ChromStruct 4: A Python Code to Estimate the Chromatin Structure from Hi-C Data

Claudia Caudai, Emanuele Salerno, Monica Zoppè, Ivan Merelli, and Anna Tonazzini

Abstract—A method and a stand-alone Python code to estimate the 3D chromatin structure from chromosome conformation capture data are presented. The method is based on a multiresolution, modified-bead-chain chromatin model, evolved through quaternion operators in a Monte Carlo sampling. The solution space to be sampled is generated by a score function with a data-fit part and a constraint part where the available prior knowledge is implicitly coded. The final solution is a set of 3D configurations that are compatible with both the data and the prior knowledge. The iterative code, provided here as additional material, is equipped with a graphical user interface and stores its results in standard-format files for 3D visualization. We describe the mathematical-computational aspects of the method and explain the details of the code. Some experimental results are reported, with a demonstration of their fit to the data.

Index Terms—Chromosome conformation capture, chromatin configuration, Bayesian estimation

1 Introduction

Chromosomal organization is involved in regulation of gene function [1]; this is why the possibility of estimating the spatial configuration of DNA in cell nuclei has recently been the focus of many studies in structural biology [2]. When completely stretched out, the human DNA double helix is about 2 m long, but must fit in a micron-size nucleus. This means that DNA must always be tightly packed. The DNA chain plus the macromolecules that mediate its packing are called *chromatin*. To enable the nuclear functions during the different phases of the cell life, the chromatin has several levels of packing, from a 10 nm "bead on a string" fiber mediated by histone proteins—probably further wrapped in a helical 30 nm fiber—to the complex configurations assumed during interphase.

A number of recent experimental techniques of the type *chromosome conformation capture* is producing an increasing interest in chromatin studies. Among these techniques, the one called Hi-C [3] produces high-throughput, genome-wide contact frequencies between pairs of genomically identified loci in the nuclear DNA of populations of homogeneous cells. Such information is directly usable to study the mutual influence of even distant genomic sites, thus clarifying many aspects of gene regulation and epigenetics. Assuming that two loci that are often in contact within the analyzed cell population are likely to be spatially close, the same data can be used to estimate the 3D structure of the chromatin.

- *C. Caudai, E. Salerno, and A. Tonazzini are with the National Research Council of Italy, Institute of Information Science and Technologies, Pisa 56127, Italy. E-mail: {claudia.caudai, emanuele.salerno, anna.tonazzini}@isti.cnr.it.*
- *M. Zoppè is with the National Research Council of Italy, Institute of Clinical Physiology, Pisa 56124, Italy. E-mail: mzoppe@ifc.cnr.it.*
- *I. Merelli is with the National Research Council of Italy, Institute of Biomedical Technologies, Milan, 20090, Italy. E-mail: ivan.merelli@itb.cnr.it.*

Manuscript received 1 June 2017; revised 14 May 2018; accepted 16 May 2018. Date of publication 21 May 2018; date of current version 5 Dec. 2019. (Corresponding author: Emanuele Salerno.)
Digital Object Identifier no. 10.1109/TCBB.2018.2838669

Basically, two strategies have been proposed to make the estimated structures useful in biological research. One aims at finding a unique consensus solution, a sort of *reference chromatin*, averaged onto the different configurations in the experimental population; the other prefers to produce a set of different solutions fitting the data, as the millions of cells in a Hi-C experiment can be in different phases of their life-cycle [4]. Within these two strategies, the estimation algorithms proposed in the literature differentiate for the data model, the solution model and the computational approach chosen. A popular data model (see, e.g., [5], [6]) translates the input contact frequencies into euclidean distances and then estimates the structures by solving a distance-to-geometry problem. The solution models adopted are either geometrical [6], [7], [8] or physical-geometrical [9], constrained by assumedly known geometrical-topological [8] or polymeric [10] chromatin properties. As far as the computational approach is concerned, the solutions proposed range from constrained optimization [7], [8], [11] to fully Bayesian approaches [12], [13]. More details can be found in [2], [14], [15], [16], [17], [18]; here, we make some short reference to previous works where we need to remark their similarities with, or differences from, our approach.

Opting for the "set-of-consistent-solutions" strategy, in [19], we proposed a model for data and solution intended to develop an efficient algorithm and overcome some of the drawbacks of other methods. Our data model takes the Hi-C data as they are, with no need to translate the contact frequencies into euclidean distances. This avoids the invariable incompatibility between the distance sets derived from Hi-C data and the euclidean geometry [20], [21]. To obtain an efficient reconstruction algorithm, we proposed a multiscale, modified-bead chain model of the chromatin fiber. At the scale provided by the experimental data, each locus is considered as a bead in a chain, and the output structures are described as lists of the 3D coordinates of the bead centroids.

A basic finding helps us to complete our solution model [1], [16]: the chromatin fiber is folded in a way that there are genomic domains characterized by many internal interactions and very few interactions with the rest of the chain. This property shows up at any scale. As examples, the genomic compartments and the topological association domains (TADs) show this property, respectively, at the 1-Mbp and the 100-kbp scales [1], [22].[1] This very property also allows us to partition the chain at the finest scale available and estimate separately the structure of each subchain thus extracted. In turn, each reconstructed subchain can be modeled as a single bead in a coarser-scale chain, which can further be partitioned and solved in the same way, provided that the original data matrix is appropriately binned to the new, nonuniform scale. This procedure can be repeated until no more isolated domains can be detected. Once all the subchains at all scales are estimated, we are left with the problem of putting them together respecting their mutual relationships and rise back to the finest scale with the complete chain. This is made possible by the particular structure we give to our beads (see Section 2.2), which, at each scale, enables us to replace the beads with their generating subchains appropriately positioned and oriented, and then reconstructing the chain at finer and finer scales, until the original genomic resolution is reached. Apparently, this is a typical recursive method. Indeed, we implemented it by several recursive codes, using Monte Carlo strategies to sample the solution spaces generated by a specially designed score function. From a proof-of-concept method and code (denoted here as *CromStruct 1* [19]), we arrived at a completely automated version with an improved score function, also equipped by a convenient GUI (ChromStruct 3.2 [25]). However, since all the scale levels and the related contact matrices can immediately be determined from the input data, the method can also be implemented iteratively, with advantages in memory use and new potentialities of parallel implementation. The Python[2] code described here in detail, ChromStruct 4.2, is our last iterative implementation of the method, obtained after adding functionalities and an optional standard PDB output format to the previous versions.

This paper can be read at different levels of detail. In Section 2, we describe our method with no reference to its implementation. In Section 3, we detail the most important procedures implemented in ChromStruct 4.2. The full detail can be achieved through the self-contained, extensively commented source code and the usage notes made available here as additional material. Section 4 features a set of experimental results, complemented with further experiments described in the appendices provided as a supplemental file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2018.2838669. Some final remarks are given in Section 5.

2 Method

In this section, we explain the rationale of our choices in devising the estimation method. For a deeper discussion, we refer to [25], where, except for the iterative implementation, we explain how our strategy was conceived.

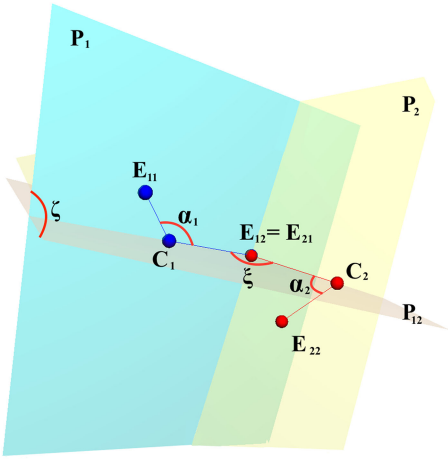

Fig. 1. Two adjacent beads represented as centroid-endpoint triples. E_{11}, E_{21}: Endpoints "1" of beads 1 and 2. E_{12}, E_{22}: Endpoints "2" of beads 1 and 2. C_1, C_2: Centroids of beads 1 and 2. P_1, P_2: Planes spanned by beads 1 and 2. P_{12}: Plane spanned by the triple $C_1{-}E_{12}{-}C_2$. α_1, α_2: Inner angles (fixed). ξ: Planar angle. ζ: Dihedral angle.

2.1 Data Model

Our input data set is a symmetric matrix where each entry is the number of times two specific loci are found in contact in a Hi-C experiment. A locus is a DNA segment whose genomic location and size depend on the restriction enzyme used in the experiment. The matrix entries can be the raw Hi-C counts, possibly transformed in contact probabilities, or can have been debiased through some specific normalization procedure [26], [27]. Most estimation algorithms proposed in the literature do not assume directly these data as their inputs: a popular approach (see, e.g., [11], [28]) is to transform the contact frequencies into required (or *equilibrium*, or *wish*) euclidean distances between the loci and then to estimate the structure by solving a distance-to-geometry problem. This is justified by the intuitive consideration that two frequently interacting loci are likely to be often close to each other, whereas loci that are seldom in contact are likely to be far apart in most configurations. By applying a number of possible frequency-to-distance relationships to real Hi-C data, we determined that the distances thus obtained are severely incompatible with the euclidean geometry [20]. Indeed, two loci that do not interact are not necessarily far from each other. For this reason (see Section 2.3), we do not translate contacts into distances.

2.2 Chromatin Model (Down to the Coarsest Scale)

We model the chromatin fiber as a bead chain. Each bead is not a simple spheroid, however, but a simplified model for a chromatin subchain that is supposed to exist at a finer scale. The spatial configuration of such a subchain is represented by its two endpoints and its centroid, and has an approximate size associated. The beads are bound to stay in their genomic order; the second endpoint of each bead coincides with the first endpoint of the successive; each bead is considered as a rigid structure (see Fig. 1), whose inner angle is not modified during the model evolution; conversely, the planar and dihedral angles formed by adjacent

1. In [23], the existence of a hierarchy of isolated domains in mammalian chromosomes is demonstrated. In [24], the genome-wide chromatin organization is reconstructed at the TAD level using both Hi-C data and the interactions of chromatin with the nuclear lamina.

2. https://www.python.org

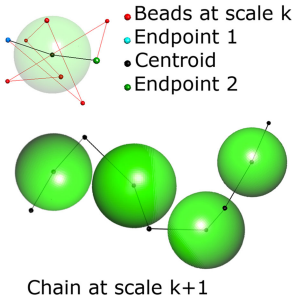

Fig. 2. Top: A chromatin subchain at a generic scale level (small red dots) and its centroid-endpoints representation (bigger dots) with the associated approximate size (large semi-transparent sphere). Bottom: A subchain at the immediately coarser scale level, made of four consecutive triples, appropriately located and 3D rotated. Note that the approximate bead sizes limit the flexibility of the chain (see Section 2.3).

beads are evolved in order to fit the data and the prior knowledge. At the finest scale available, a bead cannot be associated to any known finer structure, so it can only be modeled as a sphere, that is, as a particular triple with collinear and evenly spaced centroid and endpoints. The approximate bead size is determined as a fraction of the amplitude of the first principal component of the coordinates of the associated subchain, or, at the finest scale, from the genomic size of the associated locus and its internal interactions, represented by the corresponding diagonal element in the data matrix [25]. Associating approximate sizes to the beads allows us to avoid deep bead interpenetrations in the final solution and to get an output structure already equipped with physical dimensions. For details, see Section 3.5. As mentioned, the chromatin chain contains tightly packed subchains with a few interactions with the rest of the chain. These are mapped in a number of diagonal blocks in the contact frequency matrix. Thanks to their weak external interactions, we can detect those blocks and estimate the related structures at each scale independently of one another. Each estimated subchain then becomes a single bead in a coarser-scale chain and, binning the data matrix so that each detected block becomes a single diagonal element, the estimation can continue until no further diagonal blocks are detected. Fig. 2 exemplifies how four consecutive subchains can be modeled as modified beads and then connected to form a chain at a coarser scale.

2.3 Score Function

Following a Tikhonov-like approach to this inverse problem [29], we estimate the structure of a subchain by sampling a solution space generated by a score function Ξ. This includes a data fit term Φ, and a prior term Ψ, whose reciprocal influence is balanced by a regularization parameter λ. The two terms are sums of positive contributions, φ_{ij} and ψ_{ij}, intended to penalize unlikely interactions between the beads indexed by i and j. In formulas

$$\Xi(\mathcal{C}) = \Phi(\mathcal{C}) + \lambda\Psi(\mathcal{C}) = \sum_{(i,j)\in\mathcal{L}} \varphi_{ij} + \lambda \sum_{i\neq j\in\mathcal{C}} \psi_{ij}, \qquad (1)$$

where $\mathcal{C}$ is the chain configuration, that is, the list of all the bead positions and sizes in the current subchain, and $\mathcal{L}$ is a subset of the possible bead pairs. Thus formulated, the estimation problem is similar to a maximum a posteriori estimation of $\mathcal{C}$ maximizing the posterior distribution $\exp(-\Xi)$ built via the Bayes theorem from the likelihood $\exp(-\Phi)$ and the prior distribution $\exp(-\lambda\Psi)$. In Bayesian terms, the prior distribution is also a state of *prior knowledge*, since function Ψ is generally intended to model whatever we know on the solutions independently of the data.[3] Our choice of subset $\mathcal{L}$ depends on the fact that we do not want to use any target distance in the score function. For this reason, rather than transforming the contact frequencies into distances, we only assume that any two highly interacting beads are physically close to each other, whereas the weakly interacting beads are left free to assume any position and attitude compatible with the constraints derived from our prior knowledge. The difference between this approach and the ones that prescribe a large distance between weakly interacting beads is apparent. The details are in Section 3.3.

For the generic (i,j)th contribution to the negative log-likelihood Φ, we adopt a function that is proportional to the squared distance between the ith and the jth beads, assumed as the distance d_{ij} between their centroids minus the sum of their approximate radii, r_i and r_j

$$\varphi_{ij} = n_{ij}[d_{ij} - (r_i + r_j)]^2, \qquad (2)$$

where n_{ij} is the contact frequency between the ith and jth beads, and gives its specific weight to the (i,j)th pair, thus letting the most frequent contacts to predominate. We emphasize that this is not equivalent to require implicitly some target distance between the beads. At scales coarser than the finest available, each radius r_i is assumed as half the approximate size of the ith bead. Function Φ becomes small when the centroid-centroid distances of the pairs in $\mathcal{L}$ nearly equal the sums of the radii of the related beads. Large distances between pairs in $\mathcal{L}$ are quadratically penalized.

The negative log-prior Ψ penalizes interpenetrations between all the bead pairs in $\mathcal{C}$. Our proposed form for the individual contributions ψ_{ij} is

$$\psi_{ij} = \frac{r_i + r_j}{2d_{ij}}\left[1 - \frac{\{c[d_{ij} - (r_i + r_j)]\}^b}{1 + \{c|d_{ij} - (r_i + r_j)|\}^b}\right], \qquad (3)$$

where c is a scale factor that makes the terms in braces dimensionless, and b is an odd natural.

A plot of ψ_{ij} as a function of $d_{ij}/(r_i + r_j)$ is shown in Fig. 3. Note that: a) for d_{ij} near zero, ψ_{ij} behaves as $(r_i + r_j)/d_{ij}$; b) in an interval around $d_{ij} = (r_i + r_j)$, ψ_{ij} behaves as $(r_i + r_j)/(2d_{ij})$; c) for d_{ij} sufficiently larger than $(r_i + r_j)$, ψ_{ij} vanishes rapidly. Factor c tunes the width of the intermediate interval: large values of c make it narrow. Exponent b, in turn, tunes the slope of ψ_{ij} in the transition zones: as apparent from the figure, large values of b produce abrupt transitions. Function (3) is thus intended to prevent any two beads from interpenetrating more than some fraction of their sizes, but this is a *soft* constraint, in the sense that there is no unbreakable distance limit, and the final estimated values for each pair always depend on the mixed influence of data and prior knowledge, as resulting from the mutual interactions of *all* the possible pairs.

3. This view of inverse problems can be put in the appropriate framework by looking, e.g., at the general theory formulated in [30].

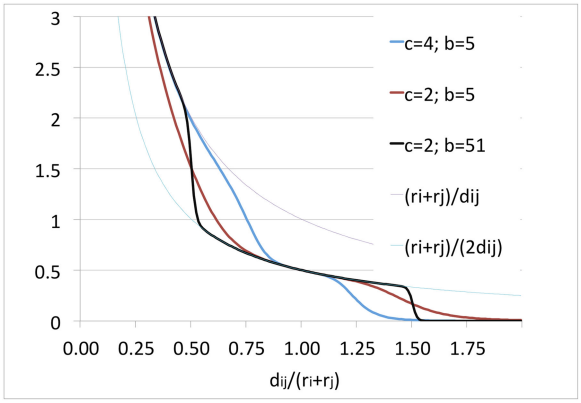

Fig. 3. (After [25]) Function ψ_{ij} in (3), plotted for a few values of c (in nm^{-1}) and b. The larger c, the narrower the moderate-slope interval around $d_{ij} = r_i + r_j$. The larger b, the steeper the slopes in the two transition regions.

The overall effect of $\Xi(\mathcal{C})$ is thus the one we desire for our solutions: each bead in $\mathcal{L}$ tends to be close to its paired beads, and each bead in $\mathcal{C}$ is either prevented from interpenetrating with other beads or left free to take any position compatible with the chain topology. Function Ψ translates biological prior knowledge into implicit geometrical constraints. The geometrical constraints for a subchain of the type in Fig. 2 can include a limiting distance between the centroids of any two beads and, to control the chain flexibility, limit values for the planar and dihedral angles. In this respect, however, function Ψ as defined here can also put effective constraints on angles through the approximated overall sizes of the coarse-scale beads. If any two genomically close substructures cannot be too close to each other, also the angles identified by the corresponding coarser-scale beads are constrained (see Fig. 2). An appropriate choice of λ, c and b can thus avoid the introduction of further constraints on angles. To choose λ, however, the standard strategies in, e.g., [29] and [31] cannot be followed. The details of our solution are in Section 3.5.

2.4 Solution Space Sampling

2.4.1 Simulated Annealing

The solution space generated by Ξ is sampled through an approximate simulated annealing [32], [33]: as explained in Section 3.5, a warm-up phase to determine the start temperature for each subchain treated is followed by a slow cooling to reach comparable final scores in different runs. Indeed, since many configurations are expected to fit the available data, the solution panorama is characterized by many nearly absolute optima, and each of them corresponds to a structure fitting both the data and the implicit constraints enforced by Ψ. Thus, each annealing run produces a different configuration. At each step, a quaternion operator is used to generate a random configuration. This is then accepted or rejected probabilistically on the basis of the differential score between the current and the proposed configurations divided by the current temperature. When the scores for a pool of consecutive accepted configurations fall within a given tolerance, the iteration is stopped.

2.4.2 Quaternions

At each annealing cycle, a single bead is moved at random; then, as the chain must remain connected, all the subsequent beads must be moved accordingly. This entails a number of rotations in the 3D space. To perform them, rather than the classical Euler angles, we use quaternions, as they avoid a number of problems and are also less expensive.[4] Further details can be found in [34].

A quaternion $\mathbf{q}$ is a generalized complex number with a real part and three imaginary parts. The sum, the product, the conjugate and the norm operators on quaternions are defined so that they form a central simple algebra of dimension 4 on the real field. A quaternion $\mathbf{v}$ with vanishing real part (a *pure* quaternion) can represent an $\mathbb{R}^3$ vector. Rotating $\mathbf{v}$ of an angle θ around a unit vector $\mathbf{u}$ only requires two quaternion multiplications

$$\mathbf{v} \rightarrow \mathbf{R_q}(\mathbf{v}) = \mathbf{q}\mathbf{v}\bar{\mathbf{q}}, \tag{4}$$

where $\mathbf{q}$ is this unit-norm quaternion

$$\mathbf{q} = cos\frac{\theta}{2} + sin\frac{\theta}{2}u_x i + sin\frac{\theta}{2}u_y j + sin\frac{\theta}{2}u_z k, \tag{5}$$

$\bar{\mathbf{q}}$ is its conjugate, u_x, u_y and u_z are the components of $\mathbf{u}$, and i, j, k are the three imaginary units defining a quaternion. Two successive rotations can be composed through the product of the corresponding quaternions: $\mathbf{R_p}(\mathbf{R_q}(\mathbf{v})) = \mathbf{R_{pq}}(\mathbf{v})$. We exploit these properties to rotate angles ξ and ζ (see Fig. 1) between any generic pair of adjacent beads, and all the subsequent beads in the chain, whose topology is maintained automatically.

2.5 Finest-Scale Chain Reconstruction

Once the chain at the coarsest scale is estimated, each bead has a spatial location and planar and dihedral angles with respect to the adjacent beads. These location and angles are conferred to the related subchain at the immediately finer scale, which can thus be used to build a new chain. This new chain, in turn, can be used to build a further chain, until no finer scale is available. Any such output is then added to the final set of solutions, useful to study the most likely chromatin configurations in the cells under test. In Section 3.6, we describe our iterative implementation of this procedure.

3 Code

In this section, we describe the nonstandard Python classes in ChromStruct 4.2, with their attributes and methods, and the functions we wrote to implement the iteration. The individual functions are the same implemented in the recursive versions of the code. What is specific of the iterative implementation is the main cycle, described by the following high-level pseudocode:

```
chromstruct(input_matrix):

1) initialize no_of_blocks (> 1);
2) initialize scale_level (= 0);
3) while no_of_blocks > 1, repeat
   a) extract blocks from input_matrix;
   b) update no_of_blocks;
   c) initialize lo-res_chain;
   d) for all the blocks, repeat
```

4. David Eberly, Geometric tools. http://www.geometrictools.com/Documentation/RotationIssues.pdf (last accessed: 2017, Feb. 16th).

i) populate set $\mathcal{L}$;
ii) set initial guess $\mathcal{C}_0$=`chain(block)`;
iii) estimate $\mathcal{C}_e$=`annealing(block,`$\mathcal{L}, \mathcal{C}_0$`)`;
iv) save $\mathcal{C}_e$;
v) compute the equivalent lo-res bead: $\texttt{bead}_\texttt{e} \leftarrow (\mathcal{C}_e)$;
vi) append $\texttt{bead}_\texttt{e}$ to `lo-res_chain`;
e) bin `input_matrix` to the new scale;
f) `scale_level` = `scale_level`+1;
4) `hi-res_chain` ← `lo-res_chain`;
5) while `scale_level` > 0, repeat
a) `lo-res_chain` ← `hi-res_chain`;
b) initialize `hi-res_chain`;
c) for all the `beads` in `lo-res_chain`, repeat
i) `subch` $\leftarrow \mathcal{C}_e(\texttt{bead}, \texttt{scale\_level} - 1)$;
ii) rotate `subch` as `bead`;
iii) append `subch` to `hi-res_chain`;
d) `scale_level` = `scale_level`−1;
6) output $\mathcal{C} =$ `hi-res_chain`;

At each scale, steps 3-d) and 5-c) can run in parallel rather than sequentially. Besides the partition of the full-resolution chain, which makes the annealing much faster, this introduces levels of potential parallelism in the entire procedure, which can become less time demanding. The iterative implementation also provides advantages over the recursive implementation. Indeed, the recursive code ChromStruct 3.2 [25] needs to hold all the intermediate-level results in the computer stack until the final estimation is complete, and is not able reconstruct the fine-scale chain in parallel (step 5-c in the above pseudocode). Version 4.2 is thus less memory demanding and more parallelizable than version 3.2. For all the rest, when run on sequential hardware, the two versions require roughly the same computing time, and the final results are perfectly equivalent.

In the following, we describe the functionalities of our code. The finest details can be appreciated by looking at the commented source code and the usage notes. References to names and functions specific to the code are included here to guide the interested reader.

3.1 Special Classes

Two custom Python classes have been defined to help programming:

3.1.1 Class `bead`

The class `bead` has been defined to include all the features of the modified beads in our chromatin model, and to allow them to be suitably positioned and rotated in the 3D space. This class has four attributes: three of them are 3D vectors containing the coordinates of the centroid and the endpoints of the bead; the fourth attribute, a real scalar, is the approximate size. Four special methods have been defined for this class: `innerangle` computes the angle formed by the segments joining the centroid with the two endpoints; `d1d2` computes the distances between the centroid and the two endpoints; `shiftbd` moves the bead rigidly of a specified 3D displacement; `zerobead` places endpoint 1 in $(0, 0, 0)$.

3.1.2 Class `chain`

The class `chain` has only one attribute, consisting in a list of `bead` objects. Its specific methods are `chainadd`, which appends a bead to an existing chain; `chainmerge`, which appends an already existing chain to the current chain; `formchain`, which places the beads so to form a connected chain; `zerochain`, which places endpoint 1 of bead 1 in $(0, 0, 0)$; `shiftchain`, which shifts the chain rigidly in a specified location; `quaplanar`, which applies a quaternion operator to rotate a bead of a specified angle within the bead plane; `quadihedral`, which rotates the dihedral angle ζ of a specified bead pair; `perturb`, which produces a random perturbation of a specified bead in the chain, followed by a rigid rotation of all the subsequent beads.

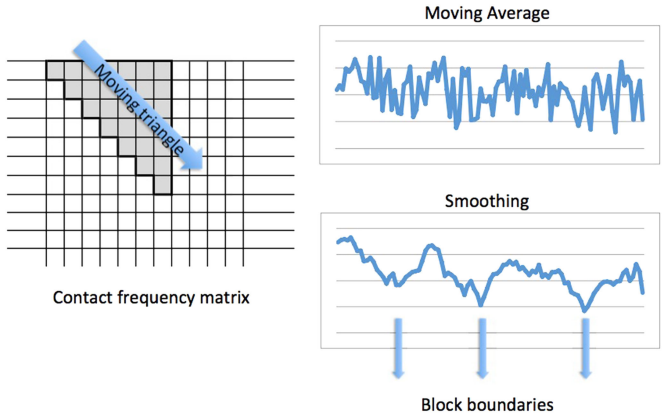

Fig. 4. Extraction of diagonal blocks from the contact frequency matrix. As the triangle on the left slides along the main diagonal of the contact matrix, the moving average function is recorded, and then smoothed. The relative minima of the smoothed function, except the ones that identify too small blocks, are assumed as block boundaries.

3.2 Partitioning the Chromatin Chain

3.2.1 Function `centr`

As mentioned, our bead-chain model partitions the chromatin in weakly-interacting domains. This cannot be made effectively when the data matrix contains large bands with missing data, as happens when centromeres and telomeres are in the genomic range considered. These chromosomic segments, indeed, are characterized by many repeated sequences, which cannot be located univocally in the DNA chain, thus producing a "hole" in the contact matrix, often replaced by a zero. This phenomenon is recognized by function `centr` (out of the above pseudocode). If large diagonal blocks with vanishing entries contain the first or the last rows of the matrix, they are classified as telomeres; if such blocks are detected in intermediate bands, they are classified as centromeres. The telomere/centromere locations are passed to functions `block` (Section 3.2.2), `inizchain` (Section 3.4) and `skipannealing`, (Section 3.5).

3.2.2 Function `block`

Function `block` identifies the nearly isolated diagonal blocks in the contact matrix. Before starting with this procedure, however, it must take into account the possible presence of centromeres and telomeres. When this is the case (input from function `centr`), these regions are taken as unique blocks, even if their size is very large. The other blocks are identified through the relative minima of the smoothed moving average over suitably-sized triangular subsets sliding along the diagonal of the data matrix (see Fig. 4) [16]. The result is a list

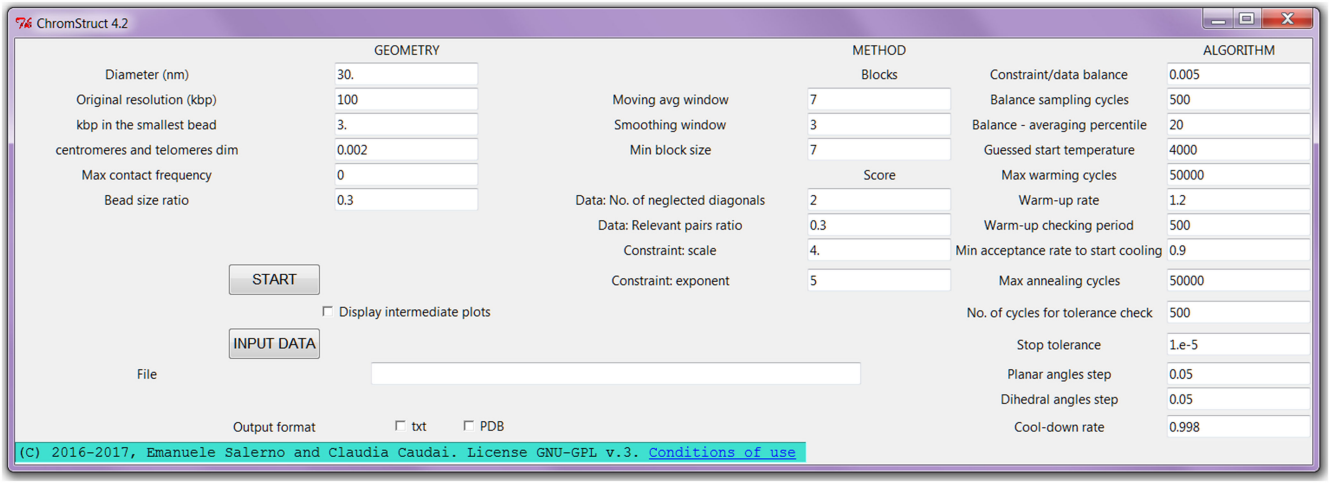

Fig. 5. ChromStruct 4.2 Graphical User Interface. All the parameters are set at their default values.

of extracted diagonal blocks, a list of block sizes and a list of block boundaries. The other inputs to `block` are the minimum required block size (smaller blocks are merged together), the size of the moving triangle, and the width of the smoothing window. Once the blocks are extracted, each of them is associated with a `chain` object at the appropriate resolution, that is, with the appropriate bead sizes. This part of the code implements steps 3-a) to 3-c) of the above pseudocode.

3.3 Selecting the Relevant Pairs

To populate $\mathcal{L}$ for each subchain thus generated, function `populate` (step 3-d-i in the pseudocode) selects the pairs exceeding a pre-defined percentile of the contact frequencies in the block they belong. This strategy can be advantageous in presence of biased data [2], since the most detrimental biases are those that affect the smallest contact frequencies. Genomically close, or even adjacent, beads are always also physically close. Their interactions are mapped in a few subdiagonals of the data matrix, and are excluded from the set $\mathcal{L}$, as the related beads are kept close to each other by the links established by the chain model. The number of subdiagonals to be excluded can be tuned through a specific parameter (see Fig. 5) whose value must not be smaller than 2, since each bead is necessarily close to itself and its immediate neighbors. At least the pairs in the main diagonal and the first subdiagonal must thus be left out of $\mathcal{L}$.

3.4 Evaluating Bead Sizes

We compute the approximate bead sizes at the finest scale by observing, as done in [9], that a 30-nm-long DNA fragment has a genomic length of about 3 kbp. A bead at any resolution, then, can be thought of as a sequence of 30-nm-diameter fragments; we use the number of such fragments in a resolution unit to determine the sizes of our beads. As far as the maximum bead size is concerned, we assume that this is reached when all the fragments are arranged side by side in a 2D array. The maximum size will then be proportional to the square root of that number. By trial and error, we found that π is an appropriate coefficient of proportionality. The minimum size, in turn, is assumed to be the diameter of a cube where all the fragments in the resolution unit are tightly packed, that is $\sqrt[3]{\frac{\text{resolution unit}}{3\text{ kbp}}} \cdot \sqrt{3} \cdot 30\text{ nm}$, where the fraction under cubic root is the number of 3-kbp fragments in the resolution-unit bead. Having evaluated the maximum and minimum bead extensions, we then use the number of internal contacts, that is, the corresponding diagonal entry in the contact matrix, to estimate the size of each individual bead. Indeed, many internal contacts mean a tightly packed fragment array and, conversely, a few internal contacts mean a loosely packed array. Thus, we assign the minimum size to the beads corresponding to the largest diagonal value in the original Hi-C matrix, then find the sizes of the other beads by linear interpolation between the maximum and the minimum sizes. This is done at the start of each run, by the functions called `constr` and `inizchain`, outside the `chromstruct` block described by the pseudocode. Function `inizchain` also reads the boundaries of centromere and telomere regions and computes the sizes of the related beads in a special way, through function `centrsize`, as the original resolution unit multiplied by a factor `crate`, to which we assigned a default value of 0.002 nm/kbp.

At coarser scales, each bead corresponds to a subchain estimated in cycle 3) of the pseudocode, so the distances of the endpoints from the centroid and the inner angle are already available. As mentioned in Section 2.2, the approximate bead size at these scales is set as a fraction of the amplitude of the first principal component of the coordinates of the associated subchain. To motivate this choice, we observe that the square root of the maximum eigenvalue of the covariance matrix of a 3D point set is proportional to its extent along its first principal component. In other words, if all the coordinates are uniformly scaled, the covariance eigenvalues are scaled by the square of the scale factor. The appropriate fraction to be used is given to ChromStruct 4.2 as an input parameter called `extrate`. The real-world subchain is far from being an impenetrable spheroid: our approximate size is only intended to prevent the chain from being too packed, and this can be obtained by choosing `extrate` by trial and error. This size-estimation procedure implements step 3-d-v) in the pseudocode.

3.5 Approximated Simulated Annealing

Our sampling of the solution space is not intended to find a unique absolute optimizer of the score function.

As mentioned, indeed, the Hi-C data result from aggregated counts on millions of cells, where many different configurations can contribute. The presence of a unique absolute minimum in the score function is thus unlikely. On the other hand, a rich set of different solutions enables the user to examine the possible chromatin configurations during the cell lifecycle, and to build synthetic contact matrices to be compared to the actual input data [25]. This is why we do not try to minimize the score function but rely on random sampling to extract significant representatives of the solution space.

Rather than a pure Monte Carlo, however, we chose to seek for configurations having similar scores, in order to ensure that all our solutions comply equally with the data and the constraints. The strategy we propose is a modified version of simulated annealing [33]. We include here just a summary of this well-known sampling technique. A set $\{\mathcal{C}_k\}$ of random configurations is generated through the following procedure:

- Let $\mathcal{C}_n$ be the current configuration
- Randomly generate $\mathcal{C}_*$ by a small perturbation of $\mathcal{C}_n$
- Set $\mathcal{C}_{n+1} = \mathcal{C}_*$ with probability

$$p = \begin{cases} \exp\left[-\frac{\Xi(\mathcal{C}_*)-\Xi(\mathcal{C}_n)}{T}\right] & \text{if } \Xi(\mathcal{C}_*) > \Xi(\mathcal{C}_n) \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

If the *temperature* parameter T is slowly decreased to zero from a sufficiently high start value, the elements of $\{\mathcal{C}_k\}$ will be distributed uniformly over the set of the absolute minimizers of $\Xi(\mathcal{C})$. Theoretical rules to find an absolute minimizer by this strategy can be found in [32]. What is important in our context is to set a start temperature that ensures a complete coverage of the solution space, and then a sufficiently slow cooling scheme to approach a solution for which Ξ is close to its minimum. In our case, we must find an appropriate start temperature for each different block and scale treated. We start from a fixed value and increase it until nearly all the configurations $\mathcal{C}_*$ are accepted, then the slow cooling is started. When the scores of a number of consecutive accepted configurations fall within a given tolerance, the iteration stops and the current configuration, $\mathcal{C}_e$, is taken as our estimated subchain. In our approximated approach, the temperature is lowered at each step.

Another specificity is that λ in (1) must also be set separately for each block and scale. Parameter λ resembles the usual regularization coefficient used in the Tikhonov-type solution to inverse problems [29], but its value cannot be established with the criteria based on the residual (e.g. [31]), since no residual can be computed from the Hi-C data in our case. What we do here is just to establish a (statistically) fixed ratio between the terms Φ and $\lambda\Psi$ in (1), and compute λ accordingly from the average Φ and Ψ values obtained from a set of random chain configurations.

This procedure is represented in step 3-d-iii) in the pseudocode, and implemented in ChromStruct 4.2 by function `annealing`, along with functions `cool` and `warm`, setting the right temperature at each iteration, `checktest` and `checkstop`, checking the acceptance and stop criteria, and `fitcomp`, `constrcomp` and `energy`, computing the score function.

As mentioned, we use quaternions to rotate the generic nth bead in the plane identified by its centroid-to-endpoint-1 segment and the centroid-to-endpoint-2 segment of bead $n-1$. In Fig. 1, let us assume that the bead $n-1$ is on the left and bead n on the right. To alter the planar angle ξ of a quantity θ, we let

$$\mathbf{v} = \overrightarrow{\mathrm{E}_{21}\mathrm{C}_2}, \tag{7}$$

and

$$\mathbf{u} = \frac{\overrightarrow{\mathrm{E}_{12}\mathrm{C}_1} \times \overrightarrow{\mathrm{E}_{21}\mathrm{C}_2}}{||\overrightarrow{\mathrm{E}_{12}\mathrm{C}_1} \times \overrightarrow{\mathrm{E}_{21}\mathrm{C}_2}||}. \tag{8}$$

Then, we evaluate $\mathbf{q}$ from Eq. (5) and apply map (4). Vector $\mathbf{v}$ has thus been rotated of a quantity θ around the unit vector $\mathbf{u}$. Repeating this procedure for all the subsequent beads rotates the chain rigidly from the nth to the last bead. The same mapping is used to modify the dihedral angle ζ between planes P_1 and P_{12}, letting

$$\mathbf{u} = \frac{\overrightarrow{\mathrm{E}_{12}\mathrm{C}_1}}{||\overrightarrow{\mathrm{E}_{12}\mathrm{C}_1}||}, \tag{9}$$

and repeating the procedure for all the subsequent pairs of adjacent beads. These rotations are performed through the methods in class `chain` (see Section 3.1.2), also exploiting the methods available in class `Quaternion`.[5]

Annealing is not applied to the telomere and centromere blocks. Where their presence is detected by function `centr`, function `skipannealing` just returns a copy of the initial configuration of these subchains established by function `centrsize`.

3.6 Recomposing the Whole Chain

When the chain at the coarsest scale has been reconstructed, the endpoints and the centroids of all the beads are placed in the 3D space so as all the planar and dihedral angles are fixed. Each bead corresponds to an already available subchain at the immediately finer scale, which can be aligned in accordance with the proper planar and dihedral angles. When this is done for all the beads, a chain at a finer scale is produced, whose beads can be substituted by the corresponding subchains at a further finer scale, if any. The entire structure at the finest scale (the one of the original data) can thus be reconstructed iteratively. This simple procedure is represented by step 5) in the pseudocode, and implemented in ChromStruct 4.2 by function `compose`.

3.7 Input-Output

3.7.1 Graphical User Interface

Fig. 5 shows the GUI provided with this version of the code. The GUI window contains all the working parameters, subdivided in three groups. At start, each field is filled with its default value. To allow for maximum flexibility, all the parameters can be modified by the user directly from the GUI, but just a few of them need a particular attention. In the GEOMETRY group, six parameters are provided:

5. https://pypi.python.org/pypi/Quaternion/ (last accessed: 2017, May 10th).

- Assumed diameter of the chromatin fiber (as in [9], we assume a default of 30 nm, even though the existence of such a packing level is questionable [35]);
- Genomic resolution of the data matrix, in kbp;
- Genomic length (kbp) of a DNA fragment as long as the assumed fiber diameter;
- Centromeres and telomeres dim, intended as the coefficient `crate` explained in Section 3.4;
- Maximum contact frequency: when set to zero, it is computed from the data matrix; user-defined values can be adopted to tune the assumed sizes for the finest-scale beads (see Section 2.2);
- Bead-size ratio (see Section 2.2), which is used to determine the assumed size for the successive scale levels. Along with "Constraint/data balance" under group ALGORITHM (used to set the λ values), this parameter influences the compactness of the final solution. If it is chosen appropriately, the final chain size will be compatible with the size of the nucleus.

Groups METHOD and ALGORITHM are used to set up the estimation method and the simulated annealing procedure (Sections 2.3 and 3.5). Modifying their defaults influences the method performance, but is not essential in most cases. The usage notes attached here give all the details needed for a knowledgeable management of these parameters.

A further field is provided to choose the file with the input data; alternatively, the choice can be made from the file system through the INPUT DATA button. The three checkboxes are used to set output preferences (see Section 3.7.3 below).

3.7.2 Input

The data file (ASCII-coded) must contain the contact frequency matrix, stored as a general matrix in tab-delimited format, with the end of each row marked by a newline character. This is a quite redundant method to code the data, and can be cumbersome when dealing with very large data sets. An easy and short addition to the code can enable more storage-efficient formats to be accepted.

3.7.3 Output

Any single run, activated by the START button in the GUI, produces an output stream in the console window, containing the relevant information for all the blocks processed and all the scale levels. This output is progressively stored in a log file, which is closed when the final configuration is plotted and stored. Optionally (by the "Display intermediate plots" checkbox), a 3D plot of each computed subchain and plots of $\Xi(\mathcal{C}_n)$, $\Phi(\mathcal{C}_n)$ and $\lambda\Psi(\mathcal{C}_n)$ can be obtained.

Also, for each block and scale, as well as as for the final result, the 3D configuration, the score sequence and the euclidean distance matrix are stored in text files. The file naming is based on the name of the input file, with a start-time stamp that identifies uniquely the particular run, the identifiers of the related subchains and scales, and the type of data stored. The 3D configuration text files list the endpoints and the centroids of all the beads. Each bead is represented by a 3×4 real array: its first row identifies endpoint 1, the second row identifies the centroid, and the third row identifies endpoint 2. In each row, the first three columns contain the x, y, z coordinates, and the fourth contains the estimated radius of the corresponding bead. The latter will of course be the same in all the three rows. The coordinates of endpoint 2 of each bead coincide with the coordinates of endpoint 1 of the subsequent bead. Again, this is not the most efficient way to store the results, but has the advantage of being easily read to exploit intermediate results for subsequent computing with the same chromatin model. Here, we also provide a separate code to visualize the results stored in these files by using the graphical capabilities of the Python language.

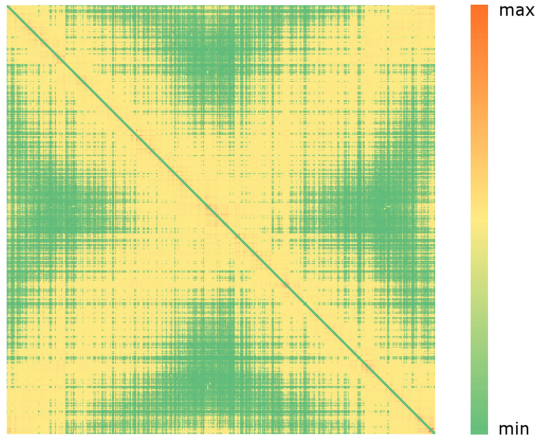

Fig. 6. Hi-C contact frequency matrix for the *C. Crescentus* chromosome (main diagonal suppressed for readability). The highest frequencies form a "cross" pattern, as the first and the last base pairs are connected.

Optionally, the final 3D configuration can also be output in Protein Data Bank (PDB) format,[6] which can be used to visualize DNA structures through several standard 3D viewers, such as Jmol[7] and Swiss-PdbViewer.[8] Our PDB file only contains the centroid coordinates in nm, scaled by a factor of 0.01 for viewing convenience, and the links between consecutive beads.

4 Some Experimental Results

In previous papers [19], [25], we validated our method against Hi-C data from the human genome at 100 kbp resolution. Although our main purpose here is to describe how we implemented the functionalities of ChromStruct, we present some results as examples obtained with different data sets and different genomic resolutions. In this section, we deal with the annular *Caulobacter Crescentus* chromosome [36]. In the appendices provided as supplemental material, available online, we treat other data sets and resolutions and introduce some comparison between our results and the corresponding ones obtained through other methods proposed in the literature.

Fig. 6 shows the 405×405 raw contact frequency matrix from a Hi-C experiment on *C. Crescentus* cells, represented as a heat map, with a genomic resolution of 10 kbp.[9] Its typical x-shape is characteristic of the annular structure of the chromosome: the first locus is always connected to the last one, so the matrix entries close to $(1, 405)$ and $(405, 1)$ are

6. http://www.rcsb.org/pdb/static.do?p=file_formats/pdb/index.html
7. http://jmol.sourceforge.net
8. http://spdbv.vital-it.ch
9. http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE45966

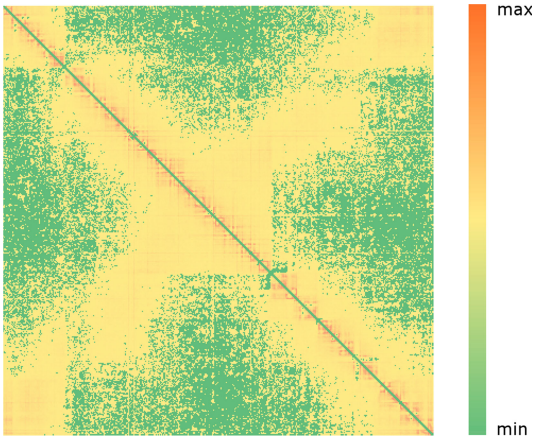

Fig. 7. Synthetic Hi-C contact frequency matrix from 1,000 ChromStruct 4.1 estimated configurations.

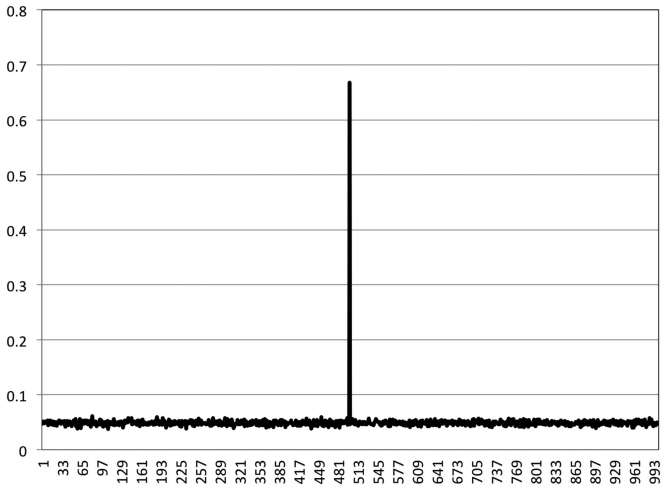

Fig. 8. Spearman correlation between the original data, the reconstructed data in Fig. 7 (at position 500) and 1,000 randomly generated matrices.

large. From these data, our code identifies either two or three scale levels, by setting (*Moving avg window, Smoothing window, Min block size*) at $(7, 3, 7)$ and $(6, 3, 4)$, respectively. In the former case, the code finds 32 blocks made of 7 to 31 beads; in the latter case, we first have 40 blocks made of 4 to 21 beads, and a further scale level with 4 blocks made, respectively, of 5, 11, 11 and 13 beads.

To check the consistency of our results with the data, we use the synthetic Hi-C matrix obtained by coadding the individual contact matrices from 1,000 different solutions. For each configuration, all the possible bead-to-bead distances are computed, and two beads are considered in contact when their distance is not larger than the maximum value significantly penalized by function Ψ. In our case, see Fig. 3, the values $c = 4, b = 5$ were chosen throughout, corresponding to a threshold distance of 1.2 times the sum of the bead radii. The result, cumulated over all the solutions, is shown in Fig. 7. Note that the original cross pattern is reproduced fairly well, along with many finer details. To have a quantitative index of how well our result fit the data, we computed the Spearman correlation between the matrices in Figs. 6 and 7, and compared the result to the correlations between the original data and 1,000 matrices of the same size, generated at random, but symmetric and with the true mean values in all the subdiagonals. The results are collected in Fig. 8: the Spearman coefficient for our reconstructed matrix is at least one order of magnitude larger than the ones obtained with the random matrices. Note that the main diagonal of the synthetic matrix counts the contacts between each bead and itself, and the first subdiagonals count the contacts between each bead and the immediately adjacent beads. This means that their values are always equal to the number of configurations generated and a comparison with the original data has no meaning. For this reason, we did not include these data in computing the Spearman coefficients. Table 1 breaks down the results obtained by different parameter settings (unmentioned parameters have been kept at their defaults shown in Fig. 5). We performed 8 series of experiments, changing some input parameters to have a sufficient variety of results, and retained all the solutions for our final population, since their Spearman correlations had always the same order of magnitude, and their value for the cumulated results was higher than the ones obtained from any separate series.

The bottom row in Table 1 reports the average computing times per configuration per core for all the series of

TABLE 1
Experimental Results Breakdown

		Series 1	Series 2	Series 3	Series 4	Series 5	Series 6	Series 7	Series 8	All results
Parameters	Mov. avg window	7	7	6	7	7	7	6	7	
	Min block size	7	7	4	7	7	7	4	5	
	Constr./data bal.	0.005	0.1	0.004	0.004	0.001	0.0001	0.001	0.001	
	Bal. sampl. cycles	500	500	500	500	200	200	200	200	
	No. cycles tol. check	500	500	200	500	500	500	200	200	
	Stop tolerance	1.00E-02	1.00E-01	1.00E-01	1.00E-02	1.00E-01	1.00E-01	1.00E-01	1.00E-01	
Output	Scale levels	2	2	3	2	2	2	3	2	
	No. of config.	200	100	100	100	100	100	100	200	1000
	Spearman corr.	0.635	0.224	0.288	0.267	0.332	0.474	0.315	0.416	0.668
	$\bar{\Xi}$	4.6E+08	6.7E+08	6.6E+07	4.3E+08	2.7E+08	6.7E+07	6.4E+07	3.4E+08	3.2E+08
	σ_Ξ	9.2E+07	1.7E+08	2.0E+08	8.5E+07	8.6E+07	2.2E+07	1.8E+08	8.1E+07	2.3E+08
	$\frac{\sigma_\Xi}{\bar{\Xi}}$ (%)	20.23	25.45	309.38	19.72	31.49	33.04	275.15	24.04	71.80
CPU time ($\times 10^3$ secs)	(avg per config.)	26.3	22.8	10.6	29.1	23.6	23.2	12.9	18.3	20.9

$\bar{\Xi}$ = *mean output score;* σ_Ξ = *output score standard deviation. See Fig. 5 for the unmentioned parameters.*

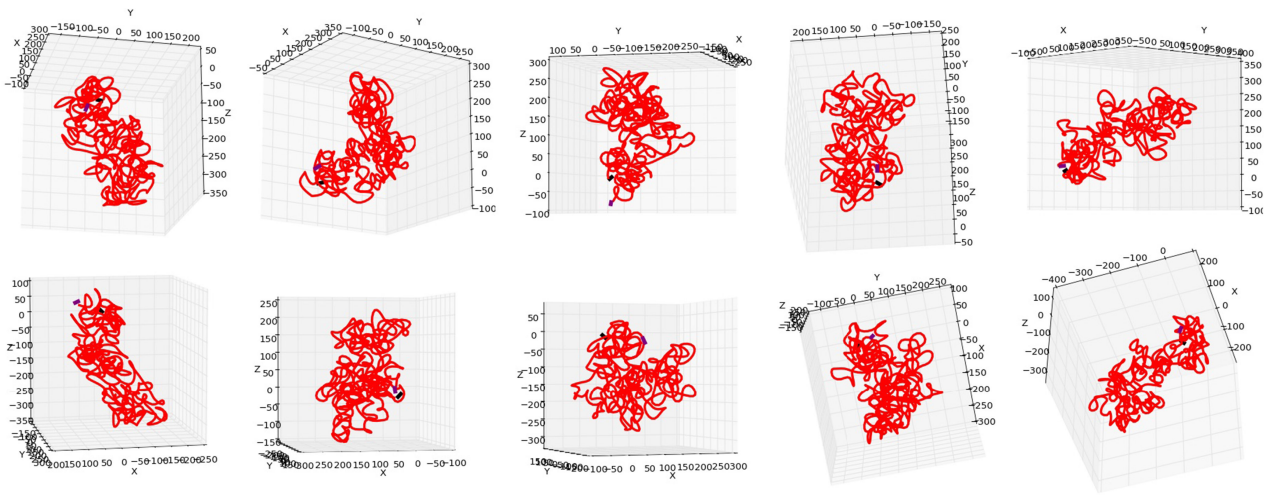

Fig. 9. Ten different output structures for the *C. Crescentus* chromosome (measurements in nanometers). The first and last beads (highlighted) are always found very close to each other.

experiments, obtained by running our sequential code on a processor 2X Intel(R) Xeon(R) 40 core, 16Mb GPU, 128 Gb RAM. Note that working with three scale levels normally needs much less time to run than working with two scale levels. This is due to the smaller sizes of the blocks extracted, which lowers considerably the computations required by simulated annealing. Nevertheless, the final scores obtained with three levels are much more dispersed around their mean values $\bar{\Xi}$.

Since we did not introduce any information about the annular structure of our chromosome, the first and last beads of our chain are free to assume any position allowed for by the score function. We take this as a way to check the effectiveness of Ξ to generate the solution space. If the two extreme beads are always found to be close to each other, then the annular structure of this chromosome is captured by our method without enforcing it externally. Of course, they can never be found in true contact by effect of function Ψ. However, Fig. 9 shows some example solutions where the two beads are very close to each other. To appreciate how this property is common to all the configurations, we computed the distances between the centroids of all the first and last beads. The overall mean distance between the centroids is less than 89 nm, that is, 2.54 average-sized beads at the finest scale. The mean distances over Series 3 and 7 in Table 1 are, respectively, 123 and 119 nm. If we exclude these series from the computation, we obtain an average distance of 79.6 nm over the remaining 800 configurations: 2.3 times the average bead size. Note that Series 3 and 7 are the ones whose parameters produce smaller blocks and 3 scale levels. We already observed that the results in these cases are much more dispersed than the others. From the average distance between the first and last beads in the configurations reconstructed, we can also say that they reproduce less accurately one of the known properties of the chromosome under study. Too small blocks thus affect negatively the result. This depends on the parameter "Min block size", Fig. 5, whose value should be tuned on the basis of a careful examination of the results. With all the data we treated so far, a value of 5 to 7 has always given results that fit well the data.

5 Conclusion

A method to estimate the 3D chromatin configurations of a cell population from Hi-C data is presented, and a Python code implementing it is described in detail. We already proposed and motivated the data model and the general strategy in [19], whereas the score function, the sampling strategy and the implementation details have been progressively refined during the study, leading us to the first complete recursive implementation [25]. In this paper, for the first time, we give a detailed description of our Python implementation, in its iterative version, and provide an extensively commented stand-alone source code. Implementing iteratively a method that would "naturally" appear recursive does not imply a significant improvement in the performance with sequential hardware, but offers some additional levels of parallelism for possible implementations in general-purpose or dedicated parallel architectures. The code features a convenient GUI that allows the user to modify all the working parameters without altering the source, and a standard output format for the final 3D configurations.

The example solutions presented in the main text and in the appendices, available in the online supplemental material, show that this method is capable of fitting well the data, allows for treating entire chromosomes, including the centromere and telomere regions, and is able to capture features that are not explicitly enforced as constraints. As expected, since the data-fit part of our score function is only based on the most reliable contact frequencies (see Section 3.3), our method is not much sensitive to biases. In Appendix A.2, available in the online supplemental material, this is demonstrated experimentally by comparing some of the results obtained from raw and debiased data.

Acknowledgments

This work has been partially supported by the Italian Flagship Project InterOmics, WP01-ISTI, and by ISTI-CNR, through scientific agreement 4249-29/09/2017 with ITB-CNR, Milan. The authors thank Luciano Milanesi and Clelia Peano for their helpful discussions.

REFERENCES

[1] J. R. Dixon, et al. "Topological domains in mammalian genomes identified by analysis of chromatin interactions," *Nature*, vol. 485, pp. 376–380, 2012.

[2] M. Imakaev, et al., "Modeling chromosomes: Beyond pretty pictures," *FEBS Lett.*, vol. 589, pp. 3031–3036, 2015.

[3] N. L. van Berkum, et al., "Hi-C: A method to study the three-dimensional architecture of genomes," *J. Vis. Exp.*, vol. 39, pp. 1869–1875, 2010.

[4] E. Sefer, et al., "Deconvolution of ensemble chromatin interaction data reveals the latent mixing structures in cell subpopulations" *J. Comput. Biol.*, vol. 23, pp. 425–438, 2016.

[5] M. R. Segal and H. L. Bengtsson, "Reconstruction of 3D genome architecture via a two-stage algorithm," *BMC Bioinf.*, vol. 16, 2015, Art. no. 373.

[6] T. Trieu and J. Cheng, "3D genome structure modeling by Lorentzian objective function," *Nucleic Acid Res.*, vol. 45, pp. 1049–1058, 2017.

[7] J. Fraser, et al., "Chromatin conformation signatures of cellular differentiation," *Genome Biol.*, vol. 10, 2009, Art. no. R37.

[8] Z. Duan, et al., "A three-dimensional model of the yeast genome," *Nature*, vol. 465, pp. 363–367, 2010.

[9] M. Di Stefano, et al., "Hi-C-constrained physical models of human chromosomes recover functionally-related properties of genome organization," *Sci. Rep.*, vol. 6, 2016, Art. no. 35985.

[10] N. Tokuda, et al., "Dynamical modeling of three-dimensional genome organization in interphase budding yeast," *Biophys. J.*, vol. 102, pp. 296–304, 2012.

[11] J. Dekker, et al., "Capturing chromosome conformation," *Sci.*, vol. 295, pp. 1306–1311, 2002.

[12] S. Wang, et al., "Inferential modeling of 3D chromatin structure," *Nucleic Acids Res.*, vol. 43, 2015, Art. no. e54.

[13] M. Hu, et al., "Bayesian inference of spatial organizations of chromosomes," *PLOS Comput. Biol.*, vol. 9, 2013, Art. no. e1002893.

[14] M. Sekelja, et al., "4D nucleomes in single cells: What can computational modeling reveal about spatial chromatin conformation?" *Genome Biol.*, vol. 17, pp. 54–63, 2016.

[15] Y. Shavit "How computer science can help in understanding the 3D genome architecture," *Briefings Bioinf.*, vol. 17, pp. 733–744, 2016.

[16] B. R. Lajoie, et al., "The Hitchhiker's guide ti Hi-C analysis: Practical guidelines," *Methods*, vol. 72, pp. 65–75, 2015.

[17] M. Forcato, et al., "Comparison of computational methods for Hi-C data analysis," *Nature Methods*, vol. 14, pp. 679–689, 2017.

[18] G. G. Yardimci and W. S. Noble, "Software tools for visualizing Hi-C data," *Genome Biol.*, vol. 18, pp. 26–34, 2017.

[19] C. Caudai "Inferring 3D chromatin structure using a multiscale approach based on quaternions," *BMC Bioinf.*, vol. 16, pp. 234–244, 2015.

[20] C. Caudai, et al., "A statistical approach to infer 3D chromatin structure," in *Mathematical Models in Biology*, V. Zazzu, et al., Eds. Switzerland: Springer, 2015, pp. 161–171.

[21] G. Duggal, et al., "Resolving spatial inconsistencies in chromosome conformation measurements," *Algorithms Mol. Biol.*, vol. 8, 2013, Art. no. 8.

[22] A. D. Schmitt, et al., "Genome-wide mapping and analysis of chromosome architecture," *Nature Rev.*, vol. 17, pp. 743–755, 2016.

[23] Y. Zhan, et al., "Reciprocal insulation analysis of Hi-C data shows that TADs represent a functionally but not structurally privileged scale in the hierarchical folding of chromosomes," *Genome Res.*, vol. 27, pp. 479–490, 2017.

[24] J. Paulsen, et al., "Chrom3D: Three-dimensional genome modeling from Hi-C and nuclear lamin-genome contacts" *Genome Biol.*, vol. 18, 2017, Art. no. 21.

[25] C. Caudai, E. Salerno, M. Zoppè, and A. Tonazzini, "Estimation of the spatial chromatin structure based on a multiresolution bead-chain model," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 16, no. 2, pp. 550–559, Mar./Apr. 2019.

[26] M. Imakaev, et al., "Iterative correction of Hi-C data reveals hallmarks of chromosome organization," *Nature Methods*, vol. 9, pp. 999–1003, 2012.

[27] E. Yaffe and A. Tanay, "Probabilistic modeling of Hi-C contact maps eliminates systematic biases to characterize global chromosomal architecture," *Nature Genetics*, vol. 43, pp. 1059–1067, 2011.

[28] B. Adhikari, et al., "Chromosome3D: Reconstructing three-dimensional chromosomal structures from Hi-C interaction frequency data using distance geometry simulated annealing," *BMC Genomics*, vol. 17, pp. 886–894, 2016.

[29] A. N. Tikhonov and V. Y. Arsenin, *Solution of Ill-Posed Problems*, Washington, DC, USA: Winston-Wiley, 1977.

[30] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2005.

[31] G. Wahba, "Practical approximate solutions to linear operator equations when the data are noisy," *SIAM J. Numerical Anal.*, vol. 14, pp. 651–667, 1977.

[32] E. Aarts and J. Korst, *Simulated Annealing and Boltzman Machines*, New York, NY, USA: Wiley, 1989.

[33] S. Kirkpatrick, et al., "Optimization by simulated annealing," *Sci.*, vol. 229, pp. 671–680, 1983.

[34] J. A. Vince, *Geometric Algebra for Computer Graphics*, Berlin, Germany: Springer, 2008.

[35] J. Hansen, "Human mitotic chromosome structure: What happened to the 30-nm fibre?" *EMBO J.*, vol. 31, pp. 1621–1623, 2012.

[36] T. B. K. Le, et al., "High-resolution mapping of the spatial organization of a bacterial chromosome," *Sci.*, vol. 342, pp. 731–734, 2013.

[37] J. Dostie and J. Dekker, "Mapping networks of physical interactions between genomic elements using 5C technology," *Nature Protocols*, vol. 2, pp. 988–1002, 2007.

[38] S. S. P. Rao, et al., "A 3D Map of the human genome at kilobase resolution reveals principles of chromatin looping," *Cell*, vol. 159, pp. 1665–1680, 2014.

[39] E. Lieberman-Aiden, et al., "Comprehensive mapping of long-range interactions reveals folding principles of the human genome," *Sci.*, vol. 326, pp. 289–293, 2009.

[40] D. Baù and M. A. Marti-Renom, "Genome structure determination via 3C-based data integration by the integrative modeling platform," *Methods*, vol. 58, pp. 300–306, 2012.

[41] L. Rieber and S. Mahony, "MiniMDS: 3D structural inference from high-resolution Hi-C data," *Bioinf.*, vol. 33, pp. 261–266, 2017.

Claudia Caudai received the MS degree in mathematics and the PhD degree in biomedical engineering, both from Pisa University, in 2003 and 2009, respectively. She has been working on modeling of ECG signals and neural and neuroglial messages in human brain, and on the evaluation of compliance and stiffness of biological and artificial muscles. From 2008 to 2011, she was with the Scientific Visualization Unit with the Institute of Clinical Physiology, National Research Council of Italy (CNR), in Pisa, working on 3D visualization of biological processes. She participated in the Italian Flagship Project InterOmics as a post-doctoral fellow with the Signals and Images Laboratory, CNR Institute of Information Science and Technologies. Since April 2018, she has been a researcher with the CNR Institute of Biomedical Technologies, Pisa Section. Her research interests include mathematical modeling, bioinformatics, biology, genomics, and proteomics.

Emanuele Salerno received the graduate degree in electronic engineering from the University of Pisa, Italy, and joined the National Research Council of Italy in 1987. Currently, he is a senior researcher with the Institute of Information Science and Technologies in Pisa, Italy. He has been working in microwave tomography, monitoring of combustion processes, computer vision for robot guidance, and astrophysical imaging. His present scientific interests include applied inverse problems, image processing, IT for cultural heritage, and computational biology. He has been teaching instrumentation and measurements and microwaves with the University of Pisa, and is a senior member of the IEEE, the Signal Processing Society, a member of the Italian Federation of Electrical and Information Engineering (AEIT), and a fellow of the Electromagnetics Academy.

Monica Zoppè received the graduate degree in biology from the University of Milan, Italy. She has been working on cellular transport, viral vectors, and gene therapy with the Department of Biochemistry, University of Birmingham, United Kingdom, at the CNR Institute of Advanced Biomedical Technologies in Milan, at the Molecular Biology and Virology Lab of the Salk Institute, La Jolla, California, and at the Gene and Molecular Therapy Lab, International Centre of Genetic Engineering and Biotechnology in Trieste, Italy. At the CNR Institute of Clinical Physiology in Pisa since 2001, she founded the Scientific Visualization Unit, which is now under her direction. She has a teaching experience in biomedicine, biotechnology and gene therapy, and molecular graphics, is a member of the International Society for Computational Biology, the Molecular Graphics and Modelling Society, and DonneScienza, the Italian society of women scientists.

Ivan Merelli received the PhD degree in computer science from the University of Milano-Bicocca, in 2009 and was a visiting scientist with Harvard University, in 2014 and with Cambridge University, in 2015. He is staff scientist with the Institute for Biomedical Technologies (ITB), Italian National Research Council (CNR). His research activities concern statistical data analysis, software development and data management in the field of genomics and proteomics, in particular for the management of biological databases and high performance computing facilities. He co-authored more than 40 papers published in international peer-reviewed journals and more than 40 contributions in international conference proceedings.

Anna Tonazzini is a senior researcher with the Institute of Information Science and Technologies, National Research Council of Italy, in Pisa. She coordinated several projects in image processing and analysis, neural networks and learning, computational biology, and document analysis, and is co-author of more than 100 published papers. She has been the ISTI responsible for the national Flagship Project InterOmics, chaired and organized special sessions in international conferences, edited journal special issues, and is an editor of *Digital Signal Processing* (Elsevier). She supervised many theses in computer science, mathematics, and information engineering, two Ercim fellowships, and various post-doctoral positions. She is a member of the IASTED Technical Committee on Image Processing.