A new Constraint Programming model for the Multiple Constant Multiplication

RAIM Meeting 2025: 16th Rencontres de l'Arithmétique en Informatique Mathématique

Théo Cantaloube[1] Xiao Peng[2] Christine Solnon[1] Anastasia Volkova[1]

[1]Emeraude, CITI, INSA Lyon / Inria, France [2]LAAS-CNRS, Toulouse, France

November 4, 2025

Introduction

Introduction

- Embedded computations everywhere
- Arithmetic operators tailored to applications' requirements

Multiple Constant Multiplication problem (MCM)

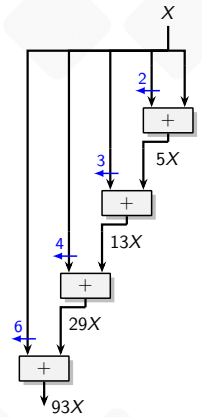
- Multiplying an integer by several fixed integer target constants
- Exploiting *a priori* knowledge of constants to design optimized implementations
- Avoiding costly generic multiplier

MCM is used in:

- Digital Signal Processing for linear digital filter evaluation [Aksoy, 2011; Garcia, 2022; Howard, 2014; Kumm, 2023; Meher, 2011]
- Discrete Transforms (Discrete Cosine, Fast Fourier) [Ghissoni, 2010; Wendt, 2015]
- Inference of Deep Neural Networks [Garcia, 2025; Habermann, 2022]

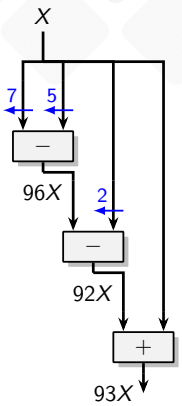
Motivation - Single Constant Multiplication

$$
\begin{aligned}
X &= \boxed{x_6}\boxed{x_5}\boxed{x_4}\boxed{x_3}\boxed{x_2}\boxed{x_1}\boxed{x_0} \\
\times\ 93 &= \boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{1} \\
= \quad & \\
1 \times & \boxed{x_6}\boxed{x_5}\boxed{x_4}\boxed{x_3}\boxed{x_2}\boxed{x_1}\boxed{x_0} \\
+\ 0 \times & \boxed{x_6}\boxed{x_5}\boxed{x_4}\boxed{x_3}\boxed{x_2}\boxed{x_1}\boxed{x_0} \\
+\ 1 \times & \boxed{x_6}\boxed{x_5}\boxed{x_4}\boxed{x_3}\boxed{x_2}\boxed{x_1}\boxed{x_0} \quad 2 \\
+\ 1 \times & \boxed{x_6}\boxed{x_5}\boxed{x_4}\boxed{x_3}\boxed{x_2}\boxed{x_1}\boxed{x_0} \quad 3 \\
+\ 1 \times & \boxed{x_6}\boxed{x_5}\boxed{x_4}\boxed{x_3}\boxed{x_2}\boxed{x_1}\boxed{x_0} \quad 4 \\
+\ 0 \times & \boxed{x_6}\boxed{x_5}\boxed{x_4}\boxed{x_3}\boxed{x_2}\boxed{x_1}\boxed{x_0} \\
+\ 1 \times & \boxed{x_6}\boxed{x_5}\boxed{x_4}\boxed{x_3}\boxed{x_2}\boxed{x_1}\boxed{x_0} \quad 6
\end{aligned}
$$



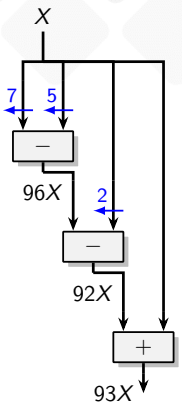
Binary approach for $T = \{93\}$:
$93_{dec} = 1011101_{bin}$

Motivation - Single Constant Multiplication

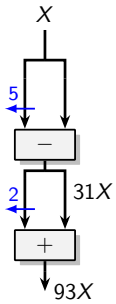


Canonical Sign Digit (CSD) approach for $T = \{93\}$:
$93_{dec} = 10\bar{1}00\bar{1}01_{CSD}$

Motivation - Single Constant Multiplication

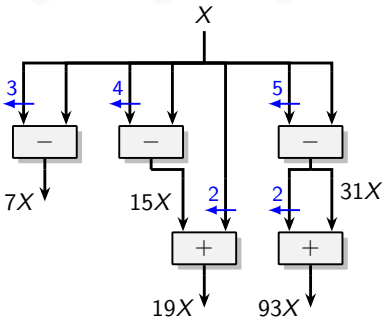


Canonical Sign Digit (CSD) approach for $T = \{93\}$:
$93_{dec} = 10\bar{1}00\bar{1}01_{CSD}$



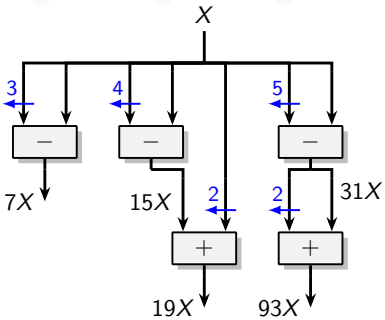
Optimal graph for $T = \{93\}$

Motivation - Multiple Constant Multiplication

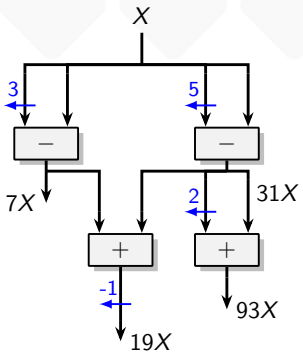


(a) Single constant approach for $T = \{7, 19, 93\}$

Motivation - Multiple Constant Multiplication

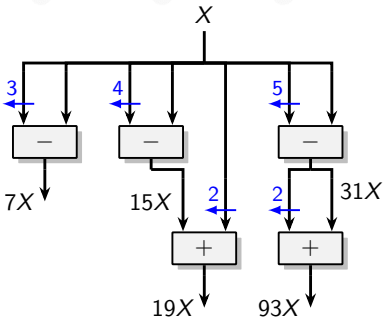


(a) Single constant approach for $T = \{7, 19, 93\}$

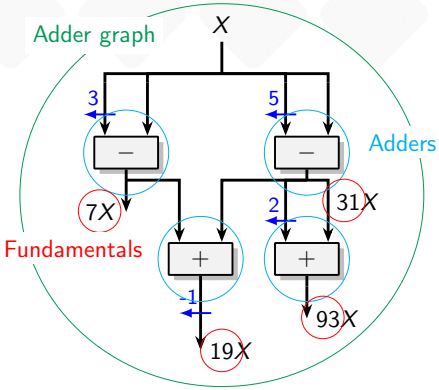


(b) Optimal graph for $T = \{7, 19, 93\}$

Motivation - Multiple Constant Multiplication



(a) Single constant approach for $T = \{7, 19, 93\}$

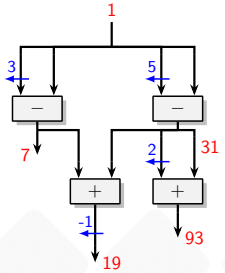


(b) Optimal graph for $T = \{7, 19, 93\}$

Formal definition - MCM Problem

- Optimization metric: number of adders
- ⚠ Doesn't perfectly reflect hardware cost but good proxy variable

Input: $T = \{7, 19, 93\}$, $w = 6$
Output:



Definition (MCM-Adders)

Problem: MCM-Adders(T, w)

Input: *Target constant set T, word-length w*

Output: *MCM implementation producing T with fundamentals written on w bits minimizing number of adders*

Motivation - Exact and declarative approaches

	ILP (CPLEX, Gurobi)	**CP** (ORTools, Choco)	**SAT** (ORTools, Gecode)
Variables	Integers / Binary	Finite domains	Boolean only
Constraints	Linear expressions	General: logical, global constraints	Logical clauses (CNF)
Objective	Optimization (min / max)	Satisfaction, optimization, enumeration	
Modeling	Laborious for combinatorial problems	Highly expressive (logic, scheduling)	Requires translation to Boolean logic
Solvers	Often copyright protected	Always license-free (maintained by community)	
Strengths	Powerful for numerical optimization	Very flexible for complex constraints	Extremely fast on Boolean problems
Limitations	Curse of linearization and numerical instabilities	Badly handles arithmetic constraints	Poorly handles counting

Motivation - Exact and declarative approaches

	ILP (CPLEX, Gurobi)	**CP** (ORTools, Choco)	**SAT** (ORTools, Gecode)
Variables	MCM problem already modeled with ILP by [Kumm, 2018] and [Garcia, 2023]	Finite domains	Boolean only
Constraints		General: logical, global constraints	Logical clauses (CNF)
Objective		Satisfaction, optimization, enumeration	
Modeling		Highly expressive (logic, scheduling)	Requires translation to Boolean logic
Solvers		Always license-free (maintained by community)	
Strengths		Very flexible for complex constraints	Extremely fast on Boolean problems
Limitations		Badly handles arithmetic constraints	Poorly handles counting

Motivation - Exact and declarative approaches

	ILP (CPLEX, Gurobi)	**CP** (ORTools, Choco)	**SAT** (ORTools, Gecode)
Variables	MCM problem already modeled with ILP by [Kumm, 2018] and [Garcia, 2023]	Finite domains	MCM problem already modeled with SAT by [Fiege, 2024]
Constraints		General: logical, global constraints	
Objective		Satisfaction, optimizatio	
Modeling		Highly expressive (logic, scheduling)	
Solvers		Always license-free (mai	
Strengths		Very flexible for complex constraints	
Limitations		Badly handles arithmetic constraints	

Motivation - Exact and declarative approaches

	ILP (CPLEX, Gurobi)	**CP** (ORTools, Choco)	**SAT** (ORTools, Gecode)
Variables	MCM problem already modeled with ILP by [Kumm, 2018] and [Garcia, 2023]	MCM problem waiting to be modeled with CP by Théo Cantaloube, Christine Solnon and Anastasia Volkova!	MCM problem already modeled with SAT by [Fiege, 2024]
Constraints			
Objective			
Modeling			
Solvers			
Strengths			
Limitations			

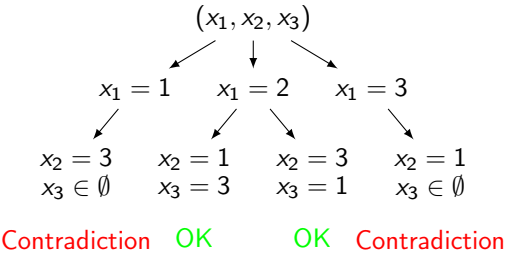
Motivation - Constraint Programming

- Each constraint is associated with a **propagator**.
- While not all variable are instantiated:
  - **Instantiate** a variable
  - **Propagate** modifications until a fix point
  - In case of contradiction, **backtrack** in the search tree

Motivation - Constraint Programming

- Each constraint is associated with a **propagator**.
- While not all variable are instantiated:
  - **Instantiate** a variable
  - **Propagate** modifications until a fix point
  - In case of contradiction, **backtrack** in the search tree

Example: $x_1 \neq x_2, \quad x_2 \neq x_3, \quad x_1 \neq x_3$ with $x_1 \in \{1,2,3\}, \quad x_2 \in \{1,3\}, \quad x_3 \in \{1,3\}$



Motivation - Constraint Programming

Global Constraints

Constraints which generally filters more values than their equivalent semantic decomposition.

Motivation - Constraint Programming

Global Constraints

Constraints which generally filters more values than their equivalent semantic decomposition.

Example: AllDifferent$(x_1, x_2, x_3) \Leftrightarrow x_1 \neq x_2, \quad x_2 \neq x_3, \quad x_1 \neq x_3$ but:

Domains	$D(x_1)$	$D(x_2)$	$D(x_3)$
Initial	{1,2,3}	{1,3}	{1,3}
After semantic decomposition propagation	{1,2,3}	{1,3}	{1,3}
After AllDifferent propagation	**{2}**	{1,3}	{1,3}

Motivation - Constraint Programming

Global Constraints

Constraints which generally filters more values than their equivalent semantic decomposition.

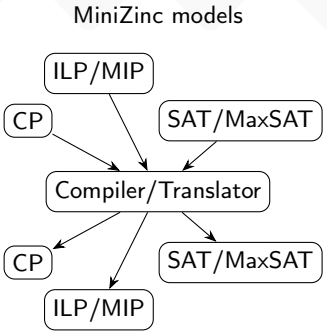
Example: AllDifferent$(x_1, x_2, x_3) \Leftrightarrow x_1 \neq x_2, \quad x_2 \neq x_3, \quad x_1 \neq x_3$ but:

Domains	$D(x_1)$	$D(x_2)$	$D(x_3)$
Initial	{1,2,3}	{1,3}	{1,3}
After semantic decomposition propagation	{1,2,3}	{1,3}	{1,3}
After AllDifferent propagation	**{2}**	{1,3}	{1,3}

Lots of global constraints: Element, Count, Among, Table, Cumulative... Up to you to create your own propagator!

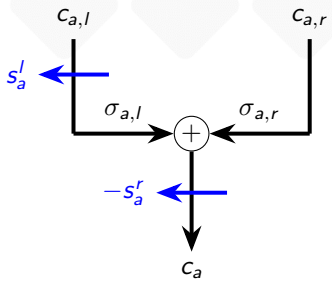
Motivation - Minizinc

- Unified Modeling Language:
  - Write your problem once in Minizinc, using CP or ILP/MIP or SAT/MaxSAT
  - Chose your CP or ILP/MIP or SAT/MaxSAT backend solver
- Easy way to test if other exact approaches are promising
- Widely used in international competitions (allows benchmarking and comparison across different solver technologies)



CP Model

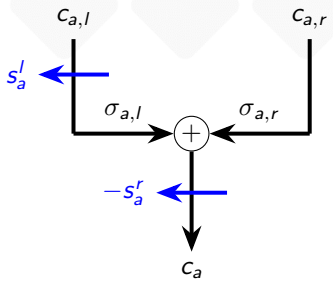
Constraints - Adder modeling



$$c_a = 2^{-s_a^r}((-1)^{\sigma_{a,l}} 2^{s_a^l} c_{a,l} + (-1)^{\sigma_{a,r}} c_{a,r})$$

Left shift only on the left input or
(exclusive) right shift on both inputs
[Dempster, 1994]

Constraints - Adder modeling



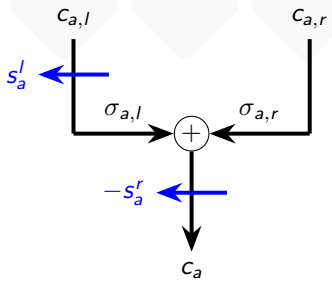
$$c_a = 2^{-s_a^r}((-1)^{\sigma_{a,l}} 2^{s_a^l} c_{a,l} + (-1)^{\sigma_{a,r}} c_{a,r})$$

Left shift only on the left input or (exclusive) right shift on both inputs [Dempster, 1994]

ILP: $\underbrace{2^{s_a^r} c_a}_{c_a^{nsh}} = \underbrace{(-1)^{\sigma_{a,l}} \underbrace{2^{s_a^l} c_{a,l}}_{c_{a,l}^{sh}}}_{c_{a,l}^{sh,sg}} + \underbrace{(-1)^{\sigma_{a,r}} \underbrace{c_{a,r}}_{c_{a,r}^{sh}}}_{c_{a,r}^{sh,sg}}$

$$\begin{cases} c_a^{nsh} = 2^s c_a \text{ if } \Psi_{a,s} = 1 & \text{w+1 var} \quad \text{2w+1 cstr} \\ c_{a,l}^{sh} = 2^s c_{a,l} \text{ if } \Phi_{a,s} = 1 & \text{w+1 var} \quad \text{2w+1 cstr} \\ c_{a,i}^{sh,sg} = -c_{a,i}^{sh} \text{ iff } \sigma_{a,i} = 1 & \text{4 var} \quad \text{4 cstr} \end{cases}$$

Constraints - Adder modeling



$$c_a = 2^{-s_a^r}((-1)^{\sigma_{a,l}} 2^{s_a^l} c_{a,l} + (-1)^{\sigma_{a,r}} c_{a,r})$$

Left shift only on the left input or (exclusive) right shift on both inputs [Dempster, 1994]

ILP: $\underbrace{2^{s_a^r} c_a}_{c_a^{nsh}} = \underbrace{(-1)^{\sigma_{a,l}} \underbrace{2^{s_a^l} c_{a,l}}_{c_{a,l}^{sh}}}_{c_{a,l}^{sh,sg}} + \underbrace{(-1)^{\sigma_{a,r}} \underbrace{c_{a,r}}_{c_{a,r}^{sh}}}_{c_{a,r}^{sh,sg}}$

$$\begin{cases} c_a^{nsh} = 2^s c_a \text{ if } \Psi_{a,s} = 1 & \text{w+1 var} \quad \text{2w+1 cstr} \\ c_{a,l}^{sh} = 2^s c_{a,l} \text{ if } \Phi_{a,s} = 1 & \text{w+1 var} \quad \text{2w+1 cstr} \\ c_{a,i}^{sh,sg} = -c_{a,i}^{sh} \text{ iff } \sigma_{a,i} = 1 & \text{4 var} \quad \text{4 cstr} \end{cases}$$

CP: $\underbrace{\underbrace{2^{s_a^r}}_{k_a} c_a}_{c_a^{nsh}} = \underbrace{\underbrace{(-1)^{\sigma_{a,l}} 2^{s_a^l}}_{k_{a,l}} c_{a,l}}_{c_{a,l}^{sh,sg}} + \underbrace{\underbrace{(-1)^{\sigma_{a,r}}}_{k_{a,r}} c_{a,r}}_{c_{a,r}^{sh,sg}}$

$$\begin{cases} c_a^{nsh} = k_a \times c_a & \text{2 var} \quad \text{1 cstr} \\ c_{a,i}^{sh,sg} = k_{a,i} \times c_{a,i} & \text{4 var} \quad \text{2 cstr} \end{cases}$$

Constraints - Reducing space search

Dempster theorems (1994) [Dempster, 1994]:

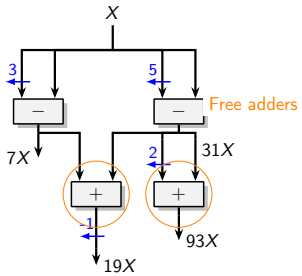
- Positiveness of fundamentals
- Left shift only on the left input or (exclusive) right shift on both inputs
- Oddness of fundamentals

Definition (Free adders)

Free adders are not connected to any other adder.

Theorem (Free target constant)

In optimal adder graphs,
$\forall\ a \in [\![1, N]\!]$,
a *is a free adder* $\Rightarrow$
c_a *is a target constant*



CP model - MCM problem

Data and variables

$N \in \mathbb{N}$: number of adders; $w \in \mathbb{N}$: fundamentals word length; $T \subset \mathbb{N}$: target constants set;

$\forall\ a \in [\![0, N]\!],\ c_a \in [\![1, 2^w - 1]\!]$: fundamental of adder a with $c_0 = 1$;

$\forall\ a \in [\![1, N]\!],\ c_a^{nsh} \in [\![0, 2^{w+1}]\!]$: value of adder a before negative shift;

$\forall\ a \in [\![1, N]\!],\ c_{a,l},\ c_{a,r} \in [\![1, 2^w - 1]\!]$: value of input i of adder a;

$\forall\ a \in [\![1, N]\!],\ c_{a,l}^{sh,sg},\ c_{a,r}^{sh,sg} \in [\![-2^{w+1}, 2^{w+1}]\!]$: signed / shifted value from left or right input of adder a;

$\forall\ a \in [\![1, N]\!],\ k_{a,l} \in \{-2^s, 2^s \mid s \in [\![0, w]\!]\}$: value of sign and left shift applied to $c_{a,l}$;

$\forall\ a \in [\![1, N]\!],\ k_{a,r} \in \{-1, 1\}$: value of sign applied to $c_{a,r}$;

$\forall\ a \in [\![1, N]\!],\ k_a \in \{-2^s, 2^s \mid s \in [\![0, w]\!]\}$: value of right shift applied to c_a^{nsh};

$\forall\ a \in [\![1, N]\!],\ ind_{a,l},\ ind_{a,r} \in [\![0, a-1]\!]$: index of left or right input of adder a;

$\forall\ a \in [\![1, N]\!],\ l_a \in [\![0, N-a]\!]$: number of adder using c_a as an input;

(a) Data / Variables of the CP model and their meaning

C1: $\forall\ a \in [\![1, N]\!],\ k_a \times c_a = c_a^{nsh}$

C2: $\forall\ a \in [\![1, N]\!], \begin{cases} k_{a,l} \times c_{a,l} = c_{a,l}^{sh,sg} \\ k_{a,r} \times c_{a,r} = c_{a,r}^{sh,sg} \end{cases}$

C3: $\forall\ a \in [\![1, N]\!],\ c_{a,l}^{sh,sg} + c_{a,r}^{sh,sg} = c_a^{nsh}$

C4: $\forall\ a \in [\![1, N]\!],\ c_{a,l} = c_{ind_{a,l}} \ \wedge\ c_{a,r} = c_{ind_{a,r}}$

C5: $T \subseteq \{c_a \mid a \in [\![1, N]\!]\}$

C6: $c_0 = 1$

C7: $\text{allDifferent}((c_a)_{a \in [\![0,N]\!]})$

C8: $\forall\ a \in [\![1, N]\!],\ k_{a,l} > 0\ \vee\ k_{a,r} > 0$

C9: $\forall\ a \in [\![1, N]\!], \begin{cases} k_a + k_{a,l} \neq 2 \\ k_a + k_{a,l} \neq 0 \end{cases}$

C10: $\forall\ a \in [\![1, N]\!],\ c_a \equiv 1 (mode\ 2)$

C11: $\forall\ a \in [\![1, N]\!],\ (\nexists\ a' \in [\![a+1, N]\!] \mid ind_{a',l} = a \ \vee\ ind_{a',r} = a) \implies c_a \in T$

(b) CP model for the Decision problem, when the number of adders is fixed

ILP model - MCM problem

Constants/Variables and their meaning
$N \in \mathbb{N}$: number of adders;
$C \in \mathbb{N}^{N_O}$: set of odd target constants;
$w \in \mathbb{N}$: fundamentals' word length;
$c_a \in [\![0; 2^w]\!], \forall a \in [\![0; N]\!]$: fundamental, or constant, obtained in adder a with c_0 fixed to the value 1, corresponding to the input;
$c_a^{\text{nsh}} \in [\![0; 2^{w+1}]\!], \forall a \in [\![1; N]\!]$: constant obtained in adder a before the negative shift;
$c_a^{\text{odd}} \in \mathbb{N}, \forall a \in [\![1; N]\!]$: variable used to ensure that c_a is odd;
$c_{a,i} \in [\![0; 2^w]\!], \forall a \in [\![1; N]\!], i \in \{l, r\}$: constant of adder from input i before adder a;
$c_{a,l}^{\text{sh}} \in [\![0; 2^{w+1}]\!], \forall a \in [\![1; N]\!]$: constant of adder from left input before adder a and after the left shift; for simplification $c_{a,r}^{\text{sh}}$ is an alias of $c_{a,r}$;
$c_{a,i}^{\text{sh,sg}} \in [\![-2^{w+1}; 2^{w+1}]\!], \forall a \in [\![1; N]\!],\ i \in \{l, r\}$: signed constant of adder from input i before adder a and after the shift;
$\sigma_{a,i} \in \{0, 1\}, \forall a \in [\![1; N]\!], i \in \{l, r\}$: sign of i input of adder a. 0 for + and 1 for −;
$c_{a,i,k} \in \{0, 1\}, \forall a \in [\![1; N]\!], i \in \{l, r\}, \forall k \in [\![0; a-1]\!]$: 1 if input i of adder a is adder k;
$\Phi_{a,s} \in \{0, 1\}, \forall a \in [\![1; N]\!], s \in [\![0; w]\!]$: 1 if left shift before adder a is equal to s;
$\Psi_{a,s} \in \{0, 1\}, \forall a \in [\![1; N]\!], s \in [\![0; w]\!]$: 1 if negative shift of adder a is equal to s;
$o_{a,j} \in \{0, 1\}, \forall a \in [\![1; N]\!], j \in [\![1; N_O]\!]$: 1 if adder a is equal to the j-th target constant.

(a) Data / Variables of the ILP model and their meaning

$$c_0 = 1 \tag{C1.1}$$

$$c_a^{\text{nsh}} = c_{a,l}^{\text{sh,sg}} + c_{a,r}^{\text{sh,sg}} \qquad \forall a \in [\![1; N]\!] \tag{C1.2}$$

$$c_a^{\text{nsh}} = 2^{s_a} c_a \quad \text{if } \Psi_{a,s} = 1 \qquad \forall a \in [\![1; N]\!], s \in [\![0; w]\!] \tag{C1.3}$$

$$\sum_{s=0}^{w} \Psi_{a,s} = 1 \qquad \forall a \in [\![1; N]\!] \tag{C1.4}$$

$$\Phi_{a,0} + \Psi_{a,s} = 1 \qquad \forall a \in [\![1; N]\!] \tag{C1.5}$$

$$c_a = 2c_a^{\text{odd}} + 1 \qquad \forall a \in [\![1; N]\!] \tag{C1.6}$$

$$c_{a,i} = c_k \quad \text{if } c_{a,i,k} = 1 \qquad \forall a \in [\![1; N]\!], i \in \{l, r\}, k \in [\![0; a-1]\!] \tag{C1.7}$$

$$\sum_{k=0}^{a-1} c_{a,i,k} = 1 \qquad \forall a \in [\![1; N]\!], i \in \{l, r\} \tag{C1.8}$$

$$c_{a,l}^{\text{sh}} = 2^s c_{a,l} \quad \text{if } \Phi_{a,s} = 1 \qquad \forall a \in [\![1; N]\!], s \in [\![0; w]\!] \tag{C1.9}$$

$$\sum_{s=0}^{w} \Phi_{a,s} = 1 \qquad \forall a \in [\![1; N]\!] \tag{C1.10}$$

$$c_{a,i}^{\text{sh,sg}} = -c_{a,i}^{\text{sh}} \quad \text{if } \sigma_{a,i} = 1 \qquad \forall a \in [\![1; N]\!], i \in \{l, r\} \tag{C1.11}$$

$$c_{a,i}^{\text{sh,sg}} = c_{a,i}^{\text{sh}} \quad \text{if } \sigma_{a,i} = 0 \qquad \forall a \in [\![1; N]\!], i \in \{l, r\} \tag{C1.12}$$

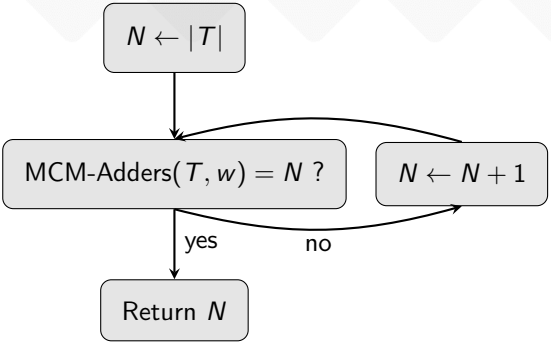
$$\sigma_{a,l} + \sigma_{a,r} \leq 1 \qquad \forall a \in [\![1; N]\!] \tag{C1.13}$$

$$c_a = C_j \quad \text{if } o_{a,j} = 1 \qquad \forall a \in [\![0; N]\!], j \in [\![1; |C|]\!] \tag{C1.14}$$

$$\sum_{a=0}^{N} o_{a,j} = 1 \qquad \forall j \in [\![1; |C|]\!] \tag{C1.15}$$

(b) ILP model for the Decision problem, when the number of adders is fixed to N

From decision to optimization



Outer loop process

Outer loop process: Transforming minimization into a succession of decision problems

Implementations and experiments

Our implementations:

- CP – Choco: outer-loop CP model implemented in Java with Choco-solver
- CP – PicatSAT: outer-loop model implemented in MiniZinc (CP) + PicatSAT + KisSAT

Comparisons with:

- ILP – [Garcia, 2023]: minimization ILP model implemented in Julia with Gurobi
- SAT – [Fiege, 2024]: outer-loop SAT model implemented in C++ with Cadical

Implementations and experiments

Our implementations:

- CP – Choco: outer-loop CP model implemented in Java with Choco-solver
- CP – PicatSAT: outer-loop model implemented in MiniZinc (CP) + PicatSAT + KisSAT

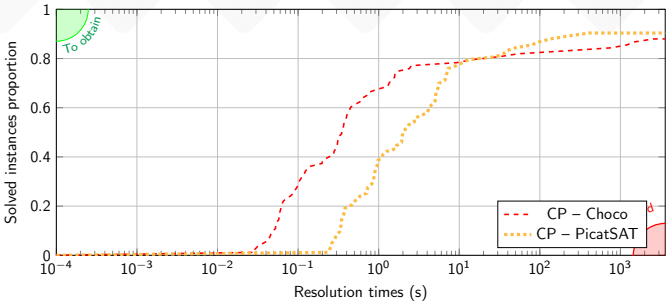
Comparisons with:

- ILP – [Garcia, 2023]: minimization ILP model implemented in Julia with Gurobi
- SAT – [Fiege, 2024]: outer-loop SAT model implemented in C++ with Cadical

- Experiments on Intel® Xeon® Gold 6444Y (16 cores, 32 threads, 45 MB L3 cache) with 256 GB RAM
- Widely-used benchmark extracted from a collection of linear digital filters that can be implemented with MCM [Nanyang Technological University, 2023]:
  - 83 instances
  - $1 \leq |T| \leq 51$
  - $4 \leq w \leq 19$

Results

Performance results - MCM problem

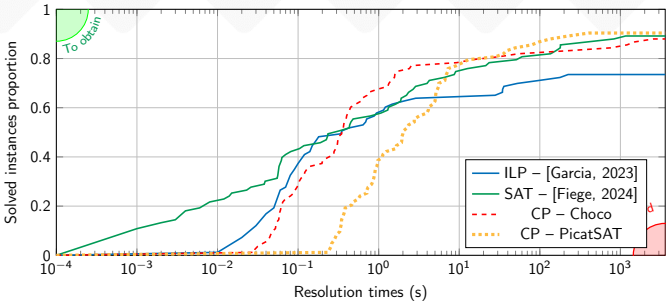


Distribution of the solved instances proportion through time

- Time-out set to 1h

Type	Solved
CP – Choco	88%
CP – PicatSAT	90%

Performance results - MCM problem



Distribution of the solved instances proportion through time

- Time-out set to 1h

Type	Solved
CP – Choco	88%
CP – PicatSAT	90%
ILP – [Garcia, 2023]	68%
SAT – [Fiege, 2024]	89%

Pseudo-polynomial-time preprocessing step

Problem: T-OPT

Input: Target constant set T and word-length w

Question: Do we have MCM-Adders$(T, w) = |T|$?

Theorem (T-OPT)

T-OPT can be solved with a pseudo-polynomial algorithm in $O(|T|^3 \log(w))$ (proof in paper).

Pseudo-polynomial-time preprocessing step

Problem: T-OPT

Input: Target constant set T and word-length w

Question: Do we have MCM-Adders$(T, w) = |T|$?

Theorem (T-OPT)

T-OPT can be solved with a pseudo-polynomial algorithm in $O(|T|^3 \log(w))$ (proof in paper).

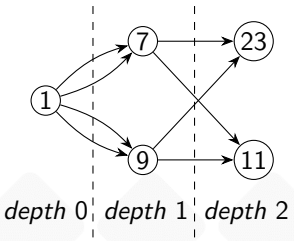
Consequence $\Rightarrow$ preprocessing step:

- If T-OPT$(T, w) =$ *true*, then stop.
- If T-OPT$(T, w) =$ *false*, then launch CP model with constraint MCM-Adders$(T, w) > |T|$.

T-OPT is true for 44 out of the 83 MCM instances of the benchmark!

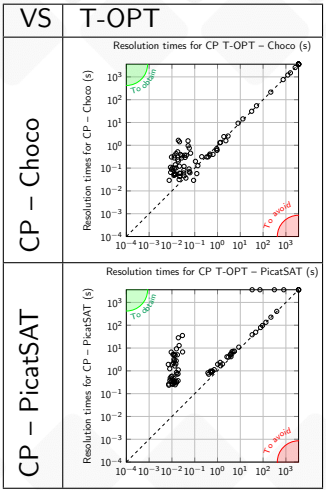
Polynomial-time preprocessing step

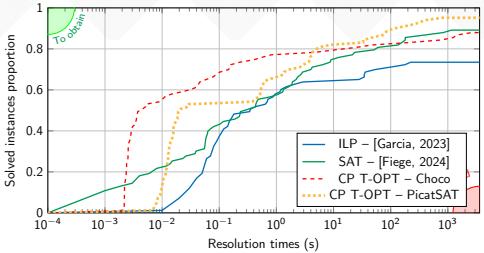
$$\text{can}(x, y, z) = \left(\begin{array}{c|l} \exists\, (\sigma_1, \sigma_2) \in \{0,1\}^2 \setminus (1,1) & z = odd((-1)^{\sigma_1} x + (-1)^{\sigma_2} y) \\ & \text{or } (-1)^{\sigma_1} x = odd(z - (-1)^{\sigma_2} y) \\ & \text{or } (-1)^{\sigma_2} y = odd(z - (-1)^{\sigma_1} x) \end{array} \right)$$




```
1  function MCM-Adders_pseudo-poly(T)
2      Let R = {1} and W = {1}
3      while W ≠ ∅ do
4          Pop x from W
5          forall z ∈ T \ R do
6              forall y ∈ R \ W do
7                  if can(x, y, z) then
8                      prec(z) ← (x, y)
9                      R ← R ∪ {z}
10                     W ← W ∪ {z}
11                     Break
12         if R = T ∪ {1} then
13             return (True, prec)
14     return (False, ∅)
```

Performance results with preprocessing step - MCM problem





Distribution of the solved instances proportion through time

Our models are very competitive with state-of-the-art!

Conclusion

Conclusion

Takeaway

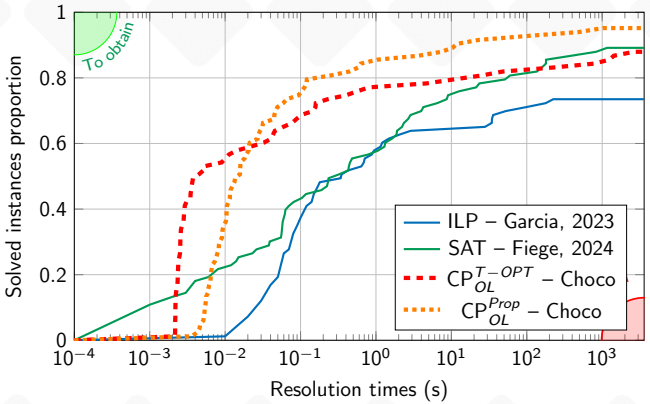
- On many combinatorial problems, modeling with ILP is very fastidious in comparison with CP
- Very constrained specific cases lead to easy resolution

Further work and perspectives

- Formalization of the $\mathcal{NP}$-complexity of MCM
- Construction of dedicated CP propagators
- Handling of very large target constants through subdivision into smaller targets
- Extensive performance tuning to assess and improve scalability with respect to word-length

Further work - Propagator

- Arithmetic constraints in CP are slow, using the operator can is faster
- Generalization of the preprocessing step
- Implementation of a propagator



Distribution of the solved instances proportion through time

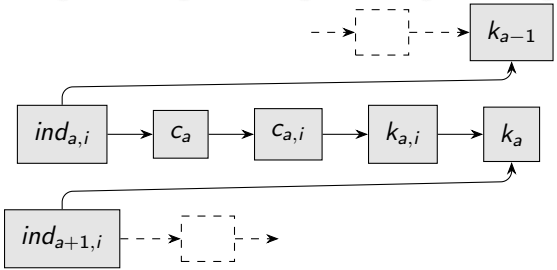
Conclusion

Thank you for your attention!

Any questions?

Appendix 1 - CP model - Search strategy

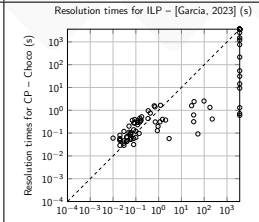
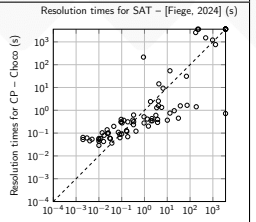
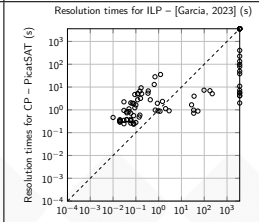
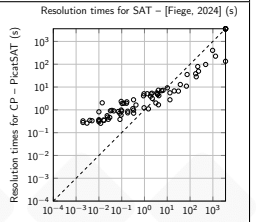
- Random exploration of the search space rarely produces valid solutions. Fixing adder 2 before adder 1 $\longrightarrow$ failure
- Solution: constructing the adder graph incrementally
- Lower Bound First strategy



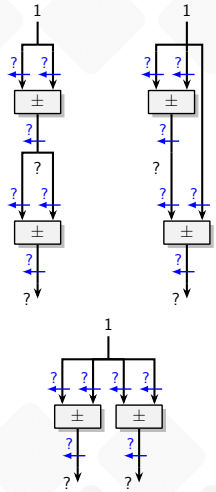
Appendix 2 - Comparison with state-of-the-art solvers

	ILP – [Garcia, 2023]	CP – Choco
Minimization	Integrated into the objective function	Outer loop
Intermediate results	Yes	No
Number of variables	$\sim (10 + N + w + \lvert T \rvert)N$	$\sim 12N$
Number of constraints	$\sim (10 + 2N + 4w + 2\lvert T \rvert)N + \lvert T \rvert$	$\sim 13N$

Appendix 3 - Performance results

VS	ILP – [Garcia, 2023]	SAT – [Fiege, 2024]
CP – Choco		
CP – PicatSAT		

Appendix 4 - Motivation - Fixing the topology



Interesting results but can we speed up even further computations?

- Fixing the topology make MCM "easier" (still NP-complete)
- **Idea:** testing the MCM problem with fixed topology with every adder graph topology

```
1 function MCM-Adders(T)
2     Let n = |T|
3     while true do
4         forall adder graph G with n vertices do
5             if MCM-Adders-FixedTopology(T, w, G) = true then
6                 return G
7         n ← n + 1
```

Appendix - References

Aksoy, Levent et al. "Optimization of gate-level area in high throughput Multiple Constant Multiplications". In: *2011 20th European Conference on Circuit Theory and Design (ECCTD)*. 2011, pp. 588–591. DOI: 10.1109/ECCTD.2011.6043602.

Dempster, Andrew G. and Malcolm D. Macleod. "Constant Integer Multiplication Using Minimum Adders". In: *IEE Proceedings - Circuits, Devices and Systems* 141.5 (Oct. 1994), pp. 407–413.

Fiege, Nicolai, Martin Kumm, and Peter Zipf. "Bit-Level Optimized Constant Multiplication Using Boolean Satisfiability". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 71.1 (2024), pp. 249–261. DOI: 10.1109/TCSI.2023.3327814.

Garcia, Rémi et al. "Hardware-Aware Design of Multiplierless Second-Order IIR Filters With Minimum Adders". In: *IEEE Transactions on Signal Processing* 70 (2022), pp. 1673–1686. DOI: 10.1109/TSP.2022.3161158.

Appendix - References

Garcia, Rémi et al. "Hardware-Aware Training for Multiplierless Convolutional Neural Networks". In: *ARITH 2025 - 32nd IEEE International Symposium on Computer Arithmetic*. El Paso, United States: IEEE, May 2025, pp. 1–8.

Garcia, Rémi and Anastasia Volkova. "Toward the Multiple Constant Multiplication at Minimal Hardware Cost". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* (2023), pp. 1–13. DOI: 10.1109/TCSI.2023.3241859. HAL: hal-03784625v3.

Ghissoni, Sidinei et al. "Radix-2 Decimation in Time (DIT) FFT implementation based on a Matrix-Multiple Constant multiplication approach". In: *2010 17th IEEE International Conference on Electronics, Circuits and Systems*. 2010, pp. 859–862. DOI: 10.1109/ICECS.2010.5724648.

Appendix - References

Habermann, Tobias et al. "Hardware-Aware Quantization for Multiplierless Neural Network Controllers". In: *2022 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. 2022, pp. 541–545. DOI: 10.1109/APCCAS55924.2022.10090271.

Howard, Charles D., Linda S. DeBrunner, and Victor DeBrunner. "Hybrid MCM Implementation for FIR Filters in FPGA Devices". In: *2014 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE, June 2014, pp. 1–6. DOI: 10.1109/EIT.2014.6871750.

Kumm, Martin. "Optimal Constant Multiplication Using Integer Linear Programming". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 65.5 (May 2018), pp. 567–571. DOI: 10.1109/TCSII.2017.2772922.

Kumm, Martin, Anastasia Volkova, and Silviu-Ioan Filip. "Design of Optimal Multiplierless FIR Filters With Minimal Number of Adders". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42.2 (2023), pp. 658–671. DOI: 10.1109/TCAD.2022.3179221.

Appendix - References

Meher, Pramod Kumar and Yu Pan. "MCM-Based Implementation of Block FIR Filters for High-Speed and Low-Power Applications". In: *Proceedings of the IEEE/IFIP 19th International Conference on Very Large Scale Integration and System-on-Chip (VLSI-SoC)*. IEEE, Oct. 2011, pp. 118–121. DOI: 10.1109/VLSISoC.2011.6081653.

Nanyang Technological University, Singapore. *FIRsuite: Suite of Constant Coefficient FIR Filters*.
https://www.firsuite.net/FIR/FromPublication. Nov. 2023.

Wendt, James B., Nathaniel A. Conos, and Miodrag Potkonjak. "Multiple Constant Multiplication Implementations in Near-Threshold Computing Systems". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Apr. 2015, pp. 1071–1075. DOI: 10.1109/ICASSP.2015.7178134.