Optimizing human contribution in citizen science puzzles using refined selection methods

Timothy David Keding

Masters of Science

School of Computer Science

McGill University

Montréal, Québec, Canada

August 4, 2022

A thesis presented for the degree of Masters of Science

Abstract

Multiple Sequence Alignment has been a long-standing challenge in the field of bioinformatics. It is a NP-complete problem that has resulted in numerous different algorithms, yet each introduces its own set of limitations. To further complicate matters, there can be multiple different ways to evaluate an alignment based on the user's objectives. Rather than develop a new method, Borderlands Science attempts to utilize citizen science to improve existing algorithms. Puzzles are constructed using a subset of the alignment, and given to players in an attempt to identify which areas can be improved.

We present a study focused on refining and optimizing puzzle generation. A collection of methods was used to improve the quality of data provided to players. This was done in the hopes that a set of refined puzzles targeting specific regions has a larger impact than an over saturated set of generic puzzles. We will also cover the multiple steps required to process player solutions in order to obtain an optimal new alignment.

Abrégé

Le problème de l'alignement multiple est un défi de longue date pour les chercheurs en bioinformatique. Il s'agit d'un problème NP-complet pour lequel de nombreux alignements ont été conçus, chacun amenant un nouvel ensemble de limitations. De surcroît, il n'y a pas de manière universelle d'évaluer un alignement: différentes méthodes sont utilisées en fonction des objectifs de l'utilisateur. Plutôt que d'ajouter un algorithme à la liste, Borderlands Science propose d'utiliser la science participative pour améliorer des algorithmes existants. Des puzzles sont conçus à partir d'un sous-ensemble d'un alignement, et envoyés aux joueurs dans l'objectif de leur faire identifier quelles régions peuvent être améliorées.

Nous présentons une étude centrée sur le raffinement et l'optimization de la génération de ces puzzles. Plusieurs méthodes sont mises à contribution pour améliorer la qualité des données fournies aux joueurs, avec pour objectif qu'un ensemble de puzzles plus sélectif, ciblant des régions spécifiques, ait un impact plus significatif que des puzzles généraux.

Nous aborderons également les nombreuses étapes de notre analyse des solutions proposées par les joueurs visant à obtenir un nouvel alignement optimal.

Contributions

Timothy David Keding has performed the research described in this thesis. He developed the methods and the comparison experiments, under the guidance of his supervisor Jérôme Waldispühl. The puzzle generation and realignment methods were built from a foundational code base written by Roman Sarrazin-Gendron. Jiayue Zheng assisted with gathering player solutions from our database, and Eddie Cai developed the methods to calculate the pareto front. Parham Ghasemloo expanded on the base realignment methods, including post-processing and width reduction.

Acknowledgements

I would like to thank my supervisor Jérôme Waldispühl for taking me on as a student, and introducing me to the field of bioinformatics. He was an incredible mentor, guiding me as I learned how to adjust to a research-based position, and provided the resources necessary to conduct the thesis work presented here.

Next, I'd like to thank my fellow lab members Roman Sarrazin-Gendron, Eddie Cai, Parham Ghasemloo, Renata Mutalova, Jiayue Zheng, and Alexander Butyaev. Each of you played a valuable role in the Borderlands Science project, and were always ready to help when asked. More importantly, your online presence and wonderful conversations made the graduate experience enjoyable despite being restricted to working from home for a majority of the program.

Lastly, I'd like to thank my friends and family. They have been there throughout this program to celebrate my successes, bear my troubles, and support my decisions. My parents (Martin Keding and Sara Rempel), my siblings (Kylie, Daniel, and Andrew Keding), my loving fiancée (Ashley Ste-Croix), as well as my future in-laws (Jennifer,

Brian, and Kevin Ste-Croix) have all played an instrumental role in my accomplishments, and they will continue to support me as I continue to advance my career.

Contents

List of Acronyms

BLS	Borderlands Science.
CS	Citizen science.
DNA	Deoxyribonucleic acid.
HW	Hard Window.
KC	Kendall-Colijn.
MSA	Multiple Sequence Alignment.
MUSCLE	MUltiple Sequence Comparison by Log-Expectation.
NP	Nondeterministic Polynomial-time.
PASTA	Practical Alignment using SATé and TrAnsitivity.
RNA	Ribonucleic acid.
SATE	Simultaneous Alignment and Tree Estimation.
SW	Soft Window.

Chapter 1

Introduction and Background

1.1 Multiple Sequence Alignment

Multiple Sequence Alignment (MSA) is a common problem in bioinformatics, and refers to a sequence alignment consisting of three or more biological sequences, typically DNA, RNA, or protein [1] as seen in Figure 1.1. In most cases, these sequences are assumed to have an evolutionary relationship, and are descendants of a common ancestor. By aligning sequences based on nucleotides, and accounting for events such as mutations, insertions, deletions, and rearrangements, homology can be inferred. The resulting output can provide insight to the purposes of certain regions, 2D/3D structure prediction, or changes which lead to species divergence [2].

Computing a MSA has long been known to be an NP-complete problem [3], a situation

Sequence 1	T	-	C	G	T	C	G	G	T	G	G	C	G	C	-
Sequence 2	T	A	C	G	A	A	G	G	G	T	G	C	A	A	G
Sequence 3	T	A	C	G	T	A	G	G	A	G	G	C	G	A	G
Sequence 4	T	A	-	G	G	C	T	T	T	T	G	C	A	A	G
Sequence 5	T	A	C	G	A	G	G	G	-	G	G	C	A	A	G
Sequence 6	T	A	C	G	G	A	-	-	A	T	G	C	A	A	G

Figure 1.1: A series of sequences forming a MSA

which explains why over 100 alternative methods have been developed in the last three decades [4]. While all these existing algorithms are effective, they are rarely perfect. In most algorithms, the goal is to achieve an alignment maximizing a particular score, often sum-of-pairs [5]. Others have proposed improved methods which take into account the phylogenetic tree [6]. Regardless of the method and scoring scheme, it still proves daunting to compute the best alignments, as they are NP-complete. Even if we were capable of obtaining a perfect alignment, various scoring schemes ensure it is difficult to validate that claim. As a result, rather than developing a new MSA algorithm, our goal is to incorporate citizen science to improve upon existing algorithms. For the purposes of this work, we used the Borderlands Science (BLS) project for all citizen science information.

1.2 Citizen Science

Citizen science (CS), also known as community science or crowd-sourced science, is scientific research conducted through crowd-sourcing using willing participants. Under the direction

of professional scientists, data is gathered and analyzed utilizing the collective strength of a community [7]. Rick Bonney et al. [2016] defined CS as projects in which nonscientists, amateur birdwatchers as an example, voluntarily contribute scientific data [8].

Citizen involvement not only provides scientists with a large dataset that would be otherwise unavailable [9], but it also encourages curiosity, and presents nonscientists an opportunity to contribute and learn. As time goes on, citizen science continues to grow and mature, but one key metric of a projects success is data quality [10]. It is vital that the right questions are asked so volunteer contributions are insightful and have a meaningful impact. As such, methods to refine information gathered act as a critical foundation to all citizen science projects.

1.3 Citizen Science in Bioinformatics

Coincidentally, humans excel with the complications found in multiple sequence alignments. The human mind is very effective at handling multiple constraints at once, and intuitively understands the delicate balance of minimizing gaps, while maximizing score [11]. Intuition has its limits however, and while a human may excel at aligning a small MSA, a set of hundreds, or thousands of sequences is overwhelming for an individual. In order to overcome this, we take an alignment generated from an existing MSA algorithm, and divide it into subsections to investigate. This approach provides humans with manageable subsections to process, a base score to improve upon, a challenge, and an easy method for crowd sourcing

a much larger problem.

1.4 Existing MSA Algorithms

There are numerous different approaches to handle a MSA problem. While many attempt to maximize the sum-of-pairs score, it is far from the only criteria. Furthermore, many algorithms are designed to produce results in a reasonable amount of time, which typically involves taking shortcuts based on defined priorities.

1.4.1 Practical Alignment using SATé and TrAnsitivity (PASTA)

PASTA is a MSA algorithm that uses divide-and-conquer plus iteration to allow base alignment methods to scale with high accuracy to large sequence datasets. PASTA computes an initial tree, and then iterates between alignment estimation, and tree estimation. Each iteration uses a small subset of at most 200 sequences, and merges the resulting alignments into a full dataset. PASTA builds upon Simultaneous Alignment and Tree Estimation (SATé). It is not only faster, but highly parallelizable, and requires relatively little memory [12]. This method has the trait of creating compact alignments, which are well suited for the nucleotide dense puzzles provided to players.

1.4.2 MUltiple Sequence Comparison by Log-Expectation (MUSCLE)

MUSCLE is a popular method for multiple sequence alignment. First introduced in 2004, it breaks the MSA into three stages. First, it builds a progressive alignment by observing similarities of pairs of sequences, and constructs a tree. Stage two involves constructing a second, more refined progressive alignment using the tree from stage one, producing a second tree, and comparing it to the first. This stage may cycle as much as necessary. Finally, MUSCLE refines the previous results to give a final output [13]. While MUSCLE is effective and reliable, the alignments it produces are notably far less dense than PASTA, resulting in fewer clusters of nucleotides well suited for puzzles. As a result, we developed a secondary approach to account for these types of alignments.

1.5 Phylo

Borderlands Science is not the first to investigate utilizing citizen science games in bioinformatics. Foldit studies protein folding [14], while Phylo and its successor Open-Phylo [15] [16] are casual games using citizen science to solve the MSA problem, and lay the foundation for BLS. While Phylo may appear similar, it differs from BLS. Phylo aims to align mammalian genes, where some prior knowledge of the sequences is known, while BLS focuses on improving upon a pre-existing microbial RNA sequence alignment,

which is a significantly larger problem. Furthermore, the sequences are not related to specific species, thus the method of evaluation differs. In order to approach these tasks, Phylo utilized a website, where they received over 350,000 submitted solutions. BLS opted for a more refined game design in order to reach a larger audience, and can be found in a mini-game in the AAA title *Borderlands 3*. While the puzzle design was simplified, it rectifies scientific accuracy through accessibility, where it received over 500,00 solutions every week throughout the duration of the project.

1.6 Borderlands Science

The overall objective of the Borderlands Science project is to use CS to improve upon an existing alignment. By achieving this, we not only produce a better alignment, but also an understanding of player methodology, and how it can be incorporated into refined algorithms. The Microsetta initiative provided a large set of V4 hypervariable regions from 16S rRNA, which was then aligned using PASTA or MUSCLE. From that point, the alignment was investigated for regions which could possibly be improved, and were then given to players. Finally, the player solutions were collected and used to construct a refined alignment.

The focus of this work was in the former half of the project, determining which areas required improvement, the impact of the base alignment, and ensuring feedback from players yielded a positive impact. Producing puzzles where players will be challenged, and contribute valuable data allowed us to achieve better results using fewer puzzles. This

refined puzzle selection also filtered out noisy puzzles which give minimal value, and hindered the realignment.

To that end, we investigated different approaches for narrowing search parameters and the process for finding sub-optimal regions. We also examined different methods of evaluation, and how they compared to player solutions to justify a citizen science approach.

Finally, the global realignment, as well as the puzzle solutions themselves, were studied to examine the impact of different parameters, and methods.

Chapter 2

Puzzle Generation

2.1 Terminology

Puzzles are not in the same orientation as the alignment and therefore terms cannot be used interchangeably. For this reason, when we use the terms 'rows' or 'columns', we are referring to the game structure in which each row is an alignment position, and each column is a sequence. To describe elements of the alignments, we will use the terms 'sequences', or 'tips' to refer to vertical coordinates, and the term 'position' for horizontal coordinates.

2.2 Rules and Objectives

When a player commences a puzzle they are provided a two-dimensional grid with a set number of columns and rows based on the difficulty. The order of the columns is randomized,

and each column in the grid is a subset of a genomic RNA sequence. Columns contain an assortment of nucleotides represented by four coloured bricks. Colours are randomly assigned to the nucleotides adenine, cytosine, guanine, and thymine at the start of a puzzle to prevent bias towards a particular colour. A guide is presented alongside the grid, with one to two coloured bricks per row. If a brick in the grid matches one of the guide bricks in the associated row, the total score increases by one, otherwise it is worth zero points. If all bricks in the row match the guide, the score for the row is equal to the number of columns multiplied by 1.15. A player is given a set number of gap bricks, which they can insert in between bricks anywhere in the grid. Once a gap brick is placed, it pushes all proceeding bricks in the column upward by one. Likewise, removing a gap brick causes all proceeding bricks in the column to collapse by one. A brick cannot surpass the boundaries of the guide. The objective of the game is to insert gap bricks into the grid to align with the guide as well as possible and maximize the score. Figure 2.1 illustrates the puzzle environment provided to players.

These are the pre-established rules of the game determined during the initial design in partnership with Gearbox Software, and are not subject to change. The only methods which can be changed are how the puzzles are generated, and how the player solutions are handled.

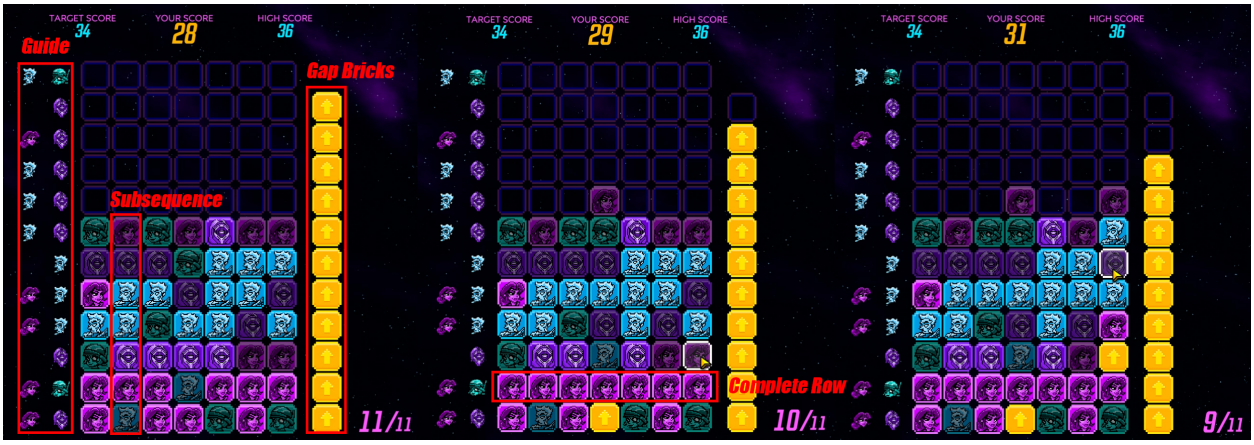


Figure 2.1: Basic overview of a Borderlands Science puzzle

2.3 Puzzle Construction

2.3.1 Finding Representatives

Given a set of one million sequences from the Microsetta initiative, the number of potential puzzles is daunting, and would result in an excessive amount of redundant data. Given a dataset of one million sequences, each consisting of 150 nucleotides, and a puzzle with the dimensions 6x8, there are $1.41e^{38}$ possible puzzle combinations containing various sequences and regions. In order to rectify this, we used CD-HIT [17], a widely used program for clustering biological sequences to reduce redundancy and improve the performance of other sequence analyses. This allowed us to break the one million sequences into ten thousand clusters based on similarity, with the first in each cluster being the definitive longest sequence. We took the first sequence in each cluster as the representative sequence and gathered all representatives into a new set used for puzzle generation. Utilizing CD-Hit reduced

our possibilities to $1.41e^{26}$. Once the new set was subjected to a puzzle solution informed realignment, each representative could be returned to their respective cluster, and a profile alignment could then be performed to align each cluster [18].

2.3.2 Puzzle Origins

Once a set of representative sequences had been produced, it could be subjected to an alignment algorithm. Due to the size of the alignment file, it is unrealistic to expect humans to produce a good alignment from scratch given small batches without insight on the larger picture. It is more reasonable to run the set of sequences through an existing alignment algorithm, and utilize that output as the baseline for puzzle production. For our purposes, the alignment file was ran through PASTA due to a dense alignment with compact regions of nucleotides ideal for our puzzles. From that point, puzzles were generated from the PASTA alignment, searching for small regions which our algorithms believed could be improved. Further testing also explored using MUSCLE alignment for comparison, versatility, and flexibility.

2.3.3 Puzzle Sizes

Prior to creating any puzzles, grid size must first be determined. Puzzles are defined by the number of sequences, displayed as columns, and number of alignment positions per sequence, displayed as rows. Puzzles also contain an additional empty two rows at the top of

the puzzle containing guide information, but no sequence nucleotides. This ensured that a certain number of gaps could be inserted without surpassing the guide. Borderlands Science utilizes ten difficulties, with a number of sequences between six to nineteen, and columns ranging from seven to twelve.

2.3.4 Window Type

When searching for regions suitable for player puzzles, a number of consecutive alignment positions must be observed based on difficulty. In order to achieve this, we used a sliding window algorithm, going over every potential region. We developed two different methods to utilize this sliding window, The first being a simple hard window, while the other being a more flexible soft window.

Hard Window

The first approach for finding alignment regions for a puzzle is a hard window. Using a sliding window with a size equal to the number of puzzle rows specified, we can obtain a region with subsets of all sequences demonstrated in Figure 2.2. While this method does have its benefits, it also has drawbacks.

First, this method is very fast, efficient, and reliable. It allows the puzzle generation to find small dense regions of nucleotides, and determine if they would benefit from inserting,

shifting, or removing gaps. Second, this window may include positions containing primarily gaps, providing opportunities to condense the alignment.

The first potential drawback is the existing guide used for solving puzzles. The guide finds the two most common nucleotides per region (treating gaps as nucleotides) and asks the player to attempt to align the puzzle to the guide as well as possible. If a region has 99% gaps, and 1% Adenine (A), the guide will display (-, A) for the row. Due to the simplicity, the puzzle will reward a player for aligning an 'A', yet will not give any incentive for inserting a gap. This causes many regions which the original alignment algorithm determined should be gaps, to collapse further. This is not necessarily bad, but it does impose a bias.

The other larger drawback is the restriction to dense clusters of nucleotides. This is less of an issue using an already dense alignment such as PASTA as the base, but does cause difficulties using alignments like MUSCLE which is far more wide spread, and may not have enough nucleotides within a defined window size. An array of positions consisting of primarily gaps may be the result of a single sequence, and by ignoring the sequence, those positions would be irrelevant and removed. Since we do not want to impose our own bias, the sequence remains, and the large array of gaps creates a divide preventing hard windows from creating certain puzzles.

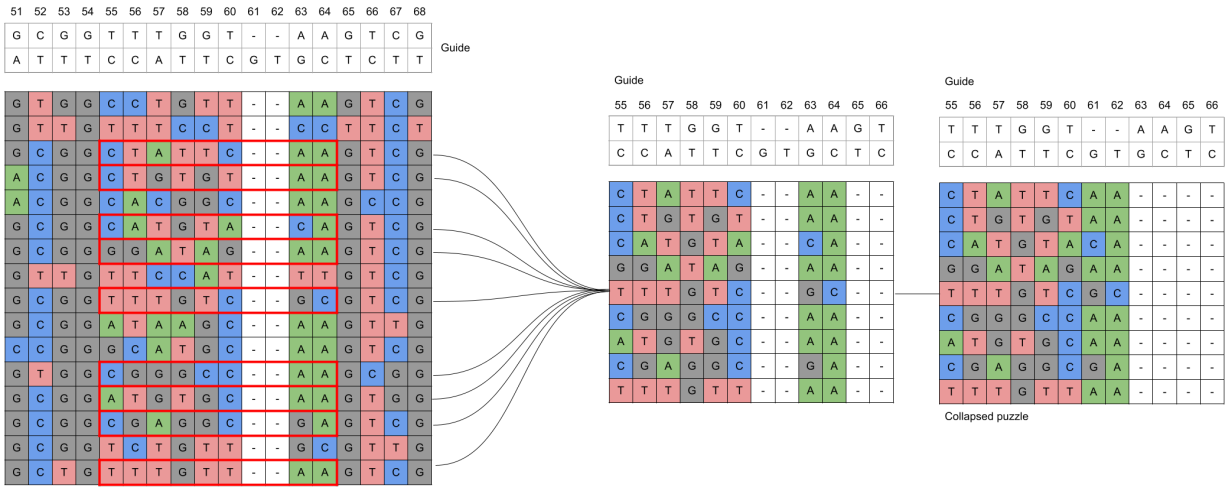


Figure 2.2: Hard window method selecting a puzzle region from a dense alignment.

Soft Window

The second method that was implemented is a soft window. Using a sliding window with a size of one, we can obtain a starting row for the puzzle. From that point the soft window retrieves the next x nucleotides per sequence, rather than the next x positions. This ensures each sequence has a subset of x nucleotides. Once a set of sequences is chosen for the puzzle, we align the sequences based on original alignment positions, and trim to retrieve the first x positions. The end result is a set of sequences which span x positions, conforming to the requirements of the puzzle, but ignoring positions consisting of only gaps irrelevant to the given sequences. Figure 2.3 illustrates the steps taken in the soft window process. As with hard windows, this method has its perks and quirks.

The primary benefit is that soft windows are better suited for wide-spanning alignments

such as MUSCLE, no longer restricted to dense regions of nucleotides, and able to produce puzzles using small clusters by ignoring arrays of gap positions.

This also addresses a feature seen in hard windows. Soft windows ignore the positions which the original alignment file determined should be filled with gaps. It does not provide incentive to collapse the alignment by removing those positions from the equation. This does impose a different bias, being unable to consider those gapped positions as part of the puzzle, so window type is important based on conditions.

The main drawback of this method is a more complicated and extensive process than the hard window. It does run slower than the hard window method, but still well within an acceptable runtime for our purposes.

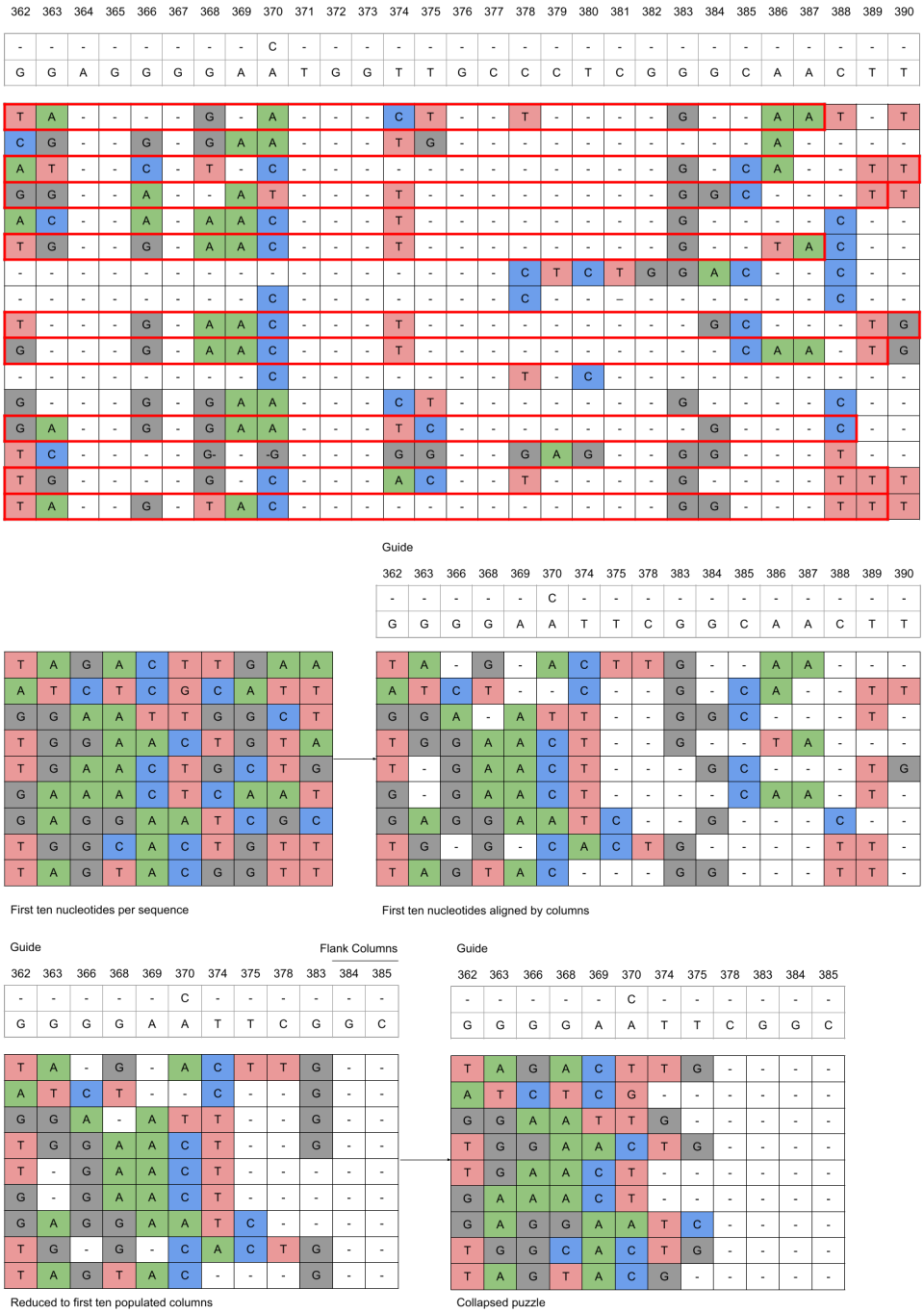


Figure 2.3: Soft window method selecting a puzzle region from a sparse alignment. Obtains the first ten nucleotides from each sequence, then is reduced to the first ten populated columns.

2.3.5 Sequence Selection

After determining a specific window, Borderlands Science randomly selects a single sequence as the core of a puzzle. The remaining sequences are then filtered based on similarity to the core using a rudimentary scoring schema. Matching nucleotides receive a -1, and mismatched nucleotides gain a +2. Sequences that are not identical, yet retain a certain degree of similarity are accepted. The core sequence, along with randomly selected sequences from the filtered results, are then used as the base for the puzzle. If filtering produced too few sequences, or some sequences contained an identical range of nucleotides, the process restarts with another sequence as the core.

2.3.6 Guide

Borderlands Science's predecessor Phylo required players to align all sequences to each other as well as possible using a phylogenetic process. Unfortunately, our set of microbial RNA lack that phylogeny information. In order to overcome this lack of data, a guide was constructed. The guide calculated the two most common nucleotides per row across all sequences, which was then given to the player alongside the puzzle. The players goal is then to align each sequence as closely as possible to the guide.

2.3.7 Direction

Puzzles generated heavily utilized gravity as part of the game mechanics. Gravity in the game acts as a constant force, pushing all nucleotides towards the bottom row. Puzzles start with all gaps removed. Adding a gap to a sequence pushes all proceeding nucleotides in the column upwards, and removing a gap allows gravity to push the affected nucleotides downwards. Furthermore, while the player scoring scheme does not penalize gaps, we wanted to minimize the number of gaps added, so the solution algorithms used for evaluation did impose a minor penalty per gap. The scoring scheme penalized adding gaps at the start of the puzzle, but did not account for gaps after the last nucleotide. In light of this, it took far less effort to align all sequences near the bottom, rather than the top. In order to account for this bias, puzzles could be generated in either direction. Puzzles could be flipped prior to testing, to ensure that the same puzzle could be solved given reversed gravity.

2.3.8 Variety

Players require stimulating games. Repeating similar puzzles can feel tedious over time, and results in less engagement. One step in rectifying this was adding a jagged edge to the top of the puzzle. If all puzzles were a simple rectangle, they would become rather boring to look at over time, and give limited gaps per column. This was alleviated by implementing a small chance of variation. Each sequence had a 10% chance to remove the last nucleotide, and an exponentially smaller chance for each proceeding nucleotide. The product was a

slightly jagged edge at the top of the puzzle which provided more visual appeal, yet the sequences were not so different that it hindered the realignment process. This solution not only addressed engagement, but aided in solution diversity, and removing potential bias.

2.4 Puzzle Testing

2.4.1 Puzzle Base Point

Before a player even starts a puzzle, each sequence subset is collapsed, removing all existing gaps from the original alignment. This provides a player free reign to choose how the local alignment should look, whether it will resemble the original alignment, or something else entirely.

2.4.2 Puzzle Scoring Schema

Player Scoring

Borderland Science's predecessor Phylo utilized an affine gap cost model [19] for its scoring scheme. This scoring is often used in pairwise alignments, and scores nucleotides accordingly. Match = +1, mismatch = -1, gap opening = -4, gap extension = -1. In order to lower the barrier of entry, and accommodate a faster paced game, the player scoring scheme was vastly simplified. If a nucleotide matches the provided guide, they receive a point. Otherwise it is worth zero points, and gaps are worth zero. If all nucleotides in a row match the guide, they

receive bonus points equal to 1.15 times the number of columns in the puzzles.

Solution Algorithm Scoring

As part of the puzzle generation process, a solution algorithm was run to mimic rudimentary player behaviour, and evaluate if a region could be improved. Despite solving the same puzzle, the solution algorithm could incorporate different scoring schemes to subject regions to greater criticism. This was done to ensure better puzzles were chosen, and player solutions would have a greater impact. For the sake of penalizing gaps, while not straying too far from the player scheme, most solution algorithm scoring imposed a -0.6 for each gap added. Later experiments also used the affine gap penalty cost scoring scheme used in Phylo, due to its previous success.

2.4.3 Puzzle Solution Algorithms

In order to determine if a subset of sequences required player analysis, each puzzle was subjected to a solution algorithm. Each algorithm followed a base logic in an attempt to achieve the best score possible. If the results surpassed the acceptance parameters the puzzle was considered worth investigating, and was given to players who would explore the problem with a perspective an algorithm lacks.

Handicapped Greedy Solution Algorithm

Originally, puzzles were subjected to a greedy solution algorithm with a handicap. The greedy algorithm attempts to place, remove, or shift a gap in a position which results in the greatest improvement to the overall score, and repeats this process until the score can no longer be improved. Rather than observing the entire puzzle for places to insert a gap, a gap could only be placed beneath a nucleotide which contained a gap or empty space above it. This leads to only being able to add a gap brick along the top row of the puzzle on the first turn, and sequential turns allow shifting the gap brick down if the score increases. This handicap allowed the runtime to be drastically reduced, only needing to consider a handful of possible positions. While this method produced millions of valid puzzles, the handicap did result in several potentially beneficial puzzles skipped due to improvements only available within the puzzle, rather than at the top row.

Greedy Solution Algorithm

While the handicapped greedy discovered many useful puzzles, it did hinder optimizing human contribution by overlooking regions which may have provided more value. As shown in Figure 2.4, the revised greedy algorithm dismissed the handicap, observing every possible position to add, remove, or shift a gap providing the greatest increase to the score, until no further improvements can be made. In some cases the solution found could not be further improved upon by players, thus the objective score was set at half of what the

greedy algorithm was able to achieve. Even when halving the objective score, the revised greedy algorithm resulted in puzzles with far higher average objective scores than its handicapped version.

The primary hindrance of the revised greedy algorithm is that to evaluate each position in the puzzle for each step, generating thousands of puzzles becomes very costly. In order for this method to be efficient, certain shortcuts were taken to minimize the necessary work.

The first condition is ignoring areas where adding a gap is impossible, or irrelevant. If a column has already reached the maximum height of the guide, no more gaps can be added, thus the column can be ignored. Furthermore, if a gap has been added, it is redundant to try adding a second gap both before and after the existing gap, as it will have the same outcome.

The second step is improving the time required to calculate the score of the puzzle. Rather than looking at each individual nucleotide and determining if it matches the guide, we can retrieve the sum of all nucleotides in a row which match the guide, and if that sum is equal to the number of columns, we can add the row bonus.

Lastly, the most critical change is reducing the number of times needed to recalculate the

score. By calculating the score of the puzzle prior to adding a new gap, called current_score, then adding a gap at the start of a column, recalculating, and calling that future_score, we already have all the information needed for the given column. If you add a gap to row three, you need to make minor adjustments for the individual row, but rows one and two will be equivalent to the rows in current_score, and all rows proceeding row three will be equivalent to future_score. This results in only calculating the score once per column, and then the score of each potential gap position in the given column is a simple equation of (current_score $<$ row) + row + (future_score $>$ row). This reduces the number of times calculating the score from n^2 to n+1.

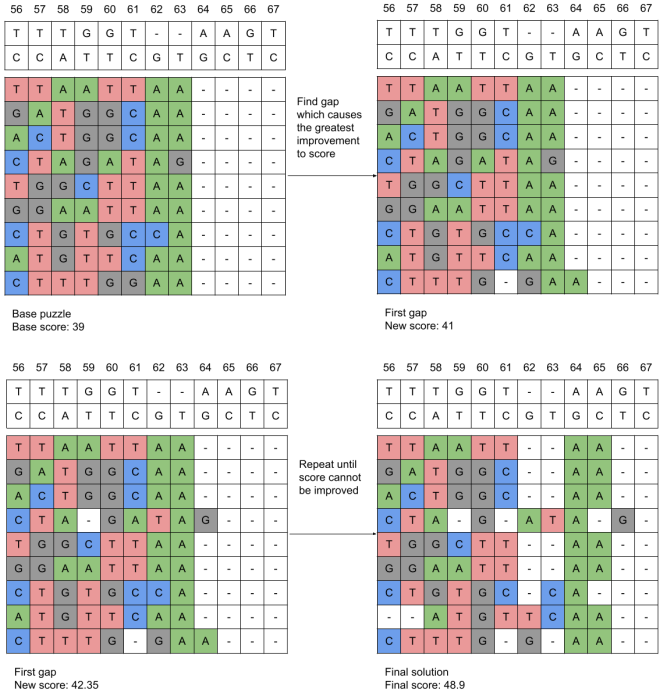


Figure 2.4: The greedy algorithm step-by-step process for solving a puzzle. Finds the move which will contribute to the greatest improvement to the score, and repeats until no more changes can be made.

Profile Alignment Algorithm

The improved greedy algorithm saw a significant increase in average score in comparison to the handicapped method, yet it still has one glaring flaw due to the nature of greedy algorithms. Given a puzzle where multiple gaps are needed before seeing a large improvement, a greedy algorithm will ignore it. Thus a second solution algorithm was developed to account for most cases. The profile solution algorithm performs a modified profile alignment on each column against the guide using an affine gap cost model. Each

column is aligned as best as possible with the guide, then the total puzzle score is calculated the same way as the greedy alignment. Due to processing each column individually, row bonus points are not factored into the column alignments, but the points are still applied should the final solution produce complete rows. The alignment process can be seen in Figure 2.5 below.

The modified profile alignment behaves as a standard profile alignment, with only two exceptions. The first being that the guide has up to two possible nucleotides rather than one, therefore it must consider both as valid options. Second, gaps beyond the sequence are not penalized. Any gaps proceeding the final nucleotide in the sequence do not receive a gap penalty.

In order to remain consistent with the greedy algorithm, the pairwise algorithm utilized the same scoring scheme for most puzzles generated. Some puzzles were generated using Phylo's scoring scheme, as the -4 penalty for opening a gap made the greedy algorithm virtually impossible. The Phylo scoring schema generated far fewer puzzles, but achieved far higher average scores.

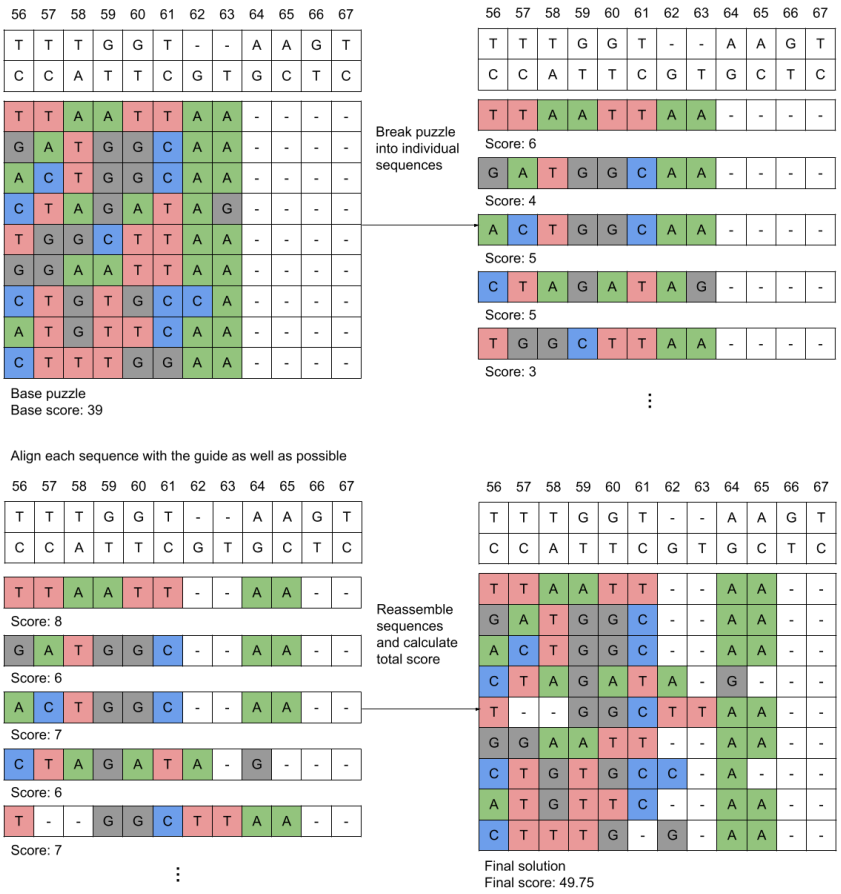


Figure 2.5: The profile alignment algorithm breakdown for constructing a solution. Performs a profile alignment with each individual sequence and the guide, before combining all results to produce a final puzzle solution.

2.5 Acceptance Parameters

2.5.1 Minimum Score

Early phases of the puzzle generation accepted any puzzle whose solution score had improved by at least three in comparison to the collapsed puzzle score. This was determined as an acceptable degree of improvement for the lowest difficulty puzzles.

Improvements were later implemented to compare the solution score to the score prior to collapsing, as that is an accurate representation of the original alignment. Furthermore, the minimum improvement of three had a decreasing impact as difficulty increased, and average score improved. There is a high degree of correlation between number of columns, rows, and average solution score. Thus, we introduced an incremental minimum score improvement, which could raise the minimum acceptance score by a set value, typically 0.5, for every column or row added per difficulty.

2.5.2 Minimum Moves

While minimum score ensured puzzles provided solutions with a greater score than their predecessor, a minimum number of moves was also implemented to ensure puzzles remained engaging to the players. If the puzzle was able to surpass the prior alignment score with a single move, it would have been a rather short lived and boring experience for the player,

and the puzzle itself would not provide much information for realignment. By implementing a minimum number of moves, the puzzles remained engaging, and opened up the possibility of various good solutions.

2.5.3 Variety

To build upon the minimum moves requirement, a minimum variety was added to ensure a certain number of subsequences were used, rather than only adding gap bricks to a single column. This assured more of the puzzle information will be relevant, rather than a single subsequence, and further instilled player engagement.

2.5.4 Puzzle Completion

If a puzzle successfully passed all the defined acceptance parameters, then it was added to a collection of puzzles. This batch of puzzles was then uploaded to Borderlands Science, where players could play and submit solutions.

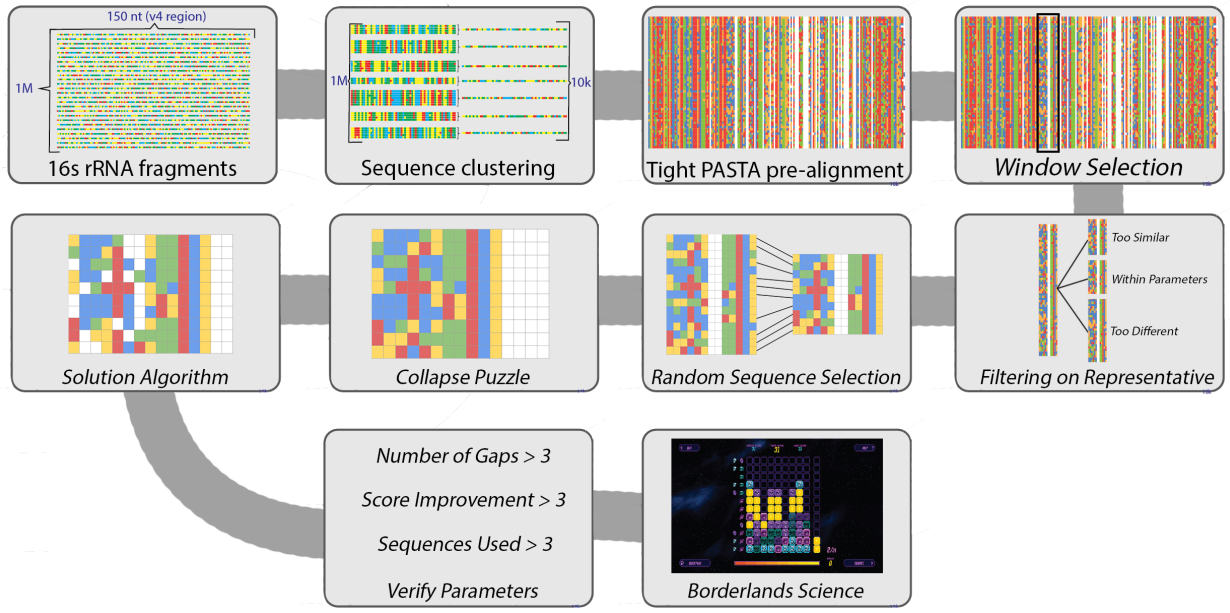


Figure 2.6: The complete puzzle generation pipeline from the initial set of sequences, to a valid puzzle uploaded to Borderlands Science.

Chapter 3

Realignment

3.1 Processing Player Solutions

Once puzzles were present in Borderlands Science for an adequate amount of time, we could retrieve player solutions. These solutions allowed us to observe the differences in perspective between humans and algorithms, and attempt to improve upon the alignment. Processing these millions of solutions required several steps to both filter outliers, and ensure player data remained the focus. This chapter will discuss in depth the methods and data structures used for realignment.

3.2 Calculating Pareto Distance

A players goal is to maximize the score in a puzzle, yet our objective is obtaining a player consensus, rather than optimization. A high score is not the only aspect observed, and one key factor not included in the puzzle design is penalizing gaps. The standard objective in sequence alignment is to align sequences as well as possible with minimal gaps, thus we produced a pareto front. The pareto front is a multi-objective optimization, considering both score and reduced gaps which cannot be directly compared. Each solution could be evaluated based on its distance from the pareto front. While a players solution may not score as high as our algorithm, it could still be considered a viable option. This not only provided a good comparison between the players and our algorithm solutions, but acted as a filter prior to realignment, allowing us to identify and remove outlier solutions.

3.3 Data Gathering

Once pareto distance outliers were excluded, the solutions were consolidated into a single dataset. Each individual solution was processed, and recorded the final positions of each nucleotide which acted as a voting system indicating the player consensus. Once all solutions had been processed, the data was then normalized to account for variance in coverage. Regions lacking player solutions were populated with PASTA or MUSCLE information.

3.4 Aligning Sequences

3.4.1 Pairwise Alignment

The dataset produced from player solutions provided a player consensus for each sequence. Using this consensus, we could perform a pairwise alignment using a modified Needleman-Wunsch scoring scheme as shown in Figure 3.1. We aligned each sequence to the array of player assessments, optimizing the following function:

$$D(i,j) = max \begin{cases} D(i-1,j-1) + consensus(s_i, v_j) \\ D(i-1,j) + consensus(-, v_j) \\ D(i,j-1) \end{cases}$$

where i and j are indexes in sequence s, and consensus is a vector of dictionaries of player votes which established a consensus per sequence and column.

3.4.2 Realignment Representatives

While the Needleman-Wunsch approach is optimal, it can result in several viable alignments without any clear indication of which is best suited for our case. In order to circumvent this, we produced a list of every viable alignment, and constructed a levenshtein distance matrix [20], where each position [x, y] indicates the distance between the alignment in row

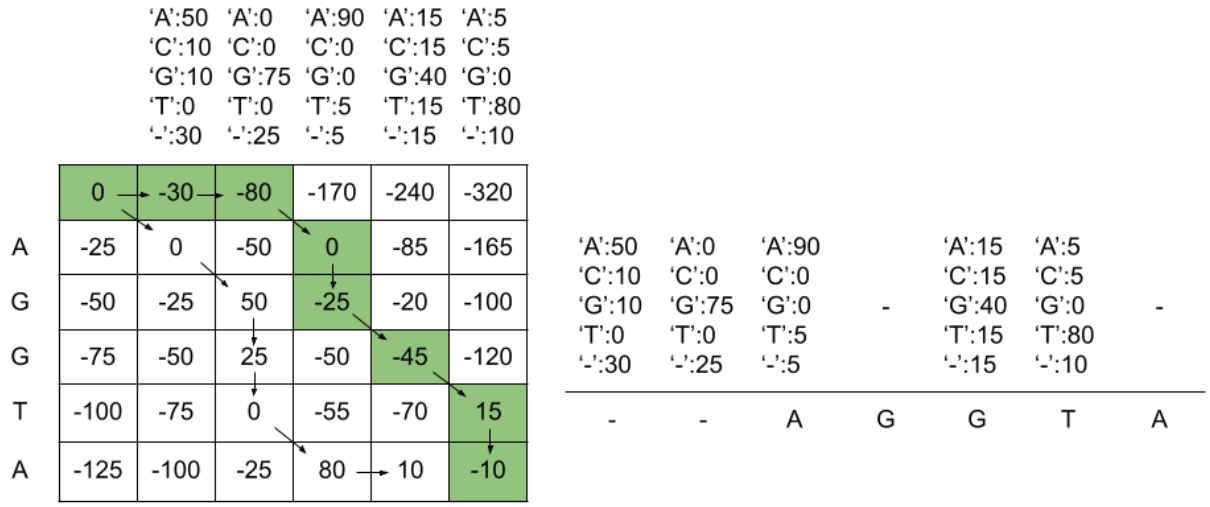


Figure 3.1: A Needleman-Wunsch pairwise alignment

x, and its counterpart in column y. Once this matrix had been assembled, we could take the sum of each row. A low levenshtein distance indicates that two alignments are very similar, thus the alignment with the lowest matrix row sum acted as a representative which most resembles all other possible alignments. This process was repeated for each individual sequence.

3.4.3 Optimizing Sum-of-Pairs

Despite the further refinement by choosing representatives, we confronted cases where there are multiple viable representatives. If there are only two new alignments, they will share the same sum levenshtein distance. Our final method for selecting alignments was optimizing sum-of-pairs, which is the sum of the alignment scores for each pair of sequences in the MSA.

Taking a random representative from each sequence, we constructed a new base alignment set, and calculated the sum-of-pairs score. Proceeding this, a random sequence with more than one valid option replaced its current representative with another. Sum-of-pairs was calculated again, and if the score improved, the new representative was kept. This was done for each sequence, and can be seen in Figure 3.2. The full process was repeated five times to consider iterations previously missed due to a sequential process. Ideally, we would consider every possible combination of representatives for every sequence, yet if only a third of the sequences had a single alternative representative, that would yield 2^{3333} combinations, which simply is not feasible. Despite the seemingly random approach, this method proved to be very effective at getting consistent results.

3.4.4 Normalizing Sequence Lengths

Due to the nature of the individual pairwise alignments, we were left with a set of sequences with different lengths. Considering the objective of a MSA is for all sequences to have the same length, we implemented a normalization method which inserted the minimum gaps necessary into each sequence. This approach ensured we had a valid multiple sequence alignment, as well as correcting sections of a sequence which differed slightly in comparison to its neighbors.

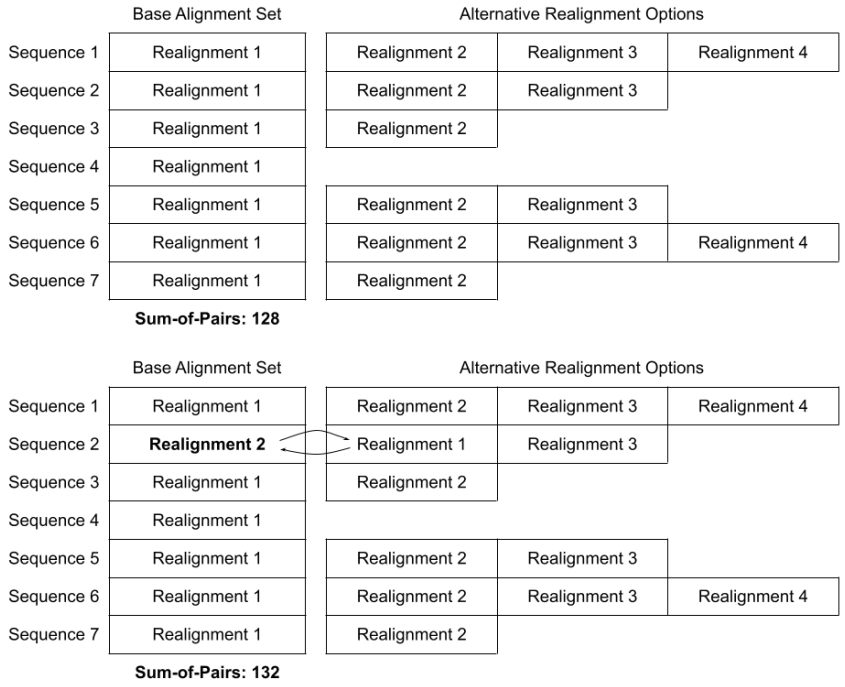


Figure 3.2: Calculating the sum-of-pairs using an initial set of representative alignments, then attempting to substitute alternative alignments to improve upon sum-of-pairs.

Progressive Alignment

We introduced a progressive alignment to ensure gaps are placed in the ideal positions. The progressive alignment finds the two most similar sequences and aligns them, to ensure they are the same length. Once they are aligned, they create a profile which will be expanded on below. This process repeats until only a single profile remains. Decisions are made using a combination of standard levenshtein distance, and an alternative levenshtein distance designed for profiles.

Initial Distances

The distance between every sequence is calculated using a standard levenshtein distance to construct a levenshtein distance matrix. We also experimented with hamming distance [21] with mixed results. The smallest score in the matrix reveals the two closest sequences, which are then aligned. If they are the same length, they are simply grouped together, if not, the shorter sequence is subjected to a profile alignment. Once the two sequences are the same length, they form a profile. The newly formed profile then calculates its distance from every other sequence. If neither sequence was changed, then the distance from the profile to another sequence is the average distance of the profile sequences. However, if a sequence has been adjusted, then distance must be recalculated.

Profile Levenshtein Distance

Typically, levenshtein measures the distance between two strings, in this case, sequences. If two characters are identical they are worth 0, otherwise 1, yet a profile contains two or more sequences which introduces more variables. Our modified levenshtein calculation utilizes a profile alignment, attempting to optimize the following function:

$$D(i,j) = min \begin{cases} D(i-1,j-1) + (1 - \alpha_i \cdot \beta_j) \\ \\ D(i-1,j) + 1 \end{cases}$$

where α_i and β_j are the normalized vectors containing the count of nucleotides for the given column. A single sequence may have the nucleotide 'A' for a given column, while a profile may have three nucleotides ('A', 'C', 'C'). Thus, the distance for the individual column would be

$$1 - (1 * 0.33 + 0 * 0.66) = 1 - 0.33 = 0.66$$

Merging Profiles

Now that we have established a method for calculating the levenshtein distance between two profiles, we calculate the distance to every remaining sequence. In this circumstance, sequences act as a profile containing a single sequence. We remove the two sequences used to create the profile from the levenshtein distance matrix, and insert a single new instance for the profile. The process is then repeated, selecting the pair with the lowest levenshtein distance, and combining them to create a profile. For each iteration the distance matrix is reduced by one, gradually grouping sequences based on similarity. Eventually profiles will be merged together. Should a profile be subjected to a profile alignment, adding a gap to a column adds a gap to every sequence in the profile. After several iterations, the final result is a single profile containing all sequences at the same length. The full progressive alignment operation is illustrated in Figure 3.3

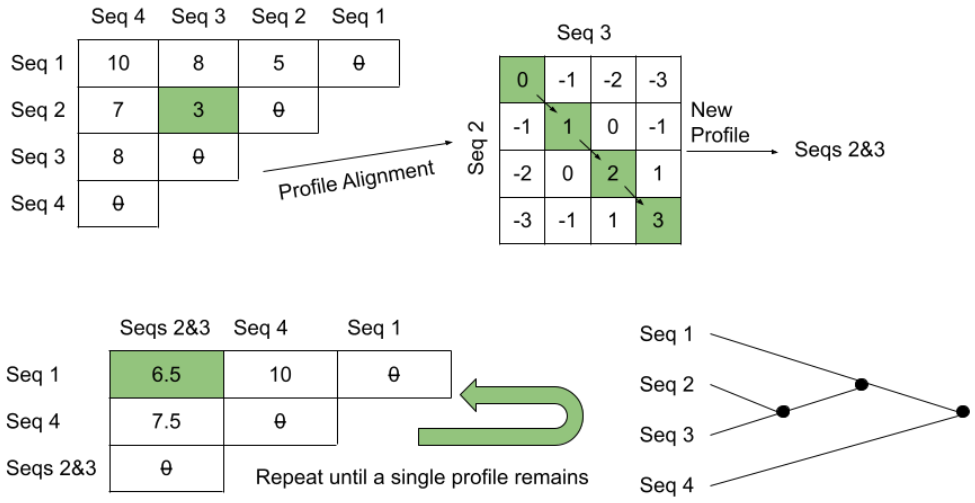


Figure 3.3: The progressive alignment workflow

3.4.5 Post-Processing

Once our realignment process produced a new set of alignments with equal length, we subjected our new set to a post-processing method. This algorithm calculated the normalized distribution of nucleotides per column to construct a consensus. Each individual sequence was then stripped of all gaps, and underwent a profile alignment to the consensus which assisted synchronizing sequences and aligning gap regions. This process cycled four times, as we saw results start to stabilize after that. This post-processing could also be performed on an alignment with no player solutions, which improved upon PASTA on every observed metric, and yielded a new objective for us to surpass with player solutions.

3.4.6 Width Reduction

With our final realignment established, we performed a quick width-reduction, which inspected for completely empty columns and removed them. Future iterations may also look at pairs of columns with complementary gaps which could be merged together, providing a tighter alignment. This may be an interesting feature to explore when using MUSCLE as the base given that MUSCLE produces a more sparse alignment. At the moment, most puzzles use PASTA, resulting in very dense alignments, therefore this feature has little impact.

3.5 Phylogenetic Tree

The realignment process successfully utilized player solutions to adjust regions of an existing PASTA alignment, resulting in a new MSA. These changes aim to provide more insight on the phylogenetic evolution of these microbial genomes. In order to evaluate if Borderlands Science had a positive impact, we constructed a phylogenetic tree using FastTree [22], and rerooted to divide archaea and bacteria. This tree depicts the estimated evolutionary descent of this collection of microbes, and acted as our principle method for evaluation.

Chapter 4

Evaluation

4.1 Parameters and Approaches

Throughout our puzzle generation and realignment discussions we have covered numerous parameters which can significantly impact the output in various ways. The focus of this thesis was optimizing puzzle generation to refine selection methods, thus we experimented with various puzzle generation parameters, while narrowing the realignment parameters to ensure each set of puzzles was subjected to the same conditions. This chapter will cover the various parameters that were considered, as well as the methods implemented to evaluate both the player solutions and final realignments.

4.2 Puzzle Parameters

There are many different approaches that can be taken when generating puzzles. In order to evaluate the effectiveness of certain parameters, we constructed several batches of puzzles, each with slightly alternating parameters for comparison. Rigorous testing showed that using puzzles in both directions further improved the results, thus all batches produced puzzles in either direction. Most puzzles used PASTA as the base alignment, and most batches were restricted to a smaller region of 40 columns for a controlled environment.

4.2.1 Solution Algorithms

Puzzles could alternate between using the greedy solution algorithm, or the profile alignment algorithm. Both having their respective benefits, we constructed batches which use one or the other. Likewise, we generated batches which required both algorithms surpassing the acceptance parameters. We also included older puzzles which utilized the handicapped greedy algorithm to act as a point of reference for our changes.

4.2.2 Soft and Hard Windows

Both hard and soft windows have their advantages. We constructed batches of puzzles with both parameters. Puzzles which used MUSCLE as the base were also included, given that soft windows were initially designed with MUSCLE in mind.

4.2.3 Acceptance Parameters

For the purposes of our tests, the minimum score improvement, the minimum number of moves, and minimum column variety were all set to three. These simple restrictions required puzzles to perform adequately, while ensuring they retained a certain degree of challenge and difficulty for players.

One of our many batches of puzzles implemented our ability to change the minimum score parameter based on difficulty. The minimum score requirement started at three for difficulty one, and would raise by 0.5 for every row or column added.

4.2.4 Scoring Scheme

Our scoring scheme is flexible, so we took the opportunity to generate one batch which utilized the scoring from Borderlands Science's predecessor Phylo. Implementing a stricter scoring required using the profile alignment algorithm, as the greedy algorithm can not comprehend the concept of an initial harsher penalty for gap opening, and a reduced penalty for a gap extension. This method generated no puzzles for difficulties one and two due to their size, and being unable to overcome the costly penalties for mismatches and gaps. We dismissed the 40 column constraint, using the entirety of the PASTA alignment due to a lack of puzzles in the original region.

4.2.5 MUSCLE

Finally, we generated two batches which used MUSCLE as the base alignment rather than PASTA. Due to a far less dense alignment, there is no clear translation for the PASTA 40 column restraint, thus we utilized the full alignment. This resulted in a small degree of bias towards MUSCLE, as the puzzle generation is better able to search for regions inaccessible to the PASTA batches.

4.3 Realignment Parameters

Despite all the effort made to hone the realignment process, there are several factors remaining that can have a significant impact on the final outcome. We performed several experiments utilizing different parameters in order to determine the approach which would most likely return promising results.

4.3.1 Columns

One of the primary parameters investigated was the number of rows to use from the player solutions. Initial observations seemed to indicate that players targeted the bottom half of the puzzle, while the top half of the puzzle saw little more than the impact from lower rows. This resulted in a more detailed bottom half and a top half with more noise than actual beneficiary data. For that reason, early tests restricted data gathered to only the first 50%

of rows per player solution. Running additional tests which added or removed a row saw varied inconclusive results, indicating that the number of rows does not have a reliable nor predictable effect. For these reasons we produced realignments using both 50% of the rows, as well as 100% of the rows.

4.3.2 Pareto Distance

Prior to any realignment processing, we calculated the pareto front for each puzzle, and filtered solutions based on their distance to the pareto front. For our primary investigations we excluded all solutions with a distance greater than 1.5. This filter removed approximately 35% of the solutions which were considered the extreme outliers. Further experiments looked at filtering using distances ranging from 0.1 to 2.0. These tests were unfortunately inconclusive, often showing some bias towards more restrictive parameters, yet too much filtering caused a lack of solutions leading to inconsistent results that were difficult to reproduce. For the sake of these tests, we concluded that a pareto distance of 1.5 remained an acceptable restriction which gave the greatest odds of success without removing too many solutions.

Once an optimal distance was established, we could then evaluate the calculation used to determine distance to the pareto front. Initial testing used an euclidian distance, finding the shortest distance possible. While this approach is logical, it implies that a single point

in the score is equivalent to a single gap, when in reality the score and the number of gaps scale at different rates. With this reasoning, we included a secondary distance calculation. As shown in Figure 4.1, given a graph where x is score increasing, while y is number of gaps decreasing, we can prioritize the score by calculating the horizontal distance to the pareto front, rather than the euclidian distance.

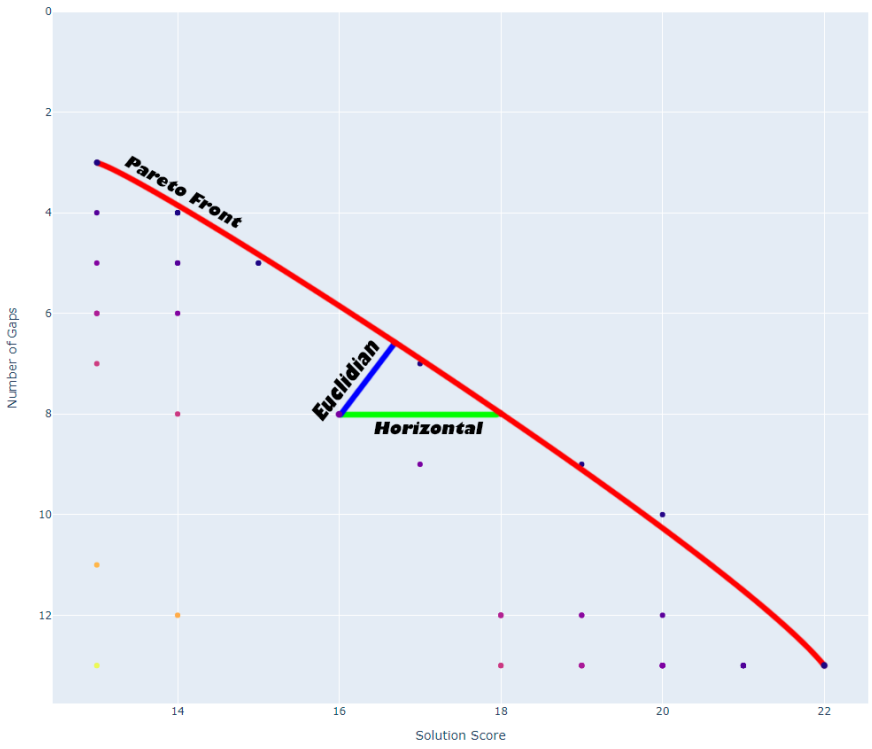


Figure 4.1: The pareto front constructed from player solutions

4.4 Evaluating Solutions

4.4.1 Comparing Pareto Distance

In order to justify a citizen science approach, we compared the pareto distance of player solutions to the algorithms used to develop these puzzles. Given more diverse solutions, we rivaled the implemented algorithms, as well as collected a larger set of varied data close to the pareto front.

4.5 Evaluating Realignments

4.5.1 Greengenes

In order to properly evaluate our realignment, we constructed a phylogeny tree which can be compared to a reference tree. This reference tree was built by placing all the sequences into Greengenes [23] 13.5 phylogenetic tree using SEPP [24], then removing all other tips, and rerooting the tree for archaea. SEPP provided us with the most accurate estimation of what the phylogeny tree should look like given exterior knowledge. Once we obtained a point of reference, we investigated using two different methods for comparison.

4.5.2 Kendall-Colijn Distance

Kendall-Colijn (KC) is a metric-based method for comparing trees which extracts distinct alternative evolutionary relationships embedded in data [25]. As Figure 4.2 shows, this method takes two rooted trees with identical sets of tip labels, and compares the placement of the most recent common ancestor of each pair of tips. This placement is calculated using a combination of both distance from the root, as well as the number of edges between the most recent common ancestor and the root. Unfortunately, at the time of this project, the KC software only supports trees with up to 400 tips, therefore we took 400 random tips from both phylogenetic trees, removed all others, and calculated the KC distance. This process was repeated 100 times to obtain an average with the objective to get a distance close to zero.

4.5.3 Triplet Distance

Demonstrated in Figure 4.3, the rooted triplet distance calculates the structural dissimilarity of two phylogenetic trees by counting the number of rooted phylogenetic trees with exactly three leaves that occur as embedded subtrees in one, but not both of them [26]. Similar to KC, the triplet distance software is not designed to handle a tree with 9667 leaves. It is also far more complex, therefore we randomly took 100 tips from both phylogeny trees, removed the others, and used the resulting smaller tree for our triplet calculation. This was performed five times to obtain an average score with the goal of minimizing the distance.

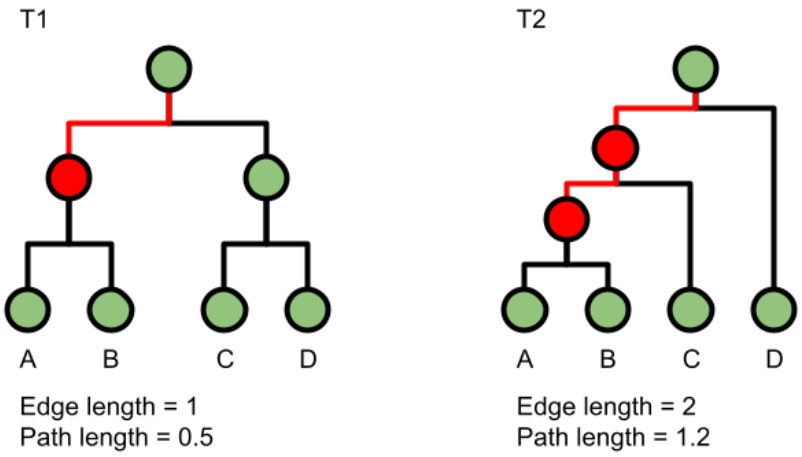


Figure 4.2: Kendall-Colijn comparison example between two trees

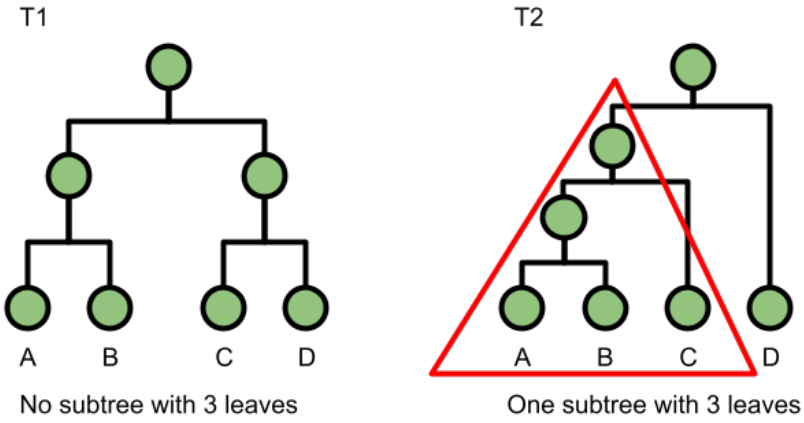


Figure 4.3: Triplet comparison example between two trees

Chapter 5

Results

Now that the various parameters have been established, we can discuss the batches of puzzle produced, the impacts of our choices, and what was successful. To begin, we developed eleven different batches of puzzles for comparison. Most batches were restricted to a region of 40 columns. Batch 5 had no limitation on columns due to a more restrictive scoring scheme severely limiting the number of puzzles. The 40 column restriction for PASTA could not be properly transposed onto the MUSCLE alignment, thus batches 9 and 10 also utilized the full alignment. The batches are detailed in table 5.1.

This chapter will cover our results and observations of player solutions. We will look at player performance in comparison to the solution algorithms, as well as the final realignments produced using player data. We will note how the differences between batches impacted player contribution, and what conclusions can be drawn from these experiments.

Batch	Solution Algorithm	Window	Base File	Total Puzzles	Notes
0	Handicapped Greedy	Hard	PASTA	143,000	
1	Greedy	Hard	PASTA	10,000	
2	Greedy	Hard	PASTA	3,700	Incremental score*
3	Greedy	Soft	PASTA	2,800	
4	Profile	Hard	PASTA	13,000	
5	Profile	Hard	PASTA	1,400	Phylo scoring scheme
6	Profile	Soft	PASTA	1,600	
7	Greedy and Profile	Hard	PASTA	8,000	
8	Greedy and Profile	Soft	PASTA	1,200	
9	Greedy and Profile	Hard	MUSCLE	13,000	
10	Greedy and Profile	Soft	MUSCLE	15,000	

Table 5.1: Puzzle generation batches. Incremental score indicates that the minimum score acceptance parameter increased based on puzzle difficulty.

5.1 Solution Results

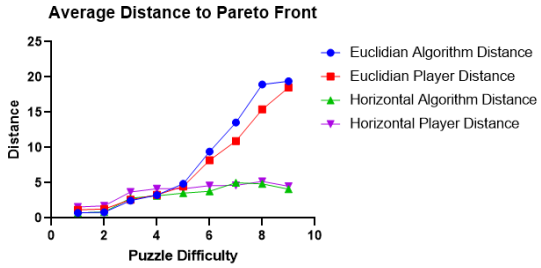
5.1.1 Pareto Distance

Comparing the average pareto distance per difficulty in Figure 5.1a, we saw similar trends between the player solutions and the solution algorithms used to produce the puzzles. Despite the solution algorithms only contributing a single solution per puzzle, while players produced numerous solutions, we discovered a strong correlation using both the euclidian distance, and horizontal distance. This may seem to indicate that player solutions are no better than algorithms, and citizen science is not needed, but this perspective is too narrow. The score is used to drive players toward good solutions, but our true objective is obtaining a player consensus. From that viewpoint players provide a larger service based on quantity rather

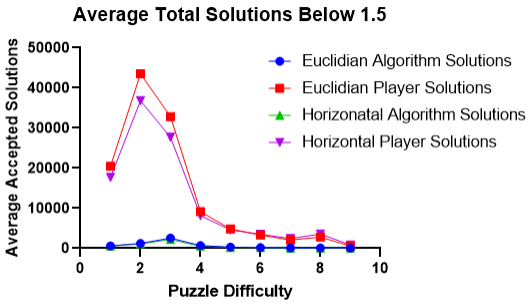
than quality.

5.1.2 Viable Solutions

As shown in Figure 5.1b, players produced a larger set of solutions with greater diversity, contributing to a clearer picture into how the realignment ought to be processed. Regardless of the average pareto distance, our filters excluded any solutions whose pareto distance exceeds 1.5. Therefore, the larger question is the number of solutions which fall under that parameter. The solution algorithms were able to produce a few thousand solutions per batch which meet these requirements. The players on the other hand, were able to create 10-36 times more valid solutions, far exceeding the capabilities of our software.



(a) The average distance to the pareto front for each difficulty.



(b) The average number of solutions with a pareto distance lower than 1.5 for each difficulty.

Figure 5.1: Comparing players solutions and puzzle solution algorithms using both euclidian distance and horizontal distance. All puzzle solutions in all batches calculate their respective distance to the pareto front using both euclidian and horizontal distance. Solutions are divided based on puzzle difficulty, and the average per difficulty is displayed.

5.2 Realignment Results

Each batch of puzzles produced four alternative realignments based on two parameters. The first parameter was the number of rows used per solution to construct the dataset, alternating between the bottom 50% of the rows, and 100% of the rows. The second parameter was the calculations used to evaluate the distance to the pareto front. Realignments used either an euclidian distance, or a horizontal distance. Given four different realignments per batch, we achieved a greater sense of assurance that conclusions drawn are attributed to the batch parameters, rather than an over refined realignment suitable to certain conditions.

When covering realignment results, it is important to preemptively discuss the values produced from the KC, and Triplet distance metrics. KC is calculated using 400 random tips, and outputs a float value indicating the estimated distance between two trees. Two identical trees will result in an output of 0. Throughout our numerous tests, KC produced varying scores averaging around 1,600, with our best results resting at 1,100. Triplet distance also desires to be as low as possible. Due to a difference in approach, Triplet distance values cannot be compared to KC. Triplet distance scores averaged around 78,000, while our best score remains 51,000.

5.2.1 Handicapped Greedy, Revised Greedy, and Profile

Early stages of Borderlands Science were developed using hard windows, and the handicapped greedy solution algorithm. As a result, we only compared the early stages to our hard window batches. Comparisons can be seen in figure A.2. Despite the revised greedy and profile solution algorithms designed to be major improvements upon the handicapped greedy, we actually see the contrary. The handicapped greedy puzzles obtained an average KC distance of 1682.95 and a Triplet distance of 73749.45. Compared to these results, our revised greedy algorithm performed slightly worse, with an average KC of 1747.01, and Triplet of 78336.1. The profile alignment algorithm is noticeably worse, obtaining an average KC of 1814.28 and a Triplet of 85991.3. This poor performance is perhaps due to the handicapped greedy solution algorithm having far less restrictive acceptance parameters. The handicapped greedy algorithm was able to achieve a greater coverage of the alignment through sheer quantity of puzzles and solutions, allowing it to possibly overcome its shortcomings. Due to the sensitivity of realignment, attempting to take control models with a randomized equal number of puzzles per batch caused a significant amount of variability, making it difficult to compare batches in that setting.

5.2.2 Greedy and Profile

Inspecting our greedy and profile puzzle batches utilizing both hard windows and soft windows found in figure A.3, we are able to make some interesting observations. While

greedy puzzles tended to yield better results in a hard window environment, profile alignment puzzles produced better results in a soft window environment. Hard windows obtained puzzles from very dense regions of nucleotides, leading to denser subsequences, and single gap insertions having a greater impact on proceeding nucleotides. As a result, they required fewer consecutive gaps to see an improvement. This led to the greedy algorithm not suffering from its usual drawbacks, and puzzles which contained more nucleotides which were beneficial for obtaining row bonuses. Thus we can predict that the greedy algorithm is better suited for finding puzzles in a hard window environment.

Soft windows can construct puzzles using collective sparse regions of nucleotides. Due to the nature of the soft window, it contained far more sporadic and consecutive gaps prior to collapsing the puzzle. While the greedy algorithm still appeared to perform quite well, this difference in approach likely allowed the profile alignment algorithm to excel. The profile alignment was able to discover several valid puzzles the greedy algorithm would otherwise overlook by inserting multiple consecutive gaps.

5.2.3 Combined Greedy and Profile

While different algorithms seem to have varying performance based on the window type, utilizing both simultaneously appeared remarkably effective. Requiring a puzzle to pass the given acceptance parameters using both solution algorithms reduced the total number of

puzzles, yet the resulting metrics were nearly as good, if not better than the individual algorithms. We theorize that by ensuring a puzzle could succeed using two different algorithms, players were more likely to obtain more diverse valid solutions. Varying solutions per puzzle allowed our realignment to produce more accurate pareto fronts, which further refined solution filtering. These results can be seen in figure A.4.

5.2.4 Hard Windows and Soft Windows

Analyzing our various batches using either hard windows or soft windows, we observed that according to figure A.5, soft windows tended to yield better results. Not only did the soft window produce more consistent metrics, but they also resulted in some of the best metrics across all our tests. This included a KC distance of 1107.77, and a triplet distance of 57915.8, both of which have been very difficult values to obtain throughout the Borderlands Science project. We suspect that these promising outcomes are due to the soft windows ability to obtain better coverage of the alignment, compared to the hard windows restriction to dense nucleotide regions.

5.2.5 Acceptance Parameter Comparison

Despite adjusting the minimum score parameter to be incremental based on puzzle difficulty, a set of only higher scoring puzzles did not seem to have a significant impact, as seen in figure A.6. It would appear that this method of filtering allowed us to remove less beneficial

puzzles without harming our realignment results. Although we did not see an improvement in metrics, we were still able to replicate similar results using roughly a third of the puzzles, which proves promising.

5.2.6 Scoring Scheme Comparison

While changing the acceptance parameters did not significantly alter the results, modifying the scoring scheme showed far more promise. Batch five used the Phylo scoring scheme for discovering puzzles, but players continued to use their regular scoring scheme for their solutions. Regardless of this limitation, the Phylo scoring scheme was able to discover a much smaller, but refined set of puzzles that played a far greater impact on the realignment. These results may appear somewhat biased due to batch four being restricted to only 40 columns, while the Phylo batch had access to the entirety of the PASTA alignment, yet the Phylo batch produced only 10% of the puzzles in comparison, and obtained far better metrics. This helps establish that more meticulous observation for puzzles is more beneficial than quantity and overall coverage. Details can be found in figure A.7

5.2.7 PASTA and MUSCLE

We chose PASTA as the base alignment for producing puzzles due to its compact nature which made it easy to find dense regions of nucleotides ideal for puzzles. In order to further justify this reasoning we produced two batches that use MUSCLE as the base alignment.

The soft window was developed with MUSCLE in mind, to handle a more sparse alignment. Despite our efforts, we saw a dramatic decrease in metrics compared to the equivalent PASTA batches with KC scores averaging around 1,850 and Triplet scores closer to 80,000. MUSCLE had no restrictions on columns, so it had every opportunity to find the ideal regions to optimize. Regardless, it could not produce better results. Even though MUSCLE was capable of producing numerous puzzles, the sparse alignment made it difficult to target influential regions due to its low density. We can see in figure A.8 that soft windows did perform better than hard windows and were able to make notable improvements, but they could not rival the results obtained from PASTA.

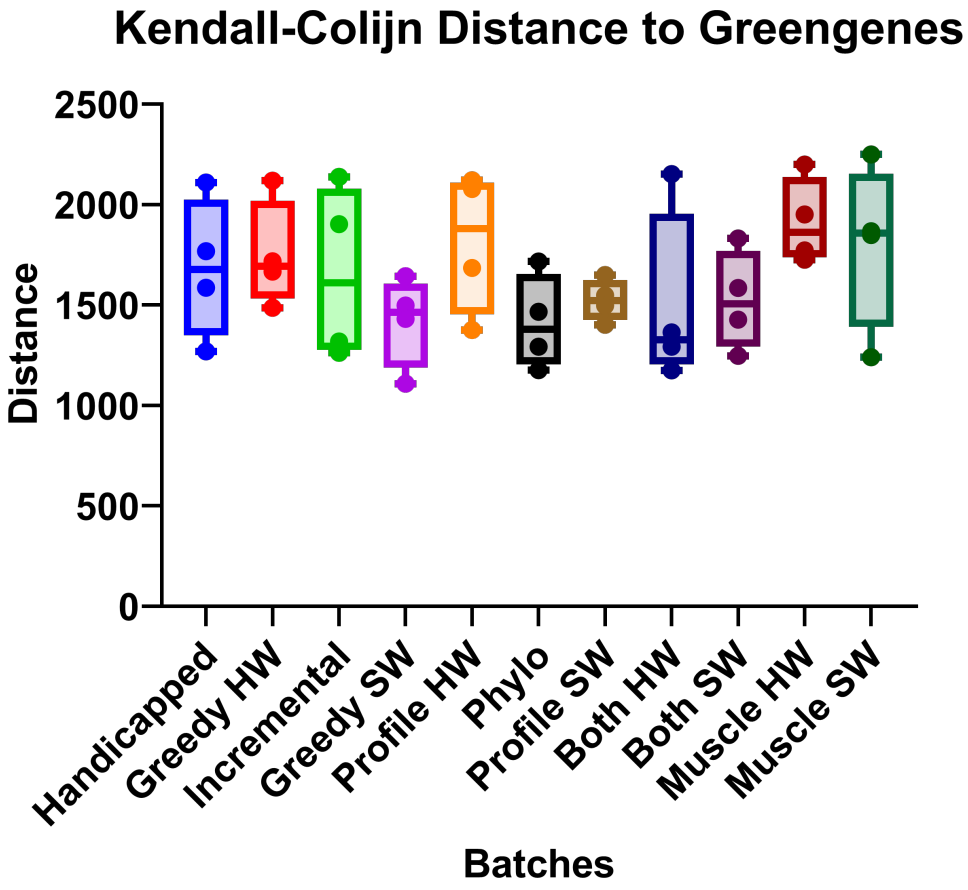


Figure 5.2: Comparison of Kendall-Colijn distances between all batches seen in table 5.1 where distance is the KC calculated difference between the BLS phylogenetic tree, and the Greengenes reference tree, with the objective to be as low as possible.

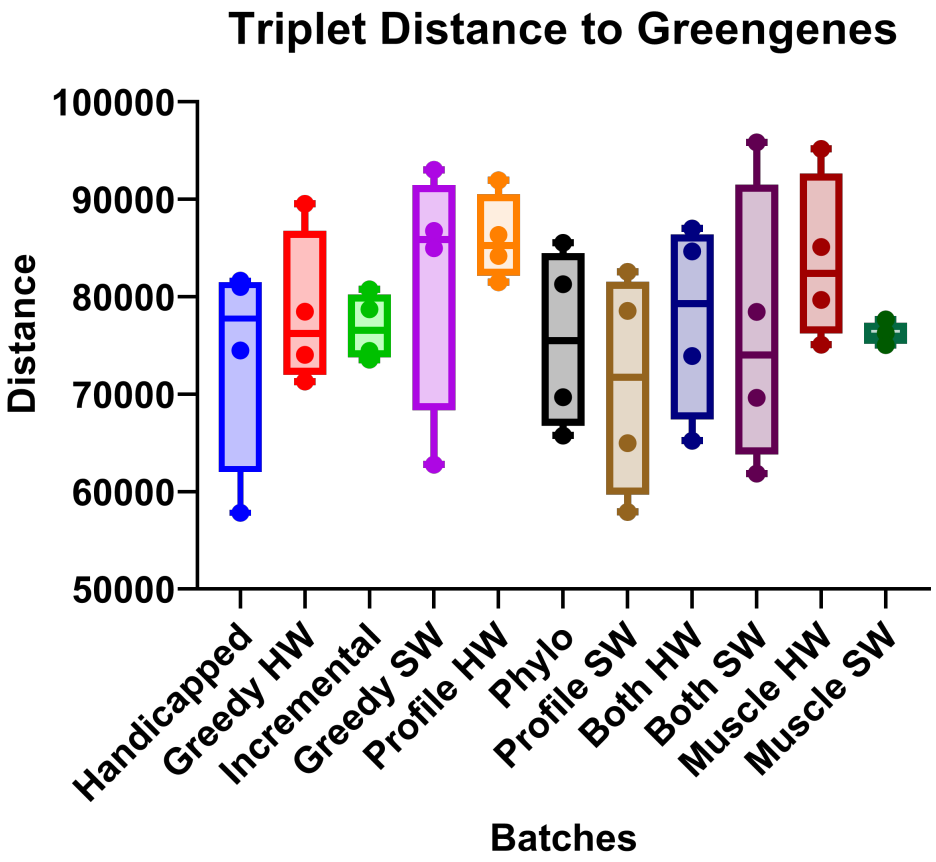


Figure 5.3: Comparison of Triplet distances between all batches seen in table 5.1 where distance is the Triplet calculated difference between the BLS phylogenetic tree, and the Greengenes reference tree, with the objective to be as low as possible.

Chapter 6

Discussion

6.1 Realignment Experiments

MSA is a very complicated process with different valid solutions based on desired output. Likewise, there are a multitude of methods to obtain these results. The final implementation of the realignment process is the result of numerous different experiments and strategies.

6.1.1 Offsets

Early in the development of Borderlands Science, we developed a method to construct puzzles with an offset. This change would cause the guide distributed to the players to be offset by one or two rows. Given an offset one puzzle, placing a gap on the bottom row of each column would result in the equivalent starting point of an identical offset zero puzzle.

An offset would cause a player to inadvertently contribute by withholding a gap brick. The offset also added more variety, resulting in far more offset one puzzles being generated.

Despite our best efforts, attempting to utilize these puzzles proved difficult. An additional set of solutions simply acted as more data, and failed to change our metrics in a significant way. Additional experiments attempted to exclude the first row of the offset one solutions, as they were primarily gaps, but to this point the offset data remains unreliable. We did generate offset one and offset two puzzles for all of our batches, but they were not used in the final results.

Future work will investigate the changes in the realignments which contributed to the greatest improvement in our metrics. Once a more clear reasoning can be established, offsets will be revisited to determine the optimal method to utilize these puzzles.

6.1.2 Propagation

When constructing the dataset using player solutions, one interesting aspect we explored was propagating results. Each sequence in a puzzle would search for other sequences which contained an identical subsequence within the same region. The results from the subsequence could then be propagated to other sequences in order for solutions to cover a larger portion of the alignment. Although seemingly logical, this likely caused some mediocre solutions

on common sequences to become abundant. Results varied greatly, though propagating while using 100% of the rows seemed more effective than only 50%, likely due to less, but more accurate propagation. This inconsistency, along with a significantly increased runtime, caused propagation to be withheld until a later time.

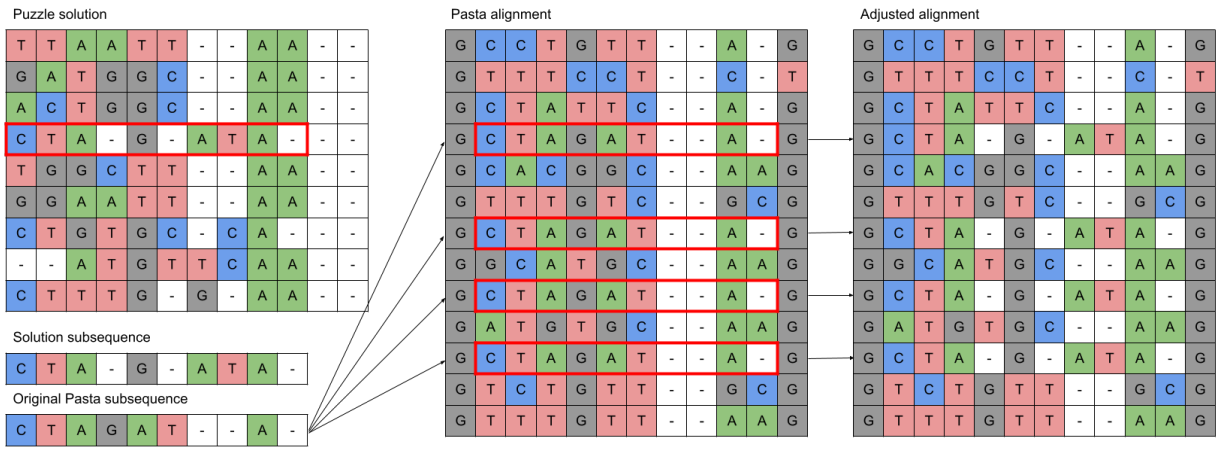


Figure 6.1: Propagating puzzle solutions to identical subsequences

6.1.3 Rfam

The Rfam database is a collection of RNA families, each represented by MSA, consensus secondary structures, and covariance models [27]. Using this database, we were able to construct secondary structure predictions for our individual sequences. The goal was to allow the secondary structure to have a small influence on the realignment process to assist and guide the player data. Unfortunately, results were inconclusive. We often saw our KC metrics decline, while the Triplet metrics varied wildly. Therefore, they did not play a part

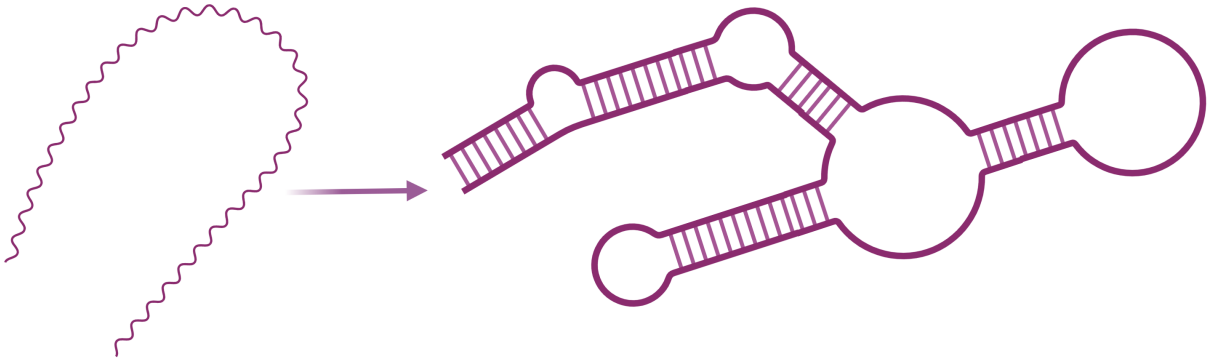


Figure 6.2: RNA secondary structure created with BioRender.com

in our our final calculations. Future tests may revisit Rfam to experiment incorporating it in different ways.

6.1.4 Flexibility

When realigning individual sequences, there may be multiple valid alignments. To solve this, we created a levenshtein distance matrix to choose the representative alignments which most resembled the others. While experimenting we included a flexibility option to allow some leniency when choosing representatives. Setting a flexibility of five, the optimal sum levenshtein distance was recorded, and any alignment with a sum levenshtein distance within five of the optimal was also accepted as a representative. This addition allowed for more representatives to be considered, and increased the odds of finding an optimal new alignment. Unfortunately, tests became unpredictable. A low flexibility value often saw worse results, while a higher flexibility occasionally saw better outcomes, yet the number of additional

options caused a great deal of randomness, and became difficult to reproduce. The flexibility was left at zero for our experiments, but there are future plans to revisit the sum-of-pairs optimization stage to make better use of this implementation, and attempt to make the promising high flexibility results reproducible.

6.1.5 Minimizing Levenshtein Distance

After we completed gathering representatives for each sequence, we took a single representative per sequence to create a new set, and attempted to maximize the sum-of-pairs score. Rather than sum-of-pairs, our initial experiments attempted to create a second levenshtein distance matrix using the new set, and attempted to minimize the global sum distance. These attempts originally seemed promising, yet proved to be very inconsistent, while the sum-of-pairs could consistently produce the same result in most cases. Sum-of-pairs was eventually accepted as the primary method.

Longest Sequence Normalization

Originally, we corrected the varying sequence lengths using a growing profile alignment. The algorithm would use the longest sequences as the starting profile. The next longest sequences were subjected to a profile alignment, and were then added to the profile. This process would repeat until all sequences were the same length. While a quick and effective method, we realized this caused an unhealthy bias towards the longest sequence, causing

two similar sequences to possibly align poorly based on what was already dictated by the profile. This method was eventually replaced with the progressive alignment.

6.1.6 Additional Metrics

Mantel Test

The Mantel test, named after Nathan Mantel [28], calculates the correlation between two matrices. In our case, each phylogeny tree is used to construct a distance matrix which contains the distances from each leaf to every other leaf. The correlation can then be computed using Pearson's product-moment correlation coefficient.

This was the primary metric for a majority of early stages in the Borderlands Science project. Sadly, it proved to be far less reliable than KC and Triplet. Various attempts to maximize the Mantel score proved inefficient, and the results often clashed with KC and Triplet. Additional tests showed that Mantel was the least indicative of an optimal MSA, thus it was ultimately dropped as a metric.

Regional Sum-of-Pairs

In addition to many well established metrics, we attempted to develop our own. Regional sum-of-pairs took windows of sequences and columns from the original alignment file, and attempted to find the closest equivalent window in the final realignment. This was accomplished by taking the start and end columns of the window, inspecting the new

realignment, and finding new columns which incorporated as much of the original window as possible while minimizing additional nucleotides. Each window would then be subjected to a sum-of-pairs calculation, determined the difference in score, and repeated. The end result was the average difference in windows between the original alignment and the new realignment. The goal of this new metric was not only to determine the quality of a new realignment, but to investigate regions which saw a significant increase or decrease in score. The capability to analyze influential regions permits a deeper investigation into what caused these changes, and if the alignment could be adjusted to optimize these observations.

Unfortunately this approach still requires a lot of refinement. Many significant changes in score are often due to poorly associated windows, and minor changes to the window size can cause a massive difference in score. Regrettably, this metric has not been included, but will play a primary role in future work identifying the direct causes for strong KC and Triplet scores.

6.2 Limitations

The results that we have discussed do show promise. Soft windows and more rigorous scoring schemes demonstrated that smarter puzzles can lead to better results using less data. Although these results are encouraging, they are extremely susceptible to the type of

realignment process used. Given a staggering amount of approaches to utilizing player solutions, we believe that our methods are well refined and each step can be logically justified. We are not certain how these puzzles may behave in a completely unique environment, but considering we produced multiple varied realignments per puzzle batch, and obtained consistent results, we are confident in the conclusions we have drawn.

One issue discovered during data collection was the euclidian distance calculation used to filter solutions. This can be seen in figure 5.1a where the euclidian distance tends to scale exponentially rather than uniformly. While not ideal, this bug has no impact on our conclusions. We are focused on the comparison between puzzle generation methods, rather than the realignment process. As long as all solutions are subjected to the same realignment criteria, our comparisons remain valid.

6.3 Future Work

Borderlands Science has proven to be a remarkable and unique study in citizen science with a larger audience, and the work is not yet complete. Further investigations have led to alternative methods for filtering player solutions. That, in conjunction with these findings, may lead to significant results using only a small number of puzzles and solutions. These results may not only contribute to new rules, but the refinement process will lead to greater independence in future games. Less puzzles, and fewer essential solutions, results in

a reduced reliance on a AAA titles audience. As we gain knowledge on how to optimize our process, we obtain more freedom and flexibility for the rules imposed on the players. That being said, the contribution from the Borderlands Science players cannot be underestimated, and the sole reason we can refine our process is due to the abundance of solutions and feedback received. Borderlands Science has shown the true value of a larger community in citizen science, and a broader community will remain a foundational stepping stone in future projects.

Future projects will likely not utilize RNA, and consider vastly different bioinformatics problems. Regardless of the project base, these results guide us towards producing higher quality puzzles, and relying less heavily on the sheer number of players associated with a AAA title.

Chapter 7

Conclusion

We presented a citizen science study focused on optimizing data quality in order to not only reduce the number of puzzles needed in Borderlands Science, but also to improve upon existing methods.

We produced multiple batches of puzzles using custom puzzle generation techniques to observe how different parameters played a role in data quality. These parameters resulted in a varying numbers of puzzles per batch, allowing us to estimate the impact of targeted selection compared to overall coverage.

Puzzle solutions from these batches were utilized in a realignment script to adjust an existing base alignment. Final realignments from each batch were compared, allowing us to conclude that different approaches to selecting puzzles had a significant influence on the final outcome. Notably, a smarter approach to selecting puzzles produced far better results

than the rudimentary alternative. This incentives future projects to spend more time in the initial development stage refining what is given to players, rather than in the later stages focused on filtering.

Bibliography

[1] M. Chatzou, C. Magis, J.-M. Chang, C. Kemena, G. Bussotti, I. Erb, and C. Notredame, "Multiple sequence alignment modeling: methods and applications," *Briefings in Bioinformatics*, vol. 17, pp. 1009–1023, 2015.

[2] J. Thompson, B. Linard, O. Lecompte, and O. Poch, "A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives," *PLoS One*, vol. 6, p. e18093, 2006.

[3] W. Just, "Computational complexity of multiple sequence alignment with sp-score," *Journal of Computational Biology*, vol. 8, no. 6, pp. 615–623, 2001. PMID: 11747615.

[4] C. Kemena and C. Notredame, "Upcoming challenges for multiple sequence alignment methods in the high-throughput era," *Bioinformatics*, vol. 25, no. 19, pp. 2455–2465, 2009.

[5] J. D. Thompson, F. Plewniak, and O. Poch, "A comprehensive comparison of multiple sequence alignment programs," *Nucleic Acids Research*, vol. 27, pp. 2682–2690, 07 1999.

[6] A. Löytynoja and N. Goldman, "Uniting alignments and trees," *Science*, vol. 324, no. 5934, pp. 1528–1529, 2009.

[7] J. A. Simpson and E. S. C. Weiner, "The oxford english dictionary."

[8] D. Cavalier and E. Zachary, *The Rightful Place of Science: Citizen Science*. 05 2016.

[9] J. Silvertown, "A new dawn for citizen science," *Trends in Ecology & Evolution*, vol. 24, no. 9, pp. 467–471, 2009.

[10] R. Bonney, J. L. Shirk, T. B. Phillips, A. Wiggins, H. L. Ballard, A. J. Miller-Rushing, and J. K. Parrish, "Next steps for citizen science," *Science*, vol. 343, no. 6178, pp. 1436–1437, 2014.

[11] J. Mounthanyvong, "Multiple sequence alignment through citizen science : Complexity and computer performance on the borderlands alignment problem," Master's thesis, McGill University, 2021.

[12] S. Mirarab, N. Nguyen, S. Guo, L. Wang, J. Kim, and T. Warnow, "Pasta: Ultra-large multiple sequence alignment for nucleotide and amino-acid sequences," *Journal of Computational Biology*, vol. 22, no. 5, pp. 377–386, 2015.

[13] R. C. Edgar, "Muscle: a multiple sequence alignment method with reduced time and space complexity," *BMC Bioinformatics*, vol. 5, no. 113, 2004.

[14] B. Koepnick, J. Flatten, T. Husain, *et al.*, "De novo protein design by citizen scientists," *Nature*, vol. 570, pp. 390–394, 2019.

[15] A. Kawrykow, G. Roumanis, A. Kam, D. Kwak, C. Leung, C. Wu, E. Zarour, P. players, L. Sarmenta, M. Blanchette, *et al.*, "Phylo: a citizen science approach for improving multiple sequence alignment," *PloS one*, vol. 7, no. 3, p. e31362, 2012.

[16] D. Kwak, A. Kam, D. Becerra, Q. Zhou, A. Hops, E. Zarour, A. Kam, L. Sarmenta, M. Blanchette, and J. Waldispühl, "Open-phylo: a customizable crowd-computing platform for multiple sequence alignment," *Genome Biol*, vol. 104, p. R116, 2013.

[17] L. Fu, B. Niu, Z. Zhu, S. Wu, and W. Li, "CD-HIT: accelerated for clustering the next-generation sequencing data," *Bioinformatics*, vol. 28, pp. 3150–3152, 10 2012.

[18] S. Mulia, D. Mishra, and T. Jena, "Profile hmm based multiple sequence alignment for dna sequences," *Procedia Engineering*, vol. 38, pp. 1783–1787, 2012. INTERNATIONAL CONFERENCE ON MODELLING OPTIMIZATION AND COMPUTING.

[19] N. C. W. Goonesekere and L. Byungkook, "Frequency of gaps observed in a structurally aligned protein pair database suggests a simple gap penalty function," *Nucleic Acids Research*, vol. 32, no. 9, pp. 2838–2843, 2004.

[20] F. P. Miller, A. F. Vandome, and J. McBrewster, *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau?Levenshtein Distance, Spell Checker, Hamming Distance*. Alpha Press, 2009.

[21] J. H. V. Lint, *Introduction to Coding Theory*. Berlin, Heidelberg: Springer-Verlag, 3rd ed., 1998.

[22] M. N. Price, P. S. Dehal, and A. P. Arkin, "Fasttree 2 – approximately maximum-likelihood trees for large alignments," *PLOS ONE*, vol. 5, pp. 1–10, 03 2010.

[23] T. Z. DeSantis, P. Hugenholtz, N. Larsen, M. Rojas, E. L. Brodie, K. Keller, T. Huber, D. Dalevi, P. Hu, and G. L. Andersen, "Greengenes, a chimera-checked 16s rrna gene database and workbench compatible with arb," *Applied and Environmental Microbiology*, vol. 72, no. 7, pp. 5069–5072, 2006.

[24] S. Mirarab, N. Nguyen, and T. Warnow, *SEPP: SATé-Enabled Phylogenetic Placement*, pp. 247–258.

[25] M. Kendall and C. Colijn, "Mapping Phylogenetic Trees to Reveal Distinct Patterns of Evolution," *Molecular Biology and Evolution*, vol. 33, pp. 2735–2743, 06 2016.

[26] J. Jansson, K. Mampentzidis, R. Rajaby, and W.-K. Sung, "Computing the rooted triplet distance between phylogenetic networks," *Algorithmica*, vol. 83, pp. 1786–1828, 2021.

[27] I. Kalvari, E. P. Nawrocki, N. Ontiveros-Palacios, J. Argasinska, K. Lamkiewicz, M. Marz, S. Griffiths-Jones, C. Toffano-Nioche, D. Gautheret, Z. Weinberg, E. Rivas, S. R. Eddy, R. Finn, A. Bateman, and A. I. Petrov, "Rfam 14: expanded coverage of metagenomic, viral and microRNA families," *Nucleic Acids Research*, vol. 49, pp. D192–D200, 11 2020.

[28] N. Mantel, "The detection of disease clustering and a generalized regression approach.," *Cancer research*, vol. 27, no. 2, pp. 209–220, 1967.

List of Figures

List of Tables

Appendix A

Realignment Comparisons

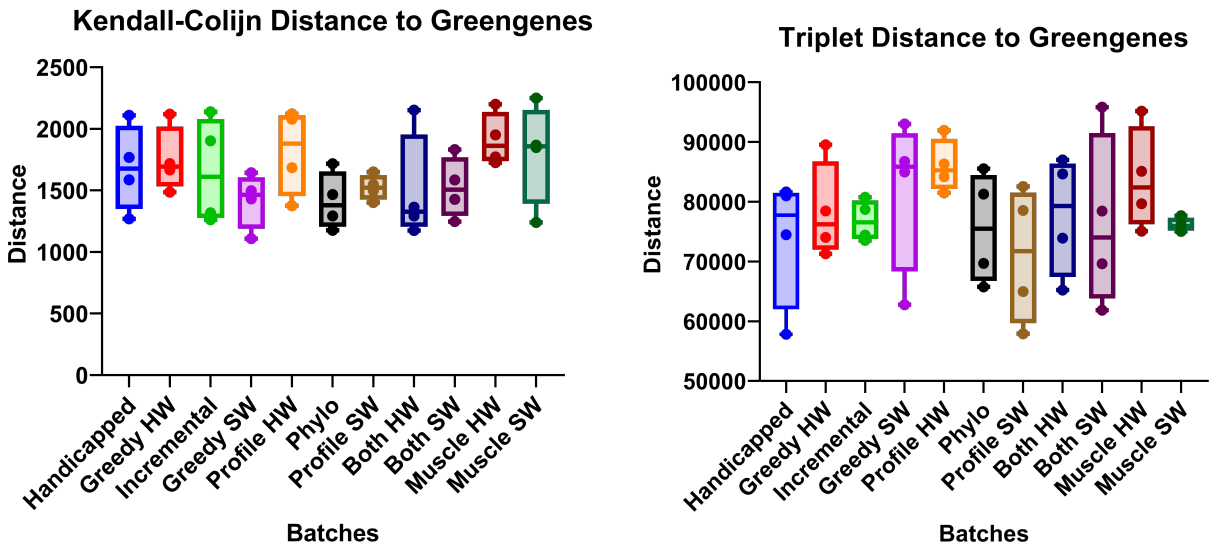


Figure A.1: Comparison between all batches seen in table 5.1 where distance is the calculated difference between the BLS phylogenetic tree, and the Greengenes reference tree, with the objective to be as low as possible.

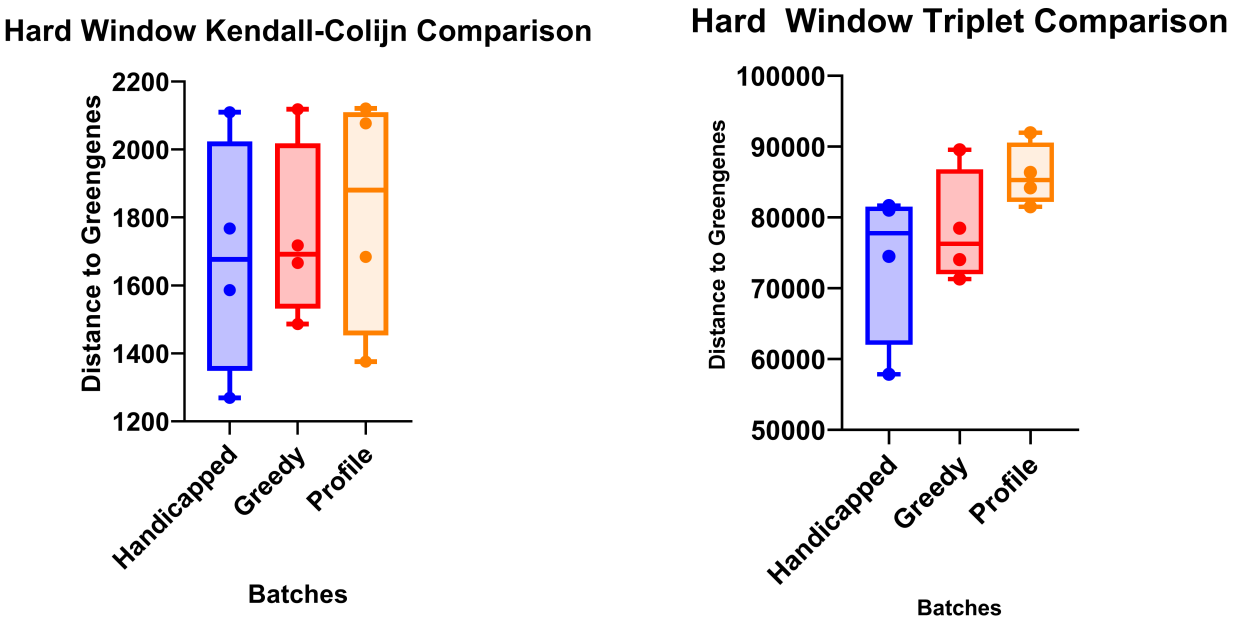


Figure A.2: Comparison of hard window batches

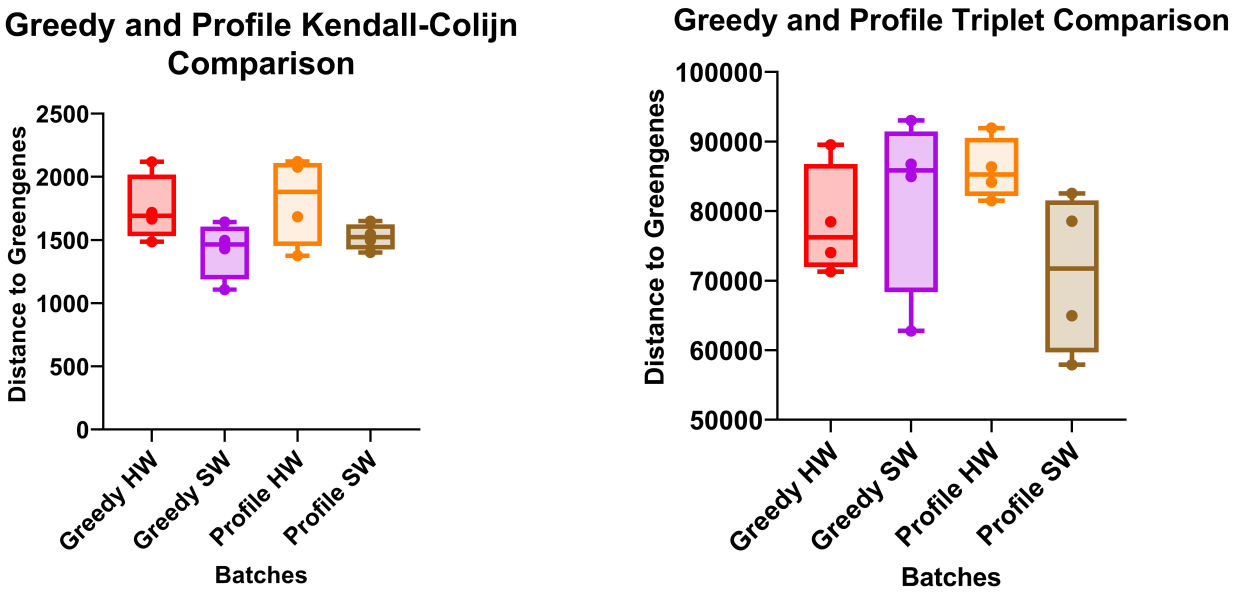


Figure A.3: Comparison of greedy and profile solution algorithm batches

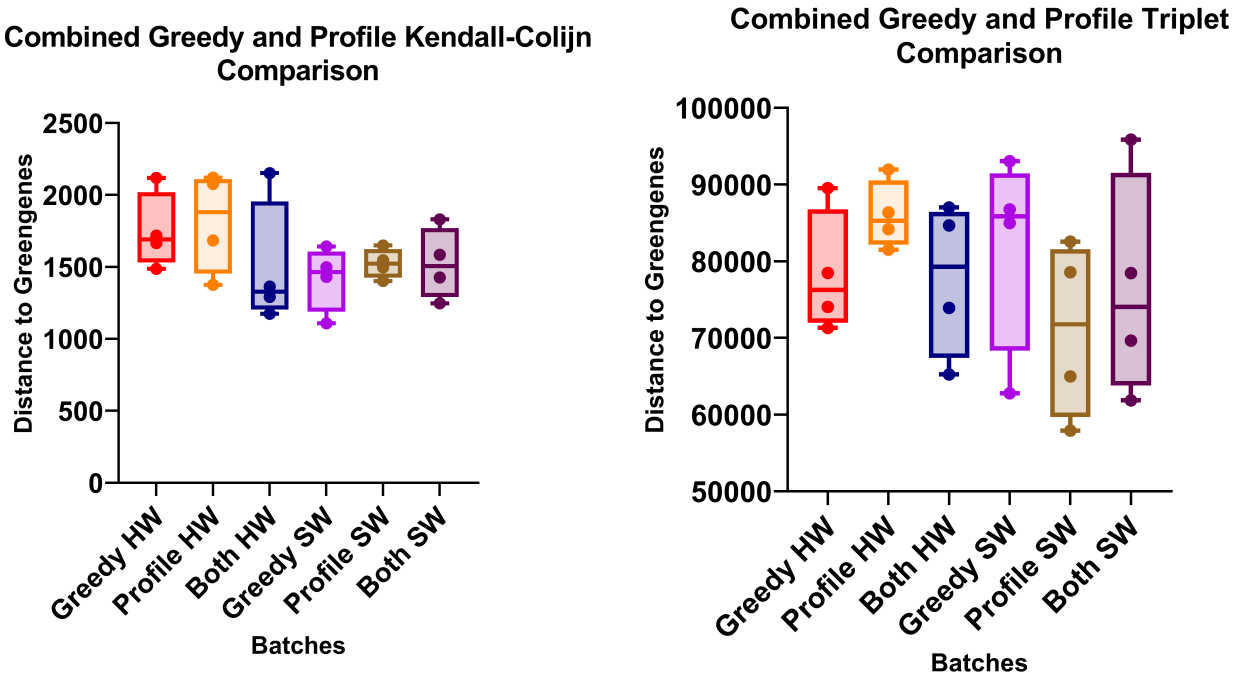


Figure A.4: Comparison of greedy and profile solution algorithm batches to batches which utilize both

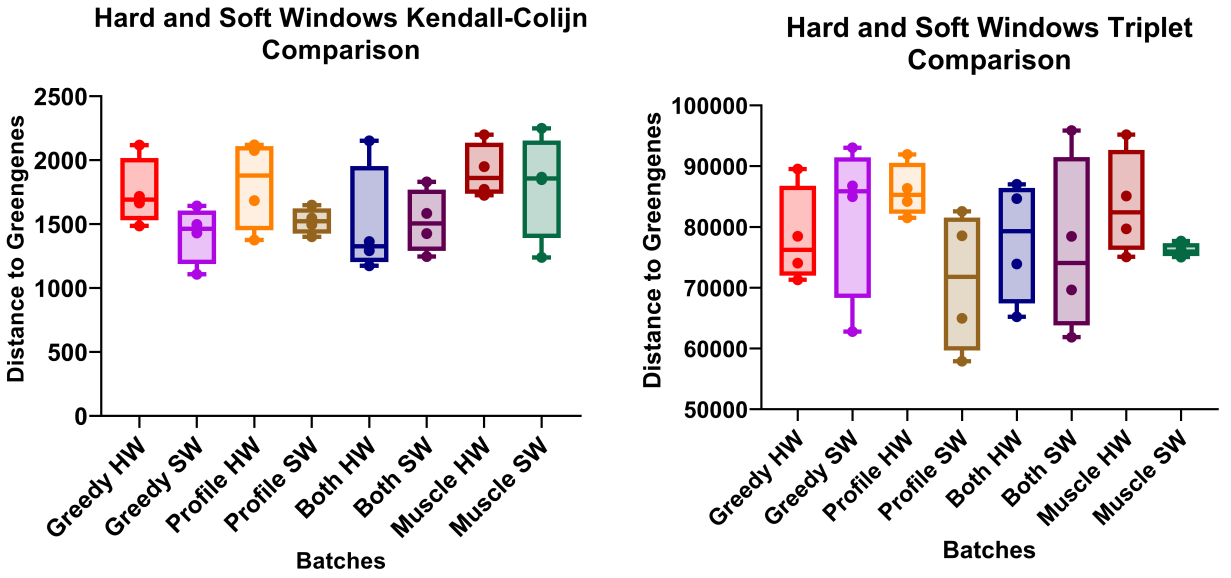


Figure A.5: Comparison of hard and soft window batches

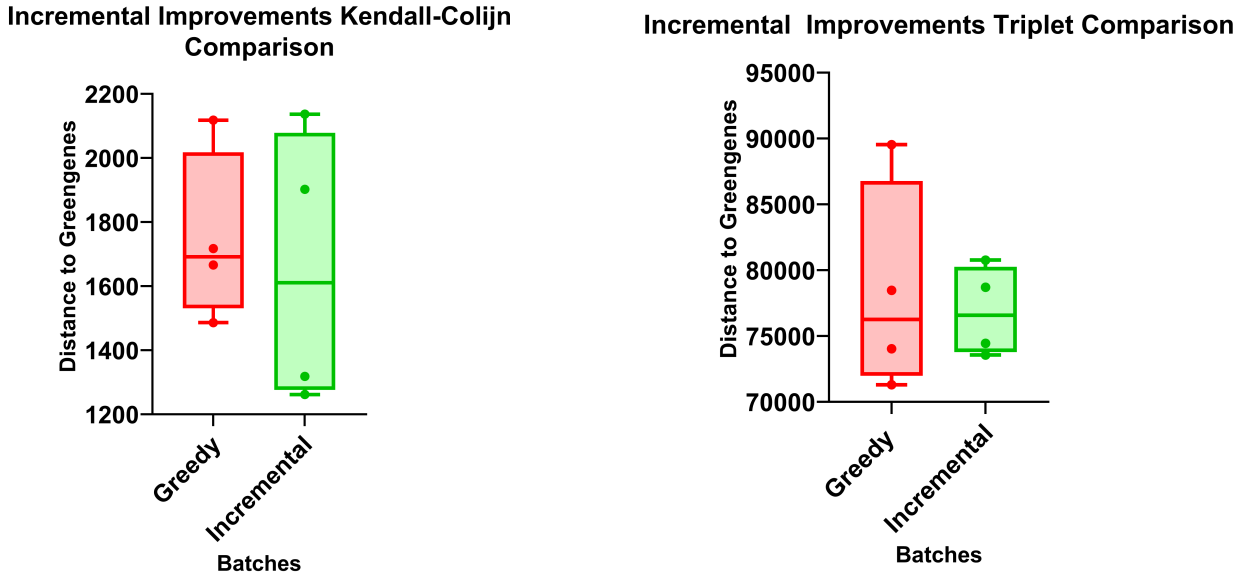


Figure A.6: Comparison of standard acceptance parameters to incremental parameters

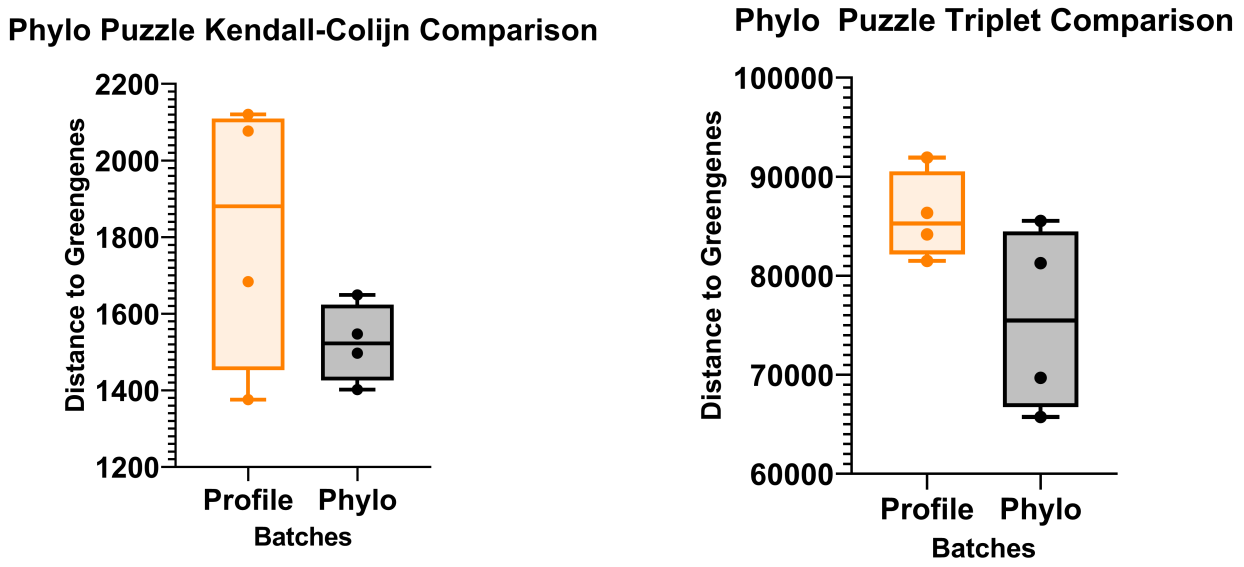


Figure A.7: Comparison of standard scoring scheme to Phylo scoring scheme

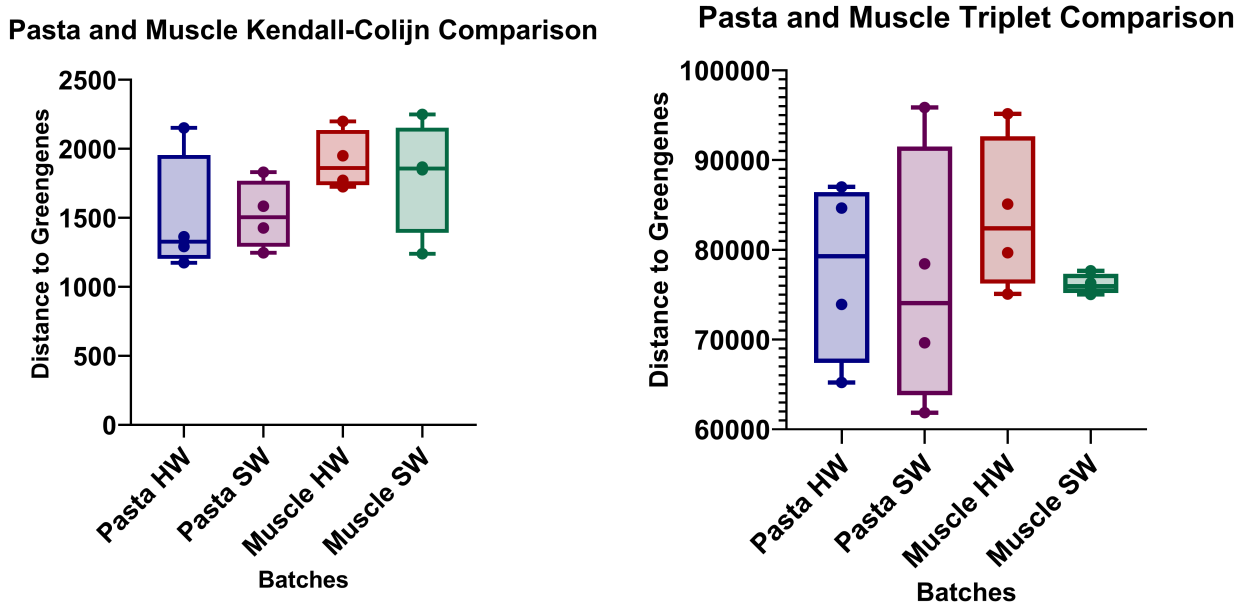


Figure A.8: Comparison of PASTA and MUSCLE based batches