Improved capacity upper bounds for the deletion channel using a parallelized Blahut-Arimoto Algorithm

Martim Pinto

Técnico-ULisboa

João Ribeiro

Inst. Telecomunicações
+ Técnico-ULisboa

Funded by
the European Union

European Research Council
Established by the European Commission

ERC STG LESYNCH 101218842

Deletions

A simple model of errors that cause loss of synchronization.

Deletions

A simple model of errors that cause loss of synchronization.

Erasures

Deletions

A simple model of errors that cause loss of synchronization.

Erasures

1 0 1 0 1 0 1 0

Deletions

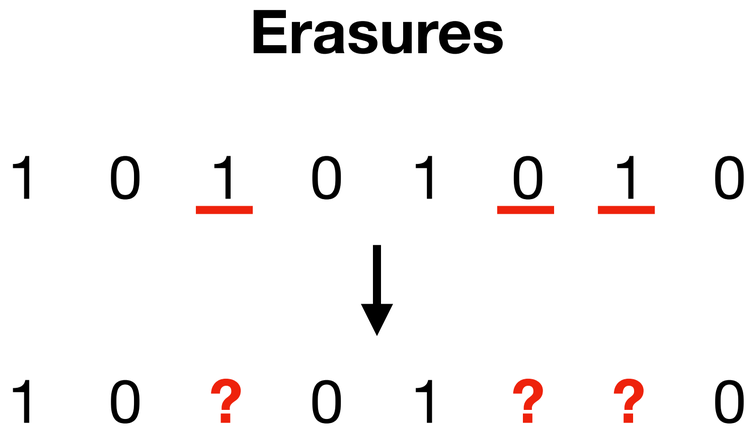
A simple model of errors that cause loss of synchronization.

Erasures

1 0 1 0 1 0 1 0

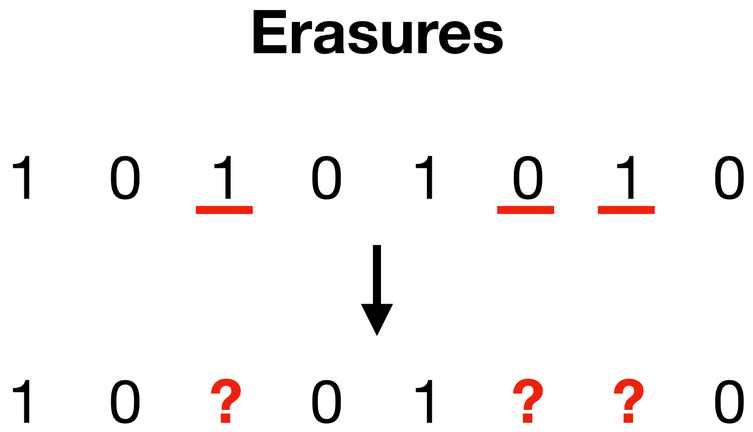
Deletions

A simple model of errors that cause loss of synchronization.



Deletions

A simple model of errors that cause loss of synchronization.

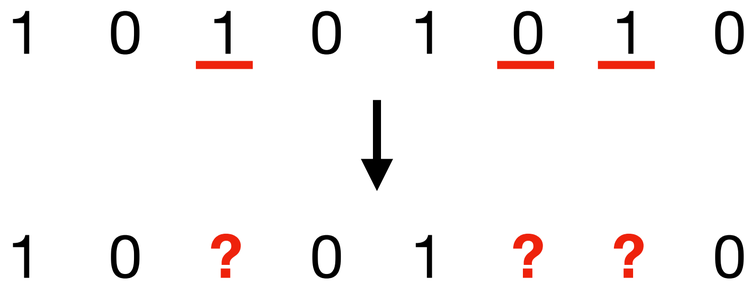


Well-understood

Deletions

A simple model of errors that cause loss of synchronization.

Erasures

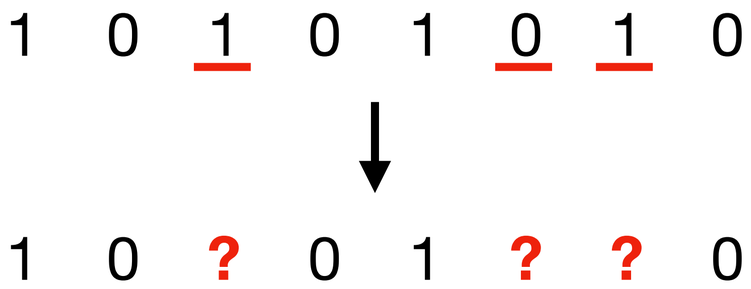


Well-understood

Deletions

A simple model of errors that cause loss of synchronization.

Erasures



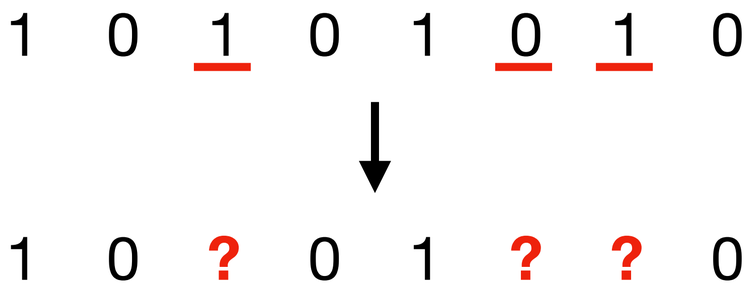
Well-understood

Deletions

Deletions

A simple model of errors that cause loss of synchronization.

Erasures



Well-understood

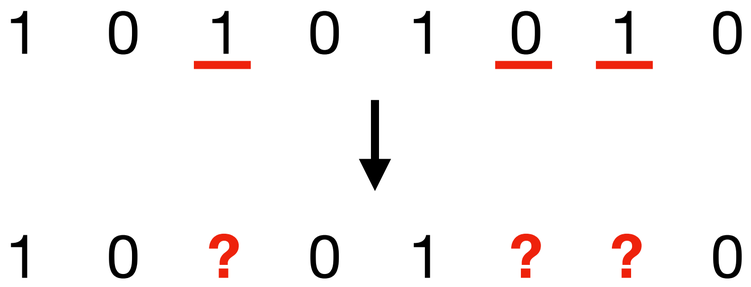
Deletions

1 0 1 0 1 0 1 0

Deletions

A simple model of errors that cause loss of synchronization.

Erasures



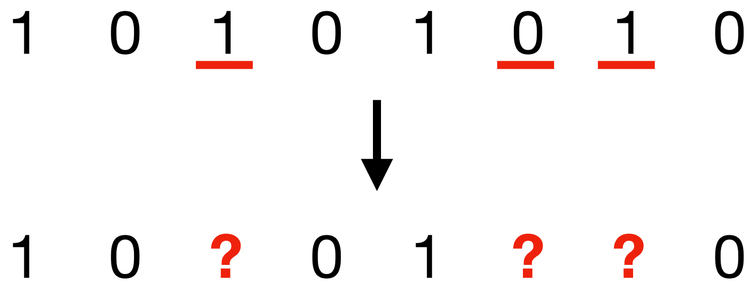
Well-understood

Deletions

Deletions

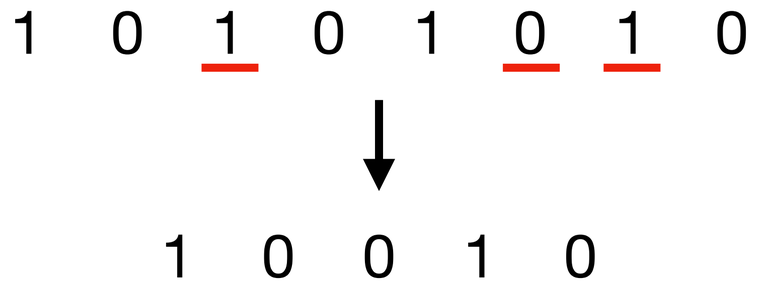
A simple model of errors that cause loss of synchronization.

Erasures



Well-understood

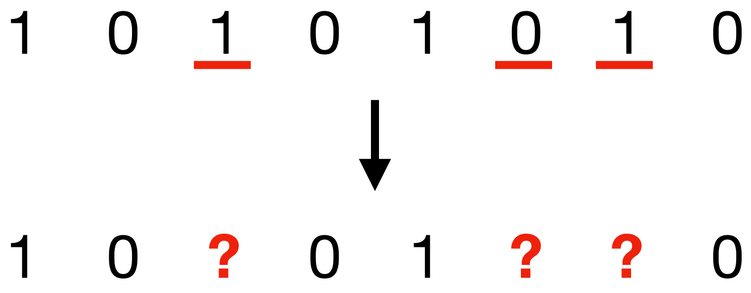
Deletions



Deletions

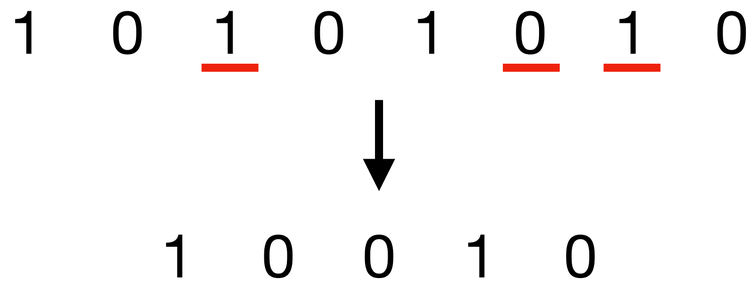
A simple model of errors that cause loss of synchronization.

Erasures



Well-understood

Deletions



New approaches required!

The deletion channel

On input a string $x \in \{0,1\}^n$, independently deletes each bit of x with probability d.

The deletion channel

On input a string $x \in \{0,1\}^n$, independently deletes each bit of x with probability d.

Major open problem:

What's the capacity $C(d)$ of the deletion channel
with deletion probability d?

The deletion channel

On input a string $x \in \{0,1\}^n$, independently deletes each bit of x with probability d.

Major open problem:

What's the capacity $C(d)$ of the deletion channel
with deletion probability d?

An analog of Shannon's noisy channel coding theorem [Dobrushin 1967]:

$$C(d) = \lim_{n\to\infty} \frac{1}{n} \sup_{X^n} I(X^n; Y^n)$$

The deletion channel

On input a string $x \in \{0,1\}^n$, independently deletes each bit of x with probability d.

Major open problem:

What's the capacity $C(d)$ of the deletion channel
with deletion probability d?

An analog of Shannon's noisy channel coding theorem [Dobrushin 1967]:

$$C(d) = \lim_{n\to\infty} \frac{1}{n} \sup_{X^n} I(X^n; Y^n)$$

n-bit input distribution

The deletion channel

On input a string $x \in \{0,1\}^n$, independently deletes each bit of x with probability d.

Major open problem:

What's the capacity $C(d)$ of the deletion channel
with deletion probability d?

An analog of Shannon's noisy channel coding theorem [Dobrushin 1967]:

$$C(d) = \lim_{n\to\infty} \frac{1}{n} \sup_{X^n} I(X^n; Y^n)$$

corresponding channel output

n-bit input distribution

The deletion channel

On input a string $x \in \{0,1\}^n$, independently deletes each bit of x with probability d.

Major open problem:

What's the capacity $C(d)$ of the deletion channel
with deletion probability d?

An analog of Shannon's noisy channel coding theorem [Dobrushin 1967]:

$$C(d) = \lim_{n\to\infty} \frac{1}{n} \sup_{X^n} I(X^n; Y^n)$$

corresponding channel output

n-bit input distribution

but doesn't simplify like for DMCs…

A very brief history of the problem

Naively: $0 \leq C(d) \leq 1 - d$

A very brief history of the problem

Naively: $0 \leq C(d) \leq 1 - d$

1960's: Some non-trivial lower bounds on $C(d)$ (Gallager, Zigangirov).

A very brief history of the problem

Naively: $0 \leq C(d) \leq 1 - d$

1960's: Some non-trivial lower bounds on $C(d)$ (Gallager, Zigangirov).

Revived in the early 2000's:

A very brief history of the problem

Naively: $0 \leq C(d) \leq 1 - d$

1960's: Some non-trivial lower bounds on $C(d)$ (Gallager, Zigangirov).

Revived in the early 2000's:

- Improved lower bounds via input distributions with memory + better decoders;
  [Diggavi-Grossglauser, Drinea-Mitzenmacher, Drinea-Kirsch, Kanoria-Montanari, Venkataramanan-Tatikonda-Ramchandran]

A very brief history of the problem

Naively: $0 \leq C(d) \leq 1 - d$

1960's: Some non-trivial lower bounds on $C(d)$ (Gallager, Zigangirov).

Revived in the early 2000's:

- Improved lower bounds via input distributions with memory + better decoders;
  [Diggavi-Grossglauser, Drinea-Mitzenmacher, Drinea-Kirsch, Kanoria-Montanari, Venkataramanan-Tatikonda-Ramchandran]
- Non-trivial **upper bounds.**
  [Diggavi-Mitzenmacher-Pfister, Fertonani-Duman, Kalai-Mitzenmacher-Sudan, Kanoria-Montanari, Rahmati-Duman, Cheraghchi, Rubinstein-Con]

The high-level idea behind capacity upper bounds

Carefully add side information to the channel so that it becomes a DMC.

The high-level idea behind capacity upper bounds

Carefully add side information to the channel so that it becomes a DMC.

The first example: Reveal locations of runs that were completely deleted.
[Diggavi-Mitzenmacher-Pfister 2007]

The high-level idea behind capacity upper bounds

Carefully add side information to the channel so that it becomes a DMC.

The first example: Reveal locations of runs that were completely deleted.
[Diggavi-Mitzenmacher-Pfister 2007]

0 0 1 1 1 0 1 0 0

The high-level idea behind capacity upper bounds

Carefully add side information to the channel so that it becomes a DMC.

The first example: Reveal locations of runs that were completely deleted.
[Diggavi-Mitzenmacher-Pfister 2007]

0 0 1 **1** 1 **0** 1 0 0

The high-level idea behind capacity upper bounds

Carefully add side information to the channel so that it becomes a DMC.

The first example: Reveal locations of runs that were completely deleted.
[Diggavi-Mitzenmacher-Pfister 2007]

The high-level idea behind capacity upper bounds

Carefully add side information to the channel so that it becomes a DMC.

The first example: Reveal locations of runs that were completely deleted.
[Diggavi-Mitzenmacher-Pfister 2007]

$\Longrightarrow$ channel output becomes memoryless across input runs.

The high-level idea behind capacity upper bounds

Carefully add side information to the channel so that it becomes a DMC.

The first example: Reveal locations of runs that were completely deleted.
[Diggavi-Mitzenmacher-Pfister 2007]

0 0 1 **1** 1 **0** 1 0 0 $\longrightarrow$ **x** 1 1 **x** 1 0 0

$\Longrightarrow$ channel output becomes memoryless across input runs.

$\Longrightarrow$ can upper bound $C(d)$ by upper bounding capacity per unit cost of the DMC

$$\ell \in \mathbb{N} \mapsto \mathsf{Bin}(\ell, 1-d)$$

The approach behind the best capacity upper bounds

The approach behind the best capacity upper bounds

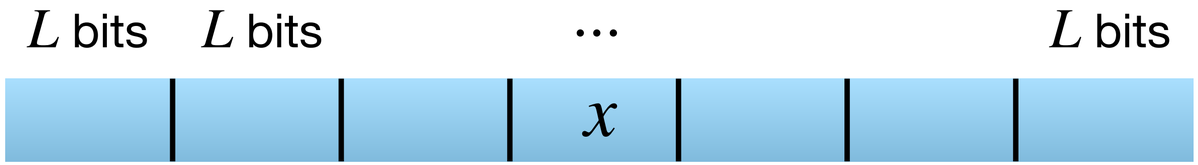
Fix an integer $L \geq 1$.

The approach behind the best capacity upper bounds

Fix an integer $L \geq 1$.

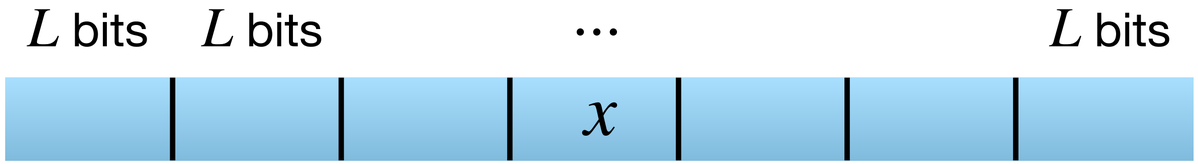
The approach behind the best capacity upper bounds

Fix an integer $L \geq 1$.



The approach behind the best capacity upper bounds

Fix an integer $L \geq 1$.

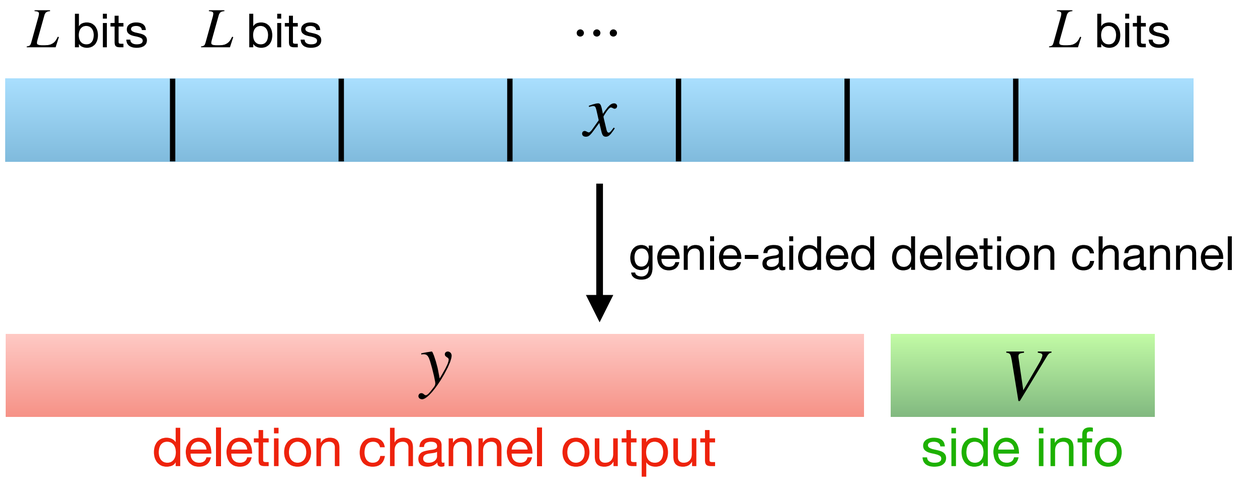


$V_i =$ nr. bits deleted in i-th block

$V = (V_1, V_2, \ldots, V_{n/L})$

The approach behind the best capacity upper bounds

Fix an integer $L \geq 1$.

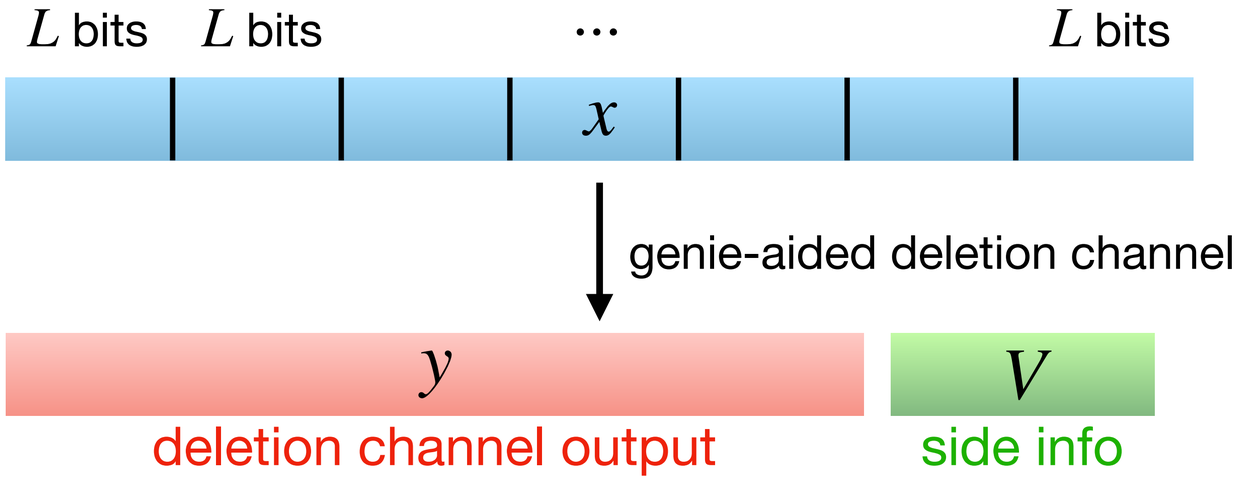


$V_i =$ nr. bits deleted in i-th block

$V = (V_1, V_2, \ldots, V_{n/L})$

The approach behind the best capacity upper bounds

Fix an integer $L \geq 1$.



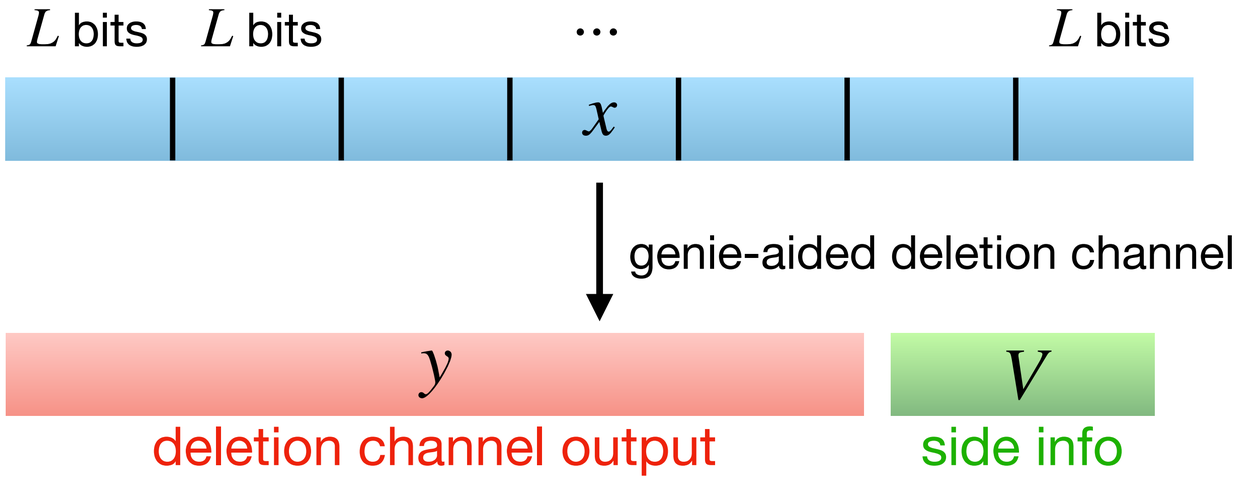
$V_i =$ nr. bits deleted in i-th block

$V = (V_1, V_2, \ldots, V_{n/L})$

For all $L \geq 1$, $\quad C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L).$

The approach behind the best capacity upper bounds

Fix an integer $L \geq 1$.



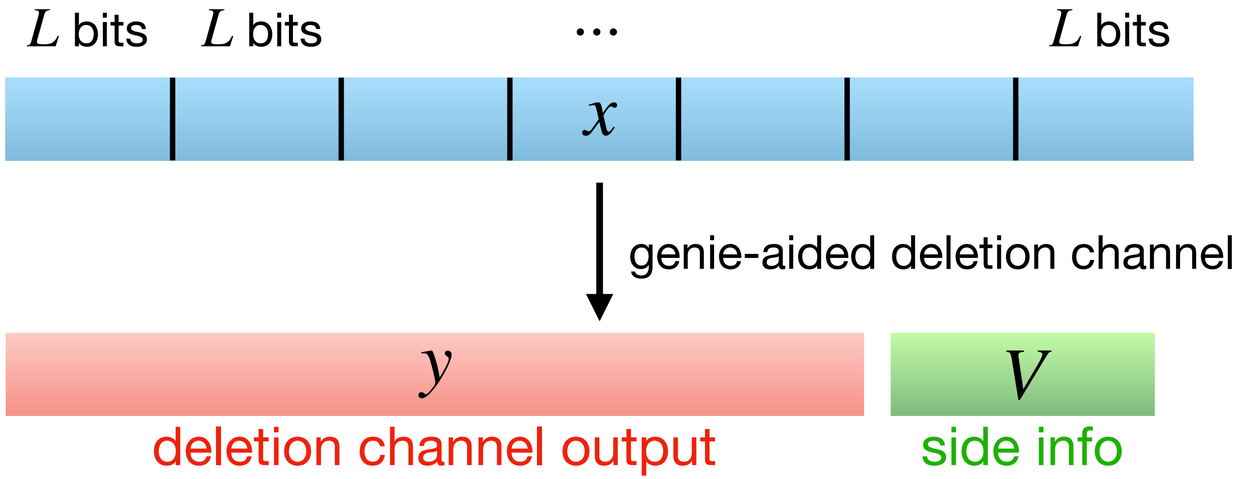
$V_i =$ nr. bits deleted in i-th block

$V = (V_1, V_2, \ldots, V_{n/L})$

For all $L \geq 1$, $\quad C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L).$

The approach behind the best capacity upper bounds

Fix an integer $L \geq 1$.



$V_i =$ nr. bits deleted in i-th block

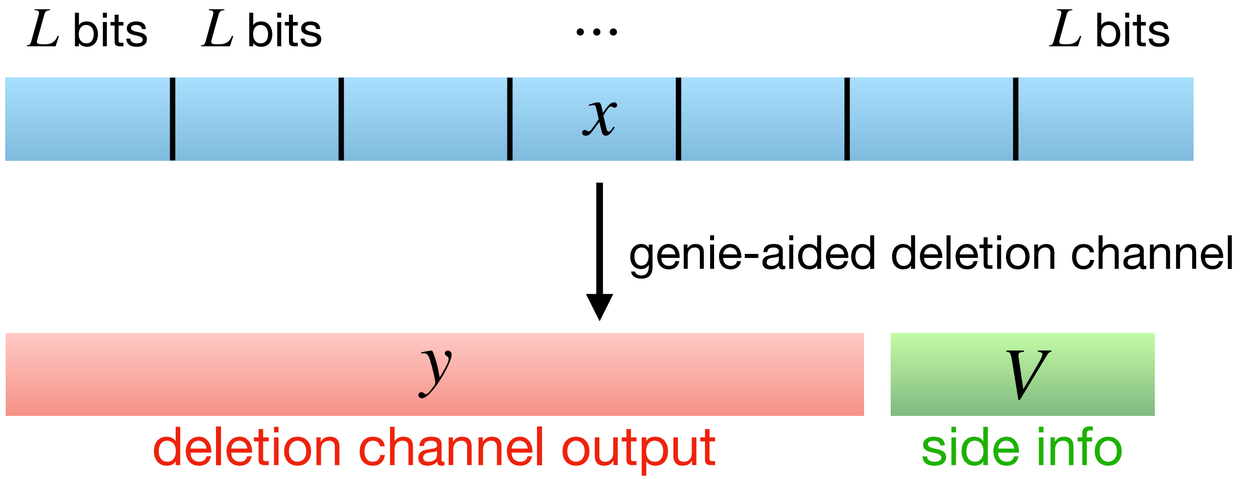
$V = (V_1, V_2, \ldots, V_{n/L})$

X^1

For all $L \geq 1$, $\quad C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L).$

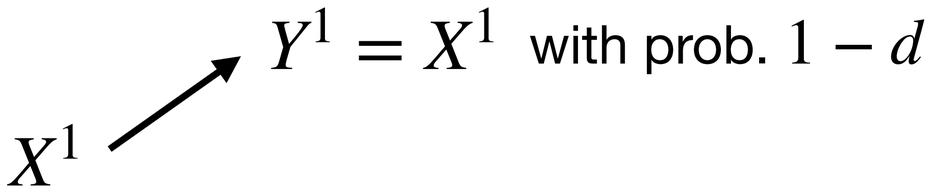
The approach behind the best capacity upper bounds

Fix an integer $L \geq 1$.



$V_i =$ nr. bits deleted in i-th block

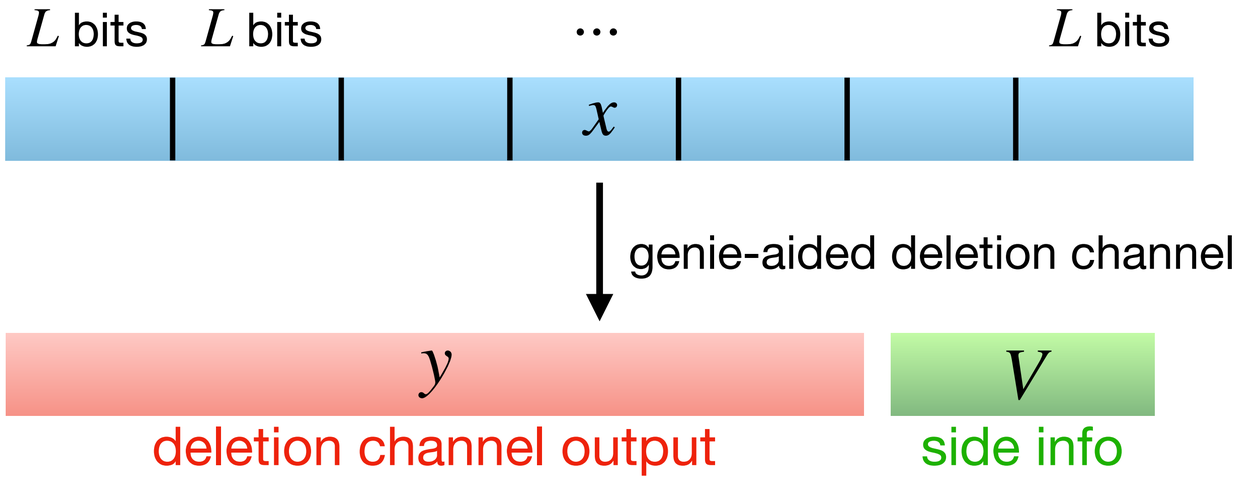
$V = (V_1, V_2, \ldots, V_{n/L})$



For all $L \geq 1$, $\quad C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L).$

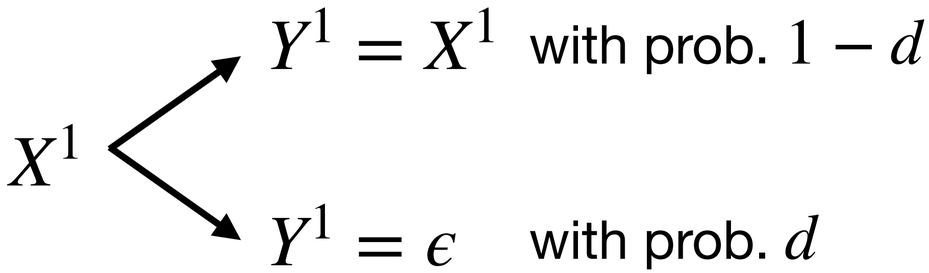
The approach behind the best capacity upper bounds

Fix an integer $L \geq 1$.



$V_i =$ nr. bits deleted in i-th block

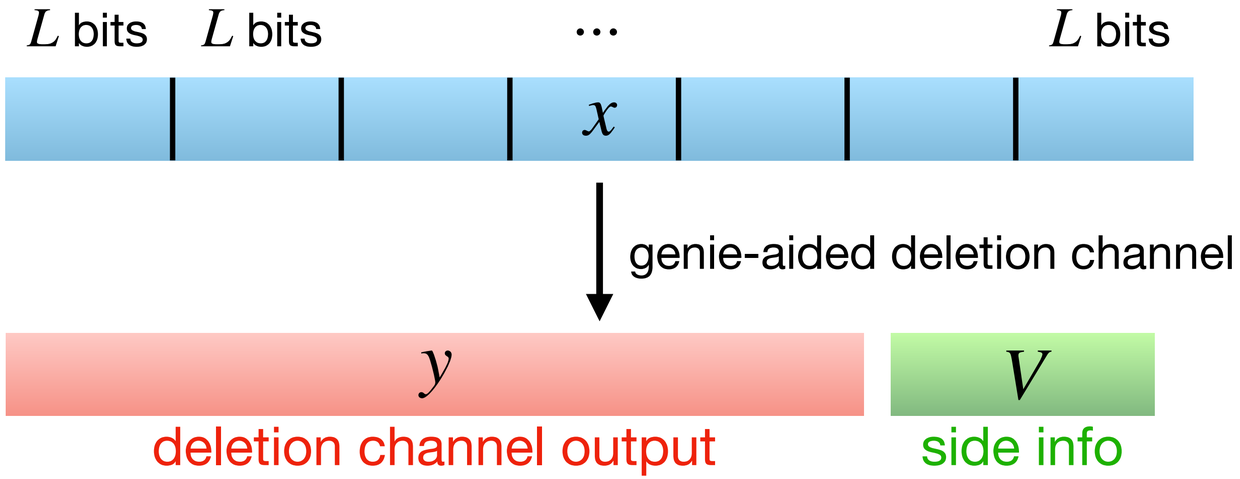
$V = (V_1, V_2, \ldots, V_{n/L})$



For all $L \geq 1$, $\quad C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L).$

The approach behind the best capacity upper bounds

Fix an integer $L \geq 1$.



$V_i =$ nr. bits deleted in i-th block

$V = (V_1, V_2, \ldots, V_{n/L})$

$$X^1 \begin{cases} Y^1 = X^1 & \text{with prob. } 1-d \\ Y^1 = \epsilon & \text{with prob. } d \end{cases}$$

$$\implies C(d) \leq \sup_{X^1} I(X^1; Y^1) = 1 - d$$

For all $L \geq 1$, $\quad C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L).$

The approach behind the best capacity upper bounds

In fact, something stronger holds:

$$C(d) = \inf_{L \geq 1} \frac{1}{L} \sup_{X^L} I(X^L; Y^L)$$

The approach behind the best capacity upper bounds

In fact, something stronger holds:

$$C(d) = \inf_{L \geq 1} \frac{1}{L} \sup_{X^L} I(X^L; Y^L)$$

Why this is useful:

The approach behind the best capacity upper bounds

In fact, something stronger holds:

$$C(d) = \inf_{L \geq 1} \frac{1}{L} \sup_{X^L} I(X^L; Y^L)$$

Why this is useful:

- For fixed L, we get a DMC with input alphabet $\{0,1\}^L$ and output alphabet $\{0,1\}^{\leq L}$;

The approach behind the best capacity upper bounds

In fact, something stronger holds:

$$C(d) = \inf_{L \geq 1} \frac{1}{L} \sup_{X^L} I(X^L; Y^L)$$

Why this is useful:

- For fixed L, we get a DMC with input alphabet $\{0,1\}^L$ and output alphabet $\{0,1\}^{\leq L}$;
- Can throw the Blahut-Arimoto algorithm at this DMC to approximate its capacity;

The approach behind the best capacity upper bounds

In fact, something stronger holds:

$$C(d) = \inf_{L \geq 1} \frac{1}{L} \sup_{X^L} I(X^L; Y^L)$$

Why this is useful:

- For fixed L, we get a DMC with input alphabet $\{0,1\}^L$ and output alphabet $\{0,1\}^{\leq L}$;
- Can throw the Blahut-Arimoto algorithm at this DMC to approximate its capacity;
- In principle, by taking L large enough we can get as close to $C(d)$ as we'd like!

The approach behind the best capacity upper bounds

In fact, something stronger holds:

$$C(d) = \inf_{L \geq 1} \frac{1}{L} \sup_{X^L} I(X^L; Y^L)$$

Why this is useful:

- For fixed L, we get a DMC with input alphabet $\{0,1\}^L$ and output alphabet $\{0,1\}^{\leq L}$;
- Can throw the Blahut-Arimoto algorithm at this DMC to approximate its capacity;
- In principle, by taking L large enough we can get as close to $C(d)$ as we'd like!

The approach behind the best capacity upper bounds

There's also a bound that doesn't need to be computed from scratch when d changes.

The approach behind the best capacity upper bounds

There's also a bound that doesn't need to be computed from scratch when d changes.

$$C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L) \leq \sum_{k=0}^{L} \binom{L}{k} d^{L-k}(1-d)^k \cdot C_{L,k}$$

The approach behind the best capacity upper bounds

There's also a bound that doesn't need to be computed from scratch when d changes.

$$C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L) \leq \sum_{k=0}^{L} \binom{L}{k} d^{L-k}(1-d)^k \cdot C_{L,k}$$

probability that exactly $L - k$ deletions occur

The approach behind the best capacity upper bounds

There's also a bound that doesn't need to be computed from scratch when d changes.

$$C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L) \leq \sum_{k=0}^{L} \binom{L}{k} d^{L-k}(1-d)^k \cdot C_{L,k}$$

capacity of channel that deletes **exactly** $L-k$ bits at random from L-bit input

probability that exactly $L-k$ deletions occur

The approach behind the best capacity upper bounds

There's also a bound that doesn't need to be computed from scratch when d changes.

$$C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L) \leq \sum_{k=0}^{L} \binom{L}{k} d^{L-k}(1-d)^k \cdot C_{L,k}$$

capacity of channel that deletes **exactly** $L-k$ bits at random from L-bit input

probability that exactly $L-k$ deletions occur

Why this is useful:

The approach behind the best capacity upper bounds

There's also a bound that doesn't need to be computed from scratch when d changes.

$$C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L) \leq \sum_{k=0}^{L} \binom{L}{k} d^{L-k}(1-d)^k \cdot C_{L,k}$$

capacity of channel that deletes **exactly** $L-k$ bits at random from L-bit input

probability that exactly $L-k$ deletions occur

Why this is useful:

- For fixed L and k, we get a DMC with input alphabet $\{0,1\}^L$ and output alphabet $\{0,1\}^k$;

The approach behind the best capacity upper bounds

There's also a bound that doesn't need to be computed from scratch when d changes.

$$C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L) \leq \sum_{k=0}^{L} \binom{L}{k} d^{L-k}(1-d)^k \cdot C_{L,k}$$

capacity of channel that deletes **exactly** $L-k$ bits at random from L-bit input

probability that exactly $L-k$ deletions occur

Why this is useful:

- For fixed L and k, we get a DMC with input alphabet $\{0,1\}^L$ and output alphabet $\{0,1\}^k$;
- Can throw the Blahut-Arimoto algorithm at this DMC to get great upper bounds on $C_{L,k}$;

The approach behind the best capacity upper bounds

There's also a bound that doesn't need to be computed from scratch when d changes.

$$C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L) \leq \sum_{k=0}^{L} \binom{L}{k} d^{L-k}(1-d)^k \cdot C_{L,k}$$

capacity of channel that deletes **exactly** $L-k$ bits at random from L-bit input

probability that exactly $L-k$ deletions occur

Why this is useful:

- For fixed L and k, we get a DMC with input alphabet $\{0,1\}^L$ and output alphabet $\{0,1\}^k$;
- Can throw the Blahut-Arimoto algorithm at this DMC to get great upper bounds on $C_{L,k}$;
- Getting great upper bounds on *selected* $C_{L,k}$'s suffices for good upper bounds on $C(d)$.

The approach behind the best capacity upper bounds

There's also a bound that doesn't need to be computed from scratch when d changes.

$$C(d) \leq \frac{1}{L} \sup_{X^L} I(X^L; Y^L) \leq \sum_{k=0}^{L} \binom{L}{k} d^{L-k}(1-d)^k \cdot C_{L,k}$$

capacity of channel that deletes **exactly** $L-k$ bits at random from L-bit input

probability that exactly $L-k$ deletions occur

Why this is useful:

- For fixed L and k, we get a DMC with input alphabet $\{0,1\}^L$ and output alphabet $\{0,1\}^k$;
- Can throw the Blahut-Arimoto algorithm at this DMC to get great upper bounds on $C_{L,k}$;
- Getting great upper bounds on *selected* $C_{L,k}$'s suffices for good upper bounds on $C(d)$.

The bottleneck

For a fixed L and k we want to approximate the capacity $C_{L,k}$ of a DMC with:

- An input alphabet of size 2^L, representing L-bit strings;
- An output alphabet of size 2^k, representing k-bit strings.

The bottleneck

For a fixed L and k we want to approximate the capacity $C_{L,k}$ of a DMC with:

- An input alphabet of size 2^L, representing L-bit strings;
- An output alphabet of size 2^k, representing k-bit strings.

So the Blahut-Arimoto algorithm will require time and space exponential in L…

The bottleneck

For a fixed L and k we want to approximate the capacity $C_{L,k}$ of a DMC with:

- An input alphabet of size 2^L, representing L-bit strings;
- An output alphabet of size 2^k, representing k-bit strings.

So the Blahut-Arimoto algorithm will require time and space exponential in L…

Prior work:

Fertonani-Duman 2010	Computed all $C_{L,k}$'s for $L = 17$ and some $C_{L,k}$'s up to $L = 22$.
Rubinstein-Con 2023	Computed all $C_{L,k}$'s for $L = 22$ and some $C_{L,k}$'s up to $L = 28$.

The bottleneck

For a fixed L and k we want to approximate the capacity $C_{L,k}$ of a DMC with:

- An input alphabet of size 2^L, representing L-bit strings;
- An output alphabet of size 2^k, representing k-bit strings.

So the Blahut-Arimoto algorithm will require time and space exponential in L…

Prior work:

Fertonani-Duman 2010	Computed all $C_{L,k}$'s for $L = 17$ and some $C_{L,k}$'s up to $L = 22$.
Rubinstein-Con 2023	Computed all $C_{L,k}$'s for $L = 22$ and some $C_{L,k}$'s up to $L = 28$.

Improved BAA implementation via time-memory tradeoffs for computing channel transition probabilities

A bird’s eye view of our work

A bird’s eye view of our work

- Look at the Blahut-Arimoto algorithm;

A bird’s eye view of our work

- Look at the Blahut-Arimoto algorithm;
- Leverage very simple observations to develop a parallelized implementation;

A bird's eye view of our work

- Look at the Blahut-Arimoto algorithm;
- Leverage very simple observations to develop a parallelized implementation;
- Compute all $C_{L,k}$'s up to $L = 29$, and some up to $L = 31$;

A bird's eye view of our work

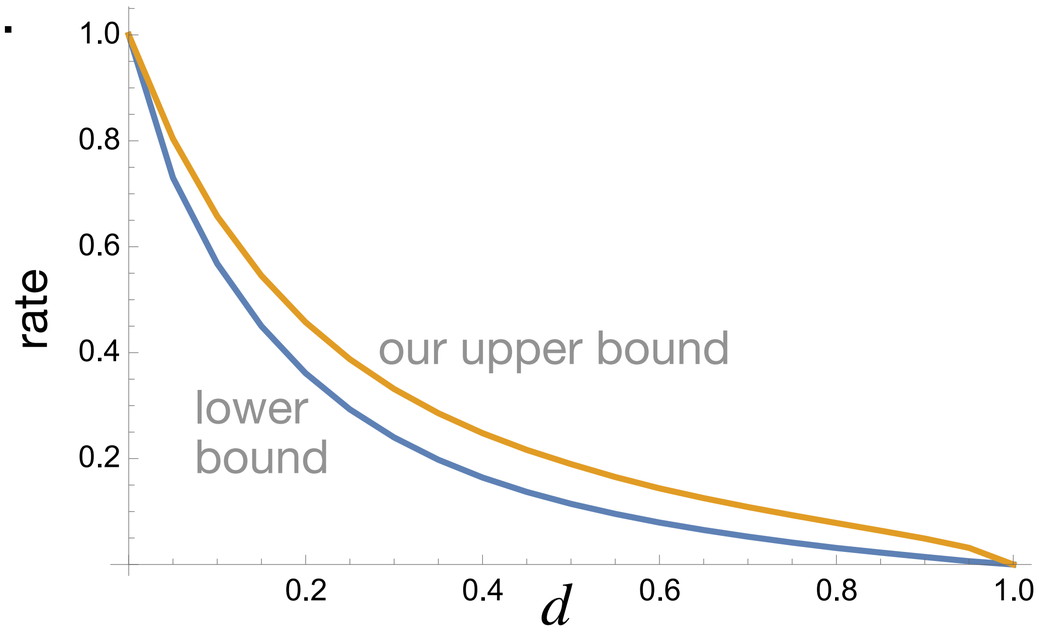
- Look at the Blahut-Arimoto algorithm;
- Leverage very simple observations to develop a parallelized implementation;
- Compute all $C_{L,k}$'s up to $L = 29$, and some up to $L = 31$;
- This took ~1 week using a consumer-grade RTX 5070 Ti GPU;

A bird's eye view of our work

- Look at the Blahut-Arimoto algorithm;
- Leverage very simple observations to develop a parallelized implementation;
- Compute all $C_{L,k}$'s up to $L = 29$, and some up to $L = 31$;
- This took ~1 week using a consumer-grade RTX 5070 Ti GPU;
- Obtain state-of-the-art upper bounds on the capacity of the deletion channel ($\approx 5\%$ improvement around $d = 1/2$ over previous bounds).

A bird's eye view of our work

- Look at the Blahut-Arimoto algorithm;
- Leverage very simple observations to develop a parallelized implementation;
- Compute all $C_{L,k}$'s up to $L = 29$, and some up to $L = 31$;
- This took ~1 week using a consumer-grade RTX 5070 Ti GPU;
- Obtain state-of-the-art upper bounds on the capacity of the deletion channel ($\approx 5\%$ improvement around $d = 1/2$ over previous bounds).



How Blahut-Arimoto works

How Blahut-Arimoto works

Starting with any input distribution P_X, iteratively repeat the following steps:

How Blahut-Arimoto works

Starting with any input distribution P_X, iteratively repeat the following steps:

1. **Compute channel output distribution:**

How Blahut-Arimoto works

Starting with any input distribution P_X, iteratively repeat the following steps:

1. **Compute channel output distribution:**

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \,|\, x)$$

How Blahut-Arimoto works

Starting with any input distribution P_X, iteratively repeat the following steps:

1. **Compute channel output distribution:**

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \,|\, x)$$

2. **Refine input distribution:**

How Blahut-Arimoto works

Starting with any input distribution P_X, iteratively repeat the following steps:

1. **Compute channel output distribution:**

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \,|\, x)$$

2. **Refine input distribution:**

$$\text{for each } x \in \{0,1\}^L, \text{ compute } W(x) \propto \prod_y \left(\frac{P_X(x) P_{Y|X}(y)}{P_Y(y)} \right)^{P_{Y|X}(y|x)}$$

How Blahut-Arimoto works

Starting with any input distribution P_X, iteratively repeat the following steps:

1. **Compute channel output distribution:**

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \,|\, x)$$

2. **Refine input distribution:**

$$\text{for each } x \in \{0,1\}^L, \text{ compute } W(x) \propto \prod_y \left(\frac{P_X(x) P_{Y|X}(y)}{P_Y(y)} \right)^{P_{Y|X}(y|x)}$$

3. Repeat with W in place of P_X until “error” is small enough.

Parallelizing Blahut-Arimoto

Each step in an iteration can be parallelized in a very simple way!

Parallelizing Blahut-Arimoto

Parallelizing Blahut-Arimoto

- **Compute channel output distribution:**

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \,|\, x)$$

Parallelizing Blahut-Arimoto

- **Compute channel output distribution:**

iterates over all length-L supersequences of y

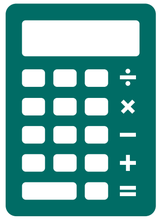
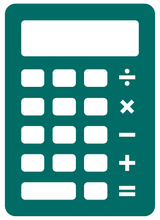
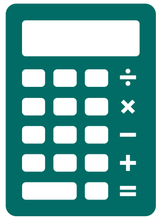
$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \,|\, x)$$

Parallelizing Blahut-Arimoto

- **Compute channel output distribution:**

iterates over all length-L supersequences of y

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \,|\, x)$$

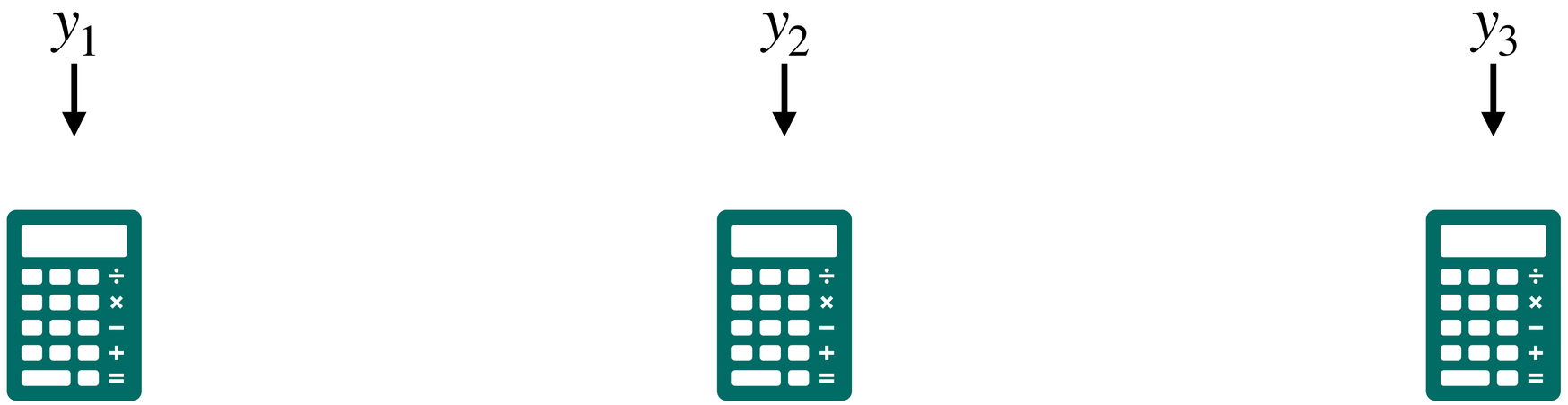




Parallelizing Blahut-Arimoto

- **Compute channel output distribution:**

iterates over all length-L supersequences of y

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \mid x)$$

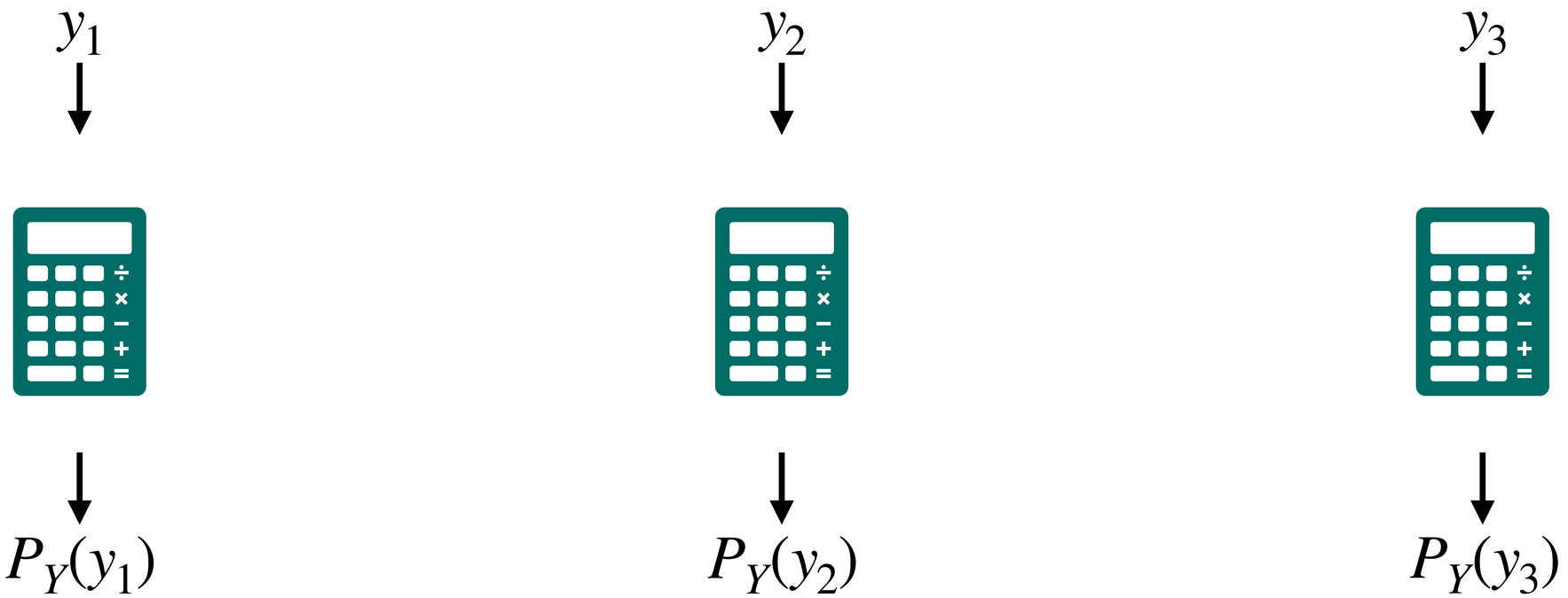


Parallelizing Blahut-Arimoto

- **Compute channel output distribution:**

iterates over all length-L supersequences of y

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \mid x)$$



Parallelizing Blahut-Arimoto

- **Compute channel output distribution:**

iterates over all length-L supersequences of y

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \,|\, x)$$

Parallelizing Blahut-Arimoto

- **Compute channel output distribution:**

iterates over all length-L supersequences of y

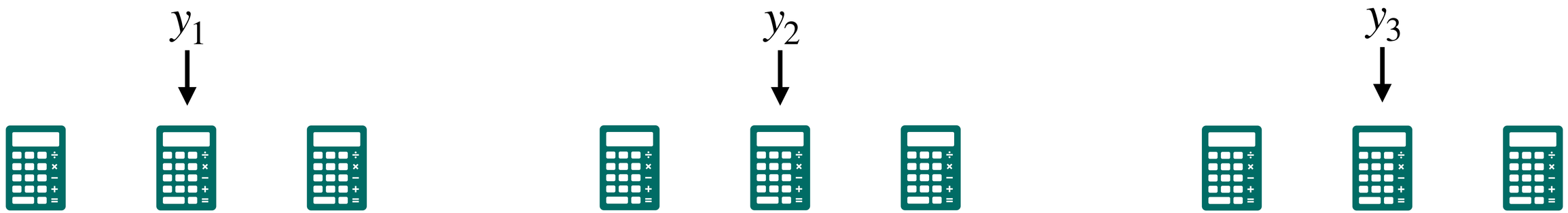
$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \,|\, x)$$

Parallelizing Blahut-Arimoto

- **Compute channel output distribution:**

iterates over all length-L supersequences of y

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \,|\, x)$$

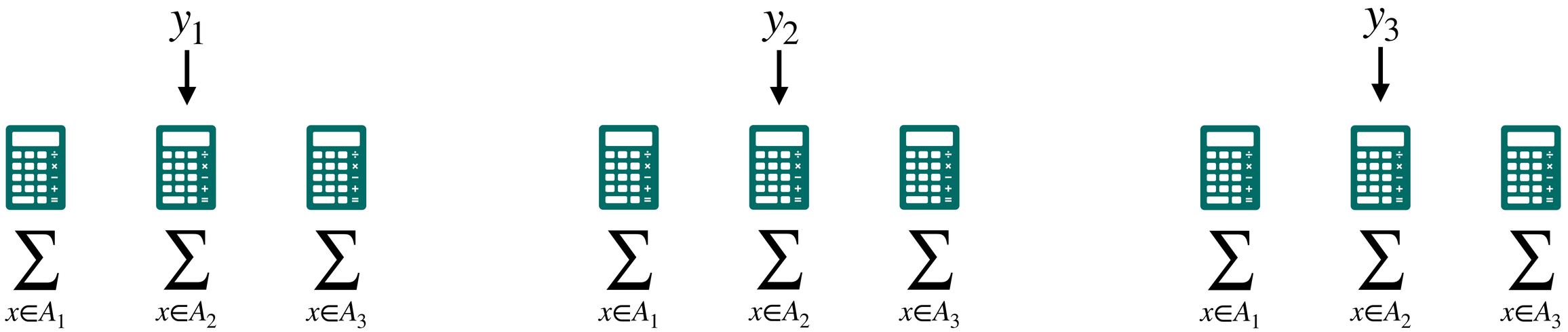


Parallelizing Blahut-Arimoto

- **Compute channel output distribution:**

iterates over all length-L supersequences of y

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \mid x)$$

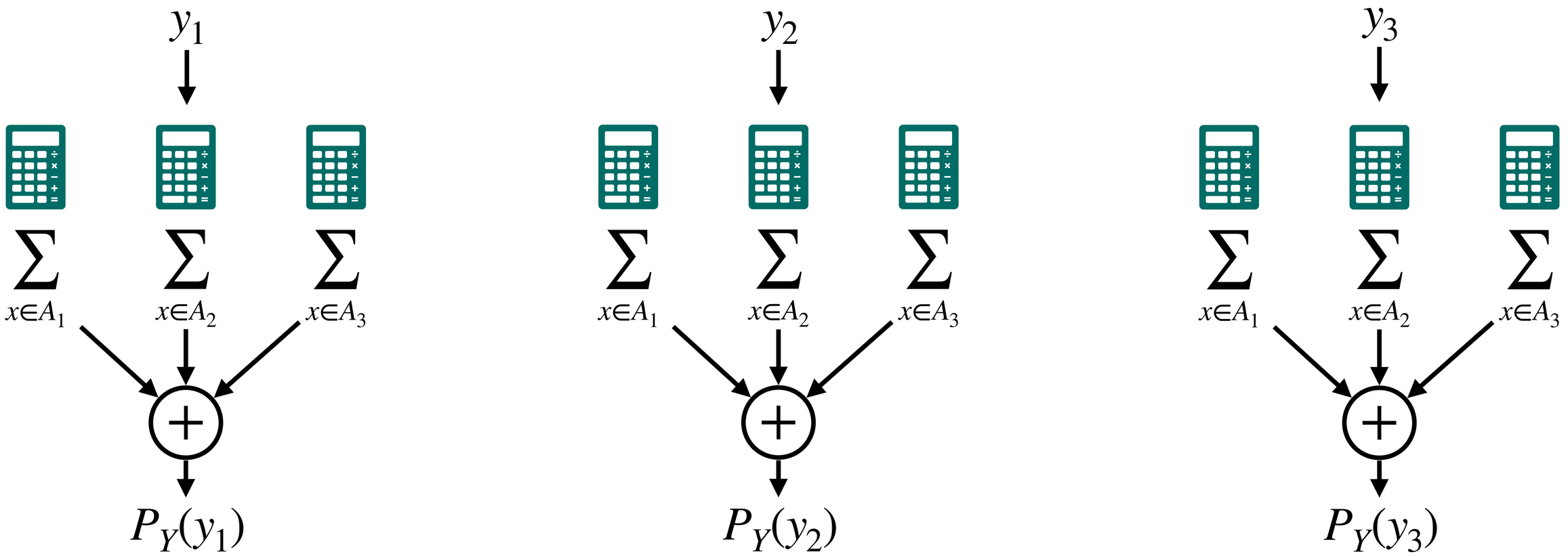


Parallelizing Blahut-Arimoto

- **Compute channel output distribution:**

iterates over all length-L supersequences of y

$$\text{for each } y \in \{0,1\}^k, \text{ compute } P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \mid x)$$



Parallelizing Blahut-Arimoto

- **Refine input distribution:**

iterates over all length-k subsequences of x

for each $x \in \{0,1\}^L$, compute $W(x) \propto \prod_y \left(\frac{P_X(x) P_{Y|X}(y)}{P_Y(y)} \right)^{P_{Y|X}(y|x)}$

Parallelizing Blahut-Arimoto

- **Refine input distribution:**

iterates over all length-k subsequences of x

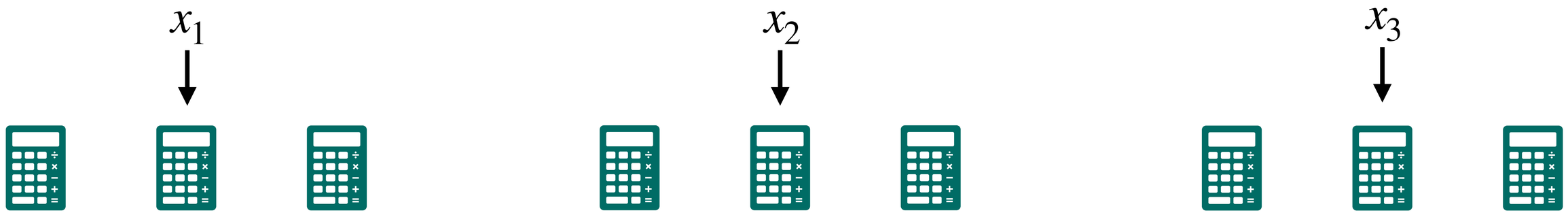
for each $x \in \{0,1\}^L$, compute $W(x) \propto \prod_y \left(\frac{P_X(x)P_{Y|X}(y)}{P_Y(y)} \right)^{P_{Y|X}(y|x)}$

Parallelizing Blahut-Arimoto

- **Refine input distribution:**

iterates over all length-k subsequences of x

$$\text{for each } x \in \{0,1\}^L, \text{ compute } W(x) \propto \prod_y \left(\frac{P_X(x) P_{Y|X}(y)}{P_Y(y)} \right)^{P_{Y|X}(y|x)}$$

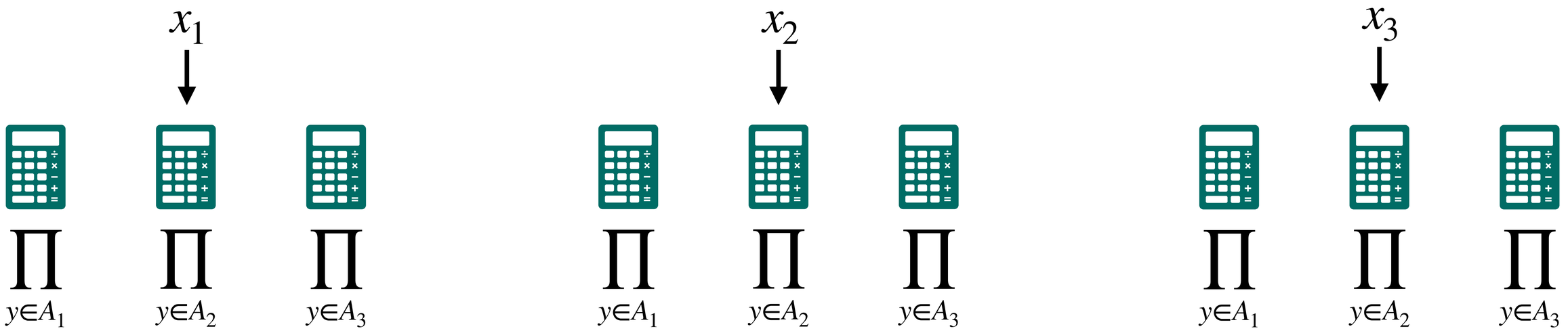


Parallelizing Blahut-Arimoto

- **Refine input distribution:**

iterates over all length-k subsequences of x

for each $x \in \{0,1\}^L$, compute $W(x) \propto \prod_y \left(\frac{P_X(x)P_{Y|X}(y)}{P_Y(y)} \right)^{P_{Y|X}(y|x)}$

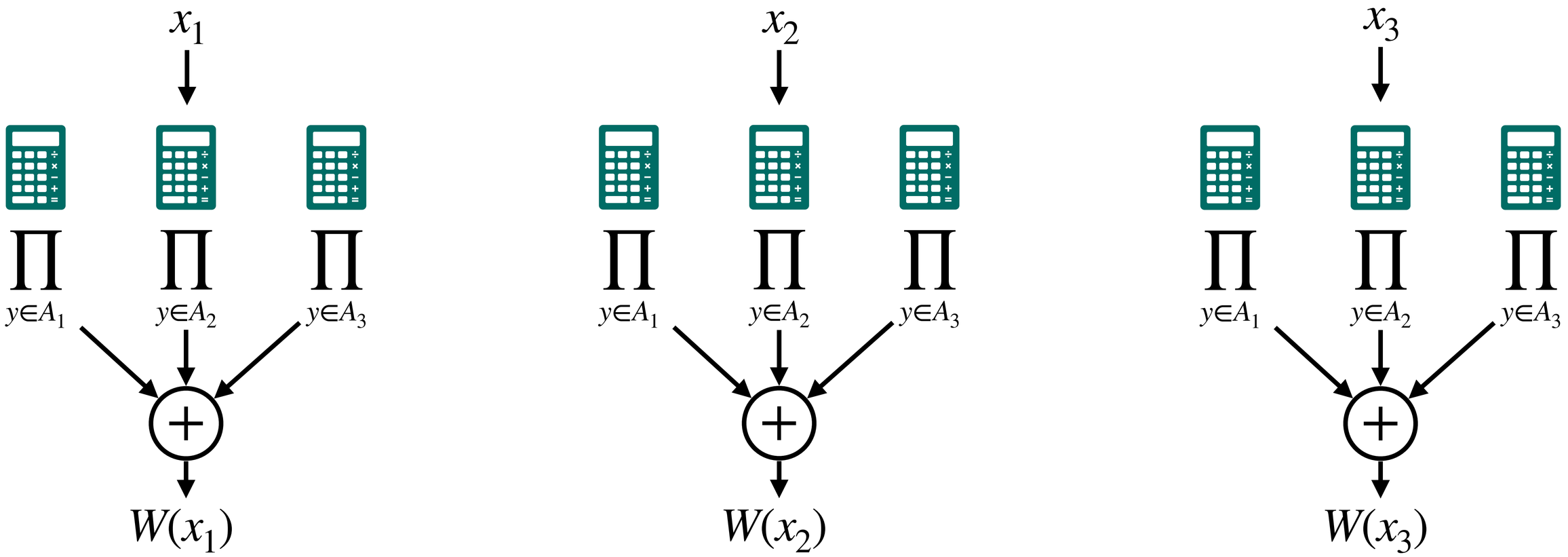


Parallelizing Blahut-Arimoto

- **Refine input distribution:**

iterates over all length-k subsequences of x

for each $x \in \{0,1\}^L$, compute $W(x) \propto \prod_y \left(\frac{P_X(x)P_{Y|X}(y)}{P_Y(y)} \right)^{P_{Y|X}(y|x)}$



Parallelizing Blahut-Arimoto

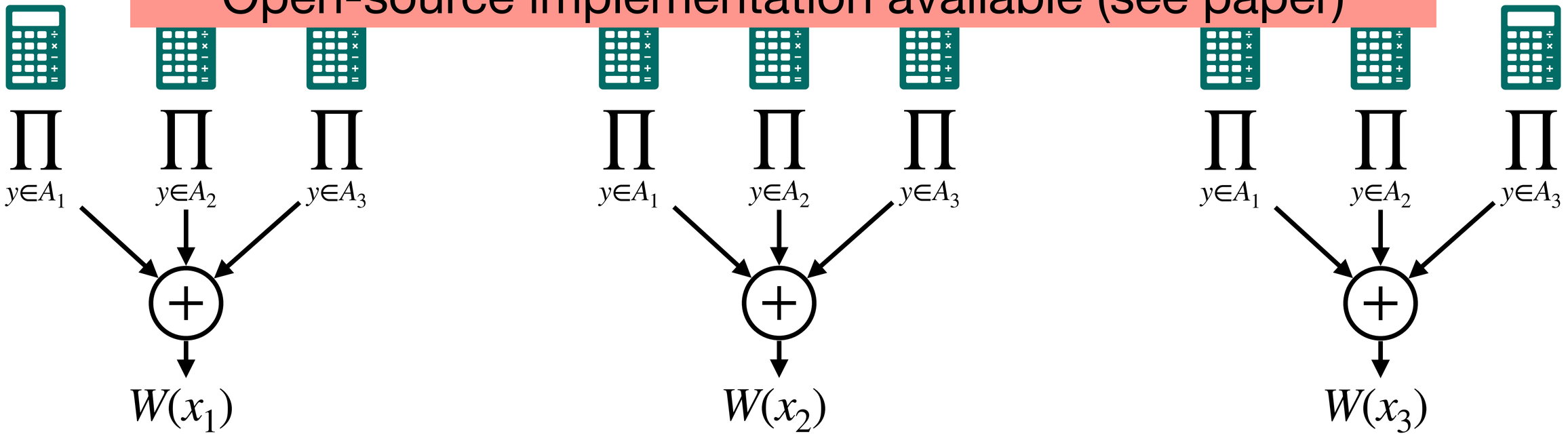
- **Refine input distribution:**

iterates over all length-k subsequences of x

$$\text{for each } x \in \{0,1\}^L, \text{ compute } W(x) \propto \prod_y \left(\frac{P_X(x)P_{Y|X}(y)}{P_Y(y)} \right)^{P_{Y|X}(y|x)}$$

These steps fit nicely into GPUs!

Open-source implementation available (see paper)

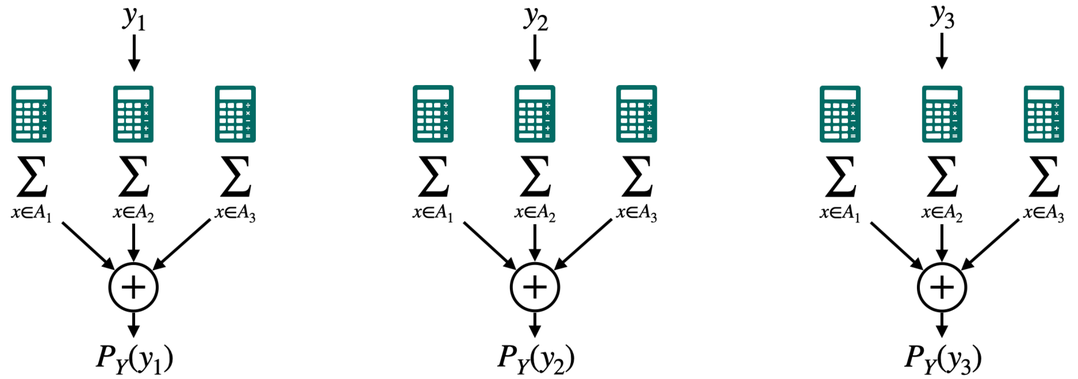


Some care is needed

- **Compute channel output distribution:**

for each $y \in \{0,1\}^k$, compute $P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \mid x)$

iterates over all length-L supersequences of y

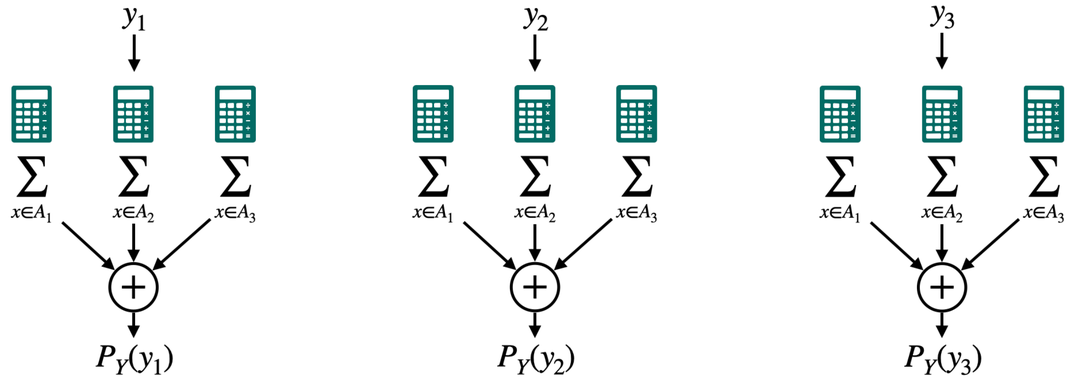


Some care is needed

- **Compute channel output distribution:**

for each $y \in \{0,1\}^k$, compute $P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \mid x)$

iterates over all length-L supersequences of y



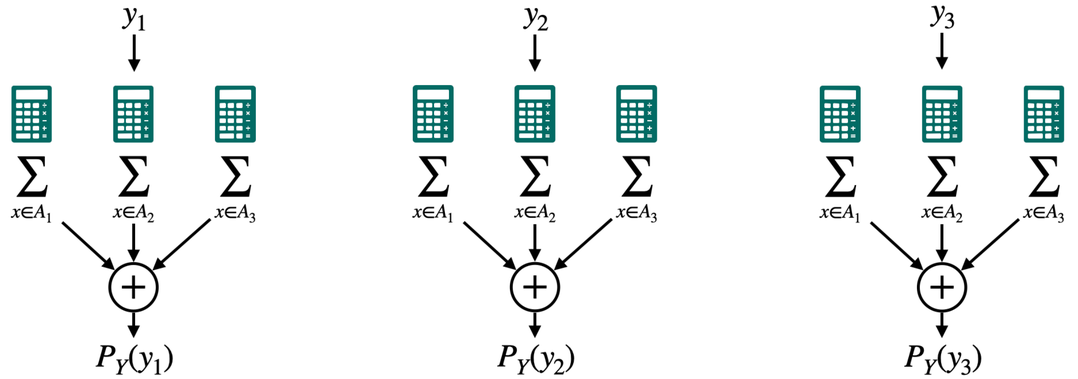
- Each thread enumerates over subset A_i of (indices of) supersequences of some y_j ;

Some care is needed

- **Compute channel output distribution:**

for each $y \in \{0,1\}^k$, compute $P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \mid x)$

iterates over all length-L supersequences of y



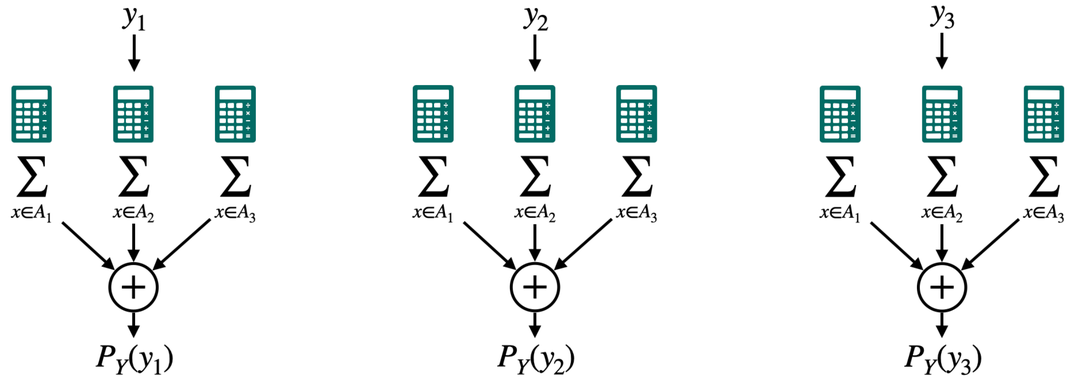
- Each thread enumerates over subset A_i of (indices of) supersequences of some y_j ;
- Threads have constrained memory, so can't rely on large look-up table;

Some care is needed

- **Compute channel output distribution:**

for each $y \in \{0,1\}^k$, compute $P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \mid x)$

iterates over all length-L supersequences of y



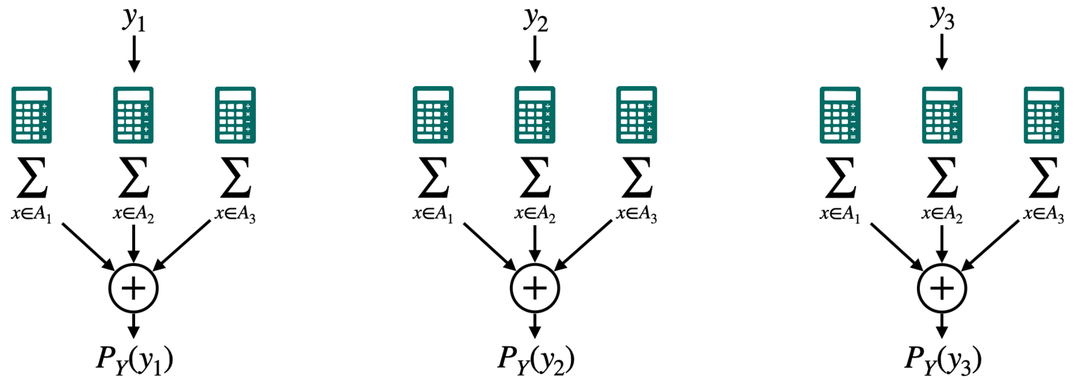
- Each thread enumerates over subset A_i of (indices of) supersequences of some y_j ;
- Threads have constrained memory, so can't rely on large look-up table;
- $A_i = \{i, i + 1024, i + 2 \cdot 1024, \ldots\}$

Some care is needed

- **Compute channel output distribution:**

for each $y \in \{0,1\}^k$, compute $P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \mid x)$

iterates over all length-L supersequences of y



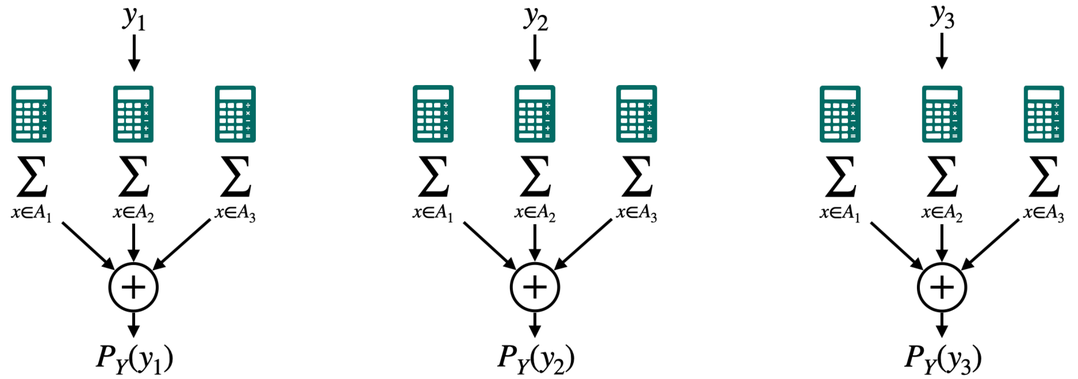
- Each thread enumerates over subset A_i of (indices of) supersequences of some y_j ;
- Threads have constrained memory, so can't rely on large look-up table;
- $A_i = \{i, i + 1024, i + 2 \cdot 1024, \ldots\}$
- Enumerate supersequences via dynamic programming;

Some care is needed

- **Compute channel output distribution:**

for each $y \in \{0,1\}^k$, compute $P_Y(y) = \sum_x P_X(x) \cdot P_{Y|X}(y \mid x)$

iterates over all length-L supersequences of y



- Each thread enumerates over subset A_i of (indices of) supersequences of some y_j ;
- Threads have constrained memory, so can't rely on large look-up table;
- $A_i = \{i, i + 1024, i + 2 \cdot 1024, \ldots\}$
- Enumerate supersequences via dynamic programming;
- Exploit basic symmetries of the deletion channel to save a bit of time & space (not new).

The asymptotics of the deletion channel capacity

Natural starting point for understanding the capacity.

The asymptotics of the deletion channel capacity

Natural starting point for understanding the capacity.

When $d \approx 0$, we know that $C(d) \approx 1 - h(d)$.
[Kalai-Mitzenmacher-Sudan, Kanoria-Montanari]

The asymptotics of the deletion channel capacity

Natural starting point for understanding the capacity.

When $d \approx 0$, we know that $C(d) \approx 1 - h(d)$.
[Kalai-Mitzenmacher-Sudan, Kanoria-Montanari]

When $d \approx 1$, we know that $\lim_{d \to 1} \frac{C(d)}{1-d} = \inf_{d \in [0,1)} \frac{C(d)}{1-d} = c^\star$ for some constant $c^\star$.
[Dalai, Fertonani-Duman, Rahmati-Duman]

The asymptotics of the deletion channel capacity

Natural starting point for understanding the capacity.

When $d \approx 0$, we know that $C(d) \approx 1 - h(d)$.
[Kalai-Mitzenmacher-Sudan, Kanoria-Montanari]

When $d \approx 1$, we know that $\lim_{d \to 1} \frac{C(d)}{1-d} = \inf_{d \in [0,1)} \frac{C(d)}{1-d} = c^\star$ for some constant $c^\star$.
[Dalai, Fertonani-Duman, Rahmati-Duman]

We knew that $0.1221 \leq c^\star \leq 0.3745$.

The asymptotics of the deletion channel capacity

Natural starting point for understanding the capacity.

When $d \approx 0$, we know that $C(d) \approx 1 - h(d)$.
[Kalai-Mitzenmacher-Sudan, Kanoria-Montanari]

When $d \approx 1$, we know that $\lim_{d \to 1} \frac{C(d)}{1-d} = \inf_{d \in [0,1)} \frac{C(d)}{1-d} = c^\star$ for some constant $c^\star$.
[Dalai, Fertonani-Duman, Rahmati-Duman]

We knew that $0.1221 \leq c^\star \leq 0.3745$.

Our results imply that $c^\star \leq 0.3578$.

What’s next?

What’s next?

- How far are we from the true capacity?

What's next?

- How far are we from the true capacity?
  - E.g., we get $C(1/2) \leq 0.1896$;

What's next?

- How far are we from the true capacity?
  - E.g., we get $C(1/2) \leq 0.1896$;
  - Best simulation-based lower bound is $C(1/2) \geq 0.1145$, via 3rd-order Markov chains. [Castiglione-Kavčić 2015]

What's next?

- How far are we from the true capacity?
  - E.g., we get $C(1/2) \leq 0.1896$;
  - Best simulation-based lower bound is $C(1/2) \geq 0.1145$, via 3rd-order Markov chains. [Castiglione-Kavčić 2015]

- How far can we push this approach?

What's next?

- How far are we from the true capacity?
  - E.g., we get $C(1/2) \leq 0.1896$;
  - Best simulation-based lower bound is $C(1/2) \geq 0.1145$, via 3rd-order Markov chains. [Castiglione-Kavčić 2015]

- How far can we push this approach?

- Our improvements don't exploit much of the structure of the deletion channel:

What's next?

- How far are we from the true capacity?
  - E.g., we get $C(1/2) \leq 0.1896$;
  - Best simulation-based lower bound is $C(1/2) \geq 0.1145$, via 3rd-order Markov chains. [Castiglione-Kavčić 2015]

- How far can we push this approach?

- Our improvements don't exploit much of the structure of the deletion channel:
  - A better choice of the initial input distribution could save heavily on the number of iterations;

What's next?

- How far are we from the true capacity?
  - E.g., we get $C(1/2) \leq 0.1896$;
  - Best simulation-based lower bound is $C(1/2) \geq 0.1145$, via 3rd-order Markov chains. [Castiglione-Kavčić 2015]

- How far can we push this approach?

- Our improvements don't exploit much of the structure of the deletion channel:
  - A better choice of the initial input distribution could save heavily on the number of iterations;
  - This motivates understanding the structure of capacity-achieving distributions for exact deletion channels.

What's next?

- How far are we from the true capacity?
  - E.g., we get $C(1/2) \leq 0.1896$;
  - Best simulation-based lower bound is $C(1/2) \geq 0.1145$, via 3rd-order Markov chains. [Castiglione-Kavčić 2015]

- How far can we push this approach?

- Our improvements don't exploit much of the structure of the deletion channel:
  - A better choice of the initial input distribution could save heavily on the number of iterations;
  - This motivates understanding the structure of capacity-achieving distributions for exact deletion channels.
- On the other hand, our parallel implementation can be applied more broadly. Where else could it be useful? Maybe you know. :)

Wrapping up

Wrapping up

- Understanding the capacity of the deletion channel is a major research direction;

Wrapping up

- Understanding the capacity of the deletion channel is a major research direction;
- Capacity of deletion channel is upper bounded by capacity of any “fixed input length” deletion channel;

Wrapping up

- Understanding the capacity of the deletion channel is a major research direction;
- Capacity of deletion channel is upper bounded by capacity of any “fixed input length” deletion channel;
- In principle, can use Blahut-Arimoto to approximate these fixed-input-length capacities;

Wrapping up

- Understanding the capacity of the deletion channel is a major research direction;
- Capacity of deletion channel is upper bounded by capacity of any “fixed input length” deletion channel;
- In principle, can use Blahut-Arimoto to approximate these fixed-input-length capacities;
- But Blahut-Arimoto time/space complexity is exponential in input length L;

Wrapping up

- Understanding the capacity of the deletion channel is a major research direction;
- Capacity of deletion channel is upper bounded by capacity of any “fixed input length” deletion channel;
- In principle, can use Blahut-Arimoto to approximate these fixed-input-length capacities;
- But Blahut-Arimoto time/space complexity is exponential in input length L;
- We exploit simple properties of Blahut-Arimoto to parallelize it and fit it into GPUs;

Wrapping up

- Understanding the capacity of the deletion channel is a major research direction;
- Capacity of deletion channel is upper bounded by capacity of any “fixed input length” deletion channel;
- In principle, can use Blahut-Arimoto to approximate these fixed-input-length capacities;
- But Blahut-Arimoto time/space complexity is exponential in input length L;
- We exploit simple properties of Blahut-Arimoto to parallelize it and fit it into GPUs;
- We use this parallelized version to approximate these fixed-input-length capacities for larger L;

Wrapping up

- Understanding the capacity of the deletion channel is a major research direction;
- Capacity of deletion channel is upper bounded by capacity of any “fixed input length” deletion channel;
- In principle, can use Blahut-Arimoto to approximate these fixed-input-length capacities;
- But Blahut-Arimoto time/space complexity is exponential in input length L;
- We exploit simple properties of Blahut-Arimoto to parallelize it and fit it into GPUs;
- We use this parallelized version to approximate these fixed-input-length capacities for larger L;
- We get improved upper bounds on the capacity of the deletion channel.

Wrapping up

- Understanding the capacity of the deletion channel is a major research direction;
- Capacity of deletion channel is upper bounded by capacity of any “fixed input length” deletion channel;
- In principle, can use Blahut-Arimoto to approximate these fixed-input-length capacities;
- But Blahut-Arimoto time/space complexity is exponential in input length L;
- We exploit simple properties of Blahut-Arimoto to parallelize it and fit it into GPUs;
- We use this parallelized version to approximate these fixed-input-length capacities for larger L;
- We get improved upper bounds on the capacity of the deletion channel.

Thanks!