D52 - Pattern-Based Queries on Genomic Datasets

Piero Montanari, Ilaria Bartolini, Paolo Ciaccia, Marco Patella

Dipartimento di Informatica Scienza e Ingegneria, Alma Mater Studiorum Università di Bologna

This deliverable introduces a specific operator of the Genometric Query Language (GMQL) introduced in the GenData project. Such operator performs pattern-based queries on genomic datasets, i.e., it provides biologists with the ability, once they identify an interesting genomic pattern, to look for similar occurrences of it in the data. Besides, this deliverable contains details about its implementation and its integration with the genome browsers.

The outline of the deliverable is as follows. Section 2 describes the problem behind our operator and briefly presents the main concepts and section 3 introduces our approach to solve this problem. Then, section 4 describes the algorithm that implements our operator and 5 presents some experimental results. Finally, section 6 draws the conclusions and introduces the future developments for our work.

1 Introduction

Nowadays, the available amount of genomic data is huge and it is growing thanks to the Next Generation Sequencing (NGS) technology. Reading and sequencing the human genome is becoming a relatively fast and inexpensive process. The corresponding potential for data analysis and querying is not a problem equally faced and it poses an important open challenge to the data management community.

At the current state-of-art, NGS data are managed in physical formats and standards that are strongly influenced by the data production process of sequencing machines, and do not offer any high-level view to queried or analyzed. Most of the work in this area is carried out manually by biologists and it can take an enormous amount of time producing poor results. In addition, biologists' observations and analyses are usually limited to specific

portions of the genome which are the target of each biological or medical experiment. These are incredible limitations to the discovery of new interesting biological phenomena (e.g., new promising regions of the genome, or new correlations between biological features) that can not be achieved at a small experiment scale.

The GenData2020 research project was born to address this challenge, enabling an efficient and effective query and analysis process of genomic data.[1]

Within GenData2020, a framework for NGS data storage, management, query, and analysis, is currently under design and development [2]. The two main elements used by this system are a Genometric Data Model (GDM) [1], which encodes experiment results in a format that takes into account the organization of the genome, and specifically its separation into genomic regions, and a Genometric Query Language (GMQL) [1], that uses those as the main data abstractions for extracting regions of interest from experiments and for computing their properties, with high-level operations for manipulating regions and for measuring their distances.

Our main contribution falls under GMQL and, in particular, under the operations that are offered by this language. It is aimed to the design and implementation of a *pattern-search operator* which provides biologists with the ability, once they identify an interesting genomic *pattern*, to look for *similar* occurrences of it in the data, thus facilitating genomic data access and use.

This could be obtained by opportunely integrating the new operator within state-of-the-art *genome browsers*. These tools are instruments that allow to graphically interact with DNA sequences and relative alignments with respect to a reference genome. The integration process will enrich the functionalities offered by current genome browsers with new user friendly visual facilities through which the biologists will draw their queries, or will select patterns of interest, for example, in analogy with the areas of the genome that they are observing, and the browser will indicate other regions with similar patterns. Further, it will be possible to browse through them to find other regions of interests or to generate other queries with different parameters based on the found results.

To this end, it will be needed to define specific interaction metaphors between the genome browser and the biologist, both for purposes of query formulation, together with the correct interpretation of their results, and for purposes of visual navigation of the genome.

For a concrete example of how the new graphical tools will look like, in Figure 1 we depict one possible visual action, i.e., a query pattern selection.

[1] GenData2020 site: gendata.weebly.com/.

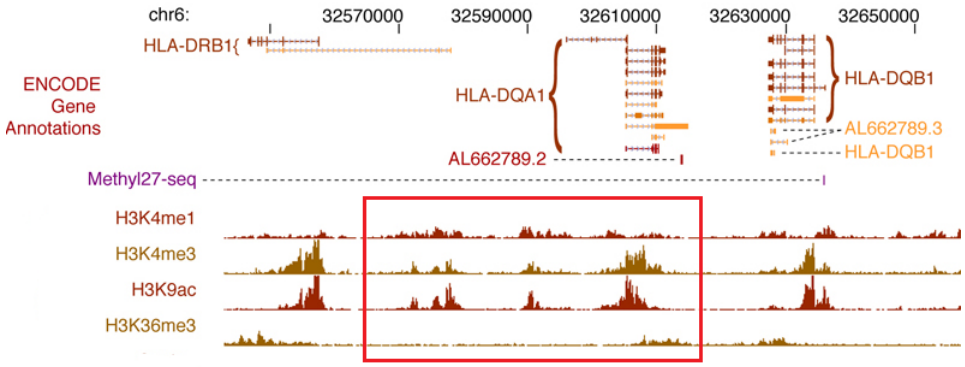

Figure 1: Graphical tool to select the query pattern (the red rectangle individuates the regions the biologist has selected as pattern)

2 The Problem

The search of patterns inside a sequence of data is a well-known problem and several different approaches have been presented in previous works. But, as far as our knowledge, the pattern matching problem has not been faced before in the genomic context.

Our goal is to provide biologists with the opportunity, once an interesting pattern has been detected, to search for similar occurrences in the data. To better understand this purpose, we must first introduce the concept of pattern in this context.

As said above, the key aspect of GDM is the notion of genomic region. A genomic region r is the quadruple $r = \langle chr; left; right; strand \rangle$ where chr is the chromosome number, $left$ and $right$ respectively the first and last ends of the region along the DNA coordinates; $strand$ is the DNA strand, encoded as either $+$ or $-$, and can be missing. We consider part of the region item r_i all the DNA nucleotides whose position is between left and right ends; according to the UCSC notation, we use 1-based inter-base coordinates, i.e. the considered genomic sequence is $[left; right)$. Left and right ends can be identical when the region represents a single nucleotide.

A dataset e is then formed by a collection of genomic regions that are results of a specific experiment. Every region r_i that belongs to a dataset is associated to a region value d_i, that depends on the experiment and that can be seen as a set of features. The dataset e is also characterized by its metadata $m(e)$ which is a set of attribute-value pairs. This metadata contains, for instance, the univocal id of the dataset. Concluding, a dataset is defined as $e = \langle \langle r_i, d_i \rangle, m \rangle, i = 1, len(e)$.

Our definition of pattern is simply an interesting subset of a dataset, i.e. a list of regions with their associated region values that are results of a

specific experiment (information contained in the metadata). The selection of these regions that form the pattern is realized according to some biological criteria of interest that are irrelevant for our work.

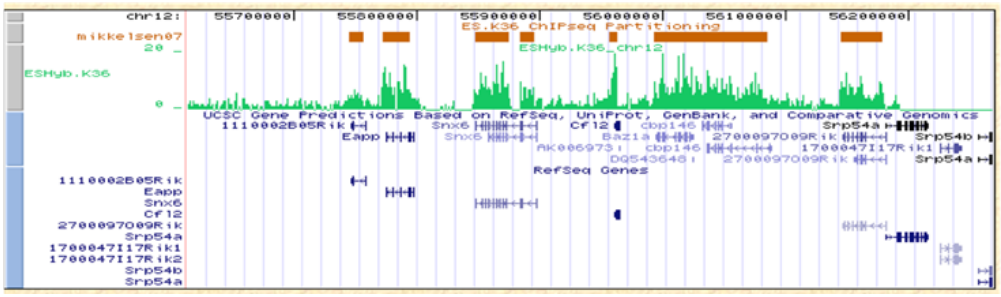

Figure 2: Example of a pattern on a genomic browser

In Figure 2, it is possible to have an idea of how a pattern is represented on a genome browser. While Figure 3 gives a schematic representation of it that is useful to understand the goal of our operator.

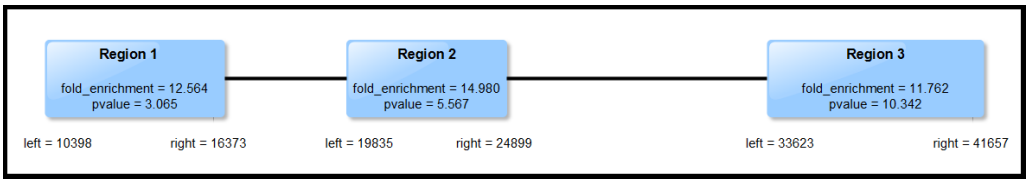

Figure 3: Schematic representation of a pattern

The pattern represented in Figure 3 is composed by three regions that have two features associated ($fold\_enrichment$ and $pvalue$). These regions have different length and different distances and this is shown in the figure. It is now immediate to have an abstract view of what we want to find in the data: ideally, we look for three regions with respectively the same features and with the same relative positions. Usually a perfect match is not possible and, thus, we look for "similar objects" (how similarity is computed is introduced in Section 3).

Formally, the concept we just introduced is called simple pattern p and it is defined as a subset of interest of a given dataset e, $p = \langle\langle r_i, d_i\rangle, m(e)\rangle$ where $i = 1, N$ and $N \ll len(e)$. We assume that every region that composes the pattern belongs to the same chromosome.

Now we need to take into consideration the more general case in which the regions of interest do not belong to the same dataset. For instance, a biologist can analyze different experiments of the same patient, trying to find an analog behavior in the experiments of another patient. Given the number of datasets M from which the regions are taken, we call it complex

pattern P and define it as $P = \{p_j\} = \{\langle\langle r_i, d_i\rangle_j, m(e_j)\rangle\}$ with $j = 1, NP$. We still assume that every region that composes the pattern belongs to the same chromosome.

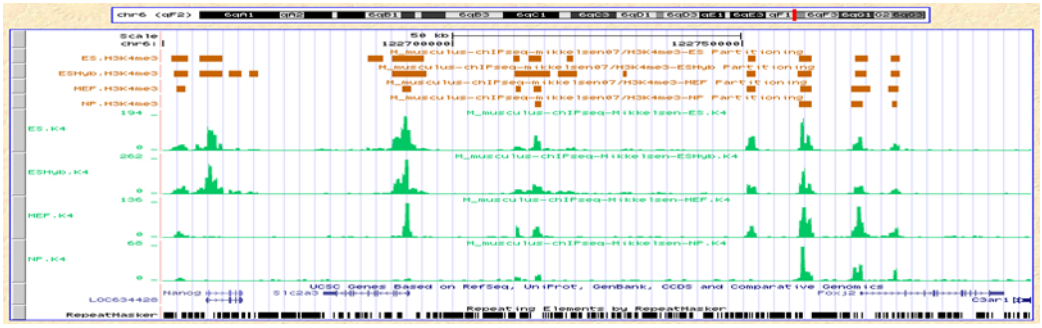

Figure 4: Example of a complex pattern on a genomic browser

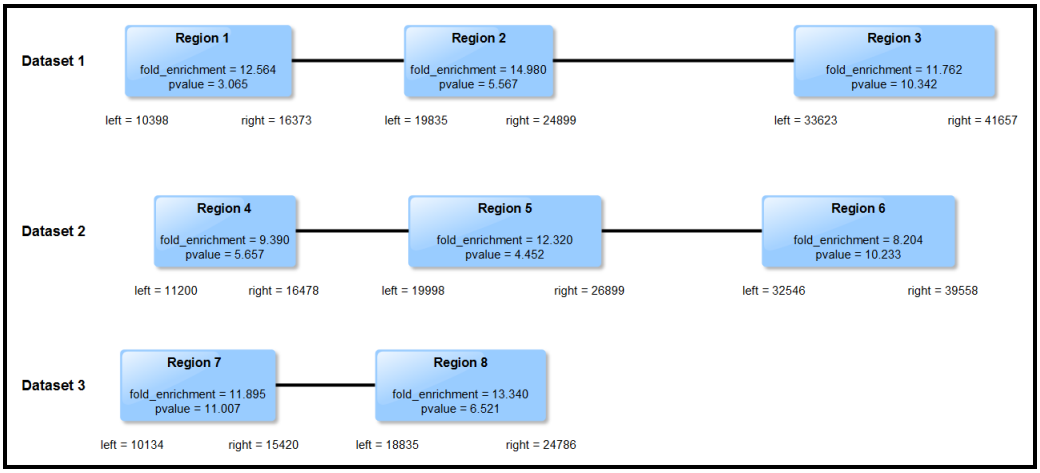

Figure 5: Schematic representation of a complex pattern

In Figure 4, it is shown a complex pattern and it is possible to see how the regions of interests belong to different datasets (e.g., 4 in this case). While Figure 5, similarly to the simple pattern case, gives a schematic representation of a complex pattern.

In the following section we formally define the problem of finding similar objects in both cases of simple pattern and complex pattern and we present our approach to the problem.

3 Proposed Solution

Our problem can be somehow considered as a matching problem of a sequence (pattern) into a bigger sequence (dataset), but there are some com-

plications with respect to the base problem, that has been widely studied in literature. First, the regions are range objects, i.e. they can not be considered as points. Second, the regions themselves have features that must be compared and so the best alignment considering the relative position can be not the best match. Finally, the pattern can be complex and, in that case, several different sequences must be searched into different datasets.

To examine this problem, we start with a simplified version of it and then we gradually introduce all the elements of complexity. In doing so, we avoid to write the metadata information unless it is strictly necessary, so as not to burden the notation.

3.1 Simplified Solution

We make the following hypotheses to simplify our problem:

- the regions are reduced to point objects, where the characterizing point of a region r is its center $c_r = (right_r - left_r)/2$;
- the regions similarity, i.e. the region values, are ignored in the computation of the results;
- P is a simple pattern, i.e. it is $P = \{p\}$.

First of all, we need to give some preparatory definitions.

Definition 3.1 (Match) *Given a simple pattern $q = \langle q_i, d_i \rangle$ with $i = 1, N$ and an input dataset $e = \langle r_j, d_j \rangle$ with $j = 1, len(e)$, we define a match as the couple $m_i = (q_i, r_{j(i)})$, i.e. the i-th region of q is matched to the $j(i)$-th region of e.*

Definition 3.2 (Alignment) *Given a simple pattern $q = \langle q_i, d_i \rangle$ with $i = 1, N$ and an input dataset $e = \langle r_j, d_j \rangle$ with $j = 1, len(e)$, we define an alignment M as a list of N regions of e, $M = \langle r_{j(i)}, d_{j(i)} \rangle$, such that $r_{j(i)}$ matches $q_i, i = 1, N$.*

Note that the concept of alignment is similar to the concept of simple pattern, since both consist in a list of regions.

Each possible alignment leads to a cost, $cost(M, q)$, that represents the cost one has to pay to perfectly align M to q. Intuitively, this should take into account only the distances between pattern regions and their matched regions. Given the above hypotheses and these definitions, we can now formally define our simplified version of the problem.

Definition 3.3 (Best-Alignment Problem) *Given a simple pattern $q = \langle q_i, d_i \rangle$ with $i = 1, N$, an input dataset $e = \langle r_j, d_j \rangle$ with $j = 1, len(e)$, a cost function $cost(M, q)$, and the set of all the possible alignments $\mathcal{M}$, we define the best-alignment problem as the problem to find the alignment $M^* \in \mathcal{M}$ that has the minimum cost, i.e. $cost(M^*, q) \leq cost(M, q), \forall M \in \mathcal{M}$.*

We make an another hypothesis here to show a concrete example of a cost function for an alignment. $cost(M, q)$ can be computed as the linear sum of the differences between the distances of pattern regions and their matching regions:

$$cost(M, q) = \sum_{i=1}^{N-1} \sum_{k=i+1}^{N} \left| dist(q_i, q_k) - dist(r_{j(i)}, r_{j(k)}) \right| \tag{1}$$

In this simplified context in which regions are reduced to point objects, the distance between two regions can be computed as the distance between their centers, i.e. $dist(r_i, r_k) = c_{r_k} - c_{r_i}$ with $c_{r_i} < c_{r_k}$. The distance between regions makes sense only if the regions belong to the same chromosome, otherwise it is undefined.

Substituting the distances in equation 1, we obtain the cost formula we use in the following example:

$$cost(M, q) = \sum_{i=1}^{N-1} \sum_{k=i+1}^{N} \left| (c_{q_k} - c_{q_i}) - (c_{r_{j(k)}} - c_{r_{j(i)}}) \right| \tag{2}$$

Example 1 *Given the simple pattern query* $q = \langle 20, 31, 47, 50 \rangle$ *and the dataset* $e = \langle 11, 16, 24, 31, 43, 45, 48, 58, 61, 64, 70, 72, 83 \rangle$*, we want to compute which alignment is the best one, i.e. the alignment with the minimum cost. For instance,* $M = \{(20, 11), (31, 24), (47, 70), (50, 72)\}$ *has cost:*

$$\begin{aligned} cost(M, q) &= |(31 - 20) - (24 - 11)| + |(47 - 20) - (70 - 11)| + |(50 - 20) - (72 - 11)| \\ &\quad + |(47 - 31) - (70 - 24)| + |(50 - 31) - (72 - 24)| + |(50 - 47) - (72 - 70)| \\ &= |11 - 13| + |27 - 59| + |30 - 61| + |16 - 46| + |19 - 48| + |3 - 2| \\ &= 2 + 32 + 31 + 30 + 29 + 1 = 125 \end{aligned}$$

The best possible alignment is $M^* = \{(20, 31), (31, 42), (47, 58), (50, 61)\}$ *which has* $cost(M^*, q) = 0$.

Although we defined the problem as a minimization of the cost function, it is convenient in the real context (for reasons we will introduce below) to define the problem as a maximization of the similarity function, where similarity is defined as:

$$sim(M, q) = \begin{cases} 1 & \text{if } cost(M, q) = 0 \\ \text{tends to } 0 & \text{as } cost(M, q) \text{ increases} \end{cases} \tag{3}$$

3.2 Root-Region Approach

We noticed through experiments that the efficiency of the above approach in computing the costs of alignments is too low and it is not usable in the

real context, in which there is a huge amount of data. The main problem we pointed out is that changing also one single match in the alignment forces to recompute the entire cost function. To overcome this efficiency problem we introduce the concept of "root" region.

Definition 3.4 (Root Region) *Given a simple pattern $q = \langle q_i, d_i \rangle$ with $i = 1, N$, the root region is the first region q_1 of the pattern.*

Now, we use this definition to change the cost function as follows: only the distances related to the root region are taken into account, while all the others are ignored. The new cost function is:

$$cost(M, q) = \sum_{i=1}^{N} \left| dist(q_1, q_i) - dist(r_{j(1)}, r_{j(i)}) \right| \tag{4}$$

The advantage of the root-region approach is that now changing one match m_i of the alignment (except for the root regions match m_1) does not require to recompute the cost function from scratch, but it is needed only to recompute the contribute of that specific match. In fact, it is possible to define the cost of a match as:

$$cost(m_i, q_i) = \left| dist(q_1, q_i) - dist(r_{j(1)}, r_{j(i)}) \right| \tag{5}$$

Equation 4 can now be written as $cost(M, q) = \sum_{i=1}^{N} cost(m_i, q_i)$ and this is an important result to improve the efficiency of our solution.

Now, we need to transform cost values into similarity values. For this purpose, we apply to the cost of a single match a function, called horizontal match function $f_H(\ )$, i.e. $sim(m_i, q_i) = f_H(cost(m_i, q_i))$, where:

$$f_H(cost(m_i, q_i)) = \begin{cases} 1 & \text{if } cost(m_i, q_i) = 0 \\ \text{tends to } 0 & \text{as } cost(m_i, q_i) \text{ increases} \end{cases} \tag{6}$$

The similarity of an alignment can be computed as the average of the similarities of its matches:

$$\begin{aligned} sim(M, q) = & \frac{\sum_{i=1}^{N} sim(m_i, q_i)}{N} = \frac{\sum_{i=1}^{N} f_H(cost(m_i, q_i))}{N} \\ = & \frac{\sum_{i=1}^{N} f_H\left(\left|dist(q_1, q_i) - dist(r_{j(1)}, r_{j(i)})\right|\right)}{N} \end{aligned} \tag{7}$$

It is immediate to derive that this definition of similarity respects the property defined in equation 3.

Among all conceivable horizontal match functions, we advocate the use of the complement of the sigmoid function.

Definition 3.5 (Sigmoid Horizontal Match Function) *The sigmoid horizontal match function* $f_H^{sigm}(\ )$ *is defined as* $1 - sigm(\ )$ *where* $sigm(\ )$ *is the Sigmoid function:*

$$f_H^{sigm}(cost(m_i, q_i)) = 1 - sigm(cost(m_i, q_i)) = 1 - \frac{1}{1 + e^{-sl(cost(m_i, q_i) - mp)}} \tag{8}$$

The rationale behind the use of this function is that it gives the user, by setting its parameters, the opportunity of calibrating it so as to obtain that:

- a relatively small cost leads to a similarity very close to 1;
- all sufficiently high cost values lead to a similarity almost equal to 0.

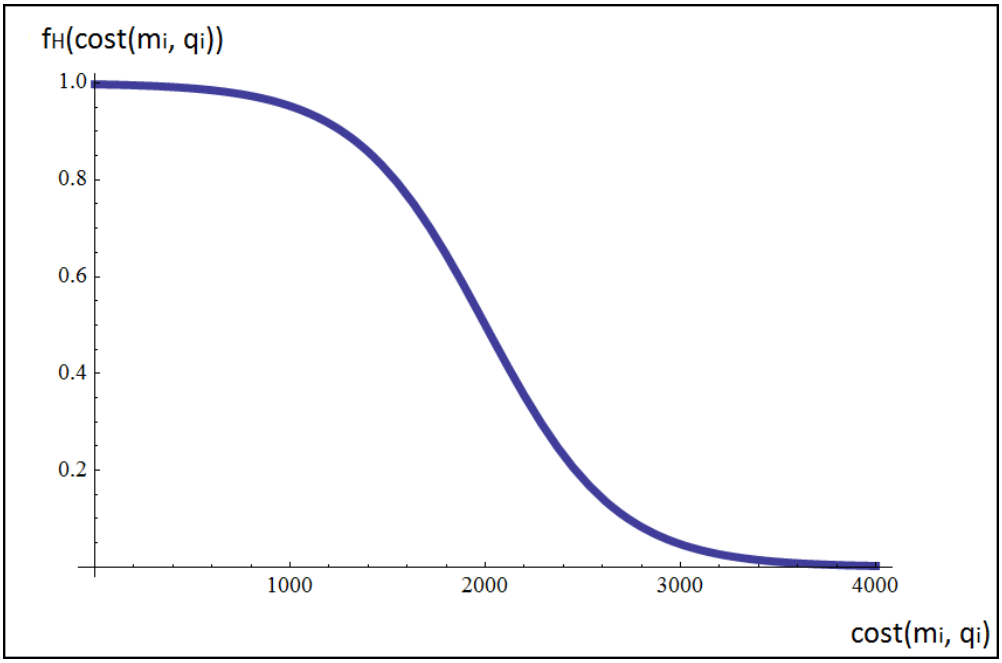

Figure 6: Horizontal match function

Equation 8 is characterized by two parameters, mid-point (mp) and slope (sl), that respectively control in which point the function assumes is middle value and how fast it grows/decreases. Obviously, the choice of these parameters influence the results, as shown in Figure 7, and if they are not chosen in a proper way the results can be meaningless.

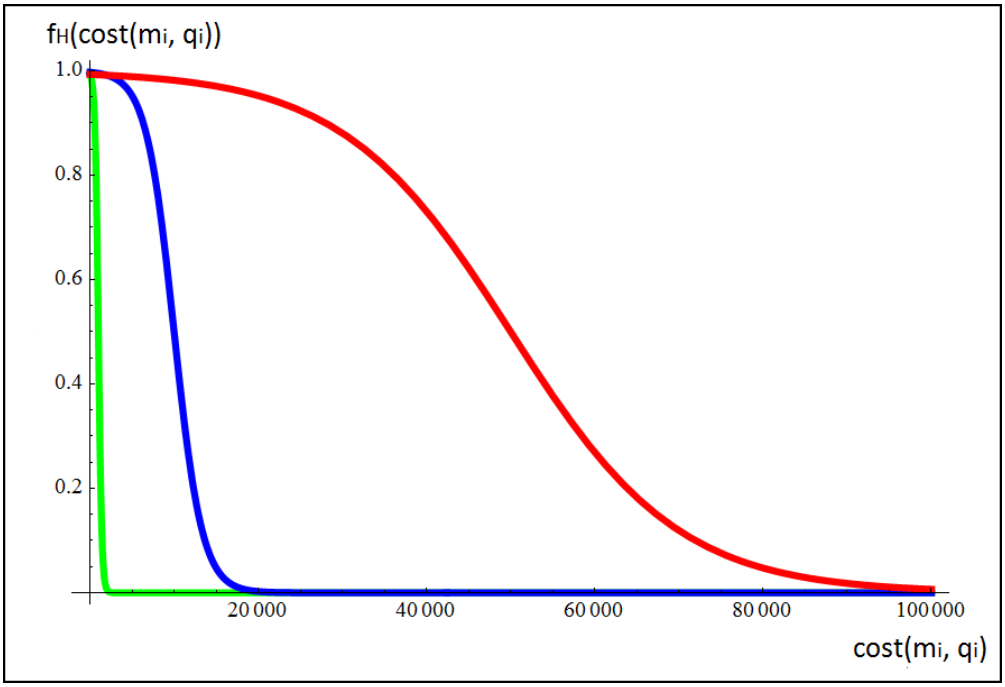

Figure 7: Contribute of parameters in the horizontal match function

3.3 Removing Simplifying Hypotheses

In this section, we gradually remove the hypotheses made in section 3.1 to address the real problem we must face to generate our operator. First, we remove the hypotheses of the cost function computed as a linear sum. Second, we consider regions as range objects and provide a proper definition of distance between them from a biological point of view. Third, we introduce the contribute given by the region values, i.e. the features of the specific experiment. Finally, we face the problem of complex patterns and define a distance between them.

Before going through this, we need to make a general consideration on the problem. In section 3.1 we defined a minimization problem in which the goal was to minimize a cost function. From now on, we address the opposite problem, i.e. we want to maximize a score function that we call similarity function.

3.3.1 Regions as Range Objects

A genomic region is not a point object and, thus, can not be identified as its center. It is defined as a range object whose ends are given by $left$ and $right$ attributes. Therefore, the definition of distance between regions must take it into account and it must also handle the possible overlapping.

Definition 3.6 (Regions Distance) *Given two regions r_1 and r_2 which belongs to the same chromosome (i.e. $chr_{r_1} = chr_{r_2}$), the distance between them $dist(r_1, r_2)$ is computed as $dist(r_1, r_2) = left_{r_2} - right_{r_1}$. While the distance between two regions that do not belong to the same chromosome is undefined.*

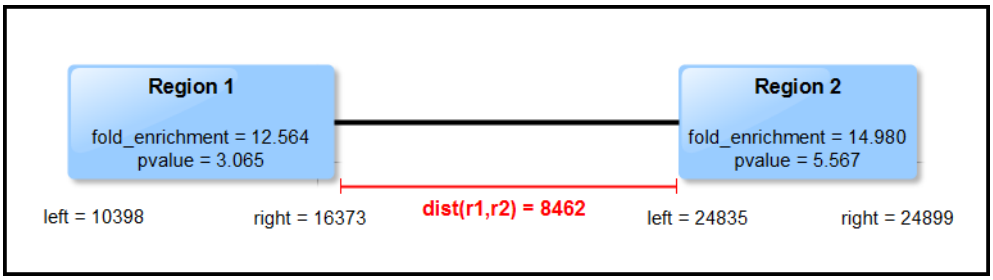

Figure 8: Distance between two regions

It can be deduced from the definition that the distance between two regions is not a commutative function, i.e. $dist(r_1, r_2) \neq dist(r_2, r_1)$. Besides, it must be noticed that this definition handles the overlap of regions. In fact, if two regions overlap, the distance is simply a negative value as it is possible to see in Figure 9.

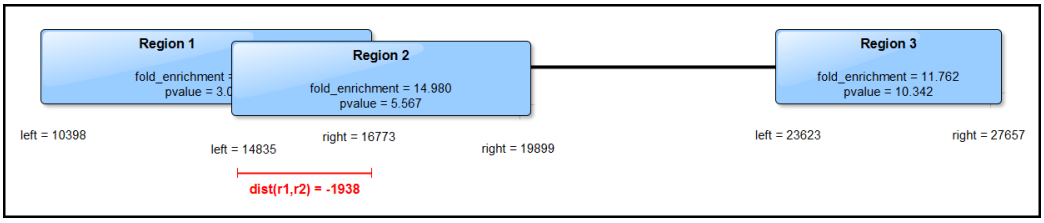

Figure 9: Regions distance in case of overlap of the regions

3.3.2 Region Similarity

As we said before, every region r_i that belongs to a dataset is associated to a region value d_i, i.e. to a set of features. Every feature $a_{i,j} \in d_i$ conveniently normalized between 0 and 1 can concur in the computation of similarity with a different weight, defined by the user through a parameter w_i.

Definition 3.7 (Region Similarity) *Given two regions r_1 and r_2, their region values $d_1 = \{d_{1,j}\}$ and $d_2 = \{d_{2,j}\}$ with $j = 1, L$, and the associated weights $\mathcal{W} = \{w_j\}$ with $j = 1, L$, the region similarity is defined as:*

$$sim_{reg}(r_1, r_2) = \frac{\sum_{j=1}^{L} w_j \cdot |a_{1,j} - a_{2,j}|}{L} \tag{9}$$

Definition 3.8 (Simple Pattern Similarity) *Given a simple pattern* $q = \langle q_i, d_i \rangle$ *with* $i = 1, N$*, an input dataset* $e = \langle r_j, d_j \rangle$ *with* $j = 1, len(e)$*, and a possible alignment* M*, the simple pattern similarity of* M *is computed as:*

$$\begin{aligned} sim(M, q) &= \frac{\sum_{i=1}^{N} sim_{reg}(q_i, r_{j(i)}) \cdot f_H(cost(m_i, q_i))}{N} \\ &= \frac{\sum_{i=1}^{N} sim_{reg}(q_i, r_{j(i)}) \cdot f_H\left(\left|dist(q_1, q_i) - dist(r_{j(1)}, r_{j(i)})\right|\right)}{N} \end{aligned} \tag{10}$$

3.3.3 Complex Pattern

A complex pattern P is a set of simple patterns $P = \{p_j\}$ with $j = 1, NP$ where NP is the number of different datasets involved.

Definition 3.9 (Complex Alignment) *Given a complex pattern* $P = \{p_j\}$ *with* $j = 1, NP$ *and a collection of* NP *datasets* $\{e_j\}$*, a complex alignment* MC *is a set of* NP *alignments defined on* $\{e_j\}$*, one for each* p_j *that forms the complex pattern.*

We first introduce the preliminary definition of distance between patters, which is based on the distance between the regions of the patterns. Due to the analogy between patterns and alignments, the following definition is also applicable to alignments.

Definition 3.10 (Pattern Distance) *Given two patterns* $p_1 = \langle r_{1,i}, d_{1,i} \rangle$ *and* $p_2 = \langle r_{2,k}, d_{2,k} \rangle$ *with* $i = 1, N_1$ *and* $k = 1, N_2$*, the pattern distance* $pdist(p_1, p_2)$ *is defined as the distance between the right attributes of their root regions:* $pdist(p_1, p_2) = right_{r_{2,1}} - right_{r_{1,1}}$.

Note that in the above definitions, only the root regions of patterns p_1 and p_2 are into account. This allows to also extend such definition to a couple of regions, i.e. the root regions rr_1 and rr_2 of p_1 and p_2 (M_1 and M_2, respectively).

It can be deduced from the definition that the distance between two patterns is not a commutative function, i.e. $pdist(p_1, p_2) \neq pdist(p_2, p_1)$. Figure 10 shows an example of computation of pattern distance.

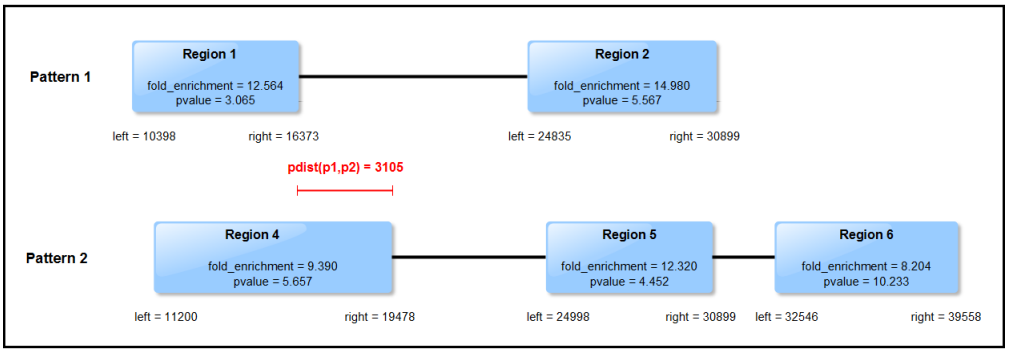

Figure 10: Distance between two patterns

The distances between patterns and alignments are then used to compute inter-pattern similarity. This notion takes into account the fact that the distance between two alignments M_1 and M_2 can differ from the distance between their respective patterns p_1 and p_2 in the complex pattern P. Intuitively, the more the two distances differ, the lower the complex pattern similarity is.

Definition 3.11 (Inter-Pattern Similarity) *Given two simple patterns p_1 and p_2 in a complex pattern P and their respective alignments M_1 and M_2 on two datasets e_1 and e_2, the inter-pattern similarity, $psim(p_1, p_2, M_1, M_2)$, is defined as $psim(p_1, p_2, M_1, M_2) = f_V(|pdist(p_1, p_2) - pdist(M_1, M_2)|)$ where $f_V(\ )$ is the vertical match function that guarantees the following property:*

$$\begin{aligned} psim(p_1, p_2, M_1, M_2) &= f_V(|pdist(p_1, p_2) - pdist(M_1, M_2)|) \\ &= \begin{cases} 1 & \text{if } pdist(p_1, p_2) = pdist(M_1, M_2) \\ \text{tends to } 0 & \text{as } |pdist(p_1, p_2) - pdist(M_1, M_2)| \text{ increases} \end{cases} \end{aligned} \tag{11}$$

For this purpose, as already done in Section 3.2, we exploit the properties of the sigmoid function.

Definition 3.12 (Sigmoid Vertical Match Function) *The sigmoid vertical match function is defined as $1 - sigm(\ )$ where $sigm(\ )$ is the Sigmoid function:*

$$f_V^{sigm}(pdist(p_1, p_2)) = 1 - sigm(pdist(p_1, p_2)) = 1 - \frac{1}{1 + e^{-sl(pdist(p_1,p_2) - mp)}} \tag{12}$$

Now, we can define the complex pattern similarity.

Definition 3.13 (Complex Pattern Similarity) *Given a complex pattern $P = \{p_j\}$ with $j = 1, NP$ and a possible complex alignment $MC =$*

$\{M_j\}$ on a set of datasets $\{e_j\}$ with $j = 1, NP$, the complex pattern similarity is defined as:

$$sim_C(MC, P) = \max_{k=1,NP} \frac{\sum_{j=1}^{NP} sim(M_j, p_j) \cdot psim(p_k, p_j, M_k, M_j)}{NP} \quad (13)$$

The above definition guarantees that the order in which simple patterns are considered does not influence the overall similarity values. Moreover inter-pattern similarities are referred to a unique simple pattern (the one that maximizes the above sum) and this highly improves the computational efficiency.

3.4 Best Pattern-Match Problem

Starting from the above definition of complex pattern similarity, we can ask ourselves, given a complex pattern $P = \{p_j\}$ with $j = 1, NP$, and a collection of NP datasets $\{e_j\}$, which complex alignments obtained on $\{e_j\}$ with respect to P are sufficiently "good". To this end the user provides a similarity threshold θ, $0 < \theta \leq 1$, requiring to output all the complex alignments MC such that $sim_C(MC, P) \geq \theta$. Although the parameter θ is provided by a biologist, it is possible that the number of results is too high to be easily understandable. Therefore, it is common practice to limit the numbers of query results by returning only the "best" complex alignments.

Definition 3.14 (Top-K Best Pattern-Match Problem) *Given a complex pattern query $P = \{p_j\}$ with $j = 1, NP$, a collection of NP datasets $E = \{e_j\}$, a similarity threshold θ, $0 < \theta \leq 1$, and a natural number K, the top-K best pattern-match problem consists in finding K complex alignments, $RES = \{MC_k\}$ with $k = 1, K$, such that $sim_C(MC_i, P) \geq \theta, i = 1, K$ and no other complex alignment $\overline{MC} \notin RES$ leads to $sim_C(\overline{MC}, P) > sim_C(MC_i, P), i = 1, K$.*

3.5 Diversity

The top-k problem faced in the section above opens a new problem: two different complex alignments can differ one another for a single region of a single alignment and if they both are part of the results, the second one does not add any knowledge for the biologists that analyze the results. Thus, we introduce the concept of diversification of the results.

To diversify the results we add the constraint that in the $(i+1)$-th result must not compare any region r that already appears in one of the previous i results. This forces the results to be in different areas of the genomic sequence and helps the biologists to get as much information as possible from the K results.

4 Algorithm

In this section we present the algorithm we use to compute the top-K results for the problem presented in section 3. In details, we introduce the idea of using *bounds* to cut the search of relevant results.

Note that, the use of the greedy approach, that, given a complex query pattern and the input datasets, generates *all* the possible complex alignments and returns the K best results, is not feasible due to the amount of biological data we have to handle.

4.1 Bounds Definition

In this section we introduce two different bounds used by the algorithm to execute some cuts during the search for good results: root-regions bound and complex alignment bound. First of all, we must state that our algorithm proceeds through two main nested loops: the outer loop considers one by one each dataset, while the inner loop considers each region of the specific dataset as current root region. Therefore, during a loop we always have a fixed dataset (whose index is k^*) and a current root region, rr_{k^*}, on that dataset.

4.1.1 Root-Regions Bound

Given the complex query pattern, a set of root regions RR, one for each dataset, and a root region rr_{k^*} in RR taken as a reference, we are interested in computing an upper bound on the similarity score that can be obtained from a complex alignment whose root regions are those in RR with rr_{k^*} as a reference.

Definition 4.1 (Root-Regions Bound) *Given complex query pattern $P = \{p_j\}$, a collection of datasets $\{e_j\}$, a set of root regions $RR = \{rr_j\}$ with $j = 1, NP$ where $rr_j \in e_j$, and a root region rr_{k^*} in RR taken as a reference, the root-regions bound is computed as:*

$$rrbound(P, RR, k^*) = \frac{\sum_{j=1}^{NP} psim(p_{k^*}, p_j, rr_{k^*}, rr_j)}{NP} \tag{14}$$

The contribute to Equation 13 of k^* is bounded by $rrbound(P, RR, k^*)$, because any possible complex alignment MC whose root-regions are RR can only reduce the value of such complex pattern similarity.

$$\frac{\sum_{j=1}^{NP} psim(p_{k^*}, p_j, rr_{k^*}, rr_j)}{NP} \geq \frac{\sum_{j=1}^{NP} sim(M_j, p_j) \cdot psim(p_{k^*}, p_j, M_{k^*}, M_j)}{NP} \tag{15}$$

since $sim(M_j, p_j) \leq 1$.

It is therefore obvious that having rr_{k^*} as a reference this configuration of root regions RR can not lead to "good" results, whenever the bound is lower than θ.

4.1.2 Complex Alignment Bound

Given the complex query pattern, a set of root regions RR, one for each dataset, and a root region rr_{k^*} in RR taken as a reference, if $rrbound(P, RR, k^*) \geq \theta$, then we are forced to compute a complex alignment MC having RR as root regions. Now, we are interested in computing an upper bound on the similarity score that can be obtained from MC with rr_{k^*} as a reference.

Definition 4.2 (Complex Alignment Bound) *Given complex query pattern $P = \{p_j\}$, a collection of datasets $\{e_j\}$, a complex alignment MC whose root regions are $RR = \{rr_j\}$ with $j = 1, NP$, and a root region rr_{k^*} in RR taken as a reference, the complex alignment bound is computed as:*

$$cabound(P, MC, k^*) = rrbound(P, RR, k^*) \cdot \sum_{j=1}^{NP} \frac{\sum_{i=1}^{N_j} f_H\left(cost((m_{j,i}, q_{j,i})\right)}{N_j} \tag{16}$$

Again, the contribute to Equation 13 of k^* is bounded by $cabound(P, MC, k^*)$, because region similarities (whose values are not greater than 1) do not appear in the above definition. Indeed, by looking at Equation 10, we derive that

$$\sum_{j=1}^{NP} \frac{\sum_{i=1}^{N_j} f_H\left(cost((m_{j,i}, q_{j,i})\right)}{N_j} \geq \sum_{j=1}^{NP} \frac{\sum_{i=1}^{N_j} sim_{reg}(q_i, r_{j(i)}) \cdot f_H(cost(m_i, q_i))}{N_j} \tag{17}$$

since $sim_{reg}(q_i, r_{j(i)}) \leq 1$. The result then follows from the definition of root-regions bound.

We also note that the complex alignment bound is tighter than root-regions bound.

4.2 Diversity Threshold

The concept of diversity, introduced in Section 3.5, allows us to perform an additional check using the two bounds defined above. Indeed, given the complex query pattern and the current (reference) root region rr_{k^*}, we check whether a complex alignment MC, that is part of the results, has rr_{k^*} as a root region. If such alignment exists then the similarity scores of new complex alignments including rr_{k^*} must exceed its similarity score.

Definition 4.3 (Diversity Threshold) *Given complex query pattern $P = \{p_j\}$, a collection of datasets $\{e_j\}$, the reference root region rr_{k^*}, and a set of result complex alignments RES, the diversity threshold θ_d is defined as*

$$\theta_d(rr_{k^*}, RES, k^*) = \begin{cases} sim_C(MC, P) \geq \theta & \text{if } \exists MC \in RES \mid rr_{k^*} \text{ is a root region of } MC \\ \theta & \text{otherwise} \end{cases} \tag{18}$$

The value of $\theta_d(rr_{k^*}, RES, k^*)$ can now be used in place of θ when checking the bounds, $rrbound(P, RR, k^*)$ and $cabound(P, MC, k^*)$.

4.3 Algorithm Execution

Now we introduce how the bounds just presented are used by our algorithm to cut the search for "good" results. Before explaining the details of the algorithm, we need to give some introducing definitions.

Definition 4.4 (Root Regions Neighbors) *Given a root regions configuration RR and the index k^* of the fixed dataset, the root regions neighbors are all the other configurations that can be obtained from RR changing only a non-fixed root region rr_j ($j \neq k^*$) with the previous (if exists) or the following (if exists) region in the original input dataset e_j.*

Definition 4.5 (Neighbors) *Given a complex alignment MC and the index k^* of the fixed dataset, the neighbors are all the possible complex alignment that are obtained from MC changing one non-root region with the previous (if exists) or the following (if exists) region in the original input dataset. A neighbor is valid and taken into account only if it does not contain two or more occurrences of the same region.*

As we said before, we have a current fixed dataset and a current root region on it. Given this region, the first step of the inner loop is to compute the best configuration of root-regions on the other datasets with respect to the complex query pattern $P = \{p_j\}$. It is possible to do it through a binary search. Once obtained this best configuration of root regions, we first compute its *rrbound*. Then, we insert this configuration in a list, called *rrlist*, that is sorted by decreasing values of *rrbound*.

Now we start a secondary loop, called *rrloop*, whose termination condition is that the *rrlist* is empty. This loop works as follows:

- First, we extract the first configuration from the *rrlist* and we check if its *rrbound* is higher than the current threshold, whose value is the biggest between θ_d and the K-th result's score. If it is not we can discard this root region and we can proceed in our inner loop with a new root region. In fact, the structure of *rrlist* guarantees that the root regions configuration under exam is the one with the highest *rrbound*.

- Otherwise, we compute its root regions neighbors and, for each of them, we calculate its *rrbound* and we insert them in the list *rrlist*. Note that we use another structure, called *checked_rrlist* that keeps track of which root regions configurations have been already checked. We add a configuration to the *rrlist* only if it is not into the *checked_rrlist*.

- Second, we consider the extracted configuration and we compute, for each root region rr_j, the best alignment M_j with respect to the simple pattern p_j considering only the regions distances (region values are temporary ignored). It is possible to do it through j binary searches.

- Finally, we calculate, for this obtained complex alignment, its *cabound* and then we insert this alignment in a list, called *calist*, that is sorted by decreasing values of *cabound*.

Now we start another nested secondary loop, called *caloop*, whose termination condition is that the *calist* is empty. This loop works as follows:

- First, we extract the first complex alignment from the *calist* and we check if its *cabound* is higher than the current threshold, whose value is the biggest between θ_d and the K-th result's score. If it is not we can discard this configuration of root regions and we can proceed in our *rrloop* with a new configuration. In fact, the structure of *calist* guarantees that the complex alignment under exam is the one with the highest *cabound*.

- Otherwise, we compute its neighbors and, for each of them, we calculate its *cabound* and we insert them in the list *calist*. Note that we use another structure, called *checked_calist* that keeps track of which complex alignments have been already checked. We add a neighbor to the *calist* only if it is not into the *checked_calist*.

- Then, we compute the similarity for this complex alignment and we check if it is higher than the current threshold, whose value is the biggest between θ_d and the K-th result's score. If it is not, we proceed in our *caloop* with a new complex alignment. Otherwise, we add this alignment to the results.

- The complex alignment is added to the results only if it passes the the diversity check, i.e. if none of its regions are already in a result with a similarity score not lower than its. On the other hand, if its regions are already in results with a lower similarity score, these results are removed.

- If the complex alignment is inserted into the results and it is necessary, we remove the $(K + 1)$-th result.

Algorithm 1 : Greedy Algorithm with Bounds

Require: query pattern $P = \{p_j\}$ with $j = 1, NP$; datasets $E = \{e_j\}$ with $j = 1, NP$; a positive integer K; a real between 0 and 1 θ; horizontal match function parameters mp_H and sl_H; vertical match function parameters mp_V and sl_V; matrix $\mathcal{W}$ of the weights of the attributes where $w_{i,j}$ represents the j-th feature of the i-th dataset.

Ensure: K best complex alignments $RES = \{RES_h\}$ with $h = 1, K$

$RES \Leftarrow$ new sorted list $\{\}$
for dataset e **in** E **do**
  $k^* \Leftarrow$ index of e
  for region rr **in** e **do**
    $rrlist \Leftarrow$ new sorted list $\{\}$
    $checked\_rrlist \Leftarrow$ new list $\{\}$
    $\theta_d \Leftarrow$ COMPUTE_DIVTHRESHOLD(rr, RES, k^*, θ)
    $RR \Leftarrow$ GET_BEST_ROOT_REGIONS(P, E, rr, k^*)
    $rrbound \Leftarrow$ COMPUTE_RRBOUND(P, RR, k^*, mp_V, sl_V)
    add RR to $rrlist$ with value $rrbound$
    while $rrlist \neq \{\}$ **do**
      $calist \Leftarrow$ new sorted list $\{\}$
      $checked\_calist \Leftarrow$ new list $\{\}$
      $RR \Leftarrow$ extract head from $rrlist$ with value $rrbound$
      add RR to $checked\_rrlist$
      if $rrbound \leq \theta_d$ **or** $rrbound \leq$ similarity score of RES_K **then**
        go to 4
      end if
      $RR\_neighbors \Leftarrow$ GET_RR_NEIGHBORS(P, E, RR, k^*)
      for RRN **in** $RR\_neighbors$ **do**
        if $RRN \notin checked\_rrlist$ **then**
          $rrboundN \Leftarrow$ COMPUTE_RRBOUND$(P, RRN, k^*, mp_V, sl_V)$
          add RRN to $rrlist$ with value $rrboundN$
        end if
      end for
      $MC \Leftarrow$ new list $\{\}$
      for $j = 1$ **to** NP **do**
        $rr_j \Leftarrow RR[j]$
        $M_j \Leftarrow$ GET_BEST_REGIONS$(P[j], E[j], rr_j)$
        $MC[j] \Leftarrow M_j$
      end for
      $cabound \Leftarrow$ COMPUTE_CABOUND$(P, MC, k^*, mp_V, sl_V, mp_H, sl_H)$
      add MC to $calist$ with value $cabound$

34: **while** $rrlist \neq \{\}$ **do**
35: $MC \Leftarrow$ extract head from *calist* with value *cabound*
36: add MC to *checked_calist*
37: **if** $cabound \leq \theta_d$ **or** $rdbound \leq$ similarity score of RES_K **then**
38: **go to** 11
39: **end if**
40: $neighbors \Leftarrow$ GET_NEIGHBORS(P, E, MC)
41: **for** MCN **in** $neighbors$ **do**
42: **if** $MCN \notin$ *checked_calist* **then**
43: $caboundN \Leftarrow$ COMPUTE_CABOUND$(P, MCN, k^*, mp_V, sl_V, mp_H, sl_H)$
44: add MCN to *calist* with value *caboundN*
45: **end if**
46: **end for**
47: $simscore \Leftarrow$ COMPUTE_SIMILARITY$(MC, P, mp_V, sl_V, mp_H, sl_H, \mathcal{W})$
48: **if** $simscore > \theta_d$ **and** $simscore >$ similarity score of RES_K **and** DIVERSITY_CHECK(MC, RES) **then**
49: DIVERSITY_REMOVAL(RES)
50: add MC to RES with value *simscore*
51: $theta_d \Leftarrow$ COMPUTE_DIVBOUND(rr, RES, k^*, θ)
52: if necessary remove the $(K+1)$-th result
53: **end if**
54: **end while**
55: **end while**
56: **end for**
57: **end for**

---

5 Experimental Results

In this section we present first preliminary experimental results on the efficiency of the algorithm presented in section 4. In particular, we point out how the parameters and the dimension of the query pattern influence the execution time. The data used for this purposes is a real genomic dataset provided by Politecnico of Milan as a first benchmark for our tests.

Note that, the greedy approach has not been considered as a relevant competitor in our experiments since its execution time is not acceptable.

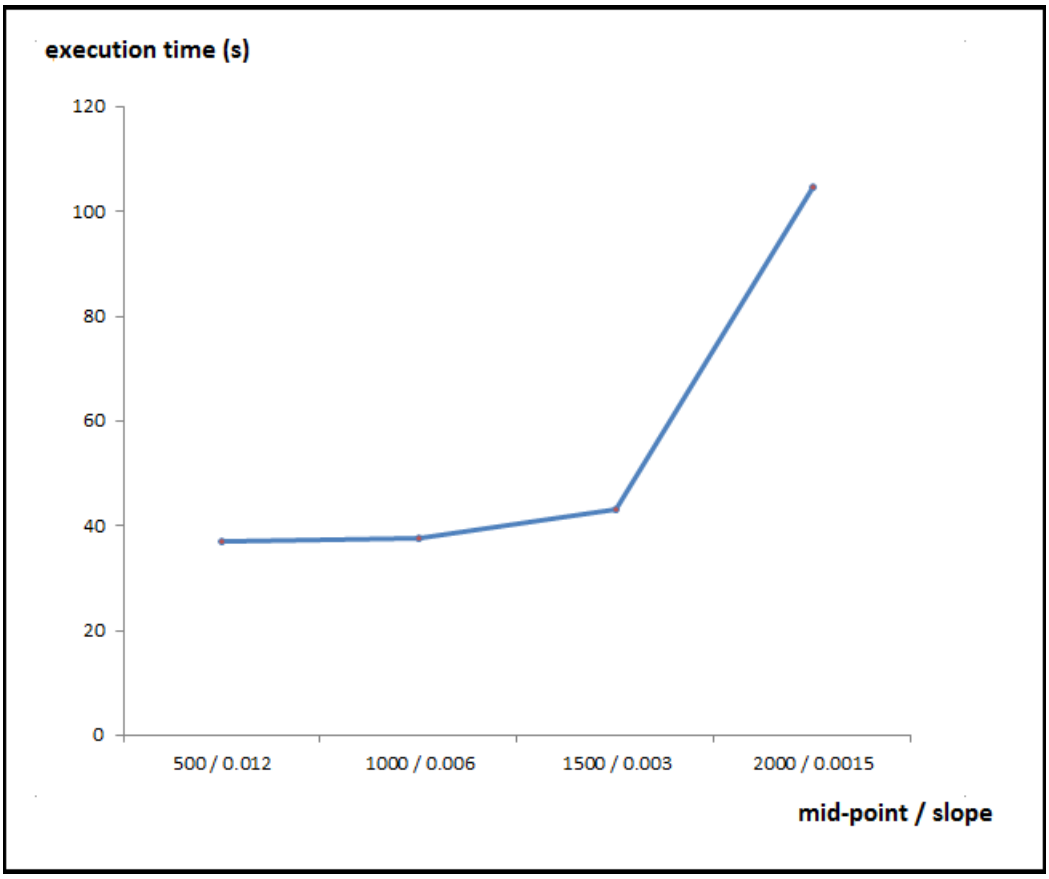

Figure 11: Relation between distance functions parameters and execution time

Figure 11 represents a graph that relates the parameters of distance match functions (mp_H, sl_H, mp_V and sl_V) to the execution time of our algorithm. To simplify the graph, we put $mp_H = mp_V$ and $sl_H = sl_V$. It is clear from the graph that increasing the parameters, the computation time increases as well. This is well explainable by the fact that the values of *pdbound* and *rdbound* are higher and so the cuts are less frequent.

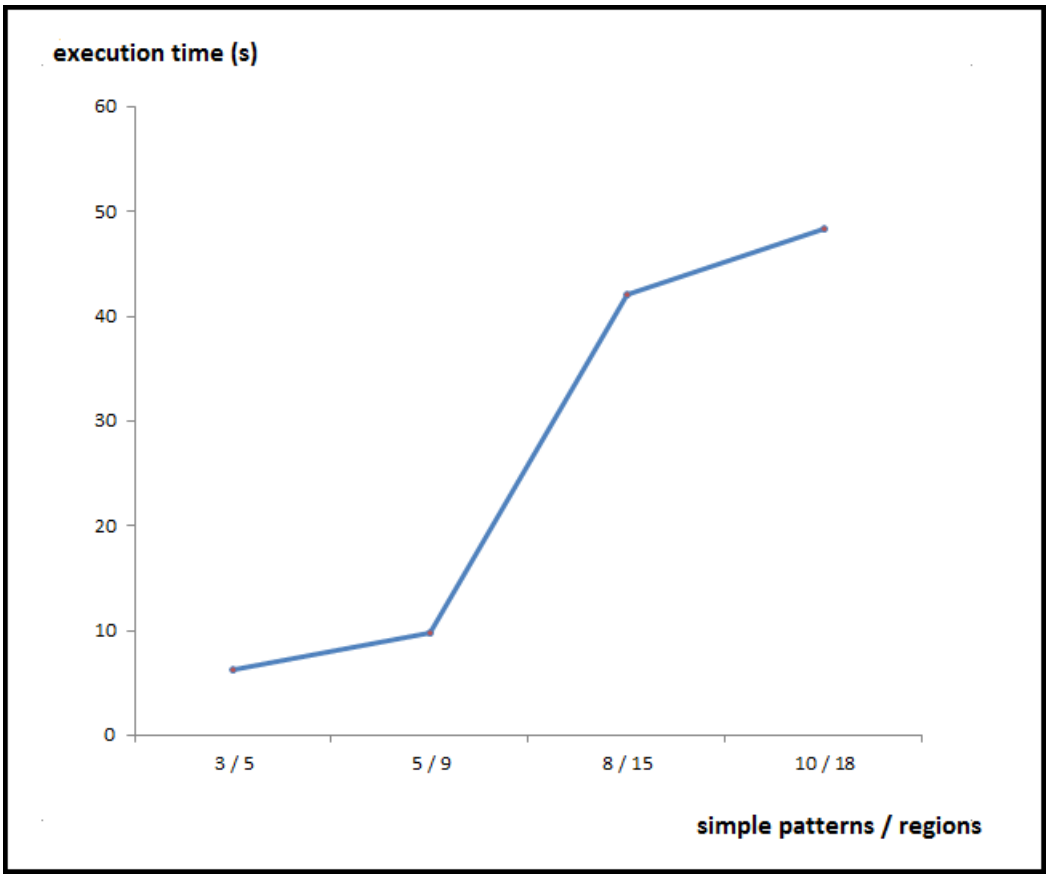

Figure 12: Relation between pattern dimension and execution time

Figure 12 represents a graph that relates the dimension of the complex pattern (e.g. number of simple patterns that form it and number of overall regions) to the execution time. A bigger pattern increases the complexity of the search and, as a consequence, it increases the execution time.

6 Conclusions and Future Work

We developed an operator to perform the search of similar patterns in a genomic sequence. In particular, we realized a top-K operator that returns the best K matches with the given query pattern. The efficiency of our algorithm is strictly connected to the choice of the involved parameters. The correct tuning of such parameters has not been exhaustively investigated and we plan to work in this direction with a strict collaboration with the biologists. Similarly, we intend to face effectiveness aspects by exploiting the biological ability to understand if a result is good or not.

Another aspect we are working on is the configuration and usage of an

Hadoop cluster within our DataLab, in order to distribute the query task and improve execution performances. For this purpose, we have already developed a distributed version of our algorithm and we have set a 8-machines Hadoop cluster. We will be able to provide first results on this aspect soon.

Finally, we are focusing on the problem of partial matches. The current version of our algorithm always search for matches composed by the same number of regions of the query pattern. However, this is too strict from a biological point of view. In fact, a region can be missing in one input dataset for some instrumental error and the results must take this into account.

References

[1] Stefano Ceri, Vahid Jalili, Abdulrahman Kaitoua, Marco Masseroli Fernando Paluzzi, Pietro Pinoli, Francesco Venco. Automatic Specifcation of Genomic Data Model (GDM) and Genometric Query Language (GMQL). Deliverable 12, GeneData2020 MIUR PRIN project, 2014.

[2] Stefano Ceri, Abdulrahman Kaitoua, Marco Masseroli, Matteo Matteucci, Pietro Pinoli, Francesco Venco. GenData 2020 Architecture. Deliverables 21 and 22, GeneData2020 MIUR PRIN project, 2014.