Time-Memory Analysis for Parallel Collision Search Algorithms

Monika Trimoska Sorina Ionica Gilles Dequen

MIS, University of Picardie Jules Verne

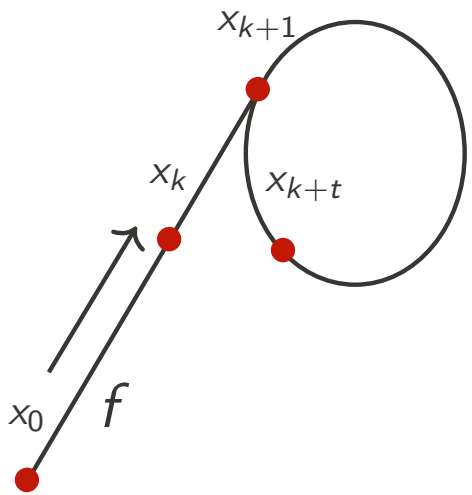
CHES 2021
13 September 2021

Collision search

Collision

Given a random map $f : S \to S$ on a finite set S of cardinality N, we call collision any pair R, R' of elements in S such that $f(R) = f(R')$.

Pollard's Rho method



- Ideally, f is a random mapping.
- Expected number of steps until the collision is found:

$$\sqrt{\frac{\pi N}{2}}.$$

Collision search applications

One collision application

- (Elliptic Curve) Discrete Logarithm Problem.

Multi-collision applications

- Attack on the 3-DES with three independent keys.
- (EC)DLP in the multi-user setting.
- Computational Supersingular Isogeny Problem.

Collision search applications

One collision application

- (Elliptic Curve) Discrete Logarithm Problem.

Multi-collision applications

- Attack on the 3-DES with three independent keys.
- (EC)DLP in the multi-user setting.
- Computational Supersingular Isogeny Problem.

Collision search applications (ECDLP)

ECDLP:

Let E be an elliptic curve over a finite field K. Given points $P, Q \in E(K)$, find $x \in \mathbb{Z}$, such that $xP = Q$.

Collision search applications (ECDLP)

Collision (recall)

Given a random map $f : S \to S$ on a finite set S of cardinality N, we call collision any pair R, R' of elements in S such that $f(R) = f(R')$.

$$f(R) = \begin{cases} R + P & \text{if } R \in S_1 \\ 2R & \text{if } R \in S_2 \\ R + Q & \text{if } R \in S_3, \end{cases}$$

Property of f

Input $(aP + bQ) \to$ Output $(a'P + b'Q)$.

Collision search applications (ECDLP)

Having two different linear combinations of a random point $R \in E(K)$

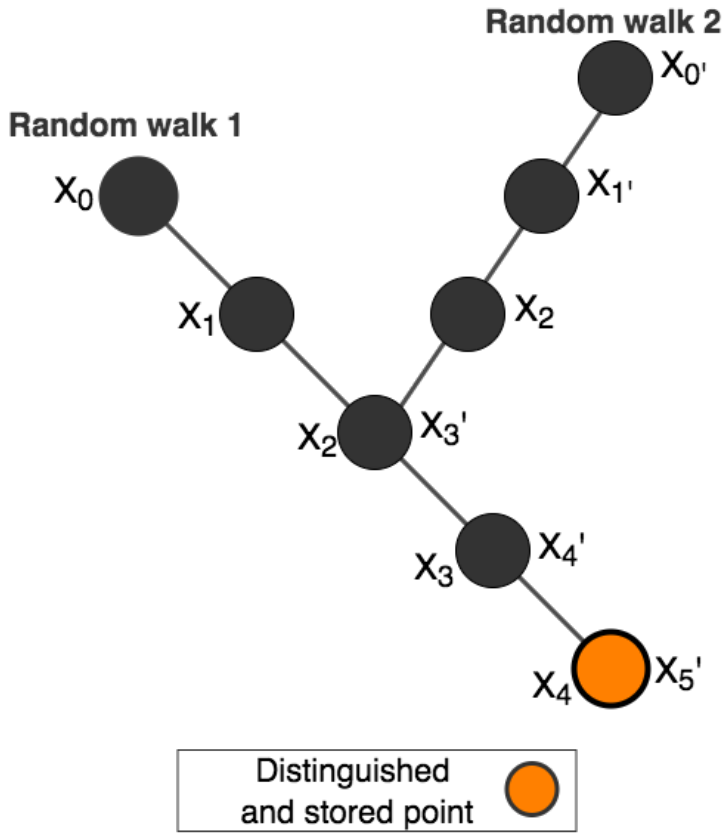
$$R = aP + bQ$$
$$R = a'P + b'Q,$$

we compute

$$x = \frac{a - a'}{b' - b} \pmod{N}.$$

Parallel Collision Search

- Proposed by van Oorschot & Wiener (1996).
- Distinguished points : a set of points having an easily testable property.
  ex. The x-coordinate has 3 trailling zero bits: 10101101000.
- Only distinguished points are stored in memory.
- θ - the proportion of distinguished points in a set S.



Time complexity analysis

1. One-collision case: revisit the proof of Van Oorschot and Wiener and extrapolate the optimal value for θ.
2. Multi-collision case: provide a more refined analysis of the running time for finding multiple collisions with a memory constraint.

Time complexity analysis

Intermediate result

Formula for the expected total number of computed distinguished points for finding m collisions:

$$S_m \approx \theta\sqrt{2mN}.$$

Time complexity analysis

Theorem. In the parallel collision search algorithm, the expected running time to find m collisions with a memory constraint of w words is:

$$\frac{1}{L}\left(\frac{w}{\theta}+(m-\frac{w^2}{2\theta^2 N})\frac{\theta N}{w}+\frac{2m}{\theta}\right).$$

L - number of used processors.
θ - proportion of distinguished points in S.
N - number of elements in S.

Time complexity analysis

Theorem. In the parallel collision search algorithm, the expected running time to find m collisions with a memory constraint of w words is:

$$\frac{1}{L}\left(\frac{w}{\theta}+\left(m-\frac{w^2}{2\theta^2 N}\right)\frac{\theta N}{w}+\frac{2m}{\theta}\right).$$

expected number of iterations needed to find and store w points

number of collisions found after storing w points

expected number of iterations needed to find one collision when w points are stored

L - number of used processors.
θ - proportion of distinguished points in S.
N - number of elements in S.

Time complexity analysis

Corollary.

The optimal proportion of distinguished points minimizing the time complexity is:

$$\theta = \frac{\sqrt{w^2 + 2Nw}}{N}.$$

Choosing this value for θ, the running time of the PCS algorithm for finding $\frac{N}{2}$ collisions is bounded by:

$$O\left(\frac{N}{L}\sqrt{1 + \frac{2N}{w}}\right).$$

Time complexity analysis

Corollary.

The optimal proportion of distinguished points minimizing the time complexity is:

$$\theta = \frac{\sqrt{w^2 + 2Nw}}{N}.$$

Choosing this value for θ, the running time of the PCS algorithm for finding $\frac{N}{2}$ collisions is bounded by:

$$O\left(\frac{N}{L}\sqrt{1 + \frac{2N}{w}}\right).$$

➔ Memory is an important factor in the running time complexity.

Data structure

Requirements

- ➔ Space efficient
- ➔ Thread-safe
- ➔ Fast look-up and insertion

Data structure

Requirements

- ➔ Space efficient
- ➔ Thread-safe
- ➔ Fast look-up and insertion

Commonly used structure:
Hash table.

Data structure

Requirements

- ➔ Space efficient
- ➔ Thread-safe
- ➔ Fast look-up and insertion

Commonly used structure:
Hash table.

Alternative:
Packed Radix-Tree-List (PRTL).

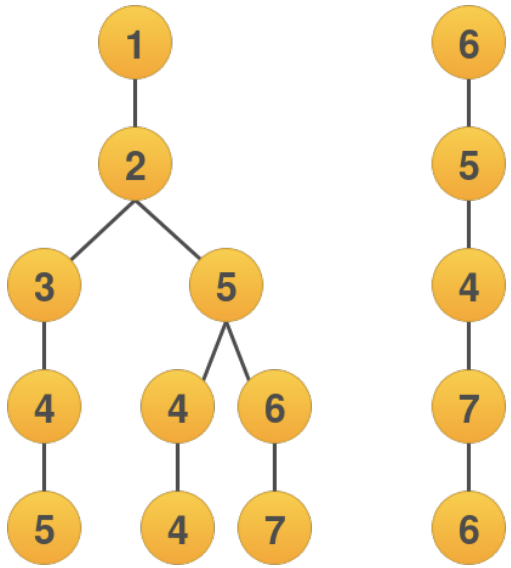


Figure: Exemple of a radix tree holding the set 12345, 12544, 12567, 65476.

Packed Radix-Tree-List

➔ Construct a radix tree up to certain level.

➔ Add the points to linked lists, each list starting from a leaf on the tree.

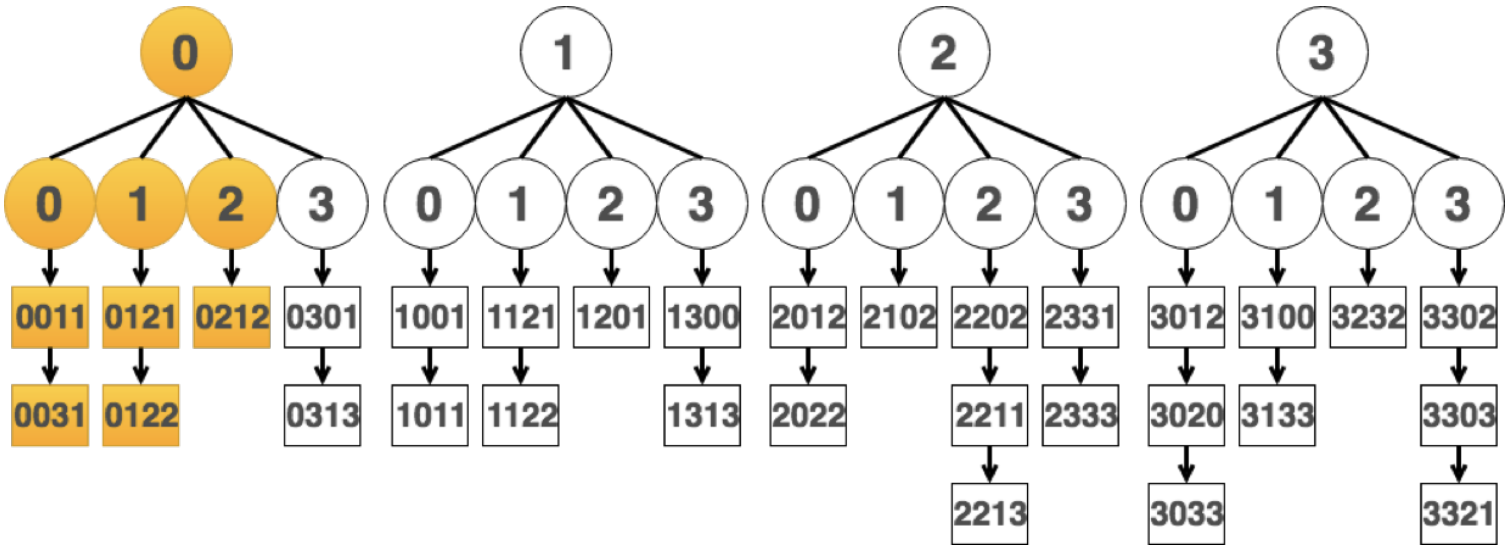
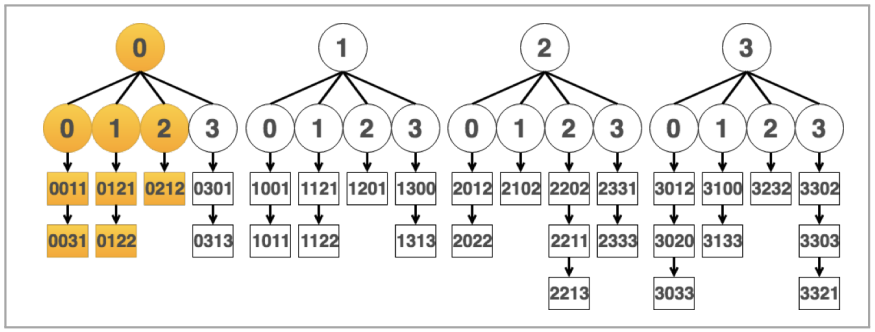
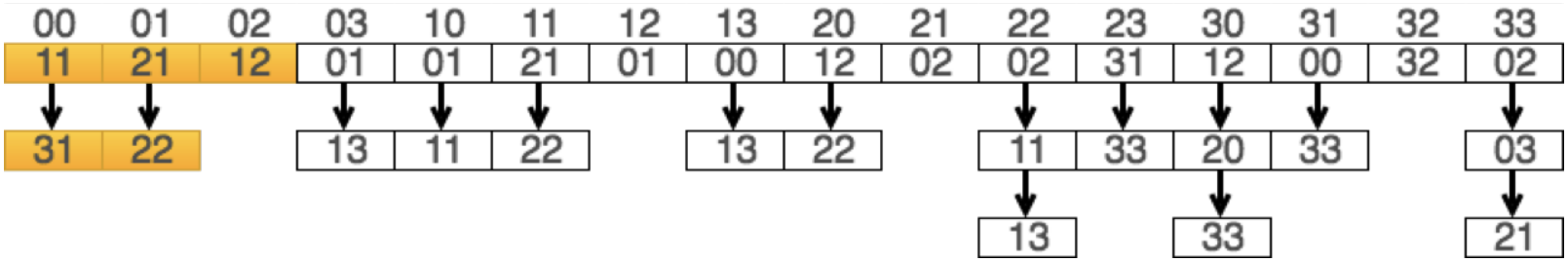


Figure: Exemple of a PRTL holding the set 0011, 0031, 0121, 0122, 0212, etc.

Packed Radix-Tree-List



PRTL implementation



- ➔ Saving space on common prefixes.
- ➔ The stored data is packed in a single vector.
- ➔ We can estimate the optimal branching level.

Optimal branching level

Find level l such that

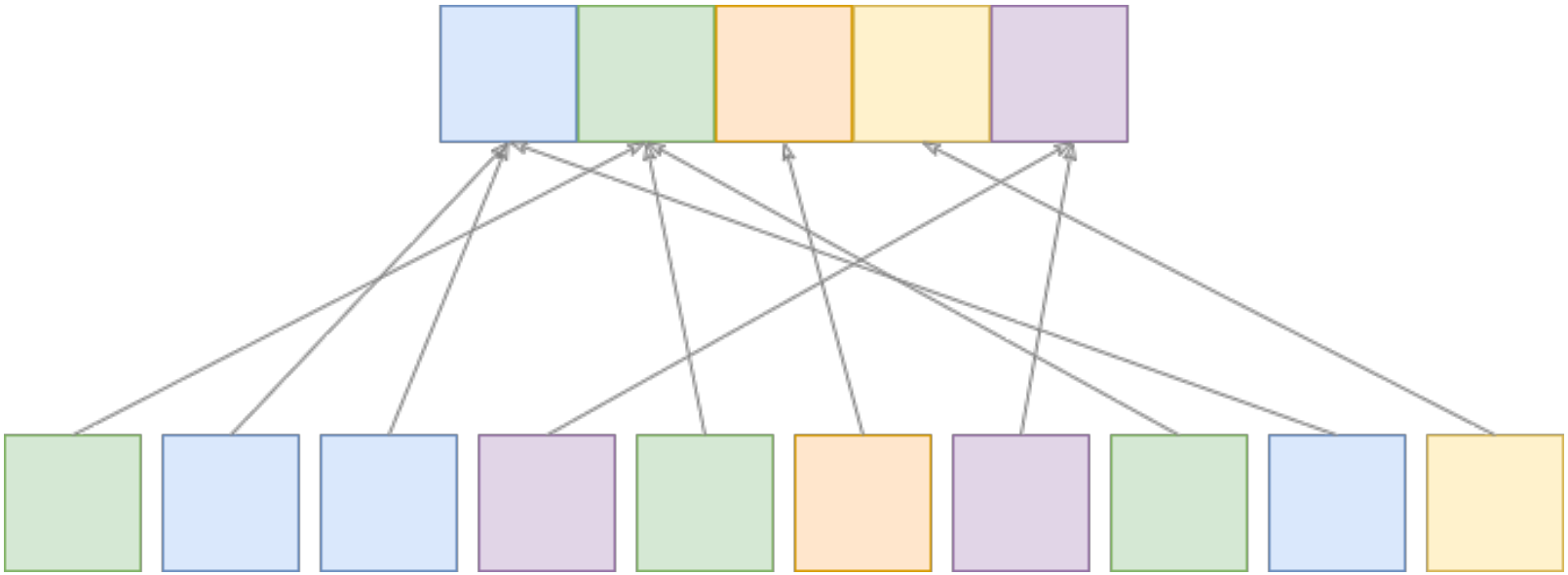
- ➔ There are no pending leaves ;
- ➔ Linked lists are as short as possible ;

Optimal branching level

Find level l such that

- ➔ There are no pending leaves ;
- ➔ Linked lists are as short as possible ;

Coupon collector's problem



Optimal branching level

Find level l such that

- ➔ There are no pending leaves ;
- ➔ Linked lists are as short as possible ;

As per the Coupon collector's problem, the optimal level is l such that

$$K \geq b^l(\ln b^l + 0.577),$$

where K is the estimated number of stored points and b is the base of their numerical representation.

Experimental results

Running a multi-collision search while limiting the available memory proves that more storage space yields a faster algorithm.

Collisions	Memory limit	Runtime		Stored points	
		PRTL	Hash table	PRTL	Hash table
4,000,000	1 GB	34.64 h	58.80 h	46,820,082	12,912,177
16,000,000	2 GB	88.18 h	137.46 h	93,640,161	25,824,345
50,000,000	4 GB	203.24 h	276.80 h	168,325,978	51,648,716

Table: Runtime for multi-collision search for a 55-bit curve using PRTLs and hash tables.

- Collision search in $E(\mathbb{F}_p)$ with p prime.
- $\theta \approx \frac{1}{2^{b/4}}$ for a b-bit curve.
- Implementation in C, using external libraries GMP and OpenMP.
- 28-core Intel Xeon E5-2640 processor.

Conclusion

- Revisited one-collision and multi-collision time complexity.
- Showed that memory is an important factor in the running time complexity.
- Proposed an alternative memory structure.

➔ Check out the artifact:
https://artifacts.iacr.org/tches/2021/a10/