Pattern Similarity Search in Genomic Sequences

Piero Montanari, Ilaria Bartolini, *Member, IEEE,* Paolo Ciaccia, Marco Patella
Stefano Ceri, and Marco Masseroli

Abstract—Genomics, with the high amount of heterogeneous data that it is generating, is opening many interesting practical and theoretical computational problems; one of them is the search for a collections of genomic regions at given distances from each other, i.e., a pattern of genomic regions, along the whole genome. In this paper we present an optimized pattern-search algorithm able to find efficiently, within a large set of genomic data, genomic region sequences which are similar to a given pattern. We start with a base version of the problem, which is solved using dynamic programming enhanced with an efficient window-based technique; then, we extend the algorithm to more complex scenarios with practical applications in revealing interesting and unknown regions of the genome, thus, making it an important ingredient in supporting biological research. We apply our algorithm to enhancer detection, a relevant biological problem, showing that the method is both efficient and accurate.

Index Terms—Genomic computing, pattern-based query processing, dynamic programming.

✦

1 INTRODUCTION

A new technology for reading the DNA, called Next Generation Sequencing (NGS), is changing biological research, and will change medical practice, thanks to its low-cost provision of millions of whole genome sequences of a variety of species, and most important of humans. Huge repositories of genomic sequence information are being collected by large consortia of research laboratories by using NGS; among them, ENCODE [10], TCGA [27], the *1000 Genomes Project* [1] and the *100,000 Genomes Project* [2]. These sequences can be integrated with specific experimental data produced at the various research or clinical centers, opening new opportunities for biological discovery and for personalized medicine.

So far, the bioinformatics community has been challenged by NGS *primary analysis* (production of sequences in the form of short DNA segments, or "reads") and *secondary analysis* (alignment of reads to a reference genome and extraction of specific genomic features, such as variants/mutations and peaks of expression); but the most important emerging problem is *tertiary analysis*, concerned with multi-sample processing, annotation and filtering of variants, and genome browser-driven exploratory analysis [23]. While secondary analysis targets *raw data* in output from NGS processors by using specialized methods, tertiary analysis targets *processed data* in output from secondary analysis and is responsible of *sense making*, e.g., discovering how multiple genomic regions, representing heterogeneous genomic features, interact with each other.

The GenData 2020 research project[1] addresses this challenge, by enabling queries and analysis of processed genomic data. The project's main results so far are a *Genomic Data Model* (GDM), which encodes processed genomic data in terms of their genomic regions and metadata, and a *Geno-Metric Query Language* (GMQL) to extract genomic regions of interest from NGS experiments and compute their properties, with high-level operations for manipulating regions and measuring their distances [20].

1.1 The Problem and our Contributions

Several tertiary analysis problems consist of searching for *patterns of regions*, i.e., co-occurrences of certain region configurations along the genome; such regions are usually heterogeneous genomic traits with particular features. The main contribution of this paper is the design and implementation of an optimized, efficient *pattern-search algorithm* which provides biologists with the ability, once they identify an interesting genomic region *pattern*, to look for *similar* occurrences of such pattern in the whole genome. Towards this end, we also developed a stand-alone desktop application,[2] described in [21], that enables biologists to define patterns of interest using the Integrated Genome Browser [22], a visualization tool commonly used to observe genomic datasets. The software application also allows executing the created pattern-search algorithm and visualizing the matching regions found along the genome for their evaluation.

The key ingredient of a pattern, borrowed from GDM, is the notion of genomic (DNA) *region*, defined as a quadruple $\langle chr, left, right, strand \rangle$, in which chr represents the chromosome[3] where the region is located, and $left$ and $right$ are respectively the left and right ends of the region along

---

1. http://www.bioinformatics.deib.polimi.it/gendata/

2. The stand-alone desktop application is available for download at http://www-db.disi.unibo.it/research/GenData/.

3. The easiest way of considering the DNA is as a string of billions of nucleotides (DNA base molecular components represented by the letters A,C,G,T) enclosed within chromosomes (23 in humans), which are disconnected intervals of the string.

the DNA coordinates; the $strand$[4] is encoded as either $+$ or $-$, and can be missing. Each region includes all the DNA nucleotides whose position is between $left$ and $right$.[5] Moreover, typically a region is associated by secondary analysis with a *feature vector*, where each feature has an *attribute name* and a *type*, and is extracted by suitable processing.

All the genomic regions produced by a specific NGS experiment constitute an *experiment sample*, and can be represented as a *track* on a genome browser. Given that regions can be ordered according to their genomic coordinates, each track can be considered as an *ordered* sequence of regions. Informally, a *pattern* is a collection of genomic regions that are present on one or more tracks. The selection of the tracks and of the regions forming a pattern depends on the specific biological problem. Figure 1 shows an example of multi-track pattern over 3 tracks, constituted by 6 regions (2 on track 1, 3 on track 2 and 1 on track 3) intertwined within other regions on the same tracks.

Fig. 1. An example of multi-track pattern (thick regions).

Each result of our pattern-search algorithm is a collection of regions with properties similar to the query pattern. In particular, we look for results in which the regions tend to have (approximately) the same spatial configuration of the query pattern (*structural similarity*) and similar values for the region attributes (*region similarity*). As Figure 2 suggests, structural similarity ignores absolute coordinate values of the pattern regions, focusing on *inter-region distances*. In addition to *regular (positive) tracks*, as discussed so far, we also consider *partial tracks*, i.e., tracks whose regions may be missing (but when they are present they strengthen the pattern's similarity), and *negative tracks*, i.e., tracks whose regions must be missing. The search for patterns within datasets is a typical problem in data sciences, but, as far as we know, the type of pattern-matching problem we consider here has not been faced before, as we discuss in the related work Section 6.
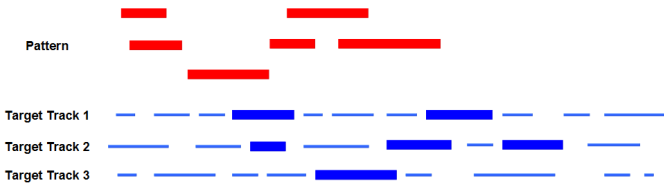

Fig. 2. Example of search result on target tracks with high structural similarity to the query pattern

4. The DNA is made of two strands, rolled-up together in anti-parallel directions, where genomic regions are located.
5. According to the University of California at Santa Cruz (UCSC) notation, we use 1-based inter-base coordinates, i.e., the considered genomic sequence is $[left, right)$.

Several relevant biological problems can be formulated as pattern searches, among them:[6]

- The *search for enhancers*. These are particular regions of the non-coding part of the genome (i.e., falling outside of the genes) which, for specific cellular tissues and biological conditions, play the role of enhancing or repressing gene expression, i.e., the ability of a gene to produce proteins. Enhancers may be found by comprehensively looking at multiple molecular signals, where the signals must be either present or absent, to qualify a DNA region as an enhancer.
- The *search for specific topological domains (TADs)*. These are revealed by loops of the genome, where loop ends are distant in the 1D space of the aligned genome, but are close in the 3D space due to genome foldings. Each TAD is a region of the genome where most enhancer-gene contacts occur. TADS are stable across different cell types and highly conserved across species, indicating that they are an inherent property of mammalian genomes.

The rest of the paper is organized as follows. Section 2 introduces the basic concepts, provides a formal description of a simplified (base) version of the problem, and describes a dynamic programming algorithm to solve it. Section 3 incrementally introduces aspects that are missing in the base version of the problem (interval regions, multiple, partial, and negative tracks, region attribute matching, Top-K distinct matchings) and describes how the algorithm has to be suitably modified so as to address them. Section 4 shows two biological applications of pattern matching, and Section 5 describes the algorithm's performance over synthetic data. Finally, Section 6 presents related work in genomic data management and pattern search, and Section 7 draws the conclusions. The appendices contain: A description of the original problem definition, a score-based cost model for normalizing attribute matching, and an illustration of the results obtained by way of our desktop application.

2 THE BASE PROBLEM

In this section we formally define the *base version* of the problem we are dealing with, in which the pattern to be searched is single-track, regions are reduced to points, and region attributes are not present. We start by providing some basic definitions.

Definition 2.1 (Track). *A track T is a strictly increasing sequence of N elements, $T = \langle t_1, \ldots, t_N \rangle$, where each $t_i \in \mathbb{N}^+$ and $t_i < t_{i+1}$, $i = 1, \ldots, N-1$.*

The restriction on track elements to be positive natural numbers is only for consistency with the value of DNA coordinates and can be removed without any consequence, thus also considering real-valued elements.

Definition 2.2 (Matching). *Given a "query" track $Q = \langle q_1, \ldots, q_M \rangle$ and a "target" track $T = \langle t_1, \ldots, t_N \rangle$, with $N \geq M$, a matching of Q in T is a strictly increasing function*

6. We have been introduced to these problems, in the context of GenData 2020, by biologists at the European Institute for Oncology (IEO), a center of excellence in oncology research, see [7].

$f : [1, M] \rightarrow [1, N]$ that assigns to each element q_i of Q an element $t_{f(i)}$ of T. We refer to $(q_i, t_{f(i)})$ as a matched pair *of elements, and to* $((i, f(i)))$ *as a* matched index pair.[7]

Notice that, although the above definition only requires $N \geq M$, it has to be understood that in practice it will be $N \gg M$. For instance, a typical query track has $M \leq 10$, whereas N can be in the order of $10^3 \div 10^6$.

In order to search for patterns in the target track that are similar to the query track, we take a cost-based approach, which is typical for similarity search problems [29], where lower cost implies high similarity and vice versa.

Definition 2.3 (τ-Matching Cost). *Given tracks Q and T and a real value τ, the τ-matching cost of a matching f, $C_f(Q, T; \tau)$, is defined as:*

$$C_f(Q, T; \tau) = \sum_{i=1}^{M} \left(t_{f(i)} - q_i - \tau \right)^2 \qquad (1)$$

Let $\delta_{i,j} = t_j - q_i$ be the *absolute* offset of elements t_j and q_i. The τ parameter in Equation 1 allows us to consider *shifted* ($\delta_{i,j} - \tau$) rather than absolute ($\delta_{i,j}$) offsets, and is used to translate Q so as to better match elements of T.

Since, for any given matching f, each value of τ will yield a different cost, it would be quite natural to consider that the cost of matching f is the minimum τ-matching cost over *all* possible τ values. However, as detailed in Appendix A, this approach would make more complex the search for a minimum-cost matching, since it would amount to solve a *variance-minimization problem*, which is a specific quadratic assignment problem [5].

For the above reason, for any given matching f, we restrict ourselves to consider a specific τ value, a major advantage being that this choice leads to a tractable (i.e., polynomial) matching problem.

Definition 2.4 (Root-element Matching Cost). *Given tracks $Q = \langle q_1, \ldots, q_M \rangle$ and $T = \langle t_1, \ldots, t_N \rangle$, $N \geq M$, and a matching f of Q in T, let $\tau^r = t_{f(1)} - q_1$. The* root-element matching cost $C_f^r(Q, T)$ *is defined as:*

$$C_f^r(Q, T) = \sum_{i=1}^{M} \left(\delta_{i,f(i)} - \tau^r \right)^2 = \sum_{i=1}^{M} \left(\delta_{i,f(i)} - \delta_{1,f(1)} \right)^2 \qquad (2)$$

Since τ^r yields a zero-cost for the first matched pair of elements of Q and T, these can be well termed as *root (reference) elements* of their corresponding track. The base problem we are going to solve is consequently defined as:

Definition 2.5 (Root-element Best-Matching Problem (R-BMP)). *Given tracks Q and T, the root-element best-matching problem is to determine the matching f^* with minimum root-element matching cost, i.e., $C_{f^*}^r(Q, T) \leq C_f^r(Q, T) \ \forall f$.*

Notice that, since $\delta_{i,f(i)} - \delta_{1,f(1)} = (t_{f(i)} - q_i) - (t_{f(1)} - q_1)$, it can be asserted that R-BMP focuses on minimizing the squared differences of offsets w.r.t. root-elements offsets.

---

7. The use of double parentheses is to avoid any confusion among matched elements and the corresponding matched positions in the two sequences.

Example 1. *Let $Q = \langle 8, 20, 22, 36 \rangle$ and $T = \langle 10, 15, 17, 27, 35, 39, 45, 50, 62, 70 \rangle$. One of the possible matchings of Q in T is $f = \{((1, 2)), ((2, 3)), ((3, 4)), ((4, 7))\}$ that assigns elements $(15, 17, 27, 45)$ to corresponding elements of Q.*
Since it is $\tau^r = 15 - 8 = 7$, the root-element matching cost of f is 108:

$$C_f^r(Q, T) = (15 - 8 - 7)^2 + (17 - 20 - 7)^2 + (27 - 22 - 7)^2$$
$$+ (45 - 36 - 7)^2 = 108.$$

2.1 Basic Dynamic Programming Algorithm

From a computational point of view, R-BMP is amenable to an efficient resolution. Indeed, the contribution of a matched pair $(q_i, t_{f(i)})$ to the overall cost is decoupled from that of the other pairs, since τ^r only depends on the root-elements and not on the whole matched elements. This observation is the key to develop a dynamic programming algorithm, called DP-RBMP, that, as it will be clarified in the following, can also be gracefully extended to cover all the aspects of the pattern-search problem.

Dynamic programming is a general solution technique that can be applied when the optimal solution to the problem at hand can be efficiently obtained from solutions to simpler subproblems, a typical example being that of finding a shortest path in a graph. In our context, we can exploit the following fact:

Lemma 2.6. *A matching f of Q in T is optimal, i.e., has minimum root-element matching cost, if and only if each of the (partial) matchings $(f(1), f(2), \ldots, f(\ell))$, $\ell = 1, \ldots, M$, has minimum cost among all the (partial) matchings $f' = (f'(1), f'(2), \ldots, f'(\ell))$ such that $f'(1) = f(1)$ and $f'(\ell) = f(\ell)$, i.e.:*

$$\sum_{i=1}^{\ell} (\delta_{i,f(i)} - \delta_{1,f(1)})^2 \leq \sum_{i=1}^{\ell} (\delta_{i,f'(i)} - \delta_{1,f'(1)})^2$$

Proof. The intuition about the proof is that, for given "start" ($f(1)$) and "end" ($f(\ell)$) positions in T, any partial matching f' which also matches q_1 to $t_{f(1)}$ ($f'(1) = f(1)$) and q_ℓ to $t_{f(\ell)}$ ($f'(\ell) = f(\ell)$), yet has a partial cost higher than that of f ($\sum_{i=1}^{\ell} (\delta_{i,f(i)} - \delta_{1,f(1)})^2 < \sum_{i=1}^{\ell} (\delta_{i,f'(i)} - \delta_{1,f'(1)})^2$) cannot be completed to yield a matching with minimum cost.

Only if. Assume that for some value of ℓ ($\ell < M$) there exists a partial matching f' such that $f'(1) = f(1)$, $f'(\ell) = f(\ell)$, and $\sum_{i=1}^{\ell} (\delta_{i,f(i)} - \delta_{1,f(1)})^2 > \sum_{i=1}^{\ell} (\delta_{i,f'(i)} - \delta_{1,f'(1)})^2$. Since $f = (f(1), f(2), \ldots, f(\ell), f(\ell + 1), \ldots, f(M))$ is assumed to be a matching, so it is $f' = (f'(1), f'(2), \ldots, f'(\ell), f(\ell + 1), \ldots, f(M))$. This follows from $f'(\ell) = f(\ell)$, which guarantees that the sequence of matched positions in T is strictly increasing in f' as well. From the assumption $f'(1) = f(1)$, we have that, for all $i > \ell$, $(\delta_{i,f(i)} - \delta_{1,f(1)})^2 = (\delta_{i,f(i)} - \delta_{1,f'(1)})^2$. It follows that $C_{f'}^r(Q, T) < C_f^r(Q, T)$, i.e., f is not optimal. When $\ell = M$, the result follows from the very definition of optimal matching.

If. This follows from the definition of optimal matching. $\square$

Based on the above lemma, the DP-RBMP algorithm starts by partitioning the problem into $(N - M + 1)$ subproblems, one for each possible value of $f(1)$, thus of τ^r.[8]

---

8. It has to be $f(1) \leq N - M + 1$ in order to respect the constraint $f(i) < f(i + 1), i = 1, \ldots, M - 1$.

Given $f(1)$, we apply the dynamic programming technique by constructing an $M \times N$ matrix $Z_{f(1)}$. The value of cell (i,j) of this matrix, also called the *cell cost* and denoted $Z_{f(1)}(i,j)$, is computed so as to be the minimum cost that is possible to obtain by matching the first $i-1$ elements of Q in the first $j-1$ elements of T and q_i with t_j. For each cell (i,j) we also maintain a list, $ML_{f(1)}(i,j)$, of the indices of the matched elements of T yielding the cell cost $Z_{f(1)}(i,j)$.

Because of the constraint $f(i) < f(i+1)$, $i = 1, \ldots, M-1$, several cells of matrix $Z_{f(1)}$ will remain *unfilled*, and their cell cost is conventionally assumed to be equal to ∞. For the other cells computation of values is performed as follows:

1) For $i = 1$, the only cell to be filled is $(1, f(1))$, i.e., the cell corresponding to the root-element $t_{f(1)}$. For such cell it is, by definition, $Z_{f(1)}(1, f(1)) = 0$ and $ML_{f(1)}(1, f(1)) = (f(1))$.
2) For $i = 2, \ldots, M$, the cells (i,j) to be filled are only those for which it is $(f(1) + i - 1) \leq j \leq (N - M + i)$, since the others violate the constraint $f(i-1) < f(i)$. In order to ensure that $Z_{f(1)}(i,j)$ equals the minimum cost of matching the first $i-1$ elements of Q in the first $j-1$ elements of T and q_i with t_j, the following recurrence is used:

$$Z_{f(1)}(i,j) = \min_{h:h<j} \left\{ Z_{f(1)}(i-1,h) \right\} + (\delta_{i,j} - \tau^r)^2 \quad (3)$$

Letting h' denote the value of index h for which the minimum is attained in the above equation, the matched index list for cell (i,j) is consequently computed as $ML_{f(1)}(i,j) = ML_{f(1)}(i-1,h') + (j)$, where '+' denotes list append.

Theorem 2.7. *When cells of each matrix $Z_{f(1)}$, $1 \leq f(1) \leq (N - M + 1)$, are filled according to Equation 3, it is:*

$$C_{f*}^r(Q,T) = \min_{f(1)} \min_j \left\{ Z_{f(1)}(M,j) \right\} \quad (4)$$

Proof. The cell cost $Z_{f(1)}(M,j)$ is the minimum matching cost obtainable for a given $f(1)$ and assuming that q_M is matched with t_j. This easily follows from Equation 3 by inductive arguments that resemble those in the proof of Lemma 2.6. Thus, taking the minimum over j yields the minimum cost for a given $f(1)$, after that the result immediately follows. $\square$

A naïve implementation of the DP-RBMP algorithm would require $\mathcal{O}(MN^3)$ time in the worst case, since the number of matrices is $N - M + 1$, each with $\mathcal{O}(MN)$ cells to fill, and computing each cell cost requires $\mathcal{O}(N)$ time according to Equation 3. Reminding that h' is the value of index h which yields the minimum in Equation 3, the complexity can be lowered to $\mathcal{O}(MN^2)$ by exploiting the observation that h' can indeed be easily determined in $\mathcal{O}(1)$ time. For this it is sufficient to keep track of the minimum cell cost when filling the cells in a row.[9] In particular, the index h' for a cell (i,j), now denoted $h'(i,j)$ for clarity, is obtained by comparing the minimum cell cost for $i-1$ and $h < j-1$, thus the one in column $h'(i,j-1)$, and the cell cost of cell $(i-1, j-1)$:

9. Notice that here we are just exploiting the well-known identity $\min\{a_1, \ldots, a_{n-1}, a_n\} = \min\{\min\{a_1, \ldots, a_{n-1}\}, a_n\}$.

$$h'(i,j) = \begin{cases} h'(i,j-1) & \text{if } Z_{f(1)}(i-1, h'(i,j-1)) < Z_{f(1)}(i-1, j-1) \\ j-1 & \text{otherwise} \end{cases}$$

Algorithm 1 The DP-RBMP algorithm

1: **for** $f(1) \leftarrow 1, N - M + 1$ **do**
2:    **for** $i \leftarrow 2, M$ **do**
3:      $h'(i, f(1) + i - 2) \leftarrow f(1) + i - 2$
4:      **for** $j \leftarrow (f(1) + i - 1), (N - M + i)$ **do**
5:        **if** $Z_{f(1)}(i-1, h'(i,j-1)) < Z_{f(1)}(i-1, j-1)$
6:        **then** $h'(i,j) \leftarrow h'(i,j-1)$
7:        **else** $h'(i,j) \leftarrow j-1$
8:        $Z_{f(1)}(i,j) \leftarrow Z_{f(1)}(i-1, h'(i,j)) + (\delta_{i,j} - \tau^r)^2$
9:        $ML_{f(1)}(i,j) \leftarrow ML_{f(1)}(i-1, h'(i,j)) + (j)$
10:    $C_f^r(Q,T) \leftarrow \min_j \{ Z_{f(1)}(M,j) \}$
11: $C_{f*}^r(Q,T) \leftarrow \min_{f(1)} \{ C_f^r(Q,T) \}$

The actual behavior of the DP-RBMP algorithm (as well as that of its variants and extensions we introduce in the following) also depends on the possibility of *early abandoning* the analysis of a matrix. In fact, if at a certain stage of the execution a matching f with cost $C_f^r(Q,T)$ has been found, a matrix Z_j, with $j \neq f(1)$, needs to be analyzed as long as it can lead to a matching that costs less than $C_f^r(Q,T)$. Thus, one can stop filling the cells of Z_j as soon as the minimum cell cost in a row i is $\geq C_f^r(Q,T)$. For the sake of brevity, this optimization is not written down in Algorithm 1.

Example 2. *Let $Q = \langle 8, 20, 22, 36 \rangle$ and $T = \langle 10, 15, 17, 27, 35, 39, 45, 50, 62, 70 \rangle$ as in Example 1. Let us consider Z_2 (shown in Figure 3), one of the $(N - M + 1) = 7$ matrices generated by DP-RBMP, for which it is $\tau^r = 15 - 8 = 7$. To see how cells are filled consider, for example, cell $(3,7)$. In this case Equation 3 yields:*

$$Z_2(3,7) = \min_{h:h<7} \{ Z_2(2,h) \} + (\delta_{3,7} - \tau^r)^2$$
$$= 0 + ((45 - 22) - 7)^2 = 256$$

Since the minimum cell cost in the 2nd row, when $j < 7$, is found in cell $(2,4)$, it is $h' = 4$, thus $ML_2(3,7) = ML_2(2,4) + (7) = (2,4,7)$.

The best matching f for Z_2 is obtained in cell $(4,7)$, that is, $f = (((1,2)), ((2,4)), ((3,5)), ((4,7)))$ that assigns $(15, 27, 35, 45)$ to $(8, 20, 22, 36)$. The cost of the matching f is $C_f^r(Q,T) = 40$. If one had already found, in a previously generated matrix, a matching with cost, say, 32, matrix Z_2 would have been filled only up to row 3, since in that row the minimum cell cost is 36, thus higher than 32.

2.2 Windowed DP Algorithm

The complexity of the DP-RBMP algorithm can be substantially improved when $M \ll N$ by exploiting the observation that T and Q are strictly increasing sequences, which makes it possible to obtain the best match for each element of Q in T through a binary search. The resulting algorithm is denoted WDP-RBMP, where the 'W' stands for "windowed".

Given the target track root-element $t_{f(1)}$ that characterizes the matrix $Z_{f(1)}$, for each row $i = 1, \ldots, M$, let $cm(i)$ be the index of the "closest match" of q_i in T:

Z_2	1(10)	2(15)	3(17)	4(27)	5(35)	6(39)	7(45)	8(50)	9(62)	10(70)
1 (8)		0 (2)								
2 (20)			100 (2,3)	0 (2,4)	64 (2,5)	144 (2,6)	324 (2,7)	529 (2,8)		
3 (22)				104 (2,3,4)	36 (2,4,5)	100 (2,4,6)	256 (2,4,7)	441 (2,4,8)	1296 (2,4,9)	
4 (36)					168 (2,3,4,5)	52 (2,4,5,6)	40 (2,4,5,7)	85 (2,4,5,8)	520 (2,4,5,9)	877 (2,4,5,10)

Fig. 3. DP-RBMP algorithm: the matrix Z_2 of Example 2. In each cell we show the cell cost and the list of the indices of matched elements of T.

Definition 2.8 (Closest Match). *Given tracks Q and T, and a value of $\tau^r = t_{f(1)} - q_1$, let $cost(i,j) = (\delta_{i,j} - \tau^r)^2$ denote the cost of matching q_i and t_j. The closest match of q_i in T is the element of T with index $cm(i)$ for which $cost(i,j)$ is minimized, i.e., $cost(i, cm(i)) < cost(i,j), \forall j \neq cm(i)$.[10] By definition, it is $cm(1) = f(1)$. Since both T and Q are strictly increasing sequences, it is $cm(i) \leq cm(i+1), i = 1, \ldots, M-1$.*

It is evident that, in absence of "conflicts" among the closest matches, i.e., when $cm(i) < cm(i+1), i = 1, \ldots, M-1$, the matching $f_{cm} = (cm(1), \ldots, cm(M))$ based on closest matches would be the best one for the given root element $t_{f(1)}$. However, this breaks down in case of conflicts ($cm(i) = cm(i+1)$), since f_{cm} would *not* be a matching anymore.

The intuition behind the WDP-RBMP algorithm is that the optimal matching has to be "close" to f_{cm}, thus for each row i of $Z_{f(1)}$ only a *window* of cells of limited size around $cm(i)$ has to be considered. The window has to be large enough in order to allow resolution of conflicts and, at the same time, to guarantee that the optimal matching can still be found. This is made precise by the following definition.

Definition 2.9 (DP-window). *Given the $Z_{f(1)}$ matrix, the DP-window of $Z_{f(1)}$ is defined as $W = \{W_i : i = 1, \ldots, M\}$, where $W_i = \{(i,j) : j \in [L_i, H_i]\}$. The extremes L_i and H_i of the i-th row W_i are iteratively computed as follows:*

$$
\begin{aligned}
L_1 = &\ H_1 = cm(1) = f(1) \\
L_M = &\ \max\{cm(M), f(1) + M - 1\} \\
L_i = &\ \max\{\min\{cm(i), L_{i+1} - 1\}, f(1) + i - 1\} \\
&\qquad\qquad\qquad\qquad (i = 2, \ldots, M-1) \\
H_i = &\ \max\{cm(i), H_{i-1} + 1\} \qquad (i = 2, \ldots, M)
\end{aligned}
$$

Consider how the H_i's are defined. For H_2 we either consider the closest match of q_2, if this is higher than $cm(1) = f(1)$, or minimally extend the window to the right, $H_2 = H_1 + 1 = f(1) + 1$. This gives us the possibility to solve conflicts, such as $f(1) = f(2)$, which are not admitted in a matching. The same is done for all others H_i's. For computing the lower extremes L_i's, first observe that the matching constraint $f(i) < f(i+1)$ implies $f(i) \geq f(1) + i - 1$, for $i = 2, \ldots, M$; otherwise, we proceed in a similar way as for H_i's, yet going backwards and starting from L_M. The following observation is now obvious:

Observation 2.10. *For $i = 1, \ldots, M-1$, it is $L_i < L_{i+1}$ and $H_i < H_{i+1}$. If any row of W is restricted, this property is lost.*

Figure 4 (a) shows a possible DP-window for the case $M = 7$, while Figure 4 (b) shows the worst-case scenario in

---

10. We omit here the very particular case when the closest match is tied among two adjacent elements in T, which would unnecessarily lengthen the description.

---

which all the $M - 1$ closest matches other than $f(1)$ are in conflict.
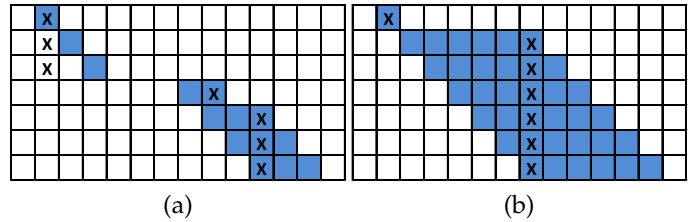

Fig. 4. Examples of DP-windows: closest matches are denoted with a **X**. (a) A possible DP-window for the case $M = 7$. (b) The worst-case scenario, when all the $M-1$ closest matches $cm(i), i > 1$ are in conflict.

Example 3. *Let Q and T, as in Example 2. The DP-window of matrix Z_2, for which it is $\tau^r = 15 - 8 = 7$, is shown in Figure 5. It is $cm(2) = 4$, since the closest match for $q_2 + \tau^r = 20 + 7$ is $t_4 = 27$. The same (conflicting) value is obtained for $cm(3)$, whereas $q_4 + \tau^r = 36 + 7 = 43$ yields $cm(4) = 7$, since $t_7 = 45$ is the closest element in T. Based on Definition 2.9 it is $W_2 = [3, 4]$, $W_3 = [4, 5]$, and $W_4 = [7, 7]$. So, besides cell $(1, 2)$, WDP-RBMP only needs to fill 5 cells in the Z_2 matrix, as compared to the 18 cells filled by DP-RBMP (compare Figure 5 to Figure 3).*

Having illustrated how the DP-window is defined, we now turn to prove its correctness (the DP-window is "large enough") and optimality (the DP-window has smallest size). The following observation, which follows from T and Q being both strictly increasing sequences, formalizes the intuition that moving away from the closest match leads to a cost increase:

Observation 2.11. *For $i = 2, \ldots, M$ the following both hold:*

$$
\begin{aligned}
cost(i, j-1) > cost(i, j) &\qquad j \leq cm(i) \\
cost(i, j+1) > cost(i, j) &\qquad j \geq cm(i)
\end{aligned}
$$

Theorem 2.12. *Given the $Z_{f(1)}$ matrix, the DP-window W in Definition 2.9 has the minimum size that guarantees to find the best possible matching f^*.*

Proof. Let us consider a matching f_{out} that is not completely contained in the DP-window W. Clearly, for each $i > 1$, only one of the following cases can occur:

1) $f_{out}(i) < L_i$;
2) $f_{out}(i) \in [L_i, H_i]$;
3) $f_{out}(i) > H_i$.

where for at least one i case 1 or 3 occurs (otherwise f_{out} would be completely contained in W).

Let us now consider a matching f_{in} contained in W and defined as follows:

Z_2	**1**(10)	**2**(15)	**3**(17)	**4**(27)	**5**(35)	**6**(39)	**7**(45)	**8**(50)	**9**(62)	**10**(70)
1 (8)		0 (2)								
2 (20)			100 (2, 3)	**0** (**2**, **4**)						
3 (22)				**104** (**2**, **3**, **4**)	36 (2, 4, 5)					
4 (36)							**40** (**2**, **4**, **5**, **7**)			

Fig. 5. WDP-RBMP algorithm: the matrix Z_2 of Example 2. Closest matches, $cm(i)$, are in boldface.

1) $f_{in}(i) = L_i$ when $f_{out}(i) < L_i$;
2) $f_{in}(i) = f_{out}(i)$ when $f_{out}(i) \in [L_i, H_i]$;
3) $f_{in}(i) = H_i$ when $f_{out}(i) > H_i$.

From Observation 2.10 we know that f_{in} is indeed a matching (no conflicts arise). When all the closest matches belong to the window (see e.g., Figure 4 (b)) then $C^r_{f_{out}}(Q,T) > C^r_{f_{in}}(Q,T)$ follows from Observation 2.11, since the costs of cases 1 and 3 of f_{out} are higher than those of f_{in}. Notice that, when $cm(i) < L_i$ (see e.g., Figure 4 (a)), case 1 ($f_{out}(i) < L_i$) cannot occur, since f_{out} is assumed to be a valid matching. Thus, if case 3 occurs Observation 2.10 still applies. Since these arguments can be applied to any matching f_{out} that is not completely contained in W, it follows that the size of W is sufficient for determining the best possible matching f^*.

In the above construction, Observation 2.10 is critical for guaranteeing the absence of conflicts in f_{in}. From the same observation it therefore follows that the size of W is the minimal one that ensures to find f^*.   □

The complexity of the WDP-RBMP algorithm (for which only differences w.r.t. DP-RBMP are summarized in Algorithm 2) is $\mathcal{O}(MN(\log N + M))$, since for each of the $(N - M + 1)$ matrices we have to execute $M - 1$ binary searches and then fill, in the worst case, $M - 1$ cells on each row of the DP-window (the cost of filling each cell is still $\mathcal{O}(1)$ as in the DP-RBMP algorithm). Notice that, when $M = o(N)$, the complexity drops down to $\mathcal{O}(N \log N)$, which makes it possible to apply WDP-RBMP also to (very) large problem instances.

Algorithm 2 The WDP-RBMP algorithm

1: **for** $f(1) \leftarrow 1, N - M + 1$ **do**
2:      Compute $cm(i), i = 2, \ldots, M$ as in Definition 2.8
3:      Compute $L_i, H_i, i = 1, \ldots, M$ as in Definition 2.9
     Execute Algorithm 1 replacing line 4. with
4:      **for** $j \leftarrow L_i, H_i$ **do**

---

3   EXTENDING THE BASE MODEL

In this section we introduce all the aspects that extend the base model: first, the genomic regions are intervals; second, the pattern can be defined on multiple tracks; third, partial and negative matchings can occur; finally, regions have attributes. All these extensions are required by genomics, and used in our biological applications (see Section 5).

3.1   Interval Regions

So far, we considered the elements t_i and q_j as points, but in genomic applications they are intervals; of course, an interval can be reduced to its *centroid* and therefore be

seen as a point. More precisely, if every genomic region is an interval $[left, right)$ on the genome, its centroid is $t_i = (r_i.right - 1 + r_i.left)/2$.

It is also possible to examine *asymmetric* approaches, where we consider regions as dimensional, but still we compute suitable t_i and q_j. Among the various solutions, we consider the one in which elements t_i are regions' *left* components, but the definition of τ^r considers root-elements' *right* components, i.e., $\tau^r = r_{f(1)}.right - qr_1.right$, where qr_i is the i-th region of the query pattern. The rationale behind this choice is that it allows us to align root-regions according to their *right* components and to base structural similarity only on the empty space (distance) between the root-regions and the other regions. Notice that, for $i > 1$, it is (refer also to Figure 6):

$$
\begin{aligned}
cost(i, f(i)) &= (t_{f(i)} - q_i - \tau^r)^2 \\
&= (r_{f(i)}.left - qr_i.left - (r_{f(1)}.right - qr_1.right))^2 \\
&= (r_{f(i)}.left - r_{f(1)}.right - (qr_i.left - qr_1.right))^2 \\
&= (b - a)^2
\end{aligned}
$$

whereas the matched root elements $(q_1, t_{f(1)})$ are still assumed to yield no additional cost.
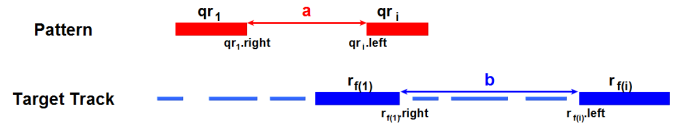

Fig. 6. The asymmetric approach for interval regions.

Both the centroid-based and the asymmetric approaches do not require any change to WDP-RBMP algorithm, as they ignore the regions' lengths in the computation of matching cost. If these are deemed to be relevant for the specific problem at hand, length can be modeled as a region attribute (see Section 3.5). In the following we use the centroid-based approach.

3.2   Multi-Track Patterns

A query pattern can be defined on NT different tracks, i.e., $\mathcal{Q} = (Q^1, \ldots, Q^{NT})$, where $Q^x = \langle q_1^x, \ldots, q_{M_x}^x \rangle$. In this case, each pattern track Q^x is searched in a different target track $T^x = \langle t_1^x, \ldots, t_{N_x}^x \rangle$ with $N_x \geq M_x$. In the following we re-define some basic concepts to adapt them to the multi-track scenario.

Definition 3.1 (Multi-track matching). *Given two ordered collections of NT tracks, $\mathcal{Q} = (Q^1, \ldots, Q^{NT})$ and $\mathcal{T} = (T^1, \ldots, T^{NT})$, we define a multi-track matching of $\mathcal{Q}$ in $\mathcal{T}$*

as $F = (f^1, \ldots, f^{NT})$ where f^x is a matching of Q^x in T^x, $x = 1, \ldots, NT$.

As in the single-track case, we denote as $\delta_{i,j}^x = t_j^x - q_i^x$ the absolute offset between elements t_j^x and q_i^x of tracks Q^x and T^x, respectively.

Definition 3.2 (Multi-track Root-element Matching Cost). *Given two ordered collections of NT tracks, $\mathcal{Q}$ and $\mathcal{T}$, and a multi-track matching F of $\mathcal{Q}$ in $\mathcal{T}$, let $\tau_y^r = t_{f^y(1)}^y - q_1^y$, $y = 1, \ldots, NT$. The multi-track root-element matching cost $C_F^r(\mathcal{Q}, \mathcal{T})$ is defined as:*

$$C_F^r(\mathcal{Q}, \mathcal{T}) = \min_y \min_{\tau_y^r} \left\{ \sum_{x=1}^{NT} \sum_{i=1}^{M_x} (\delta_{i,f^x(i)}^x - \tau_y^r)^2 \right\} \quad (5)$$

Given a multi-track matching F, the above definition basically uses as root-elements the couple $(q_1^y, t_{f^y(1)}^y)$ that introduces the minimal cost for F. This allows us to give the same importance to all the tracks, thus without pre-defining a rank between them. The multi-track R-BMP is consequently defined as the problem of finding the multi-track matching F^* with minimum multi-track root-element matching cost, i.e., $C_{F^*}^r(\mathcal{Q}, \mathcal{T}) \leq C_F^r(\mathcal{Q}, \mathcal{T}) \, \forall F$.

The DP-RBMP algorithm can be easily adapted to the multi-track case as follows. Instead of having a single matrix, the algorithm has now NT matrices $Z_{f^y(1)}^x$, $x = 1, \ldots, NT$. Assume the root-element $t_{f^y(1)}^y$ is chosen from T^y, $y = 1, \ldots, NT$. For each of the $N_y - M_y + 1$ possible values of the root-element, we fill matrix $Z_{f^y(1)}^y$ as in the single-track case. On the other hand, for each other matrix $Z_{f^y(1)}^x$, $x \neq y$, we need to also fill the 1st row, since there is no root-element defined for T^x. For the same reason, in each row i the range of cells to be filled is now $j = i, \ldots, N_x - M_x + 1$. No other changes are needed to the logic of the algorithm.

The complexity of the DP-RBMP algorithm in the multi-track case is $\mathcal{O}\left(\sum_y N_y \sum_x M_x N_x \right)$, where $\sum_y N_y$ accounts for the number of possible root elements and $\sum_x M_x N_x$ is the total size of the NT matrices. Since the dominant term is $\max_y \{N_y\} \max_x \{M_x N_x\}$, when all tracks have the same size N and the number of tracks, NT, is fixed, the complexity can be written as $\mathcal{O}(MN^2)$, where $M \overset{\text{def}}{=} \max_x \{M_x\}$.

Also the windowed DP algorithm WDP-RBMP can be adapted in a similar manner, with the additional extension that, given the root-element $t_{f^y(1)}^y$, filling $Z_{f^y(1)}^x$ with $x \neq y$ requires the closest match $cm^x(i)$ to be computed also for the first row of the matrix. Combining the above complexity analysis with the one developed for WDP-RBMP in the single-track case, it is immediate to conclude that the complexity of the multi-track WDP-RBMP is $\mathcal{O}\left(\sum_y N_y \sum_x M_x(\log N_x + M_x) \right)$.

Example 4. *Let $\mathcal{Q} = (Q^1, Q^2)$ where Q^1 is as Q in Example 2 and $Q^2 = \langle 8, 24 \rangle$. Let $\mathcal{T} = (T^1, T^2)$ where T^1 is as T in Example 2 and $T^2 = \langle 1, 3, 12, 22, 31, 60, 78, 80, 92 \rangle$. As in the previous example, we consider as matched root-elements on Q^1 and T^1 the pair $(8, 15)$, thus $\tau^r = 7$. Figure 7 shows the matrix Z_2^1 generated by the DP-RBMP algorithm (clearly, matrix Z_2^1 corresponds to matrix Z_2 in Figure 5). It can be seen that for Z_3^2 we also fill cells in the first row, since the root-element is not chosen from T^2.*

For this choice of root-elements the best multi-track matching is $F = (f^1, f^2)$, where:

- f^1 equals f in Example 2, i.e., $f^1 = (((1, 2)), ((2, 4)), ((3, 5)), ((4, 7)))$;
- f^2 for Z_2^2 is obtained in cell $(2, 5)$, with $f^2 = (((1, 3)), ((2, 5)))$ that assigns $(12, 31)$ to $(8, 24)$.

The cost of F for the considered root-elements $(8, 15)$ is $40 + 9 = 49$, but to determine the multi-track root-element matching cost $C_F^r(\mathcal{Q}, \mathcal{T})$ it is necessary to compute also the cost of F when τ^r depends on the matched root-elements $(8, 12)$ on the second track T^2 (this is done by the DP-RBMP algorithm afterwards, when it will choose the root-elements on T^2).

3.3 Negative Matchings

In this section we consider *negative matching* tracks, i.e., target tracks in which there must be no regions in the area of a result. Notice that negative tracks are meaningful only in presence of non-negative (i.e., *positive*) tracks. Given an instance of (multi-track) R-BMP, the negative matching tracks are used to limit the space of the solutions. Any negative matching track $B = \langle b_1, \ldots, b_N \rangle$ has an associated "buffer" value β that states how distant any matched element in the positive track(s) must be from any b_i.

Definition 3.3 (Valid Area). *Given a negative matching track $B = \langle b_1, \ldots, b_N \rangle$ and its associated buffer value β, a valid area VA is an interval on the genome such that, for any value $v \in VA$, it is:*

$$v \geq b_j + \beta, \quad \text{if } b_j \leq v \qquad j = 1, \ldots, N;$$
$$v \leq b_j - \beta, \quad \text{if } b_j \geq v \qquad j = 1, \ldots, N.$$

Further, VA is maximal, *i.e., it is not included in any other interval with the same properties.*

Valid areas are those intervals in the genome which can contain results (as found in the positive tracks). From an algorithmic viewpoint, the matrix columns of each element of a target track *not* belonging to any valid area can therefore be dropped before starting the search process.

Notice that, in the presence of multiple negative matching tracks, valid areas are defined as the intersection of the valid areas of each single negative matching track.

3.4 Partial Matchings

Let us now turn to consider partial matchings, i.e., matching functions that admit missing elements.

Definition 3.4 (Partial Matching). *A partial matching of a track $QP = \langle qp_1, \ldots, qp_{MP} \rangle$ in another track $TP = \langle tp_1, \ldots, tp_{NP} \rangle$ is a strictly increasing function $fp : [1, MP] \to [1, NP] \cup \{\bot\}$ that assigns to each element of QP either an element of TP or $\bot$ (null element).*

Notice that, in the above definition, "strictly increasing function" precisely means: if both $fp(i) \neq \bot$ and $fp(\ell) \neq \bot$, then $i < \ell \Rightarrow fp(i) < fp(\ell)$. The rationale behind partial matchings is that, in some cases, requiring to match all pattern elements might lead to a poor solution. In such cases, it might be preferable to "skip" one or more pattern elements, matching only the remaining ones. For instance, given $QP = \langle 10, 19, 20 \rangle$ and $TP = \langle 10, 20, 100 \rangle$, the only

Z_2^2	**1**(1)	**2**(3)	**3**(12)	**4**(22)	**5**(31)	**6**(60)	**7**(78)	**8**(80)	**9**(92)
1 (8)	196 (1)	144 (2)	9 (3)	49 (4)	256 (5)	2025 (6)	3969 (7)	4225 (8)	
2 (24)		980 (1, 2)	505 (2, 3)	90 (3, 4)	9 (3, 5)	850 (3, 6)	2218 (3, 7)	2410 (3, 8)	3730 (3, 9)

Fig. 7. The matrix Z_2^2 of Example 4 (matrix Z_2^1 equals matrix Z_2 in Figure 5).

"complete" matching would force to include the match pair $(20, 100)$. On the other hand, allowing element 19 of QP to remain unmatched, one would obtain the partial matching that assigns $\langle 10, \perp, 20 \rangle$ to QP.

Definition 3.5 (Multi-track partial matching). *Given two collections of NP tracks, $\mathcal{QP} = (QP^1, \ldots, QP^{NP})$ and $\mathcal{TP} = (TP^1, \ldots, TP^{NP})$, we define a multi-track partial matching of $\mathcal{QP}$ in $\mathcal{TP}$ as $FP = (fp^1, \ldots, fp^{NP})$ where each fp^x is a partial matching of QP^x in TP^x, $x = 1, \ldots, NP$.*

Given an instance of multi-track R-BMP, it is reasonable to require that at least one track does not allow partial matchings. The tracks that admit partial matching are called *partial matching tracks* and each of them is associated with a (possibly different) value $c(\perp)$, that is the cost to be paid for not matching an element of that track. The following definition considers, for simplicity, the case where all pattern tracks but one are partial ones.

Definition 3.6 (Partial Root-element Matching Cost). *Given tracks $Q = \langle q_1, \ldots, q_M \rangle$ and $T = \langle t_1, \ldots, t_N \rangle$, $N \geq M$, let f be a matching of Q in T, and $\tau^r = t_{f(1)} - q_1$. Given two collections of NP partial matching tracks $\mathcal{QP} = (QP^x)$ and $\mathcal{TP} = (TP^x)$, $x = 1, \ldots, NP$, with corresponding costs $c^x(\perp)$ for null elements, and a multi-track partial matching $FP = (fp^1, \ldots, fp^{NP})$ of $\mathcal{QP}$ in $\mathcal{TP}$, the partial root-element matching cost $C^r_{f,FP}(Q, T)$ is defined as:*

$$C^r_{f,FP}(Q, T) = \sum_{i=1}^{M} cost(i, f(i)) + \sum_{x=1}^{NP} \sum_{i=1}^{MP_x} cost(i, fp^x(i)) \tag{6}$$

where MP_x is the number of elements in QP^x, $cost(i, f(i)) = (\delta_{i,f(i)} - \tau^r)^2$ as usual, and

$$cost(i, fp^x(i)) = \begin{cases} (\delta^x_{i,fp^x(i)} - \tau^r)^2 & \text{if } fp^x(i) \neq \perp \\ c^x(\perp) & \text{if } fp^x(i) = \perp \end{cases}$$

Notice that the value of τ^r used in the above equation depends on the positive (i.e., non-partial) pattern track. By definition, the elements of partial matching tracks are never used as root-elements, since they can be null. In a similar manner, it is possible to define the matching cost when multiple positive tracks appear in the pattern query, along the lines of Definition 3.2.

In order to apply the DP-RBMP algorithm to the partial matching case, a matrix $ZP^x_{f(1)}$ for each partial matching track TP^x is needed. The cost of cell (i, j) of matrix $ZP^x_{f(1)}$ must now consider also the null ($\perp$) case, thus the dynamic programming equation is modified as follows:

$$ZP^x_{f(1)}(i, j) = \min\{ZP^x_{f(1)}(i - 1, h') + (\delta_{i,j} - \tau^r)^2, \\ ZP^x_{f(1)}(i - 1, j) + c^x(\perp)\} \tag{7}$$

where, as in the non-partial case, it is $h' = \text{argmin}_{h:\ h<j}\{ZP^x_{f(1)}(i - 1, h)\}$.

Notice that, since it might well be the case that the first element, qp^x_1, of a partial matching track QP^x has a null match, matrix $ZP^x_{f(1)}$ needs to be extended with an additional "0" column, corresponding to the $\perp$ case.

Example 5. *Let Q and T as in Example 2. Let $\mathcal{QP} = (QP^1)$, where $QP^1 = \langle 5, 25 \rangle$ and $c^1(\perp) = 500$. Let $\mathcal{TP} = (TP^1)$, where $TP^1 = \langle 1, 4, 12, 21, 35, 42, 60, 71, 80 \rangle$. Let us consider Z_2 as in Figure 5 and ZP^1_2 as in Figure 8, i.e., we choose as matched root-elements $(8, 15)$, with $\tau^r = 7$.*
The cell cost of, say, cell $(2, 2)$ of ZP^1_2, is given by:

$$ZP^1_2(2, 2) = \min\{ZP^1_2(1, h') + ((4 - 25) - 7)^2, ZP^1_2(1, 2) + 500\} \\ = \min\{ZP^1_2(1, 2) + 784, 64 + 500\} = 564$$

and $ML^1_2(2, 2) = ML^1_2(1, 2) + (\perp) = (2, \perp)$.
For this choice of root-elements the best matching is:

- *f for Z_2 is as in Example 2, i.e., $f = (((1, 2)), ((2, 4)), ((3, 5)), ((4, 7)))$;*
- *fp^1 for ZP^1_2 in cell $(2, 5)$: $fp^1 = (((1, 3)), ((2, 5)))$ that assigns $(12, 35)$ to $(5, 25)$.*

whose cost is $C^r_{f,FP}(Q, T) = 40 + 9 = 49$.

The WDP-RBMP algorithm can be similarly adapted to work with partial matching tracks: First, when filling matrix $ZP^x_{f(1)}$, it is also necessary to include the additional column "0"; second, after having defined the DP-window, also all the cells "below" window cells need to be considered (since they account for null matches), i.e., if cell (i, j) belongs to the DP-window, then all cells (i', j) with $i' > i$ that are not in the DP-window are also to be filled. Figure 9 shows the matrix ZP^1_2 of Example 5 filled using the windowed DP algorithm (again, matrix Z_2 is as in Figure 5).

3.5 Region Attributes

In this section we consider how to deal with the last feature that is missing in the base model, namely region attributes. As said, each genomic region is associated with a set of region attributes $A = \{a_1, \ldots, a_L\}$, which are particular to the specific experiment performed on the genome. When searching for similar patterns, region attributes are used to determine the *region distance* of each couple of matched regions, where the importance of each attribute a_i in determining such distance is given by a weight w_i. Notice that the region length, introduced in Section 3.1, can be considered as another region attribute.

Definition 3.7 (Region Distance). *Given a couple of matched regions (qr, r), with region attributes $A = \{a_1, \ldots, a_L\}$ and corresponding weights $W = \{w_1, \ldots, w_L\}$, the region distance between qr and r is defined as:*

$$d_{reg}(qr, r) = \sum_{l=1}^{L} w_l (r.a_l - qr.a_l)^2 \tag{8}$$

ZP_2^1	**0**($\perp$)	**1**(1)	**2**(4)	**3**(12)	**4**(21)	**5**(35)	**6**(42)	**7**(60)	**8**(71)	**9**(80)
1 (5)	500 ($\perp$)	121 (1)	64 (2)	0 (3)	81 (4)	529 (5)	900 (6)	2304 (7)	3481 (8)	4624 (9)
2 (25)	1000 ($\perp, \perp$)	621 (1, $\perp$)	564 (2, $\perp$)	464 (2,3)	121 (3,4)	9 (3,5)	100 (3,6)	784 (3,7)	1521 (3,8)	2304 (3,9)

Fig. 8. The matrix ZP_2^1 of Example 5 (matrix Z_2 is as in Figure 3).

ZP_2^1	**0**($\perp$)	**1**(1)	**2**(4)	**3**(12)	**4**(21)	**5**(35)	**6**(42)	**7**(60)	**8**(71)	**9**(80)
1 (5)	500 ($\perp$)			**0** (3)						
2 (25)	1000 ($\perp, \perp$)			500 (3, $\perp$)		**9** (3,5)				

Fig. 9. The matrix ZP_2^1 of Example 5 filled using WDP-RBMP (matrix Z_2 is as in Figure 5, closest matches are in boldface).

We are ready to define the overall cost of a matching:

Definition 3.8 (Root-element Overall Cost). *Given two collections of regions $QR = \langle qr_1, \ldots, qr_M \rangle$ and $TR = \langle tr_1, \ldots, tr_N \rangle$, $N \geq M$, tracks Q and T respectively extracted from QR and TR, and a matching f of Q in T, let $\tau^r = t_{f(1)} - q_1$. The root-element overall cost $C_f^r(QR, TR)$ is defined as follows, where $C_f^r(Q, T)$ is as in Equation 2:*

$$C_f^r(QR, TR) = C_f^r(Q, T) + \sum_{i=1}^{M} d_{reg}(qr_i, tr_{f(i)}) \quad (9)$$

The overall R-BMP is consequently defined as the problem of finding the matching f^* with minimum root-element overall cost, $C_{f^*}^r(QR, TR) \leq C_f^r(QR, TR) \, \forall f$. Clearly, similar definitions can be given for all the variants of the problem (interval regions, multiple, partial, and negative tracks). Notice that, for an unmatched element q_i, only the structural cost is defined, i.e., it is assumed that $d_{reg}(qr_i, \perp) = 0$.

The DP-RBMP algorithm can be used to solve the overall R-BMP, whereas the introduction of the region distance makes the WDP-RBMP algorithm inapplicable. This is due to the fact that the region distance is not monotone and, therefore, it is not possible to limit the search only to a neighborhood of the closest matches of q_i elements.

3.6 Top-K Queries

The Top-K version of R-BMP aims to discover the K matchings $F = \{f_1, \ldots, f_K\}$ with the smallest overall cost. We further require that resulting patterns have no regions in common, so as to increase the diversity of the result. This is obtained by comparing results produced by all the matrices and keeping the best K disjoint results; note that each matrix can produce at most one result, as all the matchings associated with a matrix share the same root element.

4 BIOLOGICAL EXAMPLES

We applied the developed pattern-search to the two relevant biological problems described in the introduction.

4.1 Finding Enhancer Regions

This problem involves the search for DNA regions outside of the genes, at a certain distance from a gene's transcription start site (TSS),[11] and associated with the presence or absence of specific regions, where given molecular modifications occur or certain proteins bind to the DNA, which can be found through NGS ChIP-seq experiments. In particular, biologists generally believe that an *active putative enhancer* (APE) region should be not closer than 20K bases to the closest TSS, and have presence of overlapping H3K4me1 and H3K27ac regions, and absence of H3K4me3 regions.[12] Furthermore, an APE could optionally include overlapping regions of DHS, CTCF, P300, and/or Pol2 [17], [26].[13]

Thus, based on our problem formulation, the search for APE regions can be expressed as a multi-track, interval region matching problem where H3K4me1 and H3K27ac constitute two positive matching tracks, TSS and H3K4me3 constitute two negative matching tracks, and DHS, CTCF, P300, and Pol2 constitute four partial matching tracks, respectively. To ensure that the distance between a TSS region and a result is not less than 20K bases, we set to 20K the β_{TSS} parameter. On the other hand, for H3K4me3 a lower value of $\beta_{\text{H3K4me3}} = 1K$ is set, so that results are sufficiently distant from H3K4me3 regions.

We obtained human genome experimental data as follows. For TSS regions, we used public data from SwitchGear Genomics,[14] which are provided by the UCSC annotation database.[15] For all other genomic data types, we considered ChIP-seq experiments on specimens of the K562 cell line (Chronic Myeloid Leukemia), which are publicly available in the ENCODE project repository;[16] thus, we extracted all H3K4me1, H3K27ac, H3K4me3, DHS, CTCF, P300 and Pol2 samples of ChIP-seq regions of the K562 cell line, we merged sample replicates, and created a single dataset with a single sample (track) for each considered type of genomic data, all samples with the same region attributes. The number of regions in each sample are listed in Table 1 (a).

We applied our desktop application[17] to the considered dataset, using the pattern described above. From a user point of view, it is convenient to provide *normalized* results,

11. The genome location indicating where gene transcription (i.e., activity) starts.

12. H3K4me1, H3K27ac, and H3K4me3 are three types of molecular modifications occurring in the genome, whose location can be identified through specific NGS ChIP-seq experiments.

13. DHS are regions of the genome where DNA is accessible for the binding of proteins; CTCF, P300, and/or Pol2 are proteins that bind to the DNA in specific regions, which can again be identified through particular NGS ChIP-seq experiments.

14. http://switchgeargenomics.com/

15. https://genome.ucsc.edu/cgi-bin/hgTables?org=Human&db=hg19&hgta_group=allTracks&hgta_track=switchDbTss

16. http://genome.ucsc.edu/ENCODE/

17. http://www-db.disi.unibo.it/research/GenData/

thus switching from a cost-based matching model to a score-based one, in which scores vary in the $[0, 1]$ range, with 1 denoting a perfect match; specific details are given in Appendix B. The K-th score returned by the algorithm is reported in Table 1 (b) for different values of K.

TABLE 1
(a) Considered samples and number of the genomic regions included;
(b) Scores of the K-th results in the $[0, 1]$ range and execution times for
WDP-R$_{\text{BMP}}$ and the baseline algorithm (BA$_{\text{APE}}$).

(a)			(b)			
sample	regions		K	K-th score	WDP-R$_{\text{BMP}}$ (ms)	BA$_{\text{APE}}$ (ms)
TSS	131,780					
H3K4me1	116,503		10	0.8479	1,838	1,782
H3K27ac	45,796		50	0.7991	2,046	1,808
H3K4me3	142,738		100	0.7489	2,130	1,810
DHS	360,648		250	0.6833	2,428	2,592
CTCF	318,982		500	0.6829	3,001	2,621
P300	69,370		1,000	0.6330	5,001	4,351
Pol2	177,900					

4.1.1 Evaluation of Biological Results

Using the visualization provided by the developed software application (for an example, see Appendix C), the Top-100 matchings found were visually inspected by an expert, who evaluated all of them correct. An automatic evaluation of all obtained results is difficult: there is neither consolidated knowledge of enhancers, nor a consensus on the computational method specifically designed for their discovery. Therefore, we used for comparison a different set of data, about *chromatin*[18] *state segmentation*, generated by the Broad Institute for a few cell lines, included K562, and made publicly available also in the ENCODE repository.[19] These data, denoted as ENCODE HMM, describe a set of chromatin states of the genome using a Hidden Markov Model (HMM); in particular, each 200 base pair (i.e., nucleotide) interval is assigned to its most likely state under the model; one of such states is associated with enhancers (either strong or weak).[20]

We found a very good matching between the regions denoted as enhancers by ENCODE HMM and the regions determined by our method; we evaluated the precision of our Top-K results, i.e., how many of them overlapped with DNA regions identified as enhancers in the ENCODE HMM dataset. Additionally, we quantified how many of the overall regions that we discovered overlapped a region of any type in the ENCODE HMM dataset. Results are reported in Table 2. Out of all 1,651 regions found by our method, 1,411 (85.46%) resulted correct according to the ENCODE HMM dataset, as they overlapped a DNA region identified as either strong or weak enhancer. Note that the quality of our results is only dependent on the adopted cost function and not on the specific algorithm used to solve R-BMP.

Table 1 (b) also reports the time required by the proposed dynamic programming approach WDP-R$_{\text{BMP}}$, compared against the following baseline algorithm BA$_{\text{APE}}$, which

18. Chromatin is a complex of molecules consisting of DNA, proteins, and RNA.
19. http://genome.ucsc.edu/cgi-bin/hgFileUi?db=hg19&g= wgEncodeBroadHmm
20. Detailed information about the method and model parameters can be found in [12].

TABLE 2
Number (#) and percentage (%) of results overlapping a DNA region of any of the types in the ENCODE HMM dataset (EH).

EH region type	#	%
strong enhancer	669	40.52%
weak / poised enhancer	742	44.94%
active promoter	0	0.00%
weak promoter	0	0.00%
inactive / poised promoter	13	0.79%
insulator	49	2.97%
transcriptional transition	28	1.70%
transcriptional elongation	8	0.48%
weak transcribed	27	1.63%
polycomb repressed	39	2.36%
heterochromatin, low signal	59	3.57%
repetitive / copy number variation	8	0.48%
non overlapping	9	0.54%
total	1,651	100.00%

is able to correctly solve R-BMP when each pattern track contains a single region (without region attributes):

1) Create an ordered list where to store the locations of all regions in all considered tracks.
2) Compute a "no match" area around each region of negative tracks (in the considered example, 20K and 1K bases, for TSS and H3K4me3, respectively).
3) Obtain the windows where to search for matchings as the complement of the "no match" areas.
4) Find all possible best matchings (i.e., matchings composed of regions at minimum distance among them).
5) Order the best matchings according to the distance among their regions.
6) Return the Top-K (or all) matchings found.

Execution times of WDP-R$_{\text{BMP}}$ are only slightly (less than 10% on average) higher than those of BA$_{\text{APE}}$. On the other hand, BA$_{\text{APE}}$ only works in the case of single region tracks and does not take into account region attributes.

4.2 Searching Enhancer Pairs in Topological Domains

Although the DNA is represented as linear, indeed it makes loops which, in the 3D space, bring close DNA areas that are far apart along the DNA linear representation. Loops are associated with TADs [9] that are functional organizations of the genome; they are relatively constant in different cell types and conserved across species. They represent important regions to be further studied, e.g., for their relationships to tumors [13]. We next describe the search for enhancer-dense TADs in Myeloid Leukemia; specifically, we search for the presence of patterns of two APEs within a TAD, as an indication that the TAD is a hotspot for gene regulation. Abnormally regulated genes in Myeloid Leukemia can be next searched within these TADs.

In this case, several multi-region tracks appear in the pattern, taking full advantage of the WDP-R$_{\text{BMP}}$ algorithm. In particular, the pattern used for searching APE pairs in DNA 3D loops is illustrated in Figure 10, a task that can be expressed as a multi-track, interval region matching problem, where the TAD track refers to a DNA loop and the two (red) boxes enclose two APEs, whose tracks are as described in Section 4.1. Note in Figure 10 the presence

of one positive track with a single region (TAD) and two negative tracks (TSS and H2K3me3), two positive tracks (H3K4me1 and H3K27ac), and four partial matching tracks (CTCF, DHS, P300 and Pol2) with two regions each.
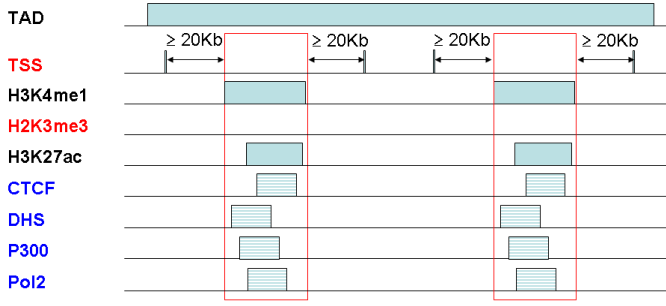

Fig. 10. Pattern used to search for APE pairs in TADs, including three positive matching tracks (black labels), two negative matching tracks (red labels), and four partial matching tracks (blue labels); red boxes indicate two APEs, which must be at least 20 Kb far from the closest TSS.

Although TADs were recently discovered [9], experimental data describing them are increasingly publicly available; we used 5,975 TAD regions for the K562 cell line obtained in [24] and available through the GEO database.[21] All other data used in our experiment were described in Section 4.1. A total of 1,081 pattern matchings were found, with a total execution time of 15.713 seconds.

4.2.1 Evaluation of Biological Results

Result evaluation was performed using the method discussed in Section 4.1.1. First, the Top-100 matchings were visually inspected by an expert, who evaluated them correct with respect to the searched pattern. Then, all the matchings were compared with the enhancer regions (of any type) in the ENCODE HMM data considered in Section 4.1.1. We evaluated the precision of our Top-K results, i.e., how many APE regions overlapped with DNA regions identified as enhancers in the ENCODE HMM dataset. Results are reported in Table 3.

TABLE 3
Precision of Top-K results obtained with our method versus DNA regions identified as enhancers in the ENCODE HMM dataset (EH). Double, Single, Not denote the number (#) and percentage (%) of matchings in which both, only one, or none of the matching APE regions overlap with enhancer regions in EH.

Top-K results	# Double	% Double	# Single	% Single	# Not	% Not
10	9	90.00%	1	10.00%	0	0.00%
50	38	76.00%	10	20.00%	2	4.00%
100	76	76.00%	21	21.00%	3	3.00%
250	187	74.80%	58	23.20%	5	2.00%
500	375	75.00%	114	22.80%	11	2.20%
(all) 1,081	742	68.64%	290	26.83%	49	4.53%

On average, 97% of the Top-100 matchings included at least one of the pair of APE regions that overlapped with an enhancer region in the ENCODE HMM dataset;

21. ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE63nnn/GSE63525/ suppl/GSE63525_K562_Arrowhead_domainlist.txt.gz

furthermore, in 76% of the matchings, both APE regions were found overlapping with enhancer regions of ENCODE HMM data. These results prove that, even in a complex pattern-matching real case, our method effectively finds correct results with high precision.

TABLE 4
Execution times for WDP-RBMP and BA_{TAD} algorithms.

K	WDP-RBMP (ms)	BA_{TAD} (ms)	BA_{TAD} overhead
10	2,176	8,430	287%
50	2,319	9,012	289%
100	4,264	10,854	155%
250	5,867	11,234	91%
500	7,297	14,093	93%
1,000	14,090	18,020	28%

Table 4 shows execution times of the WDP-RBMP algorithm and contrasts them with those obtained from an ad-hoc algorithm, BA_{TAD}, that works as follows:

1) Use the TAD track to filter out all those regions in the other tracks that do not fall into a TAD region.
2) Use the BA_{APE} algorithm to find all APEs.
3) For each "left" APE, find the best "right" APE (see Figure 10).

Ranking on APE pairs is based on the root element matching cost, where the root element is the one identified by BA_{APE} in the "left" APE. As Table 4 clearly demonstrates, WDP-RBMP runs always faster than BA_{TAD}, the latter always incurring a high overhead.

5 PERFORMANCE TESTING ON SYNTHETIC DATA

Now we evaluate the efficiency of the WDP-RBMP algorithm using synthetic datasets. We generate the position of regions by simulating a Poisson process with parameter λ representing the average distance between two adjacent regions. In particular, we varied the distance between regions in the input track(s), while the average distance between pattern regions (generated and controlled by parameter λ_p) remained constant. Table 5 summarizes the parameters involved in the synthetic data generation, as well as their ranges and default values. All implementations are in Java 8 programming language and run on a HP Pavilion dv7 PC with an i7 processor equipped with 8 GB of main memory.

TABLE 5
Parameters for synthetic data generation (default values are in bold).

parameter	symbol	range
number of pattern regions	M	5, **10**, 20, 30
avg. dist. between pattern regions	$1/\lambda_p$	10^3
number of input regions	N	10^3, $\mathbf{10^4}$, 10^5, 10^6
avg. dist. between input regions	$1/\lambda$	10^2, $\mathbf{10^3}$, 10^4

5.1 Windowed and Early Abandoning Optimizations

The first set of experiments aims at evaluating the performance improvement obtained by means of the "Early Abandoning" and the "Window" optimizations described

in Sections 2.1 and 5.1, respectively. In details, we compared four different algorithms: the basic DP-RBMP algorithm with no optimization (labeled "none" in the figures), DP-RBMP with the early abandoning optimization ("EA"), the WDP-RBMP window-based algorithm ("WIN"), and WDP-RBMP with early abandoning ("WIN_EA").

We started by assessing the efficiency of the proposed BMP solutions with respect to the base version of the problem (i.e., the single-track case), in terms of running times and number of filled cells of the matrix, by varying input data size, pattern size, and input data distribution.

Figure 11 shows (in log-log scale) the cost in term of running times (a) and number of filled cells (b) when varying the number of regions N in the target data track. Clearly, both cost measures increase with the input size for all algorithms. Due to its quadratic complexity, DP-RBMP, whether or not combined with early abandoning, is feasible for limited problem sizes only. The improvement of both versions of WDP-RBMP over DP-RBMP is more than three orders of magnitude. The early abandoning optimization (WIN_EA) is particularly effective with larger problem sizes ($N = 10^6$), where it saves about 70% of computation time.
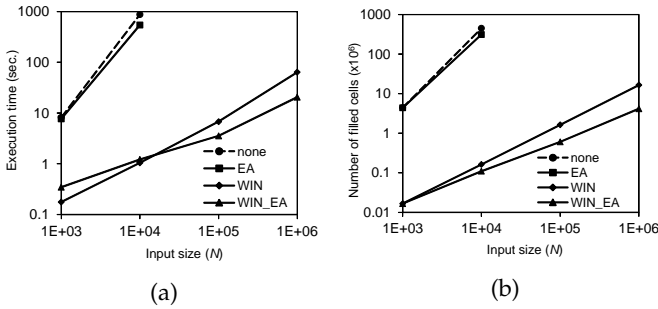

Fig. 11. Running times (a) and number of filled cells (b) by varying N.

Figure 12 shows the same cost measures, but now varying the pattern size M. As expected from the complexity analysis, in this case all algorithms grow almost linearly with M (note that here only the vertical axis is in log scale).
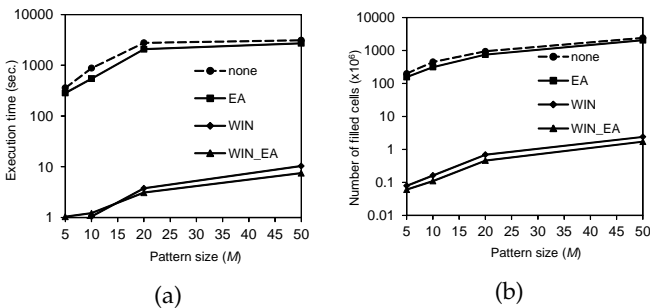

Fig. 12. Running times (a) and number of filled cells (b) by varying M.

Finally, Figure 13 shows that the performance of all algorithms is only mildly affected by the parameter λ that controls the spacing between adjacent regions.

Since we experimentally proved that *none* and EA solutions are feasible only for small data sets, in the following we focus our analysis on WIN_EA algorithm only.
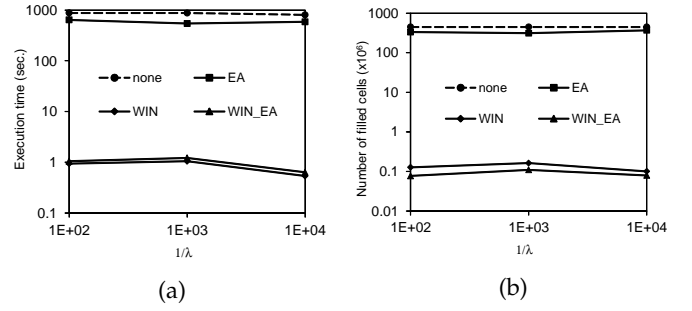

Fig. 13. Running times (a) and number of filled cells (b) by varying λ.

5.2 The Case of Multi-Tracks

Next we tested the multi-track scenario on the WIN_EA algorithm. We performed three different experiments, whose results are shown in Table 6; results were all obtained for 100,000 regions for each input data track.

In the **first experiment**, 20 pattern regions are distributed on 1, 2, 4, and 5 tracks, respectively, in order to evaluate the effect of increasing the number of tracks on performance. Execution times show a sublinear growth rate with the number of pattern tracks. To understand why this is the case, remember that, as argumented in Section 3.2, the complexity of WDP-RBMP is $\mathcal{O}\left(\sum_y N_y \sum_x M_x(\log N_x + M_x)\right)$, which simplifies to $\mathcal{O}\left(NT \cdot N(\log N \sum_x M_x + \sum_x M_x^2)\right)$ when each data track T^x has $N_x = N$ regions. Although this linearly depends on NT, the quadratic term $\sum_x M_x^2$ is indeed decreasing with the number of tracks, since $\sum_x M_x$ is constant in this experiment.

In the **second experiment**, 20 pattern regions are distributed on 4 tracks, but varying the number of regions on each track from one extreme (equi-distribution) to the other extremes (1 region on all tracks except one). The cost increase for skewed distributions is due to the fact that, again, the $\sum_x M_x$ term remains constant, while $\sum_x M_x^2$ grows with the skew of the distribution. Finally, since tracks are not ranked, the cost remains constant independently on which track has the highest M_x value (compare results for [1,1,1,17] and [17,1,1,1] distributions).

In the **third experiment**, each pattern track has 10 regions and we compare costs for 1, 2, and 4 tracks. Here the super-linear cost trend is clearly due to the term $NT \cdot N \sum_x M_x^2$.

TABLE 6
Execution times for different multi-track pattern region distributions.

pattern region distribution	time (sec.)
[20]	7.677
[10,10]	12.152
[5,5,5,5]	16.224
[4,4,4,4,4]	22.056
[5,5,5,5]	16.224
[1,1,1,17]	29.218
[17,1,1,1]	30.312
[10]	2.777
[10,10]	12.152
[10,10,10,10]	60.270

5.3 Partial and Negative Tracks

We proceeded to investigate the performance of our algorithms in presence of partial and negative tracks.

Figure 14 shows the WIN_EA algorithm working with and without partial tracks. In particular, the plot named "non-partial" refers to the case where the input pattern consists of two tracks, while for the "partial" graph the second track accepts partial matchings. Clearly, allowing partial matches increases costs, because a larger number of matrix cells has to be filled for the partial matching track.
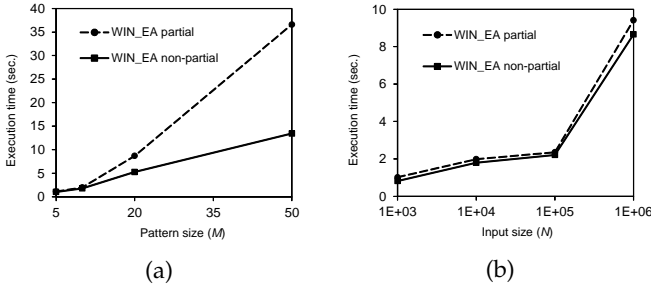

(a)   (b)

Fig. 14. Partial matching: Running times by varying M (a) and N (b).

As it can be observed from plots, the presence of partial tracks quickly increases execution times when varying the dimension of pattern (Figure 14 (a)). On the other hand, the input size has little impact with respect to the non-partial case (Figure 14 (b)).

Figure 15 shows running times (a) and number of pruned cells (b) when varying the size of negative tracks. For this experiment, the pattern query is composed by $M = 2$ regions, for which a complete matching has to be searched on a target track with $N = 10^4$ data regions with average distance $1/\lambda = 10^4$, and one negative track with buffer value $\beta = 100$. In order to obtain comparable coordinate values, the number of regions in the negative track N_- and their average distance $1/\lambda_-$ obey the relationship $N_-/\lambda_- = 10^8$. For simplicity we only show results for the "none" algorithm in which neither WIN nor EA are used. As expected, increasing the size of the negative track leads to a reduced number of cells to be filled by the algorithm, and the execution time decreases accordingly following a negative linear trend.
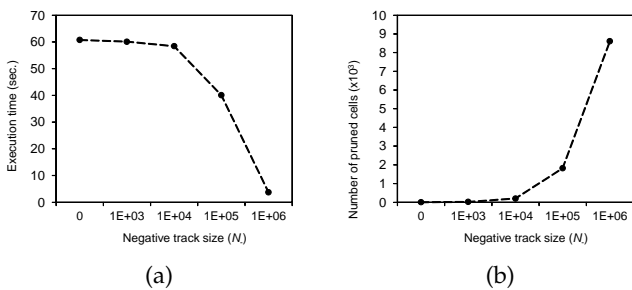

(a)   (b)

Fig. 15. Negative tracks: Running times (a) and number of pruned cells (b) by varying the size of the negative track.

6 RELATED WORK

The literature on pattern search is huge and cannot be surveyed here, because the very term "pattern" represents a variety of different things [3]; we focus our comparison on works that more closely address a problem similar to ours.

Latecki et al. [18], [19] consider the problem of "partial elastic matching" of time series, in which a sequence of real number $A = (a_1, \ldots, a_M)$ has to be matched to another sequence $B = (b_1, \ldots, b_N)$, $N \geq M$. The matching f, as in the base version of our problem, is a strictly increasing and total function. The authors prove that, when no "shift" is considered (i.e., $\tau = 0$), a dynamic programming approach can determine the optimal matching under the Euclidean distance. However, when turning to consider the general case ($\tau \geq 0$), they still claim that dynamic programming can be applied, which is definitely wrong. Besides this aspect, Latecki et al. consider neither multiple nor partial tracks, and elements to be matched do not have any attribute.

The work by Wongsuphasawat et al. [28] considers the case of multiple tracks, in which each track correspond to a specific category of events in an applicative domain. Again, dynamic programming is applied to derive an optimal multi-track matching for the case $\tau = 0$ only. It has also to be remarked that [28] is focused on the usability aspect of the problem (i.e., helping users in specifying their queries), rather than on efficiency issues.

The works dealing with Complex Event Processing (CEP) (see e.g., [8]) actually consider problems quite different from ours, in that patterns are Directed Acyclic Graphs (DAGs), each node being an event with an associated label (from a finite alphabet) that denotes the event type.

Similar patterns are also defined in stock trading, but the search for such patterns is comparatively much easier, as these patterns are fixed and stock market observations are limited to short time intervals. An approach for searching *interesting stocks* which imitate the behavior of a given stock is presented in [4], recognized by using timed automata, but the complexity of that problem is in the number of stocks (equivalent to our tracks) while considered time intervals are comparatively much shorter than the whole genome.

This work is part of the GenData 2020 project[22] that, as stated in the introduction, focuses on tertiary data analysis. The expressive power and flexibility of GenData 2020's data model (GDM) and query language (GMQL) are demonstrated in [20], where we show four very different genomic use cases; in [6] we demonstrate that our cloud-based implementation scales over large datasets.

7 CONCLUSIONS

We presented an efficient method, and its several variants, to find patterns in genomic region sequences; it has practical applications in revealing interesting and unknown regions of the genome, and therefore it is an important ingredient in supporting biological research. Our algorithms are already integrated in a tool that enables expressing patterns from within the Integrated Genome Browser, so as to facilitate the interaction of biologists. In our future work, we plan to use the method for biological research, in strong connection with biologists of the GenData 2020 project, by using experimental data produced at IEO or by connecting to public data sources.

22. http://www.bioinformatics.deib.polimi.it/gendata/

ACKNOWLEDGMENT

REFERENCES

[1] The 1000 Genomes Consortium. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491, 56-65, 2012.

[2] The 100,000 Genomes Project. http://www.genomicsengland.co.uk/ (accessed on June 2016).

[3] I. Bartolini et al. The Panda framework for comparing patterns. *Data Knowl. Eng.*, 68(2):244-260, 2009.

[4] C. Bettini et al. Discovering frequent event patterns with multiple granularities in time sequences. *IEEE Trans. Knowl. Data Eng.*, 10(2):222-237, 1998.

[5] R. E. Burkard et al. Assignment problems. SIAM, 2009.

[6] S. Ceri et al. Data management for heterogeneous genomic datasets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2016 (in press).

[7] S. Ceri et al. GenData 2020 joint research project with IEO-IIT, internal report, April 2015 (retrieved from http://www.bioinformatics.deib.polimi.it/genData/workplan-and-deliverables.html as D71).

[8] L. Ding et al. Runtime semantic query optimization for event stream processing. In *Proc. ICDE*, 676-685, 2008.

[9] J. R. Dixon et al. Topological domains in mammalian genomes identified by analysis of chromatin interactions. *Nature*, 485(7398):376-380, 2012.

[10] ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57-74, 2012.

[11] J. Ernst and M. Kellis. Discovery and characterization of chromatin states for systematic annotation of the human genome. *Nat. Biotechnol.*, 28(8):817-825, 2010.

[12] J. Ernst et al. Mapping and analysis of chromatin state dynamics in nine human cell types. *Nature*, 473(7345):43-49, 2011.

[13] A. Flavahan et al. Insulator dysfunction and oncogene activation in IDH mutant gliomas. *Nature* 529(7584):110-114, 2016.

[14] H. Gunadhi and A. Segev. Query processing algorithms for temporal intersection joins. In *Proc. IEEE ICDE*, 336-344, 1991.

[15] A. Kapadia et al. Routing with confidence: A model for trustworthy communication. UIUC Technical Report UIUCDCS-R-2006-2680 (UILU-ENG-2006-1707), 2006.

[16] W. J. Kent. The human genome browser at UCSC. *Genome Res.*, 12(6):996-1006, 2002.

[17] D. Kleftogiannis et al. Progress and challenges in bioinformatics approaches for enhancer identification. *Brief Bioinform.*, 2015.

[18] L. J. Latecki et al. Partial elastic matching of time series. In *Proc. ICDM*, 701-704, 2005.

[19] L. J. Latecki et al. An elastic partial shape matching technique. *Patt. Rec.*, 40(11):3069-3080, 2007.

[20] M. Masseroli et al. GenoMetric Query Language: A novel approach to large-scale genomic data management. *Bioinformatics*, 31(12):1881-1888, 2015.

[21] P. Montanari et al. An IGB-based application for the search of patterns in a genomic sequence. *GenData 2020 Tech. Rep.*, May 2015.

[22] J. W. Nicol et al. The Integrated Genome Browser: free software for distribution and exploration of genome-scale datasets. *Bioinformatics*, 25(20):2730-2731, 2009.

[23] Paradigm4, Inc. Accelerating bioinformatics research with new software for Big Data to Knowledge (BD2K). Waltham, MA, 2015 (available at: http://www.paradigm4.com/).

[24] S. S. Rao et al. A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. *Cell*, 159(7):1665-1680, 2014.

[25] M. Sniedovich. Solution strategies for variance minimization problems. *Computers & Mathematics with Appl.*, 21(2-3):49-56, 1991.

[26] A. Visel et al. Enhancer identification through comparative genomics. *Semin Cell Dev Biol.* 18(1):140-152, 2007.

[27] J. N. Weinstein et al. The cancer genome atlas pan-cancer analysis project. *Nat. Genet.*, 45(10):1113-1120, 2013.

[28] K. Wongsuphasawat et al. Querying event sequences by exact match or similarity search: Design and empirical evaluation. *Interacting with Computers*, 24(2):55-68, 2012.

[29] P. Zezula et al. Similarity search: The metric space approach. Springer, 2006.

Piero Montanari has a "Laurea" degree in Computer Engineering from the University of Bologna where he currently has a research fellowship sponsored by the GenData 2020 project.

Ilaria Bartolini is Associate Professor at the University of Bologna. She graduated in Computer Science and received a PhD in Electronic and Computer Engineering from the University of Bologna. She has been a visiting researcher at CWI (Amsterdam), NJIT (Newark, NJ), and HKUST (Hong Kong). Her current research interests include smart management of multimedia data, similarity and preference-based query processing, and collaborative filtering/advertising.

Paolo Ciaccia has a "Laurea" degree in Electronic Engineering and a PhD in Electronic and Computer Engineering from the University of Bologna where he is Full Professor since 2000. His current research interests include similarity and preference-based query processing, multimedia systems, and probabilistic databases. He was an Associate Editor of IEEE TKDE and currently serves on the editorial board of ACM TODS. He is one of the designers of the M-tree, an index for metric data.

Marco Patella got the "Laurea" degree in Electronic Engineering and a PhD in Electronic and Computer Engineering from the University of Bologna. Since 2006 he is an Associate Professor at University of Bologna. His current research interests include similarity- and preference-based query processing in multimedia databases. He also collaborated to the design of the metric index M-tree.

Stefano Ceri is Professor at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) of Politecnico di Milano. His research work has been generally concerned with extending database technology to incorporate new features: distribution, object-orientation, rules, streaming data, crowd-based and genomic computing. He was the coordinator of the PRIN project GenData 2020 (2013-2016). He received two Advanced ERC Grants, the seond one on Data-Driven Genomic Computing (GeCo) is currently being contracted with the EU. He is the recipient of the ACM-SIGMOD "Edward T. Codd Innovation Award" (2013), and an ACM Fellow and member of the Academia Europaea.

Marco Masseroli received a PhD in Biomedical Engineering in 1996, from Universidad de Granada, Spain. He is Associate Professor at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) of Politecnico di Milano, Italy. His research interests are in bioinformatics and biomedical informatics, focused on biomolecular databases, biomedical terminologies and ontologies to effectively retrieve, manage, analyze, and semantically integrate genomic information with patient clinical and high-throughput genomic data. He is the author of more than 170 scientific articles in international journals, books and conference proceedings.

APPENDIX A
THE BEST-MATCHING PROBLEM

In Section 2 we have considered that, for any given matching f, the τ parameter is given a fixed value, $\tau^r = t_{f(1)} - q_1$, which amounts to perfectly align the first matched elements of tracks Q and T. Without this limitation, one could consider that the matching cost of f, $C_f(Q, T)$, is the minimum τ-matching cost of f over all τ values:

$$C_f(Q, T) = \min_\tau C_f(Q, T; \tau) \qquad (10)$$

It is simple to show that the minimum is obtained for $\tau = \sum_{i=1}^{M} \delta_{i,f(i)}/M$, i.e., the matching cost equals M times the *variance* of absolute offsets $\delta_{i,f(i)}$ of matched elements:

$$C_f(Q, T) = \sum_{i=1}^{M} \left(\delta_{i,f(i)} - \frac{\sum_{i=1}^{M} \delta_{i,f(i)}}{M} \right)^2 \qquad (11)$$

This easily follows by observing that the derivative of $C_f(Q, T; \tau)$ w.r.t. τ is:

$$\frac{\partial C_f(Q, T; \tau)}{\partial \tau} = 2M\tau - 2\sum_{i=1}^{M} \delta_{i,f(i)}.$$

From this observation it follows that solving the best-matching problem (BMP), i.e., minimizing $C_f(Q, T)$ over the set of all possible matchings, amounts to finding a *minimum-variance matching*. BMP is therefore a specific case of quadratic assignment problem, which is known to be NP-hard [5]. Although this does not immediately lead to conclude that BMP is NP-hard as well, we strongly suspect this is the case. This is also supported by the observation that variance-minimization problems are reputed difficult to solve [25], although no specific lower bound on their complexity is proved.[23]

Example 6. *Let $Q = \langle 1, 7, 10 \rangle$ and $T = \langle 3, 5, 9, 11, 13, 14, 18, 21 \rangle$. One of the possible matchings of Q in T is $f = \{((1,1)), ((2,3)), ((3,8))\}$ that assigns elements $(3, 9, 21)$ to corresponding elements of Q. Assuming $\tau = 7$, the τ-matching cost of f is 66, since:*

$$C_f(Q, T; 7) = (3 - 1 - 7)^2 + (9 - 7 - 7)^2 + (21 - 10 - 7)^2 = 66$$

The matching cost of f is obtained for $\tau = ((3 - 1) + (9 - 7) + (21 - 10))/3 = 5$:

$$C_f(Q, T) = (3 - 1 - 5)^2 + (9 - 7 - 5)^2 + (21 - 10 - 5)^2 = 54$$

The solution to the BMP is $f^* = \{((1,2)), ((2,4)), ((3,7))\}$ that matches Q to elements $(5, 11, 14)$ of T, which yields $\tau = 4$. The matching cost of f^* is:

$$C_{f^*}(Q, T) = (5 - 1 - 4)^2 + (11 - 7 - 4)^2 + (14 - 10 - 4)^2 = 0$$

APPENDIX B
NORMALIZATION OF SIMILARITY SCORES

In Section 3.5 we showed how our base problem can be extended so that the overall cost of a matching can include differences between attributes of matched regions. In order

---

to provide *normalized* results, which are more easily understandable by users, it may be convenient to switch from a cost-based matching model to a score-based one. More precisely, we consider the score of a set of regions in the target track(s) with respect to the query pattern as a value in the $[0, 1]$ range, with 1 denoting a perfect match.

For obtaining the value of *structural score* for a given matching f, one could just apply a (properly normalized) monotonic decreasing transformation to $C_f^r(Q, T)$, e.g., $1 - C_f^r(Q, T)/CMAX$, with $CMAX$ a normalization factor. However, in order to mitigate the effect that single cost (score) components might have on the overall cost (resp. score) of a matching, we found it more appropriate to also introduce a non-linear transformation to be applied to the single components, $cost(i, f(i))$, of the matching f. In particular, we adopt the following sigmoid (logistic) function:

$$sigm(cost(i, j)) = 1 - \frac{1}{1 + e^{-sl(cost(i,j) - mp)}} \qquad (12)$$

where sl and mp are two parameters that define the slope of the function and the mid-point (i.e., the point at which the function equals $1/2$), respectively. Since for low values of sl the value of $sigm(0)$ (perfect match) can be much lower than 1, we measure the structural score of q_i and t_j as:

$$s_{str}(q_i, t_j) = \frac{sigm(cost(i, j))}{sigm(0)} \qquad (13)$$

Figure 16 shows how the structural score depends on the matching cost, when $sl = 0.02$ and $mp = 250$.
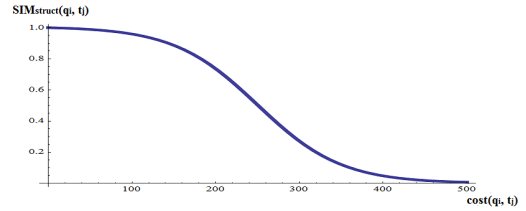

Fig. 16. Structural score vs. matching cost.

Notice that this non-linear transformation might produce an optimal matching which is different from the one that minimizes the cost, because of the reduced influence that single poorly-matched regions now have on the result.

Similarly, the *region score* takes into account differences between attributes of matched regions:

$$s_{reg}(qr, r) = 1 - \frac{\sum_{l=1}^{L} w_l(r.a_l - qr.a_l)^2}{\sum_{l=1}^{L} w_l}$$

Note that $s_{reg}(qr, r) \in [0, 1]$ because of the normalization factor $\sum_{l=1}^{L} w_l$.

Finally, the *root-element matching score* $s_f^r(QR, TR)$ is defined as:

$$s_f^r(QR, TR) = \frac{1}{M} \sum_{i=1}^{M} s_{str}(q_i, t_{f(i)}) \cdot s_{reg}(qr_i, tr_{f(i)}) \qquad (14)$$

The score R-BMP (SIM-R-BMP) is consequently defined as the problem of finding the matching f^* with maximum root-element matching score, $s_{f^*}^r(QR, TR) \geq s_f^r(QR, TR)$ $\forall f$. Clearly, similar definitions can be given for all the variants of the problem (interval regions, multiple, partial, and

---

23. A remarkable exception we are aware of is [15], which proves that finding a minimum-variance path in a cyclic graph is NP-hard.

negative tracks). Notice that, for an unmatched element q_i, only the structural score is defined with a value obtained by plugging $cost(i, \perp) = c(\perp)$ into Equation 12. For instance, with $c(\perp) = 500$, $sl = 0.02$, and $mp = 250$, we obtain from Equation 13 $s_{str}(q_i, \perp) \approx 0.0067$. In the experiment on real biological data described in Sections 4.1 and 4.2, $c(\perp)$ was set so as to obtain $s_{str}(q_i, \perp) = 0.7$ for unmatched regions of all the four partial matching tracks.

Finally, the DP-RBMP algorithm can be used to solve the SIM-R-BMP by maximizing cell score rather than minimizing cell costs, while the same considerations expressed in Section 3.5 prevent the use of the WDP-RBMP algorithm.

Example 7. *Let us use the same tracks of Example 2: $Q = \langle 8, 20, 22, 36 \rangle$ and $T = \langle 10, 15, 17, 27, 35, 39, 45, 50, 62, 70 \rangle$. Figure 17 shows two matrices: the first, Z_2, contains the cell costs, as computed in Example 2; the second one, denoted SZ_2, contains the cell scores, assuming all region scores equal 1. The parameters used for the sigmoid function are $sl = 0.01$ and $mp = 100$. The first matrix in Figure 18 shows fictitious region scores, $s_{reg}(qr_i, tr_j)$, and the matrix below shows how cell scores change when also these values are considered (compare this to matrix SZ_2 in Figure 17).*

The best matching f, when $f(1) = 2$, is obtained in cell $(4, 6)$, with overall score $s_f^r(QR, TR) = 0.75$.

APPENDIX C
VISUALIZATION OF RESULTS

The proposed DP algorithm for solving R-BMP has been developed as a stand-alone desktop application,[24] that enables biologists to define patterns of interest using the Integrated Genome Browser [22]. The software application also allows executing the created pattern-search algorithm and visualizing the matching regions found along the genome for their evaluation. As an example, Figure 19, shows the two best results found for the problem described in Section 4.1: Top-1 on chromosome 12 and Top-2 on chromosome 11. Both best matches include a region in each of the two positive matching tracks (H3K4me1 and H3K27ac), as required, and a region in only some of the partial matching tracks (DHS and Pol2 in Top-1, and DHS in Top-2), (negative tracks are not shown).

---

24. Available for download at http://www-db.disi.unibo.it/research/GenData/.

Z_2	1(10)	2(15)	3(17)	4(27)	5(35)	6(39)	7(45)	8(50)	9(62)	10(70)
1 (8)		0 (2)								
2 (20)			100 (2,3)	0 (2,4)	64 (2,5)	144 (2,6)	324 (2,7)	529 (2,8)		
3 (22)				104 (2,3,4)	36 (2,4,5)	100 (2,4,6)	256 (2,4,7)	441 (2,4,8)	1296 (2,4,9)	
4 (36)					168 (2,3,4,5)	52 (2,4,5,6)	40 (2,4,5,7)	85 (2,4,5,8)	520 (2,4,5,9)	877 (2,4,5,10)

SZ_2	1(10)	2(15)	3(17)	4(27)	5(35)	6(39)	7(45)	8(50)	9(62)	10(70)
1 (8)		0.25 (2)								
2 (20)			0.42 (2,3)	1 (2,4)	0.45 (2,5)	0.39 (2,6)	0.28 (2,7)	0.26 (2,8)		
3 (22)				0.59 (2,3,4)	0.73 (2,4,5)	0.67 (2,4,6)	0.56 (2,4,7)	0.51 (2,4,8)	0.5 (2,4,9)	
4 (36)					0.70 (2,3,4,5)	0.94 (2,4,5,6)	0.95 (2,4,5,7)	0.91 (2,4,5,8)	0.73 (2,4,5,9)	0.73 (2,4,5,10)

Fig. 17. The matrix Z_2 from Example 2 and the corresponding score matrix SZ_2.

	1(10)	2(15)	3(17)	4(27)	5(35)	6(39)	7(45)	8(50)	9(62)	10(70)
1 (8)		0.92								
2 (20)			0.53	0.81	0.62	0.92	0.23	0.62		
3 (22)				0.45	0.67	0.34	0.87	0.92	1	
4 (36)					0.87	0.80	0.57	0.31	0.98	0.72

SZ_2	1(10)	2(15)	3(17)	4(27)	5(35)	6(39)	7(45)	8(50)	9(62)	10(70)
1 (8)		0.23 (2)								
2 (20)			0.32 (2,3)	0.43 (2,4)	0.36 (2,5)	0.35 (2,6)	0.24 (2,7)	0.23 (2,8)		
3 (22)				0.40 (2,3,4)	0.58 (2,4,5)	0.49 (2,4,6)	0.49 (2,4,7)	0.44 (2,4,8)	0.43 (2,4,9)	
4 (36)					0.50 (2,3,4,5)	0.75 (2,4,5,6)	0.71 (2,4,5,7)	0.64 (2,4,5,8)	0.59 (2,4,5,9)	0.58 (2,4,5,10)

Fig. 18. The matrix of region scores (top) and the SZ_2 matrix of cell scores (bottom) considering also region scores (compare to matrix SZ_2 in Figure 17).
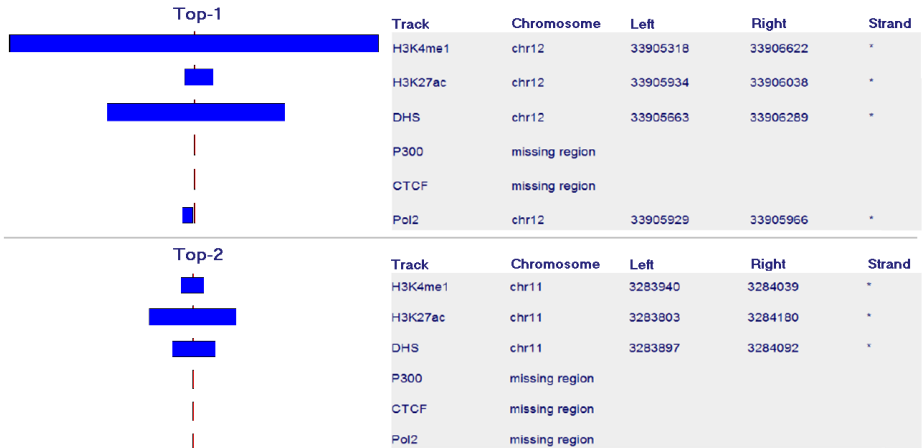

Fig. 19. Screenshot of the implemented software application showing the first (Top-1) and second (Top-2) best matching results.