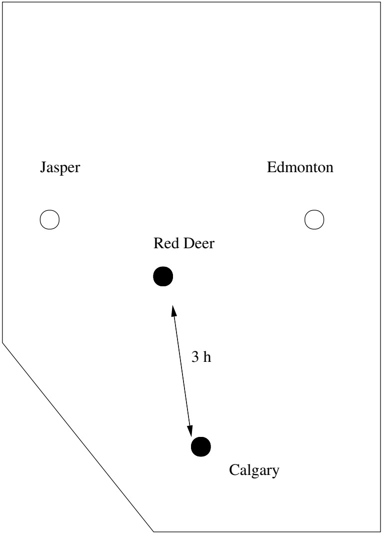
Minimum spanning tree on Alberta's map



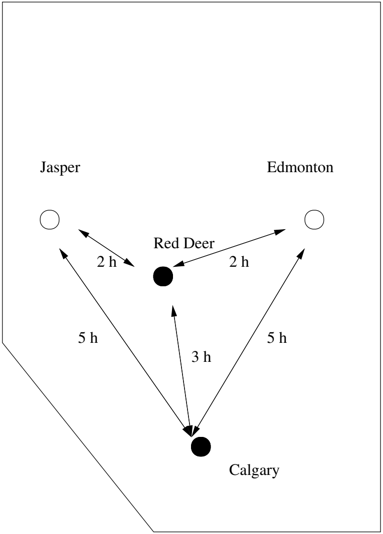
Minimum spanning tree on Alberta's map



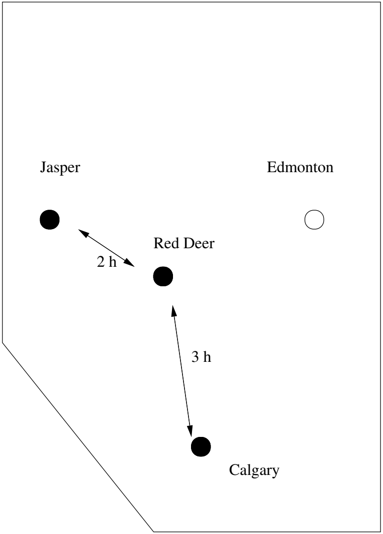
Minimum spanning tree on Alberta's map



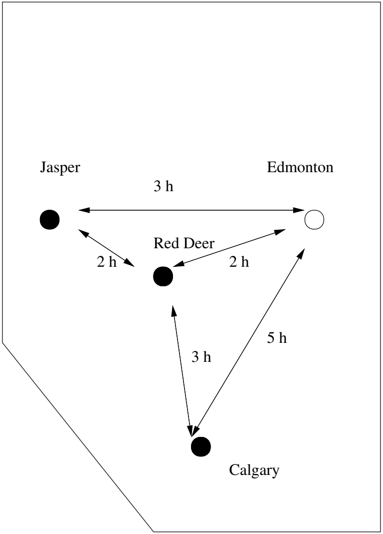
Minimum spanning tree on Alberta's map



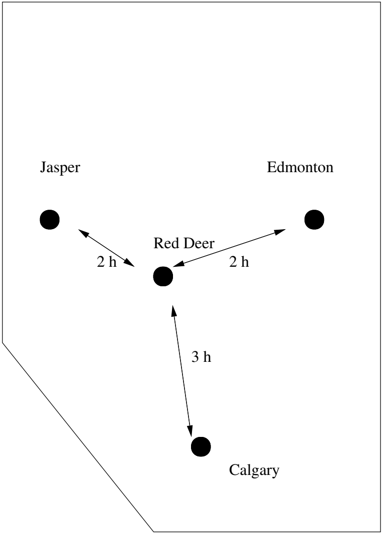
Minimum spanning tree on Alberta's map



Minimum spanning tree on Alberta's map



Minimum spanning tree on Alberta's map



Timings Gaussian elimination for $\Delta = 4(10^{60} + 3)$

Naive Gauss			Dedicated strategy		
i	Col Nb	HNF time	i	Col Nb	HNF time
5	1067	357.9	5	1078	368.0
10	799	184.8	10	806	187.2
50	596	93.7	50	580	84.3
125	542	73.8	125	515	63.4
160	533	72.0	160	497	56.9
170	532	222.4	170	493	192.6

Regulator computation

We want to avoid kernel computation and use fewer vectors. Idea due to Vollmer

Regulator computation

We want to avoid kernel computation and use fewer vectors. Idea due to Vollmer

1. We find k extra relations $\vec{r}_i$.

Regulator computation

We want to avoid kernel computation and use fewer vectors. Idea due to Vollmer

1. We find k extra relations $\vec{r_i}$.
2. We solve the k linear systems $\vec{x_i}M = \vec{r_i}$

Regulator computation

We want to avoid kernel computation and use fewer vectors. Idea due to Vollmer

1. We find k extra relations $\vec{r_i}$.
2. We solve the k linear systems $\vec{x_i} M = \vec{r_i}$
3. We augment the matrix M with the k extra rows

$$M' := \begin{pmatrix} M \\ \cdots\cdots\cdots \\ \vec{r_i} \end{pmatrix} \quad \vec{x_i}' := \left(\vec{x_i} \,\vdots\, 0 \ldots 0 \ -1 \ 0 \ldots 0 \right).$$

Regulator computation

We want to avoid kernel computation and use fewer vectors. Idea due to Vollmer

1. We find k extra relations $\vec{r_i}$.
2. We solve the k linear systems $\vec{x_i} M = \vec{r_i}$
3. We augment the matrix M with the k extra rows

$$M' := \begin{pmatrix} M \\ \hdashline \vec{r_i} \end{pmatrix} \quad \vec{x_i}' := \left(\vec{x_i} \; \vdots \; 0 \ldots 0 \; -1 \; 0 \ldots 0 \right).$$

The $\vec{x_i}'$ are kernel vectors of the new relation matrix M'.

Timings regulator computation

Kernel computation in $O(L_\Delta(1/2, \sqrt{2}))$.

Timings regulator computation

Kernel computation in $O(L_\Delta(1/2, \sqrt{2}))$.
System solving in $O(L_\Delta(1/2, 3/\sqrt{8}))$.

Timings regulator computation

Kernel computation in $O(L_\Delta(1/2, \sqrt{2}))$.
System solving in $O(L_\Delta(1/2, 3/\sqrt{8}))$.

Tab.: Regulator computation time for $\Delta = 4(10^n + 3)$

n	kernel computation	system solving
40	9.7	3.4
45	17.6	6.1
50	39.9	18.2
55	126.7	53.0
60	424.1	140.0
65	514.8	320.2
70	2728.5	791.1
75	8587.8	1775.8

Overall time comparison

Discriminants of the form $\Delta = 4(10^n + 3)$

TAB.: Overall time in seconds

n	Old	New
40	35.6	15.5
45	107.0	57.0
50	224.0	119.0
55	756.0	271.0
60	1535.0	655.0
65	24607.0	3125.0
70	38818.0	9991.0

Heroic computations

In the imaginary case, let $\Delta_n = -4(10^n + 1)$

Heroic computations

In the imaginary case, let $\Delta_n = -4(10^n + 1)$

$$\mathrm{Cl}_{\Delta_{100}} \cong C(2)^7 \times C(146249177947219527457169431585749\\5335176880879072)$$

$$\mathrm{Cl}_{\Delta_{110}} \cong C(2)^{11} \times C(857640364195029289112195513145214\\883828429420071440)$$

Heroic computations

In the imaginary case, let $\Delta_n = -4(10^n + 1)$

$$\mathrm{Cl}_{\Delta_{100}} \cong C(2)^7 \times C(146249177947219527457169431585749\\ 5335176880879072)$$

$$\mathrm{Cl}_{\Delta_{110}} \cong C(2)^{11} \times C(857640364195029289112195513145214\\ 8838284294200071440)$$

In the real case, let $\Delta_{110} = 4(10^{110} + 3)$

Heroic computations

In the imaginary case, let $\Delta_n = -4(10^n + 1)$

$$\mathrm{Cl}_{\Delta_{100}} \cong C(2)^7 \times C(1462491779472195274571694315857495335176880879072)$$

$$\mathrm{Cl}_{\Delta_{110}} \cong C(2)^{11} \times C(8576403641950292891121955131452148838284294200071440)$$

In the real case, let $\Delta_{110} = 4(10^{110} + 3)$

$$\mathrm{Cl}_{\Delta_{110}} \cong \mathbb{Z}/12\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \ ,$$

Heroic computations

In the imaginary case, let $\Delta_n = -4(10^n + 1)$

$$\mathrm{Cl}_{\Delta_{100}} \cong C(2)^7 \times C(1462491779472195274571694315857495335176880879072)$$

$$\mathrm{Cl}_{\Delta_{110}} \cong C(2)^{11} \times C(8576403641950292891121955131452148838284294200071440)$$

In the real case, let $\Delta_{110} = 4(10^{110} + 3)$

$$\mathrm{Cl}_{\Delta_{110}} \cong \mathbb{Z}/12\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z} \ ,$$

$$R_{\Delta_{110}} \approx 70795074091059722608293227655184666748799878533480399.67302$$

Conclusion

Thank you for your attention