Parallel Shortest Lattice Vector Enumeration on Graphics Cards

Jens Hermans[1]
Johannes Buchmann[2]
Bart Preneel[1]

Michael Schneider[2]
Frederik Vercauteren[1]

[1] **K.U.Leuven**

[2] **TU Darmstadt**

mischnei@cdc.informatik.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

Introduction

Lattices: crash course

GPUs

GPU-Enum Algorithm

The Future

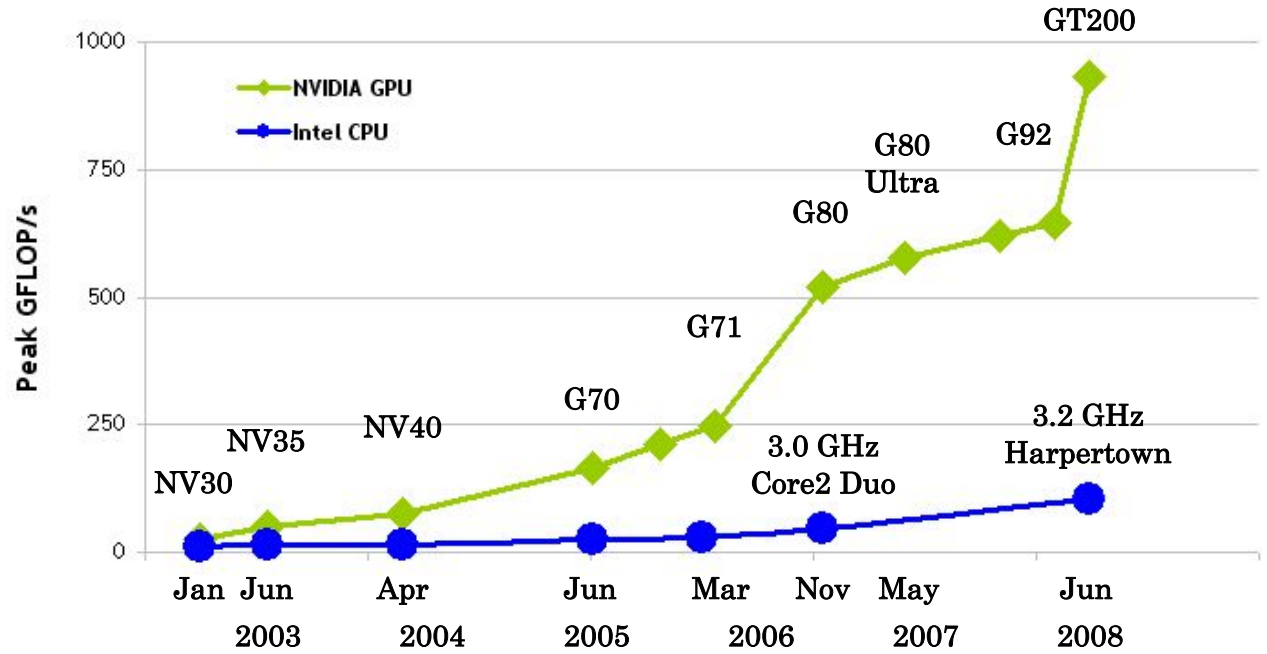
Why GPU?

TECHNISCHE UNIVERSITÄT DARMSTADT

(Source: MSI)

Why GPU?

TECHNISCHE UNIVERSITÄT DARMSTADT



GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

(Source: NVIDIA)

Why GPU?

TECHNISCHE UNIVERSITÄT DARMSTADT

Absolute transistor count:

AMD Athlon 64 X2	CPU	154 mio.
Intel Core 2 Duo	CPU	291 mio.
Intel Core 2 Quad	CPU	582 mio.
NVIDIA G8800 GTX	GPU	680 mio.
Indel Core i7-980X Extreme	CPU	1.2 mrd.
ATI RV870	GPU	2.2 mrd.
NVIDIA Fermi	GPU	3.0 mrd.

CUDA framework

TECHNISCHE UNIVERSITÄT DARMSTADT

Warning: Sales Talk

Your own personal supercomputer for $<$ €500.

Nvidia CUDA Framework:

- Run ‘general’ programs on GPU
- More complex operations, data types, branching...
- Recent GPU required
- Theory: 1TFlop (practice: 200 GFlop)

Crypto on GPU

TECHNISCHE UNIVERSITÄT DARMSTADT

Current Crypto Applications

- Ciphers:
  - RSA[a], ECC[b], AES[c]
- Cryptanalysis:
  - Factoring[d]
  - Brute force

[a] Moss, Page, Smart / Szerwinski, Guneysu / Fleissner
[b] Szerwinski, Guneysu
[c] Manavski / Harrison, Waldron
[d] Bernstein, Chen, Cheng, Lange, Yang

Introduction

Lattices: crash course

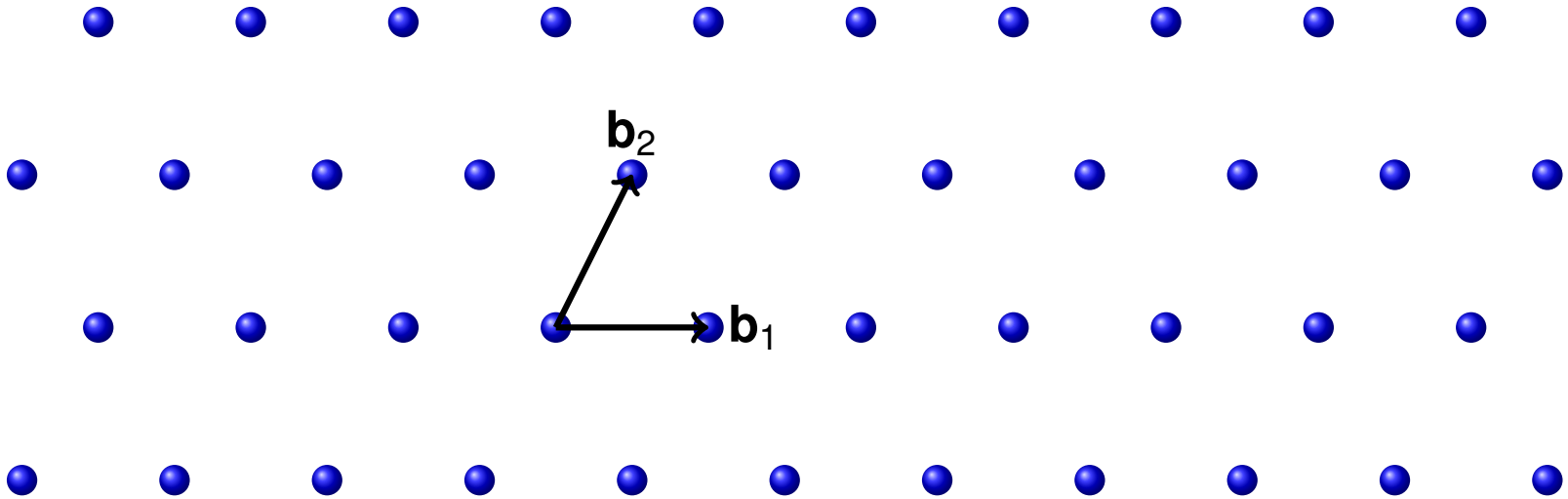
GPUs

GPU-Enum Algorithm

The Future

Lattices

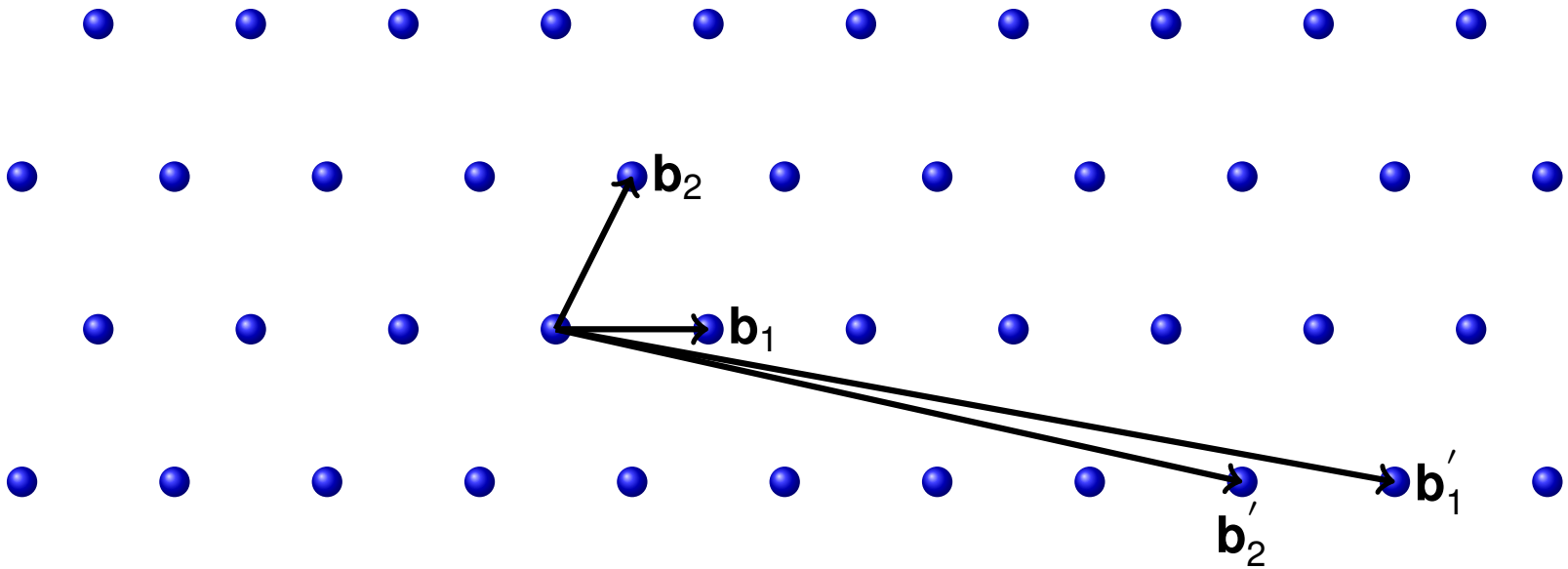
TECHNISCHE UNIVERSITÄT DARMSTADT



- Basis matrix $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ with $\mathbf{b}_i \in \mathbb{R}^d$
- Lattice: $\mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i, x_i \in \mathbb{Z}\}$

Shortest Vector Problem (SVP)

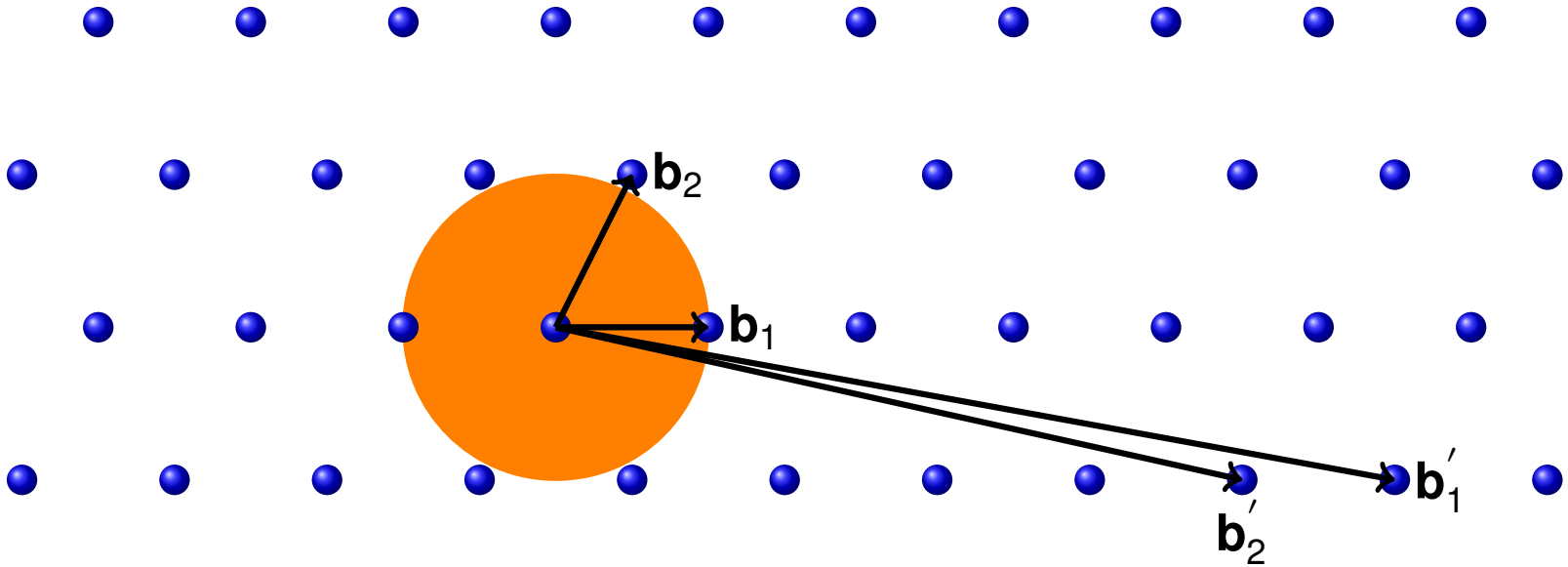
TECHNISCHE UNIVERSITÄT DARMSTADT



- Basis not unique
- Idea: 'good' basis $\mathbf{B}$ and 'bad' basis $\mathbf{B}'$
- Finding a shortest vector is hard with $\mathbf{B}'$

Shortest Vector Problem (SVP)

TECHNISCHE UNIVERSITÄT DARMSTADT



- Basis not unique
- Idea: 'good' basis **B** and 'bad' basis **B**′
- Finding a shortest vector is hard with **B**′

Algorithms for SVP

TECHNISCHE UNIVERSITÄT DARMSTADT

Shortest Vector Problem

$$\min_{\mathbf{x} \in \mathbb{Z}^n} \left\| \mathbf{Bx} \right\|_2$$

SVP Algorithms:

LLL	approximate solution, polynomial time
BKZ	better approximation
...	
Enum	exact solution, exponential time
Sieving	exact solution, exponential time, probabilistic

$\Longrightarrow$ This talk: focus on Enum

Introduction

Lattices: crash course

GPUs

GPU-Enum Algorithm

The Future

Processor

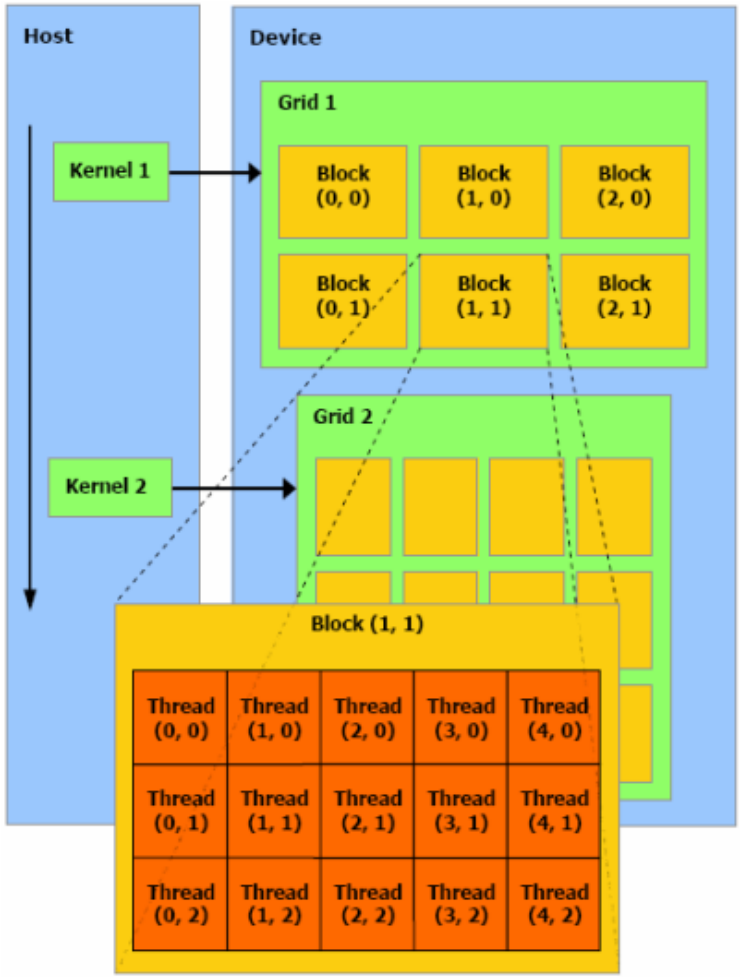
TECHNISCHE UNIVERSITÄT DARMSTADT

Nvidia GTX280:

- 240 cores, scalar processors
- 30 multiprocessors (8 cores each)
- 1.3 GHz
- 1GB Global Memory
- 32 & 64-bit integers, single + double precision FP

Programming model

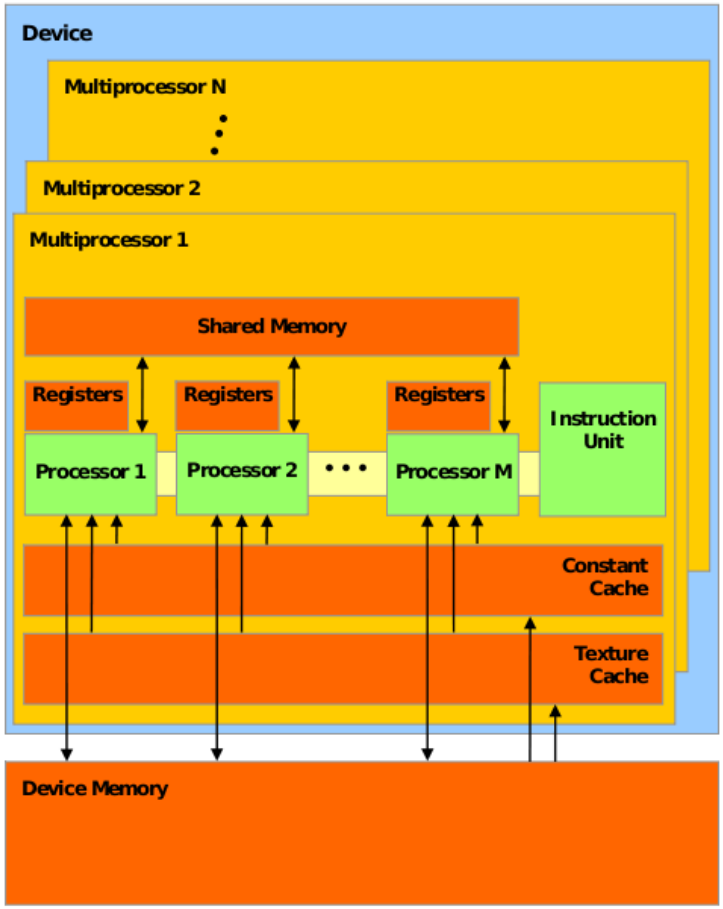
TECHNISCHE UNIVERSITÄT DARMSTADT



(Source: CUDA programming guide)

Memory types

TECHNISCHE UNIVERSITÄT DARMSTADT



(Source: CUDA programming guide)

Code Example

TECHNISCHE UNIVERSITÄT DARMSTADT

```
CUDA_SAFE_CALL(cudaMalloc((void **)&d_A, sizeof(norminttype)));
CUDA_SAFE_CALL(cudaMemcpy(d_A,h_A,sizeof(norminttype),cudaMemcpyHostToDevic
CUDA_SAFE_CALL(cudaFree(d_A));
gpuenum <<<16,32>>>(paramlist);
```

Code Example

TECHNISCHE UNIVERSITÄT DARMSTADT

```
CUDA_SAFE_CALL(cudaMalloc((void **)&d_A, sizeof(norminttype)));
CUDA_SAFE_CALL(cudaMemcpy(d_A,h_A,sizeof(norminttype),cudaMemcpyHostToDevic
CUDA_SAFE_CALL(cudaFree(d_A));
gpuenum <<<16,32>>>(paramlist);

__global__ void gpuenum(paramlist)
    for (int k = 0; k < alpha + 1; k++) {
        fptype cintsumtmp = 0.0;
        for (int j = alpha; j <= beta; j++) {
            cintsumtmp += (x[j + sharedOffset] * MU(j, k));
        }
    }
    if (startlevel >= alpha && l > localA)
        continue;
```

Introduction

Lattices: crash course

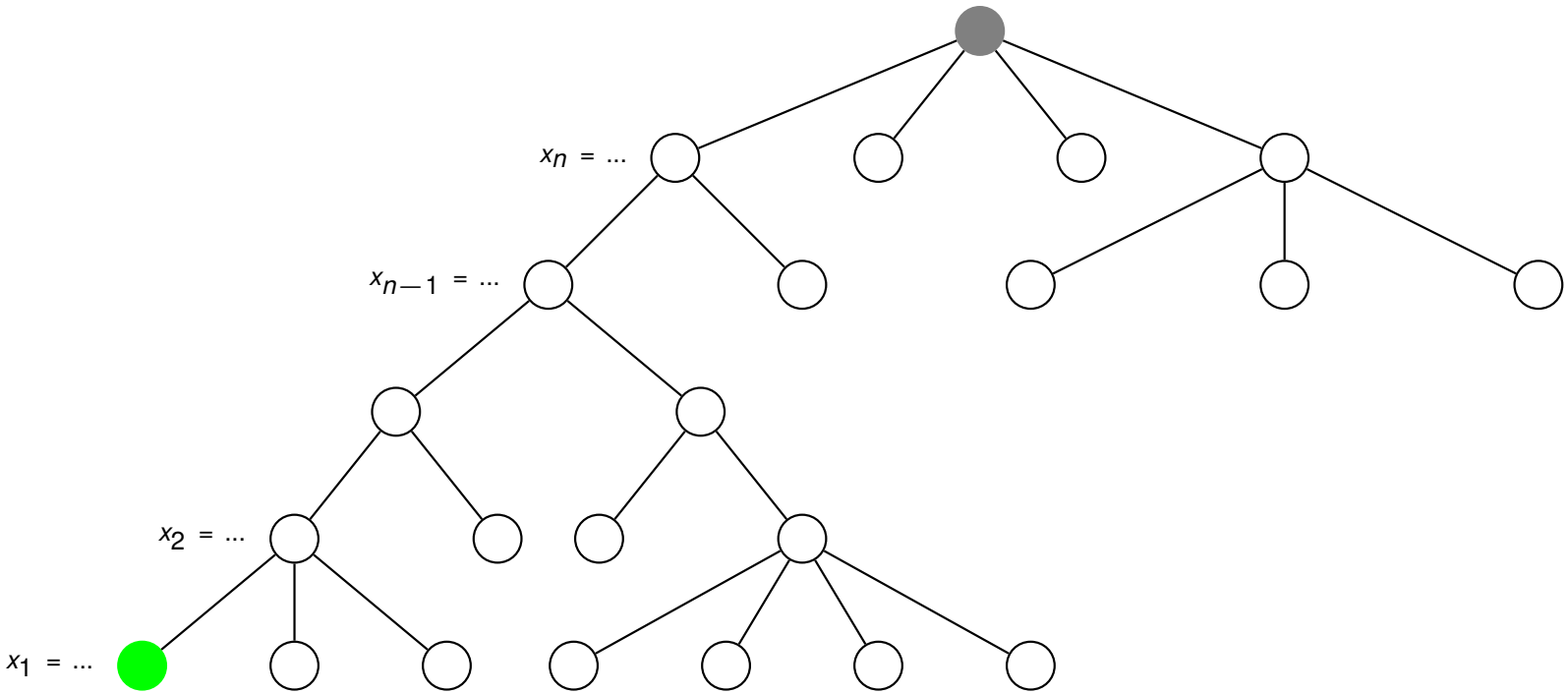
GPUs

GPU-Enum Algorithm

The Future

Enumeration

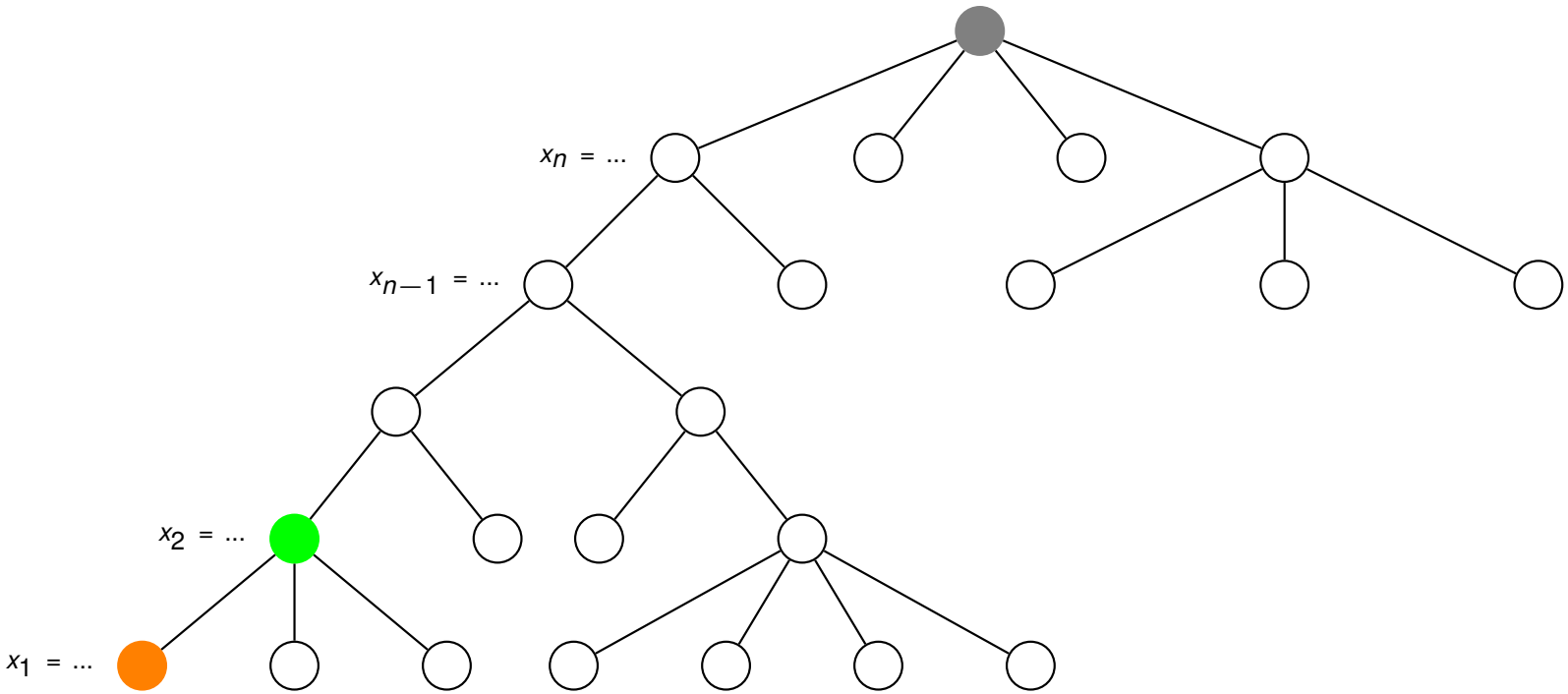
TECHNISCHE UNIVERSITÄT DARMSTADT



$\mathbf{x} = [1, 0, \ldots, 0] \quad \longrightarrow \quad \text{optimum } A = \|\mathbf{Bx}\|_2^2$

Enumeration

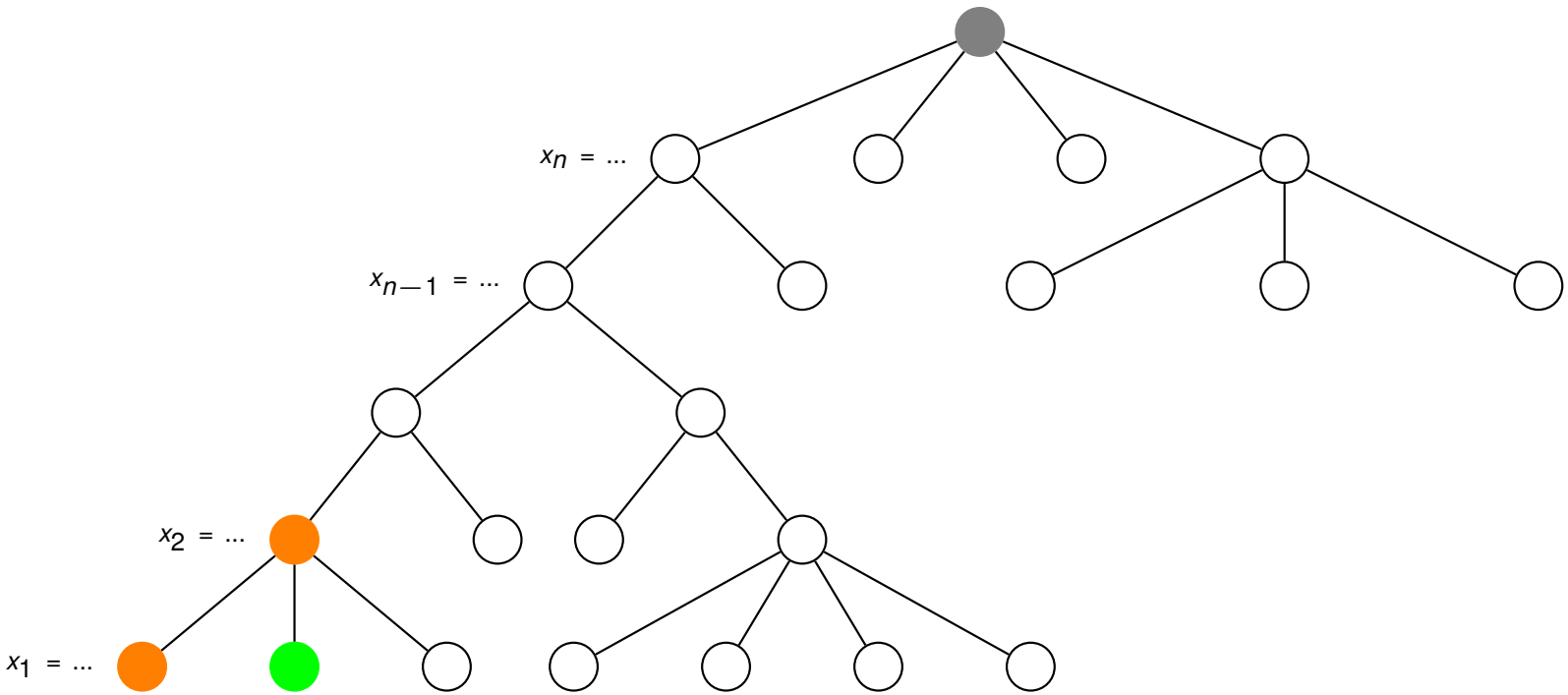
TECHNISCHE UNIVERSITÄT DARMSTADT



Intermediate norm l_2 of vector $[\star, x_2, x_3, \ldots, x_n]$
with $l_i \geq l_{i+1}$

Enumeration

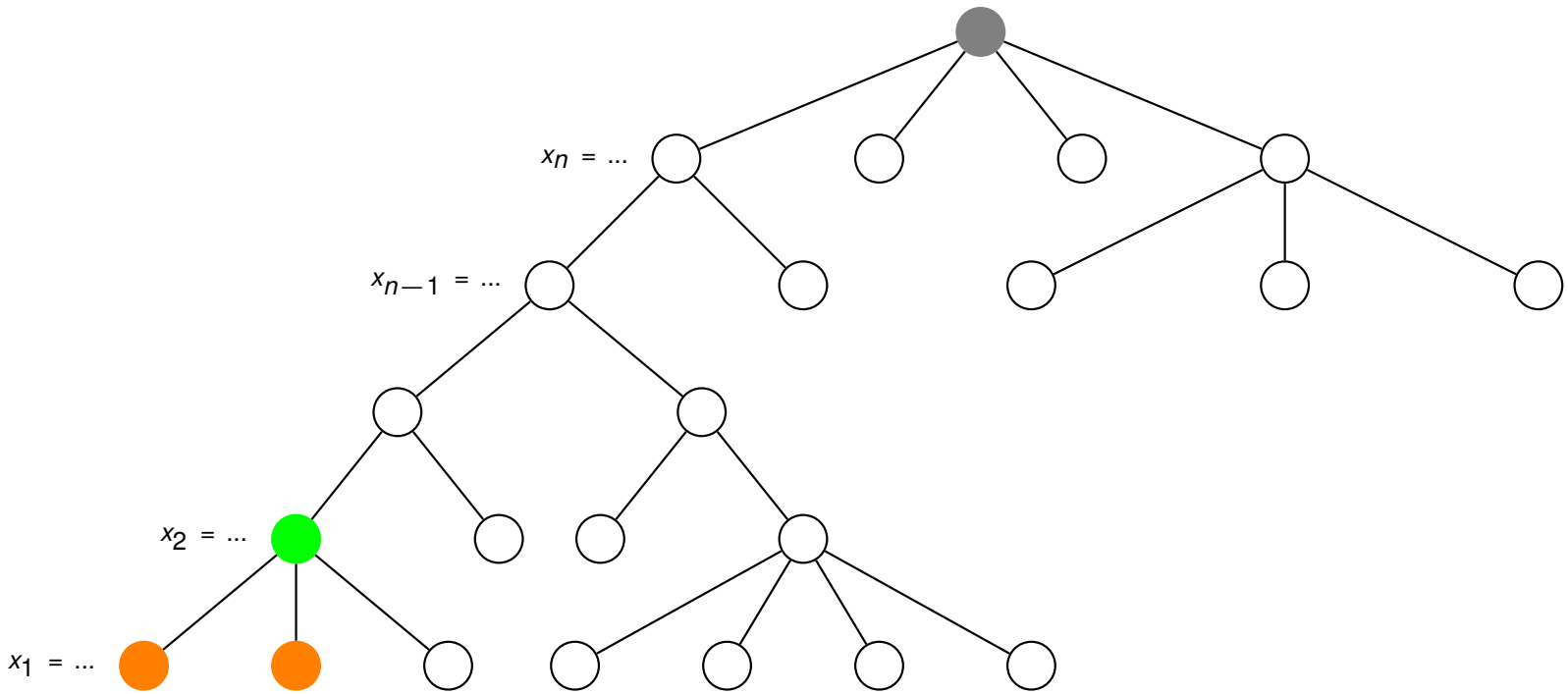
TECHNISCHE UNIVERSITÄT DARMSTADT



New optimum $A = \|\mathbf{Bx}\|_2^2$

Enumeration

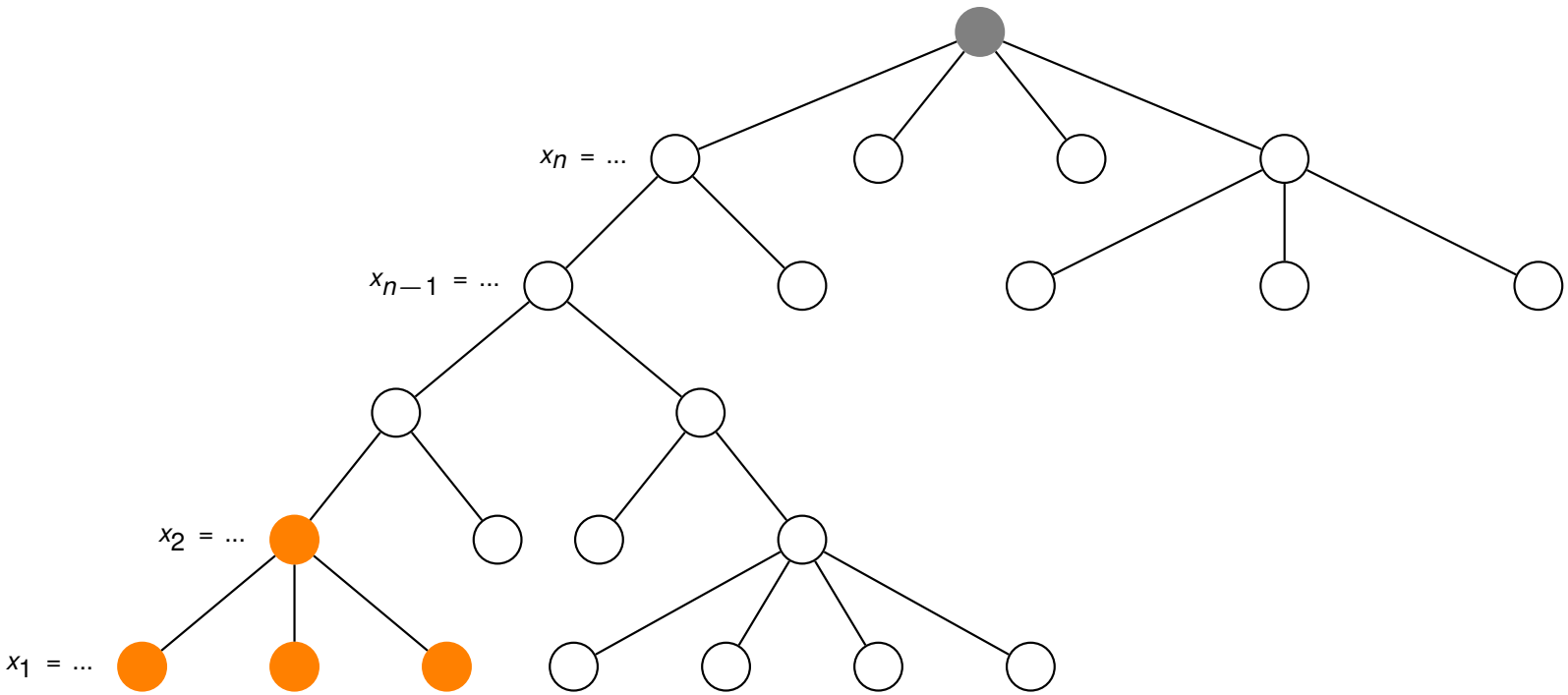
TECHNISCHE UNIVERSITÄT DARMSTADT



Cut off branch if $l_i > A$.

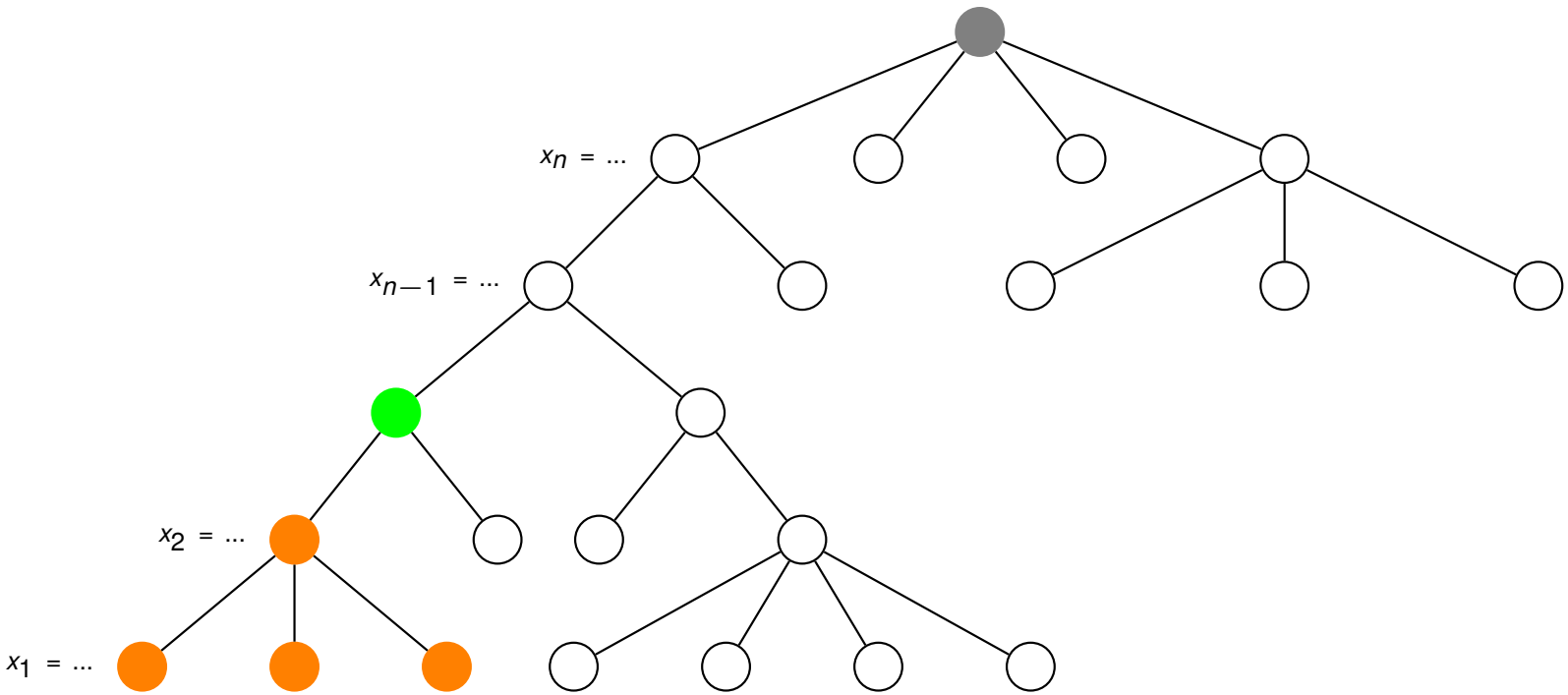
Enumeration

TECHNISCHE UNIVERSITÄT DARMSTADT



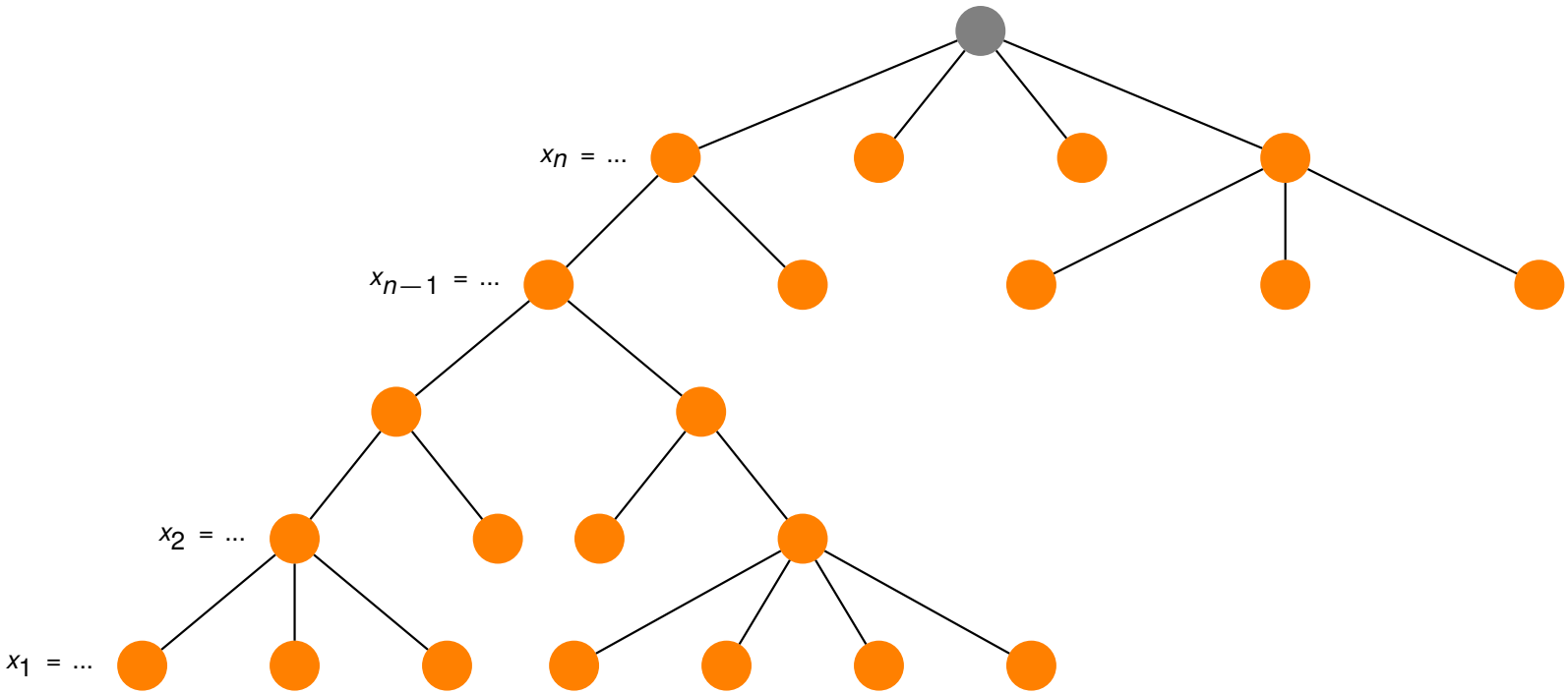
Enumeration

TECHNISCHE UNIVERSITÄT DARMSTADT



Enumeration

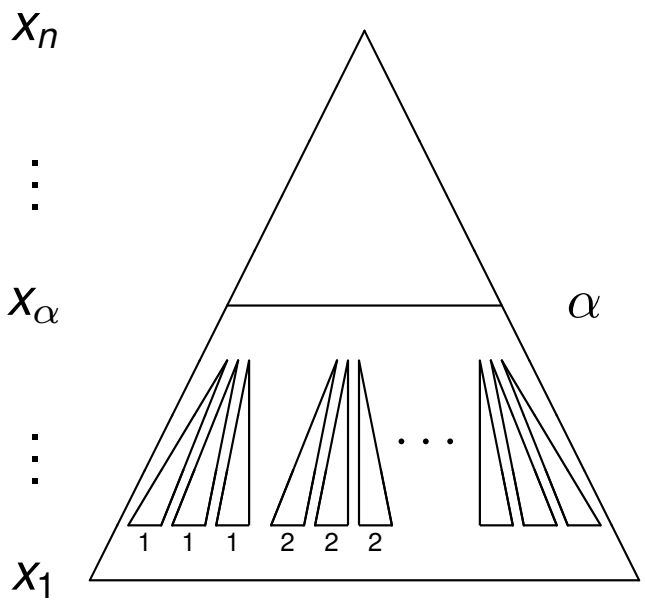
TECHNISCHE UNIVERSITÄT DARMSTADT



Algorithm Flow

TECHNISCHE UNIVERSITÄT DARMSTADT

How to find independent subtrees?



Basic idea

TECHNISCHE UNIVERSITÄT DARMSTADT

Input: $\mathbf{B}$, A, α, n

Compute the Gram-Schmidt decomposition of $\mathbf{b}_i$

Output: $(x_1, \ldots, x_n)$ with $\left\| \sum_{i=1}^{n} x_i \mathbf{b}_i \right\|$ minimal

Basic idea

TECHNISCHE UNIVERSITÄT DARMSTADT

Input: $\mathbf{B}$, A, α, n

Compute the Gram-Schmidt decomposition of $\mathbf{b}_i$
CPU: generate $\mathbf{x}_i = [0, \ldots, 0, x_\alpha, \ldots, x_n]$

Output: $(x_1, \ldots, x_n)$ with $\left\| \sum_{i=1}^{n} x_i \mathbf{b}_i \right\|$ minimal

Basic idea

TECHNISCHE UNIVERSITÄT DARMSTADT

Input: $\mathbf{B}$, A, α, n

Compute the Gram-Schmidt decomposition of $\mathbf{b}_i$

CPU: generate $\mathbf{x}_i = [0, \ldots, 0, x_\alpha, \ldots, x_n]$

GPU thread: run a sub-enum on $\mathbf{x}_i$, if new optimum $\longrightarrow$ store in $\mathbf{x}$

Output: $(x_1, \ldots, x_n)$ with $\left\| \sum_{i=1}^{n} x_i \mathbf{b}_i \right\|$ minimal

Basic idea

TECHNISCHE UNIVERSITÄT DARMSTADT

Input: $\mathbf{B}$, A, α, n

Compute the Gram-Schmidt decomposition of $\mathbf{b}_i$

CPU: generate $\mathbf{x}_i = [0, \ldots, 0, x_\alpha, \ldots, x_n]$

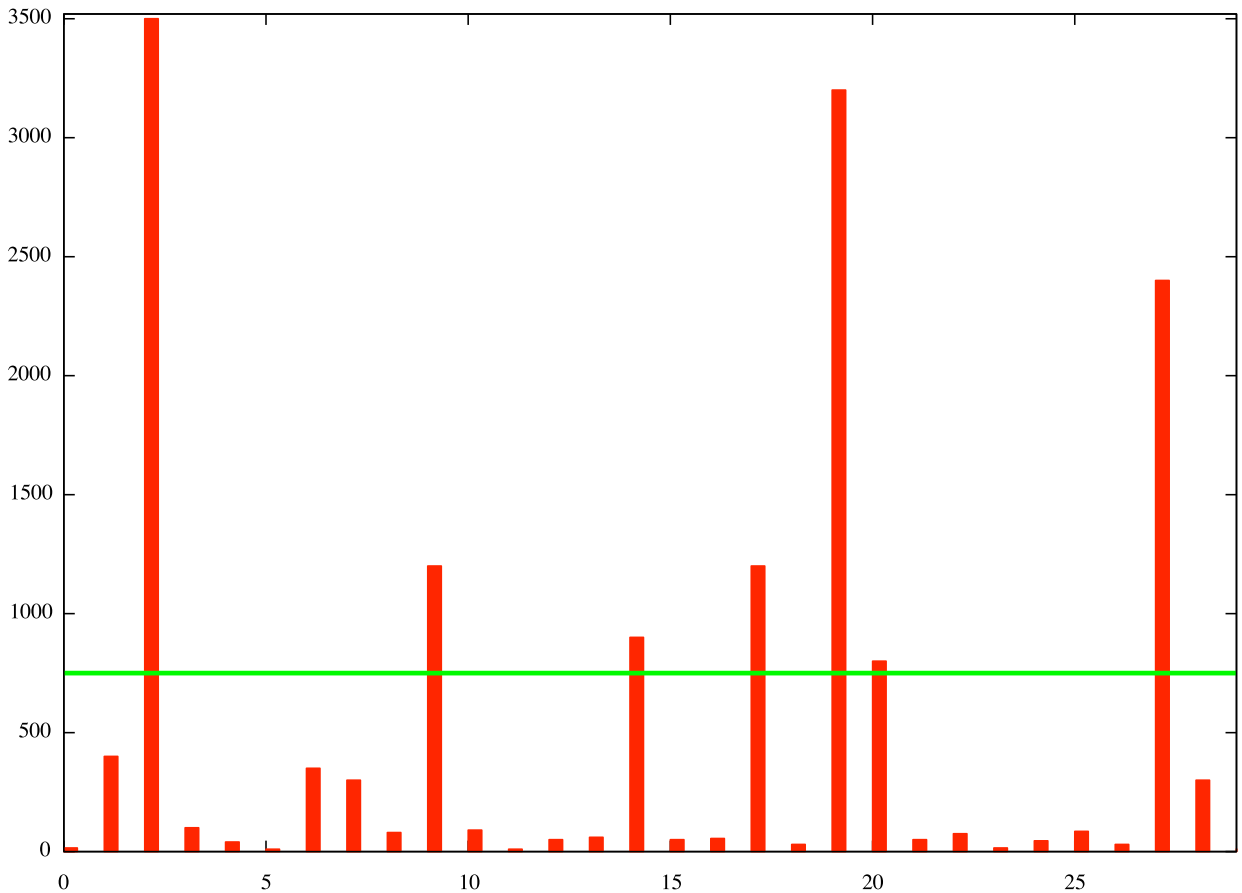
GPU thread: run a sub-enum on $\mathbf{x}_i$, if new optimum $\longrightarrow$ store in $\mathbf{x}$

Output: $(x_1, \ldots, x_n)$ with $\left\| \sum_{i=1}^{n} x_i \mathbf{b}_i \right\|$ minimal

$\Longrightarrow$ horrible performance 

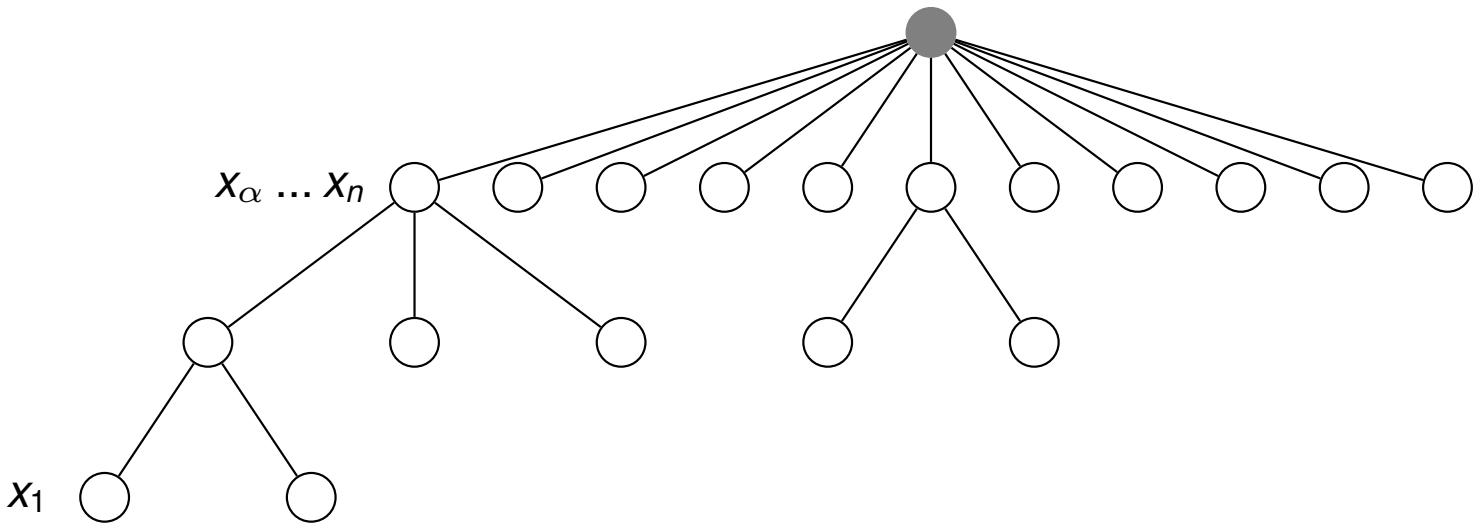
Thread execution time

TECHNISCHE UNIVERSITÄT DARMSTADT



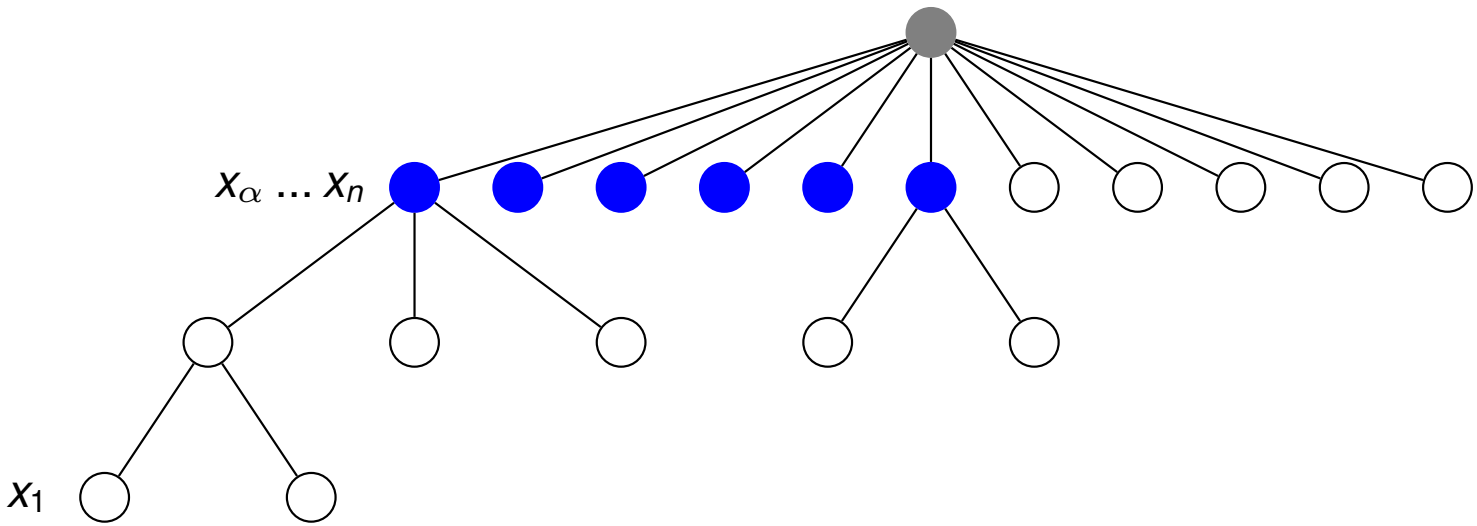
GPU Enumeration

TECHNISCHE UNIVERSITÄT DARMSTADT



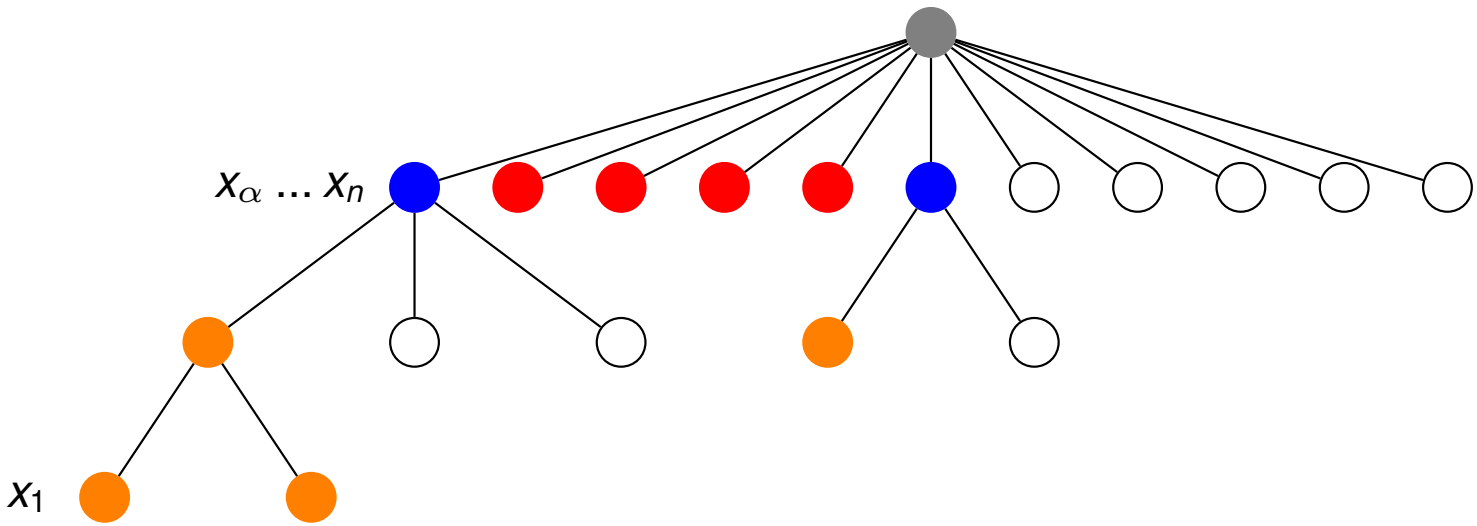
GPU Enumeration

TECHNISCHE UNIVERSITÄT DARMSTADT



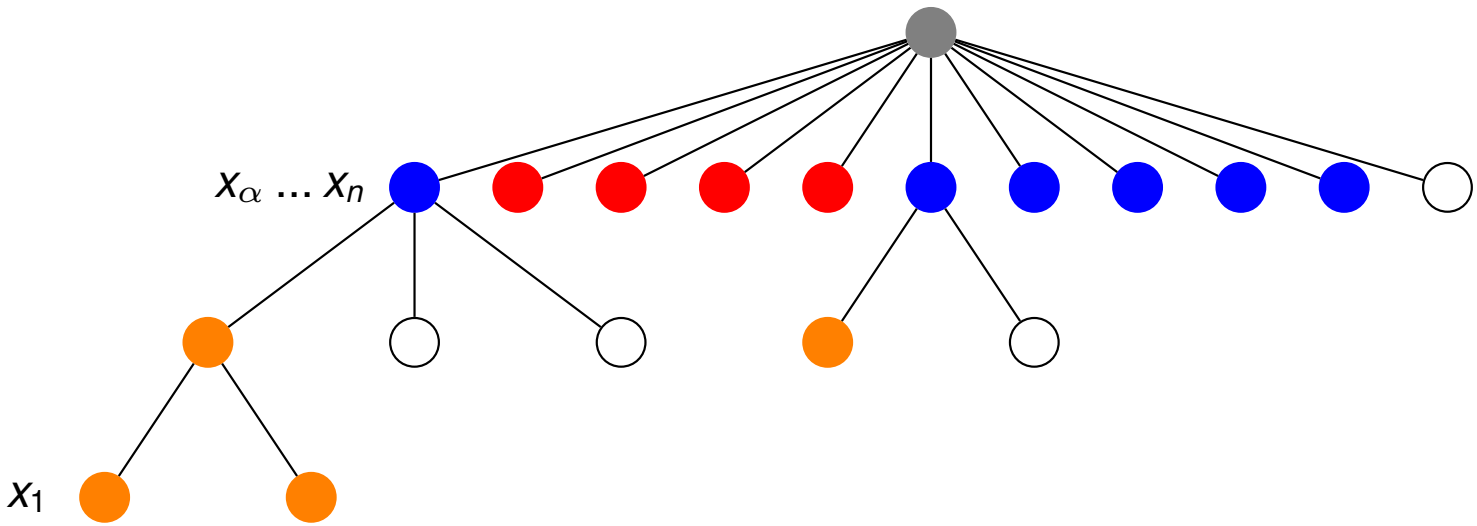
GPU Enumeration

TECHNISCHE UNIVERSITÄT DARMSTADT



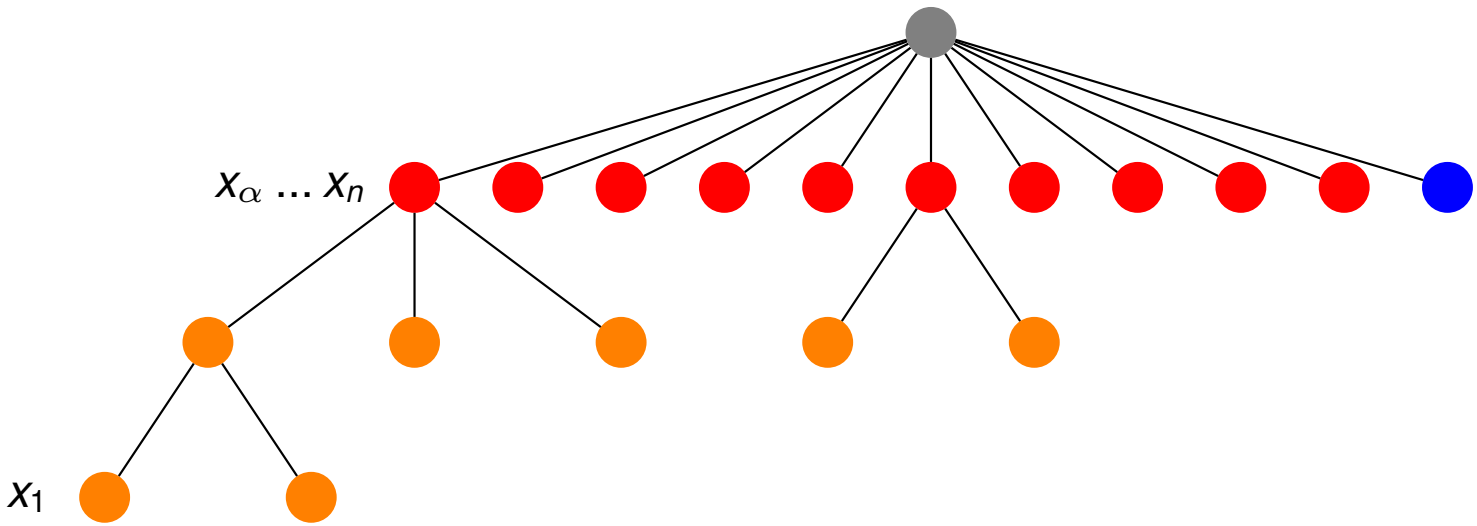
GPU Enumeration

TECHNISCHE UNIVERSITÄT DARMSTADT



GPU Enumeration

TECHNISCHE UNIVERSITÄT DARMSTADT



Iterating

TECHNISCHE UNIVERSITÄT DARMSTADT

Input: $\mathbf{B}$, A, α, n

Compute the Gram-Schmidt decomposition of $\mathbf{b}_i$

while *true* **do**

CPU: generate some $\mathbf{x}_i = [0, \ldots, 0, x_\alpha, \ldots, x_n]$

CPU: Get enum state $\bar{\mathbf{x}}_i = [\bar{x}_1, \ldots, \bar{x}_{\alpha-1}, x_\alpha, \ldots, x_n]$

end

Output: $(x_1, \ldots, x_n)$ with $\left\| \sum_{i=1}^{n} x_i \mathbf{b}_i \right\|$ minimal

Iterating

TECHNISCHE UNIVERSITÄT DARMSTADT

Input: $\mathbf{B}$, A, α, n

Compute the Gram-Schmidt decomposition of $\mathbf{b}_i$
while *true* **do**
  CPU: generate some $\mathbf{x}_i = [0, \ldots, 0, x_\alpha, \ldots, x_n]$
  GPU thread:
  while *there are* $\mathbf{x}_i$ *left* **do**
    Start enum for a certain $\mathbf{x}_i$ or continue enum for $\bar{\mathbf{x}}_i$
    Stop enum after $\mathbf{M}$ steps, store the state $\{l_i, \bar{\mathbf{x}}_i, s_i = \mathbf{M}\}$
  end
  CPU: Get enum state $\bar{\mathbf{x}}_i = [\bar{x}_1, \ldots, \bar{x}_{\alpha-1}, x_\alpha, \ldots, x_n]$
end

Output: $(x_1, \ldots, x_n)$ with $\left\| \sum_{i=1}^{n} x_i \mathbf{b}_i \right\|$ minimal

Implementation details

TECHNISCHE UNIVERSITÄT DARMSTADT

Some facts & figures:

- Dimension 50, 100,000 starting vectors $\rightarrow$ upload & download $\sim$ 20 MB of data to GPU
- CPU top enum: very fast (low dimension)
- GPU runs for $>$ 10 seconds per iteration, iteration overhead is limited
- Share new optimal values among GPU threads

Throughput

TECHNISCHE UNIVERSITÄT DARMSTADT

Throughput:

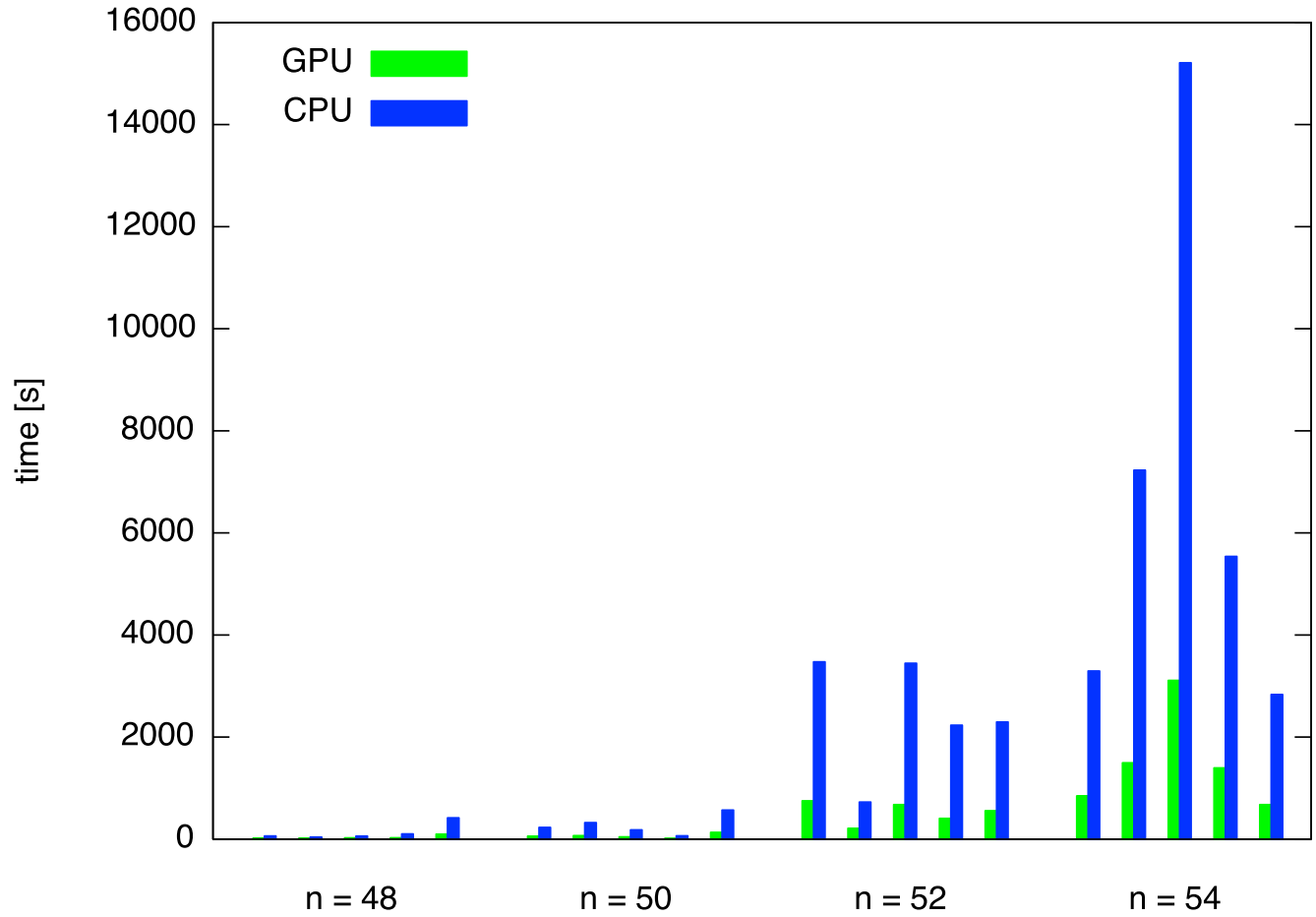
- CPU: $25 \cdot 10^6$ steps/s
- GPU: $200 \cdot 10^6$ steps/s

Throughput on GPU depends on:

- Lattice dimension n
- Length of sub-enumerations
- Number of parallel threads, uploaded points...

Results

TECHNISCHE UNIVERSITÄT DARMSTADT



Results

TECHNISCHE UNIVERSITÄT DARMSTADT

n	44	48	52	56	60
`fpLLL` - ENUM	17.7s	136s	2434s	137489s	-
CUDA - ENUM	11.7s	37.0s	520s	30752s	274268s
	66%	27%	21%	22%	-

Speedup in this model: ≈ 5

`http://homes.esat.kuleuven.be/~jhermans/gpuenum/index.html`

Results

TECHNISCHE UNIVERSITÄT DARMSTADT

CPU machine

- 4-core CPU
- $900
- speedup 4
- total cost 91 · t

GPU machine

- 4-core CPU + 2 x GTX295 GPU
- $2200
- speedup 24
- total cost 225 · t

Speedup in this model: 2.4

Ideas for the Future

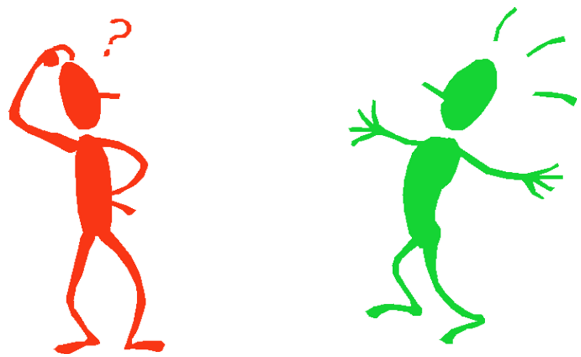
TECHNISCHE UNIVERSITÄT DARMSTADT

- Generalize ideas (not specific for gpu's... clusters?)
- Use full power of CPU (now: idle during gpu-time)
- Test on NVIDIA Fermi cards

(... and we will climb the highest mountains)

Finally...

TECHNISCHE UNIVERSITÄT DARMSTADT



Questions?

Problems

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- GTX 280 can make 13 times more fmadds than Core2Extreme CPU
- using CUDA profiler
- diverging branches, warp serializations, ...
- lots of ALUs left idle