MoDLE: High-performance stochastic modeling of DNA loop extrusion interactions

Roberto Rossini[1], Vipin Kumar[2], Anthony Mathelier[2], Torbjørn Rognes[3,4], Jonas Paulsen[1,3]*

1. Department of Biosciences, University of Oslo, 0316 Oslo, Norway; 2. Centre for Molecular Medicine Norway (NCMM), Nordic EMBL Partnership, University of Oslo, 0318 Oslo, Norway; 3. Centre for Bioinformatics, Department of Informatics, University of Oslo, 0316 Oslo, Norway 4. Department of Microbiology, Oslo University Hospital, Rikshospitalet, 0424 Oslo, Norway; *Correspondence: jonas.paulsen@ibv.uio.no

Abstract

DNA loop extrusion emerges as a key process establishing genome structure and function. We introduce MoDLE, a computational tool for fast, stochastic modeling of molecular contacts from DNA loop extrusion capable of simulating realistic contact patterns genome wide in a few minutes. MoDLE accurately simulates contact maps in concordance with existing molecular dynamics approaches and with Micro-C data, and does so orders of magnitude faster than existing approaches. MoDLE runs efficiently on machines ranging from laptops to high performance computing clusters, and opens up for exploratory and predictive modeling of 3D genome structure in a wide range of settings.

Keywords

Loop extrusion, stochastic modeling, Hi-C, Micro-C, TAD

Background

DNA loop-extrusion, in which DNA is progressively reeled into transient loops, emerges as a key process in genome structure and function. The growing list of cellular processes where loop-extrusion plays a critical role now includes transcriptional regulation [1,2], DNA repair [3], VDJ-recombination [4], and cell division [5]. Recent single-molecule imaging experiments have provided direct observations of loop extrusion in vitro [6,7].

High-throughput chromosome conformation capture sequencing, including Hi-C [8] and Micro-C [9,10], have advanced our abilities to map three-dimensional (3D) genome organization through quantification of spatially proximal genome regions. The resulting data is usually rendered as a matrix of intrachromosomal and interchromosomal contact frequencies. These data increasingly deepen our understanding of 3D genome organization, and show DNA loop extrusion as a key process shaping genome structure [11–13]. In fact, topologically associating domains (TADs), which show up as sub-megabase-sized domains covering most of higher eukaryote genomes, are formed by loop extrusion [12]. TADs are relevant units of gene expression regulation, and are associated with disease when disrupted [14].

DNA loop extrusion is carried out by ring-shaped proteins (including cohesin and condensin) belonging to the Structural Maintenance of Chromosomes (SMC) family. These proteins are often referred to as loop extrusion factors (LEFs) [15]. The exact mechanism of how loop extrusion takes place in interphase is not fully understood. There is, however, convincing evidence that SMCs bind DNA to perform ATP-dependent loop extrusion in a symmetric or asymmetric fashion. Recent evidence suggests that cohesin can extrude DNA with a “swing-and-clamp” mechanism [16], and in a nontopological configuration where DNA is not encircled by the cohesin ring [17,18]. A loop starts extruding when a LEF binds to a genomic region, and continues processively until it is stalled by a DNA-bound CCCTC binding factor

(CTCF) oriented with its N-terminus pointing towards the extruding cohesin complex. A pair of CTCFs arranged in a convergent orientation can thus stall loop growth on both sides creating semi-stable loops visible in Hi-C as a characteristic "dot" at TAD corners [19]. Similarly, when extruding loops are stalled only on one side, a “stripe” can be observed along one or both TAD borders [20]. The protein WAPL transiently releases cohesin from chromatin, terminating the loop extrusion process [21,22]. The resulting loop-extrusion patterns have been found in a range of Hi-C datasets so far, emphasizing the evolutionary conserved role of loop extrusion in shaping 3D genome organization [19,23].

Disrupting any of the key proteins involved in DNA loop extrusion has a dramatic effect on genome 3D structure. WAPL depletion causes an increase in loop stability, with an accumulation of axial elements and weakening of compartments [21,24]. Depletion of Cohesin causes a large fraction of TADs and loops to disappear [24–26]. Similarly, depletion of CTCF induces loss of loops and TADs [24,27].

Modeling and simulation of DNA-DNA contact patterns is a powerful approach for understanding underlying molecular mechanisms and for predicting the effect of DNA perturbations. Polymer simulations and molecular dynamics (MD) have been used for modeling of TADs to study their structure and dynamics [28–31]. Computational modeling and simulation of loop extrusion has proven useful for predicting the effects of perturbations to TAD borders, and to properly understand patterns seen in Hi-C data. Initial models [15,32] of loop extrusion used the Gillespie algorithm to characterize looping properties and chromatin compaction, and did not sample contact maps. Subsequent models used HOOMD particle simulation [33] to perform homopolymer simulations where modeled LEFs extrude the polymers and halt at boundaries with properties defined from CTCF motif instance orientation and ChIP-seq signal strength [11,25]. Recently, to efficiently simulate larger genome regions, a combination of one-dimensional (1D) simulations with 3D polymer modeling has been applied to sample multiple conformations combined into contact maps.

LEF binding, release and stalling probabilities are then modeled explicitly [34–36]. These simulations are typically implemented using the OpenMM molecular simulation framework [37]. The simulations can be used to explore and rule out molecular mechanisms. For example, Banigan et. al assessed the level of DNA compaction that can be achieved by different loop extrusion mechanisms, and concluded that one-sided loop extrusion alone fails to achieve the level of compaction observed in large metazoan genomes [36]. Other approaches embed epigenetic data in combination with crosslinking proteins to model and study conformational variability across complex chromatin regions [38,39]. To the best of our knowledge, no standalone software for modeling and simulation of loop extrusion exists.

We introduce MoDLE (Modeling of DNA Loop Extrusion), a high-performance stochastic model of DNA loop extrusion capable of efficiently simulating contacts from loop extrusion genome wide. In contrast to MD simulation approaches, simulating loop extrusion contacts using MoDLE is a straightforward process only requiring two input files and execution through a command line interface (CLI). MoDLE can simulate a contact matrix with the molecular interactions generated by DNA loop extrusion on the entire human genome in a matter of minutes using less than 1 GB of RAM. MoDLE opens up for rapid simulation and parameter exploration of DNA loop extrusion on genomes of any size, including large mammalian genomes.

Results

MoDLE: Modeling of DNA Loop Extrusion

MoDLE uses fast stochastic simulation to sample DNA-DNA contacts generated by loop extrusion. Binding and release of LEFs and barriers and the extrusion process is modeled as an iterative process (see Fig. 1). At the beginning of a simulation MoDLE goes through a

burn-in phase where LEFs are progressively bound to DNA, without sampling molecular contacts. The burn-in phase runs until the average loop size has stabilized. Active LEFs are extruded through randomly sampled strides along the DNA in reverse and forward directions. Each epoch, LEFs are released with a probability based on the average LEF processivity and extrusion speed. LEFs that are released in the current epoch will rebind to randomly sampled DNA regions in the next epoch. Extrusion barriers (e.g. CTCF binding sites) are modeled using a two-state (bound and unbound) Markov process. Each extrusion barrier consists of a position, a blocking direction and the Markov process transition probabilities. The occupancy of each extrusion barrier can be specified individually through the score field in the input BED file. Alternatively, users can specify a uniform barrier occupancy that is applied to all extrusion barriers. MoDLE accepts a large number of optional parameters to specify the model's behavior. For example, users can specify the number of LEFs to be instantiated for each Mbp of simulated DNA using the --lef-density parameter. LEF-barrier and LEF-LEF collisions are processed each simulation epoch. Collision information is used to update candidate strides to satisfy the constraints imposed by collision events, and to compute how extrusion in the next epoch should proceed.

During a simulation, sampled molecular contacts are accumulated into a specialized contact matrix data structure with low memory overhead. MoDLE execution continues until a target number of epochs or a target number of loop extrusion contacts are simulated. Finally, contacts generated by all simulation instances for a given chromosome are written to an output file in cooler format [40] (Fig. 1).

With default settings, MoDLE will run over 500 simulation instances for each chromosome simulated. Thus, simulation instances can run in parallel, making efficient use of the computational resources of modern multi-core CPUs. We designed MoDLE such that each simulation instance requires less than 10 MB of memory to simulate loop extrusion on large mammalian chromosomes, such as chromosome 1 from the human genome. To achieve

133 high-performance, MoDLE stores most of its data in contiguous memory. Data is indexed
134 such that extrusion barriers and extrusion units in a simulation instance can be efficiently
135 traversed in 5'-3' and 3'-5' directions. This allows MoDLE to bind/release LEFs, process
136 collisions, register contacts and extrude DNA in linear time-complexity.
137
138 More design and implementation details are available in Additional file 1 as well as MoDLE's
139 GitHub repository github.com/paulsengroup/modle.

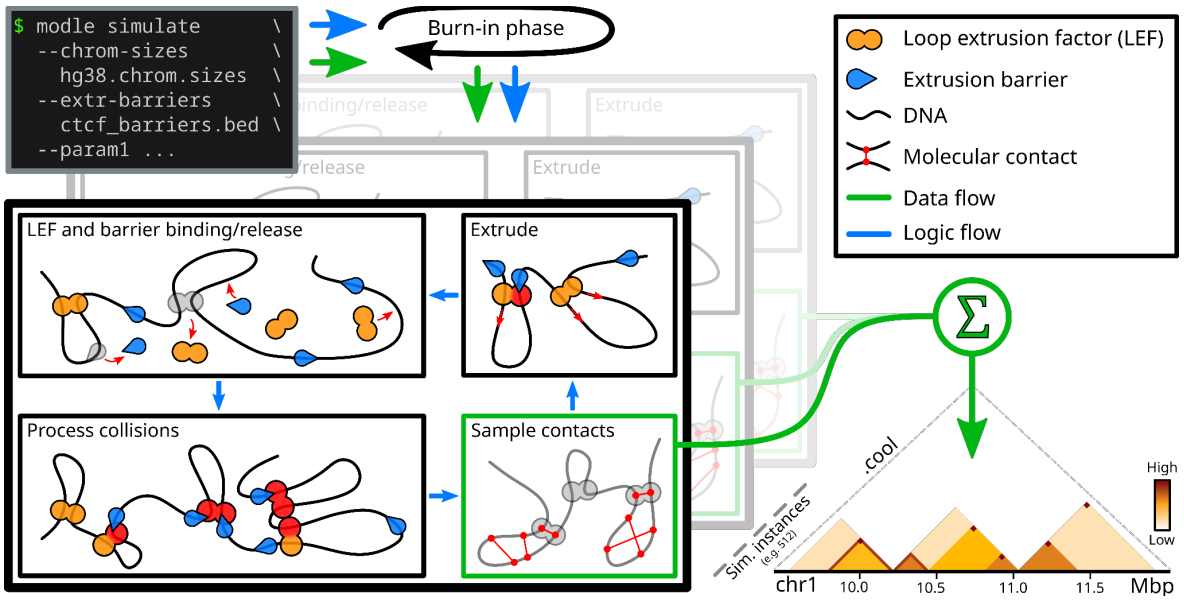

Fig 1: Schematic and simplified overview of MoDLE. Input files specify genome regions to be simulated (e.g. a chrom.sizes file) and their barrier positions (e.g. CTCF binding sites and orientation) in BED format. Optional parameters control the specifics of a simulation. Loop Extruding Factors (LEFs) bind to, extrude and release from the regions and interact with modeled barriers according to input parameters. Loop extrusion and intra-TAD contacts of a randomized subset of loops are recorded each epoch and aggregated into an output cooler file containing the final simulated contact frequencies. Simulation halts when a target number of epochs or a target number of loop extrusion contacts have been simulated.

Comparison with Micro-C data and MD simulations

To assess MoDLE's ability to reproduce contact data features known to be stemming from loop extrusion, we simulated genome-wide DNA-DNA contacts based on available CTCF and RAD21 ChIP-seq data in H1-hESC cells (see Methods). MoDLE is capable of simulating loop extrusion molecular contacts and intra-TAD contacts separately. A rendering of the resulting loop extrusion molecular contacts heatmaps show characteristic stripe and dot patterns at TAD borders (Fig. 2A). Simulated TAD contacts show enrichment of contacts within TADs, including a nested structure of the TADs (Fig. 2B). In combination, these patterns resemble well-characterized patterns observed in Micro-C data (Fig. 2C).

Even though no stand-alone software exists for direct side-by-side comparison, we adapted available code based on OpenMM [36] to systematically compare the output with that of MoDLE (see Methods). We chose OpenMM for comparison as it is an efficient and widely used system for simulating loop extrusion [12,34–36,41]. Using the same input data, we simulated contacts in five different 10 Mbp regions on five different chromosomes. In general, MoDLE produces contact patterns similar to OpenMM (Fig. 2D and Supplementary Fig. 1, Additional file 2), and MoDLE output and OpenMM correlate strongly (Pearson ρ=0.93; see Supplementary Fig. 2, Additional file 2). By comparing contacts with corresponding Micro-C data (Fig. 2D), we see a median pixel accuracy (see Methods) of 0.70 for MoDLE and 0.67 for OpenMM, signifying that MoDLE indeed simulates contacts observed in Micro-C similar to OpenMM (Fig. 2E).

Altering MoDLE's input parameters to *in silico* mimicking depletion of CTCF and WAPL, shows an expected loss of TAD insulation patterns [27] upon *in silico* depletion of CTCF, and more pronounced stripe and dot patterns [22] when mimicking WAPL depletion (Fig. 2F). We conclude that MoDLE is capable of simulating loop extrusion and TAD contact patterns similar to existing state-of-the-art molecular dynamics (OpenMM) approaches.

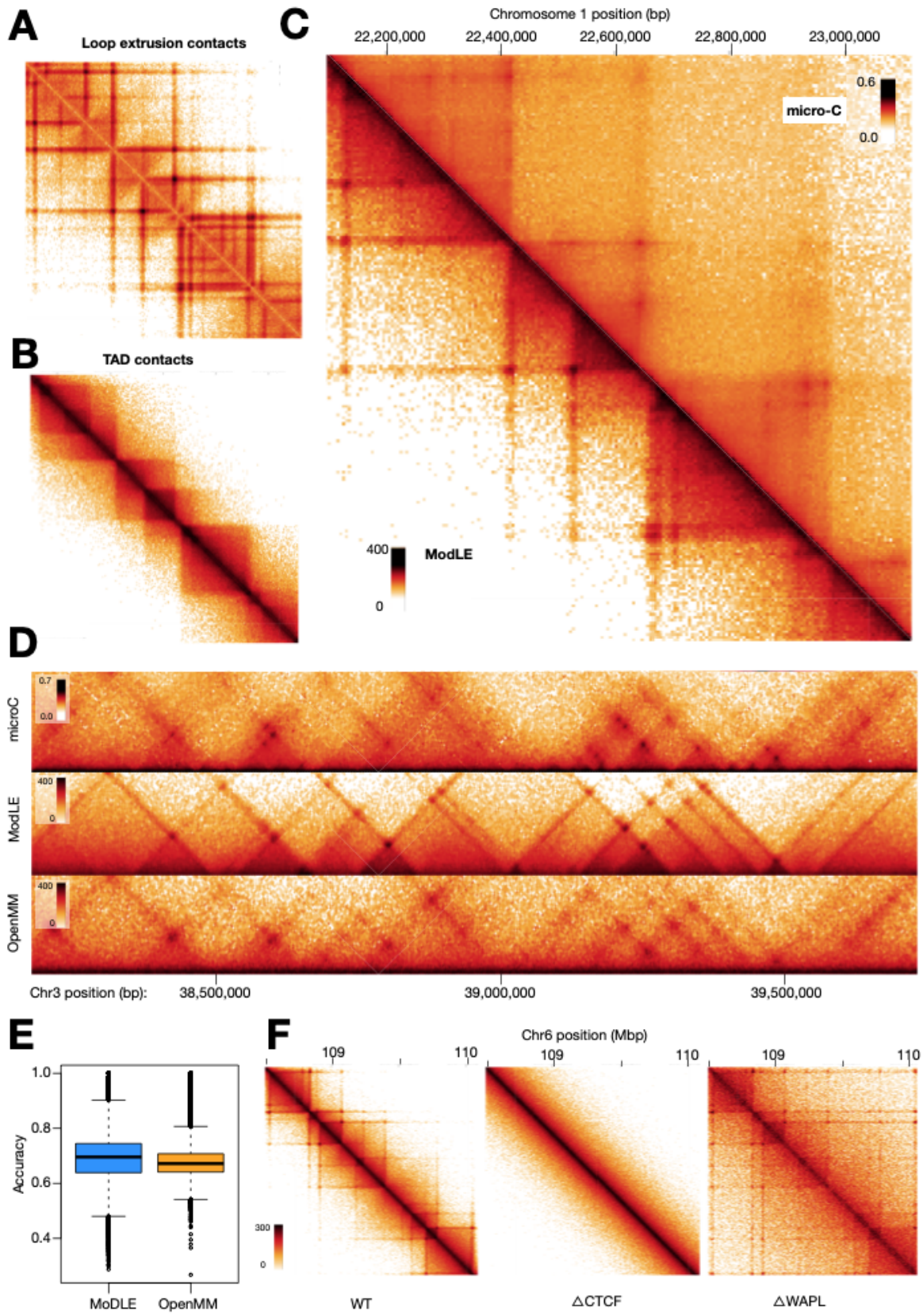

Fig. 2: Comparison of MoDLE with OpenMM and Micro-C data. **A**: Simulated MoDLE

contact frequencies solely mediated by LEFs. **B**: Intra-TAD contacts (only) generated with MoDLE. **C**: Lower triangle: Loop extrusion and intra-TAD contacts from MoDLE in the same region as for A and B. Upper triangle: Micro-C data from the same region. **D:** Side-by-side comparison of Micro-C data, MoDLE output and OpenMM output for a region on chromosome 3 in H1-hESC. **E:** Quantitative comparison of the accuracy of MoDLE and OpenMM in reproducing stripe and dot pixel-patterns observed in modeled regions in H1-hESC cells (see Methods). **F:** *in silico* simulated molecular contacts mimicking CTCF and WAPL depletion. Left: Wildtype (WT) output of MoDLE in a region on chromosome 6 in H1-hESC. Middle: effect on MoDLE output when CTCF barriers weakly associate with their binding sites. Right: effect on MoDLE output when LEFs are less likely to be released from DNA, thus mimicking WAPL depletion.

Benchmarking of execution time and memory usage

MoDLE is designed for fast genome-wide simulation of loop extrusion contact patterns. A genome-wide run with default settings, simulating loop extrusion on the entire human genome using barriers from H1-hESC (38,815 CTCF barriers and 61,766 LEFs; see Methods) takes ~40 seconds on a compute server (server A; see Table 1) and ~5 minutes on a laptop (laptop A; see Table 1), generating over 370 million contacts. To systematically compare MoDLE execution time and memory usage with OpenMM, we generated synthetic input datasets with increasing genome size (1-500 Mbp) and number of CTCF barriers (4 barriers per Mbp of DNA simulated) (see Methods for details). The inputs were identical in MoDLE and OpenMM. Each measurement was repeated 10 times for MoDLE and 5 times for OpenMM. For MoDLE we run benchmarks using 1-52 CPU cores, while for OpenMM we tested the CPU (server C; see Table 1) and GPU (server D; see Table 1) implementations. We computed median elapsed wall clock time and peak memory usage for MoDLE and OpenMM. The resulting comparisons show that MoDLE simulations using 52 CPU cores complete within 0.7-71 seconds from the smallest to the largest genome region. OpenMM

requires 2 hours and 35 minutes for the smallest genome region and over 41 hours for a genomic region of 250Mbp (Fig. 3A). Due to very long execution times, OpenMM runs above 250Mbp were not performed. For the compared genome regions, MoDLE is 4000-5000 times faster than OpenMM (Fig. 3A). OpenMM simulations without GPU acceleration were particularly slow and were only used to simulate genome regions below 5 Mbp, and required up to 35 hours 20 minutes of execution time (Fig. 3A). Thus, in practice running OpenMM requires access to GPUs, while MoDLE runs efficiently using CPUs.

Comparing peak memory usage, MoDLE uses less memory than OpenMM for regions smaller than 200Mbp, and requires more memory for larger systems. Nevertheless, memory usage of both MoDLE and OpenMM scales linearly for increasing genome region sizes and is for all practical purposes within reasonable limits on today's computers regardless of genome size (Fig. 3A-B).

Multithreading efficiently reduces MoDLE's execution time for increasingly large genome sizes. With multithreading (52 CPU cores on server B; see Table 1), MoDLE can simulate loop extrusion contacts for a genome size of 500 Mbp in a little over one minute (Fig. 3C). Using a single thread (1 CPU core on server B; see Table 1), the same run takes around 12 minutes (Fig. 3C), which is still reasonable from a practical perspective and much faster than GPU accelerated OpenMM simulations. MoDLE peak memory usage is only slightly affected by multithreading, as each simulation instance only requires an additional 1-10 MB of memory (Fig. 3D). When simulating more than one chromosome, peak memory usage does not follow a simple linear pattern (Fig. 3D), as it is affected by the order in which simulation tasks are executed. This can lead to scenarios where for a brief period, two or more contact matrices are stored in system memory. We conclude that MoDLE, in contrast to OpenMM, runs efficiently even on systems with few CPU cores, such as laptop computers.

Further, we analyzed the strong scaling properties of MoDLE by simulating loop extrusion on the entire human genome (GRCh38; 3088Mbp). Increasing the number of CPU cores from 1 to 52, MoDLE execution time scales close to theoretical optimum (see Methods for details) (Fig. 3D; blue lines). Simulating loop extrusion on the human genome takes from 1 hour and 21 minutes (1 CPU core on server B; see Table 1) to 1 minute and 48 seconds (52 CPU cores on server B; see Table 1). We conclude that MoDLE can efficiently run on machines with a wide range of capabilities, ranging from laptop computers with 4-8 CPU cores, to multi-socket servers with over 50 CPU cores. Memory usage increases with the number of CPU cores, but never beyond reasonable limits on modern computers (Fig. 3D; orange line).

In conclusion, MoDLE is orders of magnitude faster than OpenMM in simulating loop extrusion contacts, and is especially efficient in simulating large genome regions or large input data sets. MoDLE can run efficiently on machines ranging from low-powered laptop computers to powerful multi-socket servers.

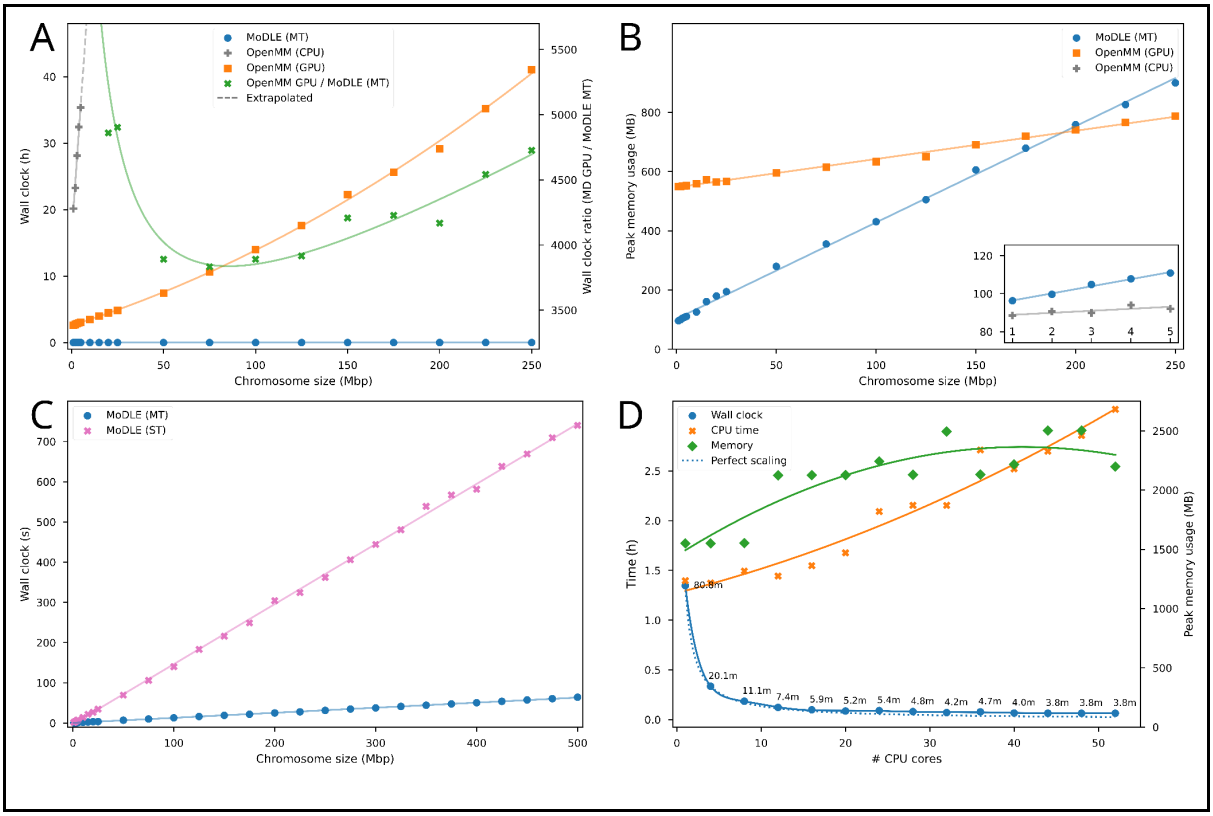

Fig. 3: Benchmarking MoDLE and OpenMM.

A: Median memory usage (in MBs) of MoDLE with multithreading (blue) compared to OpenMM with GPU (orange) for chromosome regions ranging in size from 1 to 250 Mbp. Inset shows comparison between MoDLE (blue) OpenMM with CPU (gray) for chromosome regions ranging in size from 1-5 Mbp. **B**: Median elapsed execution time (hours) of MoDLE with multithreading (blue), OpenMM with CPU (gray), OpenMM with GPU (orange) and the ratio of OpenMM (GPU) to MoDLE. Dotted lines are extrapolated. **C**: Comparison of the median elapsed execution time (seconds) of MoDLE with (blue) and without (pink) multithreading for chromosome regions ranging in size from 1 to 500 Mbp. **D:** Comparison of median elapsed execution time (hours) of MoDLE utilizing from 1-52 CPU cores. Blue line shows elapsed wall clock time (hours), whereas the orange line shows the CPU time (hours). The dotted line illustrates the corresponding theoretical perfect scaling of the executing time. Green line shows median peak memory usage (right axis; MB).

Genome wide parameter optimization

Since MoDLE simulates genome-wide loop extrusion in a few minutes, systematic exploration of features underlying loop extrusion becomes feasible. To illustrate this point, we optimized the parameters underlying the modeled binding kinetics of CTCF. MoDLE implements this as a Markov process with an “Unbound” and a “Bound” state. With this model, the self-transition probabilities P_{UU} and P_{BB} specify how stably associated CTCF is once bound to DNA. The stationary distribution of the Markov chain reflects the probability of a given CTCF binding site to be bound (π_B) in a simulation epoch (see Fig 4A). Simulation of loop extrusion contacts using MoDLE or OpenMM can take advantage of ChIP-seq data from CTCF or Cohesin to infer CTCF binding probabilities. Yet, when ChIP-seq data is not available it is possible to simulate loop extrusion using a constant and uniform CTCF binding probability that is chosen to optimize similarity with the Micro-C (or Hi-C) data. To optimize these parameters, we make use of an approach based on Bayesian optimization using Gaussian processes (see Methods). This optimization procedure attempts to minimize an objective function without making assumptions on its analytic form. To assess MoDLE’s performance we devised an objective function representing the similarity in stripe position and length between two contact matrices using H1-hESC Micro-C data (see Methods for details). After convergence (Fig. 4B), the resulting optimal parameters revealed a “path” of near-optimal combinations of transition probabilities and CTCF occupancy probabilities (Fig. 4C), with the optimal combination at $\pi_B = 0.694$ and $P_{UU} = 0.978$. Notably, the results show that the near-optimal CTCF binding probabilities have values typically >0.8, except when the stability of binding (P_{UU}) is high, supporting a dynamic exchange model of CTCF binding and a transient maintenance of loops [42]. Thus, CTCFs can potentially compensate for a lower CTCF binding probability by increasing “traffic jams” between extruding LEFs [13]. From the optimized parameters (Fig. 4C, red star), we simulated genome-wide loop extrusion contacts aiming at comparing these with MoDLE results using ChIP-seq input and with Micro-C. The resulting comparison shows that even uniform, optimized CTCF binding probabilities can

recapitulate many of the features seen in Micro-C and in the standard MoDLE output. Selecting a sub-optimal combination of parameters (Fig. 4C, green triangle) results in unrealistic contact patterns (Fig. 4D). In conclusion, MoDLE opens up for efficient exploration of parameters underlying DNA-DNA contact dynamics genome wide.

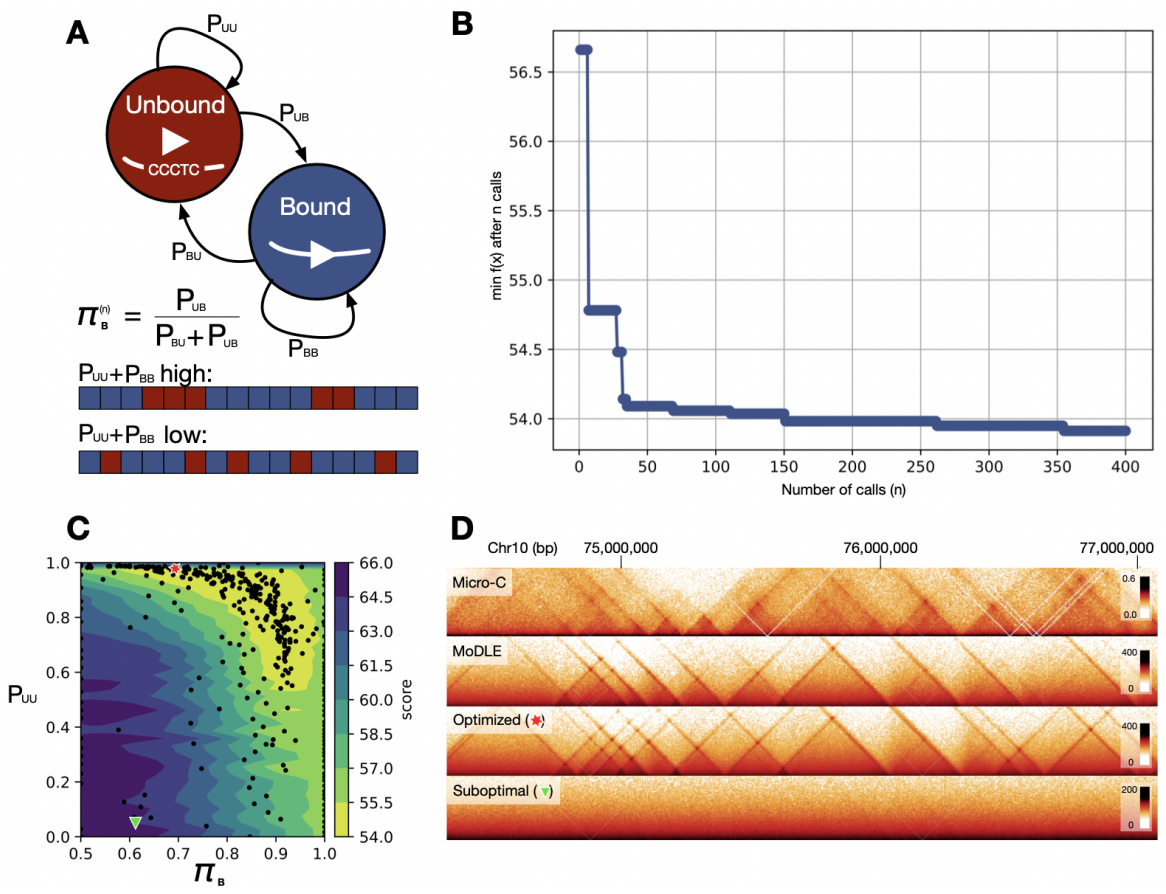

Fig. 4: Genome-wide optimization of CTCF binding kinetics underlying loop extrusion. **A**: A Markov chain with an "Unbound" and "Bound" state underlies MoDLE loop extrusion barrier modeling. The self-transition probability for the "Bound" state (P_{BB}) reflects how stably barrier elements (i.e. CTCF) are bound to their binding sites. The stationary distribution of the Markov chain (π_B) provides the CTCF binding probability at a given epoch in the simulation. **B**: Convergence of the objective function (reflecting fit with Micro-C data) during the Bayesian optimization procedure. **C**: Comparison of objective function in the parameter search space of P_{UU} and π_B. Optimal combination shown with a red star. **D**: Side-by-side comparison of H1-hESC Micro-C data (top panel), MoDLE using ChIP-seq

data for parameterization of boundaries (MoDLE), MoDLE using the optimized parameters (P_{UU} and π_B) uniformly across boundaries in place of ChIP-seq data (Optimized), and MoDLE using a sub-optimal parameter combination shown with a green triangle in C (Suboptimal).

Discussion

Efficient and realistic simulation of DNA-DNA spatial contacts is increasingly required for modeling and exploring genome structure and regulation. For example, our ability to reliably predict effects of mutations to TAD borders relies on available tools for simulating and comparing spatial contact data from normal and pathogenic states [14]. Further, simulations can be invaluable for exploring general genome folding principles [11] or underlying principles of loop extrusion [12,35,36]. Efficient tools for loop extrusion simulation will contribute to increasing our understanding of mechanisms ranging from gene regulation [1,2], to DNA repair [3]. MoDLE represents, to the best of our knowledge, the first command-line tool for high-throughput loop extrusion contact simulation. We expect MoDLE to supplement, rather than replace existing MD tools; especially in cases where large genome regions or large data sets need to be analyzed or simulated. This would in particular be the case for large-scale exploration of parameters underlying genome structure properties, as exemplified here for the binding kinetics of CTCF.

New developments in experimental techniques augmented by integrated computational modeling, will continue to shed light on new genome organization principles at a rapid pace [43]. With MoDLE's focus on computational speed and its modular architecture, new developments and knowledge are expected to easily be integrated into the tool to increase the complexity and realism of the underlying modeling parameters.

Conclusions

We have developed MoDLE (Modeling of DNA Loop Extrusion), allowing high-performance stochastic modeling of DNA loop extrusion. MoDLE simulates loop extrusion contact matrices on large genome regions in a few minutes, even on low-powered laptop computers. MoDLE is available as a command line tool and can be accessed at github.com/paulsengroup/modle.

Methods

1. MoDLE implementation and design overview

MoDLE is implemented in C++17 and is compiled with CMake. MoDLE uses a producer-consumer architecture where a single producer (a thread) communicates with multiple consumers through asynchronous message passing. The producer thread is responsible for reading input files and generating a set of simulation tasks to be consumed by a pool of worker threads. Tasks are implemented as light-weight C++ structs that are computationally cheap to generate and consume. A single task contains all the information needed for simulating DNA loop extrusion on a single chromosome in a specific simulation instance. Simulation instances are for the most part independent from each other and can thus run in parallel. We designed MoDLE such that each simulation instance requires less than 10 MB of memory to simulate loop extrusion on large mammalian chromosomes, such as chromosome 1 from the human genome. The space complexity of the thread–local state is linear with respect to the number of LEFs or extrusion barriers, whichever is largest. For a more detailed overview of MoDLE's implementation see Section 1, Additional file 1.

Most of MoDLE's memory budget is used to store molecular contacts generated by loop extrusion. MoDLE stores one instance of its custom contact matrix data structure for each chromosome that is being actively simulated. The space complexity of a contact matrix instance depends on the chromosome length, diagonal width and bin size. With default settings, representing contacts for chromosome 1 of the human genome requires approximately 120 MB of memory. Common operations on the contact matrix class are made thread-safe using lock striping implemented through hashing. For more details regarding the specialized contact matrix data structure refer to Section 2, Additional file 1.

To achieve high-performance, MoDLE stores most of its data in contiguous memory using simple data structures such as vectors and bitsets (see Section 3, Additional file 1). Data is indexed such that extrusion barriers and extrusion units in a simulation instance can be efficiently traversed in 5'-3' and 3'-5' directions (see Section 8, Additional file 1). This allows MoDLE to bind/release LEFs, process collisions, register contacts and extrude DNA in linear time-complexity and with good locality of reference. The only step relying on an algorithm with super-linear time complexity is indexing, which has a worst-case time complexity of $O(n \log n)$ while approaching $O(n)$ for the typical case.

More design and implementation details are available in Additional file 1. The latest MoDLE source code can be obtained in MoDLE's GitHub repository:

github.com/paulsengroup/modle

2. Running a simulation instance

The entire simulation instance is executed by a single worker thread and consists of the following phases:

- Wait until one or more tasks are available on the task queue.

- Setup the simulation internal state based on the task specification, this includes seeding the PRNG and setting the initial state for the extrusion barriers based on the occupancy (see Sections 1,3 and 4, Additional file 1).
- Run the simulation loop until a stopping criterion is met.

A single simulation epoch is articulated in the following steps:

- Select (inactive) LEFs that are currently not associated with DNA, and activate them. This is done by loading LEFs to a random position on the chromosome that is being simulated. The position is sampled from a uniform distribution (see Section 5, Additional file 1).
- Index extrusion units moving in the same direction so that they can be visited in 5'-3' and 3'-5' order (see Section 8, Additional file 1).
- Randomly select a subset of the active LEFs and use their position along the chromosome to generate molecular contacts in the chromosome contact matrix (see Section 9, Additional file 1).
- Generate candidate moves for each extrusion unit (see Section 10, Additional file 1).
- Update the extrusion barrier states by computing the next state in the Markov chain used to model extrusion barriers (see Section 6, Additional file 1).
- Detect collision events between LEFs and extrusion barriers as well as between LEFs (see Sections 12b-d, and g, Additional file 1).
- Update the candidate moves for extrusion units involved in collision events to satisfy the constraints imposed by the collision events (see Sections 12e-g, Additional file 1).
- Advance LEFs' extrusion units by their respective moves (see Section 5, Additional file 1). Because of the preceding steps, this will yield a new valid simulation state, as moves have been updated to satisfy all the constraints imposed by collision events.
- Iterate over active LEFs and release them based on the outcome of a Bernoulli trial whose probability of success is computed based on the average LEF processivity and LEF state (e.g. LEFs whose extrusion units are involved in collision events with a pair of extrusion barriers in convergent orientation have a lower probability of being

released). LEFs that are being released go back in the pool of available LEFs and will be loaded on a new genomic region during the next epoch (see Sections 5, Additional file 1).

MoDLE will continue iterating through the above steps until one of the simulation stopping criteria is met:

- A given number of epochs have been simulated
- Enough contacts have been registered to reach a target contact density.

Both stopping criteria can be modified by users. By default, MoDLE will simulate loop extrusion until reaching an average contact density of 1 contact per pixel.

3. Hardware specifications

Analysis and benchmark code used to generate the data accompanying was run using the hardware specifications listed in Table 1.

Table 1: Hardware specifications of computational resources used for simulation and benchmarking

Identifier	**CPU model**	**System memory**	**Operating system**	**Accelerator (GPU)**
Laptop A	Intel Core i9-9880H (8 cores)	64 GB (4x 16 GB, DDR4 UDIMM 2667 MT/s dual-channel)	Arch Linux (Linux v5.17.1)	NVIDIA Quadro RTX 4000 (8 GB)
Server A	2x AMD EPYC 7742 (2x 64 cores)	2048 GB (32x 64 GB, RDIMM DDR4 2933 MT/s eight-channel)	RHEL 8.5 (Linux v4.18.0-305)	N/A
Server B	2x Intel Xeon Gold 6138 (2x 20 cores)	192 GB (12x 16 GB, RDIMM DDR4 2666 MT/s six-channel)	RHEL 7.9.2009 (Linux v3.10.0-1160.6.1)	
Server C	2x Intel Xeon Gold 6230R (2x 26 cores)	192 GB (12x 16 GB, RDIMM DDR4 2933 MT/s six-channel)		

Server D	2x Intel Xeon Gold 6126 (2x 12 cores)	384 GB (24x 16 GB, RDIMM DDR4 2666 MT/s six-channel)		4x NVIDIA Tesla P100 (16 GB)

4. MoDLE simulations

MoDLE's data used for the heat map comparison shown in Fig. 2 were generated using the make_heatmap_comparison Nextflow [44] workflow available on GitHub and archived on Zenodo [45,46].

The list of candidate extrusion barrier positions and directions were generated by running MAST from the MEME suite [47] on GRCh38.p13 (GCF_000001405.39 [48] using the CTCF frequency matrix MA0139.1 from JASPAR 2022 [49].

The list of candidate barriers was then filtered using CTCF and RAD21 ChIP-seq data (fold-change over control and optimal IDR thresholded peaks). In brief, candidate barriers were intersected with the narrow-peak BED files for CTCF and RAD21. Then, each filtered barrier region was assigned with an occupancy computed by passing the RAD21 fold-change over control signal through a logistic function. Finally, the output of the logistic function was binned at 1 kbp to yield a barrier occupancy that is proportional to the number of CTCF motif instances as well as RAD21 fold-change over control signal. This procedure is largely based on [Fudenberg 2016]. The result of the procedure outlined above is a list of extrusion barrier occupancies binned at 1 kbp resolution. CTCF and RAD21 ChIP-seq for H1-hESC data was downloaded from ENCODE [50,51] (ENCFF255FRL [52] , ENCFF473IZV [53], ENCFF821AQO [54] and ENCFF913JGA [55].

Contact matrices were generated using MoDLE v1.0.0-rc.3 with the parameters from Supplementary Table 1, Additional file 3. Parameters not listed in the table were left at default.

Contact matrices produced by MoDLE were then subsampled to an average contact density of 3 using cooltools random-sample v0.5.1 [56]. The resulting cooler files were then

converted to multi-resolution cooler files using cooler zoomify [40]. Finally, multi–resolution contact matrices were visualized in HiGlass (v. 0.8.0) [57].

5. Molecular dynamics (OpenMM) simulations

Molecular dynamics data used for the heat map comparison in Fig. 2 were generated using the make_heatmap_comparison Nextflow workflow available on GitHub and archived on Zenodo [45,46]. This workflow uses OpenMM [37] to run MD simulations.
Simulation code is largely based on [58]. Simulations were carried out on 10 Mbp regions from chromosomes 2, 3, 5, 7 and 17 using a monomer size of 1 kbp and 200 kbp for LEF processivity and separation. Extrusion barrier positions, directions and occupancy were generated following the procedure outlined in Methods (part 1).
Contact matrices were generated with Polychrom [59] using a bin size of 5 kbp. The resulting cooler files were then converted to multi-resolution cooler files using cooler zoomify v0.8.11 [57].

6. Assessing loop extrusion feature similarities from contact frequencies

To objectively compare the contact matrices produced by MoDLE with contact matrices generated from Micro-C experiments and MD simulations we developed a specialized scoring algorithm. The algorithm was inspired by Stripenn [60].
The score is computed on rows and columns of a pair of contact matrices of identical resolutions transformed as follows.
First, both matrices are convolved using the difference of Gaussian (DoG). This highlights stripe and dot patterns found in contact matrices. Next, the transformed contact matrices are discretized using a step function mapping values below a certain threshold to 0 and all the others to 1. This results in two binary matrices, where non-zero pixels can be interpreted as

part of a stripe or dot. Finally, we take advantage of the fact that stripes produced by loop extrusion always should start from the matrix diagonal. Thus, given a row or column of pixels starting on the matrix diagonal, and extending away from it, we stipulate that the last non-zero pixel in the vector of values represents the end of a stripe produced by DNA loop extrusion.

Given the above, we can measure the similarity of stripes between two contact matrices by considering the same row of pixels in a pair of contact matrices, computing the last non-zero pixels in both rows, and counting the number of matches. The same approach can be applied to columns of pixels. Finally, counting mismatches instead of matches can be used as a measure of dissimilarity. Contact matrix convolution and discretization, as well as computing this special score can be done using MoDLE's helper tools (modle_tools transform and modle_tools evaluate respectively).

7. Contact matrix comparison

For comparison with MoDLE and OpenMM output, we used available Micro-C data from H1-hESC because these were of high resolution and had accompanying ChIP-seq data for both CTCF and RAD21 (ENCFF255FRL [52] , ENCFF473IZV [53], ENCFF821AQO [54] and ENCFF913JGA [55]). To assess stripe similarity of a pair of contact matrices we used the scoring algorithm described in Methods (part 6). The score was computed using Micro-C data as the ground truth. Pixel accuracy was computed as the ratio of correctly classified pixels to the total number of pixels in a 3 Mbp subdiagonal window around each barrier. The Pearson correlation between OpenMM and MoDLE was calculated based on all corresponding 5 kbp-pixel values in the 3 Mbp subdiagonal window of the OpenMM simulation regions.

8. Benchmark methodology

Benchmarks were run on a computing cluster using the run_benchmarks Nextflow workflow available on GitHub and archived on Zenodo [45,46].

We ran two suites of benchmarks to assess the performance of MoDLE and compare it with that of molecular dynamics simulations based on OpenMM..

The first suite (Fig. 3A-C) compared the performance of MoDLE and OpenMM when simulating loop extrusion on an artificial chromosome with increasing length (ranging from 1 to 250 Mbp).

This benchmark was run using MoDLE (1 and 52 CPU cores) as well as OpenMM GPU and CPU implementation (1 CPU core, 1 GPU and 8 CPU cores respectively). CPU benchmarks were run on server C while benchmarks relying on GPU acceleration were run on server D (see Table 1). For OpenMM CPU implementation we limited the number of CPU cores to 8 (16 SMT cores) as the CPU implementation is known to not scale well past 16 threads [61]. OpenMM CPU implementation was used to simulate chromosome lengths up to 5 Mbp for practical reasons. MoDLE was run with default settings except for the number of cells, which was set to 104 to match the maximum number of available SMT cores.

OpenMM simulations were run using a monomer size of 2 kbp and LEF processivity and separation of 200 kbp.

The second suite of benchmarks involved simulating loop extrusion on the human genome (GRCh38) using MoDLE with a number of CPU cores ranging from 1 to 52. MoDLE was run with default settings except for the number of cells which was set to 104. The extrusion barrier annotation was generated as described in Methods (part 1).

In both cases, measurements were repeated 10 times for MoDLE and 5 times for OpenMM.

9. Genome-wide extrusion barrier parameter optimization

The genome-wide optimization of parameters affecting extrusion barrier occupancies was carried out using the optimize_MoDLE_sim_params Nextflow workflow available on GitHub and archived on Zenodo [45,46].

The first step in the optimization procedure is running Stripenn v1.1.65.5 [60] on the H1-hESC Micro-C (4DNFI9GMP2J8 [62], [63]) dataset to identify architectural stripes, which resulted in the identification of 4930 stripes. A small subset of these stripes were visually validated by comparing the annotated stripes with stripes that are visible from Micro-C data. Annotated stripes were split into two equally sized datasets by random sampling without replacement. One dataset was used for parameter optimization while the other was used for validation.

Parameter optimization is performed through the Bayesian optimization from scikit-optimize v0.9.0 [64] using an objective function based on the scoring metric described in Methods (part 6).

The parameters that are being optimized are the extrusion barrier occupancy (π_B) and P_{UU}, the self-transition probability of the unbound state.

The evaluation of the objective function proceeds as follows:

- A new genome-wide simulation is performed using the parameters proposed by the optimizer.
- The resulting cooler file is transformed with modle_tools transform by applying the difference of Gaussians followed by a binary discretization step, where pixel values above a certain threshold are discretized to 1 and all the others to 0.
- The score described in Methods (part 6) is then computed row and column-wise on the entire genome using modle_tools eval, producing two BigWig files. Here, the transformed Micro-C 4DNFI9GMP2J8 [62] dataset is used as reference [62,63].
- Scores are intersected with the extrusion barrier dataset for optimization and validation considering stripe direction (i.e. vertical stripes are intersected with column-wise scores while horizontal stripes are intersected with row-wise scores).

- Scores resulting from the intersection are then averaged, producing a floating-point number that is then returned to the optimizer, which will try to minimize this number.

In the transformation step a σ of 1.0 and 1.6 are used to generate the less and more blurry contact matrices to subtract when computing the difference of Gaussians. For the binary discretization of the Micro-C data a threshold of 1.5 was used, while simulated data was discretized using 0.75 as threshold.

The optimizer evaluated the objective function 400 times, each time computing the average score for the training and validation datasets.

Finally, the parameters that yielded the best score on the training dataset were used to generate a contact matrix in cooler format (see Fig. 4D, bottom panel).

Abbreviations

3D - Three-dimensional

BED - Browser Extensible Data

CLI - Command line interface

CTCF - CCCTC binding factor

hESC - Human Embryonic Stem Cells

kbp - kilo basepair

LEF - Loop Extruding Factor

Mbp - Mega basepair

MD - Molecular dynamics

PRNG - Pseudorandom Number Generator

SMC - Structural Maintenance of Chromosomes

SMT - Simultaneous Multithreading

TAD - Topologically Associating Domain

Declarations

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Competing interests

None declared

Availability of data and materials

Project name: MoDLE

Project home page: https://github.com/paulsengroup/modle

Archived version: 10.5281/zenodo.6424697

License: MIT

Operative system(s): UNIX-like (Platform independent when using containers)

Programming language: C++

Other requirements:

- C++17 compiler (e.g. GCC 7+, Clang7+, AppleClang 10+)
- CMake 3.16 or newer
- Conan 1.43 or newer
- Python3 (to install and run Conan)
- Scipy (Python3 package, required to run unit tests)
- wCorr (R package, required to run unit tests)

The complete list of MoDLE dependencies is available in Supplementary Table 2, Additional file 3. Dependency installation is automated using CMake and Conan.

MoDLE source code is available on GitHub at https://github.com/paulsengroup/modle [65] and is archived on Zenodo at https://doi.org/10.5281/zenodo.6424697 [66]. MoDLE's version used throughout the manuscript is MoDLE v1.0.0-rc.3, except for performance benchmarks which used MoDLE v1.0.0-rc.2.

Code used for the data analysis is available on GitHub at https://github.com/paulsengroup/2021-modle-paper-001-data-analysis [45] and is archived on Zenodo at https://doi.org/10.5281/zenodo.6425490 [46].
Data produced by running the runme.sh script from the data analysis core repository, including simulated contact matrices in cooler format were archived on Zenodo (DOI: 10.5281/zenodo.6448374).

Reference genome assembly and assembly report for GCF_000001405.39_GRCh38.p13 [48] was downloaded through the NCBI FTP server [67].

ChIP-seq data for the following accession numbers were downloaded from the ENCODE portal [50,51]: ENCFF255FRL [52], ENCFF473IZV [53], ENCFF821AQO [54] and ENCFF913JGA [55].

H1-hESC Micro-C data in multi-resolution cooler format (4DNFI9GMP2J8 [62]) were downloaded from the 4DNucleome Data Portal [63].

The frequency matrix in MEME format for the CTCF motif (MA0139.1) was downloaded from JASPAR 2022 CORE non-redundant database [49].

Funding

JP acknowledges funding from the Norwegian Research Council (project 324137).

Authors' contributions

JP and RR conceived and designed the study; RR developed the software and the analysis code; RR performed benchmarking analysis and genome wide parameter optimization; JP performed comparison analyses of MoDLE with OpenMM and Micro-C data; JP and RR wrote the original draft; AM helped with the manuscript revising; VK, AM and TR provided critical feedback and assisted in the improvement of the tool and analyses; JP supervised the project. All authors read and approved the final manuscript.

Acknowledgements

The simulations were performed on resources provided by Sigma2 - the National Infrastructure for High Performance Computing and Data Storage in Norway.
We thank the ENCODE Consortium and the labs of Peter Park and Job Dekker for contributing with ENCODE data used as part of this study.

References

1. Braccioli L, de Wit E. CTCF: a Swiss-army knife for genome organization and transcription regulation. Essays Biochem. 2019;63:157–65.

2. Razin SV, Gavrilov AA, Vassetzky YS, Ulianov SV. Topologically-associating domains: gene warehouses adapted to serve transcriptional regulation. Transcription. 2016;7:84–90.

3. Arnould C, Rocher V, Finoux A-L, Clouaire T, Li K, Zhou F, et al. Loop extrusion as a mechanism for formation of DNA damage repair foci. Nature. 2021;590:660–5.

4. Peters J-M. How DNA loop extrusion mediated by cohesin enables V(D)J recombination. Curr Opin Cell Biol. 2021;70:75–83.

597 5. Goloborodko A, Marko JF, Mirny LA. Chromosome Compaction by Active Loop Extrusion.
598 Biophys J. 2016;110:2162–8.

599 6. Ganji M, Shaltiel IA, Bisht S, Kim E, Kalichava A, Haering CH, et al. Real-time imaging of
600 DNA loop extrusion by condensin. Science. 2018;360:102–5.

601 7. Golfier S, Quail T, Kimura H, Brugués J. Cohesin and condensin extrude DNA loops in a
602 cell cycle-dependent manner. Elife [Internet]. 2020;9. Available from:
603 http://dx.doi.org/10.7554/eLife.53885

604 8. Lieberman-Aiden E, van Berkum NL, Williams L, Imakaev M, Ragoczy T, Telling A, et al.
605 Comprehensive mapping of long-range interactions reveals folding principles of the human
606 genome. Science. 2009;326:289–93.

607 9. Hsieh T-HS, Weiner A, Lajoie B, Dekker J, Friedman N, Rando OJ. Mapping Nucleosome
608 Resolution Chromosome Folding in Yeast by Micro-C. Cell. 2015;162:108–19.

609 10. Krietenstein N, Abraham S, Venev SV, Abdennur N, Gibcus J, Hsieh T-HS, et al.
610 Ultrastructural Details of Mammalian Chromosome Architecture. Mol Cell. 2020;78:554–
611 65.e7.

612 11. Sanborn AL, Rao SSP, Huang S-C, Durand NC, Huntley MH, Jewett AI, et al. Chromatin
613 extrusion explains key features of loop and domain formation in wild-type and engineered
614 genomes. Proc Natl Acad Sci U S A. 2015;112:E6456–65.

615 12. Fudenberg G, Imakaev M, Lu C, Goloborodko A, Abdennur N, Mirny LA. Formation of
616 Chromosomal Domains by Loop Extrusion. Cell Rep. 2016;15:2038–49.

617 13. Brandão HB, Ren Z, Karaboja X, Mirny LA, Wang X. DNA-loop-extruding SMC
618 complexes can traverse one another in vivo. Nat Struct Mol Biol. 2021;28:642–51.

619 14. Lupiáñez DG, Spielmann M, Mundlos S. Breaking TADs: How Alterations of Chromatin
620 Domains Result in Disease. Trends Genet. 2016;32:225–37.

621 15. Alipour E, Marko JF. Self-organization of domain structures by DNA-loop-extruding
622 enzymes. Nucleic Acids Res. 2012;40:11202–12.

623 16. Bauer BW, Davidson IF, Canena D, Wutz G, Tang W, Litos G, et al. Cohesin mediates
624 DNA loop extrusion by a “swing and clamp” mechanism. Cell. 2021;184:5448–64.e22.

625 17. Golov AK, Golova AV, Gavrilov AA, Razin SV. Sensitivity of cohesin-chromatin
626 association to high-salt treatment corroborates non-topological mode of loop extrusion.
627 Epigenetics Chromatin. 2021;14:36.

628 18. Pradhan B, Barth R, Kim E, Davidson IF, Bauer B, van Laar T, et al. SMC complexes
629 can traverse physical roadblocks bigger than their ring size [Internet]. bioRxiv. 2021 [cited
630 2022 Apr 11]. p. 2021.07.15.452501. Available from:
631 https://www.biorxiv.org/content/10.1101/2021.07.15.452501v1.abstract

632 19. Rao SSP, Huntley MH, Durand NC, Stamenova EK, Bochkov ID, Robinson JT, et al. A
633 3D map of the human genome at kilobase resolution reveals principles of chromatin looping.
634 Cell. 2014;159:1665–80.

635 20. Vian L, Pękowska A, Rao SSP, Kieffer-Kwon K-R, Jung S, Baranello L, et al. The
636 Energetics and Physiological Impact of Cohesin Extrusion. Cell. 2018;173:1165–78.e20.

637 21. Tedeschi A, Wutz G, Huet S, Jaritz M, Wuensche A, Schirghuber E, et al. Wapl is an
638 essential regulator of chromatin structure and chromosome segregation. Nature.
639 2013;501:564–8.

640 22. Haarhuis JHI, van der Weide RH, Blomen VA, Yáñez-Cuna JO, Amendola M, van Ruiten
641 MS, et al. The Cohesin Release Factor WAPL Restricts Chromatin Loop Extension. Cell.
642 2017;169:693–707.e14.

643 23. Nakamura R, Motai Y, Kumagai M, Wike CL, Nishiyama H, Nakatani Y, et al. CTCF
644 looping is established during gastrulation in medaka embryos. Genome Res. 2021;31:968–

645 80.

646 24. Wutz G, Várnai C, Nagasaka K, Cisneros DA, Stocsits RR, Tang W, et al. Topologically
647 associating domains and chromatin loops depend on cohesin and are regulated by CTCF,
648 WAPL, and PDS5 proteins. EMBO J. 2017;36:3573–99.

649 25. Rao SSP, Huang S-C, Glenn St Hilaire B, Engreitz JM, Perez EM, Kieffer-Kwon K-R, et
650 al. Cohesin Loss Eliminates All Loop Domains. Cell. 2017;171:305–20.e24.

651 26. Liu NQ, Magnitov M, Schijns M, van Schaik T, van der Weide RH, Teunissen H, et al.
652 Rapid depletion of CTCF and cohesin proteins reveals dynamic features of chromosome
653 architecture [Internet]. bioRxiv. 2021 [cited 2022 Apr 11]. p. 2021.08.27.457977. Available
654 from: https://www.biorxiv.org/content/10.1101/2021.08.27.457977v1.full

655 27. Nora EP, Goloborodko A, Valton A-L, Gibcus JH, Uebersohn A, Abdennur N, et al.
656 Targeted Degradation of CTCF Decouples Local Insulation of Chromosome Domains from
657 Genomic Compartmentalization. Cell. 2017;169:930–44.e22.

658 28. Barbieri M, Chotalia M, Fraser J, Lavitas L-M, Dostie J, Pombo A, et al. Complexity of
659 chromatin folding is captured by the strings and binders switch model. Proc Natl Acad Sci U
660 S A. 2012;109:16173–8.

661 29. Naumova N, Imakaev M, Fudenberg G, Zhan Y, Lajoie BR, Mirny LA, et al. Organization
662 of the mitotic chromosome. Science. 2013;342:948–53.

663 30. Jost D, Carrivain P, Cavalli G, Vaillant C. Modeling epigenome folding: formation and
664 dynamics of topologically associated chromatin domains. Nucleic Acids Res. 2014;42:9553–
665 61.

666 31. Benedetti F, Dorier J, Burnier Y, Stasiak A. Models that include supercoiling of
667 topological domains reproduce several known features of interphase chromosomes. Nucleic
668 Acids Res. 2014;42:2848–55.

669 32. Goloborodko A, Imakaev MV, Marko JF, Mirny L. Compaction and segregation of sister
670 chromatids via active loop extrusion. Elife [Internet]. 2016;5. Available from:
671 http://dx.doi.org/10.7554/eLife.14864

672 33. Anderson JA, Glaser J, Glotzer SC. HOOMD-blue: A Python package for high-
673 performance molecular dynamics and hard particle Monte Carlo simulations. Comput Mater
674 Sci. 2020;173:109363.

675 34. Schwarzer W, Abdennur N, Goloborodko A, Pekowska A, Fudenberg G, Loe-Mie Y, et
676 al. Two independent modes of chromatin organization revealed by cohesin removal. Nature.
677 2017;551:51–6.

678 35. Nuebler J, Fudenberg G, Imakaev M, Abdennur N, Mirny LA. Chromatin organization by
679 an interplay of loop extrusion and compartmental segregation. Proc Natl Acad Sci U S A.
680 2018;115:E6697–706.

681 36. Banigan EJ, van den Berg AA, Brandão HB, Marko JF, Mirny LA. Chromosome
682 organization by one-sided and two-sided loop extrusion. Elife [Internet]. 2020;9. Available
683 from: http://dx.doi.org/10.7554/eLife.53558

684 37. Eastman P, Swails J, Chodera JD, McGibbon RT, Zhao Y, Beauchamp KA, et al.
685 OpenMM 7: Rapid development of high performance algorithms for molecular dynamics.
686 PLoS Comput Biol. 2017;13:e1005659.

687 38. Buckle A, Brackley CA, Boyle S, Marenduzzo D, Gilbert N. Polymer Simulations of
688 Heteromorphic Chromatin Predict the 3D Folding of Complex Genomic Loci. Mol Cell.
689 2018;72:786–97.e11.

690 39. Zhang S, Übelmesser N, Josipovic N, Forte G, Slotman JA, Chiang M, et al. RNA
691 polymerase II is required for spatial chromatin reorganization following exit from mitosis. Sci
692 Adv. 2021;7:eabg8205.

40. Abdennur N, Mirny LA. Cooler: scalable storage for Hi-C data and other genomically labeled arrays. Bioinformatics. 2020;36:311–6.

41. Gabriele M, Brandão HB, Grosse-Holz S, Jha A, Dailey GM, Cattoglio C, et al. Dynamics of CTCF and cohesin mediated chromatin looping revealed by live-cell imaging [Internet]. bioRxiv. 2021 [cited 2022 Apr 11]. p. 2021.12.12.472242. Available from: https://www.biorxiv.org/content/10.1101/2021.12.12.472242v1

42. Hansen AS, Pustova I, Cattoglio C, Tjian R, Darzacq X. CTCF and cohesin regulate chromatin loop stability with distinct dynamics. Elife [Internet]. 2017;6. Available from: http://dx.doi.org/10.7554/eLife.25776

43. Di Stefano M, Paulsen J, Jost D, Marti-Renom MA. 4D nucleome modeling. Curr Opin Genet Dev. 2021;67:25–32.

44. Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. Nat Biotechnol. 2017;35:316–9.

45. 2021-modle-paper-001-data-analysis: Data analysis code for the first paper about MoDLE (preprint available soon) [Internet]. Github; [cited 2022 Apr 11]. Available from: https://github.com/paulsengroup/2021-modle-paper-001-data-analysis

46. Rossini R, Kumar V, Mathelier A, Rognes T, Paulsen J. Data analysis code for the first MoDLE paper [Internet]. 2022. Available from: https://zenodo.org/record/6425490

47. Bailey TL, Gribskov M. Combining evidence using p-values: application to sequence homology searches. Bioinformatics. 1998;14:48–54.

48. GRCh38.p13 - hg38 - Genome - Assembly - NCBI [Internet]. [cited 2022 Apr 12]. Available from: https://identifiers.org/assembly:GCF_000001405.39

49. Castro-Mondragon JA, Riudavets-Puig R, Rauluseviciute I, Lemma RB, Turchi L, Blanc-Mathieu R, et al. JASPAR 2022: the 9th release of the open-access database of

717 transcription factor binding profiles. Nucleic Acids Res. 2022;50:D165–73.

718 50. ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the
719 human genome. Nature. 2012;489:57–74.

720 51. Davis CA, Hitz BC, Sloan CA, Chan ET, Davidson JM, Gabdank I, et al. The
721 Encyclopedia of DNA elements (ENCODE): data portal update. Nucleic Acids Res.
722 2018;46:D794–801.

723 52. ENCFF255FRL – ENCODE [Internet]. [cited 2022 Apr 12]. Available from:
724 https://identifiers.org/encode:ENCFF255FRL

725 53. ENCFF473IZV – ENCODE [Internet]. [cited 2022 Apr 12]. Available from:
726 https://identifiers.org/encode:ENCFF473IZV

727 54. ENCFF821AQO – ENCODE [Internet]. [cited 2022 Apr 12]. Available from:
728 https://identifiers.org/encode:ENCFF821AQO

729 55. Encff913jga – encode [Internet]. [cited 2022 Apr 12]. Available from:
730 https://identifiers.org/encode:ENCFF913JGA

731 56. Venev S, Abdennur N, Goloborodko A, Flyamer I, Fudenberg G, Nuebler J, et al.
732 open2c/cooltools: v0.5.1 [Internet]. 2022. Available from: https://zenodo.org/record/6324229

733 57. Kerpedjiev P, Abdennur N, Lekschas F, McCallum C, Dinkla K, Strobelt H, et al. HiGlass:
734 web-based visual exploration and analysis of genome interaction maps. Genome Biol.
735 2018;19:125.

736 58. Banigan EJ, Mirny LA. The interplay between asymmetric and symmetric DNA loop
737 extrusion. Elife [Internet]. 2020;9. Available from: http://dx.doi.org/10.7554/eLife.63528

738 59. Imakaev M, Goloborodko A, hbbrandao. mirnylab/polychrom: v0.1.0 [Internet]. 2019.
739 Available from: https://zenodo.org/record/3579473

60. Yoon S, Chandra A, Vahedi G. Stripenn detects architectural stripes from chromatin conformation data using computer vision. Nat Commun. 2022;13:1602.

61. openmm [Internet]. Github; [cited 2022 Apr 11]. Available from: https://github.com/openmm/openmm/issues/3267

62. 4DNFI9GMP2J8.Mcool – 4DN data portal [Internet]. [cited 2022 Apr 12]. Available from: https://identifiers.org/4dn:4DNFI9GMP2J8

63. Reiff SB, Schroeder AJ, Kirli K, Cosolo A, Bakker C, Mercado L, et al. The 4D Nucleome Data Portal: a resource for searching and visualizing curated nucleomics data [Internet]. bioRxiv. 2021 [cited 2022 Apr 11]. p. 2021.10.14.464435. Available from: https://www.biorxiv.org/content/10.1101/2021.10.14.464435v1

64. scikit-optimize: Sequential model-based optimization with a `scipy.optimize` interface [Internet]. Github; [cited 2022 Apr 11]. Available from: https://github.com/scikit-optimize/scikit-optimize

65. modle: High-performance stochastic modeling of DNA loop extrusion interactions [Internet]. Github; [cited 2022 Apr 11]. Available from: https://github.com/paulsengroup/modle

66. Rossini R, Vipin K, Mathelier A, Rognes T, Paulsen J. MoDLE: High-performance stochastic modeling of DNA loop extrusion interactions [Internet]. 2022. Available from: https://zenodo.org/record/6424890

67. O'Leary NA, Wright MW, Brister JR, Ciufo S, Haddad D, McVeigh R, et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. Nucleic Acids Res. 2016;44:D733–45.

Supplementary material

- Additional file 1 (pdf): Supplementary text. Detailed description of MoDLE's underlying simulation model and implementation.
- Additional file 2 (pdf): Supplementary figures. Supplementary Figures 1-2.
- Additional file 3 (pdf): Supplementary tables. Supplementary Tables 1-2.