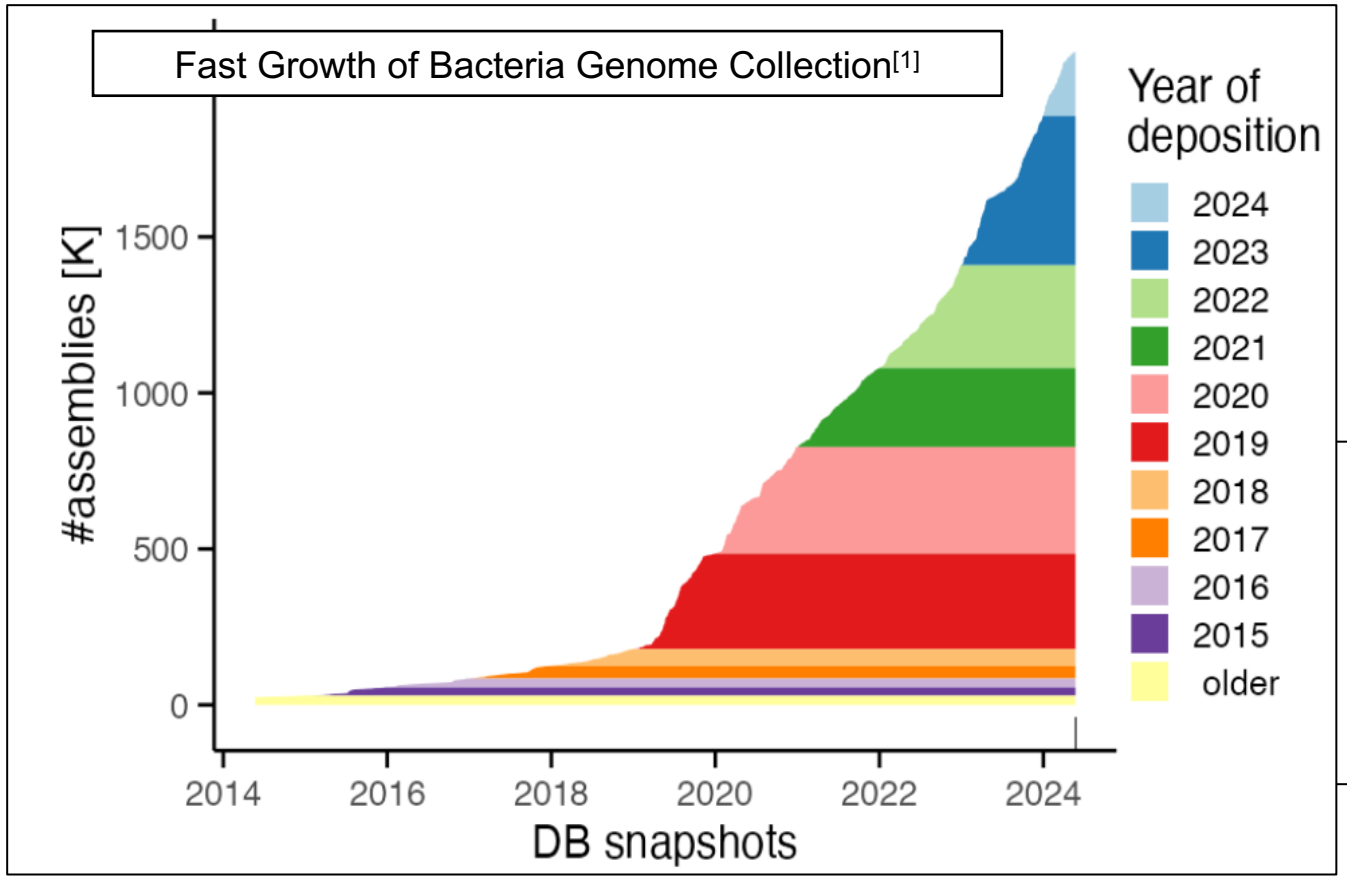
Optimizations For Efficient Compression Of Large Bacterial Genome Collections

MOTIVATION: Larger And Higher Diversity Genome Collections Are Growing Rapidly



Large Bacterial Genome Collections:

661kcollection[2](2021) nb_of_genomes = 661,405

AllTheBacteria[3] (2024) n = 2,440,377

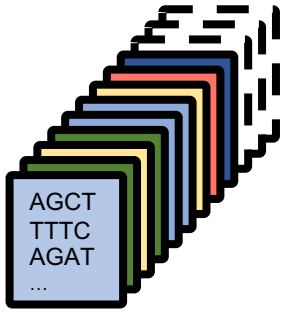
Future $n > 10^7$

Make big line here about bacterial genomes collection, then make the graph bigger

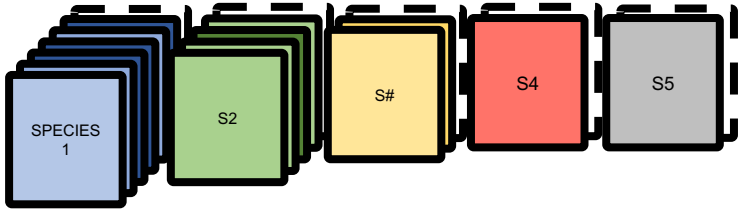
RECENT INNOVATION: Phylogenetic Compression[1] Improves Compressibility Via Reordering According To The Evolutionary History (MiniPhy protocol)

Step 1: Phylogenetic clustering & batching

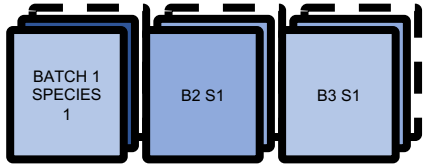
INPUT:
Collection of genomes
Size: 100GB

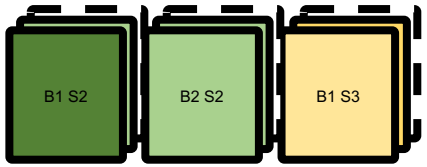


Species clustering

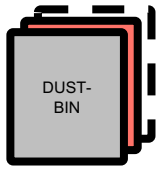


Split highly sampled clusters

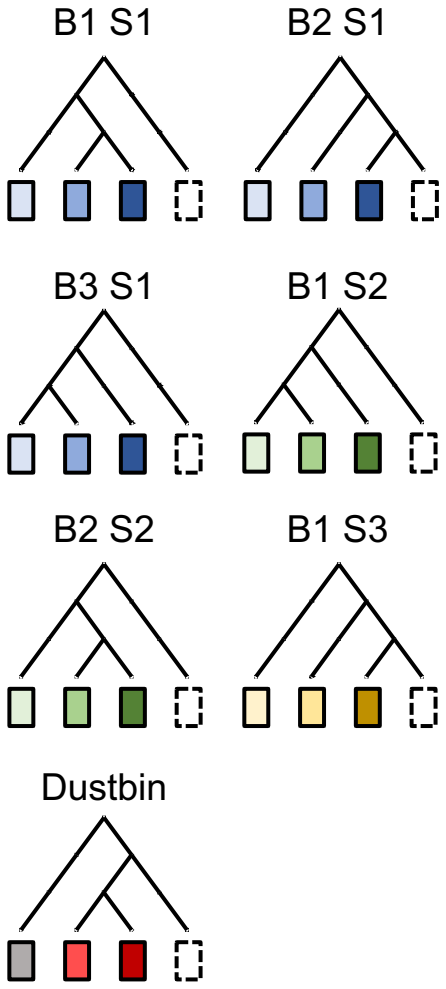




Merge sparsly sampled clusters

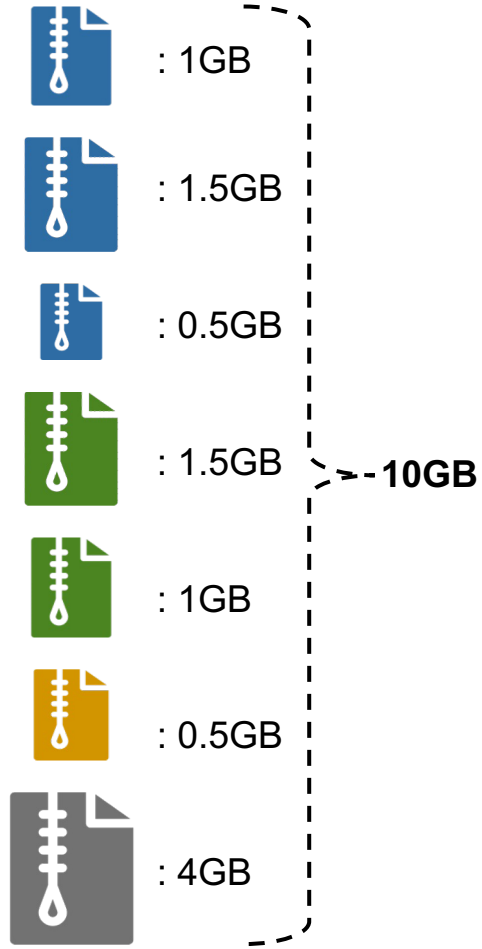


Step 2: Phylogenetic reordering

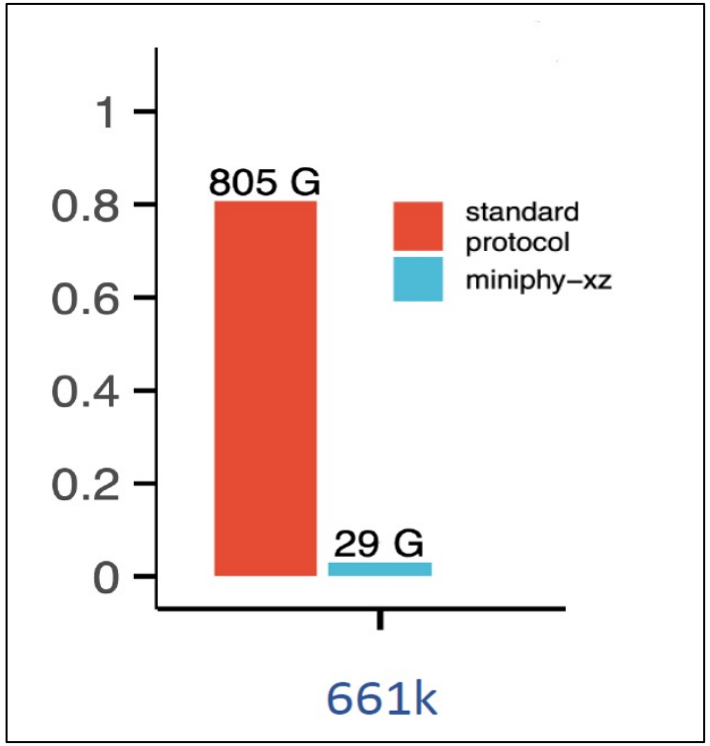


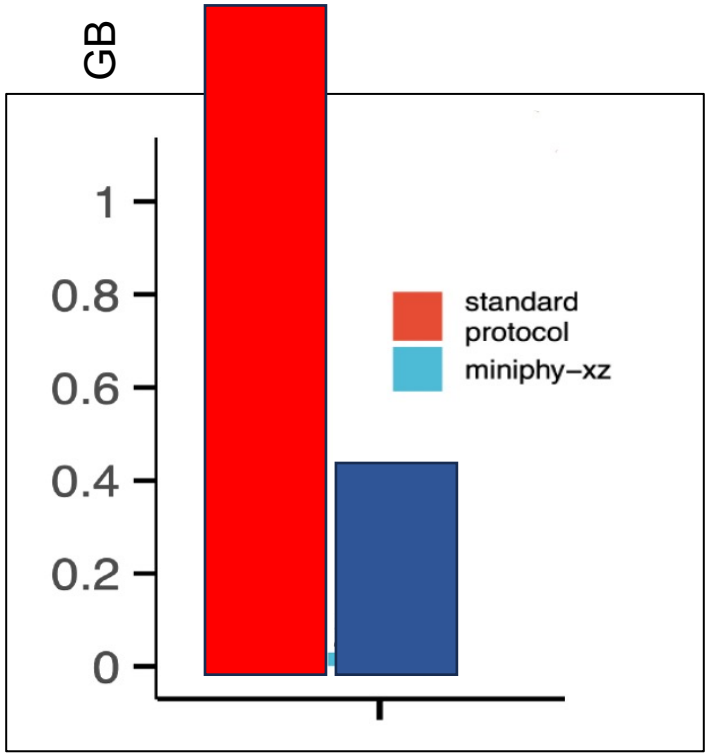
Infer evolutionary trees

Resulting compression



This Strategy Allows Lossless Compression Of 1-3 Orders Of Magnitude Across Different Genome Collections

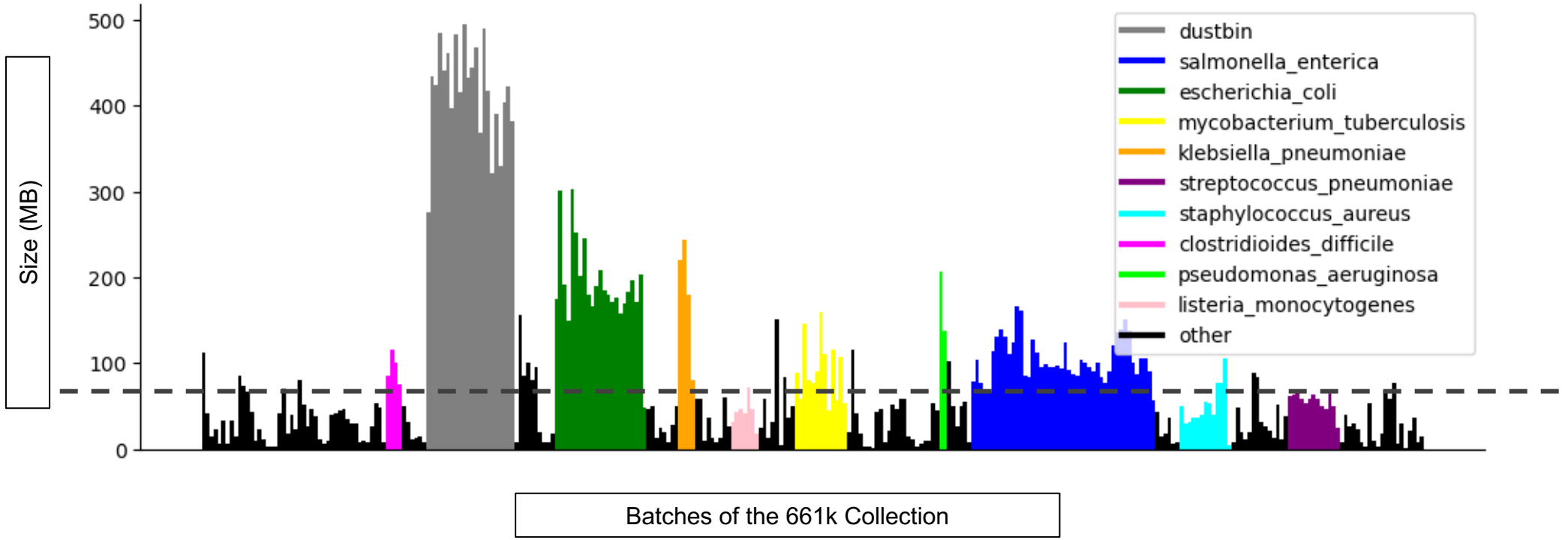




AllTheBacteria - TBA

RESULTING COMPRESSION

Current Limitation: Batching Results In Non-uniform Post-compression Sizes

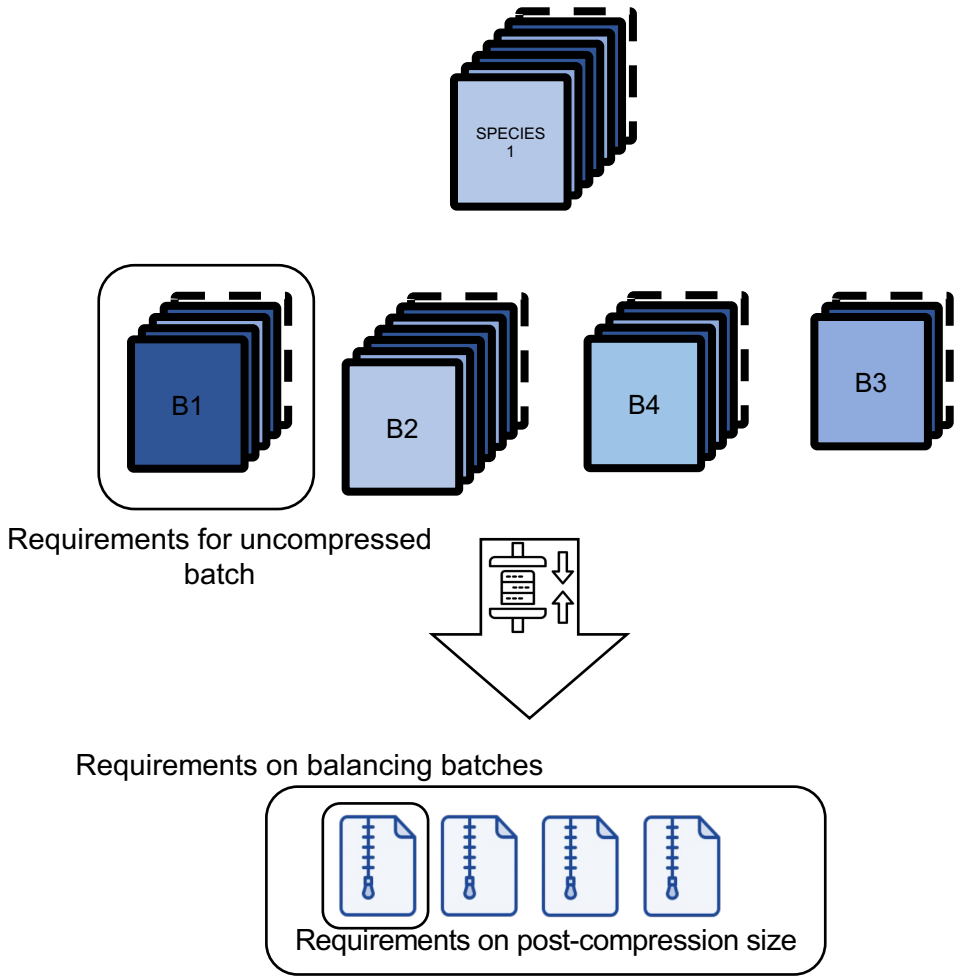


Consequences: Negative Impact on Downstream Analysis

- Unbalanced Workloads
- Hinder Parallelization
- Inconsistent Query Times
- Memory Overuse
- Inefficient Transmission

Applications: Negative Impact on Downstream Analysis

Ultimate Objective: Design An Optimized Batching Strategy For Multiple Applications



Different Requirements

Batching Problem:
Given a set of genomes.
Partition it into a set of batches.
User-input parameters for each batch i.e. nb_genomes N, uncompressed size U, post-compression size C.
Minimize the total compression size of the batches and in such a way that the constraints are satisfied.

OBJECTIVE:

$$min \sum_{i}^{Batches} \text{PostCompressionSize}(b_i)$$

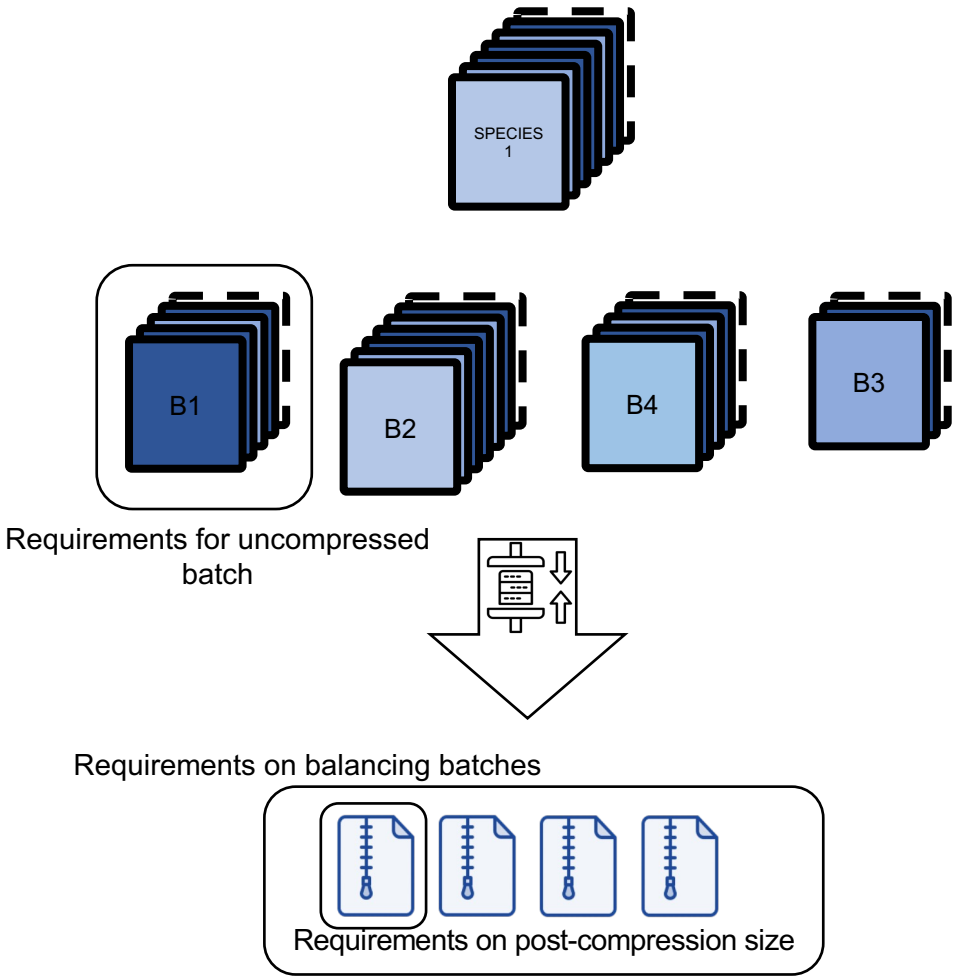
Subjects to: (requirements)
For each batch b_i:

$$\text{Cardinality}(b_i) \leq N$$
$$\text{UncompressedSize}(b_i) \leq U$$
$$\text{PostCompressionSize}(b_i) \leq C$$

Balancing requirement of all couple i and j:

$$\text{PostCompressionSize}(b_i) \approx \text{PostCompressionSize}(b_j)$$

Ultimate Objective: Design An Optimized Batching Strategy For Multiple Applications



Different Requirements

Batching Problem:
Given a set of genomes.
Partition it into a set of batches.
User-input parameters for each batch i.e. nb_genomes N, uncompressed size U, post-compression size C.
Minimize the total compression size of the batches and in such a way that the constraints are satisfied.

OBJECTIVE:

$$min \sum_{i}^{Batches} \text{PostCompressionSize}(b_i)$$

Subjects to: (requirements)
For each batch b_i:

$$\text{Cardinality}(b_i) \leq N$$
$$\text{UncompressedSize}(b_i) \leq U$$
$$\text{PostCompressionSize}(b_i) \leq C$$

Balancing requirement of all couple i and j:

$$\text{PostCompressionSize}(b_i) \approx \text{PostCompressionSize}(b_j)$$

Predicting Post-compression Size Is Non-trivial

Naive Approach: Without Considering Compression

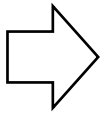
Assumption:
For simplicity, we stop considering genome compression

OBJECTIVE:

$$min \sum_{i}^{Batches} \text{PostCompressionSize}(b_i)$$

Subjects to:
For for all batches:

$$\text{Cardinality}(b_l) \leq N$$
$$\text{UncompressedSize}(b_l) \leq U$$
$$\text{PostCompressionSize}(b_l) \leq C$$
$$\text{PostCompressionSize}(b_l) - \text{PostCompressionSize}(b_j) \leq \varepsilon$$



OBJECTIVE:

$$min \sum_{i}^{Batches} b_i$$

Subjects to:
For for all batches:

$$\text{Cardinality}(b_l) \leq N$$
$$\text{UncompressedSize}(b_l) \leq U$$

Naive Approach: Without Considering Compression

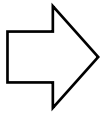
Assumption:
For simplicity, we stop considering genome compression

OBJECTIVE:

$$min \sum_{i}^{Batches} \text{PostCompressionSize}(b_i)$$

Subjects to:
For for all batches:

$$\text{Cardinality}(b_l) \leq N$$
$$\text{UncompressedSize}(b_l) \leq U$$
$$\text{PostCompressionSize}(b_l) \leq C$$
$$\text{PostCompressionSize}(b_l) - \text{PostCompressionSize}(b_j) \leq \varepsilon$$



OBJECTIVE:

$$min \sum_{i}^{Batches} b_i$$

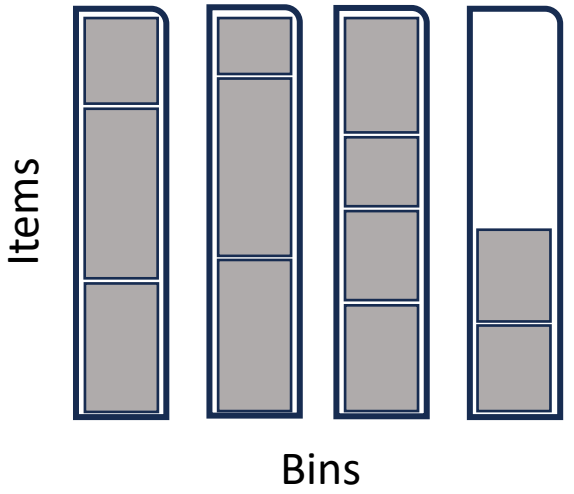
Subjects to:
For for all batches:

$$\text{Cardinality}(b_l) \leq N$$
$$\text{UncompressedSize}(b_l) \leq U$$

This becomes an instance of the classical optimization problem: **Bin Packing**

Bin Packing Problem Is One Of The First Studied Combinatorial Optimization Problem

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.



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

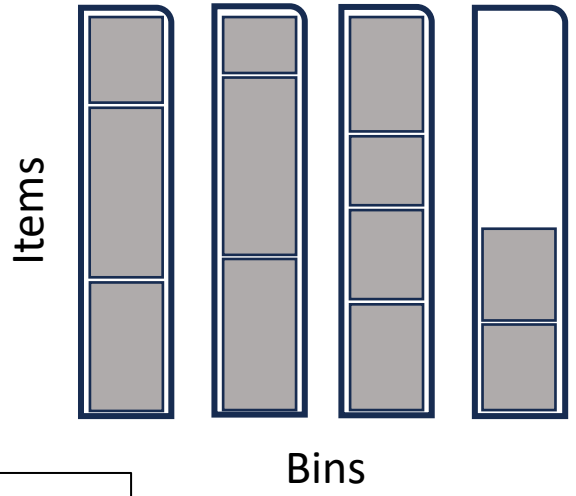
Studied in Master 2 at ISTIC

Continue to be a trending research topics (presented at ROADEF 2024)

Bin Packing Problem Is One Of The First Studied Combinatorial Optimization Problem

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.



Is it possible to approximate the post-compression size of a batch without doing the compression?

The problem is NP-complete

Classical heuristics are ordered-based algorithms.

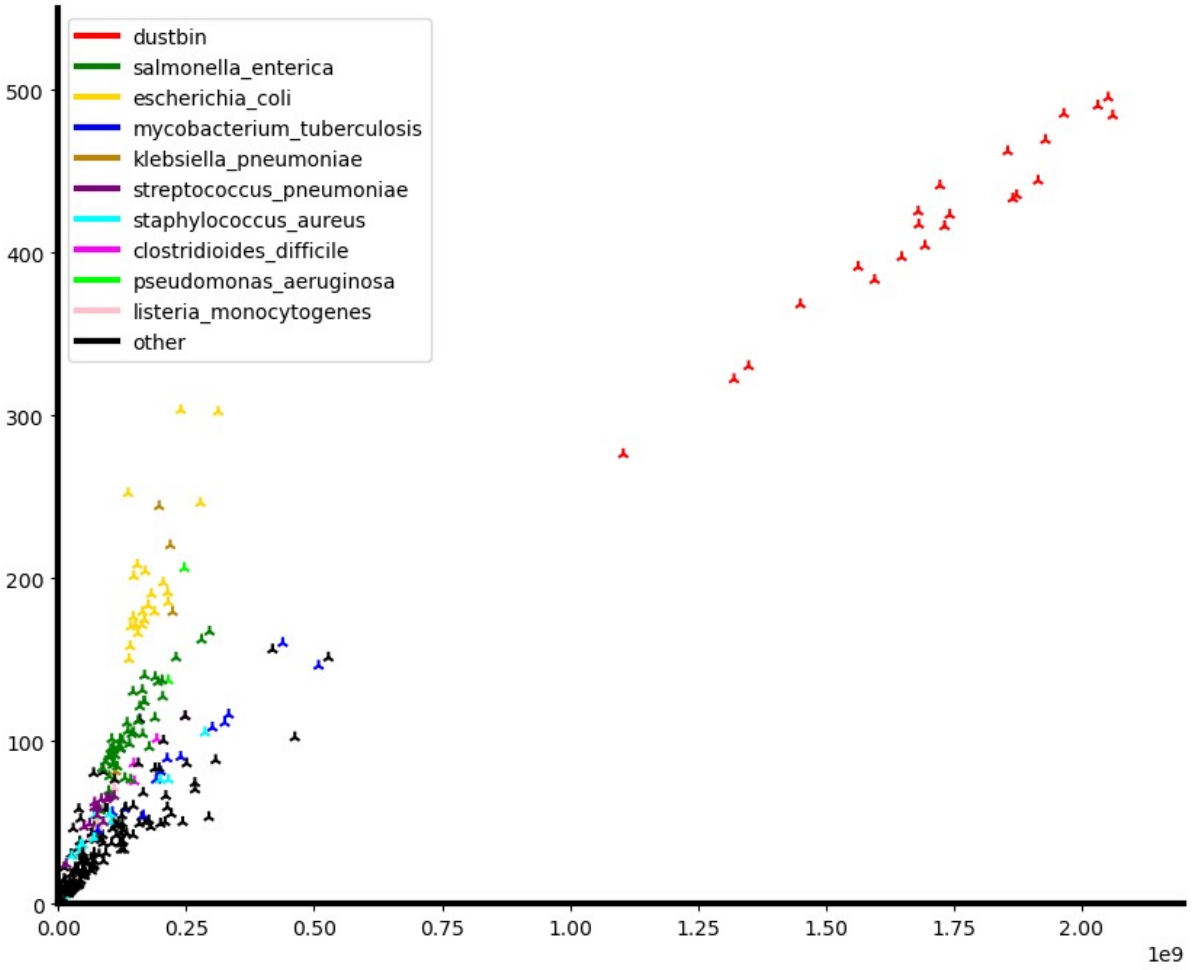
Initially, an empty bin is created. At each step, the next item is selected and packed in a bin. A new bin may be created at each step.

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

Continue to be a trending research topics (presented at ROADEF 2024)

Studied in Master 2 at ISTIC

Approximation Of Post Compression Size Via Distinct Kmers Count



From without compression to compression

Ingredient 2: Cardinality estimation using HyperLogLog sketching

Sketches : approximate data structures.

HyperLogLog sketches for cardinality est.: bit patterns,

i.e. *hash(ATGCG)* ➔ 00010100, *hash(CGTAC)* ➔ 00000010.

Fast and efficient UNION operation for sketches.

Bin Packing with distinct kmers Strategy For Genomes Batching

Pseudocode of Strategy 1

Another approach Load Balancing with distinct kmers Strategy For Genomes Batching

We have a fixed number of bin B
T >= #_unique_kmers(bi) for each bi in B
Objective function:
Min T

Recap of the Three Batching Strategies

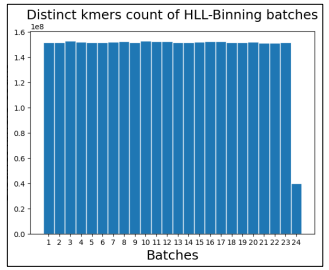
Comparisons of the batching strategíe

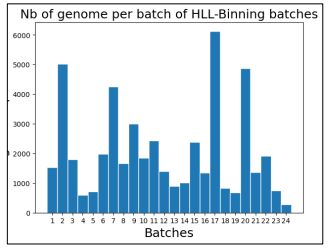
STRATEGY 1: HLL-Binning

DATA : Genomes of *Mycobacterium tuberculosis* from the 661k Collection[2], B = 24

STRATEGY 2: HLL-Balancing

Batches Obtained From Strat. 1

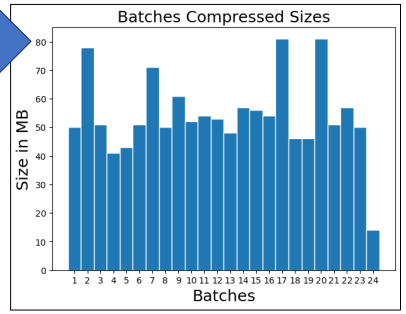




PHYLOGENETIC COMPRESSION

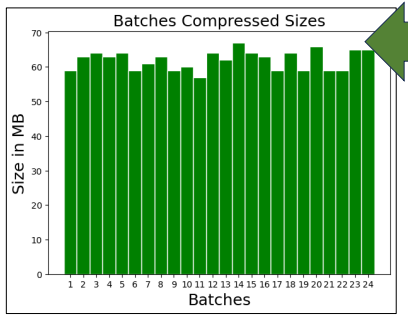
Batch capacity :
C = 152,000,000
(C obtained by linear regression)

Number of genome per batch varies



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

Evaluation strat. 1:
Allowing a capacity on distinct kmers.
The result remains somewhat imbalanced.

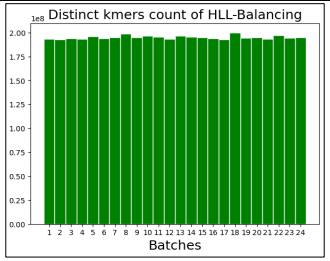


All Batches are well balanced
(between 59-67MB, max size 67MB)

Evaluation strat. 2:
Producing more balanced batches.
No control over the maximum distinct k-mer count per batch.

PHYLOGENETIC COMPRESSION

Batches Obtained From Strat. 2



Nb of genomes per batch varies but to a lesser extent compared to Strat. 1