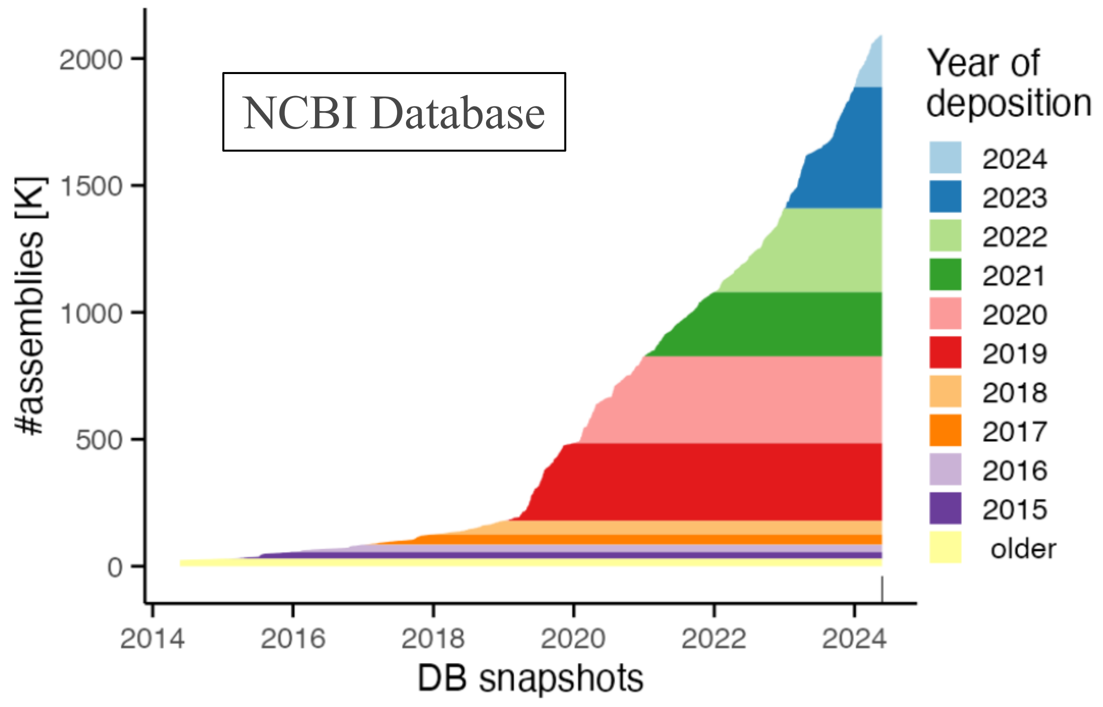
Bin Packing for Efficient Compression of Large Bacterial Genomes Collection

Tam Truong, Dominique Lavenier, Pierre Peterlongo, Karel Břinda

Introduction & State Of The Art

Motivation: Rapidly Growing Bacteria Genome Data

Fast Growth Of Bacterial Genomes Data[1]



Karel Brinda, 2024 (CC) https://doi.org/10.6084/m9.figshare.25879258

Increasing Availability of Larger Bacterial Genome Collections

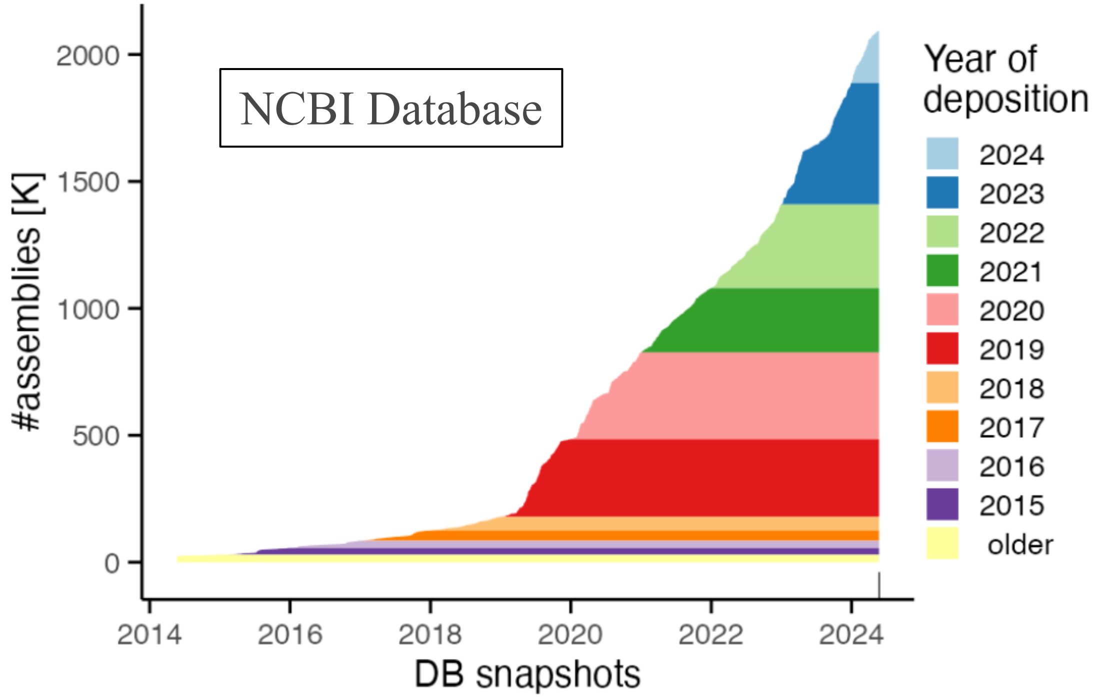
2021 Mar	661k Collection[2]	n = 661,405
2024 Mar	AllTheBacteria[3] v0.1	n = 1,932,812
2024 Nov	AllTheBacteria[3] v0.2	n = 2,440,377
Next decade	Collections	10^7

Moreover, improvements in biological and technological diversity, MAG,…

[1] Břinda et al., Efficient and Robust Search of Microbial Genomes via Phylogenetic Compression. To appear in *Nature Methods*. 2025
[2] Blackwell et al., Exploring bacterial diversity via a curated and searchable snapshot of archived DNA sequences. *PLOS Biology* 19, 11. 2021
[3] Hunt et a.l,. AllTheBacteria - all bacterial genomes assembled, available and searchable. *bioRxiv*. 2024

Motivation: Rapidly Growing Bacteria Genome Data

Fast Growth Of Bacterial Genomes Data[1]



Karel Brinda, 2024 (CC) https://doi.org/10.6084/m9.figshare.25879258

Increasing Availability of Larger Bacterial Genome Collections

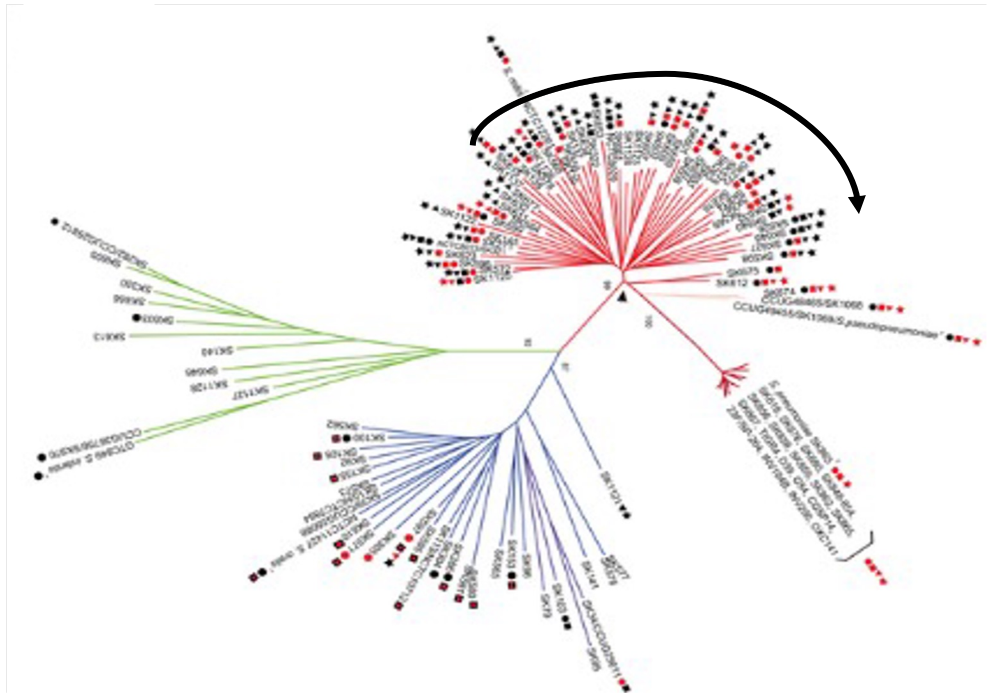
2021 Mar	661k Collection[2]	n = 661,405
2024 Mar	AllTheBacteria[3] v0.1	n = 1,932,812
2024 Nov	AllTheBacteria[3] v0.2	n = 2,440,377
Next decade	Collections	10^7

Moreover, improvements in biological and technological diversity, MAG,…

Challenging: efficient compression and search within those collections

[1] Břinda et al., Efficient and Robust Search of Microbial Genomes via Phylogenetic Compression. To appear in *Nature Methods*. 2025
[2] Blackwell et al., Exploring bacterial diversity via a curated and searchable snapshot of archived DNA sequences. *PLOS Biology* 19, 11. 2021
[3] Hunt et a.l,. AllTheBacteria - all bacterial genomes assembled, available and searchable. *bioRxiv*. 2024

Recent Innovation: Phylogenetic Compression

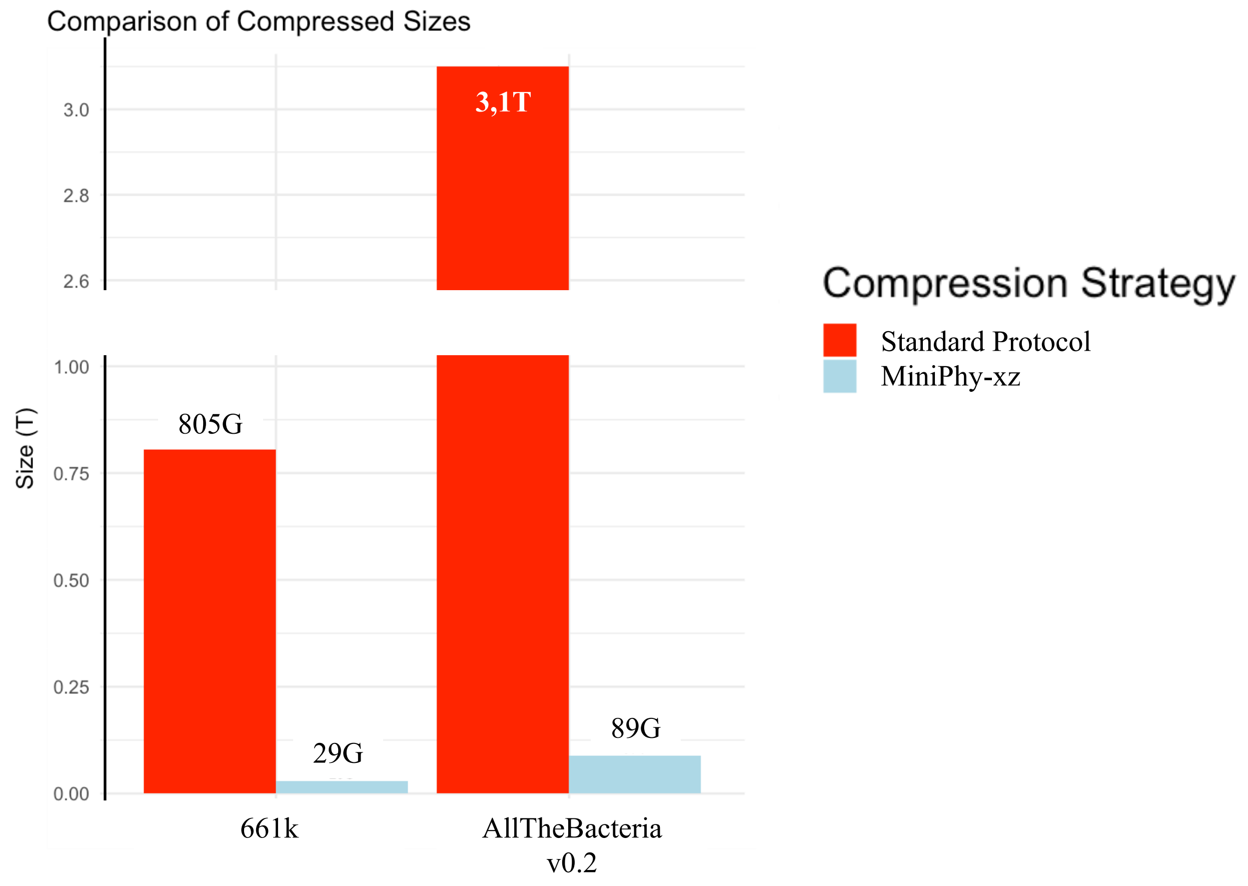


Difficulty: Compression of genomes is challenging due to the widespread redundancy in the data.

Key Idea: Reordering genomes based on evolutionary history enhances local compressibility[1]

[1] Břinda et al., Efficient and Robust Search of Microbial Genomes via Phylogenetic Compression. To be appeared in *Nature Methods*. 2025

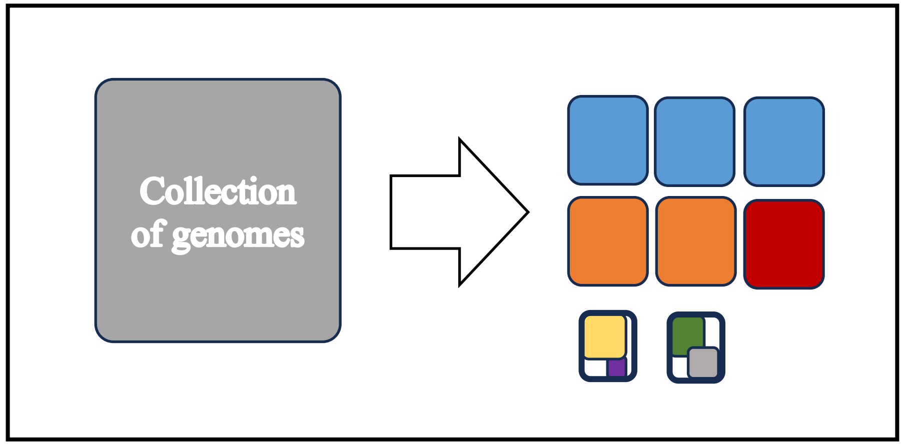
Resulting Improvement Using Phylogenetic Compression (MiniPhy[1])

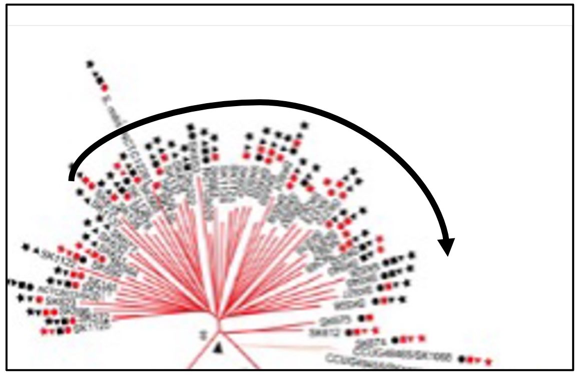


Lossless compression of 1-3 orders of magnitude over standard protocol

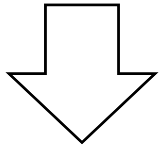
[1] https://github.com/karel-brinda/miniphy

In Practice: Phylogenetic compression involves two steps:





Step 1 : Phylogenetic Batching



Starting point of my PhD

Step 2 : Phylogenetic Reordering

Key Step In Phylogenetic Compression: Genomes Batching

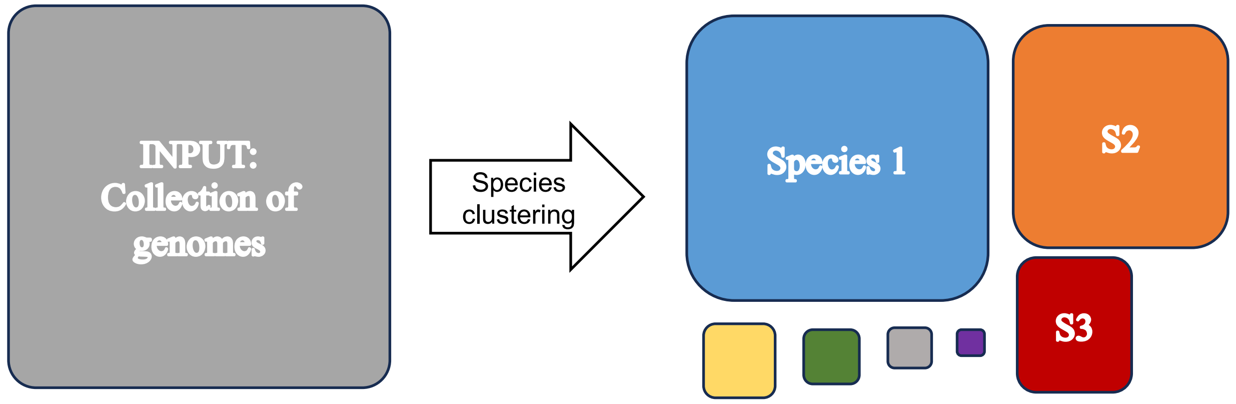
INPUT:
Collection of genomes

Key Step In Phylogenetic Compression: Genomes Batching

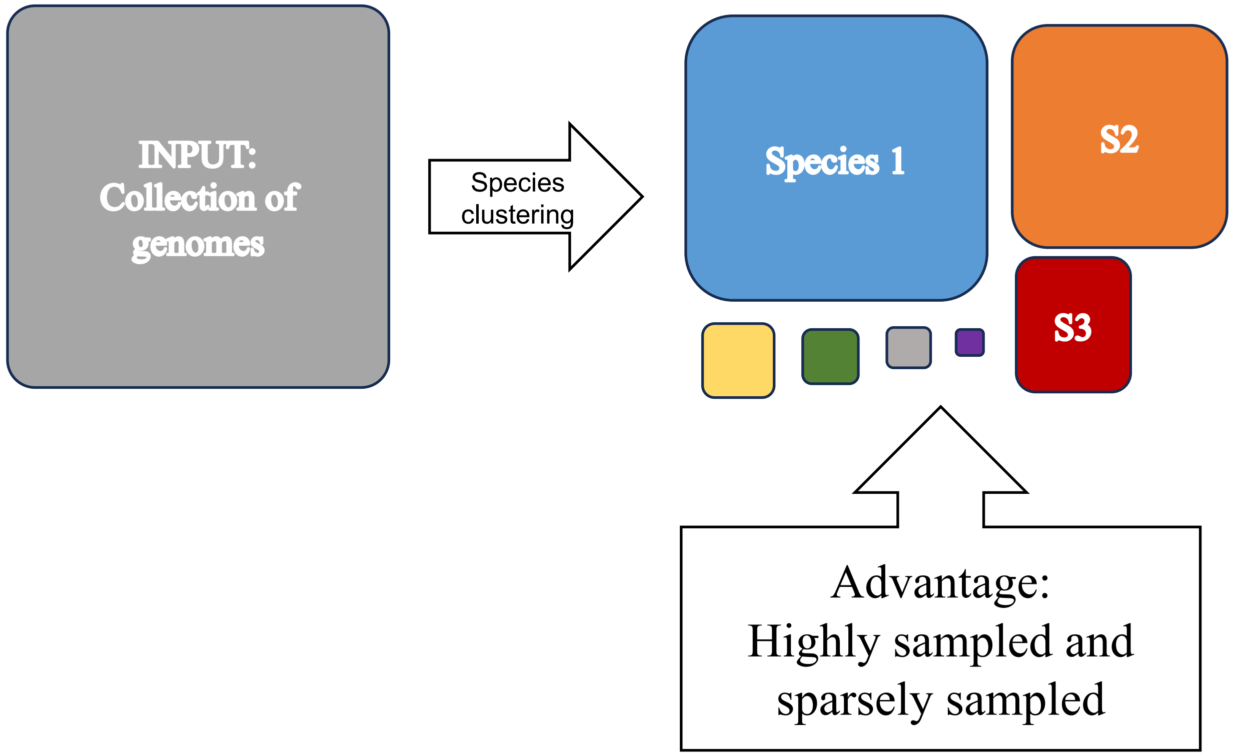
INPUT:
Collection of genomes

Species form clusters in public databases

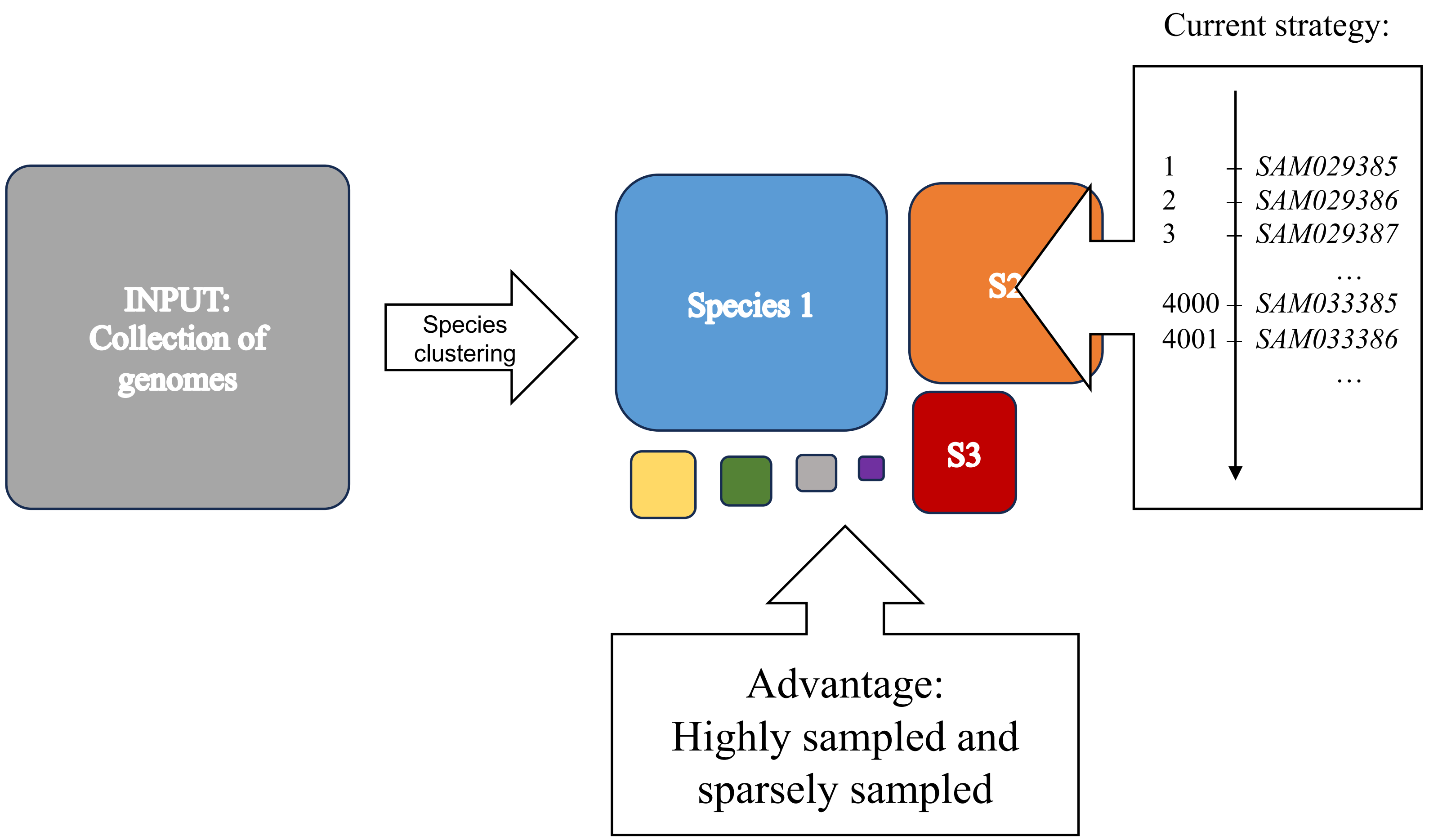
Key Step In Phylogenetic Compression: Genomes Batching



Key Step In Phylogenetic Compression: Genomes Batching

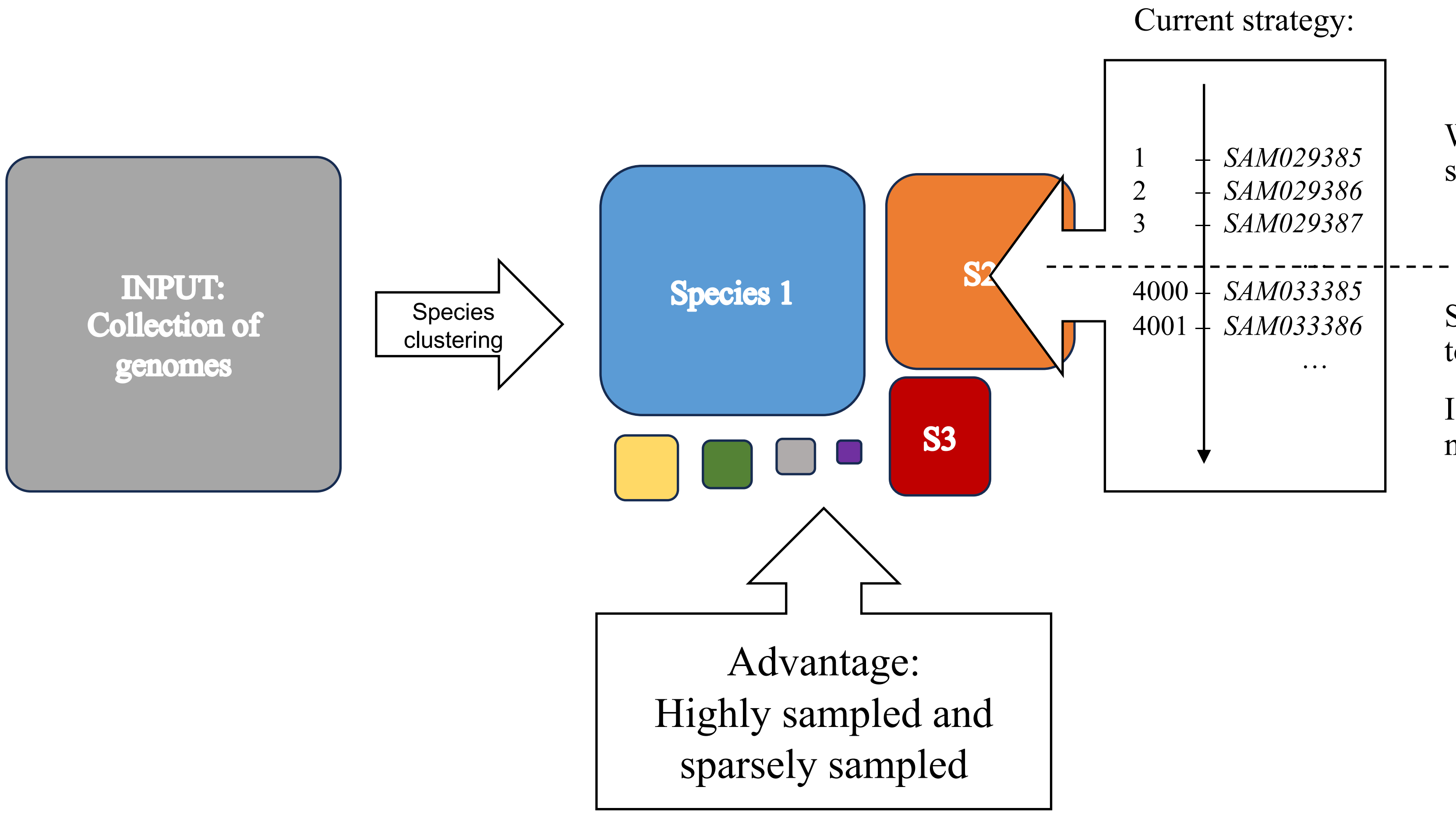


Key Step In Phylogenetic Compression: Genomes Batching



Within each cluster, sort by sequencing time (accession number)

Key Step In Phylogenetic Compression: Genomes Batching

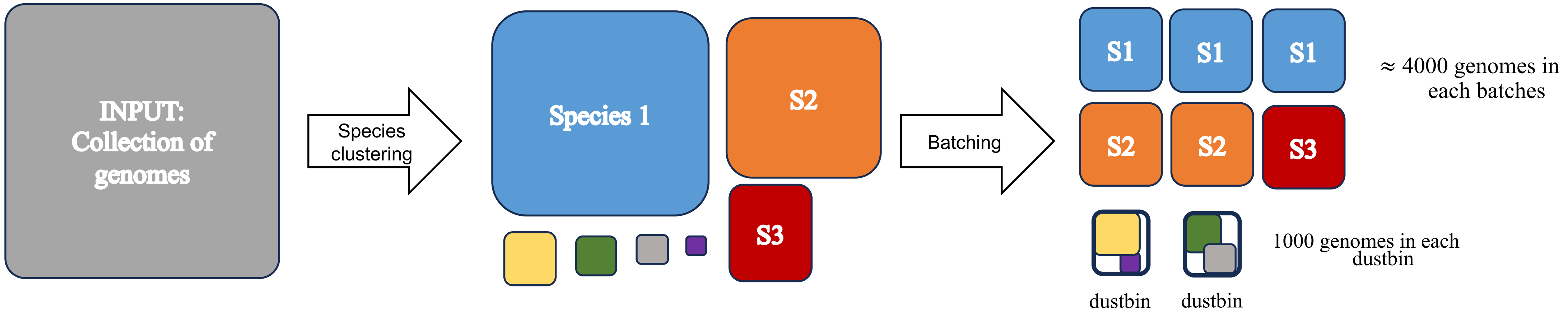


Within each cluster, sort by sequencing time (accession number)

Sequencing data is typically uploaded together.

I.e. genomes with nearby accession numbers are likely related.

Key Step In Phylogenetic Compression: Genomes Batching

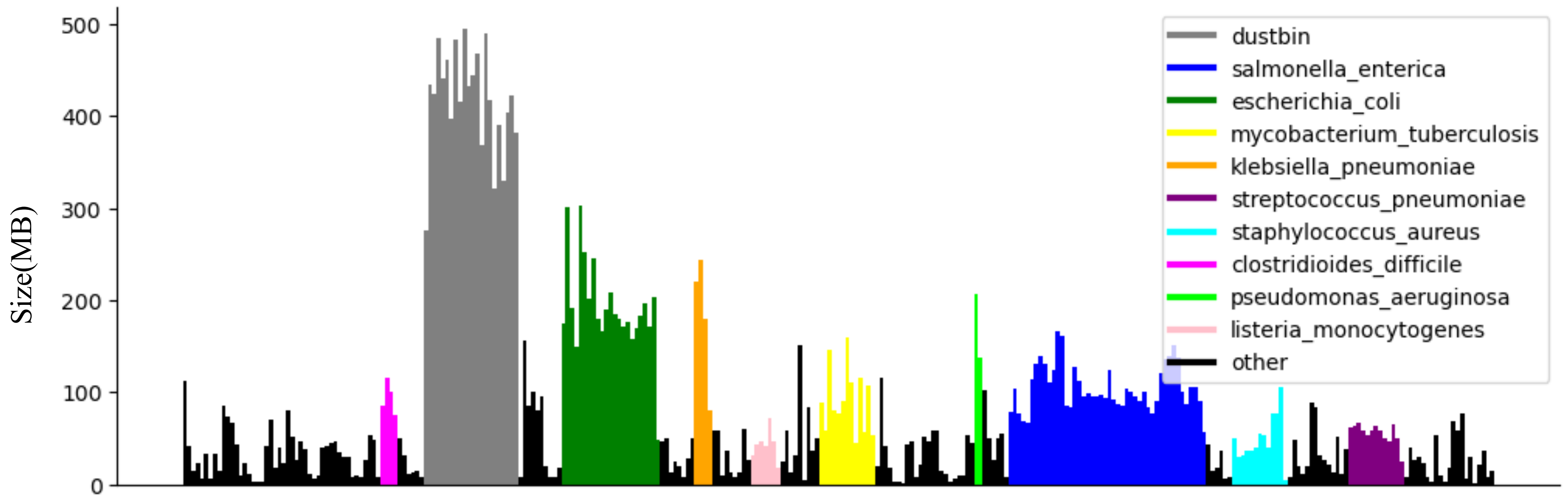

INPUT: Collection of genomes
Species clustering
Species 1
S2
S3
Batching
S1
S1
S1
S2
S2
S3
≈ 4000 genomes in each batches
1000 genomes in each dustbin
dustbin
dustbin

Split big clusters into smaller batches

Merge small clusters together into *dustbins*

Current Limitation: **Non-uniform** post-compression sizes

Batches are then reordered and compressed



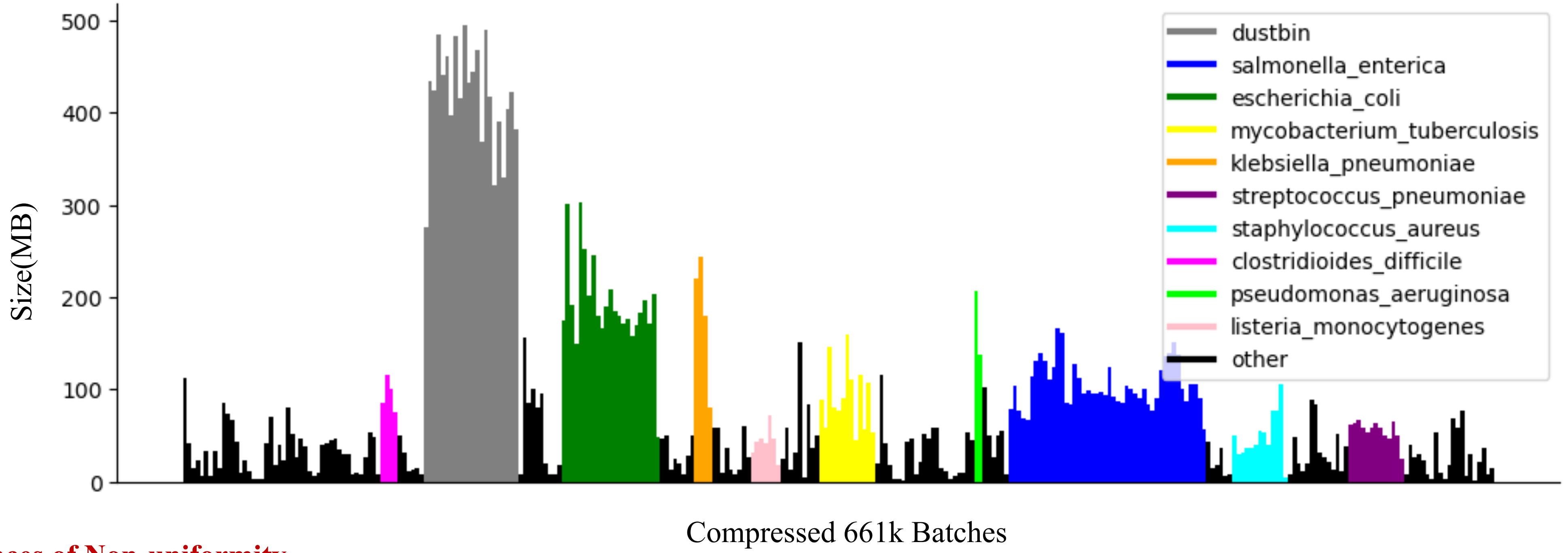
Compressed 661k Batches

Consequences of Non-uniformity

Unbalanced Workloads Inefficient Transmission Hinder Parallelization Memory Overuse Inconsistent Query Time

Current Limitation: **Non-uniform** post-compression sizes

We than proceed to reorder and compress all batches individually



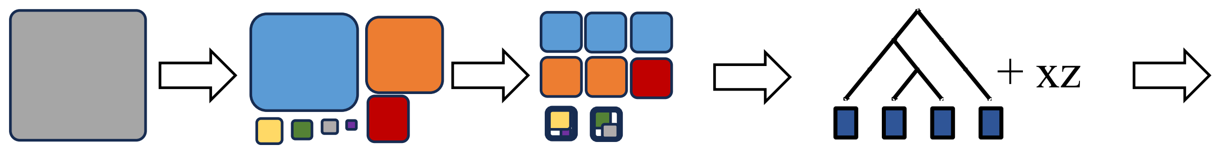
Consequences of Non-uniformity

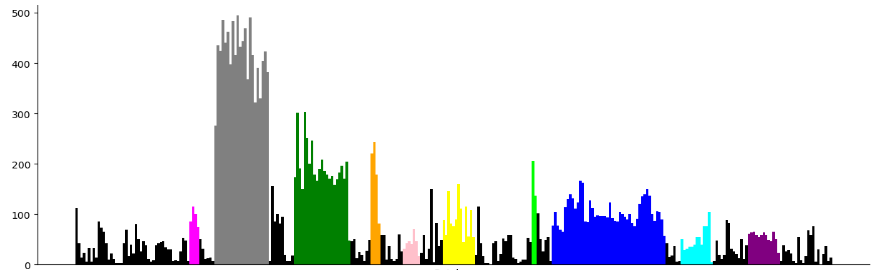
Unbalanced Workloads   Inefficient Transmission   Hinder Parallelization   Memory Overuse   Inconsistent Query Time

Application for balance batches: Portable Devives (remote setting, field work, rapid diagnostic), Parallel Platforms (GPU, Processing-in-Memory)

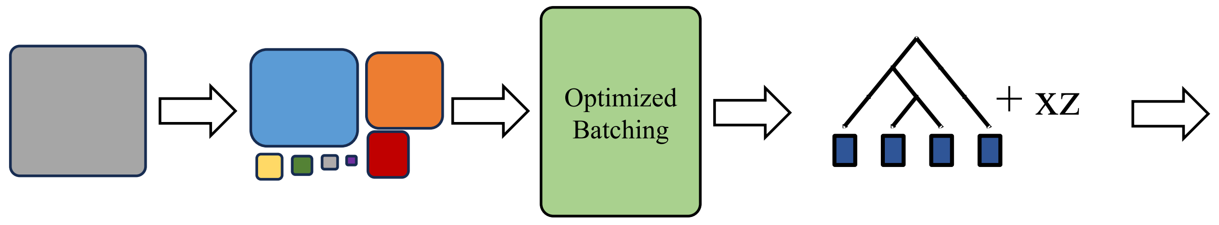
Our Goal: Design A Balancing Batching Strategy

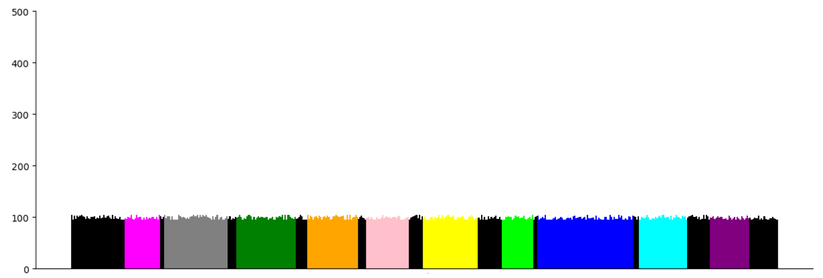
Current:





Objective:

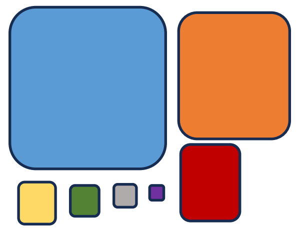




Toward The First Optimization Problem

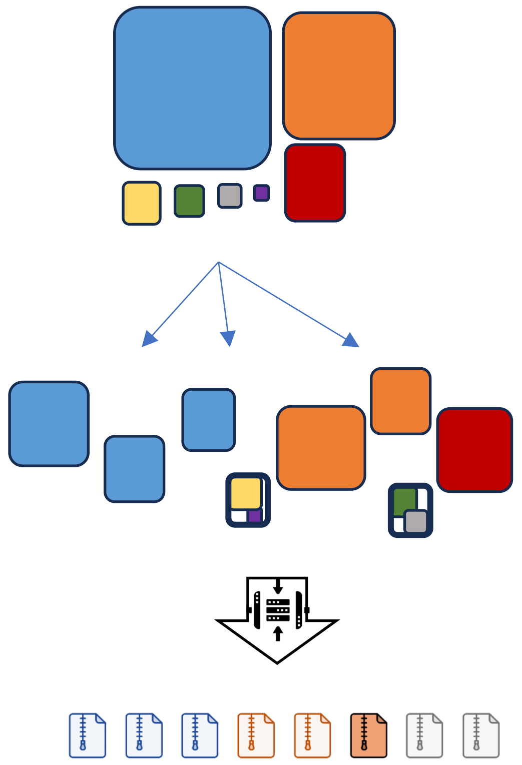
Quick Recap: What We Have So Far & Our Next Goal

- Clusters of genomes: genomes from the same species, uneven in size



Quick Recap: What We Have So Far & Our Next Goal

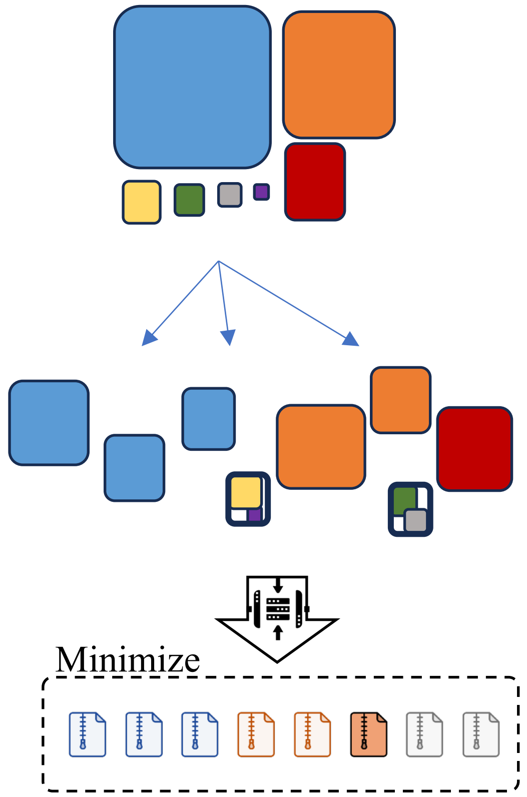
- Clusters of genomes: genomes from the same species, uneven in size
- We want to group them into batches while ensuring:
  - Requirements on batches: number of genomes, uncompressed size, …
  - Post-compression batch sizes must fit within a memory constraint (balance)



Quick Recap: What We Have So Far & Our Next Goal

- Clusters of genomes: genomes from the same species, uneven in size
- We want to group them into batches while ensuring:
  - Requirements on batches: number of genomes, uncompressed size, …
  - Post-compression batch sizes must fit within a memory constraint (balance)
- To ensure good compression ratio: maximize number of genomes per single batches.

  ➔ Corresponds to minimize the number of batches used.

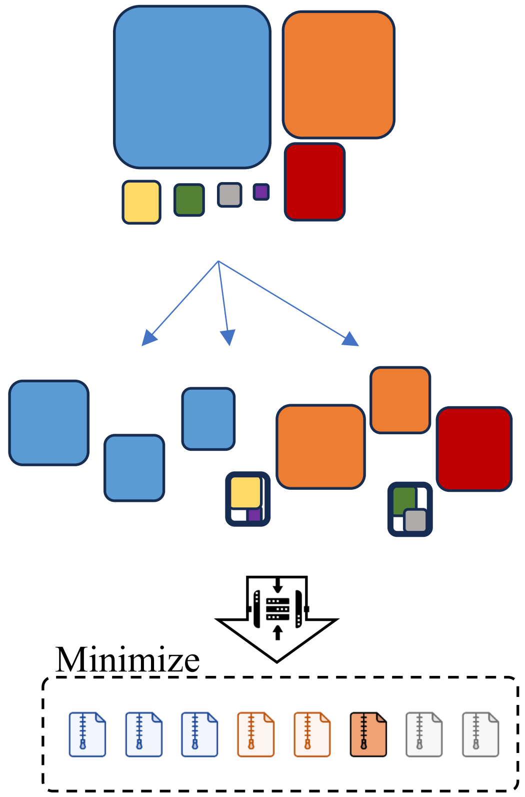


Quick Recap: What We Have So Far & Our Next Goal

- Clusters of genomes: genomes from the same species, uneven in size
- We want to group them into batches while ensuring:
  - Requirements on batches: number of genomes, uncompressed size, …
  - Post-compression batch sizes must fit within a memory constraint (balance)
- To ensure good compression ratio: maximize number of genomes per single batches.

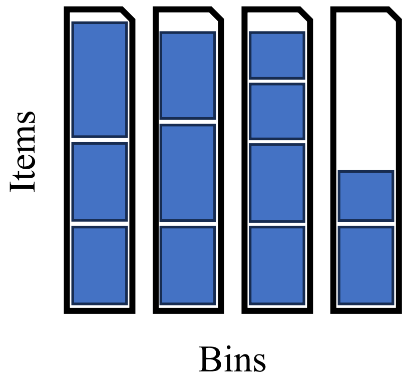
  ➔ Corresponds to minimize the number of batches used.

➔ **Optimization Problem: Bin Packing**



Bin Packing Problem: Definition

Bin Packing Problem:
Given a list of items i = 1, . . . , n, each having a size ci ∈ Z+, and an integer value CAPACITY. Find the minimum number of bin to pack all items in such a way that the sum of the item sizes in one bin is always smaller than CAPACITY.



A classic combinatorial optimization problem.

The problem is NP-complete

Classical heuristics are ordered-based algorithms.

Initially, an empty bin is created. At each step, the next item is selected and packed in a bin. A new bin may be created at each step.

- First-fit: choose the first possible bin
- Best-fit: choose largest remaining CAPACITY bin
- Worst-fit: choose smallest remaining CAPACITY bin

Still a trending research topics (presentation at ROADEF 2024)

Bin packing problems

François Clautiaux

The First Optimaztion Model For Balancing Batching

- Let $G = \{g_1, g_2, \dots, g_n\}$ be the set of genomes.
- $B = \{b_1, b_2, \dots, b_m\}$ be the set of batches, where $b_j \subseteq G$. All genomes need to be assigned, one genome in one batch.

The First Optimaztion Model For Balancing Batching

- Let $G = \{g_1, g_2, \dots, g_n\}$ be the set of genomes.
- $B = \{b_1, b_2, \dots, b_m\}$ be the set of batches, where $b_j \subseteq G$. All genomes need to be assigned, one genome in one batch.

$$b_j = \begin{cases} 1 & \text{if the } j \text{ batch is used} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if genome } i \text{ is assigned to batch } j \\ 0 & \text{otherwise} \end{cases}$$

The First Optimaztion Model For Balancing Batching

- Let $G = \{g_1, g_2, \ldots, g_n\}$ be the set of genomes.
- $B = \{b_1, b_2, \ldots, b_m\}$ be the set of batches, where $b_j \subseteq G$. All genomes need to be assigned, one genome in one batch.

$$b_j = \begin{cases} 1 & \text{if the } j \text{ batch is used} \\ 0 & \text{otherwise} \end{cases} \qquad x_{ij} = \begin{cases} 1 & \text{if genome } i \text{ is assigned to batch } j \\ 0 & \text{otherwise} \end{cases}$$

- The compression size of each batch must be less than or equal to A MB:
  $|post\_compression\_size(b_j)| \leq A, \forall j \in \{1, \ldots, m\}$

Objective function: $$\min \sum_{j=1}^{m} b_j$$

The First Optimaztion Model For Balancing Batching

- Let $G = \{g_1, g_2, \dots, g_n\}$ be the set of genomes.
- $B = \{b_1, b_2, \dots, b_m\}$ be the set of batches, where $b_j \subseteq G$. All genomes need to be assigned, one genome in one batch.

$$b_j = \begin{cases} 1 & \text{if the } j \text{ batch is used} \\ 0 & \text{otherwise} \end{cases} \qquad x_{ij} = \begin{cases} 1 & \text{if genome } i \text{ is assigned to batch } j \\ 0 & \text{otherwise} \end{cases}$$

- The compression size of each batch must be less than or equal to A MB:

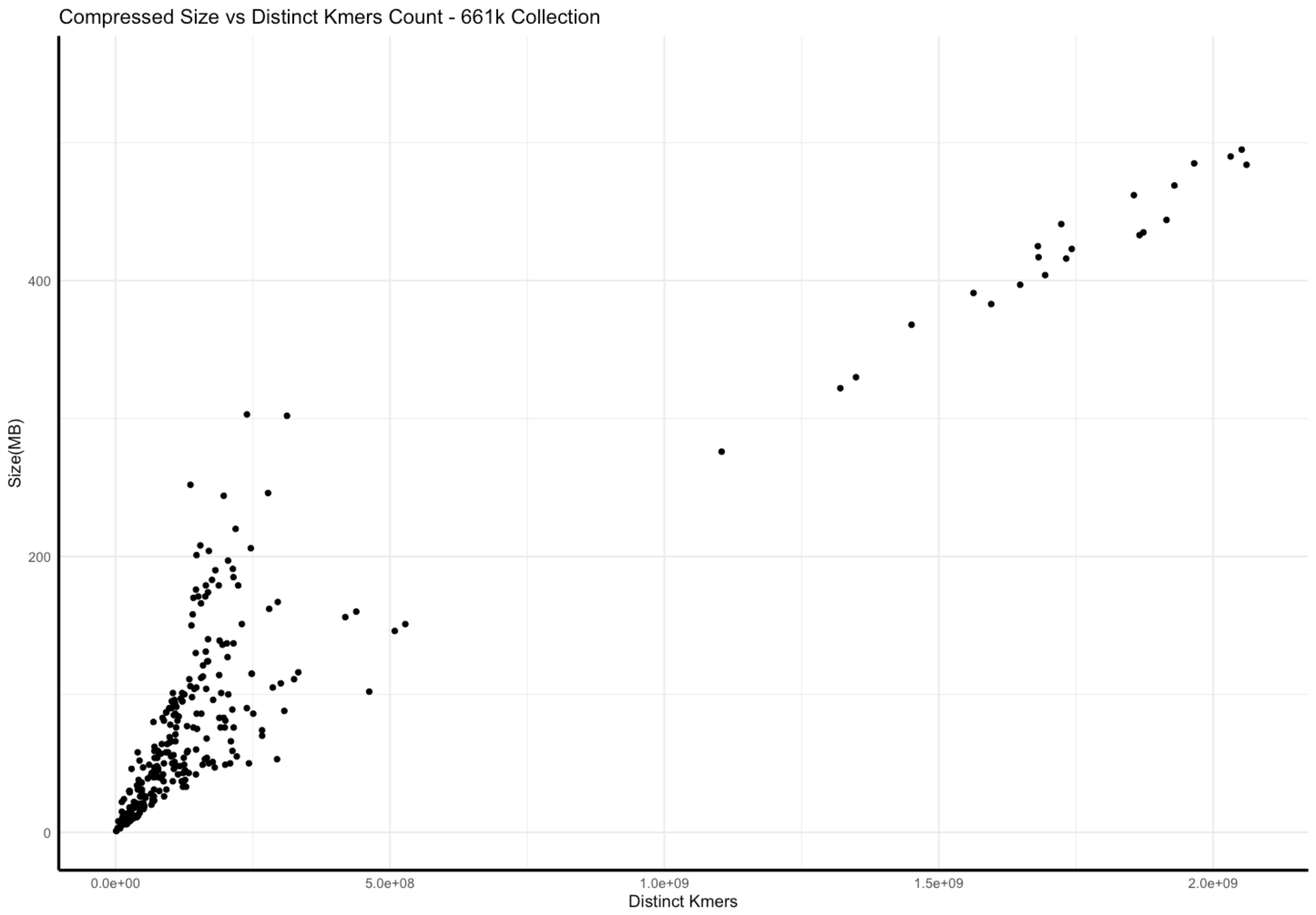
$$|post\_compression\_size(b_j)| \le A, \forall j \in \{1, \dots, m\}$$

Getting the compression size is non-trivial
➔ xz compression speed ≈ 1 genome per sec
➔ 1h20m for a batch with n = 5000

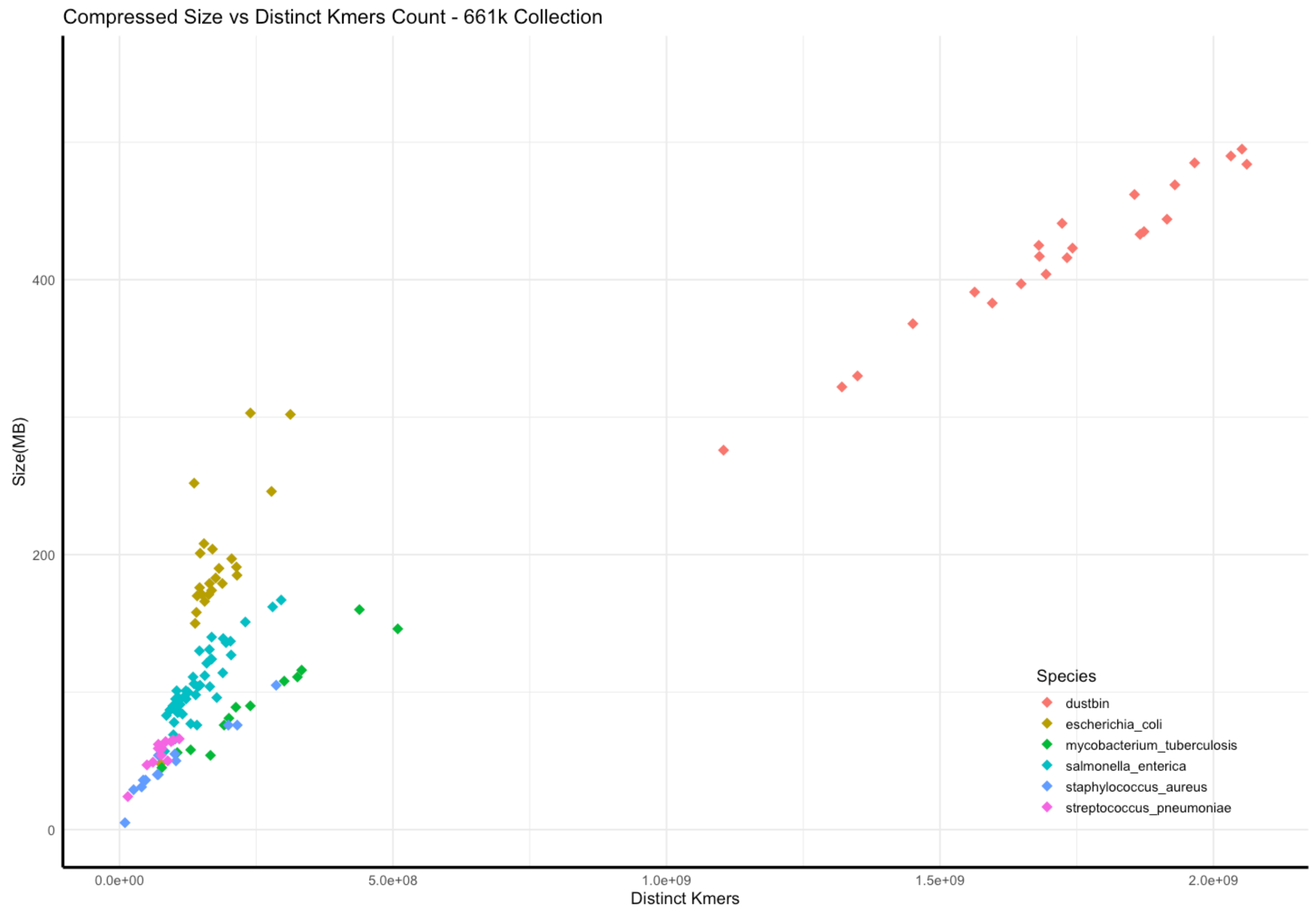
Objective function: $\min \sum_{j=1}^{m} b_j$

Implementation Of The Balancing Batching Model

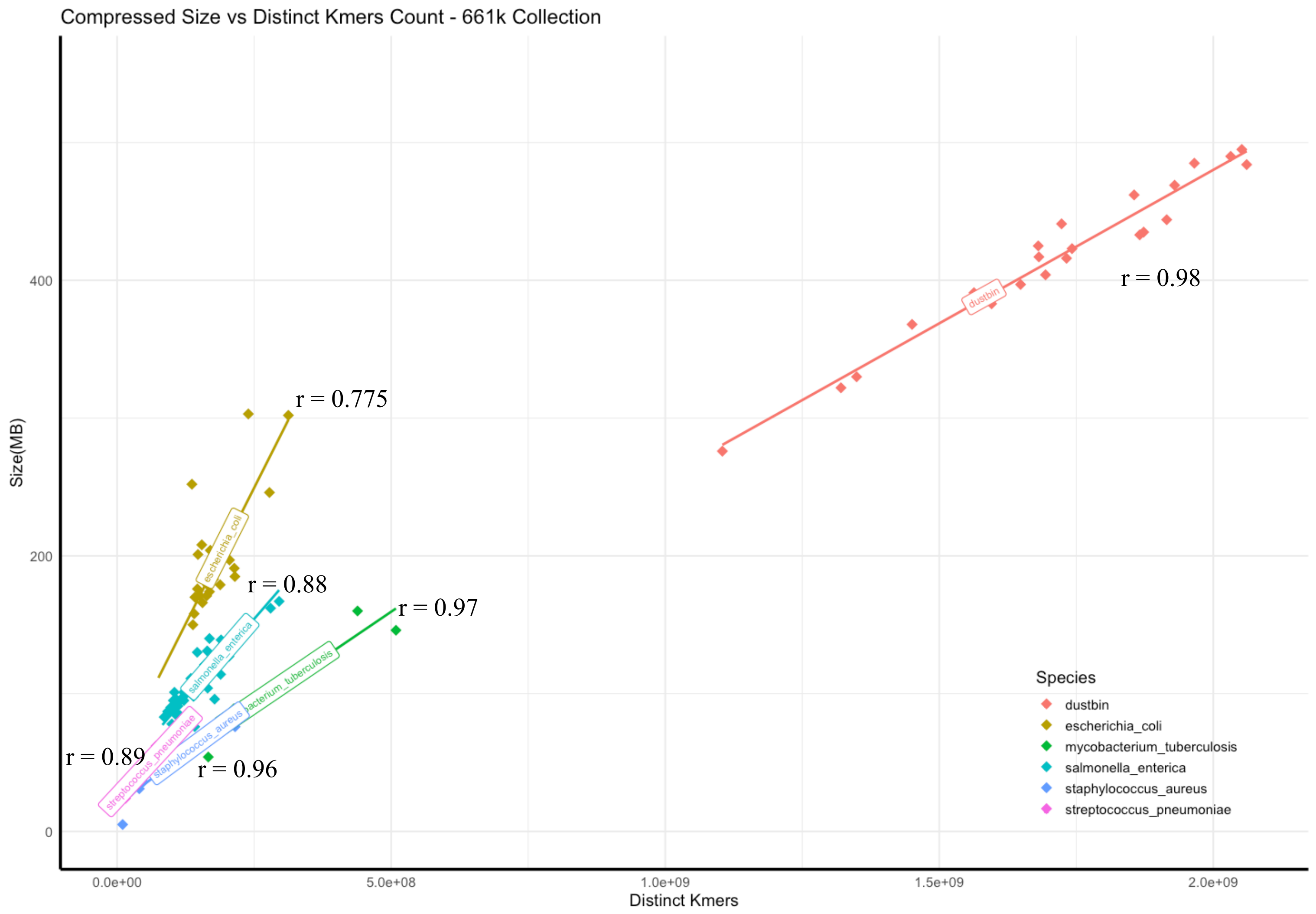
Ingredient 1: Post-compression Sizes Correlates With Distinct Kmers Count In Genomes Collections



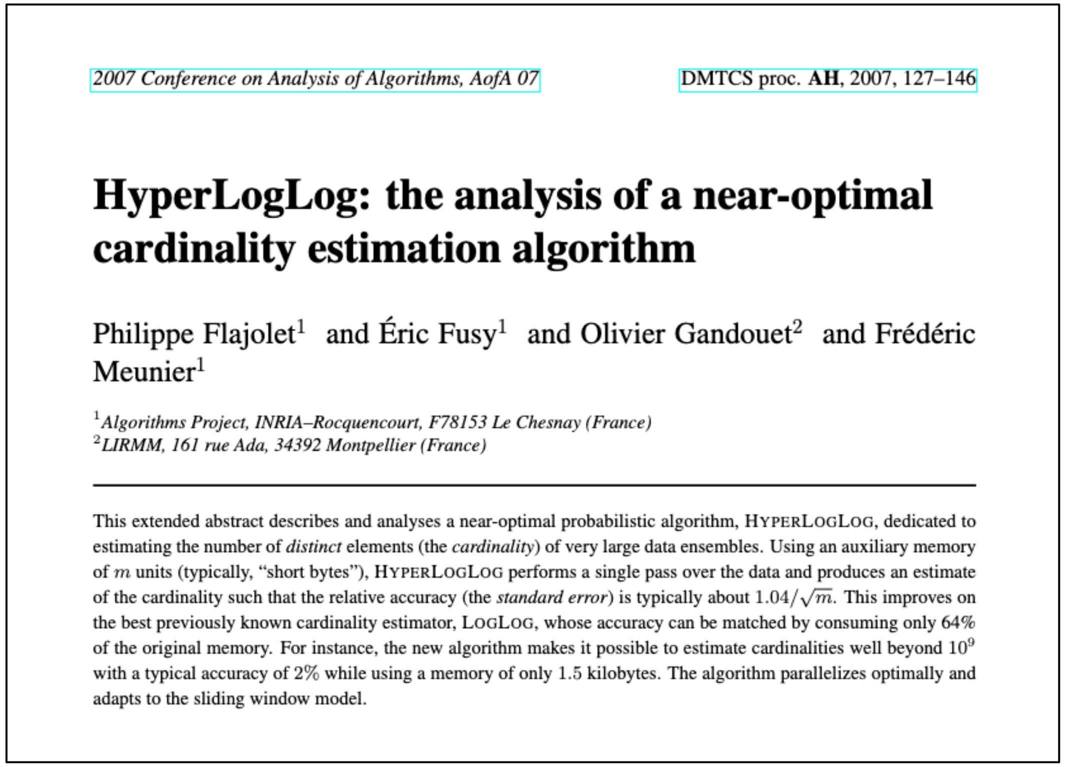
Ingredient 1: Post-compression Sizes Correlates With Distinct Kmers Count In Genomes Collections



Ingredient 1: Post-compression Sizes Correlates With Distinct Kmers Count In Genomes Collections



Ingredient 2: Fast Distinct Kmers Counting via HyperLogLog



2007 Conference on Analysis of Algorithms, AofA 07 DMTCS proc. **AH**, 2007, 127–146

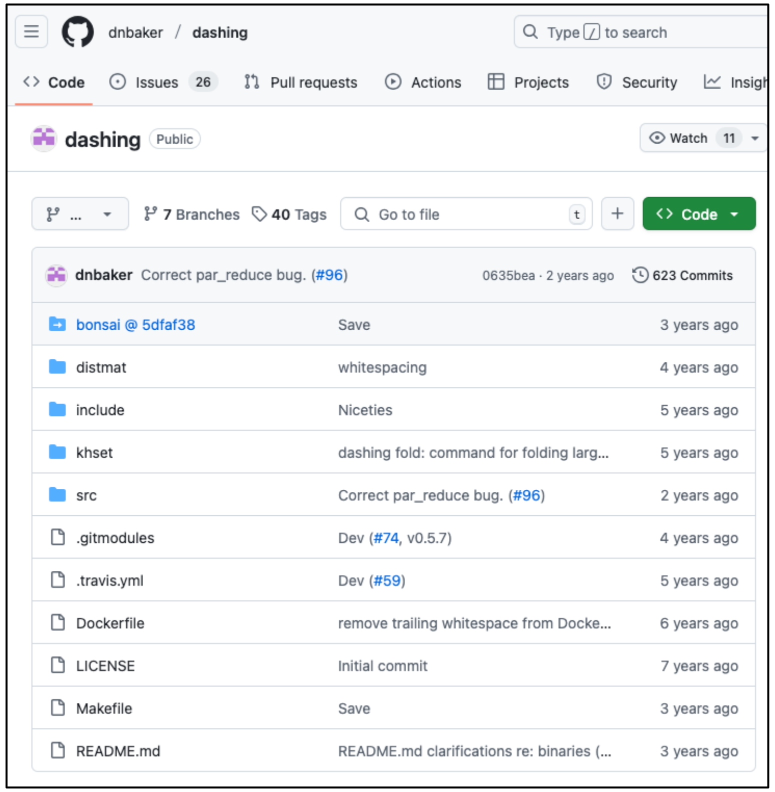
HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm

Philippe Flajolet[1] and Éric Fusy[1] and Olivier Gandouet[2] and Frédéric Meunier[1]

[1] *Algorithms Project, INRIA–Rocquencourt, F78153 Le Chesnay (France)*
[2] *LIRMM, 161 rue Ada, 34392 Montpellier (France)*

This extended abstract describes and analyses a near-optimal probabilistic algorithm, HYPERLOGLOG, dedicated to estimating the number of *distinct* elements (the *cardinality*) of very large data ensembles. Using an auxiliary memory of m units (typically, "short bytes"), HYPERLOGLOG performs a single pass over the data and produces an estimate of the cardinality such that the relative accuracy (the *standard error*) is typically about $1.04/\sqrt{m}$. This improves on the best previously known cardinality estimator, LOGLOG, whose accuracy can be matched by consuming only 64% of the original memory. For instance, the new algorithm makes it possible to estimate cardinalities well beyond 10^9 with a typical accuracy of 2% while using a memory of only 1.5 kilobytes. The algorithm parallelizes optimally and adapts to the sliding window model.

HyperLogLog Cardinality Estimation[1]



dnbaker / dashing

dashing Public

7 Branches 40 Tags

dnbaker Correct par_reduce bug. (#96) 0635bea · 2 years ago 623 Commits

bonsai @ 5dfaf38	Save	3 years ago
distmat	whitespacing	4 years ago
include	Niceties	5 years ago
khset	dashing fold: command for folding larg...	5 years ago
src	Correct par_reduce bug. (#96)	2 years ago
.gitmodules	Dev (#74, v0.5.7)	4 years ago
.travis.yml	Dev (#59)	5 years ago
Dockerfile	remove trailing whitespace from Docke...	6 years ago
LICENSE	Initial commit	7 years ago
Makefile	Save	3 years ago
README.md	README.md clarifications re: binaries (...	3 years ago

Implemented in Dashing[2,3]

[1] Philippe Flajolet, Éric Fusy, Olivier Gandouet, Frédéric Meunier. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. *AofA: Analysis of Algorithms*, Jun 2007
[2] Baker, D.N., Langmead, B. Dashing: fast and accurate genomic distances with HyperLogLog. *Genome Biol* 20, 265. 2019
[3] https://github.com/dnbaker/dashing

Ingredient 3: Bin Packing Greedy Algorithms – 2 Variations

STRATEGY 1 : given unlimited batches with capacity C

$$\boldsymbol{Minimize\ nb\ of\ batch\ B}$$

s.t.

$distinct\_kmers(bj) < C,$ for (j = 1, … , $\boldsymbol{m}$)

Sort the genomes by acession number.

Initially, an empty bin is created.

At each step, the next genomes is selected and packed in the first available bin.

Distinct kmers count in each step is calculated using Dashing

Create new bin as needed.

STRATEGY 2 : given a fixed number of batch n

$$T \geq distinct\_kmers(b_j), for\ j = 1, \dots, m$$

$$\boldsymbol{Minimize\ T}$$

Sort the genomes by acession number.

Create n bins.

At each step, the next genomes is selected and packed in the bin with the smallest CAPACITY.

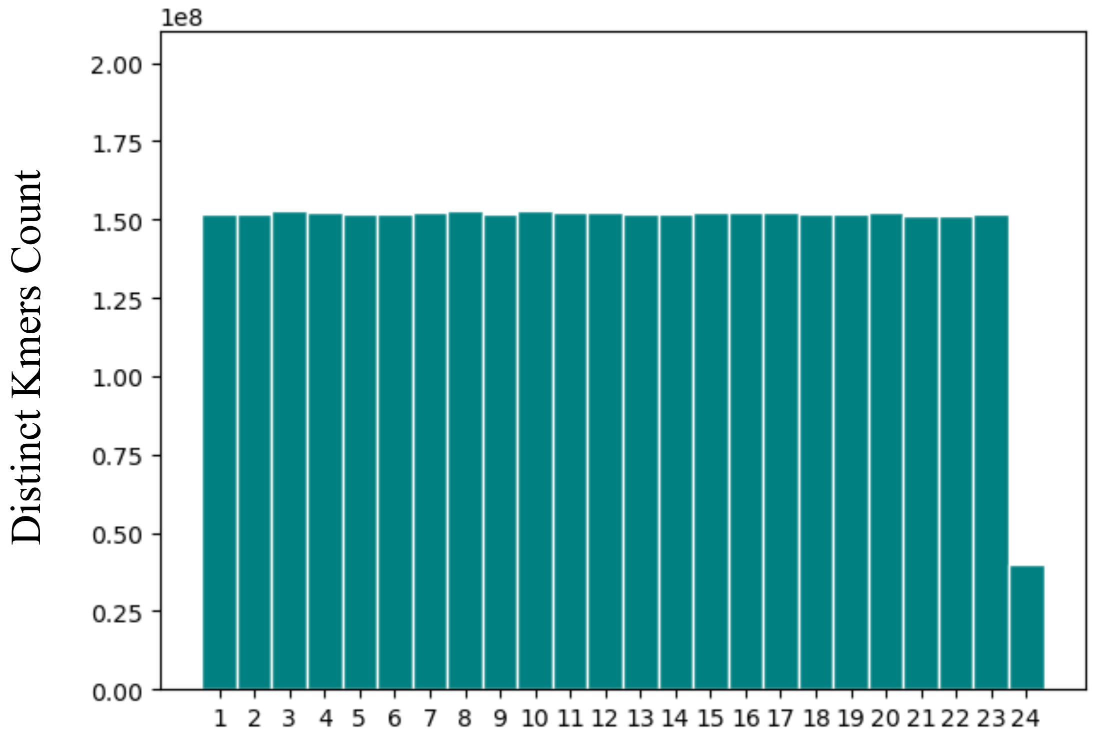
Distinct kmers count in each step is calculated using Dashing

First Results

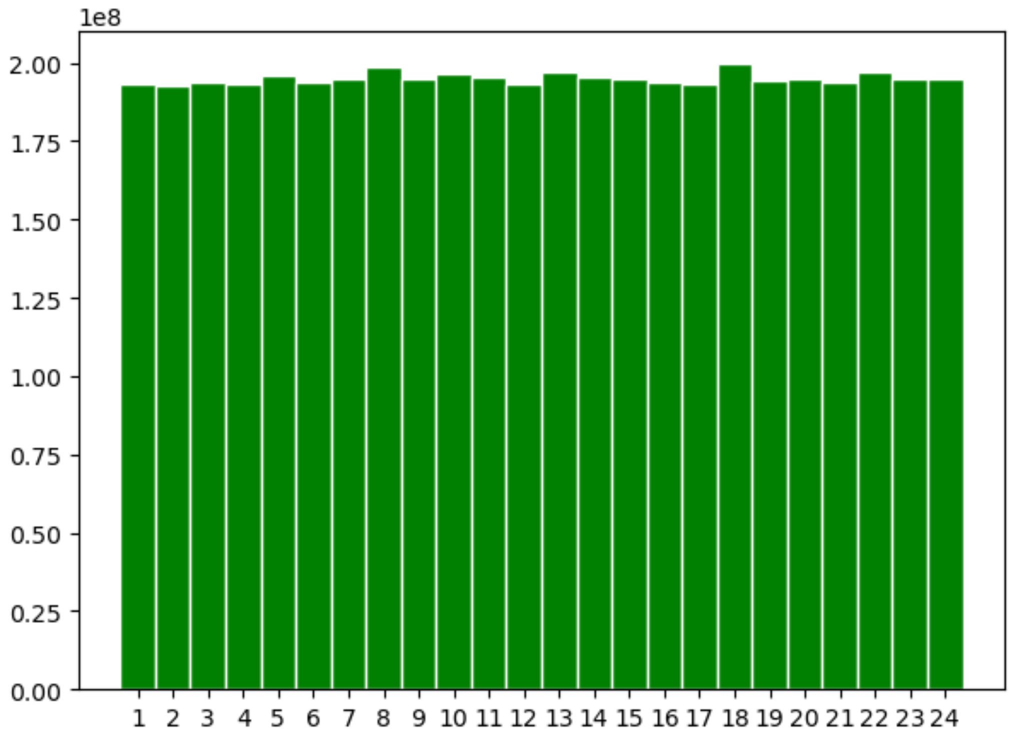
Dataset:

- Assemblies of *Mycobacterium tuberculosis* from 661k collection
- Number of Genomes: around 49,000
- Uncompressed Size: 218 GB
- CAPACITY of batches: 152,000,000

Batching Results Comparison: Distinct Kmers Count

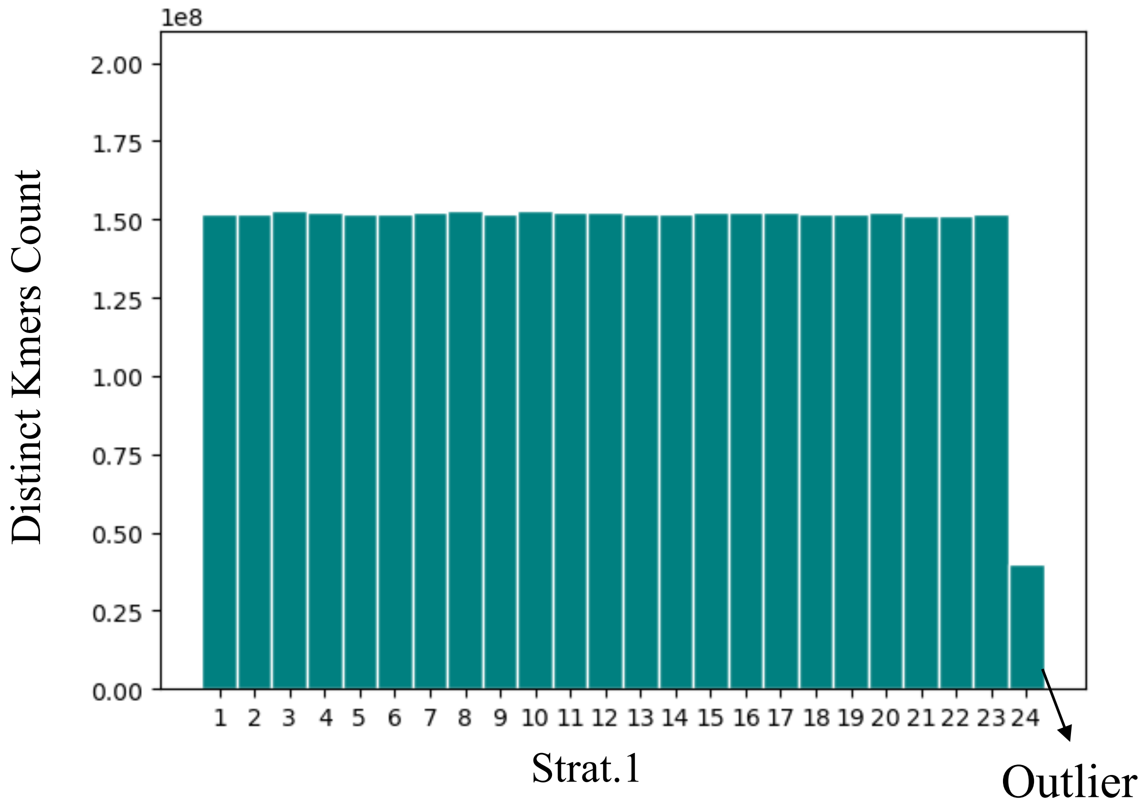


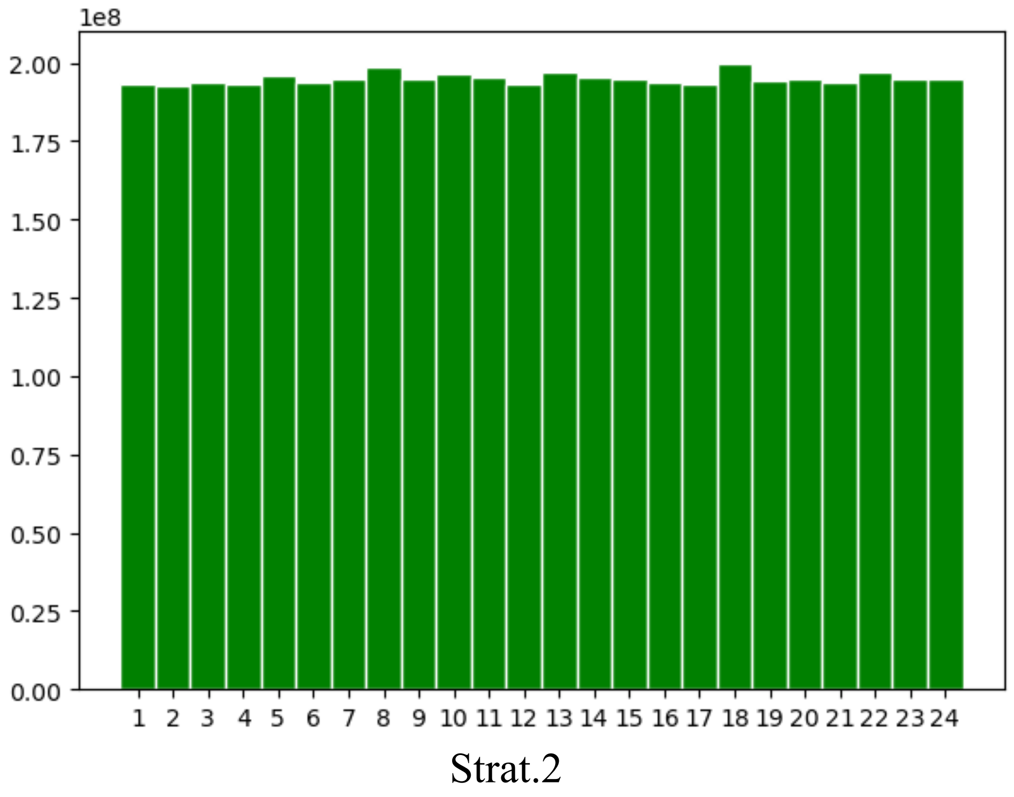
Strat.1



Strat.2

Batching Results Comparison: Distinct Kmers Count



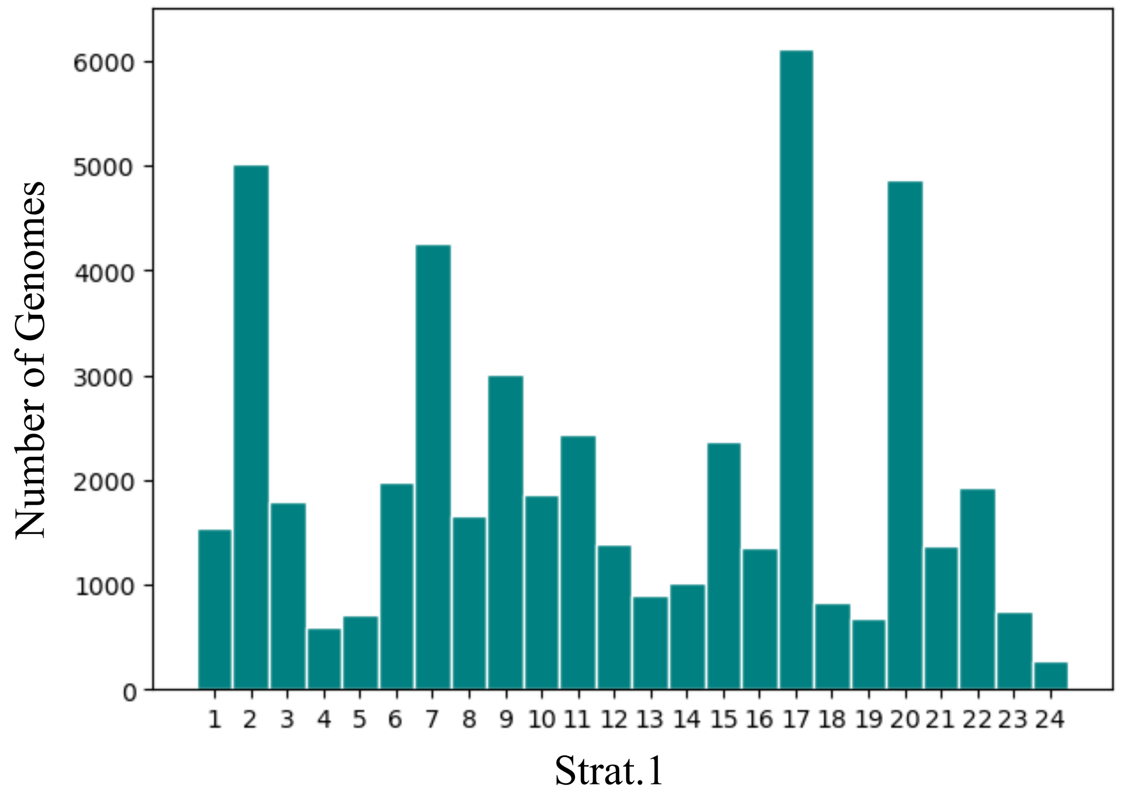


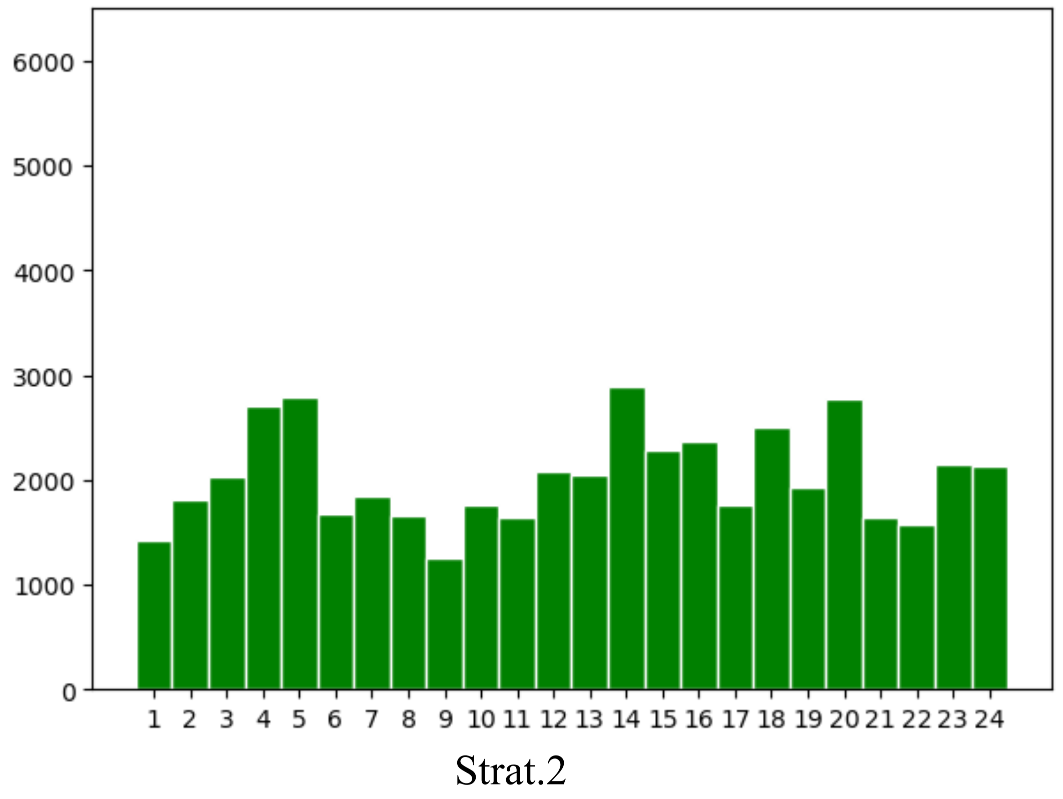
Strat.1 keep the order of accession number

➔ Lower number of distinct kmers count per batch

➔ Distinct Kmers Counts is balanced

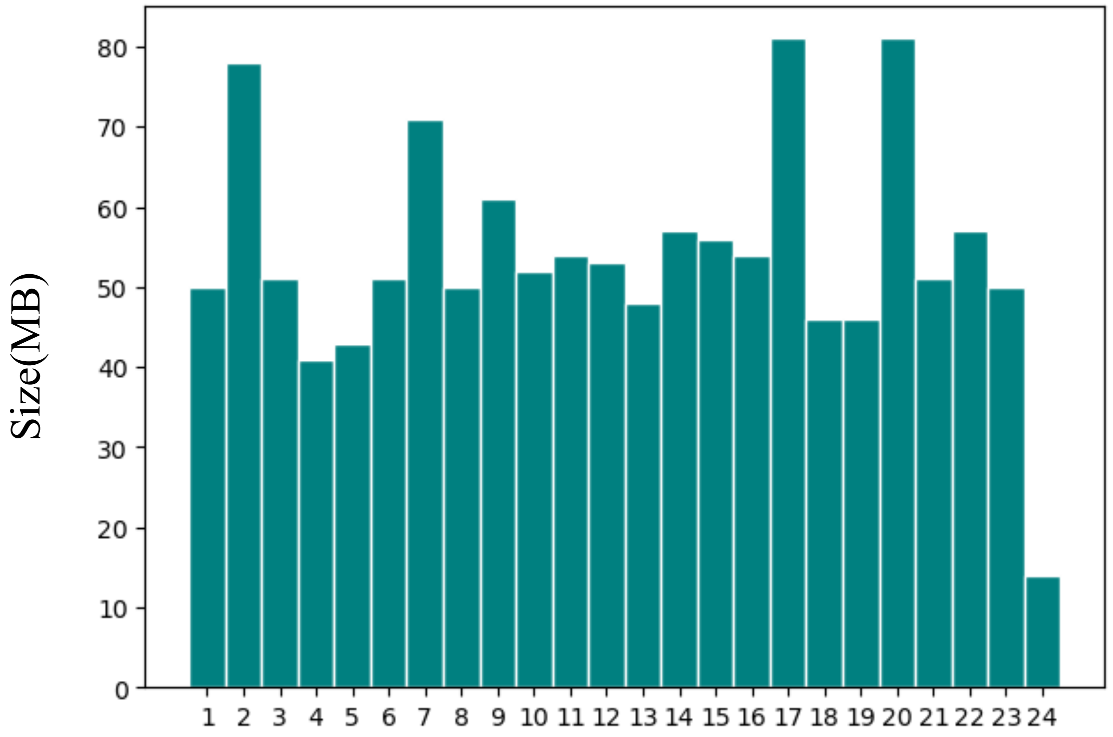
Batching Results Comparison: Number Of Genomes



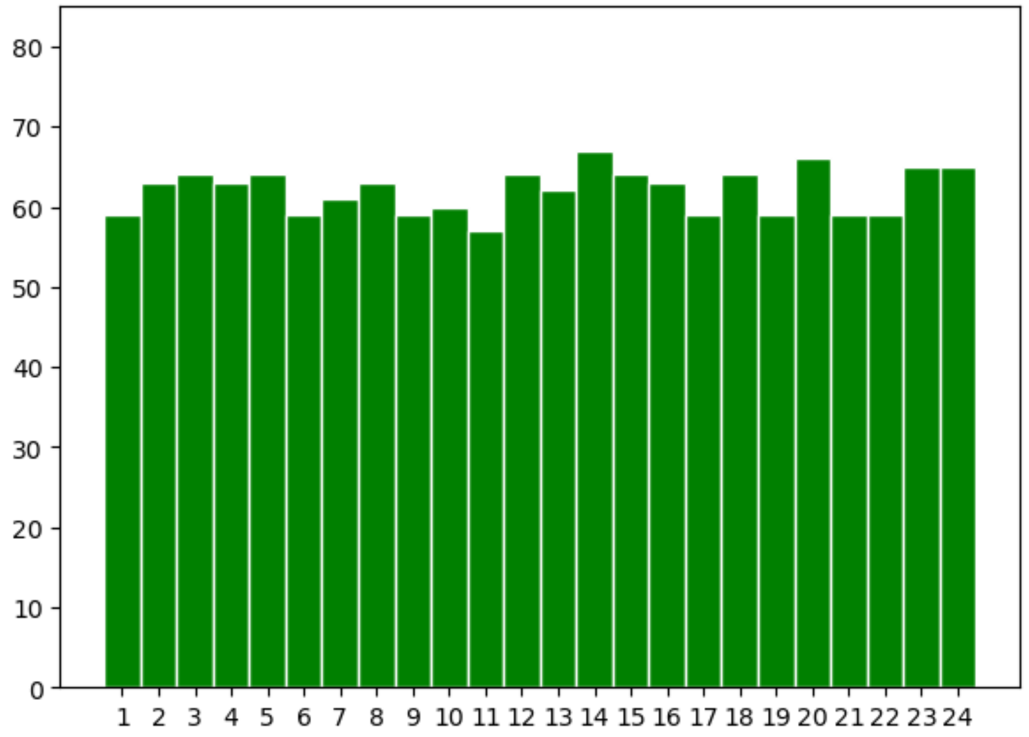


Number of genomes per batch varies, strat.2 to a lesser extent compared to strat.1

Batching Results Comparison: After Compression

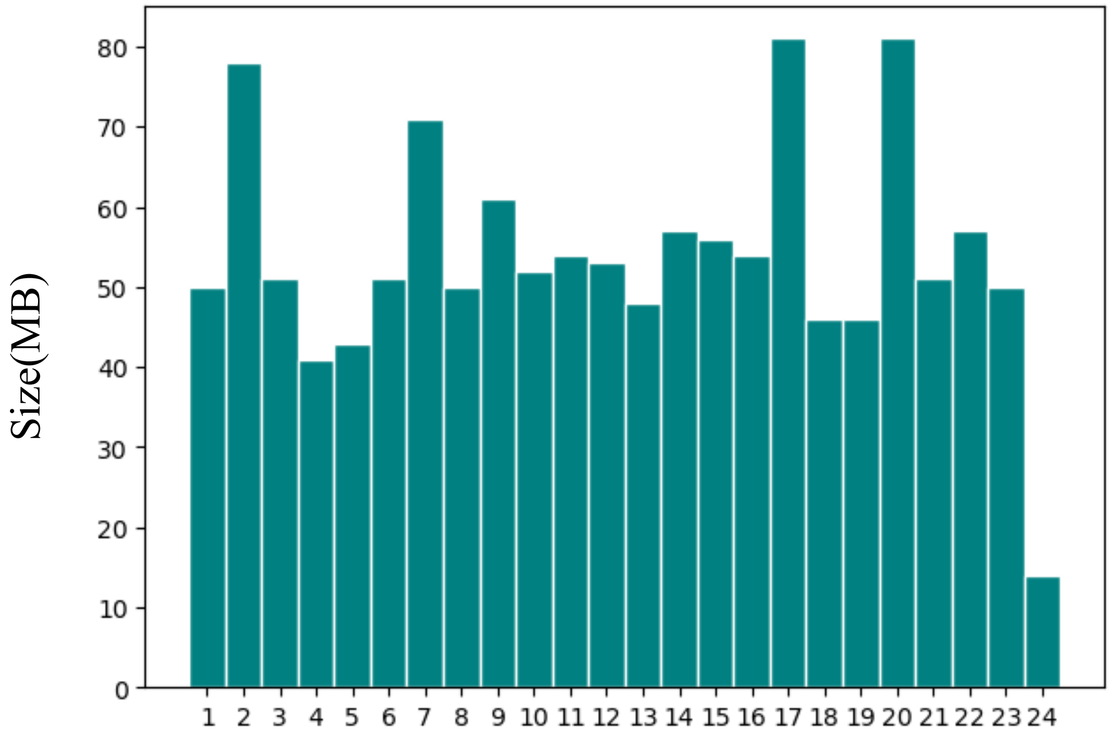


Strat.1



Strat.2

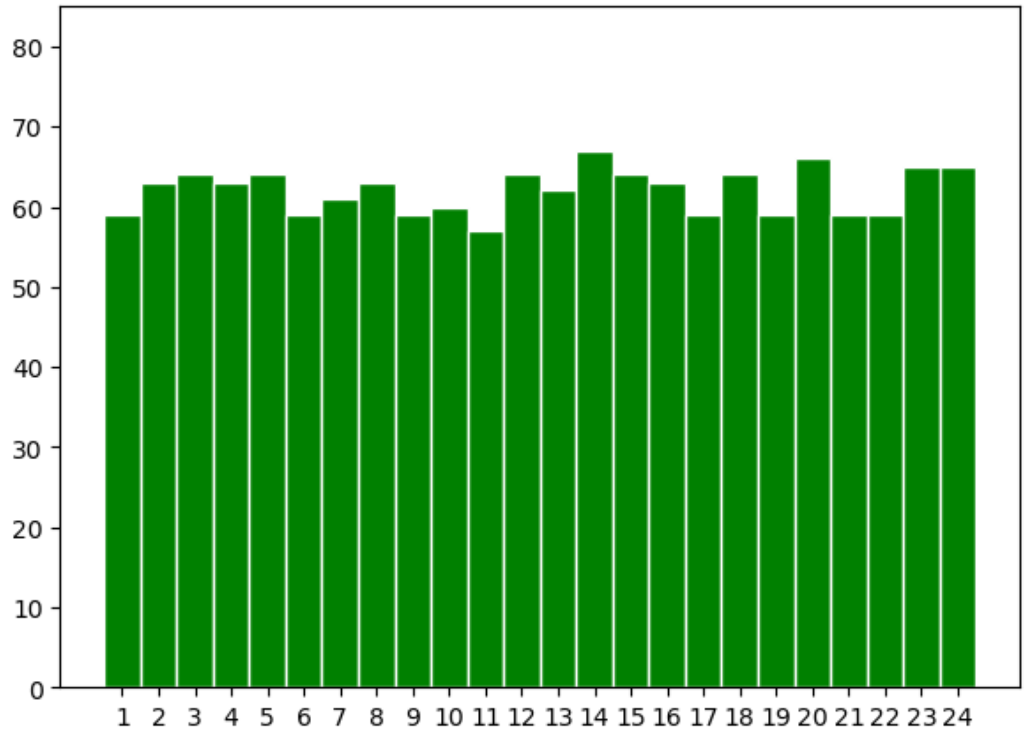
Batching Results Comparison: After Compression



Strat.1

Post-compression size: 1,3G

Most of the batches are balanced (between 40-50MB, max size 81MB)

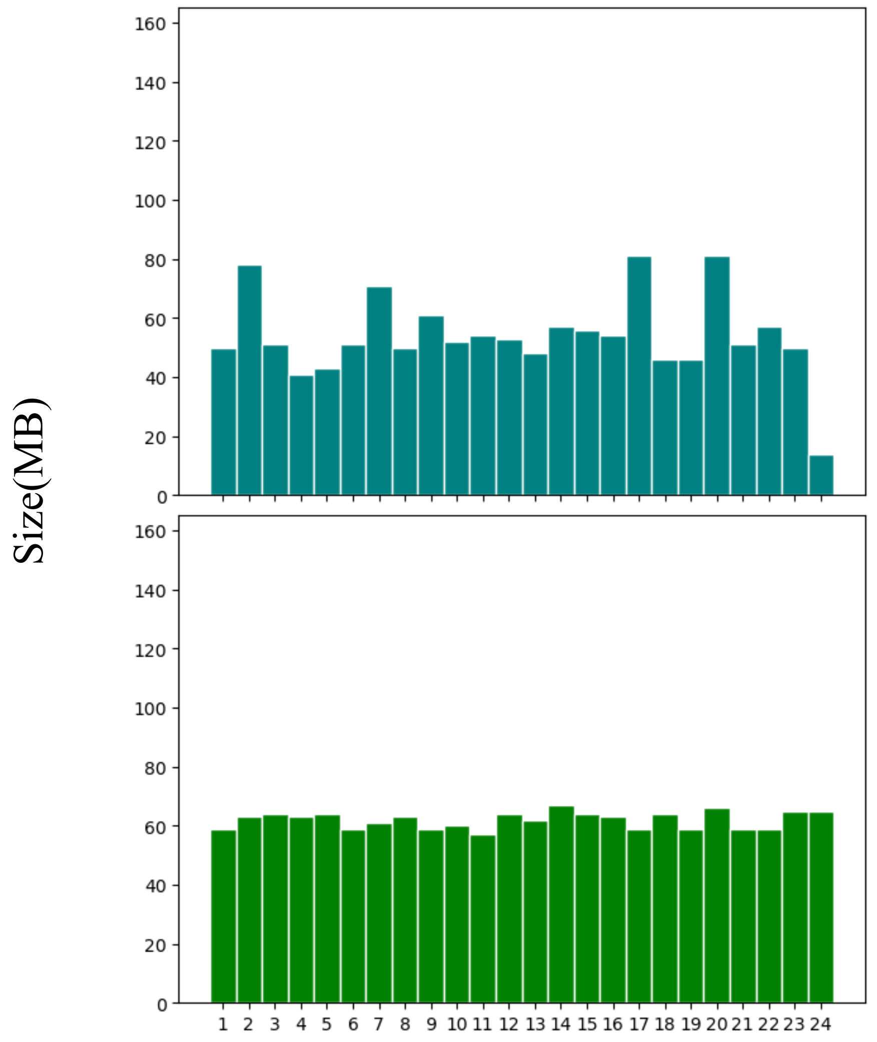


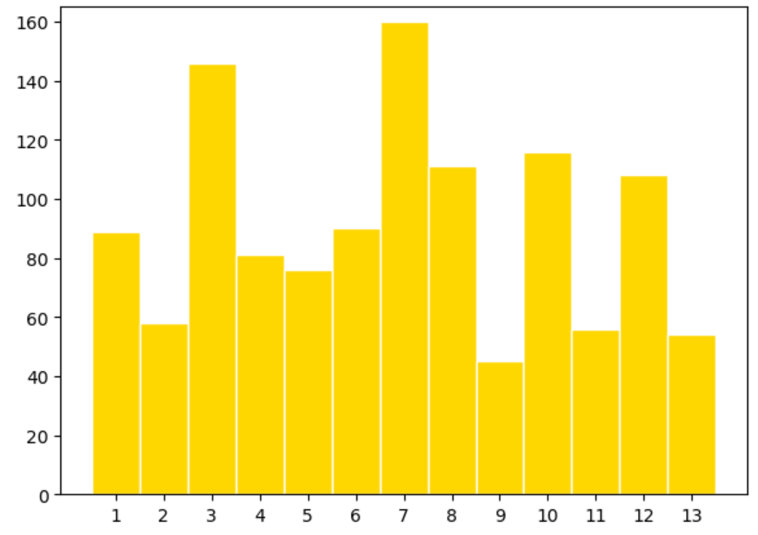
Strat.2

Post-compression size: 1,6G

All Batches are well balanced (between 59-67MB, max size 67MB)

Batching Results Comparison: Compare With the Original Batching





Post-compression size: 1,2G

Highly imbalance

Conclusion & Perspectives

Batching is a crucial step in Phylogenetic Compression.

Batching by estimating compression size via HyperLogLog and Distinct K-mer counts improves balancing of the final compressed sizes *Mycobacterium tuberculosis*.

First results:

- First model for Optimization Batching as a Bin Packing Problem.
- Workflows of the bin packing batching strategies:
  - *https://github.com/tam-km-truong/HLL-Binning*
  - *https://github.com/tam-km-truong/HLL-Balancing*

Perspectives:

- Currently scaling up the results and methods to the 661k and the AllTheBacteria Collections
- Introducing new constraints such as Max Number of genomes per batch
- Application in other data structure such as Bloom filter, on PIM and GPU

Thank You