5-6-2024

Efficient Algorithms for Large Scale Analysis of Viral Genome Sequencing Data

Daniel Novikov

Recommended Citation
Novikov, Daniel, "Efficient Algorithms for Large Scale Analysis of Viral Genome Sequencing Data." Dissertation, Georgia State University, 2024.
doi: https://doi.org/10.57709/36973044

Efficient Algorithms for Large Scale Analysis of Viral Genome Sequencing Data

by

Daniel Novikov

Under the Direction of Alex Zelikovsky, Ph.D.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2024

ABSTRACT

Extracting insights from the vast dataset of viral genome sequences collected throughout the COVID-19 pandemic requires the development of novel algorithms that are tailored to its unique properties. These properties, such as high sampling density, unambiguous knowledge of the phylogenetic root sequence, and completeness with respect to the virus's evolutionary history in humans, make it distinct among viral genome datasets. This dissertation details the development and application of advanced computational methodologies to analyze the SARS-CoV-2 genomic dataset. We introduce a suite of computational techniques that are tailored to this data, beginning with SPHERE, an algorithm for scalable phylogeny reconstruction that adapts to the high density of the genomic data. The next is (ε, τ)-MSN, which forms genetic relatedness networks by joining all possible minimum spanning trees and sensibly augmenting the network with additional edges, to capture groups of similar sequences. Furthermore, we present an unsupervised learning approach for finding a clustering of genomic sequences that minimizes cluster entropy. We also propose a method for implementing evolutionary jumps within genetic algorithms, simulating the punctuated equilibrium phenomena observed in SARS-CoV-2 sequencing data, which was shown to improve the speed of convergence for hard instances of the 0-1 Knapsack Problem. Collectively, these works detail new, efficient ways in which to consider modeling and extracting information from large scale viral sequencing datasets.

INDEX WORDS: SARS-CoV-2 Genomics, Phylogenetic Analysis, Computational Biology, Unsupervised Learning, Evolutionary Algorithms

Efficient Algorithms for Large Scale Analysis of Viral Genome Sequencing Data

by

Daniel Novikov

Committee Chair: Alex Zelikovsky

Committee: Pavel Skums

Murray Patterson

Serghei Mangul

Electronic Version Approved:

Office of Graduate Services

College of Arts and Sciences

Georgia State University

May 2024

DEDICATION

This dissertation is dedicated to my parents, Elina and Leonid, and my sister, Laura, whose unwavering support was instrumental to my completion of this research.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Alex Zelikovsky and my dissertation committee Dr. Skums, Dr. Patterson, and Dr. Manghul, for helping guide my focus throughout this dissertation. I would also like to thank my lab colleagues at Georgia State for being a consistent source of inspiration. Lastly, I'd like to thank my mentor at the CDC, Dr. Bill Switzer, and his lab for giving me the opportunities that I needed to thrive as a researcher.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 Favorable Properties of SARS-CoV-2 Genomic Data

The SARS-CoV-2 genomic data has properties that make it unique among other viruses. It is the first virus introduced to humans recently, allowing researchers to sequence it with high density throughout its evolution. For older viruses like HIV and HPV, their first cases in humans occurred prior to the next-generation sequencing technology needed to understand their genomic composition. As a result, their data contain only the recent products of their evolution, the leaves of the evolutionary tree. By contrast, immediate and high-frequency sequencing of SARS-CoV-2 since its inception in humans means that we are capturing the entire evolutionary tree of the virus in real-time, not just the leaf nodes. Consequently, the data contains a high-resolution image of the evolutionary tree of the virus, which may enhance the explanatory power of bioinformatics methods if tailored to extract this information.

Phylogenetic trees model the evolutionary relationships of a set of sampled sequences, providing a hierarchical clustering into a dendrogram, giving the evolutionary order in which those species emerged. At the leaf node level, we see all the species observed today, while internal nodes represent common ancestors, all the way up to the root, which is the inferred ancestor of all observed species. In this evolutionary model, internal node ancestors are not present in the observed data; rather, the existence of these ancestor species is supported by inference. Such inference is necessary, as many of the organisms we study have evolutionary histories that far predate our ability to study them.

However, this is not the case for SARS-CoV-2 data. As soon as the virus first appeared in late 2019 in Wuhan, researchers responded immediately by sequencing the genome of the collected viral samples. Such sequencing was dense and frequent and continued throughout the pandemic, as the virus mutated into its various strains and spread across the world. This mass sequencing effort has amassed over 16 million DNA sequences of the virus, available in the public database GISAID (2). This is out of approximately 700 million total cases of COVID-19, implying that more than 2.2% of all cases of SARS-CoV-2 were sequenced, and the viral genomes made publicly available.

Due to this unprecedented sequencing density, many key challenges in phylogenetic analysis are not challenges for SARS-CoV-2. We no longer have a need for ancestral inference, and there is no question of how to root the phylogenetic tree, as these sequences would be present in the observed data. It is for this motivation that we proposed SPHERE, which used a different evolutionary tree model that places observed sequences at internal nodes in the tree, not just on the leaf nodes.

Another key property of SARS-CoV-2 data is that its mutation rate is relatively low but still high enough that we can correlate mutations with transmission events. From the beginning of the pandemic in December 2019 until October 2020, each strain of the virus collected 2 mutations per month on average in the global population (3). Coupled with high sequencing density, this fact allows us to reconstruct transmission networks, providing a real-time epidemiological account of the transmission dynamics of the virus. By evaluating the efficacy of public health measures, we can also predict future outbreaks. This fact also allows for a genomic account of the evolutionary hierarchy among sampled genome sequences, by which we can determine which sampled sequence

is the ancestor of another.

1.2 Contributions to Large-Scale Genomic Data Analysis

This dissertation covers research projects that improve upon state-of-the-art bioinformatics methods developed for general viral genomes, with algorithms tailored to SARS-CoV-2 data. The resulting methods perform better on standard metrics for this data and can make stronger inferences than we could from the sequencing data of older viruses.

The first presented work is SPHERE, an algorithm for reconstructing the phylogenetic tree of SARS-CoV-2 genomes. This algorithm accepts aligned SARS-CoV-2 genome sequences and produces trees showing the evolutionary relationships between the genomes. It is scalable to the millions of sequences available in public databases, and uniquely, the trees it produces place observed sequences in internal nodes as well, not just the leaf nodes, as is usually done in other methods. This allows us to determine which sequence was the ancestor of another and infer transmission events. We showed that our method agrees with state-of-the-art method Nextstrain's phylogenetic trees while being orders of magnitude faster and more stable in its results across increasing time-densities of the data.

The second presented work is (ϵ, τ)-MSN, a method for creating genetic relatedness networks by joining all possible minimum spanning trees, augmented by additional edges whose weights are within a factor of ϵ of the minimum but no greater than τ.

The third presented work is Entropy based Clustering of SARS-CoV-2 Sequences, in which a Monte Carlo entropy minimization procedure is developed and applied to find the minimal entropy

clustering of a set of SARS-CoV-2 sequences.

The fourth presented work is Entropy based Clustering of SARS-CoV-2 Sequences, in which a Monte Carlo entropy minimization procedure is developed and applied to find the minimal entropy clustering of a set of SARS-CoV-2 sequences.

Lastly, the fifth and final presented work aims to improve the genetic algorithm with a new procedure called ”Evolutionary Jumps” which are similar to an observed pattern in SARS-CoV-2 known as punctuated equilibrium, wherein a new sequence emerges that contains pairs of mutations which were observed to be correlated across multiple different branches of the evolutionary tree.

CHAPTER 2

Scalable Reconstruction of SARS-CoV-2 Phylogeny with Recurrent Mutations

Daniel Novikov, Sergey Knyazev, Mark Grinshpon, Pelin Icer, Pavel Skums, Alex Zelikovsky

Journal of Computational Biology 2021

ABSTRACT

This paper presents a novel, scalable, character-based phylogeny algorithm for dense viral sequencing data called SPHERE (**S**calable **PH**ylog**E**ny with **RE**current mutations). The algorithm is based on an evolutionary model where recurrent mutations are allowed, but backward mutations are prohibited. The algorithm creates rooted character-based phylogeny trees, wherein all leaves and internal nodes are labeled by observed taxa. We show that SPHERE phylogeny is more stable than Nextstrain's, and that it accurately infers known transmission links from the early pandemic. SPHERE is a fast algorithm that can process more than 200,000 sequences in less then 2 hours, which offers a compact phylogenetic visualization of GISAID data.

2.1 Introduction

Equipped with the Next Generation Sequencing tools which are much more productive than ever before, the scientific community have collected an unprecedented amount of SARS-CoV-2 genomic data, enabling tracking the entire history of SARS-CoV-2 evolution with high precision (2). This tracking requires advanced phylogeny reconstruction software. However, the current state-of-the-art phylogeny algorithms were created to handle significantly sparser genomic data than what is available for SARS-CoV-2. The majority of the popular phylogenetic tools assume that only the

final product of evolution is available, while all intermediate evolutionary taxa are unknown. Also, these tools usually require significant computational resources, taking hours and sometimes days to reconstruct the SARS-CoV-2 phylogeny even for a small subset of the available genomes.

The SARS-CoV-2 sequencing data are similar to single cell sequencing data in cancer studies, where sequenced mutations from thousands of cancer cells offer a much closer look at cancer evolution. For such densely sequenced samples, perfect phylogeny models are more insightful than maximum likelihood models (4). The perfect phylogeny model requires each mutation to occur only once and never disappear. More realistic cancer evolution models allow widespread loss and recurrence of mutations (5; 6; 7).

In contrast to cancer evolution, in viral evolution backward mutations are rarer than recurrent mutations (8). In the evolution of the SARS-CoV-2 virus, recurrent mutations are mostly induced by the host's non-specific immune response. As they tend to be selectivity neutral, these mutations appear with higher frequency (9).

These properties of the SARS-CoV-2 genomic data, its density and a relatively high frequency of recurrent mutations, motivate the need for a parsimony-based phylogeny algorithm that is scalable to the entire collection of SARS-CoV-2 sequences available on GISAID (which numbers about 2.3 million sequences at the time of writing).

In this work, we follow the approach proposed in (10), which uses mutation trees (4) associated with character-based phylogenies that keep track of the accumulation of mutations in viral populations. We choose to employ parsimony-based phylogenetic analysis, because it explains evolutionary history with the fewest number of mutations to reproduce the variations in the genomic

data. Targeting both accuracy and resolution as aspects of information contained in a phylogenetic tree, the maximum parsimony approach yields the results that are at least as good or better than probabilistic approaches (11).

We propose SPHERE, **S**calable **PH**ylog**E**ny with **RE**current mutations, as an efficient phylogeny reconstruction method that incorporates this model. Using this tool, we analyzed the available GISAID data with over 300,000 genome sequences. We compared the trees produced by SPHERE with those produced by Nextstrain, and demonstrated that SPHERE trees are more stable with respect to extending datasets. Finally, we validated that the phylogeny trees produced by SPHERE more reliably discover valid transmission links than other state-of-the-art algorithms.

2.2 Methods

2.2.1 Most Parsimonious Phylogeny Problem

Given a set of aligned sequences, possibly containing missing positions, and a reference sequence with no missing positions, find a character-based phylogenetic tree that is rooted at the reference sequence, that has the minimum total edge length, and that does not admit backward mutations.

An algorithm to meet these criteria should infer the phylogeny tree from a set S of size n of aligned SARS-CoV-2 genome sequences. All of these sequences are built on the nucleotide alphabet (A,C,T,G) and may contain missing positions (N). The reference sequence is required to have no missing positions, i.e., no occurrences of N. Under the assumption of allowing recurrent mutations but not allowing backward mutations, our algorithm creates a maximum parsimony phylogeny tree for the given dataset rooted at the reference sequence.

Nodes in the phylogeny tree represent sequence haplotypes that match one or more sequences in the data. Edges in the tree are directed, and represent the ancestor/descendant relationship between two sequences. The length of an edge is the Hamming distance between the sequences that label its endpoints.

2.2.2 Algorithm Overview

For a given set S of aligned sequences and a reference sequence, the proposed Algorithm 1 finds a maximum parsimony phylogenetic tree on the haplotypes in S, rooted at the reference sequence, with no backward mutations allowed.

A tree rooted at the reference sequence is initialized, and all sequences in S are inserted into a queue. Each sequence is then assigned a set of positions at which the sequence contains a different nucleotide (i.e., a mutation) from the reference sequence. Each sequence's priority in the queue is determined by the size of its set of reference mutations, i.e., by its Hamming distance from the root. Finally, each sequence's parent is initialized to be the root by default.

Sequences are removed from the queue and added to the tree in increasing order of Hamming distance from the root. To achieve the minimal total edge length, the parent of a node must be on the shortest path from the root, and it must be the lowest such parent in the tree. By default, the parent of a sequence is the root, and we look for a better parent as we add the sequence to the tree. Once the parent is determined and the sequence is inserted into the tree, we fill any missing positions in the sequence from its parent. If a sequence's Hamming distance to its parent is 0, the sequence is collapsed to the parent node and a new node is not created.

Algorithm 1 Character-Based Phylogeny

Input: Set S of aligned sequences (with possible missing positions), reference sequence (with no missing positions);

Output: Character-based phylogenetic tree on aligned sequences, rooted at the reference sequence. Backward mutations are not allowed, recurrent mutations are allowed.

```
Initialize a tree rooted at reference, and a queue of all sequences
For each sequence, assign set of positions where the sequence differs from reference
Set root as initial parent of all sequences
while the queue is not empty do
    Dequeue minimum priority sequence x
    for each node v in reversed order of vertex set do
        Check if v is a parent of x
        Break when a parent is found.
    end for
    if Hamming distance to x's parent is 0 then
        collapse x to its parent
    else add x to the tree:
        Connect x to its parent
        Fill missing positions in x from the parent
    end if
end while
```

2.2.3 Parent Selection

When adding a sequence to the tree, the process of choosing its parent looks similar to a Dijkstra's shortest-path algorithm comparison. A sequence $\mathbf{u}$ is a parent of a sequence $\mathbf{v}$ if and only if (see Figure 2.1):

- $(\mathbf{root}, \mathbf{u}) + (\mathbf{u}, \mathbf{v}) = (\mathbf{root}, \mathbf{v})$;
- $\mathbf{u}$ is the lowest such node in the tree.

Together, these two conditions imply that $\mathbf{u}$ is on the shortest path from $\mathbf{root}$ to $\mathbf{v}$, that $\mathbf{u}$ immediately precedes $\mathbf{v}$, and that the total length of the tree after inserting $\mathbf{v}$ is the minimal possible.

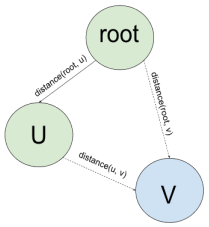

Figure 2.1 A visual representation of the parent selection constraint.

In the original implementation, see Algorithm 2, we updated the parents of all nodes in the queue on each insertion, as shown in Figure 2.2. Whenever we would pop a new sequence from the queue to add to the tree, we iterate through all nodes in the queue and check if the popped sequence is a better parent.

Algorithm 2 Parent Selection

```
1: Pop node x from queue and add it to the tree
2: Add edge from x's parent to x
3: for each node v in the queue do
4:     Update parent of v with x, if necessary
5: end for
```

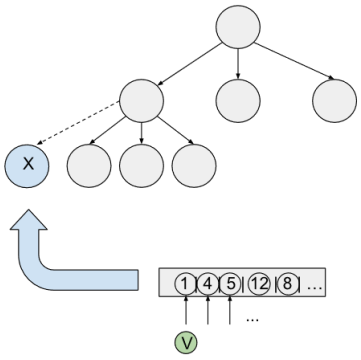

Figure 2.2 Choosing parents, original implementation. On each node insertion, for each sequence **v** remaining in the queue, check if the most recently inserted node **x** is a better parent of **v**.

With this approach, the parent selection procedure checks every possible edge between nodes

throughout the execution of the algorithm, and thus gives us a quadratic run time of $O(\text{Size of queue})$ comparisons for each of the $O(n)$ node insertions, where n is the number of sequences.

With this implementation, the performance of our algorithm exceeded quadratic runtime (Figure 2.3). This is too slow for our goal of designing a scalable tool capable of processing available SARS-CoV-2 genomic data in a reasonable amount of time.

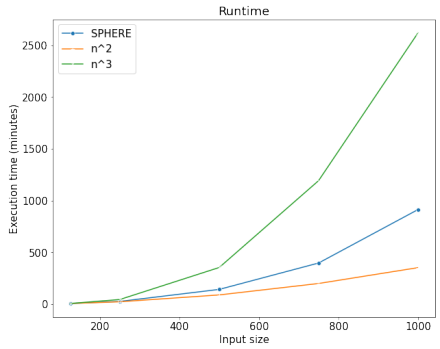

Figure 2.3 Runtime of our phylogeny algorithm in comparison to $O(n^2)$ and $O(n^3)$ runtimes. The blue curve represents the runtime of our algorithm, the orange curve represents $O(n^2)$ complexity, and the green curve $O(n^3)$ complexity. This version of parent selection admits a runtime that is slightly greater than $O(n^2)$.

2.2.4 Performance Improvements

Speeding Up Parent Selection

Originally, after each node insertion, we iterated through the queue updating parents as needed. This mode of parent selection results in a quadratic runtime complexity, as each node is compared to each other node throughout execution of the algorithm. Instead, as shown in Figure 2.4, we decided to iterate through the tree vertices when looking for parents, rather than updating parents in the queue. In the worst case, this still has a quadratic runtime; however, this mode of operation allows us to escape the parent selection procedure early when a parent is found.

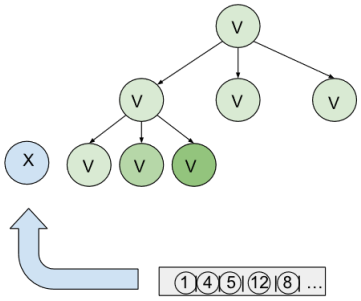

Figure 2.4 When adding node **x** to the tree, we search the nodes in the tree in reverse order of insertion (starting with the most recently inserted node) looking for parents that satisfy the triangle equality condition. We can stop iteration early every time we find a parent.

On each node insertion, our algorithm now iterates through the tree vertices in reverse order of insertion, looking for the parent that satisfies the triangle equality parent selection constraint shown in Figure 2.1. We can break this iteration through the tree early as soon as a parent better than the root is found, saving on the number of comparisons we need to make, see Algorithm 3.

Algorithm 3 Faster Parent Selection

```
Pop node x from queue and add it to the tree
for each node v in reversed order of vertex set do
    Check if v is a parent of x
    Break when a parent is found
end for
Add edge from v to x
```

We reduced the average complexity of our algorithm to below $O(n^2)$. Furthermore, the hidden complexity coefficient also dropped. In Figure 2.3, which illustrates the original runtime, we see that processing 1,000 sequences required almost 1,000 minutes of runtime. As is evident in Figure 2.5, this change to the parent selection algorithm increased our speed, so that we could now process 1,000 sequences in just a couple of minutes. However, 8,000 sequences still required

almost 6 hours of runtime.

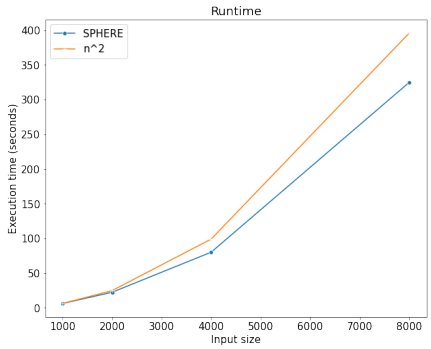

Figure 2.5 Performance after implementing the change to parent selection. This change has brought our algorithm's runtime down to below $O(n^2)$ in the average case. The hidden complexity coefficient has also decreased slightly, allowing us to process notably larger datasets in the same amount of time.

Speeding up Hamming distance

The length of SARS-CoV-2 genome is 30,000 nucleotides, and mutations have already been observed in more than 20,000 of them. However, any two available SARS-CoV-2 genome sequences differ by no more than 300 mutations.

We assigned each sequence a set of positions where it has mutated from the reference sequence. Then we compute the Hamming distance between two sequences as follows:

- The size of the symmetric difference between the two sets is added to the Hamming distance immediately.
- For each position in the intersection of the two sets, check if the sequences differ at those positions.

2.3 Results

2.3.1 Datasets

For comparison and evaluation purposes, we use the following five datasets:

- `C2C`: The Coast–to–Coast dataset consists of 168 global SARS-CoV-2 sequences, including 9 sequences from COVID-19 patients identified in Connecticut (12).
- `F22C`: This dataset consists of 1,293 global SARS-CoV-2 sequences, which are all GISAID sequences recorded up until February 22th, 2020, as well as the sequences in the `C2C` dataset. all GISAID sequences recorded up until February 22th, 2020, as well as the sequences in the C2C dataset.
- `M14`: This dataset consists of 9,286 global SARS-CoV-2 sequences, which are all GISAID sequences recorded up until March 14th, 2020.
- `M22`: This dataset consists of 21,473 global SARS-CoV-2 sequences, which are all GISAID sequences recorded up until March 22th, 2020.
- `ETL`: The Early Transmission Links dataset consists of 294 global SARS-CoV-2 sequences collected before March 9th, 2020. This dataset was constructed to match the 25 known transmission links. These transmission links were collected from news articles detailing transmissions prior to the pandemic declaration, in the MIDAS 2019 Novel Coronavirus Repository.

Since all sequences in the `C2C` and `ETL` datasets were recorded before March 14th, 2020, both are entirely contained in the `M14` and `M22` datasets.

2.3.2 Validation Metrics

Comparing Phylogenetic Trees One of the standard tools for comparing phylogenetic trees is the Robinson-Foulds (RF) distance, which is the size of the symmetric difference of the sets of bipartitions in two trees on the same set of taxa. Since the number of bipartitions in a SPHERE tree is significantly less than in the Nextstrain tree for the same taxa, we separately report two differences, each representing the number of bipartitions in one tree that are not present in the other tree.

However, the RF metric suffers from the several drawbacks including small range, over-sensitivity to minor differences, and assigning higher distances to more balanced trees (11). Therefore, we also report the triplet and quartet distances that provide more precise measures of dissimilarity that don't suffer from the same shortcomings as bipartions (11).

We use `Dendropy` (13) and `tqDist` (14) to calculate the RF distance and the triplet and quartet distances, respectively. Both tools require input trees in the Newick format with only leaves labeled by taxa. We convert a SPHERE tree to the Newick format as follows: each internal node labeled by a taxon is replaced by an unlabeled node with a child labeled by the same taxon; if a node is labeled by several taxa, we replace it with a new internal node, which is the parent of the new leaf nodes, each labeled by a single taxon (Figure 2.6).

Transmission Network Comparison When geographical metadata for SARS-CoV-2 sequences is available, the phylogeny trees produced by our method imply a SARS-CoV-2 transmission net-

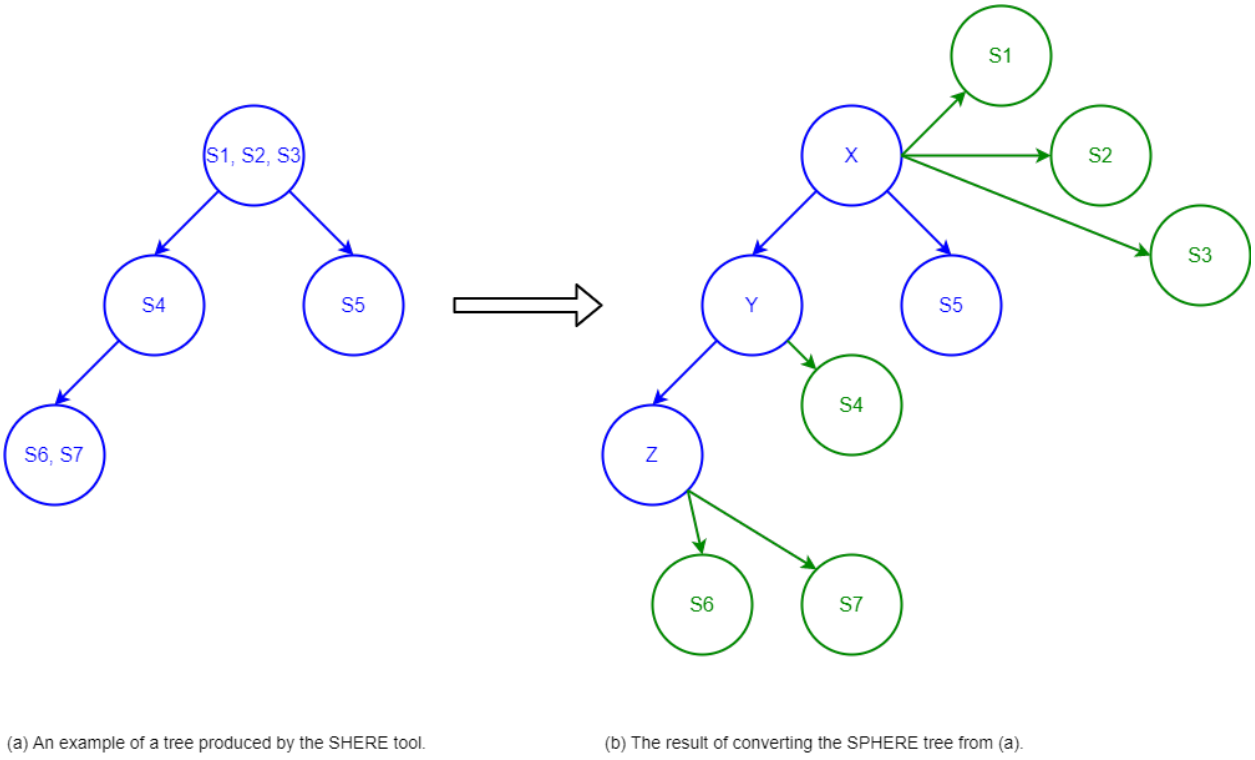

Figure 2.6 Example of converting a SPHERE tree into the Newick format. Three new internal nodes X, Y, and Z are introduced. The node X becomes the parent of S1, S2, and S3. The node Y becomes the parent of S4. The node Z becomes the parent of S6 and S7.

work. We analyze the predictive value of the transmission network by computing a phylogeny tree on the `ETL` dataset, extracting its implied network, and comparing it to the known transmission links that accompany the dataset.

In a SPHERE phylogeny, a directed transmission link between two locations is defined by a parent/child relationship in the tree between two sequences sampled at those locations. For matching sequences collapsed into a single node in the tree, we resolve the direction of their transmission link as earlier date $\rightarrow$ later date. We calculate precision and recall of the transmission network as follows:

$$\text{Precision} = \frac{\text{Number of true links predicted by the tool}}{\text{Total number of predicted links}}$$

$$\text{Recall} = \frac{\text{Number of true links predicted by the tool}}{\text{Total number of given true links}}$$

2.3.3 Phylogenetic trees for C2C data

The SPHERE phylogeny tree has all internal nodes annotated (Figure 2.7) in comparison to the Nextstrain tree (Figure 2.8). Nodes in both trees are colored by the locations they represent, where multi-color nodes in the SPHERE tree have assigned sequences from different locations. The sizes of the nodes in the SPHERE tree are proportional to the number of sequences they represent. Edges in the SPHERE tree are labeled by the number of mutations from parent to child haplotype. Some edges in the Nextstrain tree are labeled by codes of mutations.

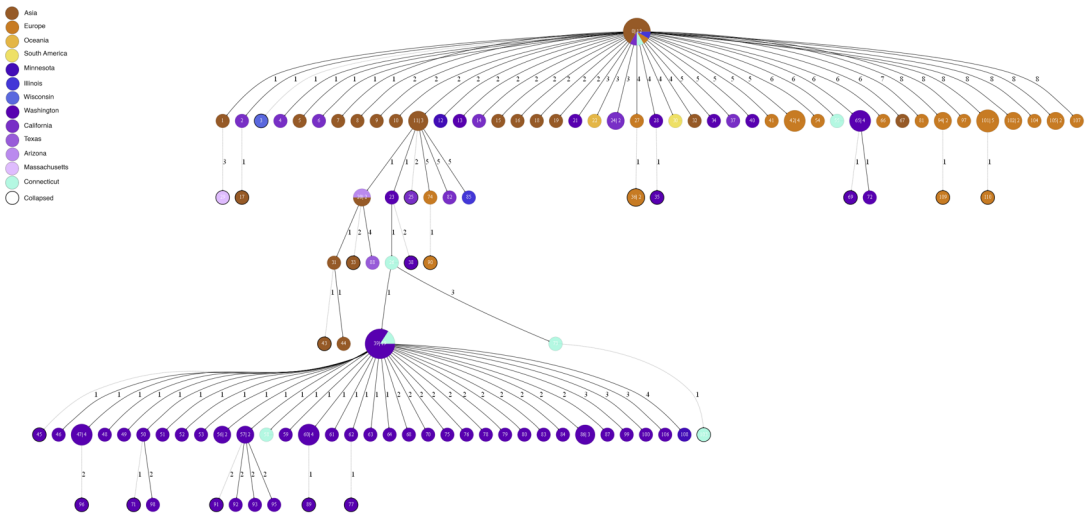

Figure 2.7 The phylogeny tree on the C2C dataset produced by SPHERE. Each edge is annotated by the number of mutations between the parent and the child. The sizes of the nodes represent the number of sequences assigned to the node. Multi-color nodes have assigned sequences from different locations.

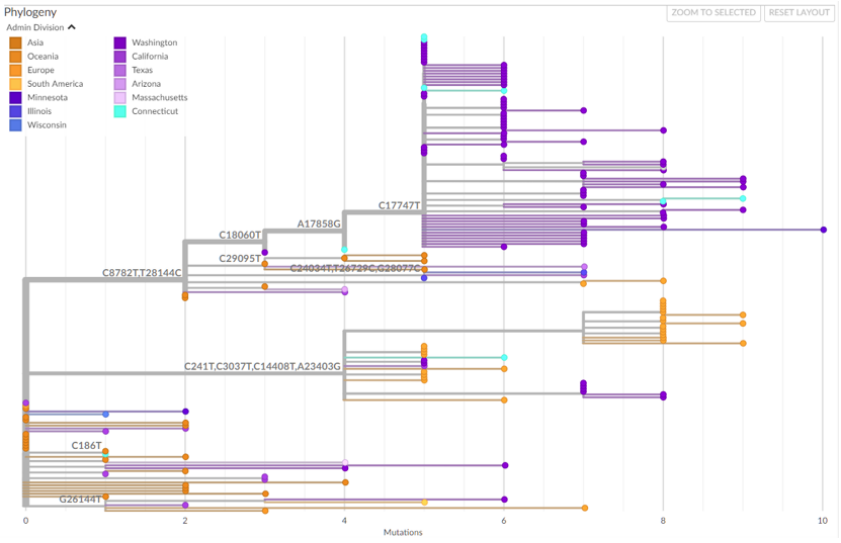

Figure 2.8 The phylogeny tree on the C2C dataset produced by Nextstrain.

2.3.4 Comparing Phylogenetic Trees

We compare eight trees created by applying the two phylogeny tools, SPHERE and Nextstrain, to the four datasets: C2C, F22C, M14, and M22 (see Table 2.1). Nextstrain prunes highly divergent sequences, leading to a slight reduction of the number of sequences for F22C and M14. The number of edges in SPHERE trees is much smaller than in Nextstrain trees since SPHERE does not introduce internal nodes and collapses taxa that agree with each other in the sequenced positions.

Tree	C2C_S	C2C_N	F22C_S	F22C_N	M14_S	M14_N	M22_S	M22_N
# Taxa	168	168	1,293	1,283	9,286	9,265	21,473	21,473
# Edges	110	277	694	2,265	3,843	17,108	9,010	39,722

Table 2.1 Eight phylogeny trees are created by applying SPHERE ("_S") and Nextstrain ("_N") to the four datasets C2C, F22C, M14, and M22.

	C2C_S	C2C_N	F22C_S	F22C_N	M14_S	M14_N	M22_S	M22_N
C2C_S	0	48.39	6.45	45.16	16.13	45.16	16.13	45.16
C2C_N	85.19	0	83.33	57.41	80.56	56.48	79.63	55.56
F22C_S	21.62	51.35	0	50.26	12.89	52.11	18.04	52.58
F22C_N	87.12	65.15	90.32	0	89.49	63.37	89.29	63.88
M14_S	38.1	50.0	19.14	49.76	0	49.68	18.28	51.43
M14_N	85.22	59.13	91.03	64.63	93.08	0	92.02	69.17
M22_S	43.48	52.17	25.7	50.0	26.19	47.48	0	50.68
M22_N	86.29	61.29	91.25	66.16	93.28	69.14	93.0	0

Table 2.2 The normalized directional RF distances between trees, given as percentages. Each entry represents the number of bipartitions in the row tree that are not present in the column tree, normalized by the total number of bipartitions in the row tree.

	C2C_S	C2C_N	F22C_S	F22C_N	M14_S	M14_N	M22_S	M22_N
C2C_S	0	30.44	9.41	31.19	11.13	36.23	10.18	31.4
C2C_N	30.44	0	22.53	8.36	21.38	18.0	22.38	7.59
F22C_S	9.41	22.53	0	49.98	0.26	57.37	0.6	52.07
F22C_N	31.19	8.36	49.98	0	49.92	25.33	50.18	18.98
M14_S	11.13	21.38	0.26	49.92	0	37.95	16.12	23.95
M14_N	36.23	18.0	57.37	25.33	37.95	0	44.59	22.48
M22_S	10.18	22.38	0.6	50.18	16.12	44.59	0	29.54
M22_N	31.4	7.59	52.07	18.98	23.95	22.48	29.54	0

Table 2.3 Triplets comparisons. Values represent the normalized triplet distance between each pair of trees, given as percentages.

	C2C_S	C2C_N	F22C_S	F22C_N	M14_S	M14_N	M22_S	M22_N
C2C_S	0	30.96	10.18	33.84	13.28	35.89	12.58	32.82
C2C_N	30.96	0	23.93	17.76	22.48	15.48	23.38	15.15
F22C_S	10.18	23.93	0	45.97	0.55	50.31	1.22	47.25
F22C_N	33.84	17.76	45.97	0	45.83	26.83	46.28	24.43
M14_S	13.28	22.48	0.55	45.83	0	33.84	23.55	31.68
M14_N	35.89	15.48	50.31	26.83	33.84	0	38.84	12.05
M22_S	12.58	23.38	1.22	46.28	23.55	38.84	0	37.4
M22_N	32.82	15.15	47.25	24.43	31.68	12.05	37.4	0

Table 2.4 Quartets comparisons. Values represent the normalized quartet distance between each pair of trees, given as percentages.

For each pair of trees, we report the directional Robinson-Foulds distance (see Table 2.2), the triplet distance (see Table 2.3), and the quartet distance (see Table 2.4). All distances are with respect to the common taxa between the trees being compared, normalized by the total number of bipartitions, triplets, or quartets, respectively.

Our results show that SPHERE is more stable than Nextstrain. Indeed, consider the chain of datasets C2C $\subset$ F22C $\subset$ M14 $\subset$ M22. A more stable phylogeny reconstruction method has lesser distances between trees for consecutive datasets. The corresponding normalized directed RF distances for SPHERE are 6.45%, 12.89%, and 18.28%, respectively; while for the trees produced

by Nextstrain the distances are much larger 57.41%,63.37%, and 69.17%, respectively. Similarly, the normalized triplet distances for SPHERE are 9.41%, 0.26%, and 16.12%, respectively; while for Nextstrain, they are 8.36%, 25.33%, and 22.48%, respectively. Finally, the normalized quartet distances for SPHERE are 10.18%, 0.55%, and 23.55%, respectively; while for Nextstrain, the quartet distances are 17.76%, 26.83%, and 12.05%, respectively. We can see that in most cases SPHERE method is more stable than Nextstrain.

2.3.5 Inferring Transmission Links

We have compared precision and recall of transmission networks inferred with SPHERE, the ILP-based character state phylogeny (CS-phylogeny) (10) and the character-based phylogeny NETWORK5011CS (15). SPHERE has the best recall over existing methods (Table 2.5). Note that all methods have small precision because the ETL dataset contains only verified transmission links. The number or such links is only 25 for 294 nodes. There should be other transmission links, however they are not validated. SPHERE has comparable precision to other methods that indicates that all methods output similar number of predicted transmission links.

Tool	Recall %	Precision %
SPHERE	**88**	4.3
CS-phylogeny	80	4.76
NETWORK5011CS	72	4.99
SPHERE-directed	84	4.3

Table 2.5 Comparison of SPHERE with CS-phylogeny and NETWORK5011CS tools without taking in account the transmission direction. SPHERE-directed also takes in account the transmission direction.

2.3.6 Runtime

We ran SPHERE on the cluster hardware consisting of 128 cores Intel(R) Xeon(R) CPU E7-4850 v4 CPU @ 2.10GHz, with 3 TB of RAM, running Ubuntu 16.04.7 LTS.

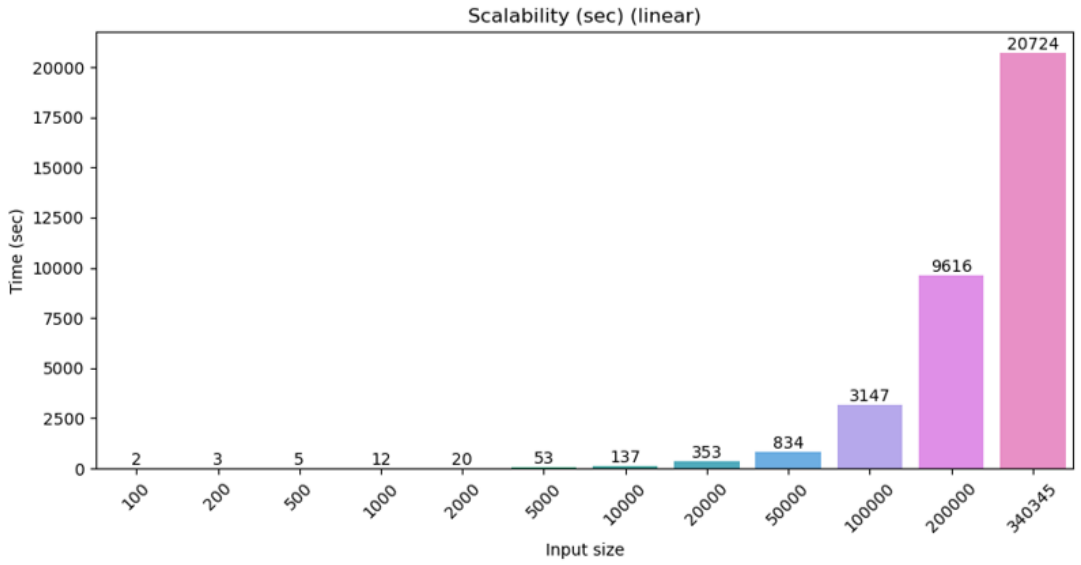

Figure 2.9 A graph of input size vs runtime. Units are in seconds. This figure highlights the significant performance improvements observed after optimizing the parent selection and Hamming distance methods.

Figure 2.9 shows that SPHERE is indeed a scalable method with a subquadratic runtime. For example, it is able to process 200,000 sequences in two hours, while Nextstrain requires 2 days to process 21,000 sequences on the same hardware, with 32 cores dedicated to the process.

2.4 Conclusion and Future Work

It is shown that the development of a character-based shortest-path phylogenetic tree is viable. First, a shortest-path phylogeny is fast and scalable. Second, the resulting maximum parsimony trees produced by our method are more stable than the Nextstrain's maximum likelihood tree.

Third, the inferred transmission network quality is higher or comparable with existing tools. We plan to incorporate sparse backward mutations into the algorithm and add Steiner points corresponding to internal vertices.

CHAPTER 3

A Novel Network Representation of SARS-CoV-2 Sequencing Data

Sergey Knyazev, Daniel Novikov, Mark Grinshpon, Harman Singh, Ram Ayyala, Varuni Sarwal, Roya Hosseini, Pelin Icer Baykal, Pavel Skums, Ellsworth Campbell, Serghei Mangul, Alex Zelikovsky

ISBRA 2021

ABSTRACT

The unprecedented level of genome sequencing during the SARS-CoV-2 pandemic brought about the challenge of processing this genomic data. However, the state-of-the-art phylogenetic methods were mostly designed for analyzing data that are significantly sparser and require extensive subsampling of strains. We present (ε, τ)-MSN, a novel tool that reconstructs a viral genetic relatedness network based on genetic distances, that can process hundreds of thousands of sequences in under several hours. We applied (ε, τ)-MSN to the global COVID-19 outbreak data and were able to build a genetic network on more than 100,000 SARS-CoV-2 sequences. We show that (ε, τ)-MSN can accurately detect transmission events and build a genetic network with significantly higher assortativity with respect to continent and country attributes of SARS-CoV-2 samples. The source code for this software suite is available at `https://github.com/Sergey-Knyazev/eMST`.

3.1 Introduction

The tendency of RNA viruses to rapidly change their genomes is the major reason for vast inter-host and intra-host viral diversity and fast viral evolution (16). This evolution can be tracked with high precision thanks to the rapid growth of capacity for viral genome sequencing that has been occurring over the past decade (17). To see what that data can reveal about viral evolution and transmission, numerous analytical methods have been proposed (18; 19; 20; 21; 22; 23; 24; 25; 26; 27; 28; 29). One of the essential tasks in analysis of viral genomic data is representation of genetic relatedness between viral samples. For this purpose, standard phylogenetic methods as well as network-based methods that were initially applied only to specific viruses such as HIV (30) and HCV (31) have been proposed for SARS-CoV-2 analysis too.

The standard approach for representation of viral genomic data is based on phylogenetic trees. In general, finding an optimal phylogenetic tree that fits a biologically relevant model of evolution, e.g., using maximum likelihood approaches, is an NP-hard problem (32). As a result, tools for phylogeny reconstruction are too slow or inaccurate on large datasets. Indeed, the quality of advanced phylogeny tools' outputs significantly decreases with the growth of the number of input sequences. For this reason, these methods rely heavily on subsampling to maintain the acceptable quality of phylogeny reconstruction.

Network-based methods, designed to represent the most likely pairs of connected viral genomes rather than viral evolution, offer an alternative approach to phylogenetic tree reconstruction (18; 19). This is convenient because establishing genetic relatedness between viral samples is the basic step in viral outbreak investigations. Furthermore, network-based methods are more promising in

the era of big genomic data because they are much more simple and scalable than phylogenetic methods. The success of using networks-based method for HCV and HIV outbreak investigations (33; 34; 35; 36) motivated the Centers of Disease Control and Prevention (CDC) to adopt them for wider use.

Currently, the CDC is actively advancing the following two alternative approaches for tracing genetic relatedness:

1. The Global Hepatitis Outbreak and Surveillance Technology (GHOST) is a cloud-based system that allows users to analyze and visualize data regardless of computational expertise. It uses the intra-host viral haplotypes for tracing HCV epidemics, the Hamming distance between genomic sequences as a metric for genetic relatedness, and k-step networks (introduced as minimal spanning networks in (18)) for choosing genetically related vertices (33; 37; 31).

2. For HIV outbreaks, the current tool of choice is HIV-TRACE, which performs high-scale analysis of genomes in HIV surveillance systems (30; 38). It identifies groups of closely related genomes using the Tamura-Nei 93 (TN93) genetic distance metric (39). HIV-TRACE accepts either a Sanger sequence or a consensus sequence from NGS sequencing experiment for each individual. These sequences are then used to look for evidence of relatedness between them. Relatedness is suspected when the genetic distance between sequences is below a certain threshold (we will refer to this construction as τ-networks). This simple approach has demonstrated high reliability for detecting rapidly growing outbreak clusters (40).

In this paper we introduce a novel network-based method for constructing genetic similarity networks, which generalizes both GHOST and HIV-TRACE.

First, we present ε-MSN, an algorithm that generalizes the methodology implemented in GHOST. Given a set of genomic sequences along with distances between them, ε-MSN builds a network that includes the union of all minimal spanning trees (MST), as well as some additional edges: those that were close to being included in an MST, i.e., edges whose weights are just a little (within a specified parameter ε) more than the largest edge weight in a path connecting its endpoints in an MST.

We further improve ε-MSN by incorporating the τ-networks of HIV-TRACE. This second tool, (ε, τ)-MSN, effectively builds an ε-MSN first, but then removes all edges that are too heavy, i.e. all edges whose weight is more than a specified parameter τ.

We compare our tool with minimum spanning networks (MSN) and τ-networks (41; 30) and demonstrate that ε-MSN and (ε, τ)-MSN results are of better quality in terms of attribute assortativity, recall, and precision. Our method allows us to construct phylogeny networks and report results for SARS-CoV-2 sequence datasets having up to more than hundred thousand strains, with a potential of being scalable to much larger datasets.

3.2 Methods

When analyzing evolutionary relationships in a set of genomes, the first step may be to create the complete graph of the genomes and measure the distances between them. Such a complete graph incorporates all those relationships that we would like to see, but it has too many edges,

making it impossible to discern the useful information from it. In fact, as a complete graph, it contains all possible edges, including those between really distant genomes, i.e., edges whose weights represent very long genetic distances. To extract edges connecting related genomes only, we introduce a threshold parameter τ and remove edges with length greater than τ, thus getting rid of edges that are too long.

Another idea for improving the graph is to use minimal spanning trees (MST). Generally, the standard greedy algorithms for building MST's always pick the shortest of available edges towards the closest neighbor genomes. However, if one neighbor of a vertex (genome) is just slightly closer than some others, the MST would include the shortest edge only and leave the rest out, even though all of them have a close enough genetic relationship with each other. Thus we introduce another parameter ε that allows us to include edges that are only slightly longer than the closest neighbors.

In summary, our goal is to build a graph $G = (V, E)$, where each vertex $v \in V$ represents a viral genomic sequence, and where an edge $e \in E$ connects two vertices u and v whenever u and v represent genetically related viral genomes. Previous studies proposed to take G as the union of all possible MST's in the weighted graph whose nodes are the viral genomes and whose edges are weighted by a genetic distance between these genomes (18; 33). We extend this approach and propose to include some additional edges.

[ε-MSN] Given an $\varepsilon \geq 0$, ε-MSN is a graph in which two vertices u, v are connected if $d(u, v) \leq (1 + \varepsilon) \cdot d(x, y)$, where $d(x, y)$ is the weight of the heaviest edge on the u-v path in an MST.

An efficient algorithm for constructing ε-MSN is given in Algorithm 4.

Following the τ-network methodology, we allow setting a threshold τ for additionally filtering out edges that are too long.

[(ε, τ)-MSN] Given an $\varepsilon \geq 0$ and a $\tau > 0$, (ε, τ)-MSN is a graph in which two vertices u, v are connected if they are connected in ε-MSN and $d(u, v) \leq \tau$.

(ε, τ)-MSN is a generalization of both the MSN and τ-network methods. Indeed, MSN is a special case of (ε, τ)-MSN if we set the parameters to $\varepsilon = 0$ and $\tau = \infty$. Similarly, τ-network is a special case of (ε, τ)-MSN when $\varepsilon = \infty$.

An implementation of the (ε, τ)-MSN algorithm as a software tool is freely available on GitHub at `https://github.com/Sergey-Knyazev/eMST`. The software can accept sequences in FASTA format and compute (ε, τ)-MSN using either of the two genetic metrics of choice, Hamming distance or TN93. The user can also provide their own distance matrix in the list of edges format.

For efficiently computing Hamming distance between sequences, we implemented the following speed up technique (42). Initially, we infer a consensus of all input sequences. Then, for each sequence in the input, we determine a set of positions where each sequence has mutated from the consensus. Finally, for each pair of sequences the Hamming distance is computed in two steps. First, we initialize the value of Hamming distance to be the size of the symmetric difference between the two sets. Second, for each position in the intersection of the two sets, we check if the sequences differ at this position, and if they do, we increment the value of Hamming distance by one.

Algorithm 4 (ε-MSN)

MSA: Multiple Sequence Alignment of Strains
G: Fully connected distance graph obtained from strains
ε: $\varepsilon \geq 0$, parameter for ε-MSN
function ADDEPSILONEDGES(MST, $LongestEdge$, E, ε)
    for $(x, y) \in E$ **do**
        if $d(x, y) \leq (1 + \varepsilon) \cdot (LongestEdge(x, y))$ **then**
            `add` (x, y) `to` MST
        end if
    end for
    return MST
end function
function GETLONGESTEDGES(MST, E)
    for $(x, y) \in E$ **do**
        $LongestEdge(x, y) \leftarrow \max(e_i)\ \forall e_i \in MST_{x \to y}$
    end for
    return $LongestEdge$
end function
procedure EMSN($A = MSA$ or G, ε)
    `If` $A = MSA$, `obtain` $G(V, E)$ `using a distance metric (e.g., Hamming, TN93, etc)`
    $MST \leftarrow getMST(G)$
    $LongestEdge \leftarrow getLongestEdges(MST, E)$
    $eMST \leftarrow addEpsilonEdges(MST, LongestEdge, E, \varepsilon)$
end procedure

3.3 Results

To demonstrate the usability of the (ε, τ)-MSN methodology, we benchmarked it against other methods.

First, we compared ε-MSN and (ε, τ)-MSN with the two state-of-the-art methods for constructing τ-networks (used in HIV-TRACE) and minimum spanning networks (used in GHOST) on COVID-19 sequences available from GISAID using assortativity analysis.

Second, we examined ε-MSN, (ε, τ)-MSN, τ-networks, and MSN on their ability to infer transmission events and compared them with other available tools including CS-phylogeny (43),

NETWORK5011CS (44), RAxML (45), outbreaker (46), and phybreak (47). For this test, we used a SARS-CoV-2 sequencing dataset with known ground truth about infective transmission events, and we measured precision and recall of each of the methods when applied to infer these events from the sequencing data.

Third, we showed the scalability potential of (ε, τ)-MSN to process networks up to a size of more than hundred thousand of sequences.

3.3.1 Datasets

1. For comparison of the methods via assortativity analysis, we used the coast-to-coast (C2C) dataset, which contains 168 SARS-CoV-2 sequences collected from different countries, including 9 sequences from COVID-19 patients identified in Connecticut (12). Each sample in this dataset has geographical attributes named Continent, Country, and Division.

2. For comparison of precision and recall of the methods in inferring transmission links we used the Early Transmission Links (ETL) dataset, which consists of 293 global SARS-CoV-2 sequences collected before March 9th, 2020. Each sequence has a known country of origin. This dataset was constructed to match the 25 known country-to-country transmission links that were collected from news articles detailing transmissions prior to the pandemic declaration, in the MIDAS 2019 Novel Coronavirus Repository (43).

3. For scalability analysis, we created datasets consisting of the initial 100, 200, 500, 1000, 2000, 5000, 10000, 20000, 50000, and 100000 SARS-CoV-2 sequences from the masked multiple sequence alignment from GISAID. To generate these datasets, we ordered se-

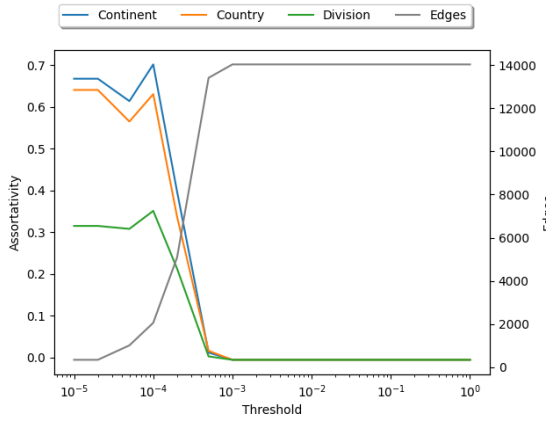

Figure 3.1 Attribute assortatitivity on the C2C dataset for different values of edge threshold τ, using τ-network with TN93 distance.

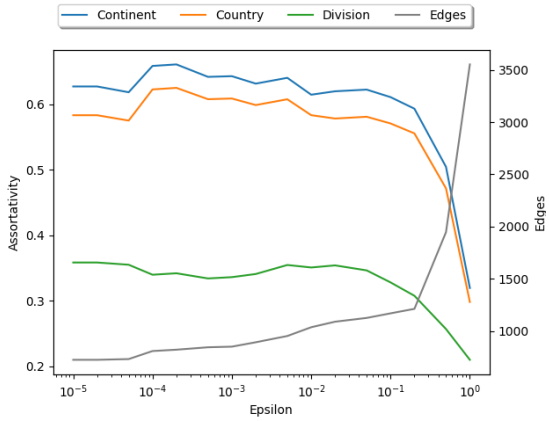

Figure 3.2 Attribute assortativity on the C2C dataset for different values of ε, using ε-MSN with TN93 distance and edge threshold $\tau = \infty$.

quences by date, and picked the earliest date when the number of sequences exceeds the desired number.

3.3.2 Assortativity analysis

We ran MSN, ε-MSN, τ-network, and (ε, τ)-MSN on the C2C dataset using TN93 as measurement of genetic relatedness, and we evaluated attribute assortativity for continent, country, and division.

Figure 3.1 shows the dependence of attribute assortativity on $\tau \in [0, 1]$ for τ-network. The optimal value for continent assortativity is 0.702 when τ is 0.0001. Figure 3.2 shows the dependence of attribute assortativity on $\varepsilon \in [0, 1]$ for ε-MSN. The optimal value for continent assortativity is 0.661 when ε is 0.0002. Figure 3.3 shows the dependence of attribute assortativity on $\varepsilon \in [0, 1]$ for (ε, τ)-MSN, with fixed threshold $\tau = 0.0001$ that maximized assortativity in the τ-network analysis from Figure 3.1. The optimal value for continent assortativity is 0.7573 when ε is 0.0002.

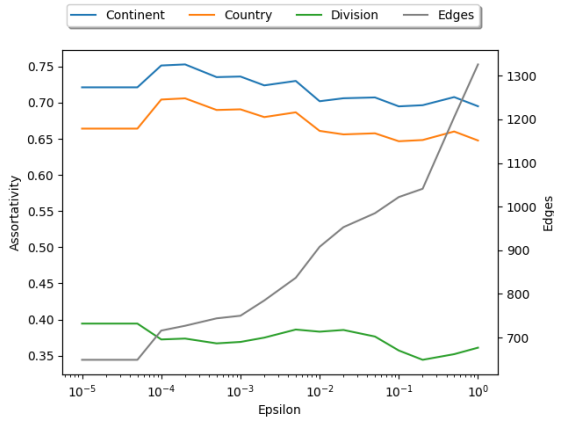

Figure 3.3 Attribute assortatitivity on the C2C dataset for different values of ε, using (ε, τ)-MSN with TN93 distance and edge threshold $\tau = 0.0001$. The maximum assortativity occurs when $\varepsilon = 0.0002$.

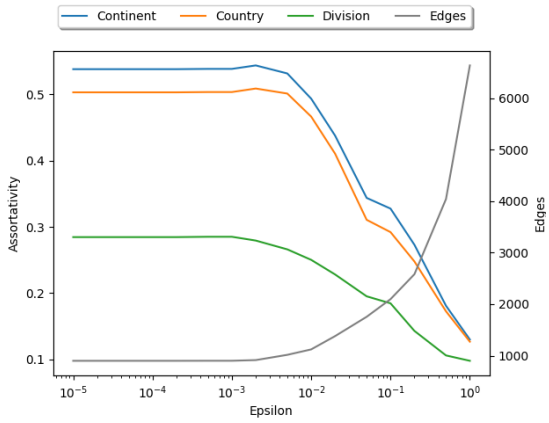

Figure 3.4 Attribute assortatitivity on the C2C dataset for different values of ε, using ε-MSN with Hamming distance.

Figure 3.4 shows the dependence of attribute assortativity on $\varepsilon \in [0, 1]$ for ε-MSN with Hamming distance instead of TN93. The optimal value for continent assortativity is 0.546 when ε is 0.002.

Table 3.1 shows that the maximum assortativity on the C2C dataset was achieved by (ε, τ)-MSN's mixture of both parameters, with $\varepsilon = 0.0002$ and $\tau = 0.0001$. The resulting continent assortativity value of 0.753 is higher than the other methods, and the same is seen for country and division assortativity.

Method	ε	τ	No. of edges	Assortativity		
				Continent	Country	Division
MSN	0	∞	717	0.626	0.581	0.360
τ-network	∞	0.0001	2056	0.702	0.631	0.351
ε-MSN	0.0002	∞	821	0.661	0.625	0.342
(ε, τ)-MSN	0.0002	0.0001	727	**0.753**	**0.706**	**0.374**

Table 3.1 This table shows the attribute assortativity values for optimal choices of ε and τ for MSN, ε-MSN, (ε, τ)-MSN, and threshold-based network, each using TN93 distance.

We find that (ε, τ)-MSN performs the best in terms of country, continent, and division assortativity values across all four methods.

3.3.3 Transmission network analysis

We evaluated the precision and recall of (ε, τ)-MSN on the ETL dataset. To evaluate the transmission network quality of (ε, τ)-MSN, we define an undirected transmission link to be the pair of sequence locations of the two vertices connected by an edge in (ε, τ)-MSN. The set of all unique undirected transmission links forms the transmission network.

For each method shown in Table 3.2, we produced its transmission network and evaluated the precision and recall against the known links provided in the ETL dataset. We calculate precision as the ratio of the number of known true links predicted by the method and the total number of predicted links, and we calculate recall as the ratio of the number of known true links predicted and the total number of known true links in the ETL dataset.

Table 3.2 shows that MSN performed best in Precision and F1-Score. τ-network performed best in Recall but not as well in precision or F1-score. ε-MSN and (ε, τ)-MSN performed comparably well to MSN, and together these network based methods all outperformed the other standard methods being compared.

3.3.4 Scalability analysis

To examine scalability of the proposed methods, we applied the ε-MSN tool to datasets of increasing sizes of up to several hundred thousand sequences. For each of these datasets, we ran ε-MSN in TN93 mode and Hamming distance mode separately and recorded the running times. Figure 4.1

shows the results of the analysis. We see that ε-MSN has a quadratic runtime in both modes, but that Hamming distance is significantly faster because of its efficient implementation.

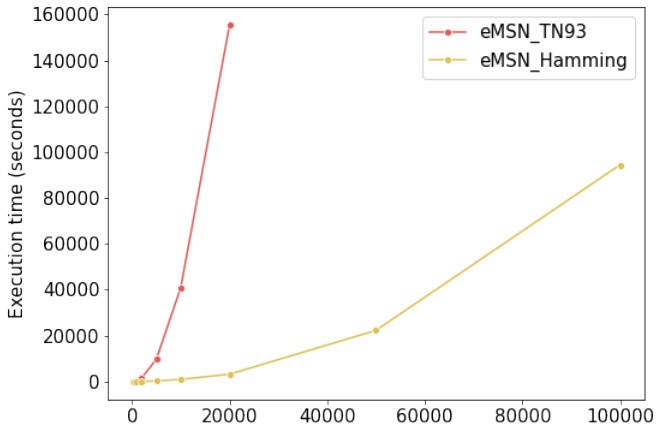

Figure 3.5 Runtime analysis of ε-MSN on increasing input sizes. ε-MSN is a quadratic algorithm in both TN93 and Hamming distance modes, although Hamming distance, with its efficient implementation, is much faster.

Tool	Recall*	Precision*	F1-Score*
MSN ($\varepsilon = 0, \tau = \infty$, TN93)	80%	**7.6%**	**0.139**
τ-network ($\varepsilon = \infty, \tau = 0.0001$, TN93)	**96%**	2.5%	0.049
ε-MSN ($\varepsilon = 0.0002, \tau = \infty$, TN93)	80%	7.4%	0.135
(ε, τ)-MSN ($\varepsilon = 0.0002, \tau = 0.0001$, TN93)	72%	6.6%	0.121
CS-phylogeny	80%	4.76%	0.090
NETWORK5011CS	72%	4.99%	0.093
RAxML	64%	4.26%	0.080
Bitrugs	52%	3.38%	0.063
outbreaker	28%	5.83%	0.097
phybreak	4%	0.83%	0.076

Table 3.2 Recall and precision comparison across different methods ran on the ETL dataset. MSN methods were ran using the TN93 distance metric. Recall is defined as the ratio of known true links formed by the tool to the total number of known true links. Precision is defined as the ratio of known true links formed by the tool to the total number of links formed by the tool. F1-Score is defined as the twice the product of precision and recall divided by the sum of precision and recall. * The ground truth is only partially known.

3.4 Conclusion

We have developed two versions of a new network-based tool, ε-MSN and (ε, τ)-MSN, which generalize the minimal spanning networks and τ-network approaches to representing genetic relationships.

We compared the proposed tools with other network-based methods using attribute assortativity values. The experiments show that (ε, τ)-MSN Hamming distance does not perform as well as with TN93 distance. With TN93, (ε, τ)-MSN outperforms all the other methods in continent, country, and division attribute assortativity.

Further, we validated multiple tools, including the proposed ones, on known transmission networks. We evaluated recall, precision and F1-score for each tool. We found that network-based tools perform better than the others, including those that are phylogeny-based.

The results validated the ε-MSN network and showed that the structure of the ε-MSN network correlates with phylogenetic trees. ε-MSN is interpretable, integrable and scalable.

Users of our proposed tools can fit the parameters ε and τ to any dataset using the same methodology we used in our analysis, namely, fixing one parameter and varying the other, then fixing the other and varying the first.

Our methodology is implemented in MicrobeTrace(29), a tool currently in use by the CDC for viral outbreak investigation.

CHAPTER 4

Entropy Based Clustering of Viral Sequences

Akshay Juyal, Roya Hosseini, Daniel Novikov, Mark Grinshpon, Alex Zelikovsky

ISBRA 2022

Journal of Computational Biology 2023

ABSTRACT

Clustering viral sequences allows us to characterize the composition and structure of intrahost and interhost viral populations, which play a crucial role in disease progression and epidemic spread. In this paper we propose and validate a new entropy based method for clustering aligned viral sequences considered as categorical data. The method finds a homogeneous clustering by minimizing information entropy rather than distance between sequences in the same cluster. We have applied our entropy based clustering method to viral sequencing data. We report the information content extracted from the sequences by entropy based clustering. Our method converges to similar minimum-entropy clusterings across different runs and limited permutations of data. We also show that a parallelized version of our tool is scalable to very large datasets.

4.1 Introduction

Clustering viral sequences allows us to characterize the composition and structure of intrahost and interhost viral populations, which play a crucial role in disease progression and epidemic spread. For intrahost populations, clustering allows us to detect the distinct viral variants present in the patient, including minor low-frequency variants, which can cause immune escape, drug

resistance, and an increase of virulence and infectivity (48; 49; 50; 51; 52; 53; 54). Furthermore, such minor variants are often responsible for transmissions and establishment of infection in new hosts (55; 56; 57).

In this paper we propose a Monte Carlo entropy minimization method for clustering of viral sequences considered as categorical data. The method finds a homogeneous clustering by minimizing information entropy rather than distance between sequences in the same cluster. We discuss advantages and disadvantages of both entropy and distance based approaches, and further validate the meaningful information content extracted by entropy based clustering. We demonstrate that the proposed method is stable, moving towards the same minimal entropy configuration across multiple runs. We also show that it is fast and scalable to hundreds of thousands of sequences.

By clustering viral populations across different hosts, we determine major strains of closely related viral samples, which is helpful for tracking transmissions and informing public health strategies (58). For transmission tracking, clustering can identify the source of an outbreak and whether the source is present in the sampled population. It can also determine whether two viral samples belong to the same outbreak, and whether one infected the other (59). Therefore, using clustering to obtain an accurate characterization of viral mutation profiles from infected individuals is essential for viral research, therapeutics, and epidemiological investigations.

Viral sequences are strings from a fixed nucleotide alphabet, and hence they can be viewed as vectors of categorical data. In the best possible clustering, sequences in each cluster will be as homogeneous as possible in each site. Typically, this is achieved by minimizing the Hamming distances between sequences within the same cluster or the distances to the cluster's consensus

(60). However, Hamming distance carries the implicit assumption that all mutations at all sites are of equal cost, and also it does not consider the distribution of values in a given category, counting each mismatch equally.

In this paper we propose to use entropy based clustering for viral sequences. Entropy considers the distributions of nucleotides in each site, allowing us to capture different kinds of mismatches. Minimizing entropy instead of distance also avoids the need to introduce the abstraction of equal transition costs, implicit in Hamming distance. Thus, entropy as an objective for clustering makes fewer assumptions on the data and is more informative for clustering categorical data.

We have applied our entropy based clustering method to the viral sequencing data. The unprecedented effort in sequencing its genome has created vast databases of sequences, such as GISAID (61). Clustering techniques can provide new insights into the evolution of the virus, assist with phylogenetic and phylodynamic analyses, and offer new tools for constructing transmission networks to help with understanding the spread of the pandemic (58).

We validate effectiveness of the entropy based clustering of viral sequences on real datasets. We measure the information content extracted from the sequences by the resulting clustering. We also demonstrate that our method converges to the same minimum-entropy clustering across different runs, thus marking stability of the method. Finally, we describe a tag selection procedure, which selects the highest entropy sites to represent sequences leading to a significant decrease in runtime without major loss of information.

4.2 Methods

In this section, first we define the entropy and Hamming distance of clustering of a set of aligned viral sequences. We use each of these two measures as an objective function to be minimized for clustering. Then we describe a Monte Carlo clustering algorithm, which is a modification of an algorithm proposed in (62). Finally, we describe a tag selection preprocessing step, which significantly reduces the runtime of the algorithm.

4.2.1 Entropy Based Clustering of Viral Sequences

Entropy of a category across a set of categorical vectors quantifies the heterogeneity of values in the category. Entropy is low when a single value is highly frequent, and it is at its highest when all values are equally frequent in the category. Since we are treating viral sequences as vectors of categorical data, the categories here are sites along the sequence, and their values are from the nucleotide alphabet $\{,,,\}$. A clustering with minimal entropy will have the highest possible homogeneity of nucleotides in each site for sequences in the same cluster.

Formally, we have a set S of aligned nucleotide sequences on a set X of genomic sites. Since the sequences $s \in S$ are aligned, they can be viewed as rows of a matrix, and the sites $x \in X$, can be viewed as columns of this matrix. Let the alphabet $= \{,,,\}$ be the four nucleotides, not counting the gap () character. Following (62), the entropy $H(C_x)$ of a site $x \in X$ in cluster C is defined as

$$H(C_x) = -\sum_{s \in C} \sum_{a \in} p(s_x = a) \cdot \log p(s_x = a). \tag{4.1}$$

Note that $p(s_x = a)$, the probability that a sequence $s \in C$ has nucleotide $a \in$ at site x,

essentially amounts to the *relative frequency* of the nucleotide a in cluster C at site x (ignoring gap characters).

The entropy $H(C)$ of a cluster C of viral sequences on a set X of sites is then defined as

$$H(C) = \sum_{x \in X} H(C_x), \tag{4.2}$$

that is, we simply sum up the entropies at the individual sites.

Finally, given a clustering of the set S, the *entropy* of is defined as follows:

$$H() = \sum_{C \in} \frac{|C|}{|S|} \cdot H(C) = \frac{1}{|S|} \sum_{C \in} |C| \cdot H(C). \tag{4.3}$$

In other words, the entropy of clustering is the sum of cluster entropies weighted by their relative sizes.

In (62), the authors prove that the entropy defined in equation (4.3) is a convex function, allowing any optimization procedure to reach a global minimum. It is because of this property that we can use techniques aimed directly at minimizing clustering entropy as the objective.

4.2.2 Hamming Distance Based Clustering of Viral Sequences

Similarly, we define a different clustering objective as Hamming distance (HD) instead of entropy. This objective is the sum of the Hamming distances from each sequence s to the consensus of the cluster containing s.

Formally, for a cluster C and for a site $x \in X$, the Hamming distance from the consensus letter in this cluster at this site is

$$HD(C_x) = \sum_{a \in} C_x(a) - \max_{a \in} \{C_x(a)\} \tag{4.4}$$

where $C_x(a)$ is the number of occurrences of the letter $a \in$ in site x in cluster C.

Then the Hamming distance $HD(C)$ of a cluster C of viral sequences on a set X of sites is defined as

$$HD(C) = \sum_{x \in X} HD(C_x), \tag{4.5}$$

and the *Hamming distance* of the clustering is defined as

$$HD() = \sum_{C \in} HD(C). \tag{4.6}$$

4.2.3 Algorithm Description

In general, Monte Carlo methods optimize an objective by attempting random changes and accepting a change only if it improves the objective. In our case, the objective is to minimize either the clustering entropy or the clustering Hamming distance, defined in subsections 4.2.1 and 4.2.2 above. A trial step consists of moving a randomly selected sequence to a different randomly selected cluster, and accepting the move only if the objective function is reduced.

Algorithm 5, Monte Carlo based clustering, implements this approach, with several modifications intended to improve its runtime and the quality of its outputs.

The algorithm takes an existing clustering as its starting point. In our experiments, we use clusterings generated by the CliqueSNV tool (63) from the datasets described in subsection 4.3.1. We supply such clustering as an input to the algorithm in order to generate a new clustering with reduced clustering entropy $H()$ or reduced Hamming distance $HD()$. Additional inputs to the algorithm are two parameters I and K, where I defines the number of consecutive rejected trials

Algorithm 5 Monte Carlo based clustering

Input: Initial clustering (default from CliqueSNV), Number of rejected moves I (default $I = 800$), Relative difference K (default $K = 0.00001$);

Output: Clustering with reduced entropy or Hamming distance.

1: Compute nucleotide counts for each column in each cluster.
2: Compute entropy (or Hamming distance) for each cluster and total H.
3: Initialize number of rejected moves $T = 0$.
4: **while** $T \leq I$ **do**
5: Pick a random sequence s.
6: Move s from its cluster A to a randomly selected cluster B ($B \neq A$).
7: Update nucleotide counts for A and B.
8: Compute Δ, the change in overall entropy (or Hamming distance).
9: **if** $\Delta/H \geq K$ **then**
10: Accept the move, update $H = H - \Delta$, reset $T = 0$.
11: **else**
12: Reject the move, increment T, return s to A.
13: **end if**
14: **end while**

before stopping, and K defines the relative objective function reduction threshold for accepting a move.

To achieve the goal of finding a new clustering with reduced entropy or Hamming distance as the objective, the algorithm applies Monte Carlo optimization by repeatedly trying to move a randomly selected sequence from its current cluster to another randomly selected cluster; any such move is accepted only if the relative improvement to the objective function is higher than the threshold value K.

In the initialization phase, lines 1–3 of the algorithm, it starts by computing nucleotide counts for each column in each cluster, which are then used to compute the values of the entropy and the Hamming distance for each cluster, as well as the overall clustering entropy or Hamming distance, H.

A Monte Carlo optimization procedure is then implemented in the loop in lines 4–17 as follows:

- Line 5: Randomly pick a sequence s.
- Line 6: Remove sequence s from its cluster A and place into another randomly selected cluster B.
- Line 7: Recalculate entropy (resp., Hamming distance) for both clusters A and B.
- Line 8: Compute the entropy (resp., Hamming distance) reduction between the new and the previous clusterings.
- Lines 9–16: If entropy (resp., Hamming distance) has reduced by at least the relative difference parameter K, keep the new clustering; otherwise, revert to the previous clustering. By default, we set $K = 0.00001$.
- Repeat lines 4–17 until the clustering converges. Specifically, the algorithm will stop if we do not accept any moves for sufficiently long time, i.e., if the clustering does not change for I consecutive iterations. By default, we set $I = 800$.

4.2.4 Tag Selection

To improve runtime, we apply a preprocessing tag selection step that allows us to represent sequences by a smaller subset of sites. Preferring tags with highest variability, the procedure chooses the n sites with highest entropy, where n is some predefined value. Then clustering proceeds with each sequence now of length n corresponding to the selected sites.

4.3 Settings for Validation of Clustering Methods

We validate entropy based clustering by estimating improvement over an existing clustering technique. To that end, we apply this Monte Carlo based algorithm to the clustering obtained by the CliqueSNV tool (63).

4.3.1 Datasets

For validation, we use two of the datasets of sequences that were also used by Melnyk et al in (64). For both datasets, an initial clustering was obtained by the CliqueSNV-based method.

D1: This dataset includes all sequences submitted to the global GISAID viral database (61) from the beginning of the pandemic up until the beginning of March 2020. It consists of $3{,}688$ aligned sequences, all sequences $29{,}891$ nucleotides long. CliqueSNV produced an initial clustering of this dataset consisting of 28 clusters.

D2: This dataset includes all sequences submitted to the UK-based EMBL-EBI database from the end of January 2020 to the end of December 2020 (65). It consists of $148{,}000$ aligned sequences, all sequences $29{,}903$ nucleotides long. CliqueSNV produced an initial clustering of this dataset consisting of 15 clusters.

4.3.2 Tag Selection Effects on Runtime

To measure the effects of tag selection on runtime, the proposed method was run on the D1 dataset of $3{,}688$ sequences, using the initial clustering generated by CliqueSNV as a starting point. The tag selection procedure was employed to produce four subdatasets consisting of the same sequences,

but of reduced lengths of 100, 1000, 3000, and 5100 tags. We expect that the Monte Carlo method, when applied to a dataset consisting of shorter sequences, will be able to take more trial steps in the same amount of time, and thus reduce its clustering entropy quicker. The total number of SNPs in the input data was exactly 5100; all other positions did not mutate. Thus, this largest number of tags contains information equivalent to the full length sequences for the purposes of clustering.

The program was run on each length of sequences. Every hour, the current clusterings of each run were evaluated by their entropy on the full-length sequences. These hourly entropy values are shown on an entropy-over-time graph, Figure 4.1, which compares the speed of entropy reduction for different sequence lengths.

4.3.3 Discerning Signal from Noise with Monte Carlo Based Clustering Optimization

We estimate the amount of meaningful information extracted by the clusterings obtained by our entropy minimization method. To distinguish between sample-specific noise and meaningfully extracted information, we run our method on a perturbed version of the input with the same starting entropy. For this experiment we use the D1 dataset with $3{,}688$ sequences, alongside the initial clustering from CliqueSNV of 28 clusters.

The permutation procedure is as follows. Within each cluster, every site is shuffled into a random permutation. Importantly, by respecting clusters during permutation, the initial nucleotide frequencies within each site in each cluster stay the same. Thus, the permuted input has the same starting entropy as the original input. What changed is the haplotypes being clustered.

We run the program on both of these inputs for exactly $100{,}000$ Monte Carlo trials each, accepting all moves that reduce entropy. We compare the resulting entropy reductions between the

two runs. Any entropy reduction present in the permuted data is sample-specific noise extracted by our method, while the difference in resulting entropies between the original and permuted inputs corresponds to the amount of meaningful information extracted by our method.

4.3.4 Stability of Optimized Clustering

https://www.overleaf.com/project/6223ed0d808eaece9a5d8920 Now we evaluate the robustness of our method against slight permutations of the input data as well as changes in random seed. Rather than completely shuffling each site as in 4.3.3, we only shuffle a small percentage p of nucleotides at each site. We still respect clusters when permuting the data, to ensure that nucleotide frequencies in each site in each cluster remain unchanged.

We chose two values of p to create slightly permuted data sets for validation, $p = 1\%$ and $p = 5\%$, to be compared with the original data with 0% permutation. For each of the three datasets (two permuted and one original), we run our method three times, on two different objectives: first minimizing entropy, and second minimizing Hamming distance between sequences and their cluster consensus. As a result, for each degree of permutation and for each Monte Carlo objective, we obtain three minimum entropy clusterings.

The Rand index, measuring the degree of agreement between two clusterings, is measured between the initial clustering and all resulting clusterings, to get a sense of how far away the resulting clusterings have moved from the initial one under varying degrees of permutation. Further, we also measure the Rand index between the resulting clusterings, to determine whether the proposed method converges to similar clusterings across multiple runs.

4.4 Validation Results

We ran the proposed method on the cluster hardware consisting of 128 cores IntelXeonCPU E7-4850 v4 CPU @ 2.10GHz, with 3 TB of RAM, running Ubuntu 16.04.7 LTS.

We ran our entropy based Monte Carlo clustering algorithm on different selections of tags, selected based on highest entropy contribution. Figure 4.1 shows the results for different numbers of tags. The maximum number of tags, 5100, corresponds to all SNP positions present in the input data; all other sites were homogeneous. Since homogeneous sites have zero entropy, they can be ignored from entropy calculations. Therefore, the yellow line representing 5100 tags corresponds to using all sites.

When the number of tags is 1000, using tag selection yields better results of reducing entropy for some time durations (up to 3 hours). But in general, the effect of using tags on reducing entropy is not significant.

Compared to previous work (64), we were able to reduce the runtime by 95.83%. For accomplishing this, we initially stored the counts of each nucleotide across a given tag in a cluster of sequences.

After further improving our entropy based Monte Carlo clustering algorithm and implementing parallel computing, we made it scalable for running on large data sets. This version performs 12K–13K iterations on average per hour, which is approximately **10** times faster than all our previous implementations.

Interestingly, after running the algorithm for **83K** iterations on the D2 dataset (which contains $148{,}000$ aligned sequences each $29{,}903$ nucleotides long) originally distributed across 15 clusters,

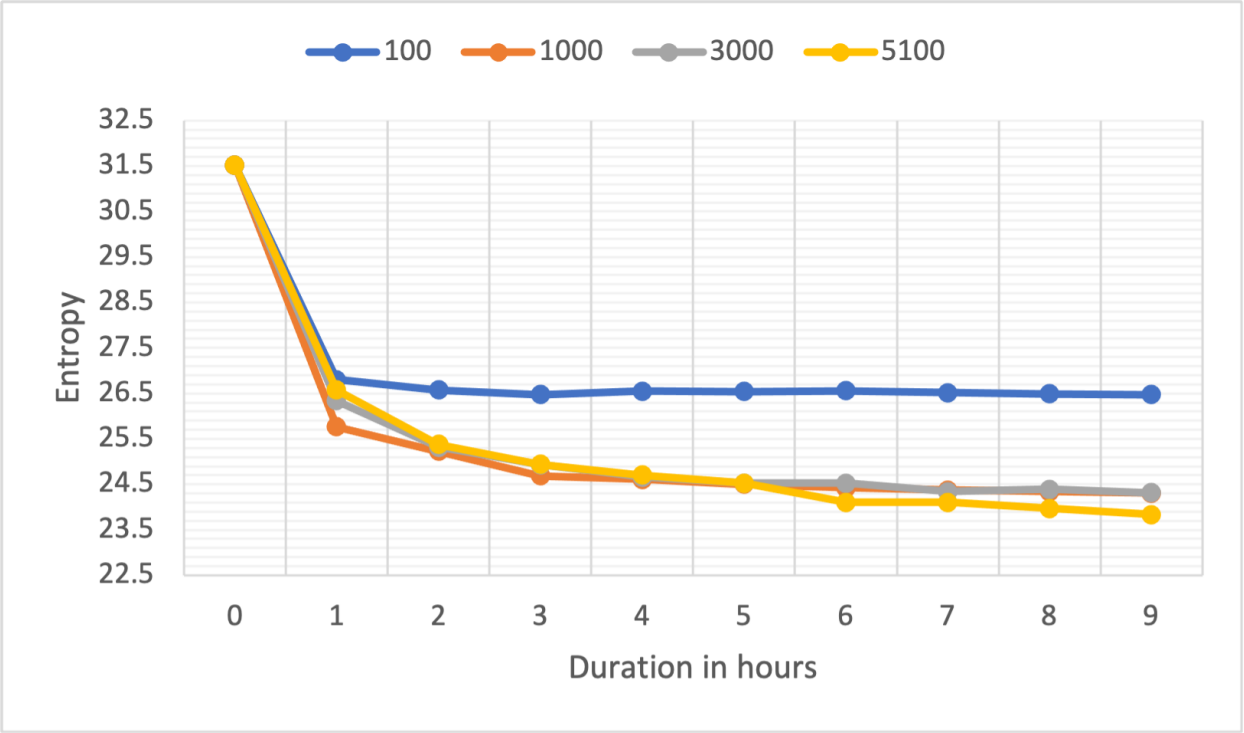

Figure 4.1 Entropy reduction over runtime for different numbers of selected tags. After 1 hour, the 1000 tags representation was able to reach the lowest entropy.

we came to the conclusion that in instances where sequence data is clustered into a smaller number of clusters there are some clusters where sequences are more dense than in others. For instance, after this run, in the output clustering the biggest cluster consisted of $34{,}995$ sequences, while the smallest had only $3{,}571$ sequences.

Therefore, moving one sequence at a time is not always beneficial for overall entropy reduction. We tried to tackle this problem by updating our move acceptance threshold to accept even smaller positive changes to entropy, from our previous relative difference threshold of $K = 10^{-5}$ to $K = 10^{-7}$. Although this change made our algorithm run for many more iterations before stopping (recall that the algorithm stops when it reaches the stopping threshold of $I = 800$ unsuccessful

moves), the reduction to the overall entropy was still very small.

We believe that moving similar sequences together within clusters rather than moving just one could be a possible way of overcoming the problem we face here.

4.4.1 Picking signal over noise in clustering

By minimizing entropy on the permuted data, we find that the method reduces entropy to 29.6, while on the original, unshuffled data the method reaches a much lower entropy of 24 (see Table 4.1). This difference in entropies, $29.6 - 24 = 5.6$, of resulting clusterings accounts for the amount of meaningful information, which is not noise, that our method was able to extract from the real data.

Entropy MC reduced					Hamming distance MC reduced				
Initial	Original		Permuted		Initial	Original		Permuted	
	Avg	Min	Avg	Min		Avg	Min	Avg	Min
31.524	24.77	24.7	29.62	28.65	1008.41	373.14	369.61	770.39	689.97

Table 4.1 Results after running Monte Carlo for 1000 tags selected in decreasing order of entropy across 100 datasets obtained by applying the random permutation procedure described in section 4.3.3 to the D1 dataset 100 times. *Average Iterations in Monte Carlo*: 53804.39. *Average successful moves*: 615.31.

4.4.2 Stability of Monte Carlo Output

Table 4.2 shows the results of stability validation, in which we compare clustering similarity for varying degrees of permutation of the input data (see subsection 4.3.4).

The first column compares resulting clusterings to the initial clustering. Without any permutations, the resultant clustering moves significantly further away from the initial one, giving a Rand index of 0.93. As the permutation degree increases, we observed that the clusterings produced by the Monte Carlo algorithm do not move as far away from the initial clustering; in other words,

% permutation	Cluster similarity(Rand index)			
	Entropy		Hamming	
	With original	With runs	With original	With runs
0	0.936476	0.970195	0.936114	0.970135
1	0.978898	0.979435	0.936126	0.970374
5	0.980458	0.980688	0.936180	0.970616

Table 4.2 Clustering similarity (Rand index) across three choices of degree of permutation. The proposed method was run three times for each permuted instance, each run consisting of 100.000 Monte Carlo trials. Reported are average Rand index similarity of the resulting clusterings to the initial clustering, as well as between resulting clusterings.

even after Monte Carlo was applied, the resulting clusterings had high degree of agreement with the initial clustering.

The second column in Table 4.2 gives the average Rand index between multiple runs of Monte Carlo for a given permutation. We see that for all degrees of permutation the method stably converges towards similar clusterings, with Rand index scores of 0.97–0.98. The same trends can be observed when using Hamming distance to cluster consensus as the objective.

4.4.3 Results for Large Datasets

Running our entropy based Monte Carlo method on the large dataset D2, which consist of 143,000 aligned sequences, we get the initial entropy for our initial clustering from CliqueSNV of 80.2750171. After 82,786 iterations, the final entropy for the resultant clustering was 79.509444, and the total runtime for this was 9 hours 15 minutes.

4.5 Conclusion

We have developed a scalable method to find minimum-entropy clusterings of datasets viral genomic sequences. The method is scalable to hundreds of thousands of sequences, and is made even faster without significant loss of accuracy by picking a subset of tags with maximum entropy

to represent the sequences. We estimate the amount of meaningful information extracted by the method. We also show that our method converges toward similar minimum-entropy clusterings across multiple runs, demonstrating its stability.

For future directions, we believe the Monte Carlo entropy minimization approach can be improved by using simulated annealing, whose tolerance of suboptimal moves can allow us to escape local minima. We are also going to add Monte Carlo entropy minimization method to CliqueSNV's clustering of intrahost populations.

CHAPTER 5

Genetic Algorithm with Evolutionary Jumps

Hafsa Farooq, Daniel Novikov, Akshay Juyal, Alexander Zelikovsky

ISBRA 2023

ABSTRACT

It has recently been noticed that dense subgraphs of SARS-CoV-2 epistatic networks correspond to future unobserved variants of concern. This phenomenon can be interpreted as multiple correlated mutations occurring in a rapid succession, resulting in a new variant relatively distant from the current population. We refer to this phenomenon as an *evolutionary jump* and propose to use it for enhancing genetic algorithm. Evolutionary jumps were implemented using C-SNV algorithm which find cliques in the epistatic network. We have applied the genetic algorithm enhanced with evolutionary jumps (GA+EJ) to the 0-1 Knapsack Problem, and found that evolutionary jumps allow the genetic algorithm to escape local minima and find solutions closer to the optimum.

5.1 Introduction

The unprecedented density of SARS-CoV-2 sequencing data allows to follow the viral evolution much closer than in pre-pandemic time (66; 67). Epistatic networks of SARS-CoV-2 constructed on GISAID data, contain densely linked subgraphs of mutations which correspond to known variants of concern, and also allow us to predict and early detection of future variants (68). The network has non-additive phenotypic effects and their vertices are single nucleotide polymorphism(SNPs)

and its edges are correlated pair of mutations, where dense subgraphs have high density of connectivity among their vertices. It is remarkable that altered phenotype of variants of concern (VOC) do not appear gradually since one cannot observe intermediate variants containing substantial subsets of mutations defining the VOC. Such phenomenon was previously observed in Paleontology and referred as punctuated equilibrium. This phenomenon can be interpreted as if multiple correlated mutations occur in a rapid succession, resulting in appearing of a novel variant which is relatively distant from the closest representative of the current population. In this paper, we propose to apply this evolutionary mechanism (referred as evolutionary jumps) to enhance genetic algorithm (GA).

Genetic algorithm mimics the natural evolution and select the fittest individuals which further reproduce using genetic operators. The drawback of genetic algorithm as well as other local optimization methods that they can stuck in local minima. We propose to rectify this drawback by applying evolutionary jumps when no significant improvement is achieved for several generations by GA. Instead of dense subgraphs in the epistatic network, our approach is to find cliques that's why we use C-SNV algorithm (C-SNV) (69). It identifies cliques (maximal complete subgraphs) and use them to assemble viral variants present in the sequencing data. When GA stuck for a number of generations, we run C-SNV on all individual solutions constructed so far to identify new individuals with all mutations corresponding to identified cliques.

In order to evaluate the quality of the GA and compare with the proposed enhancements, we applied it to hard instances of 0-1 Knapsack Problem recently proposed in (1). Since the simple GA solutions can be infeasible or extended, we first enhance GA with repairing and packing (GA-RP) and then further introduce evolutionary jumps (GA-EJ). Our experiments show that GA-RP

significantly outperform the simple GA on all instances while GA-EJ further improves GA-RP for harder instances, i.e., for cases where finding optimal solution requires large runtime and GA-RP stuck significantly far away from the optimum.

Section 5.2 describes a genetic algorithm and proposed enhancements using evolutionary jumps. Section 5.3 applies GA to the 0-1 Knapsack Problem and gives details of enhancing GA with repairing and packing as well as evolutionary jumps. Section 5.4 describes hard problem instances and compares results achieved by GA, GA with repairing and packing (GA-RP), and GA with repairing and packing & evolutionary jumps (GA-EJ).

5.2 Genetic Algorithm with Evolutionary Jumps

5.2.1 Simple Genetic Algorithm

Genetic algorithm is a metaheuristic inspired by the process of natural evolution relying on biologically inspired operators such as mutation, crossover and selection (70). GA is a heuristic search-based evolutionary algorithm developed by John Holland in 70's. Holland developed an electronic organism named chromosomes consisting of binary encoded strings (71) or unit entity known as gene. Those randomly generated binary encoded strings based chromosomes are also called individual solutions, and these potential solutions altogether are the initial population.

After the creation of initial population, evolution begins. The fitness function evaluation is performed for each individual of the population. The fitness score represents the ability of the individual to compete for mating and its quality in the solution. The individuals with higher fitness values are chosen for mating pool, called parents. After selecting the best fitted individuals from

the population, selected individuals perform reproduction.

Crossover is a process of combining genetic material of parents by inheriting their traits in offspring. Crossover randomly chooses a point or locus in the individuals and exchange before and after sub-strings of individuals to create offspring. For example for single point crossover, consider the two individuals and crossover point at the 5th position of the individual, shown in Fig.5.1(a). The offspring 1 has first five genes from individual 1 and next five genes from individual

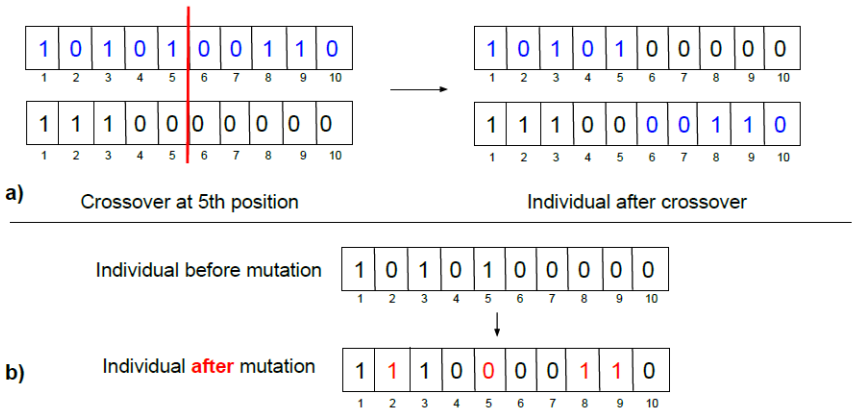

Figure 5.1 a) Two individuals are performing crossover. Red line represents the point of crossover. b) Mutation is performed on 2nd, 5th, 8th and 9th gene of the individual.

2. Similarly, offspring 2 has left side of genes from individual 2 and right side of genes from individual 1.

After crossover, individuals undergo mutation. The mutation operator changes one or more genes randomly. It changes the gene value from 1 to 0 or vice versa, shown in Fig.5.1(b). The type of crossover, mutation and its probability can be defined and depends on the problem under experimentation. Both the parents and the offspring now comprise the next generation of the genetic algorithm, where the process repeats. GA terminates after the fitness is not improved for predefined number of generations or number of generations exceeds a given number.

5.2.2 Punctuated Equilibrium and Epistatic Network of SARS-CoV-2

The genomic evolution of SARS-CoV-2 shows that the rate of mutations is not constant – the gradual relatively slow evolution is replaced with brief and fast bursts resulting in emerging of new viral variants with altered phenotypes including increased transmissibility. Such new variants (e.g., Alpha- and Omicron-variants) are referred as Variants of Concern (VOCs) or Variants of Interest (VOIs)). This phenomenon has been labeled as punctuated equilibrium. It has been shown recently that punctuated equilibrium events for SARS-CoV-2 which we refer to as *evolutionary jumps* can be predicted from its epistatic network (68).

Following (68), we define an epistatic network as a graph with vertices corresponding to mutated positions, e.g. single nucleotide polymorphisms (SNPs) or single amino-acid variations (SAVs). Two vertices are connected if the mutations in the the corresponding positions i and j are significantly more frequently observed in the same haplotype than they are expected if the mutations would be independent.

Formally, let 0 and 1 denote the reference and mutated alleles in positions i and j, respectively. Assuming that positions are biallelic, each possible haplotype h in positions i and j belong to the set $\{00, 01, 10, 11\}$. Let O_h (resp. $E(h)$) be the observed (resp. expected) number of haplotypes h in the sequencing data. It has been proved in (68) that if haplotype 11 is not viable (does not produce descendants), then

$$E_{00} * E_{11} \leq E_{01} * E_{10} \tag{5.1}$$

In the epistatic network we connect to vertices if the the corresponding haplotype 11 is viable, i.e., when the $O_{00} * O_{11}$ is significantly larger than $O_{01} * O_{10}$.

Recently, it has been shown that evolutionary jumps in SARS-CoV-2 evolution correspond to dense subgraphs of the epistatic network (68). Formally, the densest subgraph, i.e. the one with the maximum ratio of edges over vertices, frequently consists of mutations that differentiate an emerging viral variant from the reference. Therefore, rather than performing a random combination of mutations, evolutionary jumps include multiple mutually linked mutations.

5.2.3 Enhancement of GA with Evolutionary Jumps

The genetic algorithm uses selection pressure to push future generations closer to the optimum, with limited differences between consecutive generations. It can be observed that standard genetic algorithm is prone to getting stuck in local minima, because crossovers and mutations alone are not enough to escape them. Therefore, we propose to enhance the genetic algorithm with evolutionary jumps. Evolutionary jumps involve the appearance of new individual solutions in the population, which include genes or mutations that are observed to be correlated in previous generations.

Our procedure decides when to perform evolutionary jumps by monitoring the fitness of the best solution across generations. If the number of generations without fitness improvement exceeds a predefined threshold, then the result of an evolutionary jumps are added to the next generation. Instead of dense subgraphs in the epistatic network, our approach is to find cliques that's why we use C-SNV (69). It identifies cliques (maximal complete subgraphs) and use them to assemble viral variants present in the sequencing data.

5.3 Application of Genetic Algorithm with Evolutionary Jumps to the 0-1 Knapsack Problem

5.3.1 The 0-1 Knapsack Problem

Given a set of items with weights and profits and a maximum capacity for the knapsack, the 0-1 Knapsack Problem asks for a subset of items that maximizes total profit without exceeding the knapsack capacity. A solution to the 0-1 Knapsack Problem is a vector with binary coordinates corresponding to items. The coordinate equals 1 when the corresponding item is selected, and 0 otherwise. We use the 0-1 Knapsack Problem as a benchmark to evaluate the performance of our proposed improvement to the genetic algorithm.

5.3.2 Implementation of Genetic Algorithm

As a base implementation of the genetic algorithm, we employ the PyGAD genetic algorithm Python library. (72) An initial population is created by randomly generating solutions that fill the knapsack up to capacity. For a fitness function, we use the sum of profits of the items included in the knapsack, unless the sum of their weights exceeds the knapsack capacity, in which case the solution receives the minimum fitness of -1.

In each generation, a tournament selection procedure identifies high-fitting solutions to be parents for the next generation. The chosen parents are grouped into pairs, and each pair is crossed-over and randomly mutated to produce a pair of offspring for the next generation, as shown in Fig.5.1. This procedure repeats for the given number of generations.

5.3.3 Repairing and Packing

Throughout the execution of the standard genetic algorithm on the 0-1 Knapsack Problem, solutions frequently either exceed the knapsack capacity or are under-filled, meaning there are still items remaining which can fit in the solution without bringing it over capacity. Rather than discarding these solutions, we propose a procedure for repairing solutions that are over-filled, and for filling solutions that are under-filled. We call this procedure repairing and packing, illustrated in Fig. 5.2.

To repair and pack a given solution, the procedure begins by sorting the items of the problem instance such that all items which are included in the solution (1s) come first, and all non-included items ($0s$) come afterwards. Then, it randomly shuffles the included items and the non-included items separately. Ordering the array in this manner allows us to consider the items in a random order subject to constraint that all items included in the solution preceed those that are not included.

Our procedure repairs and packs solutions in a single pass through this sorted array of items. The procedure starts a new, empty solution, and begins iterating the sorted items array. While iterating through the included items region, it tries to add each item to the solution, and does so as long as the solution remains under capacity. If the item cannot fit (i.e., after its addition the total capacity exceeds the upper limit), we change it to 0, removing it from the knapsack. This ensures over-filled solutions will be brought back down to capacity. When we reach the non-included items region of the sorted array, we try to add those items to the solution as well, and do so as long as they fit. By applying the repairing and packing procedure, we can guarantee that solutions are not over-capacity, and that there are no items remaining which could still fit in the knapsack.

We apply this procedure to each solution throughout the execution of the genetic algorithm. The initial population, next generation offsprings, and evolutionary jumps solutions are all repaired-and-packed according to this procedure, ensuring each solution is both feasible and maximally filled.

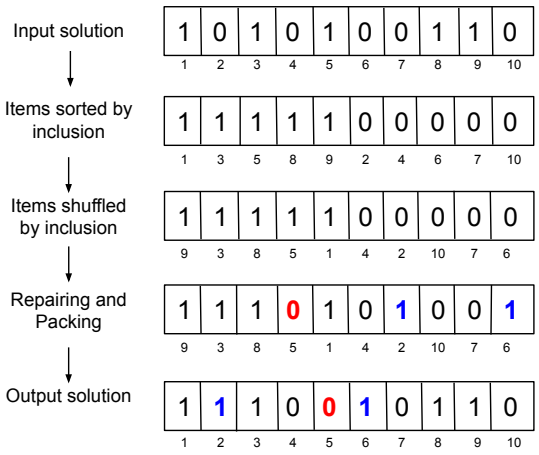

Figure 5.2 Repairing and packing procedure on an example instance of 10 items

5.3.4 Evolutionary Jumps Implementation

Evolutionary Jumps involve the introduction of new individuals to the genetic algorithm population. These new solutions include items that are observed to be correlated in the past evolutionary history. Our procedure decides when to perform evolutionary jumps by monitoring the fitness of the best solution across generations. Each time we observe 10 consecutive generations without improvement to the best fitting solution, the evolutionary jump procedure is triggered.

To facilitate the evolutionary jumps, C-SNV is employed to find correlated pairs of mutations. C-SNV is a tool which finds characteristic haplotypes to describe a set of input sequences. Internally, the tool implements a procedure for identifying pairs of mutations which are correlated by

high co-occurence in the input data.

Treating each solution vector as a sequence, we pass the entire evolutionary history, i.e, all solutions from all generations so far, as an input to C-SNV, encoding 0 as A and 1 as C, using a Fasta file format.

After finding correlated pairs of positions, C-SNV constructs an epistatic network over sequence sites with edges given by correlations, and finds cliques in this graph to reconstruct characteristic haplotypes for the evolutionary history. Each clique relates a set of items which have frequent pairwise co-occurrence. A haplotype is created from each clique, containing mutations for the sites that appeared in the clique. C-SNV typically returns $2 - 10$ haplotypes. For each haplotype, we create a new knapsack solution to add to the population.

The procedure for creating knapsack solutions from C-SNV haplotypes begins with a solution vector containing 1s where the haplotype had a 1. We observe 10 consecutive generations without improvement towards the best fitting solution, then evolutionary jump procedure triggers. Then, to each solution, we add the items included in the current best solution. In the result, the newly created solutions are guaranteed to take the items where the C-SNV haplotype contained a 1 for that position, and then additionally, they take the items included in the best solution observed so far, so long as those don't bring it over capacity. The newly created solutions each represent an evolutionary jump, and these solutions are added back to the genetic algorithm population, prior to starting the next generation. The new solutions are added by replacing the currently worst-fitting solutions with the new ones created by the jump procedure.

5.4 Results

5.4.1 Instances of the 0-1 Knapsack Problem

For validation of genetic algorithms on the 0-1 Knapsack Problem, we chose instances recently generated in (1). It was proposed a new class of hard problem and shown that they are hard to solve to optimality. Many hard problem instances were not even possible to solve on a supercomputer using hundreds of CPU-hours. Out of 3240 instances we have selected the first ten instances which were solved to optimality.

The selected problem instances are listed in the Table 5.1 together with the nomenclature from (1). Different letters in the names of the instances represents;

- n: Number of items of a problem instance
- c: Capacity of the knapsack
- g: Number of groups of items of a problem instance
- f: Approximate fraction of items in the last group
- ε: Noise parameter
- s: Upper bound for profits and weights of items in the last group

The runtime for a solver to reach an optimal solution is given in the column Runtime. Note that problem instances 1-5 are harder since they require significantly more runtime to reach optimality then problem instances 6-10.

Table 5.1 0-1 Knapsack Problem Instances (1)

ID	Problem Instance	Optimal Fitness	Runtime(sec)
1	n_1000_c_10000000000_g_10_f_0.1_eps_0.0001_s_100	9999946233	2943
2	n_1000_c_10000000000_g_10_f_0.1_eps_0.0001_s_300	9999964987	6474
3	n_1000_c_10000000000_g_10_f_0.1_eps_0.01_s_100	9999229281	555
4	n_1000_c_10000000000_g_10_f_0.1_eps_0.01_s_200	9999239905	742
5	n_1000_c_10000000000_g_10_f_0.1_eps_0.01_s_300	9999251796	896
6	n_1000_c_10000000000_g_10_f_0.1_eps_0.1_s_100	9996100344	17
7	n_1000_c_10000000000_g_10_f_0.1_eps_0.1_s_200	9996105266	18
8	n_1000_c_10000000000_g_10_f_0.1_eps_0.1_s_300	9996111502	26
9	n_1000_c_10000000000_g_10_f_0.1_eps_0_s_100	9980488131	74
10	n_1000_c_10000000000_g_10_f_0.1_eps_0_s_200	9980507700	96

5.4.2 Parameter Tuning

We tuned the parameters of the genetic algorithm and the evolutionary jump procedure to optimize the output profit. In genetic algorithm, these parameters include the size of the population, number of generations, mutation probability, and so on. For evolutionary jumps, the parameters include the number of non-improving generations to wait before jumping, how sensitive C-SNV should be to determine links, and more. To find the optimal parameters for our problem instances, we applied multiple parameter configurations to each problem instance 10 times, observing the average performance under each configuration.

The finalized values of parameters are shown in the Table 5.2. Some parameters have several values, e.g., we report GA for 500, 1000, and 1500 generations.

Table 5.2 Parameters Table

GA Parameter	Value		C-SNV Parameter	Value
Population Size	1000		Jump Threshold	10
Num of Gen	500, 1000, 1500		Min.Gen.Wait	20
Num of Parents	500		Jump Type	Global Best
Crossover type	Single Point		C-SNV Timeout	120sec
Parent Type	Tournament		Threshold.Freq	0.01
K-Tournament	3(default),25		Threshold.Freq+	0.01
Mutation Type	Inversion		Memory	20GB
Mutation Pr	0.02		Edge Limit	1000

5.4.3 Performance Comparison of GA, GA+RP, and GA+EJ

The results of our experiments are shown in Tables 5.3-5.5. "Min.Error" is the minimum difference between optimal fitness and the best fitness over 10 runs. "Avg.Error" is the average difference between optimal fitness and the best fitness over 10 runs. "Runtime" is shown in the minutes and its the average runtime over 10 runs.

We ran the GA and GA+RP on instances for K=3,K=25 and number of generations G= 500, 1000,1500 (see Tables 5.3-5.4) and GA+EJ just for K=25 and number of generations G=500 (see Table 5.5).

It is easy to see that GA+RP significantly outperform simple GA on all instances and all configurations. Also GA+RP with K=25 outperforms GA+RP with K=3 for all instances. Therefore we decided to run GA+EJ just for K=25. For the harder instances 1,3,4,5 GA+EJ significantly outperform GA-RP even for G=1500 the novel evolutionary enhancement of GA.

Table 5.3 Simple Genetic Algorithm Results

	Results	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
GA	**Min.Error**	4.7e5	1.4e6	9695	1.9e4	3.0e4	5914	9908	1.5e4	1.7e4	3.5e4
K=3	**Avg.Error**	1.5e6	2.1e6	9798	1.9e4	3.0e4	4.3e6	4.6e6	4.3e6	1.8e4	3.6e4
G=500	**Runtime**	28	29	39	37	33	58	62	63	28	27
GA	**Min.Error**	4.7e5	4.9e5	9620	1.9e4	2.9e4	5433	1.0e4	1.5e4	1.8e4	3.5e4
K=3	**Avg.Error**	1.6e6	1.8e6	9721	1.9e4	3.0e4	2.7e6	5.4e6	4.3e6	1.8e4	3.6e4
G=1K	**Runtime**	43	43	56	60	60	94	109	93	43	43
GA	**Min.Error**	4.7e5	4.9e5	9599	1.9e4	3.0e4	5953	1.0e4	1.4e4	1.7e4	3.5e4
K=3	**Avg.Error**	1.1e6	6.9e5	9726	1.9e4	3.0e4	4.3e6	4.3e6	4.3e6	1.8e4	36174
G=1.5K	**Runtime**	78	80	122	106	108	186	167	175	80	79
GA	**Min.Error**	4.7e5	4.8e5	9285	1.8e4	2.9e4	5498	3.9e6	1.3e4	1.7e4	3.4e4
K=25	**Avg.Error**	4.7e5	4.9e5	9472	1.9e4	2.9e4	3.5e6	5.8e6	3.1e6	1.7e4	3.5e4
G=500	**Runtime**	80	59	74	74	75	105	106	101	57	54
GA	**Min.Error**	4.7e5	4.8e5	9254	1.8e4	2.9e4	5286	9362	1.4e4	1.7e4	3.3e4
K=25	**Avg.Error**	4.7e5	4.8e5	9414	1.9e4	2.9e4	3.1e6	3.5e6	1.9e6	1.7e4	3.5e4
G=1K	**Runtime**	81	80	122	129	133	181	177	167	68	91
GA	**Min.Error**	4.7e5	4.8e5	9283	1.8e4	2.8e4	5396	9655	1.3e4	1.7e4	3.4e4
K=25	**Avg.Error**	4.7e5	4.8e5	9428	1.8e4	2.9e4	3.9e6	3.1e6	3.1e6	1.7e4	35017
G=1.5K	**Runtime**	159	159	194	200	214	286	263	275	133	126

Table 5.4 Results for Genetic Algorithm with Repairing and Packing

	Results	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
GA-RP	**Min.Error**	1102	3298	2662	4242	7433	71	167	235	4406	1.1e4
K=3	**Avg.Error**	1157	3481	3170	5958	8577	120	275	400	6090	1.2e4
G=500	**Runtime**	98	98	102	99	99	136	149	141	118	106
GA-RP	**Min.Error**	917	3181	2314	4063	6990	87	174	195	4556	8891
K=3	**Avg.Error**	1084	3422	3026	5289	8531	115	255	371	5324	1.1e4
G=1K	**Runtime**	203	197	189	188	201	244	241	243	217	218
GA-RP	**Min.Error**	915	3166	2353	4569	6777	67	163	184	3502	6445
K=3	**Avg.Error**	1067	3395	2763	5294	8166	95	226	297	4778	9392
G=1.5K	**Runtime**	281	282	294	292	291	397	390	400	307	334
GA-RP	**Min.Error**	874	2540	2742	4827	8388	14	36	48	306	573
K=25	**Avg.Error**	1046	2901	3343	6160	9528	34	93	105	450	903
G=500	**Runtime**	99	98	109	123	104	164	171	158	139	143
GA-RP	**Min.Error**	959	2335	2566	4586	5406	14	31	39	214	525
K=25	**Avg.Error**	1037	2897	3120	5830	8867	26	55	75	361	956
G=1K	**Runtime**	188	184	198	203	207	278	275	274	261	255
GA-RP	**Min.Error**	780	1451	2365	5010	7646	17	40	39	265	644
K=25	**Avg.Error**	955	2642	3197	5670	9181	22	78	61	469	849
G=1.5K	**Runtime**	283	299	316	318	341	472	473	462	408	407

Table 5.5 Results for Genetic Algorithm Evolutionary Jumps (GA+EJ)

	Results	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
GA-EJ	**Min.Error**	541	1950	1557	3972	6000	27	50	113	350	595
K=25	**Avg.Error**	751	2425	2761	5329	8596	47	83	178	686	951
G=500	**Runtime**	2234	1949	2081	1883	2171	3056	2829	3153	2441	2603

5.5 Conclusion

In this paper, we upgraded genetic algorithm with evolutionary jumps to mimic punctuated equilibrium seen in SARS-CoV-2 sequencing data. The enhanced genetic algorithm was tested on challenging cases of the 0-1 Knapsack Problem. Initially, the Genetic algorithm for the Knapsack Problem was refined by applying repairing and packing methods. Subsequently, we boosted GA

with evolutionary jumps integrated through CliqueSNV. Our tests demonstrated that evolutionary jumps notably enhance GA for extremely challenging instances of the 0-1 Knapsack Problem.

CHAPTER 6

Summary of Contributions

The unprecedented size and density of the SARS-CoV-2 Genomic data has driven the development of novel bioinformatics algorithms seeking to extract its insights. To facilitate this extraction, this dissertation details four unsupervised approaches that are tailored to these unique data characteristics.

The first contribution, SPHERE, is a scalable phylogeny reconstruction algorithm designed to adapt to the high density of genomic data. It utilizes novel computational techniques to efficiently map the evolutionary relationships within the SARS-CoV-2 virus, enabling a more detailed and accurate understanding of its genetic lineage.

The second contribution introduces (ε, τ)-MSN, a method for constructing genetic relatedness networks. This approach integrates all possible minimum spanning trees with additional edges, uncovering clusters of similar sequences to facilitate the identification of closely related viral strains and their transmission pathways.

The third contribution presents an innovative unsupervised learning strategy aimed at minimizing cluster entropy among genomic sequences. This method optimizes the grouping of sequences to reflect their natural organization, thereby enhancing the ability to discern patterns and dynamics of viral evolution.

Lastly, the fourth significant contribution is the introduction of evolutionary jumps within genetic algorithms. This groundbreaking approach simulates the punctuated equilibrium phenomena observed in SARS-CoV-2 sequencing data, offering profound insights into the dynamics of viral

evolution. By optimizing for very hard instances of the 0-1 Knapsack Problem, this method showcases a significant advancement in genetic algorithms, enriching our understanding of complex patterns of genetic changes over time.

Collectively, these contributions significantly enhance the toolkit available for bioinformatics research, offering sophisticated analyses of viral genomes and contributing to the broader field of computational biology with innovative solutions to complex biological problems.

CHAPTER 7

List of Publications and Presentations

7.1 List of Publications

1. Novikov, D, et al. “Scalable Reconstruction of SARS-COV-2 Phylogeny with Recurrent Mutations.” Journal of Computational Biology, vol. 28, no. 11, 2021, pp. 1130–1141.

2. Knyazev, S, Novikov, D, et al. “A Novel Network Representation of SARS-COV-2 Sequencing Data.” Bioinformatics Research and Applications, ISBRA 2021

3. Juyal, A., Hosseini, R., Novikov, D., et. al. (2022). “Entropy Based Clustering of Viral Sequences.” Bioinformatics Research and Applications. ISBRA 2022.

4. Juyal, A., Hosseini, R., Novikov, D., et. al. (2023). ”Reconstruction of Viral Variants via Monte Carlo Clustering” Journal of Computational Biology, 2023

5. Farooq, H., Novikov, D., et. al. (2023) ”Genetic Algorithm with Evolutionary Jumps”. Lecture Notes in Computer Science, ISBRA 2023.

7.2 List of Presentations

“Scalable Reconstruction of SARS-COV-2 Phylogeny with Recurrent Mutations.”

1. ICCABS Conference 2020: CANGS Workshop

2. ICCABS Conference 2021: CANGS Workshop

3. ICCABS Conference 2021: CAME Workshop

4. MBD Fellowship Retreat 2022

REFERENCES

[1] J. Jooken, P. Leyman, and P. De Causmaecker, "A new class of hard problem instances for the 0–1 knapsack problem," *European Journal of Operational Research*, vol. 301, no. 3, pp. 841–854, 2022.

[2] S. Elbe and G. Buckland-Merrett, "Data, disease and diplomacy: GISAID's innovative contribution to global health," *Global Challenges*, vol. 1, pp. 33–46, 2017.

[3] W. T. Harvey, A. M. Carabelli, and J. B. et al., "SARS-CoV-2 variants, spike mutations and immune escape," *Nat Rev Microbiol*, 2021.

[4] K. Jahn, J. Kuipers, and N. Beerenwinkel, "Tree inference for single-cell data," *Genome biology*, vol. 17, no. 1, p. 86, 2016.

[5] P. Skums, V. Tsyvina, and A. Zelikovsky, "Inference of clonal selection in cancer populations using single-cell sequencing data," *Bioinformatics*, vol. 35, no. 14, pp. i398–i407, 2019.

[6] V. Tsyvina, A. Zelikovsky, S. Snir, and P. Skums, "Inference of mutability landscapes of tumors from single cell sequencing data," *PLoS Computational Biology*, vol. 16, no. 11, p. e1008454, 2020.

[7] S. Ciccolella, C. Ricketts, M. S. Gomez, M. Patterson, D. Silverbush, P. Bonizzoni, I. Hajirasouliha, and G. D. Vedova, "Inferring cancer progression from single-cell sequencing while allowing mutation losses," *Bioinformatics*, vol. 37, pp. 326–333, February 2021.

[8] L. van Dorp, M. Acman, D. Richard, L. P. Shaw, C. E. Ford, L. Ormond, C. J. Owen, J. Pang, C. C. Tan, F. A. Boshier, A. T. Ortiz, and F. Balloux, "Emergence of genomic diversity and

recurrent mutations in sars-cov-2," *Infection, Genetics and Evolution*, vol. 83, p. 104351, 2020.

[9] L. van Dorp, D. Richard, C. C. S. Tan, L. P. Shaw, M. Acman, and F. Balloux, "No evidence for increased transmissibility from recurrent mutations in sars-cov-2," *Nature communications*, vol. 11, p. 5986, November 2020.

[10] P. Skums, A. Kirpich, P. I. Baykal, A. Zelikovsky, and G. Chowell, "Global transmission network of sars-cov-2: from outbreak to pandemic," *medRxiv*, 2020.

[11] M. R. Smith, "Bayesian and parsimony approaches reconstruct informative trees from simulated morphological datasets," *Biology Letters*, vol. 15, 2019.

[12] J. R. Fauver, M. E. Petrone, E. B. Hodcroft, K. Shioda, H. Y. Ehrlich, A. G. Watts, C. B. Vogels, A. F. Brito, T. Alpert, A. Muyombwe, *et al.*, "Coast-to-coast spread of sars-cov-2 during the early epidemic in the united states," *Cell*, vol. 181, no. 5, pp. 990–996, 2020.

[13] J. Sukumaran and M. T. Holder, "Dendropy phylogenetic computing library, `https://dendropy.org/`," 2009-2021.

[14] A. Sand, M. K. Holt, J. Johansen, G. S. Brodal, T. Mailund, and C. N. S. Pedersen, "tqDist: a library for computing the quartet and triplet distances between binary or general trees," *Bioinformatics*, vol. 30, pp. 2079–2080, 03 2014.

[15] P. Forster, L. Forster, C. Renfrew, and M. Forster, "Phylogenetic network analysis of sars-cov-2 genomes," *Proceedings of the National Academy of Sciences*, vol. 117, no. 17, pp. 9241–9243, 2020.

[16] R. Sanjuán and P. Domingo-Calap, "Mechanisms of viral mutation," *Cellular and Molecular*

Life Sciences, vol. 73, pp. 4433–4448, July 2016.

[17] C. J. Houldcroft, M. A. Beale, and J. Breuer, "Clinical and biological insights from viral genome sequencing," *Nature Reviews Microbiology*, vol. 15, pp. 183–192, jan 2017.

[18] L. Excoffier and P. E. Smouse, "Using allele frequencies and geographic subdivision to reconstruct gene trees within a species: molecular variance parsimony.," *Genetics*, vol. 136, pp. 343–359, Jan. 1994.

[19] H. J. Bandelt, P. Forster, and A. Rohl, "Median-joining networks for inferring intraspecific phylogenies," *Molecular Biology and Evolution*, vol. 16, pp. 37–48, Jan. 1999.

[20] S. Prabhakaran, M. Rey, O. Zagordi, N. Beerenwinkel, and V. Roth, "HIV haplotype inference using a propagating dirichlet process mixture model," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, pp. 182–191, Jan. 2014.

[21] D. S. Campo, G.-L. Xia, Z. Dimitrova, Y. Lin, J. C. Forbi, L. Ganova-Raeva, L. Punkova, S. Ramachandran, H. Thai, P. Skums, S. Sims, I. Rytsareva, G. Vaughan, H.-J. Roh, M. A. Purdy, A. Sue, and Y. Khudyakov, "Accurate genetic detection of hepatitis c virus transmissions in outbreak settings," *Journal of Infectious Diseases*, vol. 213, pp. 957–965, Nov. 2015.

[22] O. Glebova, S. Knyazev, A. Melnyk, A. Artyomenko, Y. Khudyakov, A. Zelikovsky, and P. Skums, "Inference of genetic relatedness between viral quasispecies from sequencing data," *BMC Genomics*, vol. 18, Dec. 2017.

[23] P. Skums, A. Zelikovsky, R. Singh, W. Gussler, Z. Dimitrova, S. Knyazev, I. Mandric, S. Ramachandran, D. Campo, D. Jha, L. Bunimovich, E. Costenbader, C. Sexton, S. O'Connor, G.-L. Xia, and Y. Khudyakov, "QUENTIN: reconstruction of disease transmissions from vi-

ral quasispecies genomic data," *Bioinformatics*, vol. 34, pp. 163–170, June 2017.

[24] C. Wymant, M. Hall, O. Ratmann, D. Bonsall, T. Golubchik, M. de Cesare, A. Gall, M. Cornelissen, and C. F. and, "PHYLOSCANNER: Inferring transmission from within- and between-host pathogen genetic diversity," *Molecular Biology and Evolution*, vol. 35, pp. 719–733, Nov. 2017.

[25] S. Knyazev, L. Hughes, P. Skums, and A. Zelikovsky, "Epidemiological data analysis of viral quasispecies in the next-generation sequencing era," *Briefings in Bioinformatics*, vol. 22, pp. 96–108, June 2020.

[26] A. Melnyk, S. Knyazev, F. Vannberg, L. Bunimovich, P. Skums, and A. Zelikovsky, "Using earth mover's distance for viral outbreak investigations," *BMC Genomics*, vol. 21, dec 2020.

[27] S. Knyazev, V. Tsyvina, A. Shankar, A. Melnyk, A. Artyomenko, T. Malygina, Y. B. Porozov, E. M. Campbell, W. M. Switzer, P. Skums, S. Mangul, and A. Zelikovsky, "Accurate assembly of minority viral haplotypes from next-generation sequencing through efficient noise reduction," *Nucleic Acids Research*, July 2021.

[28] A. Melnyk, F. Mohebbi, S. Knyazev, B. Sahoo, R. Hosseini, P. Skums, A. Zelikovsky, and M. Patterson, "Clustering based identification of SARS-CoV-2 subtypes," in *Computational Advances in Bio and Medical Sciences*, pp. 127–141, Springer International Publishing, 2021.

[29] E. M. Campbell, A. Boyles, A. Shankar, J. Kim, S. Knyazev, R. Cintron, and W. M. Switzer, "MicrobeTrace: Retooling molecular epidemiology for rapid public health response," *PLOS Computational Biology*, vol. 17, p. e1009300, Sept. 2021.

[30] S. L. K. Pond, S. Weaver, A. J. L. Brown, and J. O. Wertheim, "HIV-TRACE (TRAnsmission

cluster engine): a tool for large scale molecular epidemiology of HIV-1 and other rapidly evolving pathogens," *Molecular Biology and Evolution*, vol. 35, pp. 1812–1819, Jan. 2018.

[31] A. G. Longmire, S. Sims, I. Rytsareva, D. S. Campo, P. Skums, Z. Dimitrova, S. Ramachandran, M. Medrzycki, H. Thai, L. Ganova-Raeva, Y. Lin, L. T. Punkova, A. Sue, M. Mirabito, S. Wang, R. Tracy, V. Bolet, T. Sukalac, C. Lynberg, and Y. Khudyakov, "Ghost: global hepatitis outbreak and surveillance technology," *BMC Genomics*, vol. 18, dec 2017.

[32] J. Felsenstein, *Inferring Phylogenies*. Sinauer Associates is an imprint of Oxford University Press, paperback ed., 9 2003.

[33] D. S. Campo, Z. Dimitrova, L. Yamasaki, P. Skums, D. T. Lau, G. Vaughan, J. C. Forbi, C.-G. Teo, and Y. Khudyakov, "Next-generation sequencing reveals large connected networks of intra-host HCV variants," *BMC Genomics*, vol. 15, July 2014.

[34] E. M. Campbell, H. Jia, A. Shankar, D. Hanson, W. Luo, S. Masciotra, S. M. Owen, A. M. Oster, R. R. Galang, M. W. Spiller, S. J. Blosser, E. Chapman, J. C. Roseberry, J. Gentry, P. Pontones, J. Duwve, P. Peyrani, R. M. Kagan, J. M. Whitcomb, P. J. Peters, W. Heneine, J. T. Brooks, and W. M. Switzer, "Detailed transmission network analysis of a large opiate-driven outbreak of HIV infection in the united states," *The Journal of Infectious Diseases*, vol. 216, pp. 1053–1062, Oct. 2017.

[35] I. Alexiev, E. M. Campbell, S. Knyazev, Y. Pan, L. Grigorova, R. Dimitrova, A. Partsuneva, A. Gancheva, A. Kostadinova, C. Seguin-Devaux, and W. M. Switzer, "Molecular epidemiology of the HIV-1 subtype b sub-epidemic in bulgaria," *Viruses*, vol. 12, p. 441, Apr. 2020.

[36] I. Alexiev, E. M. Campbell, S. Knyazev, Y. Pan, L. Grigorova, R. Dimitrova, A. Partsuneva,

A. Gancheva, A. Kostadinova, C. Seguin-Devaux, I. Elenkov, N. Yancheva, and W. M. Switzer, "Molecular epidemiological analysis of the origin and transmission dynamics of the HIV-1 CRF01_AE sub-epidemic in bulgaria," *Viruses*, vol. 13, p. 116, Jan. 2021.

[37] D. S. Campo, J. Zhang, S. Ramachandran, and Y. Khudyakov, "Transmissibility of intra-host hepatitis c virus variants," *BMC Genomics*, vol. 18, Dec. 2017.

[38] K. M. Grande, C. L. Schumann, M. C. B. Ocfemia, J. M. Vergeront, J. O. Wertheim, and A. M. Oster, "Transmission patterns in a low HIV-morbidity state — wisconsin, 2014–2017," *MMWR. Morbidity and Mortality Weekly Report*, vol. 68, pp. 149–152, Feb. 2019.

[39] K. Tamura and M. Nei, "Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees.," *Molecular Biology and Evolution*, May 1993.

[40] A. M. Oster, A. M. France, N. Panneer, M. C. B. Ocfemia, E. Campbell, S. Dasgupta, W. M. Switzer, J. O. Wertheim, and A. L. Hernandez, "Identifying clusters of recent and rapid HIV transmission through analysis of molecular surveillance data," *JAIDS Journal of Acquired Immune Deficiency Syndromes*, vol. 79, pp. 543–550, Dec. 2018.

[41] A. S. Gonzalez-Reiche, M. M. Hernandez, M. J. Sullivan, B. Ciferri, H. Alshammary, A. Obla, S. Fabre, G. Kleiner, J. Polanco, Z. Khan, B. Alburquerque, A. van de Guchte, J. Dutta, N. Francoeur, B. S. Melo, I. Oussenko, G. Deikus, J. Soto, S. H. Sridhar, Y.-C. Wang, K. Twyman, A. Kasarskis, D. R. Altman, M. Smith, R. Sebra, J. Aberg, F. Krammer, A. García-Sastre, M. Luksza, G. Patel, A. Paniz-Mondolfi, M. Gitman, E. M. Sordillo, V. Simon, and H. van Bakel, "Introductions and early spread of sars-cov-2 in the new york city

area," *Science*, vol. 369, pp. 297–301, May 2020.

[42] D. Novikov, S. Knyazev, M. Grinshpon, P. I. Baykal, P. Skums, and A. Zelikovsky, "Scalable reconstruction of SARS-CoV-2 phylogeny with recurrent mutations," *Journal of Computational Biology*, Oct. 2021.

[43] P. Skums, A. Kirpich, P. I. Baykal, A. Zelikovsky, and G. Chowell, "Global transmission network of SARS-CoV-2: from outbreak to pandemic," Mar. 2020.

[44] P. Forster, L. Forster, C. Renfrew, and M. Forster, "Phylogenetic network analysis of SARS-CoV-2 genomes," *Proceedings of the National Academy of Sciences*, vol. 117, pp. 9241–9243, Apr. 2020.

[45] A. Stamatakis, "RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies," *Bioinformatics*, vol. 30, pp. 1312–1313, Jan. 2014.

[46] F. Campbell, X. Didelot, R. Fitzjohn, N. Ferguson, A. Cori, and T. Jombart, "outbreaker2: a modular platform for outbreak reconstruction," *BMC Bioinformatics*, vol. 19, Oct. 2018.

[47] D. Klinkenberg, J. Backer, X. Didelot, C. Colijn, and J. Wallinga, "New method to reconstruct phylogenetic and transmission trees with sequence data from infectious disease outbreaks," Aug. 2016.

[48] N. Beerenwinkel, T. Sing, T. Lengauer, J. Rahnenführer, K. Roomp, I. Savenkov, R. Fischer, D. Hoffmann, J. Selbig, K. Korn, H. Walter, T. Berg, P. Braun, G. Fätkenheuer, M. Oette, J. Rockstroh, B. Kupfer, R. Kaiser, and M. Däumer, "Computational methods for the design of effective therapies against drug resistant HIV strains," *Bioinformatics*, vol. 21, pp. 3943–3950, September 2005.

[49] D. C. Douek, P. D. Kwong, and G. J. Nabel, "The rational design of an AIDS vaccine," *Cell*, vol. 124, no. 4, pp. 677–681, 2006.

[50] B. Gaschen, J. Taylor, K. Yusim, B. Foley, F. Gao, D. Lang, V. Novitsky, B. Haynes, B. H. Hahn, T. Bhattacharya, and B. Korber, "Diversity considerations in HIV-1 vaccine selection," *Science*, vol. 296, no. 5577, pp. 2354–2360, 2002.

[51] J. Holland, J. De La Torre, and D. Steinhauer, "RNA virus populations as quasispecies," *Current Topics in Microbiology and Immunology*, pp. 1–20, 1992.

[52] S.-Y. Rhee, T. F. Liu, S. P. Holmes, and R. W. Shafer, "HIV-1 subtype B protease and reverse transcriptase amino acid covariation," *PLOS Computational Biology*, vol. 3, pp. 1–8, May 2007.

[53] D. S. Campo, P. Skums, Z. Dimitrova, G. Vaughan, J. C. Forbi, C.-G. Teo, Y. Khudyakov, and D. T.-Y. Lau, "Drug resistance of a viral population and its individual intrahost variants during the first 48 hours of therapy," *Clinical Pharmacology and Therapeutics*, vol. 95, pp. 627–635, June 2014.

[54] P. Skums, L. Bunimovich, and Y. Khudyakov, "Antigenic cooperation among intrahost HCV variants organized into a complex network of cross-immunoreactivity," *Proceedings of the National Academy of Sciences*, vol. 112, no. 21, pp. 6653–6658, 2015.

[55] D. S. Campo, G.-L. Xia, Z. Dimitrova, Y. Lin, J. C. Forbi, L. Ganova-Raeva, L. Punkova, S. Ramachandran, H. Thai, P. Skums, S. Sims, I. Rytsareva, G. Vaughan, H.-J. Roh, M. A. Purdy, A. Sue, and Y. Khudyakov, "Accurate genetic detection of hepatitis C virus transmissions in outbreak settings," *The Journal of Infectious Diseases*, vol. 213, pp. 957–965, March

2016.

[56] O. Glebova, S. Knyazev, A. Melnyk, A. Artyomenko, Y. Khudyakov, A. Zelikovsky, and P. Skums, "Inference of genetic relatedness between viral quasispecies from sequencing data," *BMC Genomics*, 2017.

[57] P. Skums, A. Zelikovsky, R. Singh, W. Gussler, Z. Dimitrova, S. Knyazev, I. Mandric, S. Ramachandran, D. Campo, D. Jha, L. Bunimovich, E. Costenbader, C. Sexton, S. O'Connor, G.-L. Xia, and Y. Khudyakov, "QUENTIN: Reconstruction of disease transmissions from viral quasispecies genomic data," *Bioinformatics*, vol. 34, pp. 163–170, June 2017.

[58] M. Bousali, A. Dimadi, E.-G. Kostaki, S. Tsiodras, G. K. Nikolopoulos, D. N. Sgouras, G. Magiorkinis, G. Papatheodoridis, V. Pogka, G. Lourida, A. Argyraki, E. Angelakis, G. Sourvinos, A. Beloukas, D. Paraskevis, and T. Karamitros, "SARS-CoV-2 molecular transmission clusters and containment measures in ten european regions during the first pandemic wave," *Life*, vol. 11, no. 3, 2021.

[59] A. Melnyk, S. Knyazev, F. Vannberg, L. Bunimovich, P. Skums, and A. Zelikovsky, "Using earth mover's distance for viral outbreak investigations," *BMC Genomics*, vol. 21, no. 582, 2020.

[60] W. F. de la Vega, M. Karpinski, C. Kenyon, and Y. Rabani, "Approximation schemes for clustering problems," in *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pp. 50—-58, Association for Computing Machinery, 2003.

[61] S. Khare, C. Gurry, L. Freitas, M. B. Schultz, G. Bach, A. Diallo, N. Akite, J. Ho, R. T. Lee, W. Yeo, G. Core Curation Team, and S. Maurer-Stroh, "GISAID's role in pandemic

response," *China CDC weekly*, vol. 3, no. 49, pp. 1049—1051, 2021.

[62] T. Li, S. Ma, and M. Ogihara, "Entropy-based criterion in categorical clustering," in *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, vol. 3, pp. 536–543, 2004.

[63] S. Knyazev, V. Tsyvina, A. Shankar, A. Melnyk, A. Artyomenko, T. Malygina, Y. B. Porozov, E. M. Campbell, W. M. Switzer, P. Skums, S. Mangul, and A. Zelikovsky, "Accurate assembly of minority viral haplotypes from next-generation sequencing through efficient noise reduction," *Nucleic Acids Research*, vol. 49, pp. e102–e102, July 2021.

[64] A. Melnyk, F. Mohebbi, S. Knyazev, B. Sahoo, R. Hosseini, P. Skums, A. Zelikovsky, and M. Patterson, "From Alpha to Zeta: Identifying variants and subtypes of SARS-CoV-2 via clustering," *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, vol. 28, no. 11, pp. 1113–1129, 2021.

[65] EMBL-EBI, "EMBL's European Bioinformatics Institute."

[66] S. Knyazev, K. Chhugani, V. Sarwal, R. Ayyala, H. Singh, S. Karthikeyan, D. Deshpande, P. I. Baykal, Z. Comarova, A. Lu, *et al.*, "Unlocking capacities of genomics for the covid-19 response and future pandemics," *Nature Methods*, vol. 19, no. 4, pp. 374–380, 2022.

[67] D. Novikov, S. Knyazev, M. Grinshpon, P. Icer, P. Skums, and A. Zelikovsky, "Scalable reconstruction of sars-cov-2 phylogeny with recurrent mutations," *Journal of Computational Biology*, vol. 28, no. 11, pp. 1130–1141, 2021.

[68] F. Mohebbi, A. Zelikovsky, S. Mangul, G. Chowell, and P. Skums, "Community structure and temporal dynamics of sars-cov-2 epistatic network allows for early detection of emerging

variants with altered phenotypes," *bioRxiv*, pp. 2023–04, 2023.

[69] S. Knyazev, V. Tsyvina, A. Shankar, A. Melnyk, A. Artyomenko, T. Malygina, Y. B. Porozov, E. M. Campbell, W. M. Switzer, P. Skums, *et al.*, "Cliquesnv: an efficient noise reduction technique for accurate assembly of viral variants from ngs data," *bioRxiv*, vol. 264242, 2020.

[70] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[71] J. H. Holland, "Adaptation in natural and artificial systems," *University of Michigan Press*, 1975.

[72] A. F. Gad, "Pygad: An intuitive genetic algorithm python library," 2021.