Multiple Sequence Alignment through Citizen Science

Complexity and Computer Performance on the Borderlands Alignment Problem

Julien Mounthanyvong

May 22, 2021

Contents

Common Notations

- $R = \{S^1, \ldots, S^n\}$ is a set of sequence over an alphabet Σ.
- The letters of a sequence will be written as $S^i = s_1^i \ldots s_m^i$.
- $G = (g_1, \ldots, g_M)$ with $g_j \in P(\Sigma), 1 \leq |g_j| \leq 2$ is a guide.
- $b \geq 1$ is the bonus value of the puzzle.
- An alignment to the guide $\mathcal{A}$ is obtained by inserting spaces "_" at the beginning, into or at the end of each sequence S^i such that the resulting S'^i is of the same length M as the guide.
- The resulting sequences in an alignment will be denoted as $S'^1, \ldots, S'^n$, where $|S'^1| = \cdots = |S'^n| = M$ (with $S'^i = s_1'^i \ldots s_M'^i$).
- t is the maximum number of gaps that can be used in a puzzle.
- Index i refers to the i^{th} sequences, and index j refers to the j^{th} position.
- p is the scoring function for an alignment.

Abstract

In this work, we study the Multiple Sequence Alignment problem, which is very important in the field of bioinformatics. A lot of different algorithm have been proposed in order to solve it, but none of them produce perfect alignments. In order to improve current alignments, we propose Borderlands Science, a novel approach for the Multiple Sequence Alignment problem. Here, we want to harness the collective human power and use it to get good solutions to small snippets of a global alignment problem. We hope that human players will be able to respect an intuitive trade-off between the number of gaps used and the quality of the alignment, and that these improvement to small parts of the problem will lead to a better global alignment. This study is part of the Borderlands Science project, and focus on examining two of the hypothesises used when it was proposed : that the puzzles we are sending to the players are complex from a computational point of view, and that these players do not follow simple heuristic that can be easily replicated with algorithms. Here, we first want to define the problem that is being solved by the human players. Then, we will provide a formal study of the problem and its complexity. Finally, we will take a look at some simple algorithms that can be used to solve the problem and see how human players perform compared to these algorithms.

Abrégé

Dans cette étude, nous nous intéressons à l'alignement multiple de séquences, qui est très important dans le domaine de la bio-informatique. De nombreux algorithmes différents ont été proposé pour résoudre ce problème, mais aucun d'entre eux n'est capable de produire des alignements parfaits. Afin d'améliorer les alignements actuels, nous proposons Borderlands Science, une nouvelle approche au problème de l'alignement multiple de séquences. Ici, nous souhaitons canaliser la puissance humaine collective et de l'utiliser pour obtenir de bonnes solutions sur des petites parcelles d'un problème d'alignement global. Nous espérons que les joueurs humains seront capable de réaliser un compromis intuitif entre le nombre d'espaces utilisé et la qualité de l'alignement, and que ces améliorations à de petites parties du problème permettront de créer un meilleur alignement global. Cette étude s'inscrit dans le cadre du projet Borderlands Science, et se concentre sur l'analyse de deux des hypothèses formulées lors de la proposition du sujet : que les puzzles envoyés au joueurs sont complexe d'un point de vue informatique, et que ces joueurs ne suivent pas une heuristique simple facilement reproductible par des algorithmes. Ici, nous souhaitons tout d'abord définir le problème résolu par les joueurs humains. Ensuite, nous fournirons une étude formelle de ce problème et de sa complexité. Finalement, nous étudierons quelques algorithmes simples pouvant être utiliser afin de résoudre ce problème and nous analyserons comment les joueurs humains performent comparé à ces algorithmes.

Acknowledgements

I would like to thank my supervisor Jérôme Waldispühl for his guidance before and throughout my Master's studies, for introducing me to the project, and for his help and support in preparing and redacting this thesis. I would also like to thank Roman Sarrazin-Gendron, Mathieu Blanchette, Alexander Butyaev, Chris Drogaris, Eddie Cai, Tim Keding and Renata Mutalova for their assistance and their feedback on this work.

Contributions

Julien Mounthanyvong has performed the research described in this thesis. He devised the proofs, the methods and the experiments under the guidance and the supervision of Jérôme Waldispühl.

Context

0.1 Multiple Sequence Alignment

Multiple Sequence Alignment is a very central problem of bioinformatics that consists of finding the best way to arrange multiple (3 or more) biological sequences (DNA, RNA or protein sequences) to highlight the most similarity between those sequences. This is very useful as it helps us find homology between sequences, infer the evolution, or predict the sequence's function for example. it is also one of the earliest computational biology problem studied, with works dating back to the 1970's [1].

In order to solve this problem, a lot of different methods have been developed [2, 3]. In most of these formulations, the goal is to find the alignment maximising a given score, with the sum-of-pairs score for example which has been used in most study. This is a problem that is specifically hard on large datasets, with some works focusing on these cases for DNA sequences [4] or proteins sequences [5] for example. Some research have also proposed ways of improving these scoring schemes so as to take into account the phylogenetic tree in order to find alignments that are better fitted to this knowledge [6, 7]. Some of the commonly used programs we can note include CLUSTALW [8], MUSCLE [9], T-COFFEE [10], SATé-II [11], and PROBCONS [12].

Among these methods, we especially note PASTA which will be relevant to our project [13]. PASTA alignment works by repeating these following steps until convergence : estimate a guide tree, separate the problem into smaller ones based on this tree, use MAFFT [14] to align each of these sub-problems, then reconstruct a global alignment. This method has the particularity of creating alignments that are pretty compact, which is well suited to our project.

Still, the best alignment remains very hard to compute in any of these problem formulations as they are NP-hard for any reasonable scoring scheme

[15]. In fact, even for small problems, we usually have no efficient algorithm to find the best answer. Moreover, mathematically defining the quality of an alignment is also difficult as we have no universal way of comparing two different alignments since there are different criteria that can be optimised. Therefore it is hard to decide what is the best scoring scheme that we should use to evaluate the alignments.

0.2 Citizen Science

As it turns out, humans are quite proficient in solving these kind of problem, as the human mind is good at solving multiple constraints at the same time, and has a nice intuitive understanding of the trade-off between the number of gaps used and the number of nucleotides or proteins that are correctly aligned.

The idea of using the collective human abilities in order to solve a hard scientific problem is known as citizen science [16]. This concept first appeared with the Audubon Society's Christmas Bird Count which is a project that used the help of volunteers to carry out a global bird census [17]. Citizen science was greatly developed with the use of computers as it allowed for the work to reach more peoples and opened up efficient ways of storing and analysing the data.

The public participation in these research projects can come in different forms : They can for example provide some processing power for large-scale distributed computing. The Berkeley Open Infrastructure for Network Computing (BOINC) middleware system has been developed to help these kinds of projects [18]. Some successful attempts include SETI@home used for signal analysis [19], and Folding@home about protein folding that notably studied SARS-CoV-2 [20].

Other projects require the participant to complete some tasks, like Galaxy

Zoo where users have to identify some pictures [21]. It should be noted that scientific expertise is not necessarily required to participate in these kind of projects : If we can reduce the problem to a simply defined task with a clear interface, then it might be solved without having to understand the underlying scientific problem. For example, we might try to formulate the task in the form of an accessible, entertaining game so as to appeal to a broader audience and get more participation [22]. For instance, Eyewire is a game used to help in mapping the brain [23], while Quantum Moves assist scientists in the development of quantum computers.

0.3 Introduction + Outline

Citizen science games have been used in the context of biological problems too, for example with Foldit that study protein folding [24, 25], and more recently with Phylo that covers Multiple Sequence Alignment [26].

Here we study Borderlands Science which is a citizen science initiative concerning Multiple Sequence Alignment. It differs from the Phylo project on different points : First, they focus on different datasets. Phylo concerns mammalian genes, where we have some prior knowledge about the origins of the sequences. In Borderlands Science, we want to improve some pre-existing microbial DNA sequences alignment, which is a significantly larger problem. Moreover, since these sequences are not related to specific species, we have to change the method of evaluating the alignment. The support has also been changed : While Phylo relied on a game that can be played on a website, Borderlands Science use a mini-game that is directly implemented in the AAA game "*Borderlands 3*" in order to reach a broader audience. The game interface was also modified in order to fit this change as well as to make the game more intuitive and attractive.

More precisely, the overall idea of this project is as such : We study the

microbial DNA sequences obtained by the Microsetta initiative, and we start with the PASTA alignment of these sequences. Since this alignment is pretty compact, we believe that adding more gap will likely be interesting. To do that, we select small parts of the alignment that we feel could be improved and make puzzle out of them. Then, we feed these puzzles to the player base and gather their solutions. Finally, we will use this data to change the original alignment.

When designing this project, we supposed that the problem sent to the players was not easy to solve for a computer, and that human players would provide answers that are not easily replicated by simple algorithms. In this work, we will study these assumptions. To that end, we want to provide a formal study of the game aspect of this project. We will first give mathematical formulations of the objects we are studying. Then we will analyse the complexity of the problem. Finally, we will discuss some algorithmic ways of solving the puzzles that can be used in practice and try to compare these methods to the results obtained by the human players through the Borderlands Science game.

1 Introducing the Problem

1.1 Definition of the problem

Let's define the problem for a set of sequences $R = \{S^1, \ldots, S^n\}$ over an alphabet Σ (Sequences $S^i \in \Sigma^*$), a guide $G = (g_1, \ldots, g_M)$ with $g_j \in P(\Sigma), 1 \leq |g_j| \leq 2$ for $j \in \{1, \ldots, M\}$ ($M \geq |S^i|$ for $i \in \{1, \ldots, n\}$), and a bonus value $b \geq 1$.

Note that the constraint of having at most 2 letters for each part of the guide was chosen arbitrarily with nucleotides sequences in mind, as we believed that this would lead to more relevant results. Still, as we will see later, this constraint can be changed without impacting the overall complexity of this problem.

An alignment to the guide $\mathcal{A}$ is obtained by inserting spaces "_" at the beginning, into or at the end of each sequence S^i such that the resulting S'^i is of the same length M as the guide (the space can be considered as an added letter "_" to the alphabet). The resulting sequences will be denoted as $S'^1, \ldots, S'^n$, where $|S'^1| = \cdots = |S'^n| = M$. The number of gaps used by the alignment is the number of spaces that are not at the end of a sequence.

Letters of a given sequence S will be denoted as $S = s_1 \ldots s_m$.

Borderlands Alignment (BLA) :

The score of an alignment $\mathcal{A}$ (denoted as $p(\mathcal{A})$) is computed as such :

$$p(\mathcal{A}) = \sum_{j=1}^{M} b_j p_j$$

where $p_j = |\{i | s'^i_j \in g_j\}|$

and $b_j = b$ if $p_j = n$, and $b_j = 1$ otherwise

An example of such an alignment is given in Figure 1. Correctly aligned nucleotides are in a bold square.

The goal of BLA is, given a fixed number t, to find the alignment $\mathcal{A}$ using at most t gaps giving the highest possible score.

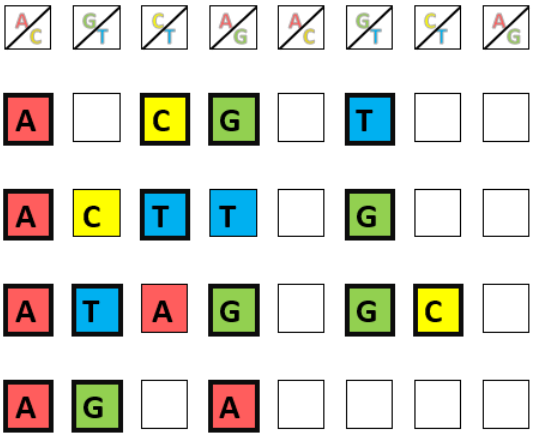


Figure 1: Example of a Borderlands Alignment

The first row corresponds to the guide, the other rows each corresponds to a sequence. Correctly aligned nucleotides to the guide are put in bold cases.

$$\text{Find } \mathcal{A}^* = \arg\max_{\mathcal{A}} p(\mathcal{A})$$

1.2 Presentation of the Game

The players are sent a puzzle consisting of a guide $G = (g_1, \ldots, g_M)$ along with an alignment $\mathcal{A}_0$ of the set of sequences $R = \{S^1, \ldots, S^n\}$ where the spaces are inserted at the end of each sequences, i.e. for each $i = 1 \ldots n$, we have that $s'^i_j = s^i_j$ for $j = 1 \ldots |S^i|$, and $s'^i_j = \_$ for $j = (|S^i| + 1) \ldots M$. Alongside this puzzle, they are given a maximum number of gaps t as well as a par score p_{par}, whose purpose will be explained later.

From this starting alignment, the player will have to move the spaces around in order to increase the score of the alignment while using a limited number t of gaps, where the number of gaps used by an alignment consists

of the number of spaces that were not inserted at the end of a sequence (for example, for the starting alignment $\mathcal{A}_0$, the number of gaps used is 0 since all the spaces are inserted at the end of each sequences).

The moves available to the players can be divided into 3 categories :

- Adding a gap : Take a space at the end of a sequence and move it at the beginning or into the sequence. This kind of move increases the number of gaps used by one.
- Moving a gap : Take a space that is not at the end of a sequence and move it at the beginning or into the sequence. This kind of move does not change the number of gaps used.
- Removing a gap : Take a space that is not at the end of a sequence and move it at the end of the sequence. This kind of move decreases the number of gaps used by one.

The game can be completed once the player reach an alignment whose score exceed the predefined par score p_{par}.

Each puzzle is attributed a difficulty ranging from 1 to 9 that roughly correlates to its size (number of sequences in the puzzle and length of each sequence), with 1 corresponding to the smallest puzzles and 9 to the biggest ones.

Note that this game possesses some significant differences compared to the most commonly seen formulations of MSA : Here, the alignment is made in order to match a given guide which defines the scoring function. Moreover, the number of gaps that can be used is a hard constraint in these puzzles as they are strictly limited, whilst other tend to use a soft constraint for this criteria by giving a penalty in the score for each gap used. Because of these changes, some of the well known methods used to solve MSA will not be suited for this game, and therefore we need to think about other efficient ways of treating this problem.

1.3 Complexity (Brute-force Approach)

Let's study the complexity of a brute-force approach to solve the following problem :

Given a set of sequences $R = \{S^1, \ldots, S^n\}$ over an alphabet Σ (Sequences $S^i \in \Sigma^*$), a guide $G = (g_1, \ldots, g_M)$ with $g_j \in P(\Sigma), 1 \leq |g_j| \leq 2$ for $j \in \{1, \ldots, M\}$ ($M \geq |S^i|$ for $i \in \{1, \ldots, n\}$), and a bonus value $b \geq 1$: Find an alignment $\mathcal{A}$ maximising the score.

Let's first consider the case where we can use an unbounded number of gaps.

We denote as for all i, $l_i = |S^i|$ the length of each sequence. The number of possible alignments is :

$$brut((R, G)) = \prod_{i=1}^{n} \binom{M - l_i}{M}$$

Indeed, for each sequence S^i, the number of spaces for this sequence in the alignment is $M - l_i$. Therefore, there are $\binom{M-l_i}{M}$ configurations possible for this sequence. Since we have no restrictions regarding the number of gaps we can use, any combination of the sequences are accepted. Hence, the number of alignments we have to test is $\prod_{i=1}^{n} \binom{M-l_i}{M}$.

Supposing that $0 < l_i < M$ (since otherwise, the alignment of this sequence is trivial), we can find a lower bound to this complexity : We know that $\binom{M-l_i}{M} \geq M$, and therefore $brut((R, G)) \geq M^n$. So the complexity of a brute-force approach to solve this problem will be exponential with regard to the number of sequences (and at least polynomial regarding the length of the sequences).

With a limited number t of gaps :

$$brut((R, G)_t) = \sum_{(t_1, \ldots, t_n) \in T^t_{n,M}} \left(\prod_{i=1}^{n} \binom{t_i}{l_i + t_i - 1} \right)$$

where $T^t_{n,m} = \left\{(t_1, \ldots, t_n) \middle| \forall i = 1 \ldots n, l_i + t_i \leq M, \sum_{i=1}^n t_i \leq t\right\}$
Note that we have the following properties :

$$
\begin{aligned}
|T^t_{n,M}| &= \sum_{k=0}^{t} |T^k_{n-1,M}| \\
|T^t_{1,M}| &= \min\{t, M - l_1\} \\
|T^0_{n,M}| &= 1
\end{aligned}
$$

Explanation : Let's characterise a correct alignment. For an alignment, let's count t_i the number of gaps used in the i^{th} sequence S_i, i.e. the spaces that are not at the end of the sequence. Since the total length is m, we know that $t_i + l_i \leq m$. Since we are only using at most t spaces, this means that $(t_1, \ldots, t_n)$ is in T^t_n.

Then, for a given $(t_1, \ldots, t_n) \in T^t_n$, let's count the number of possible configuration. For the sequence S_i, we have t_i gaps that are used. This means that the last letter of the sequence is at the $(l_i + t_i)^{th}$ position, and hence that the t_i gaps used are distributed among the $(l_i + t_i - 1)^{th}$ positions. Since any of these distribution is correct, we have $\binom{t_i}{l_i + t_i - 1}$ possibilities, and therefore we have $\prod_{i=1}^n \binom{t_i}{l_i + t_i - 1}$ combinations using this distribution of gaps.

Hence, the total number of alignments that have to be considered is

$$\sum_{(t_1,\ldots,t_n) \in T^t_{n,M}} \left(\prod_{i=1}^{n} \binom{t_i}{l_i + t_i - 1} \right)$$

As we don't have an easier way of explicitly formulating the number of correct configurations that have to be considered, let's instead take a look at a rough lower bound of this complexity : If we take the same assumption as before (which is that $0 < l_i < M$ for all i), and add the supposition that the number t of gaps isn't too low with $t \geq n$, then we find that

$$brut((R,G)_t) \geq \prod_{i=1}^{n} l_i \geq l_{min}^n$$

where $l_{min} = \min_{1 \leq i \leq n} l_i$. Hence, the number of accepted solution will rise exponentially when we add non trivial sequences, and at least in polynomial growth when we increase the length of the shortest sequence.

As such, the space of alignments that respect all the constraints is still too large to be studied as a whole efficiently. This motivates the search of a way of getting the best solution or at least "good" solutions (as in close to the best) in a more practical way.

1.4 NP-Hardness

Let's show that BLA is an NP-complete problem.

For this proof, we only considered the case where every part of the guide contains exactly two letters, i.e. that $G = (g_1, \ldots, g_M)$ with $g_j \in [\Sigma]^2 = \binom{\Sigma}{2}$. This is a bit more restrictive than the usual problem and make the proof a bit more complex, but it can be easily adapted to guide containing any number of letters.

Proof

To show that BLA is an NP-complete problem, let's reduce an instance of Longest Common Subsequence to an instance of BLA. For the Decision problem form of BLA, we want to know if there is an alignment $\mathcal{A}$ respecting all of the constraints such that $p(\mathcal{A}) \geq c$ for a given objective score c.

Definition : Subsequence

Given a sequence $S = s_1 \ldots s_m$, we say that $S' = s'_1 \ldots s'_k$ is a subsequence of S (denoted as $S' < S$) iff there is an increasing function $\phi : [[1,k]] \to [[1,m]]$ s.t. $\forall i \in \{1, \ldots, k\}$, $s_{\phi(j)} = s'_j$.

Longest Common Subsequence (LCS) :

The LCS problem consists of : Given a set of sequences $R = \{S^1, \ldots, S^n\}$ over an alphabet Σ, find (one of) the longest S such that $\forall i \in \{1, \ldots, n\}$, $S < S^i$.

For the Decision problem form of LCS, we want to know whether there exists a common subsequence S such that $|S| \geq k$ for a given $k \in \mathbb{N}$. For example, Figure 2 shows a set of sequences whose longest subsequence has length 2.

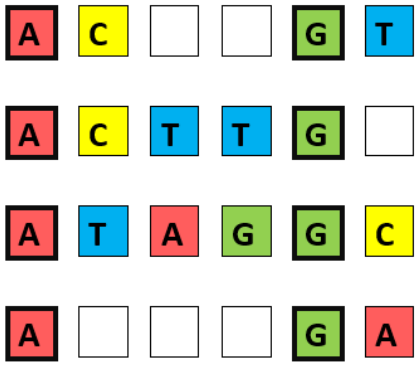


Figure 2: Example of a Longest Common Subsequence

This is an alignment (without a guide) showcasing a case of Longest Common Subsequence. Each row corresponds to a sequence. Letter that belong in the longest common subsequence are put in bold cases.

We know that LCS is an NP-complete problem for any alphabet Σ such that $|\Sigma| \geq 2$.

- Let's reduce an instance of LCS with $\Sigma = \{a, b\}$ into an instance of BLA :

  The problem we are studying is : Given a set of sequences $R = \{S^1, \ldots, S^n\}$ over the alphabet $\Sigma = \{a, b\}$, is there a common subsequence of length at least k ?

Let Σ' be the alphabet Σ with an added letter x ($\Sigma' = \{a, b, x\}$). Let $M = \sum |S^i|$, and let's define the following guide : $G = (g_1, \ldots, g_{2M})$, with $g_{2k-1} = \{a, x\}$ and $g_{2k} = \{b, x\}$ for $k \in \{1, \ldots, M\}$. We define a bonus such that $bn > 2M(n-1)$, i.e. $b > 2M\frac{n-1}{n}$. The objective c is defined as $c = kbn$, and the number of gaps available is $t = 2nM$.

Some explanations : the alphabet is changed and the guide is defined as such so we can make sure we know what letter we have at each position when it is aligned to the guide. The length of the guide and the number of gaps available are set as to not be a constraint, so that we will be able to consider any alignment of the sequences. Finally, the bonus is taken high enough so as to put the emphasis on achieving bonuses on as many positions as possible : Indeed, here, the score we get from having one position with a bonus is higher than the highest score we can get from an alignment that doesn't get any bonus. Then, the objective score is set so that it will be beaten if we manage to achieve bonuses on k positions.

Therefore, the corresponding BLA problem is : Given the set of sequences R over the alphabet Σ', is there an alignment $\mathcal{A}$ such that $p(\mathcal{A}) \geq c$?

(Note that sequences of Σ can naturally be seen as sequences of Σ'.)

- If we find an alignment $\mathcal{A}$ with a score higher than c :

  Let's separate the positions between those where a bonus is achieved and those where it isn't :

  Let $B = \Big\{j \in \{1, \ldots, 2M\} | b_j = b\Big\} = \Big\{j \in \{1, \ldots, 2M\} | p_j = n\Big\}$ be the set of positions where a bonus is achieved. Let's note $q = |B|$.

$$p(\mathcal{A}) = \sum_{j \in B} b \times p_j + \sum_{j \notin B} p_j$$

We know that if $k \in B$, then $p_j = n$, and $p_j < n$ otherwise. Remember that the bonus was set so that the score achieved by position without bonuses is negligible. Therefore, we have the following upper bound :

$$\sum_{j \in B} b \times p_j = |B|bn \text{ and } \sum_{j \notin B} p_j \leq \sum_{j \notin B} n - 1 \leq 2M(n-1)$$

and hence :

$$|B|bn \leq sc(\mathcal{A}) \leq |B|bn + 2M(n-1) < (|B|+1)bn$$

meaning that $sc(\mathcal{A}) \geq c = kbn$ iff $|B| \geq k$.

Moreover, we know that if a position j is in B, then that means that every letter s_j^i is in the guide $g_j = (g_j^1, x)$ where $g_j^1 \in \Sigma = \{a, b\}$. Since we know that the sequences are words of Σ, this means that each $s_j'^i \neq x$, and so $s_j'^i = g_j^1$ for $i = 1 \dots n$. Let's write B as $\{b_1, \dots, b_q\}$ with $b_1 < \dots < b_q$. Then the word $g_{b_1}^1 \dots g_{b_q}^1$ is a common subsequence of length $q = |B| \geq k$.

- If we find a common subsequence S of length longer than k :

  We can find an alignment $\tilde{\mathcal{A}}$ (without a guide) of length lesser or equal than M such that every letter of S is perfectly aligned. First, we add spaces to the end of each sequences of this alignment until we get a length of M.

  From this alignment $\tilde{\mathcal{A}}'$, let's create an alignment to the guide G $\mathcal{A}$.

  - If $\tilde{s}_j'^i = \_$, then $s_{2j-1}'^i = s_{2j}'^i = \_$.
  - If $\tilde{s}_j'^i = a$, then $s_{2j-1}'^i = a$ and $s_{2j}'^i = \_$.
  - If $\tilde{s}_j'^i = b$, then $s_{2j-1}'^i = \_$ and $s_{2j}'^i = b$.

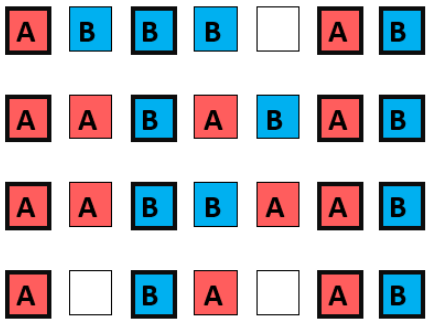


Figure 3: Initial Alignment obtained with the Longest Common Subsequence

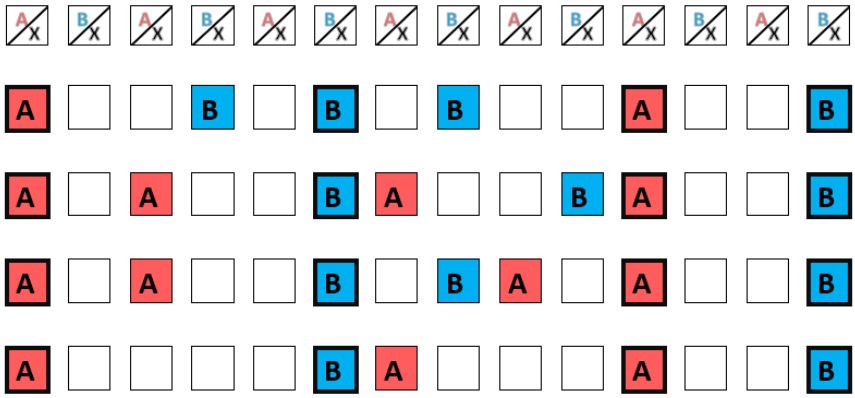


Figure 4: Corresponding Borderlands Alignment

An example of this transformation is shown with Figure 3 (original alignment $\tilde{\mathcal{A}}$) and Figure 4 (alignment to the guide $\mathcal{A}$).

This is a correct alignment to the guide, and since the letters of the subsequence S are perfectly aligned, then the score of this alignment is at least of $|S|bn \geq kbn = c$.

This complete the proof that BLA is NP-complete. Note that we can adapt this proof to the case where each part of the guide contains k letters by adding an appropriate number of letters to the new alphabet Σ' and using the same reasoning.

2 Heuristics

Let's analyse some greedy algorithms that we can use to solve the BLA problem, and compare their results to the solution proposed by the players.

We note that others MSA's formulations do not usually include a hard constraint regarding the number of gaps that can be used, and therefore most of the commonly used methods are not well suited to solve BLA. Because of this, we will be focusing mostly on greedy players where we can easily implement this new restriction.

2.1 Definitions

2.1.1 Naive Greedy Player

Let's first define a naive greedy player. Starting from the original alignment $\mathcal{A}_0$, this player will recursively consider all the possible moves as defined in 1.2 (that is adding a gap, moving a gap, and removing a gap), and choose to play the one that increase the overall score of the alignment the most. We repeat this process until we can't find a better alignment respecting the constraints anymore.

While this mostly gives us results that are better than what we achieve by choosing moves randomly (Figure 5), we believe that we won't be able to get the optimal solution with this method. Therefore, we still wish to find ways of improving this greedy player.

2.1.2 Depth Search Player

An important caveat of this method is that it does not look forward for future actions and it fails to consider what are the next moves that should be made in order to reach the best result. In order to fix this issue, we propose a player that will study the next possible moves with an in-depth search by

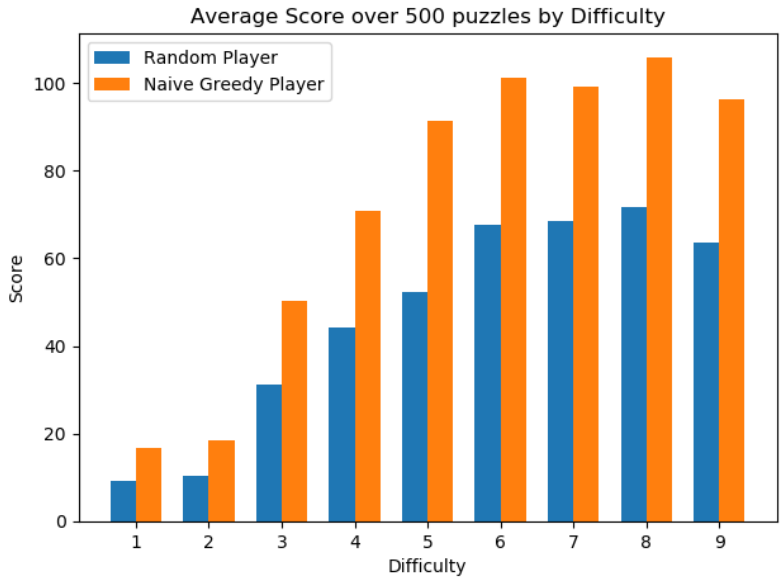


Figure 5: Comparison of a Random Player and a Naive Greedy Player

looking k steps ahead :

From an alignment, the player looks at all the possible accepted sequences of at most k moves, retains the one that leads to the best score, and play the first move from this combination. This process is done recursively until we can't find a better alignment.

However, this algorithm is too slow and isn't really useful in practice. In fact, it can't reasonably be used to solve any puzzle that allows us to use more than 10 gaps (Figure 6).

2.1.3 Heuristic Player : Methods

Another way to circumvent this issue is to change the objective function of the player : Instead of choosing the move that will maximise the score, we will choose the move maximising a linear combination h of the alignment score and some heuristics that we will define now.

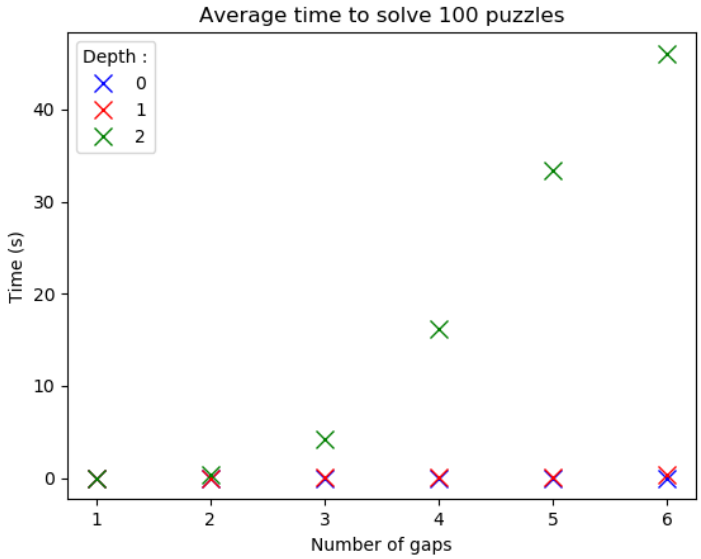


Figure 6: Run time of a Depth Search Player depending on the maximum number of gaps and the depth (Puzzle of difficulty 1)

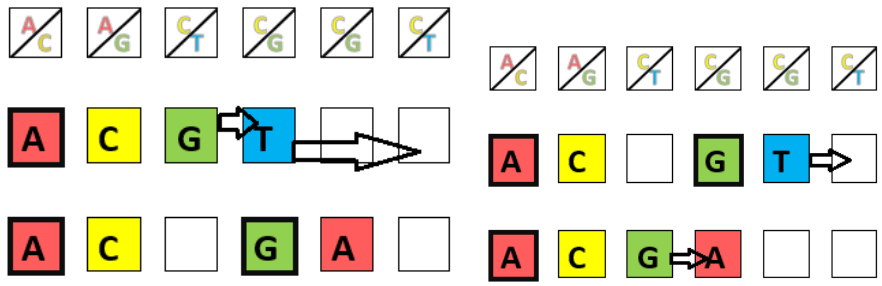
$$h(\mathcal{A}) = p(\mathcal{A}) + \sum_i \alpha_i h_i(\mathcal{A})$$

Note that we will take $\alpha_i > 0$ when we want to increase the heuristic, and $\alpha_i < 0$ when we want to decrease it.

- First, we want to know if the letters in our alignment are close to being aligned to the guide or not (Figure 7). To that end, we want to define a distance from our alignment to the guide, which our greedy player will aim to decrease in order to find better alignments. We propose the following function as our measure :

  Distance to guide :

  For an alignment $\mathcal{A}$: Let's write each aligned sequence S^i as $s^i_1 \dots s^i_m$, and define the distance $d(s^i_j, G)$ of the letter s^i_j to the guide G as :



(a) Worse case

A letter is two gaps away from being aligned, another one is one gap away

(b) Better case

Two letters are one gap away from being aligned

Figure 7: Comparing two alignments distance to the guide

- $d(s_j^i, G) = 0$ if $s_j^i =$ "_".
- $d(s_j^i, G) = \min\left\{\left\{k \in \mathbb{N} \middle| s_j^i \in g_{j+k}\right\} \bigcup \{m+1-j\}\right\}$ otherwise.

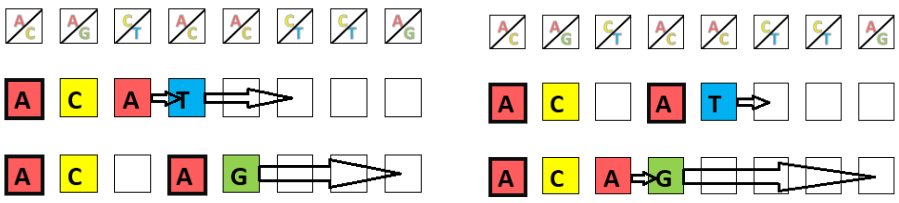
The idea here is to count minimal the number of gaps that would need to be used in order to align letter s_j^i to the guide (i.e. the number of spaces that would need to be moved from the end of the sequence to before s_j^i).

Then, we define the overall distance of the alignment to the guide as :

$$d(\mathcal{A}, G) = \sum_{i,j} d(s_j^i, G)$$

Discounted distance to guide :

It might be more interesting to give more relative importance to short distances, since these are the letters that we are more likely soon to be aligned (Figure 8). For that purpose, we propose the following definition of a new discounted distance :



(a) Worse case

A letter is two gaps away from being aligned, another one is three gaps away

(b) Better case

A letter is one gap away from being aligned, another one is four gaps away

Figure 8: Comparing two alignments distance to the guide

- $d'(s^i_j, G) = 0$ if $s^i_j =$ "_".
- $d'(s^i_j, G) = \sum_{k=1}^{d(s^i_j,G)} \lambda^{k-1} = \frac{1-\lambda^{d(s^i_j,G)}}{1-\lambda}$ otherwise.

Here, the first gap that would need to be used in order to align s^i_j to the guide increases the distance by one, and then the next ones yields decreasing returns. That way, moving an almost aligned letter closer to the guide will decrease the distance more than moving a letter far from being aligned closer to the guide.

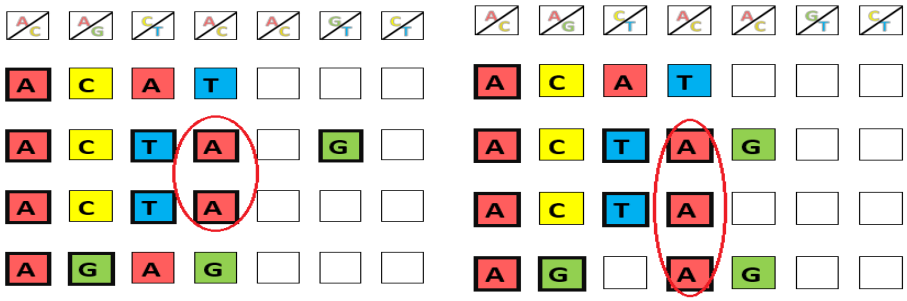
Then, we define the discounted distance of the alignment to the guide as :

$$d'(\mathcal{A}, G) = \sum_{i,j} d'(s^i_j, G)$$

Note that for both of these definitions, the distance of a letter s^i_j that cannot be aligned by adding gaps anymore is misleading, so it might be interesting to change the definition to consider this specific case.

- Now, we also want to encourage the player to prioritise alignments that achieve bonus on multiple positions (Figure 9). To that end, we can

look for each position at how many letters are already aligned, and use a function f that will yield increasing return for this number of letters (i.e. we want $f : \mathbb{N} \to \mathbb{R}$ such that $\forall n \in \mathbb{N}, f(n+2) - f(n+1) > f(n+1) - f(n)$). Any strictly convex, increasing function f is suited to that purpose. Hence, we will try to increase the value given by this function on our alignment in order to get closer to achieving a bonus.



(a) Worse case
A position has two gaps aligned

(b) Better case
A position has three gaps aligned

Figure 9: Comparing two alignments proximity to achieving a bonus

Here are some examples of such functions :

- Sum of pairs

  Sum of pair is a common score measure used in Multiple Sequence Alignment.

  For a given alignment $\mathcal{A}$, we define the sum of pairs of letters that are correctly aligned as

  $$f(\mathcal{A}) = \sum_j \binom{2}{p_j}$$

  where $p_j = |\{i | s_j^i \in g_j\}|$.
- Power function : $f : n \in \mathbb{N} \mapsto n^k \in \mathbb{R}$ with $k > 1$.
- Exponential : $f : n \in \mathbb{N} \mapsto e^n \in \mathbb{R}$.

2.2 Results

2.2.1 Datasets

The puzzles we use will concern the DNA sequences of gut bacteria. They are originally made from PASTA alignments, which are quite dense. From these original alignments, we select short parts that we feel could be improved. From this part, we remove the gaps in the middle of the sequence, and we create a guide according to the most represented letter at each position on the original PASTA alignment. The size of the part selected to make a puzzle will vary depending on the desired difficulty we want to set, going from 6 to 19 sequences of around 3 to 10 nucleotides.

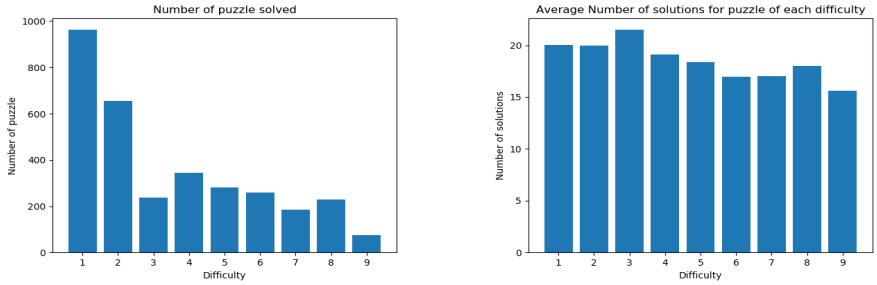
Difficulty	1	2	3	4	5	6	7	8	9
Number of sequences	6	6	7	9	11	15	17	19	19
Size of the guide	6	7	11	11	11	11	11	11	11

To set the par score, we use the following algorithm : First, we take the first sequence of the family, and we try any possible alignment for this one sequence, keeping the one providing the best overall score (without moving the other sequences). Then, we move on to the next sequence of the family and align it the same way. Once we went through all of the sequences, if no modification have been made in this loop (i.e. if none of the sequences have been moved), then we terminate the algorithm and keep the score of the current alignment as our par score, with the number of gaps allowed being the number of gaps used in this alignment. Otherwise, we repeat the steps from the beginning, starting with the current alignment.

Note that this algorithm do not provide any control on the number of gaps used for the alignment. In order to create a new puzzle for the same family of sequences, but with modified par score and maximum number of gaps, we proceed in a greedy way : From the alignment obtained with the previous algorithm, we check all of the gaps that are used and try removing

them. Then, we actually remove the gap such that the score decrease for the alignment is as small as possible. We can repeat this as many time as needed to remove any number of gaps.

We select a random batch of puzzles on which we will do our study. Notice that the number of puzzles we have decrease as the difficulty increase (Figure 10.(a)). Still, we have about the same number of solutions per puzzle regardless of the difficulty, so this shouldn't be an issue (Figure 10.(b)).



(a) Number of puzzles (b) Average number of solutions

Figure 10: Number of puzzles and solutions in the selected batches

Our test will be run on around hundreds of puzzles selected randomly among the puzzles that have been deployed in the game. For each puzzle, we have access to a varying number of solutions that have been found by human players. For each of these solution, we can see the score achieved, the number of gaps used, the actual alignment found, and the player id among other.

2.2.2 Results

Let's first take a look the average score achieved by each greedy players. For that purpose, we selected random batches of up to 250 puzzles for each difficulty.

To restrict the number of parameters, we will only consider an objective function h^1 combining the distance to the guide d and the sum of pairs f, and an objective function h^2 combining the discounted distance to the guide d' and the sum of pairs f. The coefficients are chosen so that we can expect that the variation of the score and of each heuristic will be of comparable magnitude. We report these results in Figure 11.

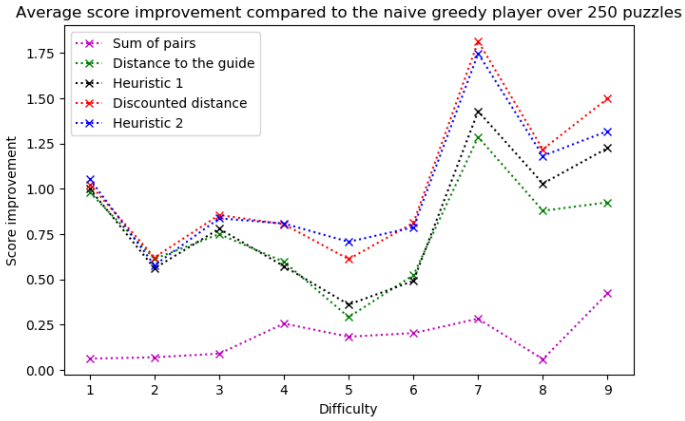
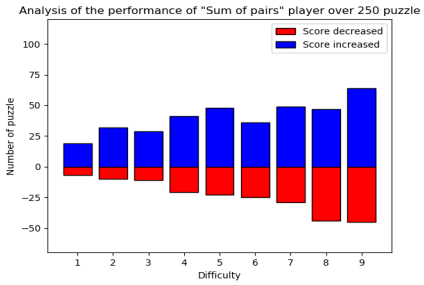


Figure 11: Comparison of the average performance of different greedy players

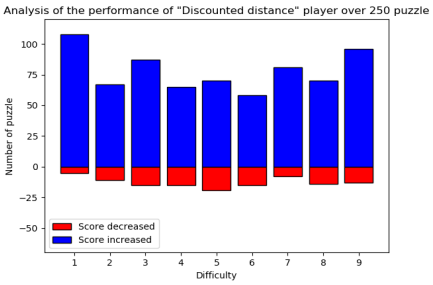
We note that both heuristics tend to improve the result of the greedy player. The most important one seems to be the heuristics regarding the distance to the guide. It looks like this improvement tends to be more important the higher the difficulty of the puzzle is.

However, if we look more in detail, we can see that there is a non negligible number of puzzles where the naive greedy player performs better than the new ones.

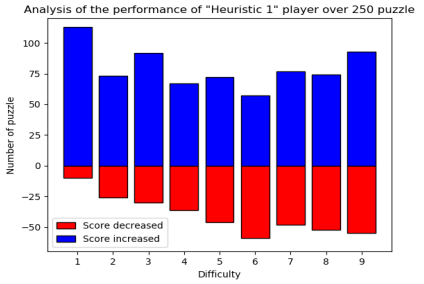
From Figure 12, we can note that the "Sum of pairs" measure have a bigger effect the higher the difficulty is, changing the score of more different puzzles (whether it is improving it or not). In contrast, the distance to the guide variance seems to correlate less to the difficulty of the puzzles.



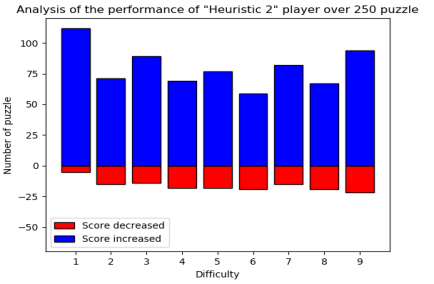
(a) Sum of Pairs



(b) Discounted Distance

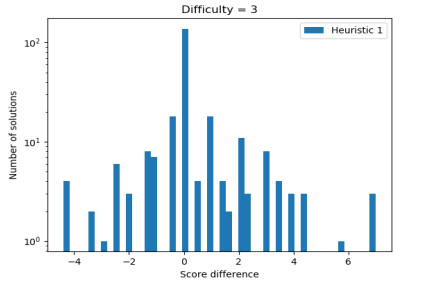


(c) Heuristic 1

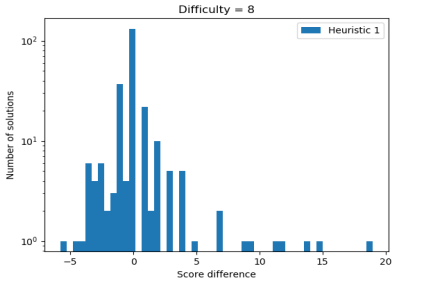


(d) Heuristic 2

Figure 12: Number of puzzles improved by each heuristic



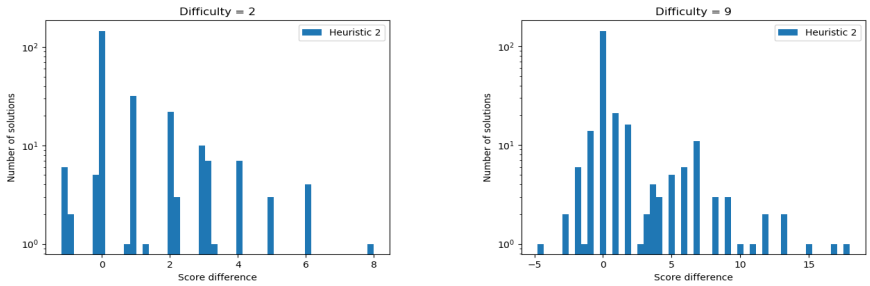
(a) Difficulty 3



(b) Difficulty 8

Figure 13: Score Difference Distribution with Heuristic 1

We can see with Figures 13 and 14 that when the score has decreased, the variation is never very important, rarely being over a 5 point difference. On the contrary, while the increase usually isn't very important either for improved puzzles (often less than a 5 point increase), there can be more important variations, especially when the puzzles difficulty get higher, where we can see increase of more than 15 points.



(a) Difficulty 2 (b) Difficulty 9

Figure 14: Score Difference Distribution with Heuristic 2

In fact, the parameters that gives the best result vary greatly from puzzle to puzzle, as we can see with Figure 15. Therefore if we want to fit the parameters we should train the model on a very large dataset, which would take a long time. Instead, it might be a better approach to try a few different set of parameters on each puzzle and keep the best result.

2.2.3 Time Analysis

We now look at the average time taken by the greedy players to solve a puzzle (Figure 16).

We notice that the greedy players with a changed objective function are significantly slower than the naive greedy player, with the difference being greater the harder the puzzle is. Still, they have a reasonable enough runtime to be used in practice.

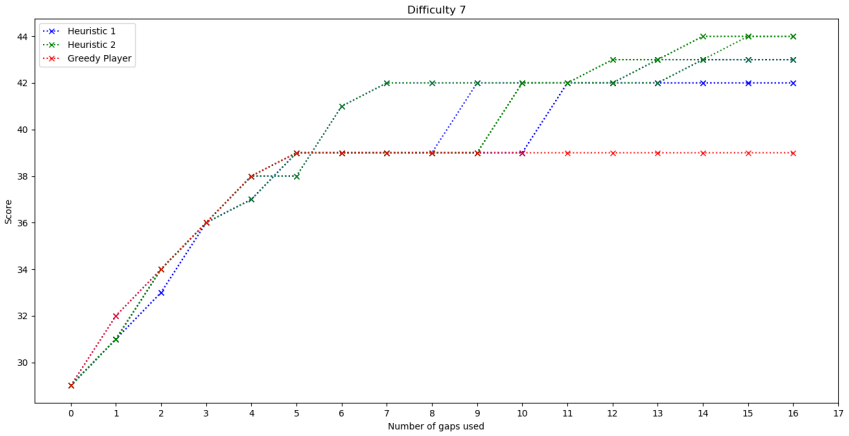


Figure 15: Comparison of the performance of different greedy players

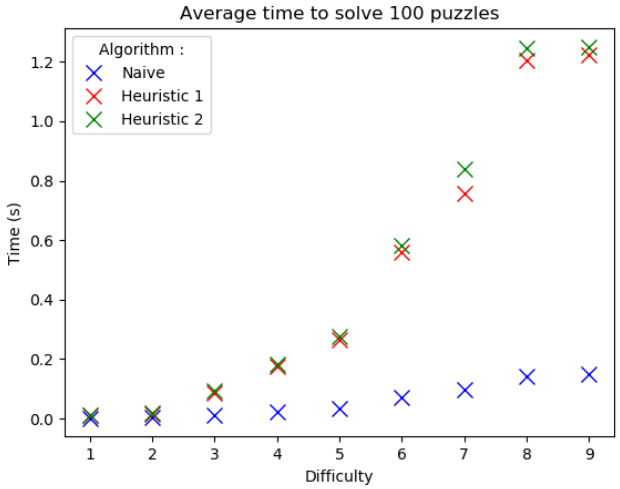


Figure 16: Run time of a Depth Search Player depending on the maximum number of gaps and the depth

Using the discounted distance is slightly slower than using the original one. This should be due to the computation of a power function. Still, this difference is not significant.

As mentioned before, this can still be seen as a problem if we want to search for the best parameters. Indeed, we don't have any better way than to try different combinations of parameters on a batch of puzzles and choose the best performing one. Since we don't have a closed form for the solution, we can't use any gradient based method for example.

2.3 Comparison with Players

2.3.1 Comparison with the Naive Greedy Player

We begin by comparing the human players' solutions to the naive greedy player.

Difficulty	1	2	3	4	5	6	7	8	9
Number of puzzles (Set of sequences)	963 (321)	654 (218)	237 (79)	345 (115)	282 (94)	260 (89)	186 (62)	228 (76)	93 (37)
Number of solutions	19158	12961	5075	6534	5130	4351	3119	4048	1398
Percentage of players beating the naive greedy player	50.2	44.8	19.1	15.8	15.0	13.1	12.8	10.3	15.1
Percentage of set of sequences improved	88.2	89.0	77.2	75.7	56.4	55.1	50	61.8	64.9
Average improvement	3.08	2.96	2.05	2.47	2.91	4.22	4.71	3.01	3.81

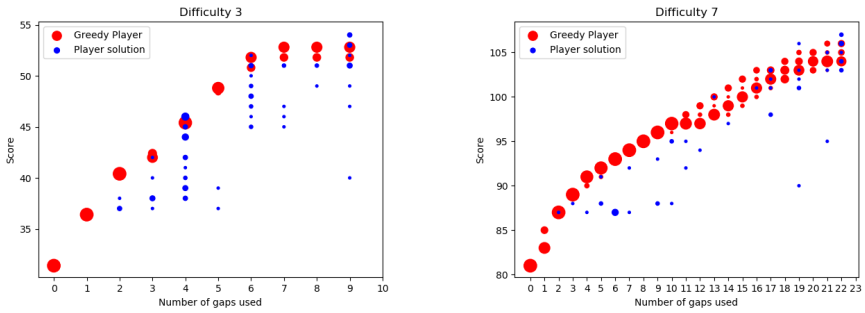
We can see that a significant portion of the human players beat the naive greedy player, especially on lower difficulties. However, we can see that the magnitude of improvement isn't too great.

We can also note that human players have a harder time beating the naive greedy player on puzzles with lots of nucleotides, and when every part of the

guide contains two letters. Players that outperform the naive greedy player also tend to use more gaps.

2.3.2 Score Comparison

It is quite hard to evaluate the quality of these algorithm since we do not know of any efficient ways of computing the optimal solution. Instead, let's first compare the score achieved by the greedy players to the human solutions. We used a total of 25 different combinations of parameters chosen arbitrarily for this comparison. We show our results in Figure 17.



(a) (b)

Figure 17: Example of score comparison on different difficulties

First, we can see that score variations for the greedy players are quite small compared to the human solutions. Most of the players are unable to beat the best score found by the greedy players. In fact, there are numerous puzzles where the greedy players outperform all of the human solution in term of score. This makes us feel that these algorithms perform quite well on the problem. We believe we can assume that these solutions are quite close to the pareto front.

Still, we can see a non negligible number of solutions with a better score than those found by the greedy players, proving that they do not always give

the optimal solution. Moreover, on basically every puzzles, human players are able to at least come very close to the best score computed by the greedy players. It is also interesting to see that these "good scores" are achieved with different number of gaps, which means that the human players can perform well regardless of the number of gaps available. This is promising as we believe that the solutions that are close to the pareto front are the ones that will be worth investigating, Indeed, we expect the solutions that will improve the original alignment to have good Borderlands Alignment score, and therefore to be nearly pareto optimal.

Finally, we can also note that on puzzles of lower difficulty, the greedy players tend to find a local optimum and do not use all of the available gaps, while human players are able to find solutions that use more gaps.

2.3.3 Distance

On another topic, we want to see if our algorithms are able to replicate the behaviour of the human players. To do that, we wish to measure how similar the solutions obtained by our algorithms are to the solutions given by human players, and for that purpose, we must define a distance on the space of possible alignments. Knowing that we only wish to compare solutions of the same problems, which implies that we will only be considering alignments of the same set of sequences and of the same size, an intuitive way to evaluate the distance between two solutions is to look at the number of common positions :

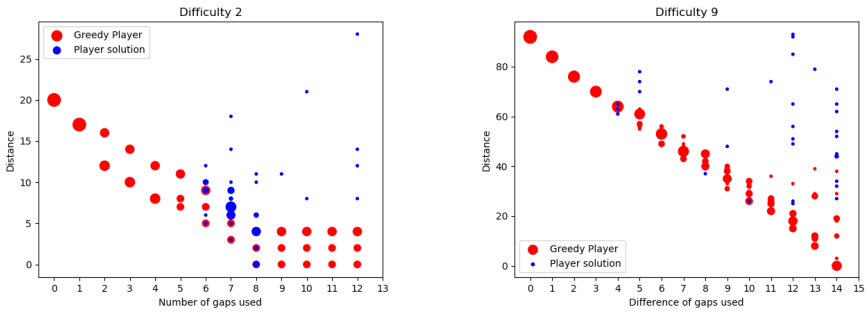
Let $\mathcal{A}$ and $\mathcal{A}'$ be two alignments of the same puzzle. Let's write $\mathcal{A}$ as $(S^1, \dots, S^n)$ where $S^i = s^i_1 \dots s^i_m$, and similarly, $\mathcal{A}'$ as $(S'^1, \dots, S'^n)$. We define the distance between $\mathcal{A}$ and $\mathcal{A}'$ as :

$$d(\mathcal{A}, \mathcal{A}') = \sum_{i,j} \mathbb{1}(s^i_j \neq s'^i_j)$$

where $\mathbb{1}(s_j^i \neq s_j'^i) = 1$ if $s_j^i \neq s_j'^i$ and $\mathbb{1}(s_j^i \neq s_j'^i) = 0$ otherwise.

In other word, we count the number of positions where the characters are different.

Now, we take the best scoring alignment that we found using the greedy players, and measure the distance from each of the solutions we have to that alignment. We choose to only compare to the best alignment as we felt that it was easier and clearer. Some examples are shown in Figure 18.

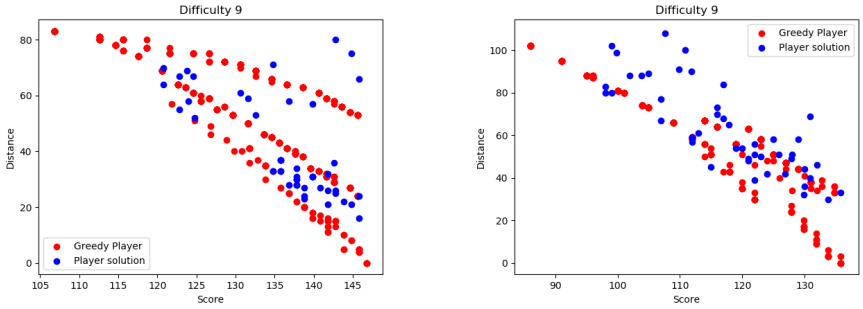


(a) (b)

Figure 18: Distance to the best solution in function of the number of gaps used

First, we see that greedy players tend to produce very similar solutions, regardless of the difficulty of the puzzle, with the alignments getting closer to the best one as more gaps are used as we could have expected given how they were defined. On the other end, for human players, we can see that while on lower difficulties, they are able to find solutions close to the one provided by the computer, it is very rarely the case on higher difficulties. Moreover, on any of those puzzles, human players are always able to find solutions that are vastly different to those found by the greedy players. It is especially interesting to see that we can often find high scoring alignment that are widely different from the one used as a baseline, as we can see in Figure 19, which shows there are other solutions worth considering.

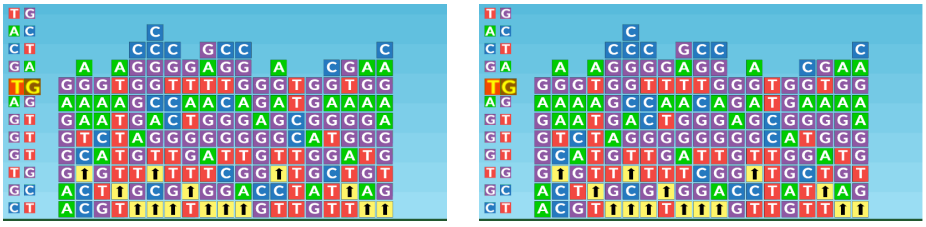
Most importantly, human players provide us a large array of different solutions which we can consider to usually be pretty reasonable ones. Hence, we can hope that some of these solutions will lead to better global alignment when we re-inject them in the global problem from which the puzzles were originally constructed.



(a) (b)

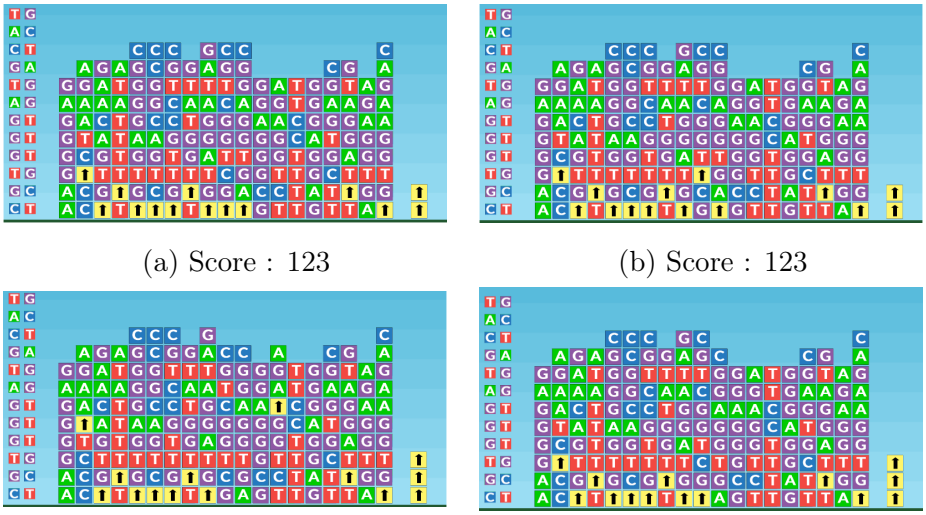
Figure 19: Distance to the best solution in function of the score

Example



(a) (b)

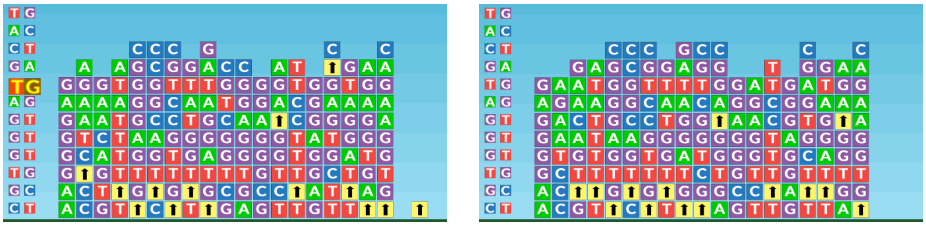
Figure 20: Best solution found by the greedy player (Score : 129)
Here each column corresponds to a sequence, the guide is on the left, and letters start of the bottom in the initial alignment.



(a) Score : 123 (b) Score : 123

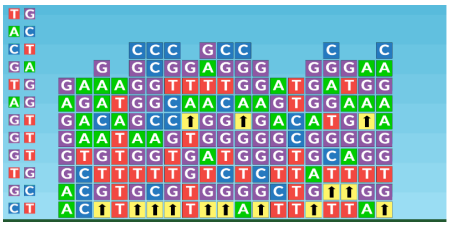
(c) Score : 121 (d) Score : 120

Figure 21: Some good scoring solutions found by the greedy player

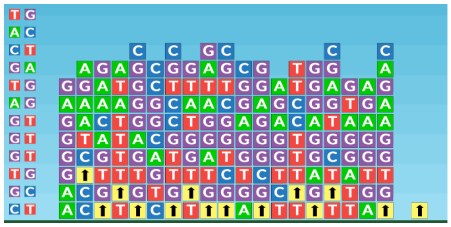


(a) Score : 126

(b) Score : 124



(c) Score : 121



(d) Score : 120

Figure 22: Some good scoring solutions found by the human players

We can see that while greedy players tend to insert gaps at similar position each time, human players tend to explore more diverse options.

2.3.4 Improving solutions

We now want to check if there are easy ways of improving the players' solution. More precisely, let's see if we can find moves (as defined in 1.2) that could improve the score without having to use more gaps. To do that, we can use the naive greedy player, fixing the gap constraint to the current number of gaps used. For each puzzle, we compute the percentage of solutions that we managed to improve in this way. We will refer to solutions that cannot be improved by a single move as local optimum.

From Figure 20, we can see that while on lower difficulty (1 or 2), human players tend to frequently find local optimum (at least 80% of the solutions for about half of the puzzles), that is not the case on puzzles of difficulty of 3 or higher. This might be due to the significant increase in the size of

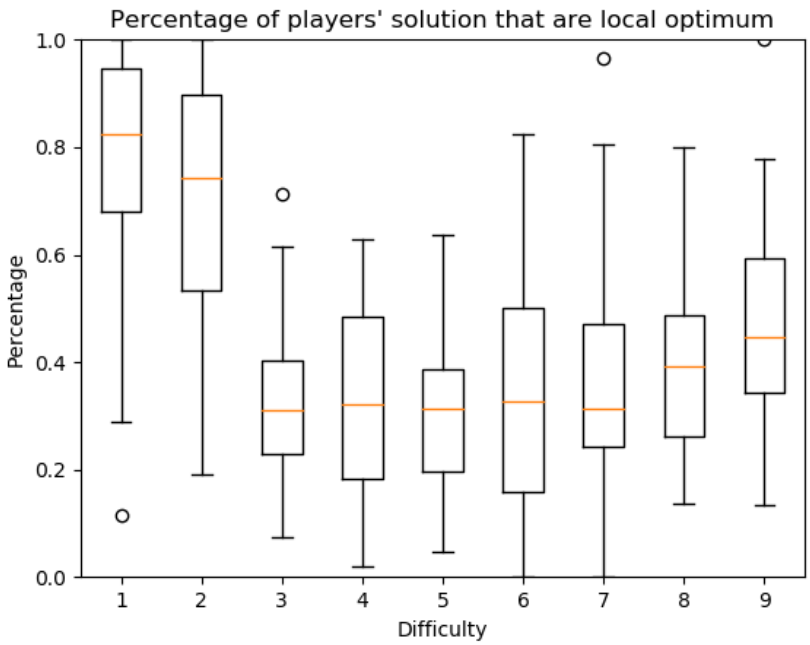
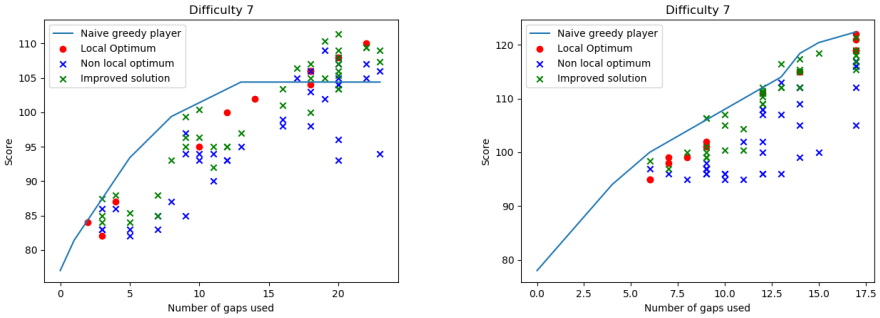


Figure 23: Analysing players' solution

For each difficulty level, for 50% of the puzzles, human players give a percentage of local optimum above the red line. 50% of the puzzles also have this percentage included in the white box



(a) (b)

Figure 24: Improving players' solution

The blue line represents the result obtained by the naive greedy player, the red points are players' solution that are local optimum, the blue points are players' solution that are not local optimum, and the green points are the result obtained by improving these latter solutions.

the problem between difficulty 2 and 3. In the latter case, we can find at least 40% of the solutions that can be improved by the naive greedy player on the vast majority of the puzzles. This strengthen our belief that human players are not usually focused on only maximising the score by any means, and tend to select solutions that feel more natural to them.

Looking closer at the scores achieved (Figure 21 for example), we can see that the local optimum are close to what we assume as the pareto optimal line (regarding the score versus number of gaps trade-off). Hence, improving these solutions allow us to remove solution that were far from being pareto optimal. We can also see that improving players' solution sometimes outperforms the original naive greedy player. That is notably the case when the naive greedy player gets stuck in a local optimum and is unable to use all the gaps available. Indeed, improving a previous solution allows us to use more gaps which can help us get a better score.

Another approach that might be interesting would be to try to combine

some parts of different solutions to form a better one, though this might not be easy to do given that we have a constraint regarding the number of gaps that a solution can use.

3 Discussion

3.1 Contributions

The first objective of our work was to formally define the problem that we are considering, and see if this problem was interesting to study or not. Since we were able to prove that the Borderlands Alignment problem is NP-complete, this justify trying an approach such as Citizen Science in order to solve the problem. Indeed, we know that it is highly unlikely that we will find an efficient algorithm that will provide us with the best possible answer. Moreover, we have also shown the inadequacy of a brute-force approach, as the number of accepted combinations is too high to be studied efficiently. Hence, we can hope that human players will help us find some optimal or close to optimal solutions that we will be able to use to improve the original alignment.

In the second part of our study, we wanted to compare the behaviour of players to some algorithm that could be used to solve the Borderlands Alignment problem. To that end, we defined some heuristics so we could define some algorithms that would choose their moves in a way that we believe is reasonable.

While our algorithms usually gets better score than the human players, we can still find some submissions that outperform the greedy players solution. Moreover, the help of human players provide us with a wide array of solutions which we can use to find vastly different "good" solutions (solutions that are almost optimal), whereas the greedy players tend to find very similar results. This might allows us to restrict the solution space to a smaller set that we can study in the hope of getting the best result when we re-inject it in the original problem, while still keeping a sufficient variety of different solutions.

3.2 Limitations

The main issue with our algorithmic approach comes with the high number of possible parameters coupled with the fact that we lack a way of efficiently comparing their performance. Hence, in order to fit the parameters of our model, we have to evaluate the results on the whole dataset with each combination of parameters that we want to consider, which will take an important amount of time for large dataset. Moreover, since the performance of a given greedy players is highly volatile from puzzle to puzzle, we would like to to keep different combinations of parameters and run them on all puzzles. Therefore we get a trade-off between the number of greedy players that we will want to run and as such the time complexity of our algorithm, versus the quality of our answer.

Moreover, since we are using a greedy approach, our algorithms are naturally prone to getting stuck in local optima instead of reaching the global optimum. Still, this doesn't appear to be too much of a problem since we can see that human players do not find solutions with a tremendously better score.

Another point of contention is the way we compute the similarity between solutions. Indeed, one of the main caveat of the number of common positions is the fact that the spaces are treated the same way as the other letters, though having spaces at the same position isn't as important as for letters. This method of measuring also does not consider the similarity between the different sequences, and as such, two close to equivalent answer might appear as very distant.

3.3 Future Perspectives

To continue our study further, we could look a deeper look at the performance of the greedy players with different heuristics. Indeed, in order to

keep our run-time reasonable enough, we restricted the objective function to only include a combination of the distance to the guide measure with the sum of pairs measure, or a combination of the discounted distance to the guide measure with the sum of pairs measure. It would be interesting to see if the same results still hold with other objective functions.

It also might be interesting to consider a non-deterministic algorithm as it is possibly a better way to simulate the human player behaviour. This could give us a wider variety of solutions which is something we lacked. A way to do that would be to consider all the possible moves, and select the one to take randomly giving more weights to the moves that lead to higher value for the objective function. It might also be interesting to consider techniques such as simulated annealing.

The measures defined for our heuristic based greedy players could be used to develop a reinforcement learning framework. Knowing that the actual best solutions for the original global problem might not be the one with the highest score, we might be able to learn some better evaluation criteria from the "good" human players solutions. More precisely, we could try to train a model that would evaluate the state of the puzzle by computing various measures such as the Borderlands score, the sum of pair score, the distance to the guide, etc. . . , and then the model would be able to choose a specific move to play next based on this knowledge.

Conclusion

In the framework of Borderlands Science, we defined a new scoring scheme for Multiple Sequence Alignment suited to make a game that can be played by a wide array of players. We showed that this problem was non-trivial to solve, and that it was still NP-complete like the other commonly used scoring scheme.

We then proposed some simple algorithms to study this problem and showed that they were able to get good results. We started from a naive greedy player, and showed some heuristics that could improve it. From what we have seen in this work, simple algorithms appear to perform quite well on this problem. In fact, the solutions provided by human players never seem to significantly outperform the computer's solution. Still, human players are regularly able to best the high score gotten from these algorithms. Moreover, human players provide a wider variety of solutions, and it looks like their behaviour cannot be reproduced by a simple algorithm, which should prove to be useful for the continuation of the project.

Overall, with this work, we consolidate that the Borderlands Science initiative is interesting as we confirm the hypothesis that the problem we are studying does not appear to be easily solvable by a computer. Moreover, we showed that the players' community is able to provide a different insight to the subject compared to simple algorithms. This is promising as this reinforce our belief that the collective human power can improve alignments by giving intuitive solutions to the smaller puzzles with a spontaneous trade-off between the number of gaps used and the overall score.

Bibliography

References

[1] David Sankoff, Cristiane Morel, and Robert J. Cedergren. "Evolution of 5S RNA and the Non-randomness of Base Replacement". In: *Nature New Biology* 245 (1973), pp. 232–234.

[2] Mathieu Blanchette. "Computation and analysis of genomic multi-sequence alignments". In: *Annual Review of Genomics and Human Genetics* 8.1 (2007), pp. 193–213. DOI: 10.1146/annurev.genom.8.080706.092300.

[3] Cédric Notredame. "Recent evolutions of multiple sequence alignment algorithms." In: *PLoS Computational Biology* 3.e123 (2007).

[4] Serafim Batzoglou. "The many faces of sequence alignment". In: *Brief Bioinform* 6.1 (2005), pp. 6–22. DOI: 10.1093/bib/6.1.6.

[5] Robert C. Edgar and Serafim Batzoglou. "Multiple sequence alignment". In: *Curr Opin Struct Biol.* 16.3 (2006), pp. 368–373. DOI: 10.1016/j.sbi.2006.04.004.

[6] Ari Löytynoja and Nick Goldman. "Uniting Alignments and Trees". In: *Science* 324.5934 (2009), pp. 1528–1529.

[7] Mathieu Blanchette et al. "Alignment Multiple Genomic Sequences With the Threaded Blockset Aligner". In: *Genome Research* 14.4 (2004), pp. 708–715.

[8] Fabian Sievers and Desmond G. Higgins. "Clustal Omega". In: *Current Protocols in Bioinformatics* 48.1 (2014), pp. 3.13.1–16. DOI: 10.1002/0471250953.bi0313s48.

[9] Robert C. Edgar. "MUSCLE: multiple sequence alignment with high accuracy and high throughput". In: *Nucleic Acids Research* 32.5 (2004), pp. 1792–1797. DOI: 10.1093/nar/gkh340.

[10] Cédric Notredame and Jaap Heringa Desmond G. Higgins. "T-coffee: a novel method for fast and accurate multiple sequence alignment". In: *Journal of Molecular Biology* 302.1 (2000), pp. 205–217. DOI: 10.1006/jmbi.2000.4042.

[11] Kevin Liu et al. "SATe-II: very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees". In: *Syst Biol.* 61.1 (2012), pp. 90–106. DOI: 10.1093/sysbio/syr095.

[12] Chuong B Do et al. "ProbCons: Probabilistic consistency-based multiple sequence alignment". In: *Genome Res.* 15.2 (2005), pp. 330–340. DOI: 10.1101/gr.2821705.

[13] Siavash Mirarab et al. "PASTA : Ultra-Large Multiple Sequence Alignment for Nucleotide and Amino-Acid Sequences". In: *Journal of Computational Biology* 22.5 (2015), pp. 377–386.

[14] Kazutaka Katoh et al. "MAFFT : a novel method for rapid multiple sequence alignment based on fast Fourier transform". In: *Nucleic Acids Research* 30.14 (2002), pp. 3059–3066.

[15] Lusheng Weng and Tao Jiang. "On the complexity of multiple sequence alignment". In: *Journal of Computational Biology* 1.4 (1994), pp. 334–348.

[16] *Wikipedia. Citizen Science.* URL: http://en.wikipedia.org/wiki/Citizen_%20science.

[17] *Audubon Society's Christmas Bird Count.* URL: https://www.audubon.org/conservation/science/christmas-bird-count.

[18] David P. Anderson. "BOINC: A Platform for Volunteer Computing". In: *Journal of Grid Computing* 18 (2020), pp. 99–122.

[19] David P. Anderson et al. "SETI@home an experiment in public-resource computing". In: *Communications of the ACM* 45.11 (2002), pp. 56–61.

[20] Maxwell I. Zimmerman et al. "SARS-CoV-2 Simulations Go Exascale to Capture Spike Opening and Reveal Cryptic Pockets Across the Proteome". In: *BioRxiv* (). DOI: 10.1101/2020.06.27.175430.

[21] Kate Land et al. "Galaxy zoo: the large-scale spin statistics of spiral galaxies in the sloan digital sky survey". In: *Monthly Notices of the Royal Astronomical Society* 388.4 (2008), pp. 1686–1692.

[22] Luis Von Ahn and Laura Dabbish. "Designing Games With a Purpose". In: *Communications of ACM* 51.8 (2008), pp. 58–67.

[23] Ramine Tinati et al. "'/Command' and Conquer: Analysing Discussion in a Citizen Science Game". In: *WebSci '15: Proceedings of the ACM Web Science Conference* 26 (2015), pp. 1–10. DOI: 10.1145/2786451.2786455.

[24] Seth Cooper et al. "Predicting protein structures with a multiplayer online game". In: *Nature* 466.7307 (2010), pp. 756–760. DOI: 10.1038/nature09304.

[25] Firas Khatib et al. "Crystal structure of a monomeric retroviral protease solved by protein folding game players." In: *Nature Structural and Molecular Biology* 18.10 (2011), pp. 1175–1177. DOI: 10.1038/nsmb.2119.

[26] Kawrykow Alexander et al. "Phylo : A Citizen Science Approach for Improving Multiple Sequence Alignment". In: *PLoS ONE* 7(3).e31362 (2015). DOI: 10.1371/journal.pone.0031362.

Appendix - Figures

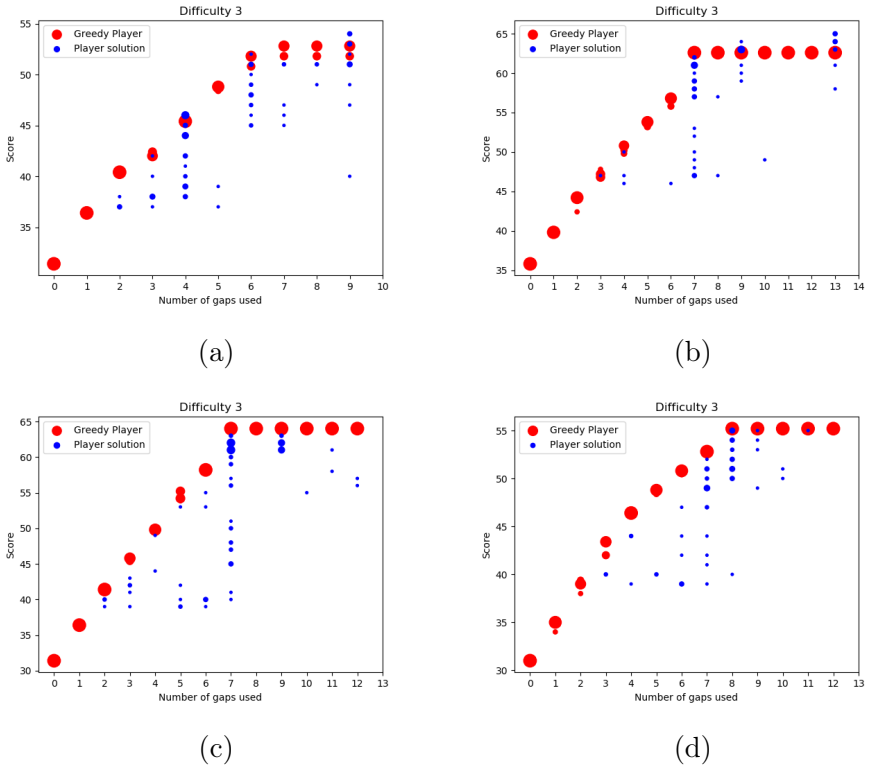
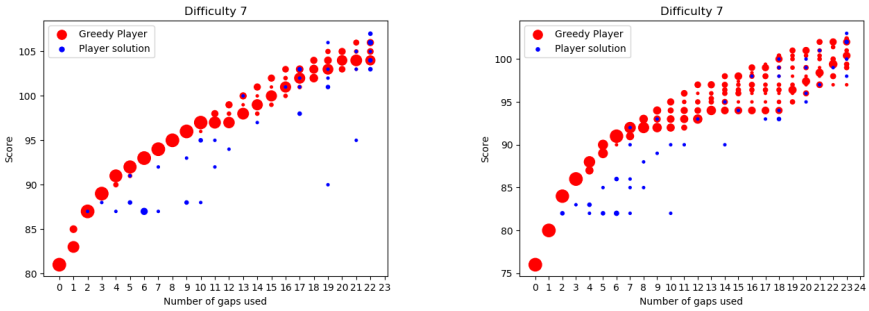
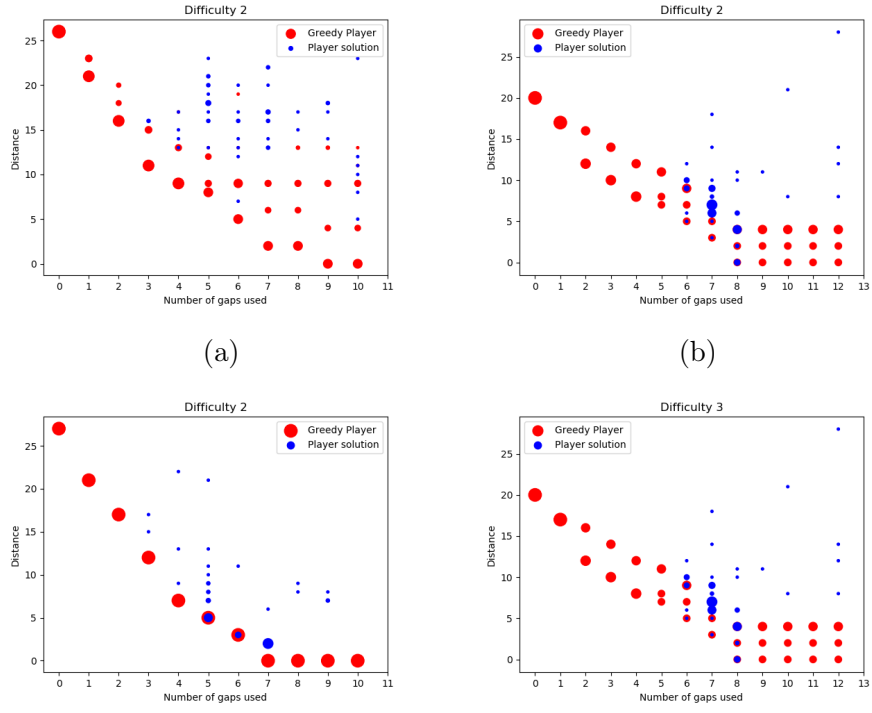


Figure 25: Score comparison - Difficulty 3



(a) (b)

Figure 26: Score comparison - Difficulty 7



(a) (b)

(c) (d)

Figure 27: Distance to the best greedy player's solution as a function of the number of gaps used - Difficulty 2

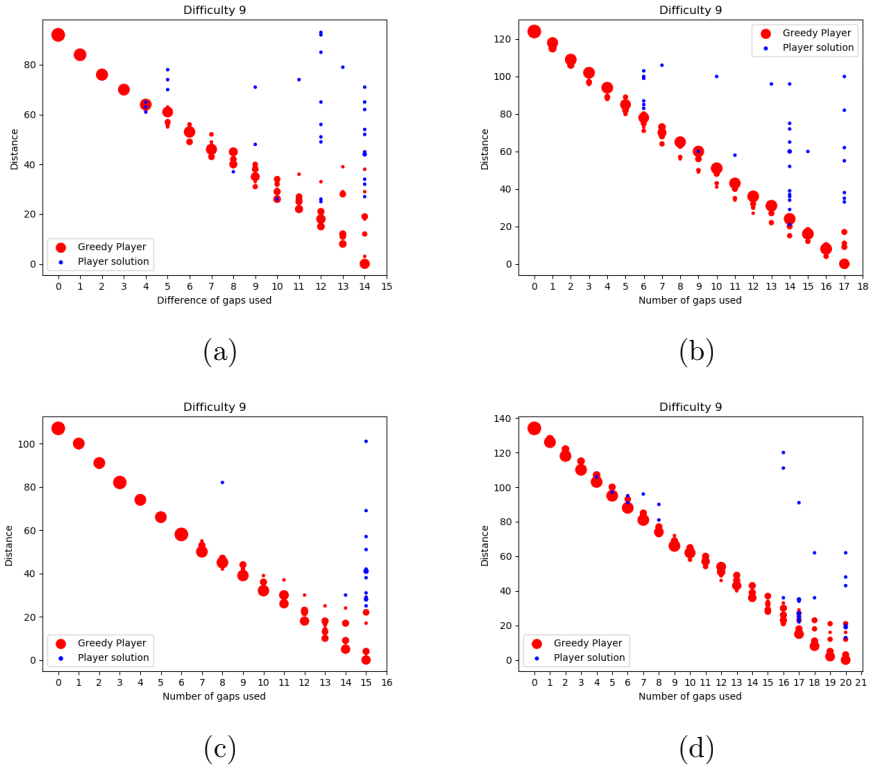
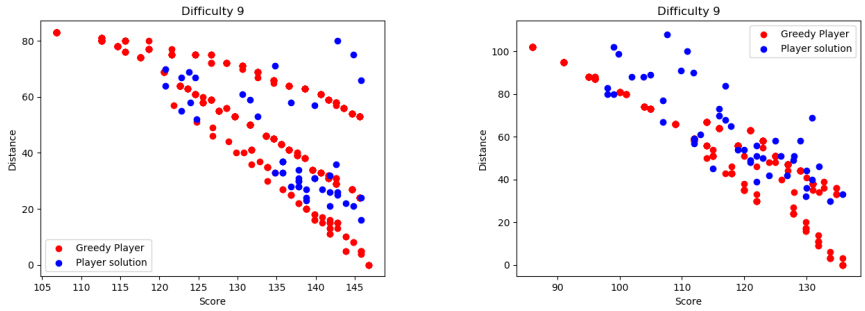
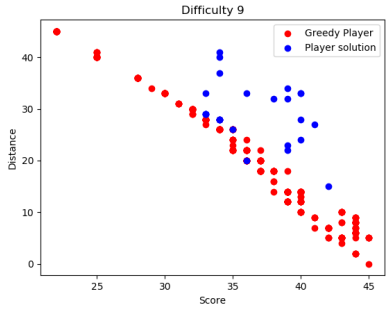


Figure 28: Distance to the best greedy player's solution as a function of the number of gaps used - Difficulty 9

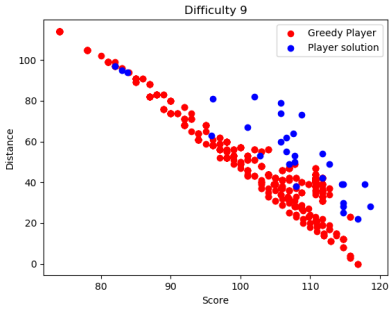


(a)

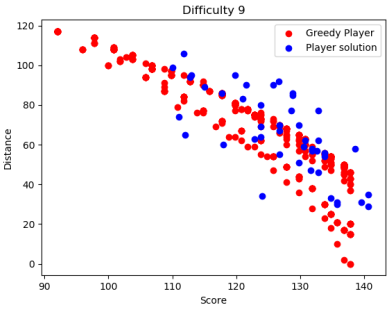
(b)



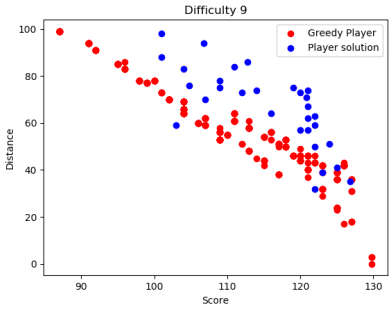
(c)



(d)



(e)



(f)

Figure 29: Distance to the best greedy player's solution as a function of the score

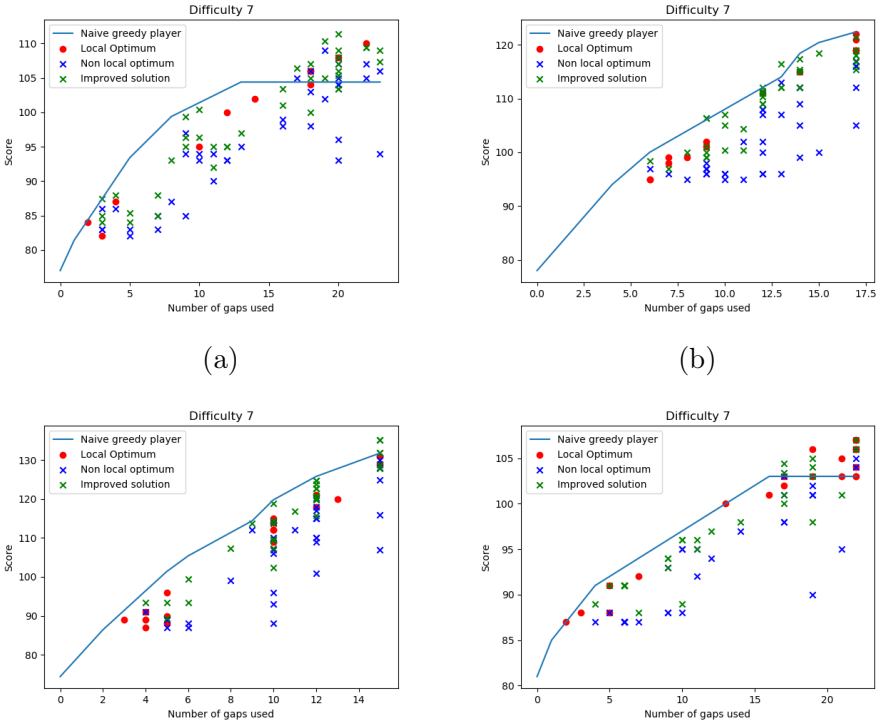


Figure 30: Improving players' solution