Graphic Processors in Computational Applications

Part 5 – Applications

dr inż. Krzysztof Kaczmarski
2022

Materiały sponsorowane przez:

Projekt „NERW 2 PW. Nauka – Edukacja – Rozwój – Współpraca” współfinansowany jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Zadanie 10 pn. „Modyfikacja programów studiów na kierunkach prowadzonych przez Wydział Matematyki i Nauk Informacyjnych”, realizowane w ramach projektu „NERW 2 PW. Nauka – Edukacja – Rozwój – Współpraca”, współfinansowanego jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Fundusze Europejskie Wiedza Edukacja Rozwój
Rzeczpospolita Polska
Politechnika Warszawska
Unia Europejska Europejski Fundusz Społeczny

References

This material is based on several papers:

- Krzysztof Kaczmarski, Pawel Rządżewski, and Albert Wolant. Massively parallel construction of the cell graph. volume 9573 of *Lecture Notes in Computer Science*, pages 559–569. Springer, 2015
- Krzysztof Kaczmarski and Albert Wolant. Radix tree for binary sequences on GPU. volume 10777 of *Lecture Notes in Computer Science*, pages 219–231. Springer, 2017
- Krzysztof Kaczmarski and Piotr Przymus. Fixed length lightweight compression for GPU revised. *J. Parallel Distributed Comput.*, 107:19–36, 2017
- Krzysztof Kaczmarski, Pawel Rządżewski, and Albert Wolant. Parallel algorithms constructing the cell graph. *Concurr. Comput. Pract. Exp.*, 29(23), 2017
- Krzysztof Kaczmarski and Albert Wolant. GPU r-trie: Dictionary with ultra fast lookup. *Concurr. Comput. Pract. Exp.*, 31(19), 2019

Part 5 – Applications

Faculty of Mathematics and Information Science
WARSAW UNIVERSITY OF TECHNOLOGY

Map:

One thread – one read, one write

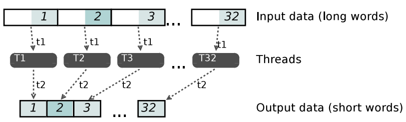
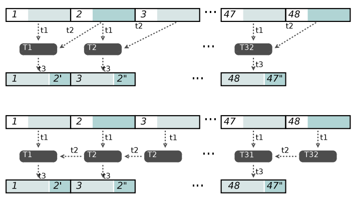


Figure: Example of Fixed Length Compression – Remove leading zeros of fixed length in each input value

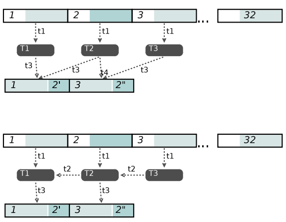
Gather:

One thread – many reads, one write



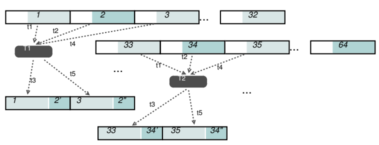
Scatter:

One thread – one read, many writes



Allgather:

One thread – many reads, many writes



Part 5 – Applications

Faculty of Mathematics and Information Science
WARSAW UNIVERSITY OF TECHNOLOGY

Compression Example

(with Piotr Przymus)

Allgather FL algorithm, compression using 3 bits encoding. 32 input values are encoded using 3 output values.

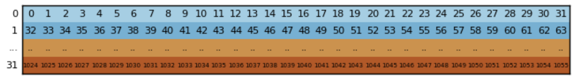


Figure: Compression – each thread reads one data row (colors denote threads, numbers indicate subsequent values in input array)

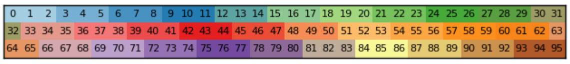


Figure: Compressed output data memory alignment (colors denote threads, numbers indicate subsequent values in the output array).

Compression Example

Allgather AFL algorithm, compression and decompression using 3 bits encoding. 32 input values are encoded using 3 output values.

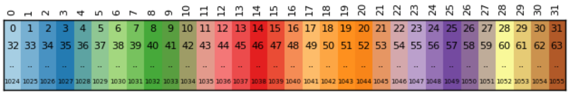


Figure: During compression each thread reads one data column (colors denote threads, numbers indicate subsequent values in input array)

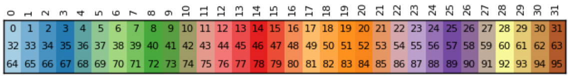


Figure: Compressed data memory alignment. During decompression each thread reads one column (colors denote threads, numbers indicate subsequent values in the output array)

Compression Example

Compression and decompression bandwidth for 1Gb of data. In each plot compression bandwidth (Gb/s) is in the upper part of the plot, and decompression bandwidth (GB/s) is in the lower part of the plot.

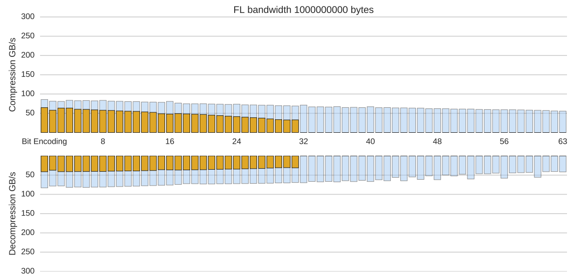


Figure: FL algorithm

Compression Example

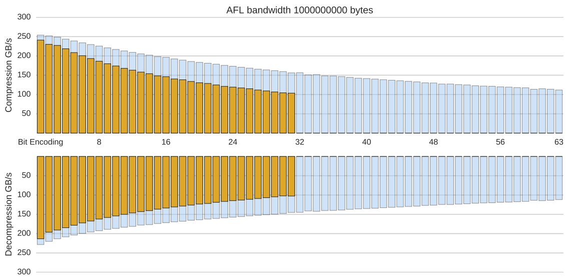


Figure: AFL algorithm

int long

Compression Example

Bandwidth of whole compress-decompress process. Measured for data already on GPU – first being compressed and then decompressed.

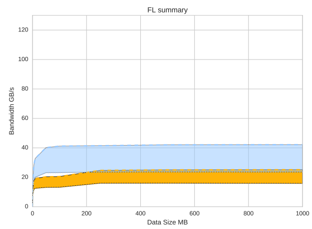


Figure: FL algorithm

int max, int min, long max, long min, int, long

Compression Example

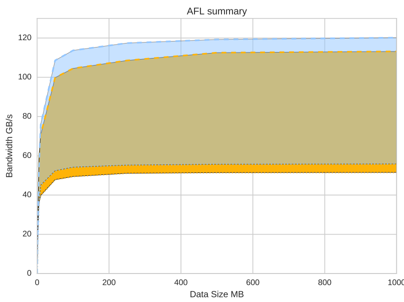


Figure: AFL algorithm

int max, int min, long max, long min, int, long

Compression Example

Bandwidth of whole compress-decompress process and in detail for compression and decompression. Measured for data already on GPU

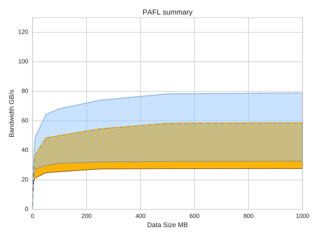
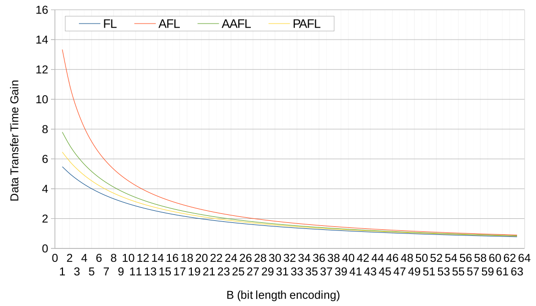


Figure: PAFL algorithm

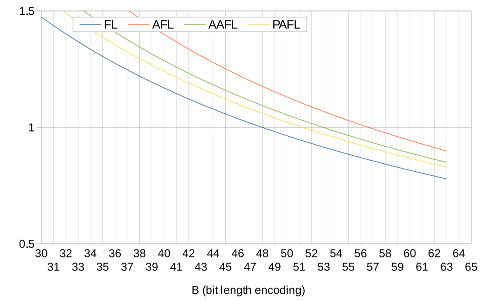
int optim., int pessim., long optim., long pessim., int, long

Compression Example



Compression Example

(zoomed)



Part 5 – Applications

Faculty of Mathematics and Information Science
WARSAW UNIVERSITY OF TECHNOLOGY

Parallel Threads Behavior

Compression Example

Simulated annealing in Monte Carlo Chromatin Spatial Modeling

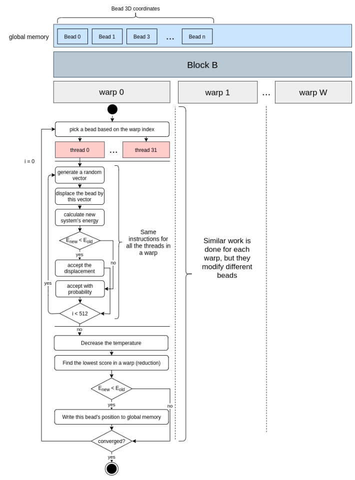
R-Trie – Retrieval Tree with variable bit stride

Tests with Longest Prefix Match problem

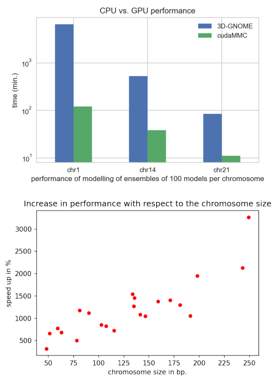
Fast Detection of Neighboring Vectors – Case Study

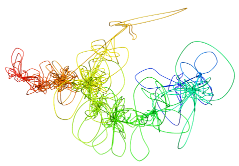
Simulated annealing

A probabilistic technique for approximating the global optimum of a given function.
Specifically, it is a metaheuristic to approximate global optimization in a large search space for an optimization problem.



Simulated annealing – results





- Michał Własnowolski, Paweł Grabowski, Damian Roszczyk, Krzysztof Kaczmarski and Dariusz Plewczyński. cudaMMC - GPU-extended Multiscale Monte Carlo Chromatin Spatial Modelling (to be submitted) 2022.

Part 5 – Applications

Faculty of Mathematics and Information Science
WARSAW UNIVERSITY OF TECHNOLOGY

Parallel Threads Behavior

Compression Example

Simulated annealing in Monte Carlo Chromatin Spatial Modeling

R-Trie – Retrieval Tree with variable bit stride

Tests with Longest Prefix Match problem

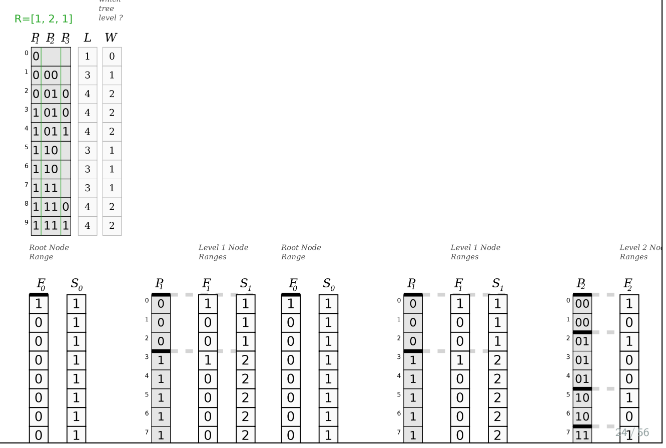
Fast Detection of Neighboring Vectors – Case Study

Introduction II – Problems

- Effective parallel tree creation
  - Optimal bit stride selection (R-sequence)
    Sequential dynamic programming alg. on binary tree.
  - Parallel allocation of tree levels
  - Compression of unused tree levels in some branches
- Parallel search procedure
- Parallel tree updates – keys deletion, keys insertion

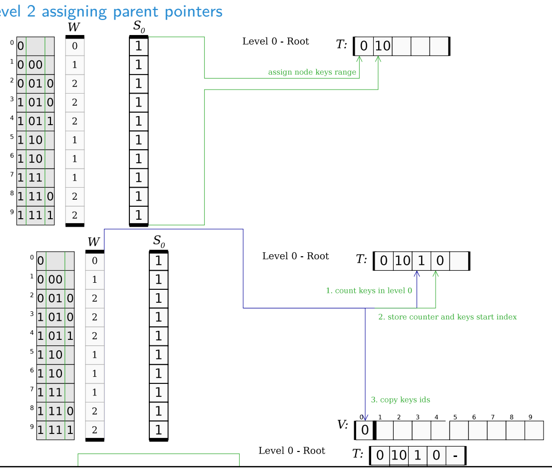
Parallel Construction Algorithm - preprocessing

Finding Node Ranges for Level 2 - Counting Children

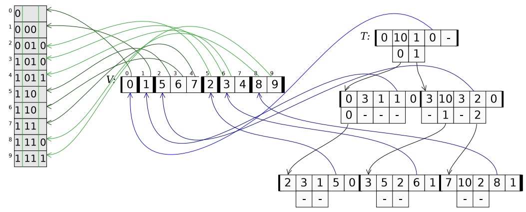


Parallel Construction Algorithm - levels allocation

Level 2 assigning parent pointers

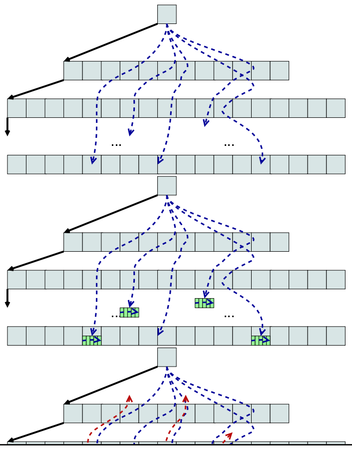


Constructed Tree



Parallel Searching Process

3.Searching Up



Part 5 – Applications

Faculty of Mathematics and Information Science

Parallel Threads Behavior

Compression Example

Simulated annealing in Monte Carlo Chromatin Spatial Modeling

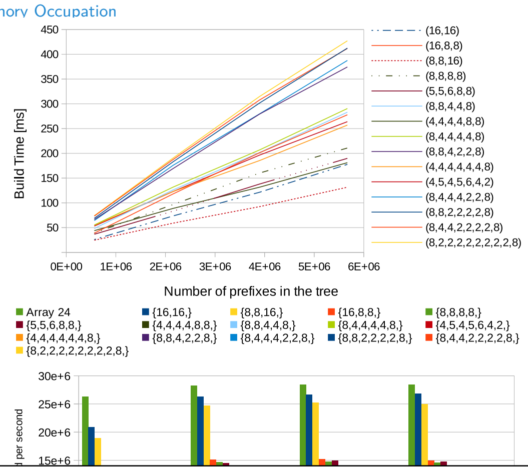
R-Trie – Retrieval Tree with variable bit stride

Tests with Longest Prefix Match problem

Fast Detection of Neighboring Vectors – Case Study

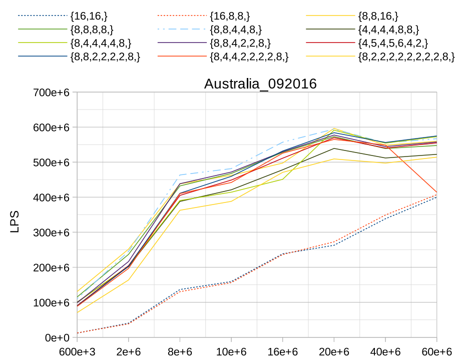
Results I – Tree Creation

Memory Occupation



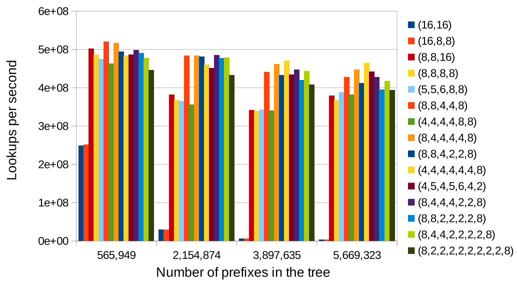
Results II – Retrieval

Lookups per second for different batch sizes



Results II – Retrieval

Lookups per second for different tree sizes



Future Works – Open Problems

- How can we find optimal R sequence?
  - sequential dynamic programming algorithm (10 years old)
  - bi-objective optimization – memory, search time
  - deep-learning
- Path compression
  - may highly improve tree size
  - but may increase branch divergence
  - no parallel build algorithm so far (but we work on it)

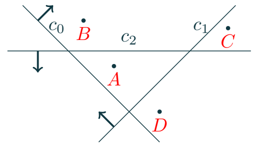
Part 5 – Applications

What are neighboring vectors?

Fast Detection of Neighboring Vectors – Case Study

Cell Graph

System of inequalities $c_0, c_1, \ldots, c_{\ell-1}$ (constraints) describes the boundaries that partition the space into a number of pairwise disjoint regions, called *cells*.



point	representation
A	111
B	110
C	100
D	101

Cells are neighboring ⇔ their Hamming distance is 1

Work Complexity and Known Algorithms

Fast Detection of Neighboring Vectors – Case Study

Cell graph construction was deeply studied by many authors[1]:

- naive algorithm improved with heuristics $O(n^2 \cdot \ell)$
  - obvious checking of all pairs
- optimized tree-based $O(n \cdot \ell^2)$
  - build RST tree of the vectors
  - for each vector: search for l possible neighbours in the tree,
- optimal tree-based with two way searching $O(n \cdot \ell)$
  - build RST tree of the vectors
  - search bottom-up and top-down finding pairs

Naive algorithm with heuristics

Fast Detection of Neighboring Vectors – Case Study

Triangle inequality:

$$\text{dist}(x, y) + \text{dist}(y, z) \geqslant \text{dist}(x, z)$$

may be transformed to

$$\text{dist}(x, y) \geqslant |\,\text{dist}(x, z) - \text{dist}(y, z)|$$

Computing all distances $\text{dist}(x_i, z)$ gives a quick negative test:

$$|\,\text{dist}(x_i, z) - \text{dist}(x_j, z)| \geqslant k \Rightarrow \text{dist}(x_i, x_j) \geqslant k$$

Naive algorithm

Complexity: $O(n^2\ell)$, n-number of vectors, ℓ-vector length

Realistic example:

- 200 inequalities
- 200k sample points

$$\frac{200}{8} \cdot 200 \cdot 10^3 \cdot 200 \cdot 10^3 = 1TB$$

Observation:

Naive and heuristic algorithms do not use information about the problem.

Heuristic algorithm parallel implementation

Fast Detection of Neighboring Vectors – Case Study

Precomputing Distances

Input: $X = \{x_0, x_1, \ldots, x_{n-1}\} \subseteq [2]^\ell, h \in \mathbb{N}$
initialize $dist(x_i, x_j) = 0$ for all $i \in [h], j \in [n]$
for $i \in [h]$ *and* $j \in \{i+1, \ldots, n-1\}$ **do in parallel (threads)**
  initialize $dist(x_i, x_j) = 0$
  for $k \leftarrow 0$ **to** $\ell - 1$ **do**
    if $x_i(k) \neq x_j(k)$ **then** $dist(x_i, x_j) \leftarrow dist(x_i, x_j) + 1$;

Heuristic algorithm parallel implementation

Fast Detection of Neighboring Vectors – Case Study

Parallel Heuristic Algorithm

Input: $X = \{x_1, x_2, \ldots, x_n\} \subseteq [2]^\ell, h \in \mathbb{N}$
$dist \leftarrow ComputeDist(X, h)$
$results \leftarrow$ vector of w zeros
for $h \leqslant i \leqslant n-1$ *and* $i < j \leqslant n-1$ **do in parallel (threads)**
  if $|\,\text{dist}(x_i, x_d) - \text{dist}(x_j, x_d)| \leqslant 1$ *for all* $d \in [h]$ **then**
    $count \leftarrow 0$
    for $k \leftarrow 0$ **to** $\ell - 1$ **do**
      if $x_i(k) \neq x_j(k)$ **then** $count \leftarrow count + 1$;
      if $count \geqslant 2$ **then Break**;
    if $count = 1$ **then output** (x_i, x_j);

Optimized Tree-based Algorithm

Basic Idea

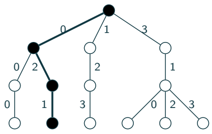
1. Build radix search tree T — $O(n \cdot \ell)$
2. For each vector v: — $O(n)$
   - 2.1 For each bit of v: — $O(\ell)$
     - 2.1.1 negate this bit and produce v' — $O(1)$
     - 2.1.2 search for the vector v' in the tree T — $O(\ell)$

Overall complexity $O(n \cdot \ell) + O(n \cdot \ell \cdot \ell) = O(n \cdot \ell^2)$

Tree-based Algorithm

Radix search tree

At each level we consider r bits of the vectors.
We get 2^r possible children of each node.

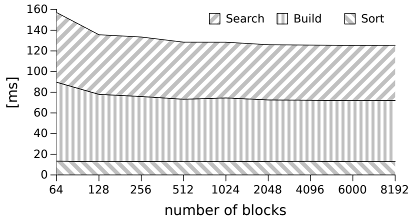


$r = 2.$

x	$\widetilde{x}$
00 00 00	000
00 10 01	**021**
01 10 11	123
11 01 00	310
11 01 10	312
11 01 11	313

Results of Experiments

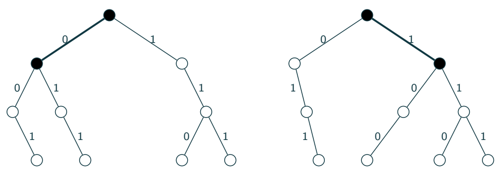
Time division of algorithm steps



Optimal Searching

Normal order and reverse order RST

$$X = \begin{array}{c|c} x_0 & 110 \\ x_1 & \mathbf{0}01 \\ x_2 & \mathbf{0}11 \\ x_3 & 111 \end{array} \qquad X' = \begin{array}{c|c} x'_0 & 011 \\ x'_1 & \mathbf{1}00 \\ x'_2 & \mathbf{1}10 \\ x'_3 & 111 \end{array}$$



Optimal Searching

XORing and scanning of consecutive vectors

$$X = \begin{array}{c|c} x_1 & 001 \\ x_2 & 011 \\ x_0 & 110 \\ x_3 & 111 \end{array} \qquad X' = \begin{array}{c|c} x'_0 & 011 \\ x'_1 & 100 \\ x'_2 & 110 \\ x'_3 & 111 \end{array}$$

$$a_i = \text{XOR}(x_i, x_{i+1}), \quad a'_i = \text{XOR}(x'_i, x'_{i+1})$$

1 Exclusive scan of rows with bitwise OR $O(n \cdot \ell)$

2 Exclusive scan of columns with arithmetic sum $O(n \cdot \ell)$

A		A'			A		A'			A		A'	
a_1	010	a'_0	111	$\xrightarrow{1}$	a_1	001	a'_0	011	$\xrightarrow{2}$	a_1	000	a'_0	000
a_2	101	a'_1	010		a_2	011	a'_1	001		a_2	001	a'_1	011
a_0	001	a'_2	001		a_0	000	a'_2	000		a_0	012	a'_2	012
a_3	000	a'_3	000		a_3	000	a'_3	000		a_3	012	a'_3	012

Optimal Searching

Finding Solution

3 Create table N of tuples (x, h, p, p') $O(n \cdot \ell)$

4 Sort it with respect to values (h, p, p'). $O(n \cdot \ell)$

A	h_0	h_1	h_2
a_1	0	0	0
a_2	0	0	1
a_0	0	1	2
a_3	0	1	2

A'	h_2	h_1	h_0
a'_0	0	0	0
a'_1	0	1	1
a'_2	0	1	2
a'_3	0	1	2

$\xrightarrow{3}$

$x\ , h, p, p'$
$x_0, 0, 0, 0$
$x_0, 1, 1, 0$
$x_0, 2, 2, 0$
$x_1, 0, 0, 1$
$x_1, 1, 0, 1$
$x_1, 2, 0, 0$
$x_2, 0, 0, 2$
$x_2, 1, 0, 1$
$x_2, 2, 1, 0$
$x_3, 0, 0, 2$
$x_3, 1, 1, 1$
$x_3, 2, 2, 0$

$\xrightarrow{4}$

sorted(N)
$x_0, 0, 0, 0$
$x_1, 0, 0, 1$
$x_2, 0, 0, 2$
$x_3, 0, 0, 2$
$x_1, 1, 0, 1$
$x_2, 1, 0, 1$
$x_0, 1, 1, 0$
$x_3, 1, 1, 1$
$x_1, 2, 0, 0$
$x_2, 2, 1, 0$
$x_0, 2, 2, 0$
$x_3, 2, 2, 0$

Optimal Searching

Finding Solution

5 Vectors are neighbors iff subsequent rows are equal $O(n \cdot \ell)$

$x_0, 0, 0, 0$
$x_1, 0, 0, 1$
$x_2, 0, 0, 2$
$x_3, 0, 0, 2$
$x_1, 1, 0, 1$
$x_2, 1, 0, 1$
$x_0, 1, 1, 0$
$x_3, 1, 1, 1$
$x_1, 2, 0, 0$
$x_2, 2, 1, 0$
$x_0, 2, 2, 0$
$x_3, 2, 2, 0$

$\rightarrow$

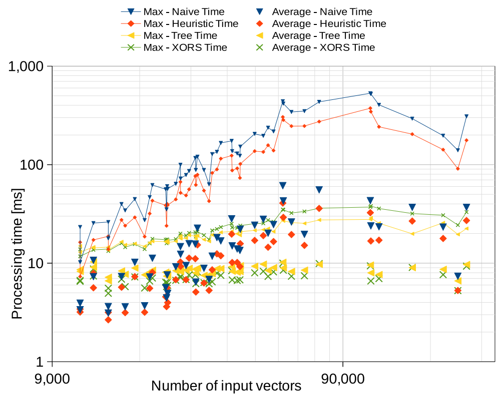
	x_0	x_1	x_2	x_3
x_0				
x_1				
x_2		1		
x_3	1		1	

x_0	110
x_1	001
x_2	011
x_3	111

The overall complexity of this algorithm is $O(n \cdot \ell)$

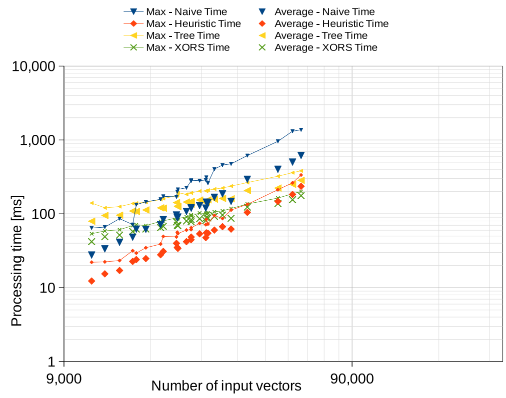
Results of Experiments

Algorithms Time Comparison: K40, vector length (ℓ):32



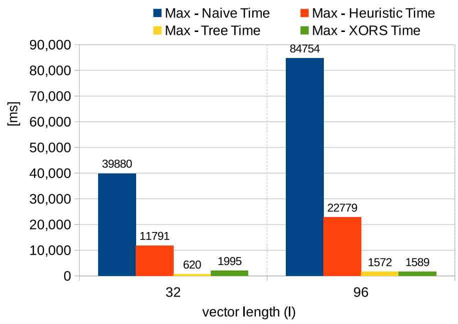
Results of Experiments

Algorithms Time Comparison: K40, vector length (ℓ):192



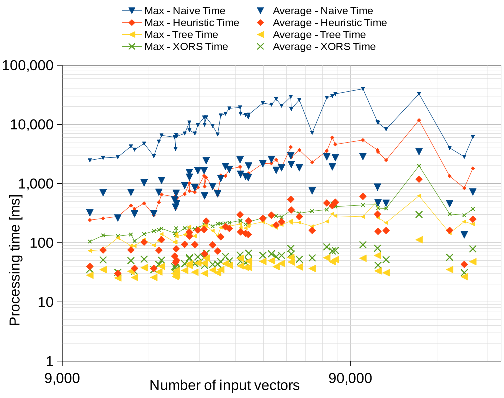
Results of Experiments

Worst Scenario: Jetson TK1, vectors (n):90k



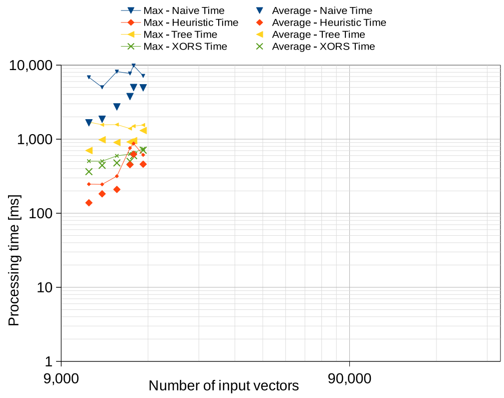
Results of Experiments

Algorithms Time Comparison: TK1, vector length (ℓ):32



Results of Experiments

Algorithms Time Comparison: TK1, vector length (ℓ):192



Tree Searching

Parallel Top-Down level by level

Input: $X = \{x_0, x_1, \ldots, x_{n-1}\} \subseteq [2]^\ell$

sort X

$T \leftarrow ConstructTree(\widetilde{X})$

for $x \in X$ **do in parallel (blocks)**

  for $k \in [\ell]$ **do in parallel (threads)**

    $x' \leftarrow x$ with the k-th bit negated

    $C \leftarrow$ the root of T

    for $h \leftarrow 0$ **to** $\ell/r - 1$ **do**

      $v \leftarrow \widetilde{x'}(h)$

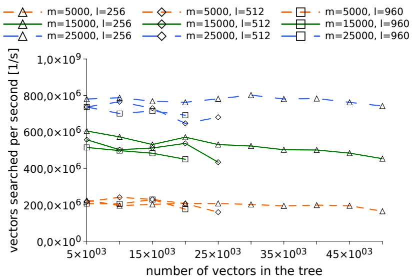
      if *there is no v-child of C* **then Exit thread**;

      $C \leftarrow v$-child of C

    output (x, x')

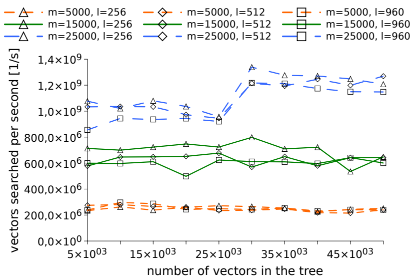
Results: Batch Dictionary Search

Uniform Tree



Results: Comparison of Different Solutions

Degenerated Tree



Bibliography

Krzysztof Kaczmarski and Piotr Przymus. Fixed length lightweight compression for GPU revised. *J. Parallel Distributed Comput.*, 107:19–36, 2017.

Krzysztof Kaczmarski, Paweł Rzążewski, and Albert Wolant. Massively parallel construction of the cell graph. volume 9573 of *Lecture Notes in Computer Science*, pages 559–569. Springer, 2015.

Krzysztof Kaczmarski, Paweł Rzążewski, and Albert Wolant. Parallel algorithms constructing the cell graph. *Concurr. Comput. Pract. Exp.*, 29(23), 2017.

Krzysztof Kaczmarski and Albert Wolant. Radix tree for binary sequences on GPU. volume 10777 of *Lecture Notes in Computer Science*, pages 219–231. Springer, 2017.

Krzysztof Kaczmarski and Albert Wolant. GPU r-trie: Dictionary with ultra fast lookup. *Concurr. Comput. Pract. Exp.*, 31(19), 2019.

Materiały sponsorowane przez:

Projekt „NERW 2 PW. Nauka – Edukacja – Rozwój – Współpraca" współfinansowany jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Zadanie 10 pn. „Modyfikacja programów studiów na kierunkach prowadzonych przez Wydział Matematyki i Nauk Informacyjnych", realizowane w ramach projektu „NERW 2 PW. Nauka – Edukacja – Rozwój – Współpraca", współfinansowanego jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

Fundusze Europejskie Wiedza Edukacja Rozwój
Rzeczpospolita Polska
Politechnika Warszawska
Unia Europejska Europejski Fundusz Społeczny