ECC 2008
September 22-24, Utrecht

Double-Base Number Systems and Applications

MACQUARIE
UNIVERSITY ~ SYDNEY

Christophe Doche

http://www.ics.mq.edu.au/~doche/

Acknowledgement

Joint works with

Vassil Dimitrov

Laurent Habsieger

Laurent Imbert

David R. Kohel

Francesco Sica

Overview

Classical techniques

NAF, JSF, τ-NAF

DBNS

Extended DBNS

Tree-Based Approach

Joint-DBNS

τ-DBNS

Scalar Multiplication

We are interested in techniques to compute a scalar multiplication as efficiently as possible

Definition. Given an integer n and a point P on a curve, a scalar multiplication consists in computing

$$[n]P = \underbrace{P + \cdots + P}_{n \text{ times}}$$

Scalar Multiplication

The standard way to compute $[n]P$ is the double-and-add method

The method relies on the following operations:

- addition $P + Q$, when $P \neq \pm - Q$
- doubling $[2]P$

Scalar Multiplication

We won't discuss the complexities of doublings and additions

There are several parametrizations and many coordinate systems

See `http://www.hyperelliptic.org/EFD/`

Signed Digit Representations

A signed-binary representation of n is any expansion such that

$$n = \sum_{i=0}^{\ell-1} n_i 2^i, \quad \text{with} \quad n_i \in \{-1, 0, 1\}$$

One class called the Non-Adjacent Form plays a special role

Every integer n has a unique NAF expansion such that $n_i n_{i+1} = 0$

Signed Digit Representations

The density of the NAF is $\frac{1}{3}$

Among all the signed-binary representations this number is minimal for the NAF

In case there is some memory available to store more points, we can use nontrivial coefficients

Signed Digit Representations

The window-NAF method of length w denoted by NAF_w is a straightforward generalization

It requires $2^{w-2} - 1$ precomputations

The density of the NAF_w is $\frac{1}{w+1}$

Multi-Scalar Multiplication

For certain protocols, e.g. a signature verification, it is necessary to compute a double-scalar multiplication of the form

$$[n]P + [m]Q$$

Instead of computing $[n]P$ and $[m]Q$ separately, we can try to compute them *simultaneously*

Multi-Scalar Multiplication

One idea, known as Shamir's trick, consists in representing n and m jointly

$$\binom{n}{m} = \begin{pmatrix} n_{\ell-1} \ldots n_0 \\ m_{\ell-1} \ldots m_0 \end{pmatrix}$$

Then mimick the double-and-add method...

Multi-Scalar Multiplication

Remarks.

This divides the number of doublings by 2

Also, if we precompute $P + Q$, at most one addition is necessary at each step

If we precompute $P - Q$ as well, we have more freedom to use signed digit representations

For instance, the Joint Sparse Form whose density is $\frac{1}{2}$

Multi-Scalar Multiplication

Example.

The joint sparse form of $n = 542788$ and $m = 462444$ is equal to

$$\binom{n}{m} = \begin{pmatrix} 100\bar{1}000100\bar{1}0100\bar{1}0\bar{1}00 \\ 100001001000001000100 \end{pmatrix}_{\mathrm{JSF}}$$

The computation of $[n]P + [m]Q$ requires 20 doublings and 9 additions, given that $P+Q$ and $P-Q$ are precomputed and stored

Koblitz curves

A Koblitz curve E_a is an elliptic curve of the form

$$E_a : y^2 + xy = x^3 + ax^2 + 1, \text{ with } a = 0 \text{ or } 1$$

The Frobenius map $\phi(x, y) = (x^2, y^2)$ is an endomorphism of $E_a(\mathbb{F}_{2^d})$

Koblitz curves

Let $\tau \in \mathbb{C}$ satisfies such that $\tau^2 + (-1)^a \tau + 2 = 0$

Every integer n has a τ-NAF expansion

$$n = \sum_i n_i \tau^i, \quad \text{with} \quad n_i \in \{0, \pm 1\}$$

and such that $n_i n_{i+1} = 0$

The density of the τ-NAF is $\frac{1}{3}$

Koblitz curves

This implies that

$$[n]P = \sum_i n_i \phi^i(P)$$

and $[n]P$ can be obtained with a Frobenius-and-add method

Double-Base Number System

Double-Base Number System (Dimitrov *et al.* 1995)

Representation of an integer n as

$$n = \sum_{i=1}^{\ell} \pm 2^{a_i} 3^{b_i}$$

Example. $841232 = 2^7 3^8 + 2^1 3^6 - 2^2 3^2 + 2^1$

In general, DBNS expansions are very sparse

Double-Base Number System

Such a representation always exists and is not unique

Used with optimized triplings, this system seems promising

There is a greedy algorithm to find a DBNS expansion

Greedy algorithm

Starting from $t = n$

- Find at each step the best approximation of t as $z = 2^a 3^b$

- Do $t \leftarrow |t - z|$

- Repeat this operation until t is 0

Greedy algorithm

Example.

$$841232 \quad = \quad 2^7 3^8 + 1424$$

Greedy algorithm

Example.

$$\begin{aligned} 841232 &= 2^7 3^8 + 1424 \\ 1424 &= 2^1 3^6 - 34 \end{aligned}$$

Greedy algorithm

Example.

$$\begin{aligned} 841232 &= 2^7 3^8 + 1424 \\ 1424 &= 2^1 3^6 - 34 \\ 34 &= 2^2 3^2 - 2 \end{aligned}$$

Greedy algorithm

Example.

$$\begin{aligned} 841232 &= 2^7 3^8 + 1424 \\ 1424 &= 2^1 3^6 - 34 \\ 34 &= 2^2 3^2 - 2 \end{aligned}$$

Finally,

$$841232 = 2^7 3^8 + 2^1 3^6 - 2^2 3^2 + 2$$

Best approximation

To find the best approximation of t as $2^a 3^b$, it is possible to scan the points with integer coordinates near the line

$$y = -x \log_3 2 + \log_3 t$$

Remark. This method is not really efficient and does not seem to lead to realistic implementations when the scalar is chosen on the fly

Best approximation

Other idea

Precompute binary expansions of $3^0, 3^1, \ldots, 3^m$ and order them with respect to lexicographic order

Then consider the binary expansion of n and find the closest element in the table

Finally, adjust the length by multiplying by the appropriate power of 2

Example.

The binary expansion of $n = 841232$ is

$$(11001101011000010000)_2$$

Example.

$$
\begin{array}{ll}
[(1)_2, & 0] \\
[(100010001011)_2, & 7] \\
[(1001)_2, & 2] \\
[(100110011100011)_2, & 9] \\
[(1010001)_2, & 4] \\
[(1011011001)_2, & 6] \\
[(11)_2, & 1] \\
[(1100110100001)_2, & 8] \\
[(11011)_2, & 3] \\
[(1110011010101001)_2, & 10] \\
[(11110011)_2, & 5]
\end{array}
$$

Example.

$$
\begin{array}{ll}
[(1)_2, & 0] \\
[(100010001011)_2, & 7] \\
[(1001)_2, & 2] \\
[(100110011100011)_2, & 9] \\
[(1010001)_2, & 4] \\
[(1011011001)_2, & 6] \\
[(11)_2, & 1] \\
\color{red}{[(1100110100001)_2,} & \color{red}{8]} \\
\color{red}{[(11011)_2,} & \color{red}{3]} \\
[(1110011010101001)_2, & 10] \\
[(11110011)_2, & 5]
\end{array}
$$

Example.

$$
\begin{aligned}
2^7 3^8 &= (11001101000010000000)_2 \\
841232 &= (11001101011000010000)_2 \\
2^{15} 3^3 &= (11011000000000000000)_2
\end{aligned}
$$

A direct computation shows that $2^7 3^8$ is the closer approximation of n

Greedy algorithm

Theorem.

For every positive integer n, the length of the expansion of n returned by this greedy approach is

$$O\left(\frac{\log n}{\log\log n}\right)$$

The proof uses a result of Tijdeman on the distribution of integers of the form $2^a 3^b$

Greedy algorithm

Heuristic argument.

Consider the first bits of powers of 3

There are $\log_3 n$ powers of 3 less than n

The first bit is 1 followed by a random sequence

Looking at the first $\log_2 \log_3 n$ bits, all the possible sequences are represented

Greedy algorithm

$$
\begin{array}{lr}
[(1)_2, & 0] \\
[(100010001011)_2, & 7] \\
[(1001)_2, & 2] \\
[(1001100111000 11)_2, & 9] \\
[(1010001)_2, & 4] \\
[(1011011001)_2, & 6] \\
[(11)_2, & 1] \\
[(1100110100001)_2, & 8] \\
[(11011)_2, & 3] \\
[(1110011010101001)_2, & 10] \\
[(11110011)_2, & 5]
\end{array}
$$

Greedy algorithm

Heuristic argument.

Consider the first bits of powers of 3

There are $\log_3 n$ powers of 3 less than n

The first bit is 1 followed by a random sequence

Looking at the first $\log_2 \log_3 n$ bits, all the possible sequences are represented

So we can clear $\log_2 \log_3 n$ bits at each step

Double-Base Number System

Let us try to compute $[841232]P$ with this representation

$$841232 = 2^7 3^8 + 2^1 3^6 - 2^2 3^2 + 2$$

We can proceed from right-to-left or left-to-right

Double-Base Number System

$$841232 = 2^7 3^8 + 2^1 3^6 - 2^2 3^2 + 2$$

From right-to-left, we start with $[2]P$, then apply 1 doubling and 2 triplings to get $[2^2 3^2]P$

To get the next term, we cannot use $[2^2 3^2]P$, as we would be obliged to divide by 2

So, if we didn't store $[2^1 3^2]P$ we have to compute $[2^1 3^6]P$ from scratch

It doesn't work!

Double-Base Number System

$$841232 = 2^7 3^8 + 2^1 3^6 - 2^2 3^2 + 2$$

From left-to-right, we start with $[2^6 3^2]P$

Then we add P, and we should apply $2^{-1}3^4$ to the result

It doesn't work either!

Double-Base Number System

That is why DBNS expansions are not used under this form

Indeed, if the two sequences of exponents are not simultaneously decreasing, it seems impossible to use only $\max a_i$ doublings **and** $\max b_i$ triplings

$$n = \sum_{i=1}^{\ell} \pm 2^{a_i} 3^{b_i}$$

Double-Base Number System

That is why DBNS expansions are not used under this form

Indeed, if the two sequences of exponents are not simultaneously decreasing, it seems impossible to use only $\max a_i$ doublings **and** $\max b_i$ triplings

That is why the concept of double-base chain has been introduced where one asks also:

$$a_1 \geqslant a_2 \geqslant \cdots \geqslant a_\ell \quad \text{and} \quad b_1 \geqslant b_2 \geqslant \cdots \geqslant b_\ell$$

Double-Base Number System

The greedy algorithm can easily be modified to compute such a DB-Chain

Double-Base Number System

The greedy algorithm can easily be modified to compute such a DB-Chain

Example.

$$841232 \quad = \quad 2^7 3^8 + 1424$$

Double-Base Number System

The greedy algorithm can easily be modified to compute such a DB-Chain

Example.

$$\begin{aligned} 841232 &= 2^7 3^8 + 1424 \\ 1424 &= 2^1 3^6 - 34 \end{aligned}$$

Double-Base Number System

The greedy algorithm can easily be modified to compute such a DB-Chain

Example.

$$\begin{aligned} 841232 &= 2^7 3^8 + 1424 \\ 1424 &= 2^1 3^6 - 34 \\ 34 &= 2^2 3^2 - 2 \end{aligned}$$

Double-Base Number System

The greedy algorithm can easily be modified to compute such a DB-Chain

Example.

$$\begin{aligned} 841232 &= 2^7 3^8 + 1424 \\ 1424 &= 2^1 3^6 - 34 \\ 34 &= 3^3 + 7 \end{aligned}$$

Double-Base Number System

The greedy algorithm can easily be modified to compute such a DB-Chain

Example.

$$
\begin{aligned}
841232 &= 2^7 3^8 + 1424 \\
1424 &= 2^1 3^6 - 34 \\
34 &= 3^3 + 7 \\
7 &= 3^2 - 2
\end{aligned}
$$

Double-Base Number System

The greedy algorithm can easily be modified to compute such a DB-Chain

Example.

$$
\begin{aligned}
841232 &= 2^7 3^8 + 1424 \\
1424 &= 2^1 3^6 - 34 \\
34 &= 3^3 + 7 \\
7 &= 3^2 - 2 \\
2 &= 3^1 - 1
\end{aligned}
$$

Double-Base Number System

So

$$841232 = 2^7 3^8 + 2^1 3^6 - 3^3 - 3^2 + 3^1 - 1$$

We deduce that

$$[841232]P \ =$$

$$[3]\Big([3]\Big([3]\Big([2^1 3^3]([2^6 3^2]P + P) - P\Big) - P\Big) + P\Big) - P$$

The right-to-left approach works as well

Double-Base Number System

Remark. It is not known if the average length of double-base chains returned by this modified greedy algorithm is still

$$O\left(\frac{\log n}{\log \log n}\right)$$

But there are some heuristics against that

Double-Base Number System

If we have some memory available to store pre-computed points, it is probably worthwhile to use nontrivial coefficients

Extended DBNS

Given a finite set $\mathcal{S}$ of coefficients, we introduce the concept of extended DB-Chain, denoted by $\mathcal{S}$-DB-Chain

An integer n is then represented as

$$n = \sum_{i=1}^{\ell} s_i 2^{a_i} 3^{b_i} \quad \text{with} \quad |s_i| \in \mathcal{S} \quad \text{and}$$

$$a_1 \geqslant a_2 \geqslant \cdots \geqslant a_\ell \ \text{ and } \ b_1 \geqslant b_2 \geqslant \cdots \geqslant b_\ell$$

Extended DBNS

Remark. The shortest expansions are obtained when the coefficients in $\mathcal{S}$ are coprime with 6

Example. We have

$$841232 = 2^7 3^8 + 5 \times 2^5 3^2 - 2^4$$

We deduce that

$$[841232]P = [2^4]\big([2^1 3^2]([2^4 3^6]P + [5]P) - P\big)$$

Extended DBNS

Again, such an expansion can be obtained by a modified version of the greedy algorithm

At each step, it is enough to find the best approximation of t in terms of $s2^a3^b$ with

$$s \in \mathcal{S}, a \leqslant \alpha \text{ and } b \leqslant \beta$$

There is an additional degree of freedom

Alternative methods to find DB-Chains

The binary-ternary method (Ciet *et al.*) can be used to produce a DB-Chain

Take $n > 1$ coprime to 6

Let $n' = n - 1$ or $n' = n + 1$ so that $6 \mid n'$

Clear powers of 2 and 3 from n' and reapply the process until you reach 1

Tree-based approach

Instead of choosing between $n-1$ and $n+1$, keep both of them

Build a tree having 2 leaves $n-1$ and $n+1$

Clear powers of 2 and 3 from $n-1$ and $n+1$, and reapply the process for each node

Repeat to create a binary tree

Eventually, one of its branch will reach 1 leading to a DB-Chain expansion

Tree-based approach

Obviously, this can't be done for integers in the cryptographic range, say at least 160 bits

But, we can eliminate most branches and hope for the best

For instance, keep only the B smallest nodes before creating the next level of the tree

What is remarkable is that even small values of B give very good result

Tree-based approach

Algorithm. Tree-based DB-Chain search

INPUT: An integer n and a bound B.
OUTPUT: A binary tree to compute a DB-Chain for n.

Set $t \leftarrow f(n)$ $\quad [f(n) = n/\left(2^{v_2(n)}3^{v_3(n)}\right)]$

Initialize a binary tree $\mathcal{T}$ with root node t

repeat

for each leaf node m in $\mathcal{T}$ insert 2 children

Left child $\leftarrow f(m-1)$

Right child $\leftarrow f(m+1)$

Discard any redundant leaf node

Discard all but the B smallest leaf nodes

until a leaf node is equal to 1

return $\mathcal{T}$

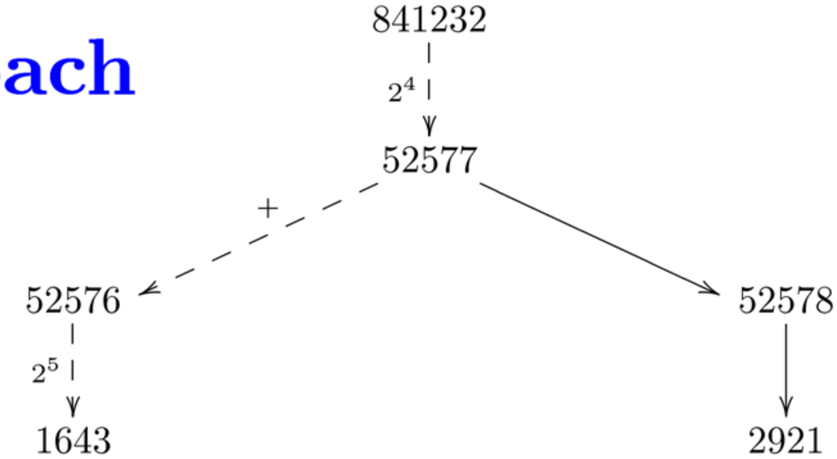
841232

Tree-based approach

$B = 2$

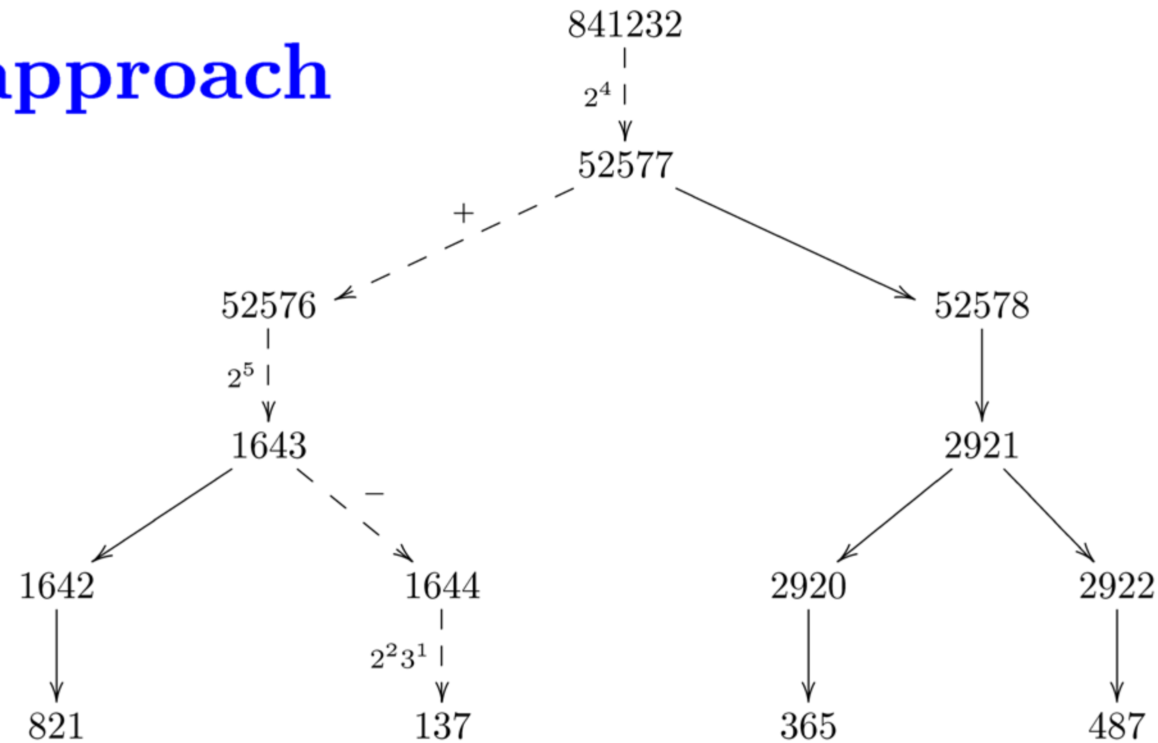
Tree-based approach

$$B = 2$$



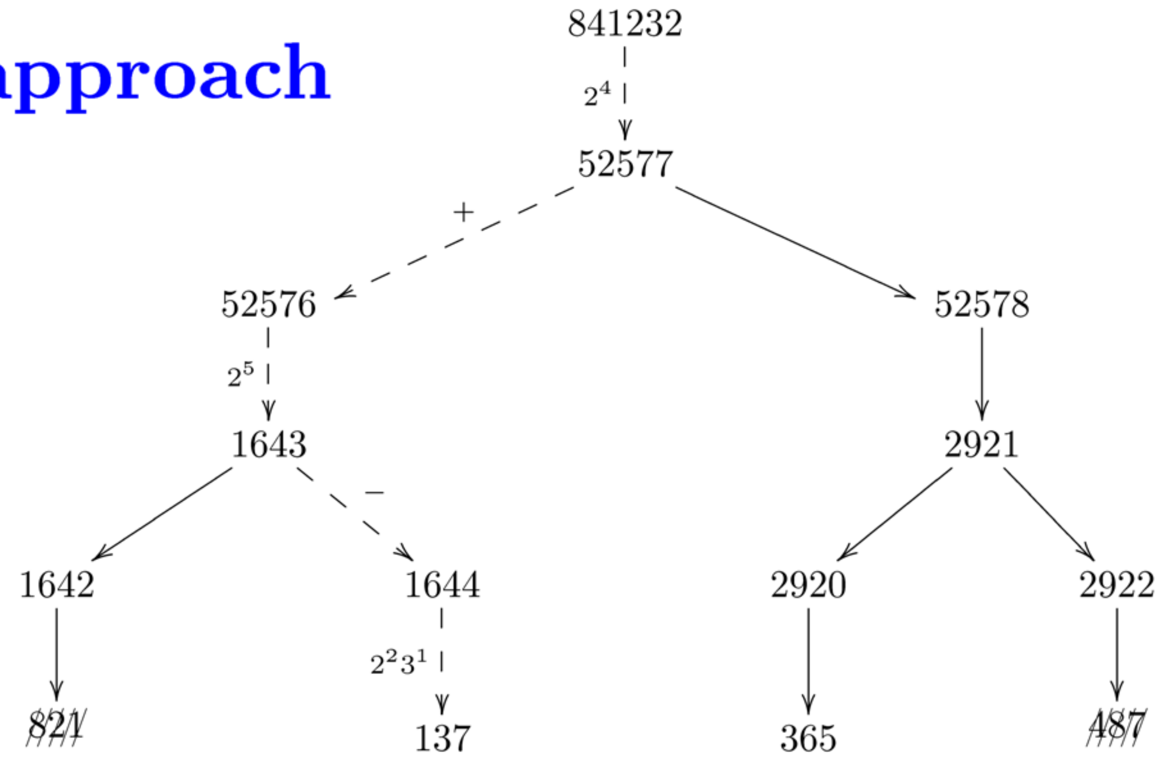
Tree-based approach

$$B = 2$$



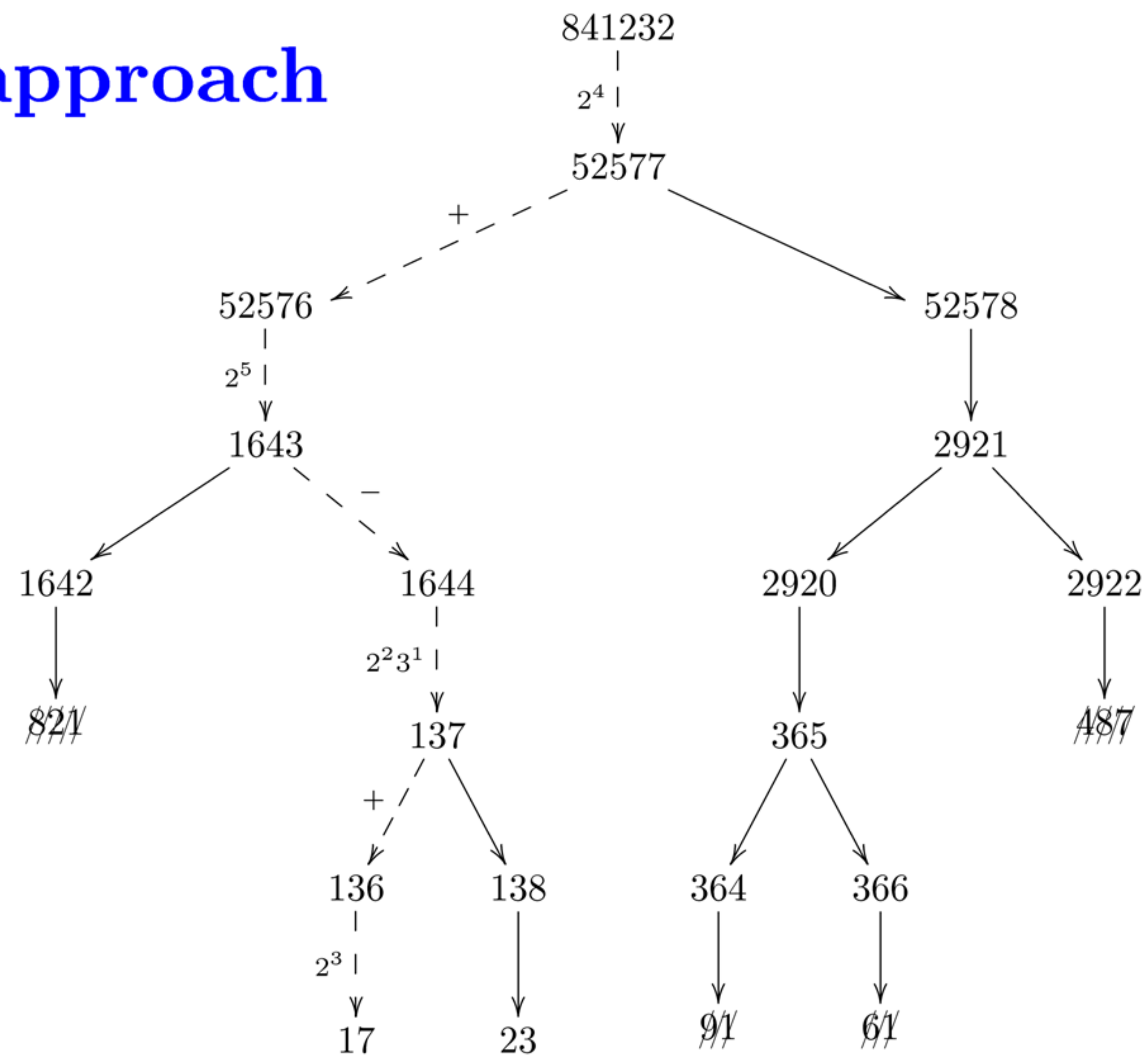
Tree-based approach

$B = 2$



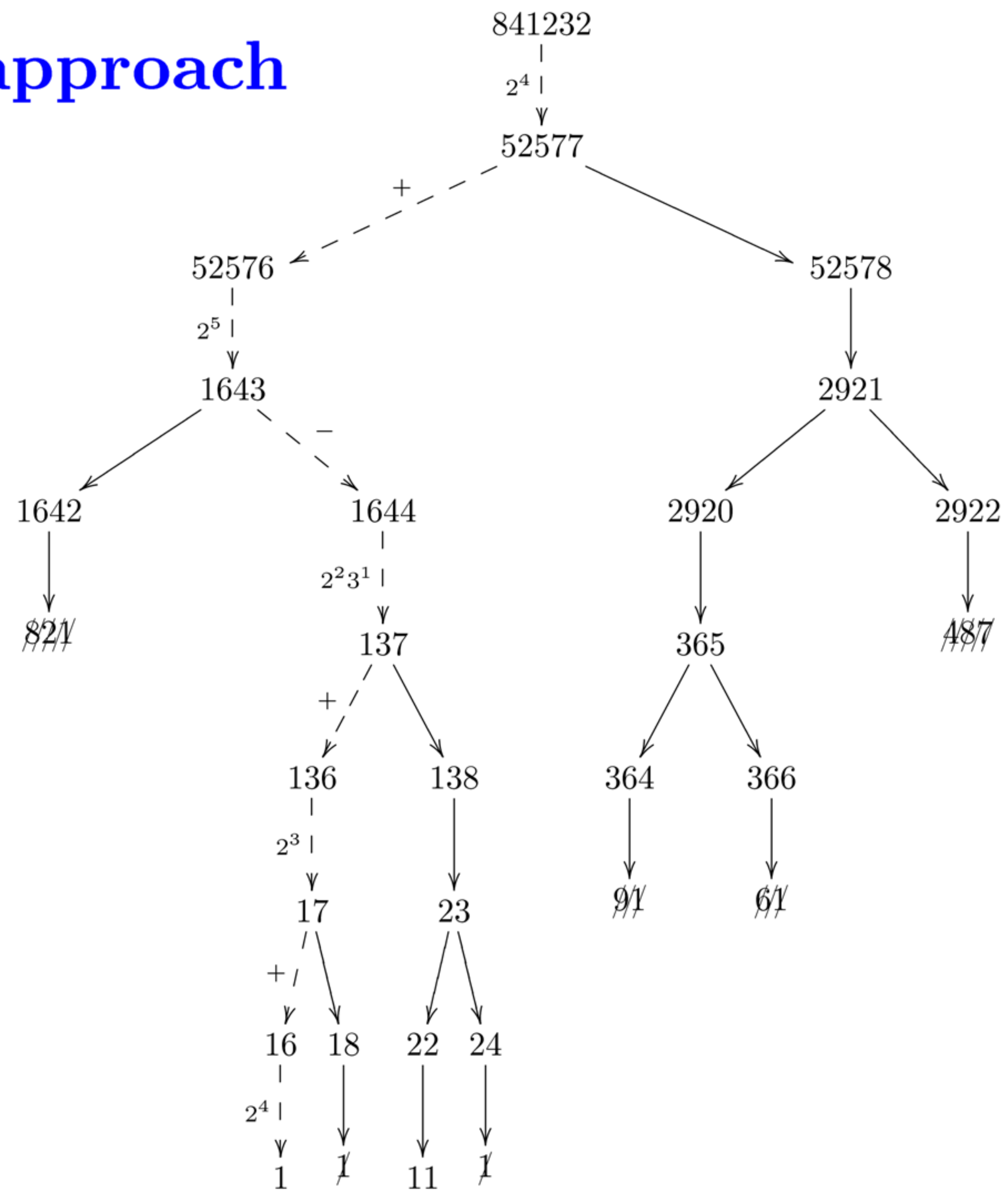
Tree-based approach

$B = 2$



Tree-based approach

$B = 2$



Tree-based approach

We deduce from this tree that

$$841232 = 2^4(2^5(2^2 3^1(2^3(2^4+1)+1)-1)+1)$$

which implies that

$$841232 = 2^{18}3^1 + 2^{14}3^1 - 2^{11}3^1 + 2^9 + 2^4$$

For $B = 2$, that is one term less than for the chain obtained with the greedy algorithm

$$841232 = 2^7 3^8 + 2^1 3^6 - 3^3 - 3^2 + 3^1 - 1$$

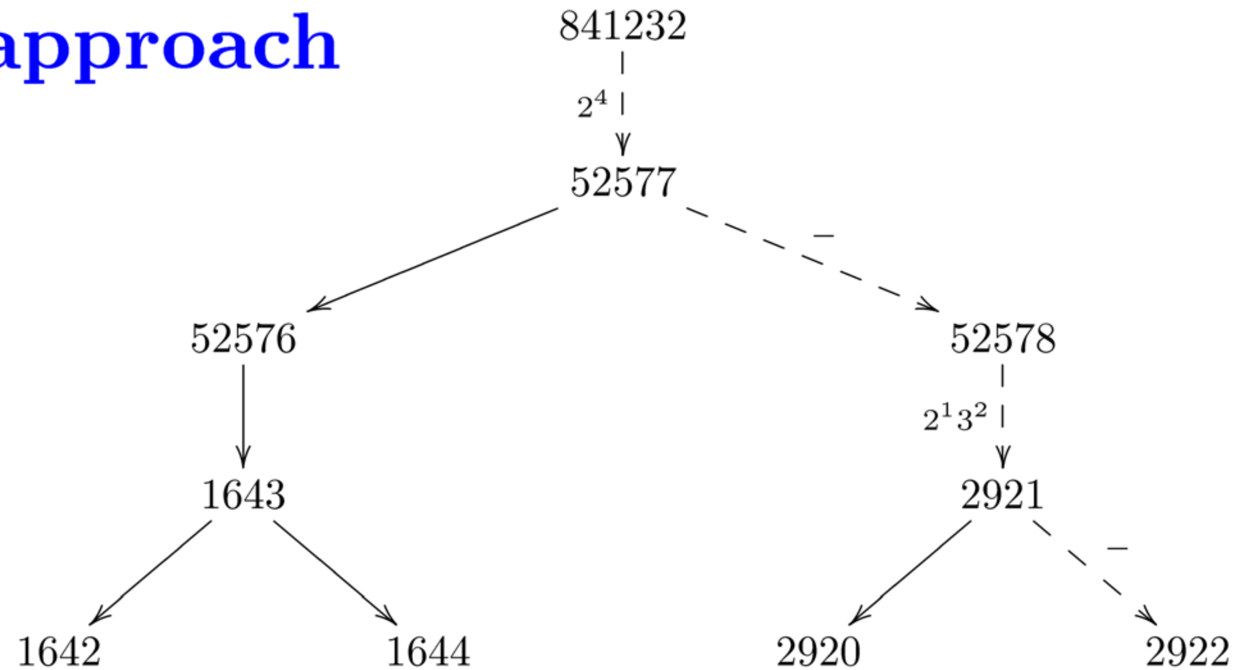
Tree-based approach

841232

$B = 4$

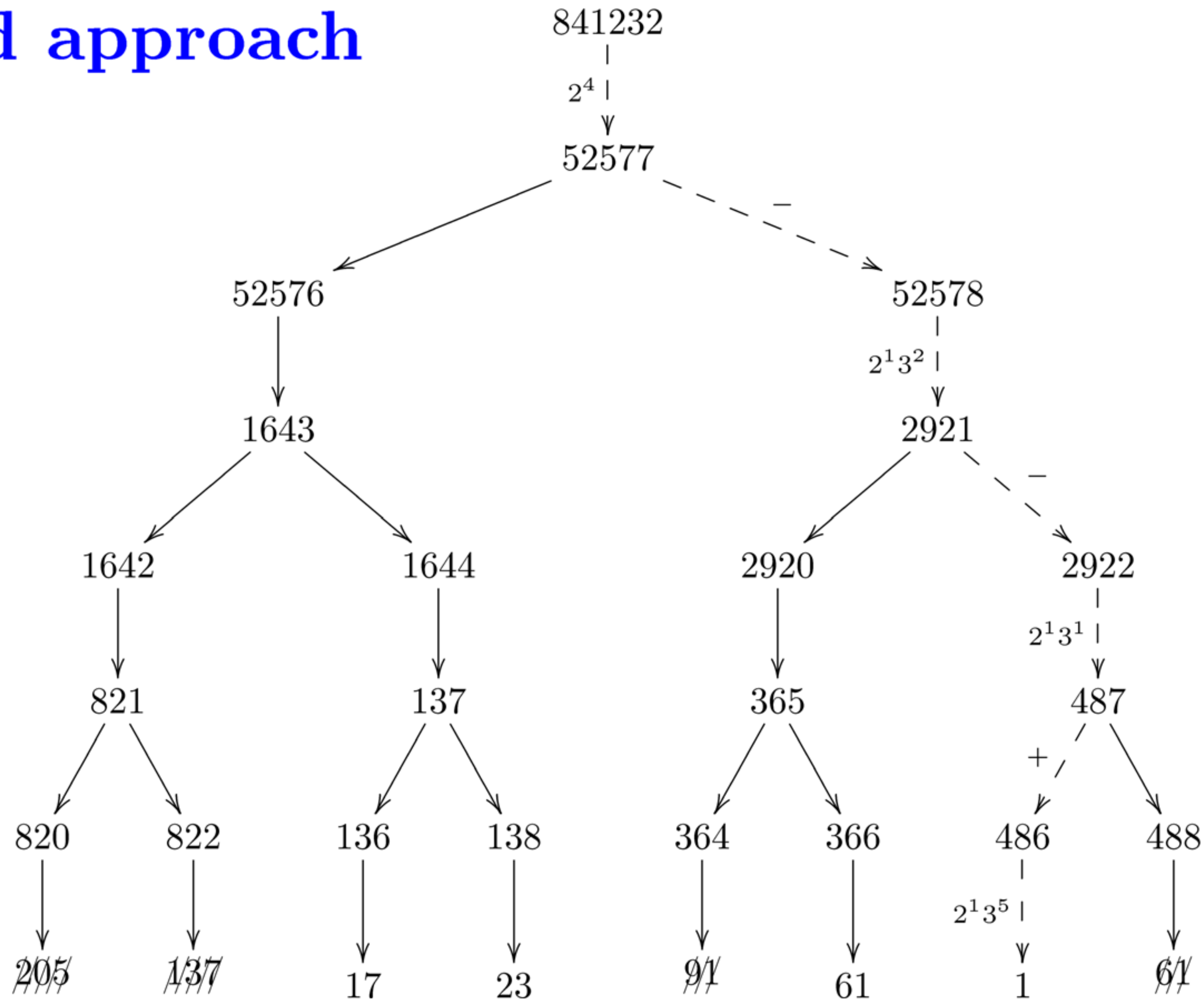
Tree-based approach

$$B = 4$$



Tree-based approach

$$B = 4$$



Tree-based approach

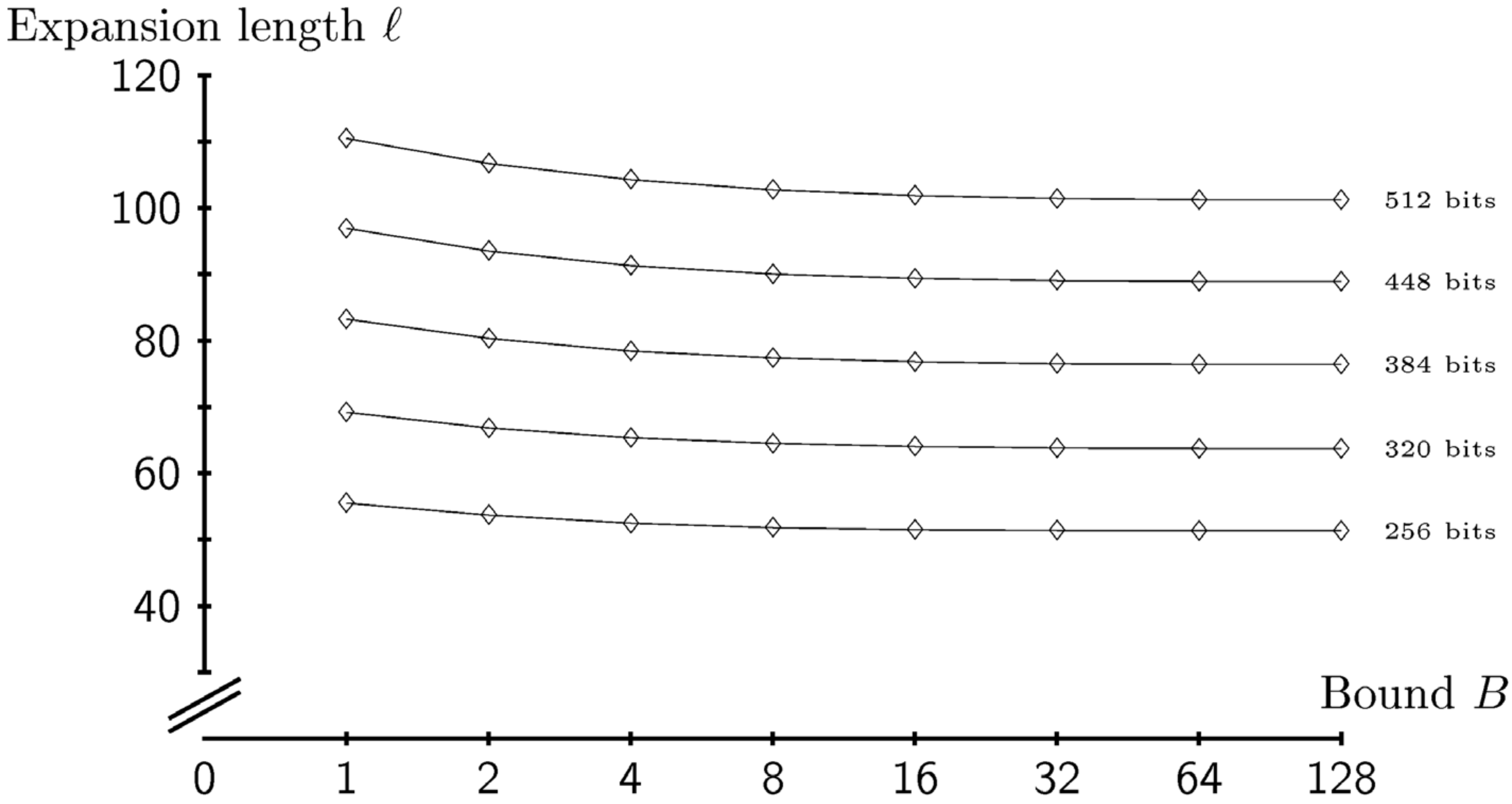
We deduce from this tree that

$$841232 = 2^4\Big(2^1 3^2\Big(2^1 3^1\big(2^1 3^5 + 1\big) - 1\Big) - 1\Big)$$

that leads to the even shorter DB-chain

$$841232 = 2^7 3^8 + 2^6 3^3 - 2^5 3^2 - 2^4$$

Experiments



Impact of B on the length of the expansion

Complexity Analysis of the Binary-Ternary

It is easy to compute the probability that a multiple of 6 is divisible by $2^\alpha 3^\beta$, for $\alpha, \beta \geqslant 1$

It is

$$\frac{1}{2^{\alpha-1}}\left(1-\frac{1}{2}\right)\frac{1}{3^{\beta-1}}\left(1-\frac{1}{3}\right)$$

The corresponding gain is $\alpha + \beta \log_2 3$

Complexity Analysis of the Binary-Ternary

So, the average number of bits gained at each step is

$$\sum_{\alpha=1}^{\infty}\sum_{\beta=1}^{\infty}\frac{\alpha+\beta\log_2 3}{2^{\alpha-1}3^{\beta}} = 2+\frac{3}{2}\log_2 3$$

$$= 4.3774\ldots$$

Complexity Analysis of the Tree-Based

We can do a little better by looking at the two branches

The same kind of probabilistic arguments shows that

$$\sum_{\alpha=3}^{\infty}\sum_{\beta=1}^{\infty}\frac{2(\alpha-1)+2\beta\log_2 3+2\max(\alpha-1,1+\beta\log_2 3)}{2^{\alpha-1}3^{\beta}}$$

$$= 4.6419\ldots$$

Tree-based approach

There are many possible generalizations

For instance, it is easy to compute extended DB-Chains

Tree-based approach

Algorithm. Tree-based extended DB-Chain search

INPUT: An integer n, a bound B, and a set $\mathcal{S}$.
OUTPUT: A binary tree to compute a DB-Chain for n.

Set $t \leftarrow f(n)$ $\quad [f(n) = n/\left(2^{v_2(n)}3^{v_3(n)}\right)]$
Initialize a binary tree $\mathcal{T}$ with root node t
repeat
    for each leaf node m in $\mathcal{T}$ insert $2|\mathcal{S}|$ children
        corresponding to $f(m \pm s)$ with $s \in \mathcal{S}$
    Discard any redundant leaf node
    Discard all but the B smallest leaf nodes
until a leaf node is equal to 1
return $\mathcal{T}$

Tree-based approach

There are many possible generalizations

Or to return expansions using 3 bases (2, 3, and 5) instead of just 2 and 3

Tree-based approach

Algorithm. Tree-based SMBR search

INPUT: An integer n and a bound B.
OUTPUT: A binary tree to compute a SMBR chain for n.

```
Set t ← g(n)                                  [g(n) = n/(2^{v_2(n)} 3^{v_3(n)} 5^{v_5(n)})]
Initialize a binary tree 𝒯 with root node t
repeat
    for each leaf node m in 𝒯 insert 2 children
        Left child ← g(m − 1)
        Right child ← g(m + 1)
    Discard any redundant leaf node
    Discard all but the B smallest leaf nodes
until a leaf node is equal to 1
return 𝒯
```

where $g(n) = n/\left(2^{v_2(n)} 3^{v_3(n)} 5^{v_5(n)}\right)$.

Results

The Tree-based method is easier to implement and returns chains 10% shorter than the greedy

Average length of the chains returned by the tree-based algorithm is proven to be $\log n/C$

So the length of DB-Chains returned by the greedy is probably not

$$O\left(\frac{\log n}{\log\log n}\right)$$

Open questions

What is the optimal length for a given size?

What can be said about the distribution of DB-Chain expansions of a given length with a given size?

Joint-DBNS

The Joint Double-Base Number System (JDBNS) allows to represent two integers n and m as

$$\begin{pmatrix} n \\ m \end{pmatrix} = \sum_{i=1}^{\ell} \begin{pmatrix} c_i \\ d_i \end{pmatrix} 2^{a_i} 3^{b_i}$$

with $c_i, d_i \in \{-1, 0, 1\}$

Use the redundancy and flexibility of signed expansions to find a common short representation

Joint-DBNS

There is a notion of Joint-Binary-Ternary Algorithm

Starting from a pair (n, m), look at all the pairs $(n - c, m - d)$ with $c, d \in \{-1, 0, 1\}$

And select the one such that $(n - c, m - d)$ has the largest factor of the form $2^\alpha 3^\beta$

Joint-DBNS

Just like the binary-ternary method, a probabilistic argument gives the complexity

But it is not totally straightforward to obtain the probabilities

Theorem. The average joint density of the expansion returned by this algorithm is less than 0.3945

Joint-DBNS

Example.

The joint sparse form of $n = 542788$ and $m = 462444$ is equal to

$$\begin{pmatrix} n \\ m \end{pmatrix} = \begin{pmatrix} 100\bar{1}000100\bar{1}0100\bar{1}0\bar{1}00 \\ 10000100100001000100 \end{pmatrix}_{\text{JSF}}$$

The computation of $[n]P + [m]Q$ requires 20 doublings and 9 additions, given that $P+Q$ and $P-Q$ are precomputed and stored

Joint-DBNS

Example.

$$\begin{pmatrix}542788\\462444\end{pmatrix} = \begin{pmatrix}1\\1\end{pmatrix}2^{11}3^5 + \begin{pmatrix}1\\\bar{1}\end{pmatrix}2^9 3^4 + \begin{pmatrix}1\\1\end{pmatrix}2^6 3^4 + \begin{pmatrix}\bar{1}\\1\end{pmatrix}2^4 3^4$$

$$- \begin{pmatrix}1\\1\end{pmatrix}2^3 3^3 + \begin{pmatrix}\bar{1}\\0\end{pmatrix}2^2 3^2 + \begin{pmatrix}1\\\bar{1}\end{pmatrix}2^2 3 + \begin{pmatrix}1\\0\end{pmatrix}2^2$$

It needs 11 doublings and 5 triplings but only 7 additions

Joint-DBNS

Even with systems having very cheap doublings (typically Inverted Edwards coordinates) for which the DBNS does not bring anything re. scalar multiplications

This Joint-DBNS is faster than the JSF analogue

τ-DBNS

It is natural to generalize the use of double-bases for Koblitz curves

For instance, we can represent an integer n as

$$n = \sum_{i=1}^{\ell} \pm \tau^{a_i} z^{b_i},$$

where $z = 3$ or $\bar{\tau}$

τ-DBNS

The best choice seems to be $\bar{\tau}$, since $\bar{\phi}$ can be evaluated very efficiently:

$$\bar{\phi}(P) = (-1)^{1-a_2} P - \phi(P)$$

In López-Dahab, one mixed-addition: 8M + 5S

Using halving: approx. 4M + 4S

Recently, we found closed formulae for $\bar{\phi}$ that need at most 2M + 2S

τ-DBNS

Take $a_2 = 1$

$$P_2 = \overline{\phi}(P_1) = P_1 - \phi(P_1) = (X_2 : Y_2 : Z_2)$$

$$\begin{aligned} X_2 &= (X_1 + Z_1)^2 \\ Z_2 &= X_1 Z_1 \\ Y_2 &= (Y_1 + X_2)(Y_1 + X_2 + Z_2) \end{aligned}$$

τ-DBNS

An algorithm to compute joint-$\tau\bar{\tau}$ expansions has also been investigated providing a speed-up of 8 to 9% over the τ-JSF