Gene Regulation In Prokaryotes Manual

Nicolae Radu Zabet[1,2,3,*] and Boris Adryan[1,2]

[1]Cambridge Systems Biology Centre, University of Cambridge, Tennis Court Road, Cambridge CB2 1QR, UK

[2]Department of Genetics, University of Cambridge, Downing Street, Cambridge CB2 3EH, UK

[*]Email: n.r.zabet@gen.cam.ac.uk

Contents

1 System Requirements

GRiP (Gene Regulation in Prokaryotes) is an asynchronous, event driven, hybrid model of the facilitated diffusion mechanism in prokaryotic systems. This software is implemented in Sun Java 1.6, which makes it widely available. The system requirements are

1. Java Runtime Environment 1.6
2. At least 3GB free RAM

The application, uses in most cases less than 2GB of RAM, but we preallocate more to cover any unexpected case. The user has the possibility to change this, when calling the `.jar` files (see below).

2 Interfaces

GRiP comes in three interfaces: (*i*) command line, (*ii*) graphical user interface and (*iii*) source code. Since we make the Java source code available, anyone can modify the program to suit various situations, specific to the user. However, the simplest usage of this application does not require knowledge of programming, but rather starting the application graphical or command line interfaces and providing a correct set of parameters for the system.

2.1 Command Line Interface (CLI)

The Command Line Interface (CLI) can be run from the terminal and, thus, can be easily used on Linux or Mac or Windows operating systems. To start the simulation, the following command needs to by typed in the terminal

```
java -Xms32m -Xmx3072m -jar gripCLI.jar PARAMETERS_FILE
 NUMBER_OF_INTERMEDIARY_POINTS  BACKUP_AFTER STOP_AFTER_SAVE
```

The first parameter (`PARAMETERS_FILE`) represents the parameters file and, although optional, it is highly recommended to be specified. The details of the parameters input file are given in section 3 and its format in section 3.1.1.

In addition, the user can specify the number of intermediary points in the simulation (`NUMBER_OF_INTERME` which can be used to track progress through the current simulation. The simulation progress (the percentage of completed simulation) is printed as a function of the number of intermediary points.

After the simulation reached an intermediary point, GRiP checks whether a backup was performed in the last `BACKUP_AFTER` seconds of simulation. If this value is not specified in the command, then GRiP uses the default value of 1 second. For long time simulations, this default value leads to a backup being performed after each intermediary point. Note that setting `BACKUP_AFTER` equal to -1 results in no backup being created.

Finally, the last parameter (`STOP_AFTER_SAVE`) is a boolean variable, which indicates whether the simulation should be stopped after an intermediary backup was performed. This is useful when implementing a checkpoint mechanism for long time simulations.

Note that we asked the system to allocate between 32MB and 3GB of RAM. The user can change the maximum used memory using different values. For smaller systems, the user can start the application with a lower maximum value, such as 2GB of RAM.

An example of this command is the following:

```
java -Xms32m -Xmx2048m -jar gripCLI.jar params.grp 100 false
```

2.2 Graphical User Interface (GUI)

GRiP comes also with a Graphical User Interface (GUI), which can be started from the terminal by typing the following command

```
java -Xms32m -Xmx3072m -jar gripGUI.jar
```

As in the case of the command line interface, the user can start the application, with less than `3GB` of RAM.

```
java -Xms32m -Xmx2048m -jar gripGUI.jar
```

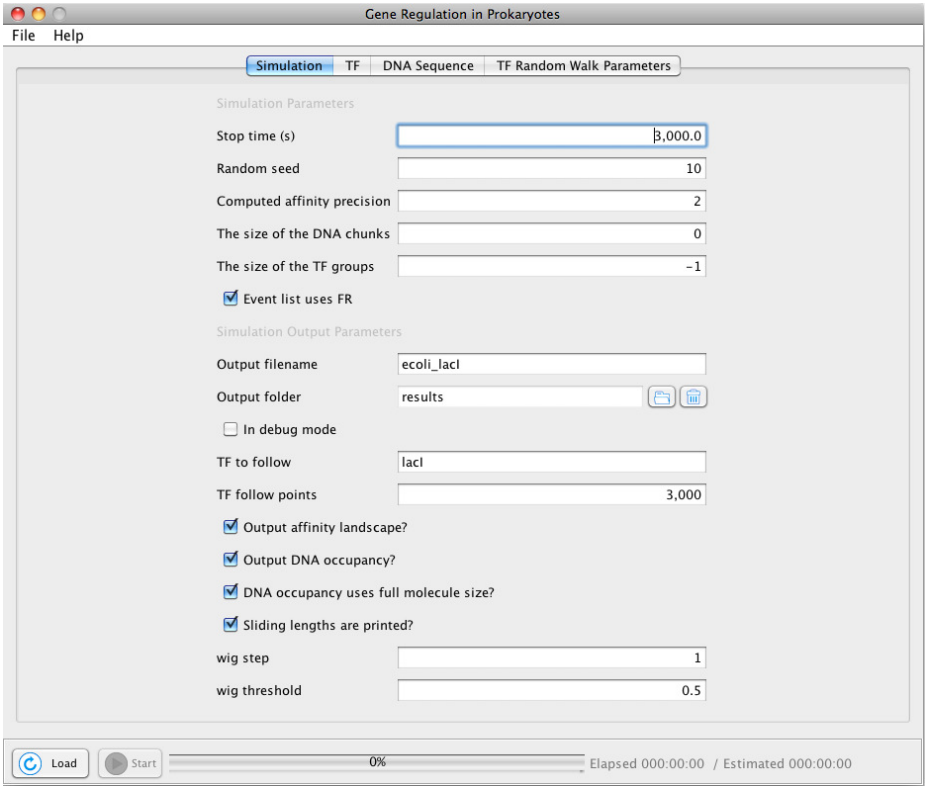

Figure 1: *The Graphical User Interface.* On the top of the window there is a menu, which allows the user to perform several action on the current project. Under the menu, a tabbed panel area contains a list of all parameters and allows editing of the current (preloaded) parameters. In the bottom of the interface the simulation can be loaded and then started.

The GUI contains three regions: (*i*) menu, (*ii*) parameters input and (*iii*) simulation area (see Figure 1). The first region, the menu contains the following options:

1. **File** menu contains four actions which can be performed on the current project, namely:
   (a) *Open* option opens a parameters file (`.grp`) and updates the fields in the interface.
   (b) *Save* option saves current parameters into a parameters file (`.grp`). If this was not saved previously, the user is requested to select a new name for the file.
   (c) *Save as* option saves current parameters into a new parameters file (`.grp`).
   (d) *Close* option closes the application.
2. **Help** can specify informations about the application
   (a) *About* provides information on the application, the version, the creators and the URL address.

Below the menu, there is the parameters input area, where the input parameters values are specified and edited. When the mouse is moved over the text field or checkbox it will generate a Tool Tip with the description of that field. In the case of file choosers, the mouse over the browse button will generate a Tool Tip with the description of the field, while over the non-editable text field a Tool Tip with the full path of the file. The parameters in the setup area are grouped in six section, each in a individual tab. The description of the input parameters is presented in section 3.

Finally, at the bottom of the GUI we find the simulation area. To start the simulation the user needs to first load the model, which is achieved by clicking the 'Load' button.

Once the model was loaded, the user does not have access to the parameters input area, but they can see the output of the status file; see Figure 2. Next, the user can unload the model by clicking 'Unload' button (to change the parameters) or start the simulation by clicking 'Start' button. The simulation progress is presented in the bottom middle progress bar, and the elapsed and estimated time of the simulation are displayed in the bottom right area.

The number of intermediary points equals the number of cell seconds (the biological time) the system simulates, but it cannot be lower than 100. This number of intermediary points also determines how fine is the update on the progress bar. Note that when the stop time is set to zero, the simulator will not update the progress bar, since the stop time is not known, i.e., the simulator will stop once all the target sites have been reached at least once.

When simulating, the user has the option to pause the simulation, by clicking the 'Pause' button. The pause will be effective only once the simulator reached the next intermediary point, which may not be instantaneous. Once the system is in pause mode, the user has the option of unloading the model (and, consequently, stop the simulation), or resume the simulation by clicking 'Start' button again.

3 Input

3.1 Input Parameters

There are four groups of parameters which represent the inputs of the simulation. Next we will list these four groups and the parameters belonging in each group. Note that for each parameter

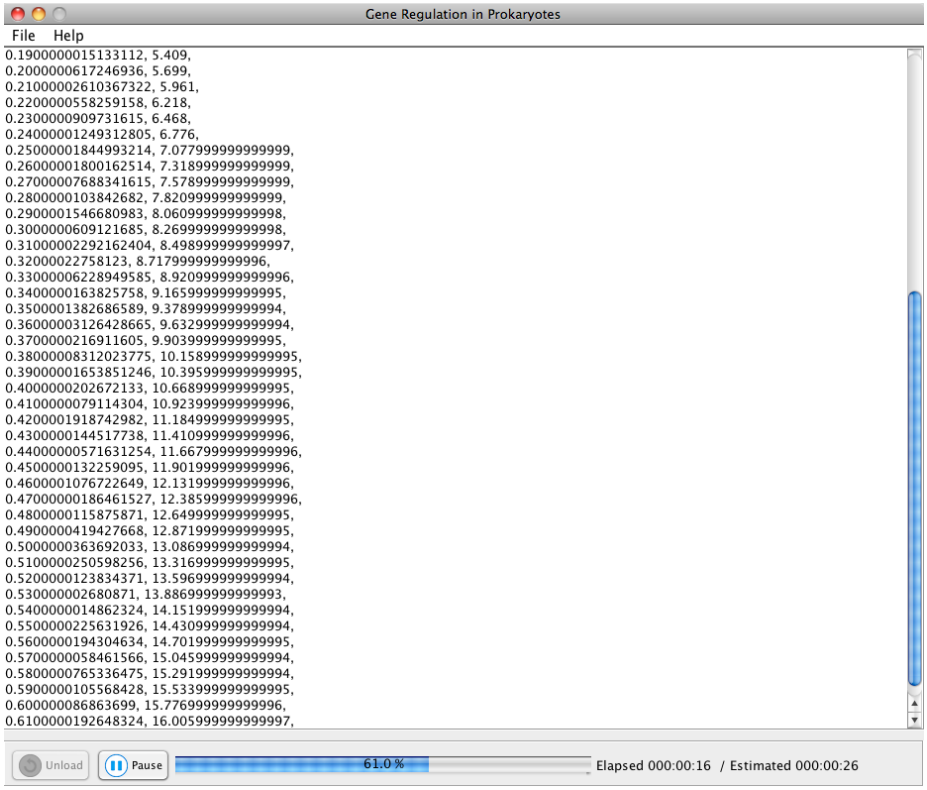

Figure 2: *The Graphical User Interface - model loaded.* The middle are of the GUI displays the output from the status file; see section 4.1.

we give a short description and then present the name of the parameter as specified in the `.grp` file with its default value. The name in bold characters of the parameter represents the label form the GUI.

1. **Simulation Parameters**

   (a) **Stop time** represents the length of the simulation measured in seconds. If this is set to 0 or a negative value and the simulator contains target sites that can be reached, then the system will run until all target sites are reached at least once.
   (`STOP_TIME = 1.0;`).

   (b) **Ensamble size** represents the number of independent replicate simulations to be performed (the values are non-zero natural numbers). The 1D statistics will be ensamble averaged. The final occupancy will represent the ensamble average of the occupancy bias. When the ensamble size is higher than 1 then the user cannot follow the TFs any more.
   (`ENSAMBLE_SIZE = 1;`)

   (c) **Random seed** represents the seed of the random number generator. Use 0 to get a different behaviour each time or a different number to get the same behaviour for multiple simulations with same set of parameters.
   (`RANDOM_SEED = 0;`)

   (d) **Computed affinity precision** represents the number of decimals when computing the affinity landscape.
   (`COMPUTED_AFFINITY_PRECISION = 2;`)

   (e) **The size of the DNA sector** represents the size of the DNA sector. Breaking the DNA into sectors increases the speed at which empty spots on the DNA are located. Put 0 for autoselect.
   (`DNA_SECTOR_SIZE = 0;`)

   (f) **The size of the event list subgroups** represents the size of the sub-lists when the event list is broken into sub-lists. This is highly recommended for Direct Method and should not be used for First reaction Method. Put 0 for autoselect.
   (`EVENT_LIST_SUBGROUP_SIZE = -1;`)

   (g) **Event list uses FR** is true if the one dimensional event list is implemented using Gillespie's First Reaction method or false if Gillespie's Dirtect Method is used (Gillespie, 1976, 1977, 2007).
   (`EVENT_LIST_USES_FR = true;`)

2. **Output Parameters**

   (a) **Output folder** represents the folder where the result files will be saved.
   (`OUTPUT_FOLDER = "results";`)

   (b) **Output filename** represents the filename where the output results will be saved. Extension will be automatically added at the end. If this is blank then a random unique name will be generated. If the file exists then a unique number will be added to the end in order to avoid overwriting data files.
   (`OUTPUT_FILENAME = "ecoli_lacI";`)

(c) **Intermediary results generated after (s)** represents the time interval in seconds after which intermediary results will be printed. If zero is used then no intermediary results will be produced.
(`PRINT_INTERMEDIARY_RESULTS_AFTER = 0;`)

(d) **Print final occupancy** is true if the occupancy at the end of the simulation is printed and false otherwise.
(`PRINT_FINAL_OCCUPANCY = false;`)

(e) **In debug mode?** is true if the simulation is in debug mode (prints all actions to the status file) and false otherwise. For long and extensive simulations this parameter should be set to false.
(`DEBUG_MODE = false;`)

(f) **TF to follow** specifies the TF species that followed through the simulations. The simulator will print information about the number of free molecules, the proportion of free molecules, the number of fee sites and the proportion of free sites. In addition, if there are less than 10 molecules of that type, the position of each molecule on the DNA will be recorded in the file. Multiple TF species are separated by comma.
(`OUTPUT_TF = "";`)

(g) **TF follow points** represents the number of intermediary time points at which the dynamic behaviour of the TF species that are followed is recorded.
(`OUTPUT_TF_POINTS = 1;`)

(h) **Follow target site occupancy?** is true if the simulator will output the dynamic behaviour of the target site occupancy.
(`FOLLOW_TS = false;`)

(i) **Output affinity landscape?** is true if the simulator will output the average waiting time profile at the end of the simulation.
(`OUTPUT_AFFINITY_LANDSCAPE = true;`)

(j) **Output binding energy?** is true if the simulator will output the binding energy instead of the average waiting time when saving the affinity landscape at the end of the simulation.
(`OUTPUT_BINDING_ENERGY = false;`)

(k) **Output DNA occupancy?** is true if the simulator will output the DNA occupancy bias at the end of the simulation.
(`OUTPUT_DNA_OCCUPANCY = true;`)

(l) **DNA occupancy uses full molecule size?** is true if a bound molecule will affect the DNA occupancy bias of the entire cover area of the DNA and is false only the first binding position of the molecule is considered when computing the DNA occupancy bias.
(`DNA_OCCUPANCY_FULL_MOLECULE_SIZE = false;`)

(m) **Sliding lengths are printed?** is true if the simulator will print all recorded sliding lengths.
(`OUTPUT_SLIDING_LENGTHS = false;`)

(n) **wig step** represents the value of the step in a fixed step wig file, used for the occupancy output.
(`WIG_STEP = 1;`)

(o) **wig threshold** represents the threshold (as percentage of the highest peak) for discarding peaks in wig files. Use -1 for auto-select and 0 for no threshold.
(`WIG_THRESHOLD = 0.0;`)

3. **TF** - the parameters associated with the TF species

   (a) **Load TFs**

      i. **TF file** represents the `.csv` file which stores the TF data. Note that this needs to contain either the absolute or the relative path to the file. The details on the file structure (columns and significance) are detailed in section 3.2.2. The TF species are mandatory for the simulator. If none is specified, then the simulator will generate randomly TF as described below.
      (`TF_FILE = "biodata/TF_5lac_10000nc.csv";`).

      ii. **TF cooperativity file** represents the `.csv` file which stores the TF cooperativity data. As well as in the case of the TF file, this parameter needs to contain either the absolute or the relative path to the file. The details on the file structure (columns and significance) are detailed in section 3.2.3. Defining this file is optional in the sense that the simulator does not need to have specified cooperative behaviour between TF species.
      (`TF_COOPERATIVITY_FILE = "";`).

      iii. **Target sites file** represents the file which stores the target sites for the. As well as in the case of the TF file, this parameter needs to contain either the absolute or the relative path to the file. The details on the file structure are detailed in section 3.2.4. Defining this file is optional in the sense that the simulator does not need to have specified cooperative behaviour between TF species.
      (`TS_FILE = "";`).

   (b) **TFs General Parameters**

      i. **TF read in both directions?** is true if TFs read in both directions and false otherwise. If TF species read in both direction, then the affinity of the TF for the DNA also depends on the orientation of the TF molecule on the DNA.
      (`TF_READ_IN_BOTH_DIRECTIONS = false;`).

      ii. **Sliding and hopping affects TF association rate?** is true if sliding and hopping affects the association rate between TF molecules and DNA and false otherwise.
      (`SLIDING_AND_HOPPING_AFFECTS_TF_ASSOC_RATE = false;`)

   (c) **Randomly Generated TFs** contains the parameters used when TF species were not loaded from a file and, consequently, they need to be generated randomly.

      i. **Number of TF species** represents the number of TF species that need to be generated randomly.
      (`TF_SPECIES_COUNT = 2;`)

      ii. **Minimum TF copy number** specifies the minimum TF copy number for a randomly generated species.
      (`TF_COPY_NUMBER_MIN = 1000;`)

      iii. **Maximum TF copy number** represents the maximum TF copy number for a randomly generated species.
      (`TF_COPY_NUMBER_MAX = 1000;`)

iv. **Minimum length of DBD** represents the minimum length of the DNA Binding Domain of TFs. This is measured in base pairs.
(`TF_DBD_LENGTH_MIN = 6;`)

v. **Maximum length of DBD** represents the maximum length of the DNA Binding Domain of TFs. This is measured in base pairs.
(`TF_DBD_LENGTH_MAX = 18;`)

vi. **Energy penalty for a nucleotide mismatch.** represents the energy penalty for a nucleotide mismatch between DBD and current position (Gerland et al., 2002). This is ε from (Gerland et al., 2002) and is measured in K_BT.
(`TF_ES = 2;` (Gerland et al., 2002))

vii. **TF size left** represents the number of base pairs covered to the left of the DBD by a TF molecule bound to the DNA.
(`TF_SIZE_LEFT = 0;`)

viii. **TF size right** represents the number of base pairs covered to the right of the DBD by a TF molecule bound to the DNA.
(`TF_SIZE_RIGHT = 0;`)

ix. **TF association rate** represents the association rate between TF molecules and DNA and is measured in s^{-1}.
(`TF_ASSOC_RATE = 1800;` (Elf et al., 2007))

x. **TF prebound proportion** represents the proportion of TF molecules that are already bound when the simulation starts.
(`TF_PREBOUND_PROPORTION = 0.9;` (Elf et al., 2007))

xi. **TF prebound to highest affinity sites** is true if the TF is prebound to the highest affinity sites when the simulation starts.
(`TF_PREBOUND_TO_HIGHEST_AFFINITY = true;`)

4. **DNA Sequence** - the parameters associated with the DNA sequence

   (a) **Load DNA Sequence**

      i. **DNA sequence file** represents the fasta file which stores the DNA sequence. A description of this file can be found in section 3.2.1.
      (`DNA_SEQUENCE_FILE = "biodata/ecoli_k12.fasta";`).

   (b) **Randomly Generate DNA Sequence** consists of the parameters used when the DNA sequence could not be load from a fasta file and, consequently, needs to be generated randomly.

      i. **DNA length** represents the length of the DNA measured in base pairs.
      (`DNA_LENGTH = 4600000;` (Riley et al., 2006))

      ii. **DNA proportion of A** specifies the proportion of adenine (A) in the randomly generated DNA.
      (`DNA_PROPORTION_OF_A = 0.246;` (Weindl et al., 2007))

      iii. **DNA proportion of T** specifies the proportion of thymine (T) in the randomly generated DNA.
      (`DNA_PROPORTION_OF_T = 0.246;` (Weindl et al., 2007))

      iv. **DNA proportion of C** specifies the proportion of cytosine (C) in the randomly generated DNA.
      (`DNA_PROPORTION_OF_C = 0.254;` (Weindl et al., 2007))

v. **DNA proportion of G** specifies the proportion of guanine (G) in the randomly generated DNA.
(`DNA_PROPORTION_OF_G = 0.254;` (Weindl et al., 2007))

vi. **DNA boundary condition** specifies the boundary condition of the DNA (absorbing/reflexive/periodic).
(`DNA_BOUNDARY_CONDITION = "reflexive";`)

5. **TF Random Walk Parameters** - the parameters associated with the TF random walk.

   (a) **Simulation Assumptions**

   i. **Check DNA occupancy on molecule binding?** is true if the simulator will check the DNA occupancy before binding and false otherwise.
   (`CHECK_OCCUPANCY_ON_BINDING = true;`).

   ii. **Check DNA occupancy on molecule sliding?** is true if the simulator will check the DNA occupancy before sliding and false otherwise.
   (`CHECK_OCCUPANCY_ON_SLIDING = true;`).

   iii. **Check DNA occupancy on molecule rebinding?** is true if the simulator will check the DNA occupancy before re-binding and false otherwise.
   (`CHECK_OCCUPANCY_ON_REBINDING = true;`).

   (b) **Default Parameters for TFs Random Walk** contains the parameters used when the TF species are generated randomly or when this information misses for certain TF species.

   i. **TF is immobile on DNA** is true if the TF after binding to the DNA becomes immobile and stays in the same position throughout the entire simulation.
   (`TF_IS_IMMOBILE =false;`)

   ii. **TF unbinding probability** represents the probability that a TF unbinds during next event.
   (`TF_UNBINDING_PROBABILITY = 0.001474111;` (Elf et al., 2007))

   iii. **TF slide left probability** represents the probability that a TF slides left during next event.
   (`TF_SLIDE_LEFT_PROBABILITY = 0.4992629;` (Elf et al., 2007))

   iv. **TF slide right probability** represents the probability that a TF slides right during next event.
   (`TF_SLIDE_RIGHT_PROBABILITY = 0.4992629;` (Elf et al., 2007))

   v. **TF jumping probability** represents the probability that a TF performs a jump when unbinding instead of returning to the DNA.
   (`TF_JUMPING_PROBABILITY = 0.1675;` (Wunderlich and Mirny, 2008))

   vi. **TF hop standard displacement** represents the standard displacement of a TF that unbinds and attempts to rebind correlated. Note that the displacement has a Gaussian distribution and is measured in base pairs.
   (`TF_HOP_STD_DISPLACEMENT = 1.0;` (Wunderlich and Mirny, 2008))

   vii. **Waiting time of a specific TF** represents the waiting time of a specific TF measured in seconds. This is $\tau_0 \cdot \exp(E_{ns})$ from (Gerland et al., 2002) and is measured in seconds.
   (`TF_NONSPECIFIC_WAITING_TIME = 0.3314193;` (Elf et al., 2007))

viii. **Left step size of the TF** represents the size of the step to the left measured in base pairs when the TF performs a left slide.
(`TF_STEP_LEFT_SIZE = 1;`)

ix. **Right step size of the TF** represents the size of the step to the right measured in base pairs when the TF performs a right slide.
(`TF_STEP_RIGHT_SIZE = 1;`)

x. **TF uncorrelated displacement size** represents the size of the uncorrelated displacement. This is measured in base pairs. If the hop distance is higher than the uncorrelated displacement size then the hop becomes a jump and the TF molecule needs to first release in the cytoplasm.
(`TF_UNCORRELATED_DISPLACEMENT_SIZE = 100;` (Wunderlich and Mirny, 2008))

xi. **TF stalls if blocked?** is true if the TF molecule stays at current position if cannot relocate trough a hop and false if it unbinds during an unsuccessful hopping event.
(`TF_STALLS_IF_BLOCKED = true;`)

xii. **TF collision unbind probability** represents the probability that if a TF molecule collides with another molecule on the DNA it will unbind.
(`TF_COLLISION_UNBIND_PROBABILITY = 0.0;`)

xiii. **TF affinity landscape roughness** represents the roughness of the affinity landscape. This is measured in K_BT and is specified for non-cognate species to generate a random non-specific affinity landscape.
(`TF_AFFINITY_LANDSCAPE_ROUGHNESS = 1.0;` (Blainey et al., 2009))

6. **3D-hop** - the parameters associated with the 3D-hop event.

   (a) **Contact Matrix**

   i. **Contact Matrix file** represents the file containing Hi-C data in `tsv` format as GInteractions (e.g. HiCExplorer `https://hicexplorer.readthedocs.io/en/latest/content/tools/hicConvertFormat.html#hicconvertformat`).
   (`HIC_CONTACT_MATRIX_FILE = "biodata/Kc167_hic_matrix.GInteractions.tsv";`).

   ii. **Bin width** represents the length of a segment of DNA according to the Hi-C data that is used to generate the Contact Matrix .
   (`BIN_WIDTH = 500;`).

   (b) **3D hopping event parameters**

   i. **Probability to stay in the microenvironment** represents the probability that a molecule will stay in the microenvironment and perform the 3D hop, otherwise it will dissociate into the nucleoplasm.
   (`PK_MICROENV = 0.15;`).

   ii. **Update time** represents the average time at which the Contact Matrix is updated to simulate a dynamic behaviour.
   (`IM_SIMULATION_TIME = 0.01;`).

3.1.1 Parameters File

The list of all input parameters is stored in a file called the parameters file. Each parameter from this file is specified using the following pattern.

```
PARAMETER_NAME = value;
```

where the `PARAMTER_NAME` is the name specified in parentheses in the previous subsection and the value can be either a number, a boolean (true/false) or a string. Note that in the case of a string, the value must be included within quotation mark. A few examples of parameters are:

```
STOP_TIME = 3000.0;
TF_STEP_RIGHT_SIZE = 1;
DEBUG_MODE = false;
DNA_SEQUENCE_FILE = "biodata/ecoli_k12.fasta";
```

Note that symbol `#` marks the start of comment line, which is not parsed by the application but can be useful in interpreting the meaning of the parameter when a user reads this file.

The user does not need to specify all parameters, but just the ones that differ from the default values. In the case a parameter is missing from the parameters file, the default value will be used.

The parameters file needs to be provided as input in both the CLI or GUI. Furthermore, when the simulator starts running, a version of the loaded parameters file will be stored in the folder specified by the **Output folder** parameter using a name which starts with the **Output filename** parameter followed by `_params.grp` suffix. If the **Output filename** is empty a new unique name will be generated with each simulation.

3.2 Input Files

The application can parse three types of file: (*i*) DNA sequence file, (*ii*) TF file and (*iii*) TF cooperativity file. Next, we will describe the structure of these files and how they are parsed.

3.2.1 DNA Sequence File

This is a generic fasta file, which consists of a DNA sequence to be loaded into the simulator. The sequence starts with `>` symbol followed by a short description of the sequence. This description can contain three elements: (*i*) the name of the chromosome (`chr`), (*ii*) the start position on the chromosome of the sequence (`start`) and (*iii*) the end position of the sequence on the chromosome (`end`). The format of this description is at follows:

```
>chr:start..end;
```

Note that the elements are delimited by `:` and `..` in the structure specified above.

Optionally, the user can specify one of the two other properties of the DNA sequence which are separated by a semicolon

```
subsequence = chrSub:startSub..endSub; boundary = "boundaryCondition"
```

If `subsequence` property is specified, the user has the possibility to consider in the simulation only a subsequence of the loaded sequence. Note that the parameters of the subsequence (start and end positions) are in absolute values and, thus, they have to be in the interval of the DNA region. In

addition, the user can specify the boundary condition for the simulation. There are three boundary conditions supported by the simulator, namely: `absorbing`, `reflexive` and `periodic`.

Starting from the second line, the sequence of the DNA is specified as a string containing the letters `A`, `T`, `C` and `G`. These can be specified in both lower or upper case, since the application will parse them correctly. The sequence ends with an end of file or a new `>` symbol. If the file contains multiple sequences only the last will be used in the simulations.

3.2.2 TF File

The TF file is a `.csv` file in which the first row lists the headers for each column and each subsequent row the values for a TF species.

There are twenty two mandatory columns which have the following headers:

1. **NAME** represents the name of the TF species.

2. **DBD** represents the DNA binding Domain of the TF species. For non-cognate species this field should be left empty and the affinity for the DNA will be generated randomly with a certain non-specific affinity (`ES`) and some variance (`AFFINITYLANDSCAPEROUGHNESS`). For cognate species the field must be specified and this can happen as either exact sequence or PFM, i.e., `"SEQ:sequence_definition"` or `"PFM:pfm_definition"`. For the method on computing the affinity landscape please consult the paper.

   (a) **SEQ** If there is one known high affinity site for a TF (high specificity) the DBD can be defined using the sequence of that site, e.g., `"SEQ:AATTGTNNNNNNNNNACAATT"`. This approach will use the mismatch energy method to compute the affinity landscape, in the sense that each mismatch between current position and the reference sequence results in a `ES` energetic penalty to the affinity; see (Gerland et al., 2002). Note that when the `N` base pair is used the simulator will disregard that nucleotide when computing the affinity landscape.

   (b) **PFM** If there are more than one known high affinity sites for a TF (lower specificity) the DBD can be defined using the PFM of that site, e.g.,

   ```
   "PFM:ENERGY:1.0:A=[3, 1, 5, 7, 3, 6, 4, 7, 1, 9, 8, 5, 4, 2];
   C=[1, 1, 2, 0, 0, 0, 1, 0, 8, 0, 0, 0, 0, 3];
   G=[4, 1, 1, 1, 0, 0, 4, 1, 0, 0, 1, 3, 0, 2];
   T=[1, 6, 1, 1, 6, 3, 0, 1, 0, 0, 0, 1, 5, 2]"
   ```

   There are four areas in this definition separated by `:`, namely: (*i*) the type of the DBD (`PFM` in this example), (*ii*) the type of PFM (it can be either `INFO` for Stormo (2000) or `ENERGY` for Berg and von Hippel (1987)), (*iii*) the pseudo-count term (ζ from main text, which ensures the computed binding energy is not $-\infty$) and (*iv*) the PFM of the motif. Note that all of this needs to be specified on a single line for correct parsing. For each of the four nucleotides (`A, T, C,` or `G`) the number of occurrences at each position in the high affinity sequences are specified. This is similar to the previous method, but instead of making a fixed penalty of `ES` a weighted one is used; see main text.

   (c) **PWM** This is similar to the PFM, but it allows the user to compute the PWM, and consequently, alternative methods can be tested (such as $\log_2$ instead of ln)

```
"PWM: A = [-0.23, -2.20, -2.20, -0.79, -2.20, -2.20, -2.20, -0.23, 0.13, -0.79];
C = [1.37, -0.29, -2.20, -2.20, -0.29, -2.20, 0.70, -0.29, 0.32, 0.70];
   G = [-2.20, -2.20, 1.37, 1.37, -2.20, -2.20, -0.29, -0.29, 0.70, -2.20];
   T = [-2.20, 0.92, -0.23, -0.79, 0.92, 1.05, 0.39, 0.39, -2.20, 0.39]"
```

(d) **LANDSCAPE** This indicates a file with already precomputed binding energies. The specification of this option is:
`LANDSCAPE:SIZE_DBM:IS_COGNATE:COL_POS:COL_ENERGY:COL_ENERGY_REVERSE:FILENAME:SKIP_LINES`
There are seven parameters, namely: (*i*) `SIZE_DBM` the size in base pairs of the DNA binding motif, (*ii*) `IS_COGNATE` a boolean indicating whether this is a cognate species or not, (*iii*) `COL_POS` the column in the file indicating the position of the line on the DNA, (*iv*) `COL_ENERGY` the column in the file indicating the binding energy at the current position, (*v*) `COL_ENERGY_REVERSE` the column in the file indicating the binding energy at the current position on the reverse strand (*vi*) `FILENAME` the file storing the binding energies and (*vii*) `SKIP_LINES` the number of lines to be skipped from the beginning of the file. The affinity landscape file has the same format as the one that is exported by GRiP; see section 4.2. An example of this option is:

```
LANDSCAPE:0:true:1:2:2:biodata/ecoli_lacI_affinity_landscape.wig:2
```

(e) **SEQS** This indicates a file with already precomputed binding energies for all possible DNA words. The specification of this option is:
`SEQS:SIZE_DBM:IS_COGNATE:DEFAULT_VALUE:FILENAME:SKIP_LINES`
There are six parameters, namely: (*i*) `SIZE_DBM` the size in base pairs of the DNA binding motif, (*ii*) `IS_COGNATE` a boolean indicating whether this is a cognate species or not, (*iii*) `DEFAULT_VALUE` the default value for binding energy if the DNA word is missing from the file (*iv*) `FILENAME` the file storing the binding energies and (*vii*) `SKIP_LINES` the number of lines to be skipped from the beginning of the file. The file specifying the binding energies for sequences is a file which contain on each line two elements separated by a comma: (*i*) the DNA word and (*ii*) the associated binding energy measured in K_BT.An example of this option is:

```
SEQS:8:false:60:biodata/suh_foldx_65536.txt:0
```

3. **ES** represents the energy penalty for a nucleotide mismatch between DBD and current position (Gerland et al., 2002). This is ε from (Gerland et al., 2002) and is measured in K_BT.

4. **COPYNUMBER** specifies the TF copy number for the TF species.

5. **SIZELEFT** represents the number of base pairs covered to the left of the DBD by a TF molecule bound to the DNA.

6. **SIZERIGHT** represents the number of base pairs covered to the right of the DBD by a TF molecule bound to the DNA.

7. **ASSOCRATE** represents the association rate between TF molecules and DNA.

8. **INITIALDROP** specifies the area where a TF molecule is first bound to the DNA. If this is unspecified, the TF molecules can bind initially anywhere on the DNA. The initial

drop region is defined similar to the description of the DNA sequence (see section 3.2.1), i.e., `"chr:start..end:prob"`. Note that DNA positions are indexed from 0. The last term represents the probability that a TF molecule will be dropped initially in this region. For example, `"K12:365552..365752:0.9"` means that `90%` of the TF molecules will be initially bound between positions `365552` and `365752` if there is space available, while the rest can be positioned anywhere on the DNA.

9. **UNBINDINGPROBABILITY** represents the probability that a TF unbinds during next event.

10. **SLIDELEFTPROBABILITY** represents the probability that a TF slides left during next event.

11. **SLIDERIGHTPROBABILITY** represents the probability that a TF slides right during next event.

12. **JUMPINGPROBABILITY** represents the probability that a TF performs a jump when unbound instead of returning to the DNA.

13. **HOPSTDDISPLACEMENT** represents the standard deviation of a TF that unbinds and attempts to rebind correlated. Note that the displacement is Gaussian distributed and centred around the original position. This is measured in base pairs.

14. **SPECIFICWAITINGTIME** represents the waiting time of a non-specific TF measured in seconds. This is $\tau_0 \cdot \exp(E_{ns})$ from (Gerland et al., 2002) and is measured in seconds.

15. **STEPLEFTSIZE** represents the size of the step to the left measured in base pairs when the TF performs a left slide.

16. **STEPRIGHTSIZE** represents the size of the step to the right measured in base pairs when the TF performs a right slide.

17. **UNCORRELATEDDISPLACEMENTSIZE** represents the size of the uncorrelated displacement and is measured in base pairs. If the hop distance is higher than the uncorrelated displacement size then the hop becomes a jump and the TF molecule needs to first release in the cytoplasm.

18. **STALLSIFBLOCKED** is true if the TF molecule stays at current position if cannot relocate trough a hop and false if it unbinds during an unsuccessful hopping event.

19. **COLLISIONUNBINDPROBABILITY** represents the probability that if a TF molecule collides with another molecule on the DNA it will unbind.

20. **AFFINITYLANDSCAPEROUGHNESS** represents the roughness of the affinity landscape. This is measured in K_BT and is specified for non-cognate species to generate a random non-specific affinity landscape.

21. **PREBOUNDPROPORTION** represents the proportion of TF molecules that are already bound when the simulation starts.

22. **PREBOUNDTOHIGHESTAFFINITY** is true if the TF is prebound to the highest affinity sites when the simulation starts.

23. **TFISIMMOBILE** is true if the TF after binding to the DNA becomes immobile and stays in the same position throughout the entire simulation.

24. **ISBIASEDRANDOMWALK** is true if the TF molecule displays a bias random walk, namely, if the probability to move left or right depends on the binding energy at the left or at the right site; see (Slutsky and Mirny, 2004).

25. **ISTWOSTATERANDOMWALK** is true if the TF molecule displays two modes for the random walk, namely, a search mode and a recognition mode; see (Slutsky and Mirny, 2004). NOT IMPLEMENTED

Note that it is not essential to use upper cases for the headers and, thus, "NAME" will be parsed correctly as well as "Name" or "name". Furthermore, the order of the columns is not important.

3.2.3 TF Cooperativity File

The TF cooperativity file is as well a `.csv` file in which the first row lists the headers for each column and each subsequent row the values for TF cooperativities.

There are twelve mandatory columns which have the following headers:

1. **FIRSTSPECIES** specifies the name of the first TF species in the cooperativity definition. The name has to be found in the TF file.

2. **SECONDSPECIES** specifies the name of the second TF species in the cooperativity definition and the name has to be found in the TF file. Note that the first and the second species can be the same if the system displays homo-cooperativity or different for hetero-cooperativity

3. **TYPE** represents the type of the cooperativity. This is 0 for direct TF-TF cooperativity and 1 for DNA mediated coopeartivity.

4. **FIRSTSPECIESDIRECTION** represents the direction in which the first TF is orientated when bound to the DNA, in order to be able to display cooperative behaviour. This can be -1 for any, 0 for left to right and 1 for right to left.

5. **SECONDSPECIESDIRECTION** represents the direction in which the second TF is orientated when bound to the DNA, in order to be able to display cooperative behaviour. This can be -1 for any, 0 for left to right and 1 for right to left.

6. **FIRSTSPECIESDNAREGION** represents the DNA region where the cooperativity of the first TF is restricted. Note that DNA positions are indexed from 0. In the case of direct TF-TF cooperativity two molecules will be cooperative only if the first one is in this region and the second one in in `SECONDSPECIESDNAREGION` region, but these DNA regions can be omitted for direct TF-TF cooperativity. If this is a DNA mediated cooperativity, when the first TF is bound to the DNA in this region in the right orientation (`FIESTSPECIESDIRECTION`) the affinity of the second TF in the region `SECONDSPECIESDNAREGION` and orientation `SECONDSPECIESDIRECTION` is changed by `AFFINITYINCREASE`. This field is mandatory for DNA based cooperativity and optional for direct TF-TF cooperativity. The

DNA region is specified in a similar way as the description of the DNA sequence (see section 3.2.1) `chr:start..end`, e.g., `"K12:365146..365166"`. Note that from the end it is subtracted the length of the TF (20 *bp*) so in this case the second TF needs to be only at position `365146`.

7. **SECONDSPECIESDNAREGION** represents the DNA region where the cooperativity of the second TF is restricted. Note that DNA positions are indexed from 0. In the case of direct TF-TF cooperativity two molecules will be cooperative only if the first one is in `FIRSTSPECIESDNAREGION` region and the second one in this region, but these DNA regions can be omitted for direct TF-TF cooperativity. If this is a DNA mediated cooperativity, when the first TF is bound to the DNA in this region in the right orientation (`FIESTSPECIESDIRECTION`) the affinity of the second TF in the region `SECONDSPECIESDNAREGION` and orientation `SECONDSPECIESDIRECTION` is changed by `AFFINITYINCREASE`. This field is mandatory for DNA based cooperativity and optional for direct TF-TF cooperativity. The DNA region is specified in a similar way as the description of the DNA sequence (see section 3.2.1) `chr:start..end`, e.g., `"K12:365639..365659"`. Note that from the end it is subtracted the length of the TF (20 *bp*) so in this case the second TF needs to be only at position `365639`.

8. **DIMMERISATIONPROBABILITY** specifies the probability that two TFs that touch each other will display cooperativity. This term introduces stochasticity for the cooperativity behaviour in the sense that there is a certain chance that although all conditions are met, two TF molecules might not display the cooperative behaviour.

9. **AFFINITYINCREASE** represents the change in the affinity once the cooperative condition is met (see below).

10. **ISREVERSIBLE** is true if the change in affinity is not conserved if the cooperative conditions are not met any more. For example in the case of DNA mediated cooperativity if the first TF (`FIRSTSPECIES`) leaves the first region (`FIRSTSPECIESDNAREGION`) then if this is true the affinity of the second TF (`SECONDSPECIES`) for the second site (`SECONDSPECIESDNAREGION`) goes back to the previous value (before it was changed by `AFFINITYINCREASE`). Otherwise, if `ISREVERSIBLE` is false, a change in the position of the first TF (`FIRSTSPECIES`) will not affect the affinity of TF molecule (`SECONDSPECIES`) already bound at the second site (`SECONDSPECIESDNAREGION`).

11. **ISADDITIVE** is true if the cooperativity is additive (see below). This parameter is used only for direct TF-TF cooperativity.

12. **ISFIXED** is true if the cooperativity is fixed (see below). This parameter is used only for direct TF-TF cooperativity.

In the case of direct TF-TF cooperativity, a TF molecule x residing at position i and touching another molecule y residing at position j experiences a change in the waiting time by a factor $c_x^y =$ AFFINITYINCREASE as follows:

$$\begin{aligned}
\text{for multiplicative cooperativity} &: \hat{\tau}_y^j = \tau_y^j \cdot c_x^y, \quad \hat{\tau}_x^i = \tau_x^i \cdot c_x^y \\
\text{for fixed additive cooperativity} &: \hat{\tau}_y^j = \tau_y^j + c_x^y, \quad \hat{\tau}_x^i = \tau_x^i + c_x^y \\
\text{for variable additive cooperativity} &: \hat{\tau}_y^j = \tau_y^j + c_x^y \cdot \tau_x^i, \quad \hat{\tau}_x^i = \tau_x^i + c_x^y \cdot \tau_y^j
\end{aligned}$$

For example, we can consider the following situations:

```
"lacI", "lacI", 1, -1, -1, "K12:365146..365166", "K12:365639..365659", 0.9, 100, true, false, false
"lacI", "lacI", 1, -1, -1, "K12:365639..365659", "K12:365547..365567", 0.9, 1, true, false, false
"lacI", "lacI", 0, -1, -1, "", "", 0.9, 10, true, true, false
"lacI", "lacI", 0, -1, -1, "", "", 0.9, 10, true, true, true
"lacI", "lacI", 0, -1, -1, "", "", 0.9, 10, true, false, false
```

The first two lines represent DNA mediated cooperativity (the first one increases the affinity by 100 fold, while the second one does not have any effect `AFFINITYINCREASE=1`) and the last three are examples of direct TF-TF cooperativity (the third is a variable additive case, the fourth a fixed additive case and the last a multiplicative case). In all cases there is a 90% change of cooperativity to emerge and the orientation of the TF is not important.

Similar as in the TF file, it is not essential to use upper cases for the headers("FIRSTSPECIES" will be parsed correctly as well as "FirstSpecies") and the order of the columns is not important.

3.2.4 TS File

This file contains on each line a series of target sites connected by a logical operations. A target site is defined similar to the description of DNA sequence (see section 3.2.1). Note that DNA positions are indexed from 0. The structure of the target site definition is `TFname:chr:start..end:orientation`. In addition to, chromosome, start and end position the target site can have TF orientation and the name of the TF for witch this DNA locus is a target site. The orientation means that the target site is reached only when the TF is in the correct orientation on the DNA. A value of `-1` indicates that the orientation is not important. For example, `lacI:K12:365146..365147:-1` means that the target site of the lacI is at position `365146` on the DNA, the lacI molecule can bind in any orientation.

Logical operations can be applied to compute complex occupancy patterns. We considered three logical operations `AND`, `OR` and `NOT`. In addition when specifying complex occupancy patterns one can use parentheses. An example of a promoter logic that can be specified in this file is the following:

```
(lacI:K12:365546..365547:-1 OR lacI:K12:365145..365146:-1) AND (NOT lacI:K12:365638..365639:-1)
```

This specification means that a gene could be turned on if at least one of the two DNA positions (`365,546` and `365,145`) are occupied by lacI molecules (in any orientation) and at the same time there is no lacI molecule at position `365,638` (in any orientation).

3.2.5 Contact Matrix File

The file is a .tsv file and contains the two segments of DNA (bins) and on column 7 the score of the interaction. It is of the form:

3R	21777000	21778000	3R	21777000	21778000	54.6032437	X_	Y_
3R	21777000	21778000	3R	21778000	21779000	19.0771798	21777000	21777000
3R	21777000	21778000	3R	21779000	21780000	24.1291193	21777000	21778000
3R	21777000	21778000	3R	21781000	21782000	11.8553383	21777000	21779000
3R	21777000	21778000	3R	21782000	21783000	17.9880062	21777000	21781000
3R	21777000	21778000	3R	21783000	21784000	8.7048639	21777000	21782000
3R	21777000	21778000	3R	21784000	21785000	9.9344502	21777000	21783000

Where for each bin it is given the name of the chromosome (example used above is 3R), the start and end position (column 2,3 and 5,6) and the score (column 7) between the two regions. The chromosome is checked to match the DNA sequence chromosome.

4 Output

The application generates nine types of output files. The output files will be saved in the folder specified by the **Output folder** parameter. Furthermore, the names of this files starts with the **Output filename** parameter followed by different suffix for each file.

1. **parameters file** with suffix `_params`
2. **DNA sequence file** with suffix `_DNA_seq`
3. **status file** with suffix `_status`
4. **affinity landscape file** with suffix `_affinity_landscape`
5. **occupancy file** with suffix `_occupancy`
6. **target sites file** with suffix `_target_site`
7. **sliding lengths file** with suffix `_TF_sliding_lengths` or `_TF_observed_sliding_lengths` where `TF` is the name of the TF species
8. **dynamic behaviour file** with suffix `_params`
9. **TF species file** with suffix `_TF_species`
10. **Cumulative Matrix file** with suffix `_cumulative_simulated_matrix`

If the **Output filename** is empty the new unique name generated for the input file is used and the suffixes are the ones mentioned above.

The parameter file was discussed in detail in section 3.1.1. Furthermore, only when a random DNA sequence is generated, the simulator will also output the sequence in a format similar to the one described in section 3.2.1

The steady state files and the dynamic behaviour file(s) are written in `.csv` files. We optioned for this type of file because it can be easily imported and processed in various statistical software (such as R, Matlab/Octave , Maple) and appropriate columns (or functional combination of columns) can be easily plotted in graphical software (such as Gnuplot).

Next we will describe the structure of each output file that was not presented above.

4.1 Status File

The status file displays information regarding the status of the simulation. The information provided by this file can be summarised as follows:

1. the file were the input parameters were saved,
2. information on the seed of the random number generator (if the same seed is used or different seed is used),
3. the stop time of the simulation (biological time),
4. status of the simulation run
   (a) If in debug mode, all events that take place are printed and described in a detail one line text. For example, this could look like

   ```
   0.054633340014343425: TF 0 of type nonCognate slide left from position 713 by 1 bp
   0.05463636745459822: attempted to slide left TF 0 of type nonCognate
                from position 712 by 1 bp, but this is blocked by molecule 1 (TF lacI)
   0.05463766870843872: TF 0 of type nonCognate slide right from position 712 by 1 bp
   ```

   Note that each line starts with the time at which the event was performed followed by a brief description of the event.

   (b) If not in debug mode, the following information at various time steps is printed
      i. **cellTime** represents the time of the cell. This is measured in seconds.
      ii. **elapsedTimeSec** specifies how long it took to simulate up to the current cellTime. This is measured in seconds.
5. the time in seconds required to simulate the system,
6. the total number of collisions events on the DNA.
7. the statistics on the speed at which the system was simulated: (*i*) total number of events performed and (*ii*) number of events performed per second.
8. a list of the files where the output was saved.

4.2 Affinity Landscape File

The affinity landscape file is a `.wig` file which contains as a first line a header with information on the file

```
fixedStep  chrom=chr  start=0  step=1000  span=1000
```

the information stored is the fact that this is a fixed step wig file and contains data from chromosome `chr` starting at base pair `0`, lines describe positions that are separated by `1000` base pairs and each line describes a span of `1000`. Following this line there is another header describing the content of each column. There are $n + 1$ columns where n is the number of TF species in the system (cognate or non-cognate). The first column is the position described in current line, while the next ones describe a species. For example, the following line is a header of a system with two species a cognate and a non-cognate one.

```
"position", "nonCognate", "lacI"
```

The position represents the left most cover base pair by the bound TF on the positive strand and the right most cover base pair on the negative one.

Note that in parentheses the threshold value used is denoted. Any value under this threshold value was discarded.

Each value in the subsequent line represents the average time a TF molecule of that type spends at that position. Furthermore, if the line describes on chunk which spans over multiple base pairs (`span`), then all the affinity values above the threshold (`WIG_THRESHOLD`) will be added together and the sum divided to the number of base pairs in the chunk (`span`). This represents the average affinity within the chunk (`span`) and has the purpose of smoothing the affinity landscape.

To obtain the binding energy from the average waiting time, one can apply the following transformation

$$E_i^j = -log(\tau_i^j/\tau_0),\ \forall j \tag{1}$$

4.3 Occupancy File

The occupancy file is similar to the affinity landscape one except two aspects. First, the file contains an extra column which represents the collision count, normalized to the highest value.

```
fixedStep  chrom=chr  start=0  step=1000  span=1000
"position", "collisionsCount", "nonCognate", "lacI"
```

The same as in the case of affinity file, the position represents the left most cover base pair by the bound TF on the positive strand and the right most cover base pair on the negative one.

Furthermore, each subsequent line contains information about number of collisions in current DNA region, and the time molecules of each species spend bound to the DNA in this area. The bound time represents the time the site was occupied by molecules of the current TF type and then the same smoothing approach is used as in the case of the affinity landscape. If the line describes on chunk which spans over multiple base pairs (`span`), then all the occupancy values above the threshold (`WIG_THRESHOLD`) will be added together and the sum divided to the number of base pairs in the chunk (`span`). This represents the average affinity within the chunk (`span`) and has the purpose of smoothing the affinity landscape.

In Figure 3 we plot four examples of the occupancy and affinity files.

4.4 TF Species File

The TF species output file contains information on all TF species used during the simulation. This file has the following thirty seven columns:

1. **id** represents the internal id of the TF species.
2. **name** represents the name of the TF species.
3. **copyNumber** specifies the TF copy number for the TF species.

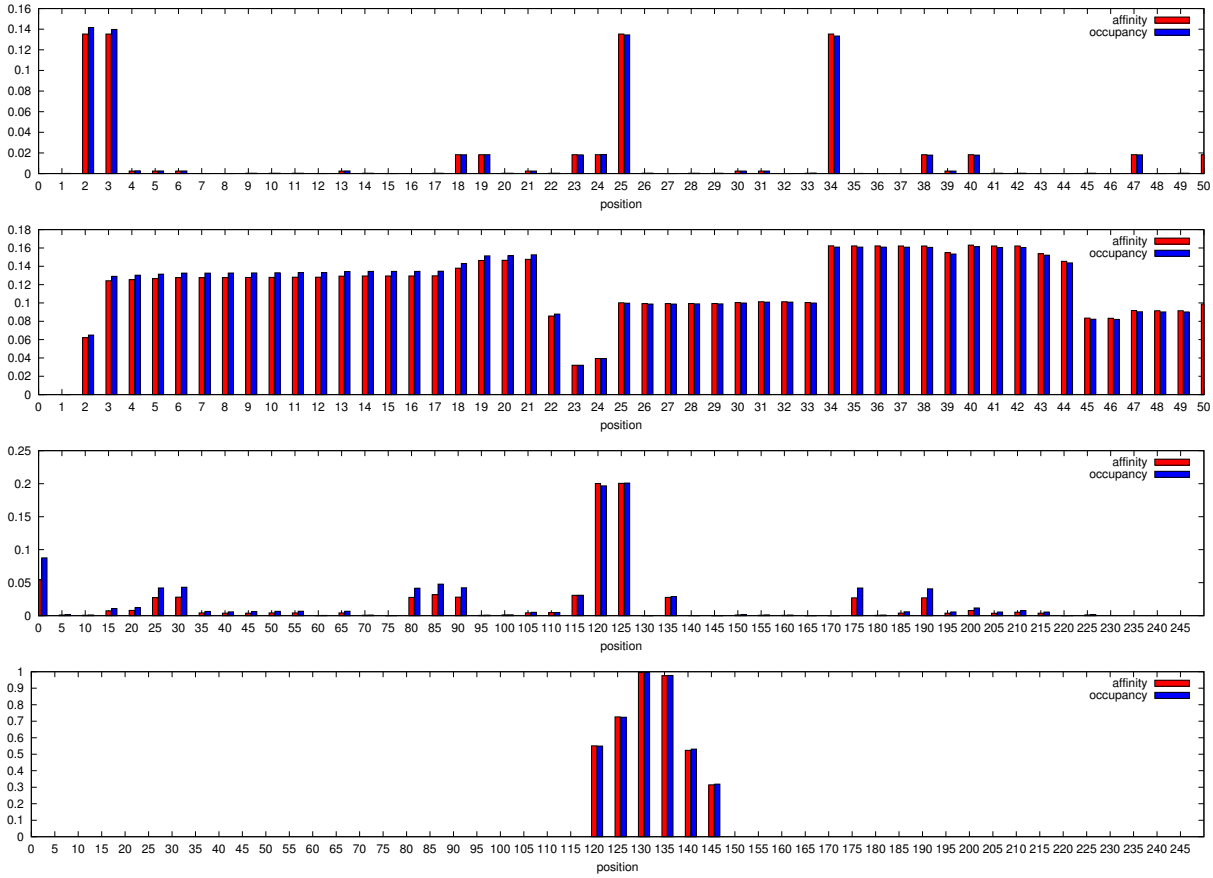

Figure 3: *Affinity vs Occupancy.* We consider a random 250 *bp* DNA and TF molecules which can bind/unbind, hop, jump, slide left/right. The top two figures consider 1 TF molecule, while the bottom two ones consider 2 TF molecules. The second and the last ones consider that a TF molecule covers more than one base pair when representing the occupancy and affinity, while the other two (first and third) the occupancy is represented as the first base pair covered. In the last two figures we use the wig step of 5 and in the last one a wig threshold of 0.5.

4. **es** represents the energy penalty for a nucleotide mismatch between DBD and current position (Gerland et al., 2002). This is ε from (Gerland et al., 2002) and is measured in K_BT.

5. **dbd** represents the DNA binding Domain of the TF species. For details see element 2 from the list of parameters in the TF file (section 3.2.2).

6. **sizeLeft** represents the number of base pairs covered to the left of the DBD by a TF molecule bound to the DNA.

7. **sizeRight** represents the number of base pairs covered to the right of the DBD by a TF molecule bound to the DNA.

8. **sizeTotal** represents the total number of base pairs covered by a TF molecule bound to the DNA (to the left, under DBD and to the right).

9. **assocRate** represents the association rate between TF molecules and DNA.

10. **timeBoundAvg** represents the average proportion of time a TF molecule of current species stays bound to the DNA.

11. **residenceTimePerBinding** represents the average time in seconds a TF molecule spends from the moment it binds to the DNA until it unbinds from the DNA.

12. **slidingEventsPerBinding** represents the average number of sliding events a TF molecule will perform from the moment it binds to the DNA until it unbinds from the DNA.

13. **slidingLengthPerBinding** represents the average number of base pairs scanned by a TF molecule from the moment it binds to the DNA until it unbinds from the DNA.

14. **observedSlidingLength** represents the average number of base pairs scanned by a TF molecule from the moment it binds to the DNA until it completely dissociates from the DNA and releases in the cytoplasm. This is the observed sliding length.

15. **initialDrop** specifies the area where a TF molecule is first bound to the DNA. For details see element 8 from the list of parameters in the TF file (section 3.2.2).

16. **isCognate** is true if the TF species is a cognate one (has specific affinity for the DNA) or false if it is a non-cognate one.

17. **unBindingProbability** represents the probability that a TF unbinds during next event.

18. **slideLeftProbability** represents the probability that a TF slides left during next event.

19. **slideRightProbability** represents the probability that a TF slides right during next event.

20. **jumpingProbability** represents the probability that a TF performs a jump when unbound instead of returning to the DNA.

21. **hopSTDdisplacement** represents the standard deviation of a TF that unbinds and attempts to rebind correlated. Note that the displacement is Gaussian distributed and centred around the original position. This is measured in base pairs.

22. **specificWaitingTime** represents the waiting time of a specific TF measured in seconds. This is $\tau_0 \cdot \exp(E_{ns})$ from (Gerland et al., 2002) and is measured in seconds.

23. **stepLeftSize** represents the size of the step to the left measured in base pairs when the TF performs a left slide.

24. **stepRightSize** represents the size of the step to the right measured in base pairs when the TF performs a right slide.

25. **uncorrelatedDisplacementSize** represents the size of the uncorrelated displacement, which is measured in base pairs. If the hop distance is higher than the uncorrelated displacement size then the hop becomes a jump and the TF molecule needs to first release in the cytoplasm.

26. **stallsIfBlocked** is true if the TF molecule stays at current position if cannot relocate trough a hop and false if it unbinds during an unsuccessful hopping event.

27. **collisionUnbindingProbability** represents the probability that if a TF molecule collides with another molecule on the DNA it will unbind.

28. **affinityLandscapeRoughness** represents the roughness of the affinity landscape. This is measured in K_BT and is specified for non-cognate species in order to generate a random non-specific affinity landscape.

29. **preboundProportion** represents the proportion of TF molecules that are already bound when the simulation starts.

30. **preboundToHighestAffinity** is true if the TF is prebound to the highest affinity sites when the simulation starts.

31. **TFisImmobile** is true if the TF after binding to the DNA becomes immobile and stays in the same position throughout the entire simulation.

32. **eventsBindingTotal** represents the total number of binding events performed by molecules of current TF species.

33. **eventsUnbindingTotal** represents the total number of unbinding events performed by molecules of current TF species.

34. **eventsSlideLeftTotal** represents the total number of sliding left events performed by molecules of current TF species.

35. **eventsSlideRightTotal** represents the total number of sliding right events performed by molecules of current TF species.

36. **eventsSlideTotal** represents the total number of sliding events performed by molecules of current TF species.

37. **eventsHoppingTotal** represents the total number of hopping events performed by molecules of current TF species.

38. **events3DhoppingTotal** represents the total number of 3D-hopping events performed by molecules of current TF species.

39. **eventsForcedJumps** represents the total number of hopping events that resulted in displacements longer than the uncorrelatedDisplacementSize and which, consequently, are transformed in jumps (release in the cytoplasm and uncorrelated rebinding).

40. **eventsHopOutsideDNA** represents the total number of hopping events that resulted in displacements outside the DNA and resulted in unbinding from the DNA.

41. **collisionsCount** represents the total number of collisions (failed movements on the DNA).

42. **pkmicroenv** represents the probability that if a TF molecule performs a jump it will remain in the microenvironment and perform a 3D hop or continue to dissociate from its location into the nucleoplasm.

43. **cooperativity** represents a summary of the cooperative interactions, which the current TF species is displaying. Several cooperativities are separated by semicolon. An example of cooperativity description is the following

```
[1->1][type=DNA][K12:365146..365147->K12:365639..365640][reversible][affinityIncrease=100.0 (multiplicative)];
[1->1][type=direct][dimerisationProbability=0.9][ireversible][affinityIncrease=10.0(additive-non-fixed)]
```

    First the cooperativity description contains the id of the species which affects (the one before `->`) and the one that is affected by cooperativity (the one after `->`). Next, the type of cooperativity is specified as either `type=DNA` for DNA mediated cooperativity or `type=direct` for direct TF-TF interaction. Next, in the case of DNA mediated cooperativity, the description contains two DNA regions, the one that affects (to the left of `->`) and the one that is affected (to the right of `->`), while in the case of direct TF-TF cooperativity the dimerization probability is mentioned. Furthermore, for both types of cooperativity, the description specify whether the cooperativity is reversible or not and the affinity increase; see section 3.2.3 for a complete description on the meaning of the parameters.

44. **isBiasedRandomWalk** is true if the TF molecule displays a bias random walk, namely, if the probability to move left or right depends on the binding energy at the left or at the right site; see (Slutsky and Mirny, 2004).

45. **isTwoStateRandomWalk** is true if the TF molecule displays two modes for the random walk, namely, a search mode and a recognition mode; see (Slutsky and Mirny, 2004).

One strategy would be that all sliding lengths are recorded only initially for short simulations with the aim to validate the system. Subsequently, the user can stop recording all sliding lengths, which would significantly increase the speed (`OUTPUT_SLIDING_LENGTHS = false;`). If sliding lengths are not recorded, then the simulator uses the following equations to compute the residence time, the number of sliding events per binding, the sliding length and the observed sliding length.

$$\text{residenceTimePerBinding} = \frac{\text{timeBoundAvg} \cdot \text{stopTime}}{\text{eventsBindingTotal/copyNumber}} \tag{2}$$

$$\text{slidingEventsPerBinding} = \frac{\text{eventsSlideTotal}}{\text{eventsHoppingTotal} + \text{eventsBindingTotal}} \tag{3}$$

$$\text{slidingLengthPerBinding} = \sqrt{2 \cdot \text{slidingEventsPerBinding}} \tag{4}$$

$$\text{observedSlidingLength} = \sqrt{2 \cdot \frac{\text{slidingEventsPerBinding} + \text{eventsHoppingTotal}}{\text{eventsBindingTotal}}} \tag{5}$$

where stopTime is the time the simulation stopped at. Note that these equations are correct for low hopping, i.e., the number of hops performs during a 1D random walk should not be higher than 10. For details on these formulas see main manuscript.

4.5 Target Sites File

The target site file is a `.csv` file which stores information on the time it took a target sites to be first occupied by a specific TF and how long it was occupied subsequently by specific TF molecules.

The `.csv` file has four columns

1. **targetSite** represents the specification of the target site logic (for details see section 3.2.4).
2. **firstReached** represents the first time the logic described in the first column evaluated to true.
3. **timesReached** represents the number of times the logic described in the first column evaluated to true.
4. **timeOccupied** represents the total amount of time the logic described in the first column evaluated to true.

4.6 Sliding lengths File

The user can export all the sliding lengths and the observed sliding lengths. The sliding length represents the number of base pairs covered before the TF unbound, independent of whether the TF rebound fast. The observed sliding lengths represent the number of base pairs occupied by a molecule, before unbound completely from the DNA (without fast rebinding). for each TF species a file is generated and in each file all sliding lengths values and sliding events are printed. The `.csv` files have three columns

1. **no** represents the id of the recording.
2. **slidingLength** represents the sliding length of the TF during that binding.
3. **slidingEvents** represents the number of sliding events of the TF during that binding.

In addition, a second file for each species will contain information on the observed sliding lengths. These is as well a `.csv` file with a two columns.

1. **no** represents the id of the recording.
2. **observedSlidingLength** represents the observed sliding length of the TF during that binding.

For example, the sliding lengths file can be used to get sliding lengths distributions. In Figure 4 we plotted the histogram of the sliding lengths for a TF species on the *E.Coli* genome. The histogram was produced in R using the `csv` file generated by the application.

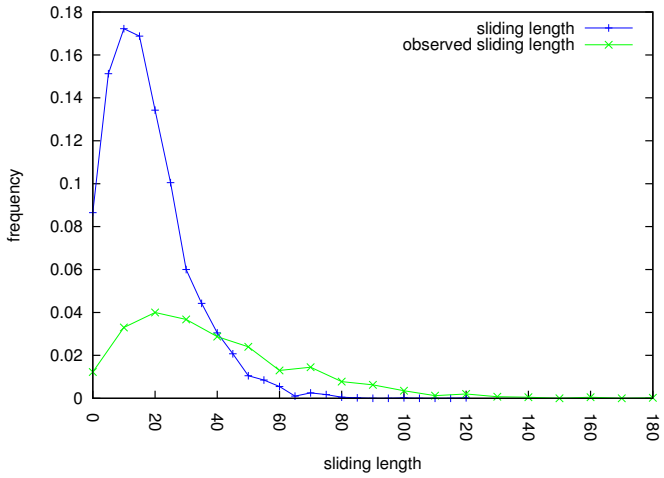

Figure 4: *Sliding lengths histogram.* We plot the sliding lengths and the observed sliding lengths of a TF on the *E.Coli* genome. The histogram is produced in R.

4.7 Target site dynamic Behaviour File

For each target site, the user can track the occupancy of the target site at each time point. The output will be stored in a `.csv` file which has a name with the suffix `_target_site_follow`. The file contains $n + 1$ columns where n is the number of target sites. The first column represents the time at which the recording is made. The time at which the occupancy of the target site is recorded is the exact time the occupancy of one of the target sites was changed.

The subsequent column(s) correspond to the occupancy of each target site. A value of 1 indicates the target site is occupied, while a value of 0 indicates that the target site is free. For example, a dynamic file of a species that has only one molecule, can look like

```
time, "lacI:K12:365546..365547:-1"
0,  0
1.666439926858294,  1
2.668742514616741,  0
2.668742514616741,  1
3.0346325493683177,  0
3.0346325493683177,  1
```

4.8 TF dynamic Behaviour File

Finally, for each TF species, the user can track the position of each molecule at various time points. The dynamic behaviour is a `.csv` file which has a name with the suffix `_TF_SPECIES`. Note that `SPECIES` is replaced by the observed name of the TF species that is followed.

The file contains $n + 5$ columns where n is the copy number of the investigated species if $n \leq 10$ or 5 if the $n > 10$. The first five columns are the following

1. **time** represents the time at which the recording is made. The time at which the position is recorded and the size of the step between consecutive recordings is compute as the ratio between `STOP_TIME` and `OUTPUT_TF_POINTS`.
2. **freeMolecules** represents the absolute number of free molecules.

3. **freeMoleculesProportion** represents the proportion of unbound molecules.
4. **freePositions** represents the absolute number of free positions on the DNA.
5. **freePositionsProportion** represents the proportion of free positions on the DNA.

The subsequent column(s) correspond to the position of the corresponding molecule on the DNA and are printed only if the species has at most 10 molecules. The column headers start with `TF` word followed by the id of the molecule that is plotted in that column. When presenting the position of a molecule, a value of -1 is used when the molecule is free in the cytoplasm.

For example, a dynamic file of a species that has only one molecule, can look like

```
# time, TF0
1.7451192827886659E-4, -1
1.7466444310461482E-4, 12
1.746656365210957E-4, 13
1.748016008986566E-4, 12
```

This file indicates that, initially, the molecule is free in the cytoplasm (at time `1.7451192827886659E-4 s`) and then it binds at position `12` (at time `1.7466444310461482E-4 s`), from where the molecule starts its random walk on the DNA.

Figure 5 shows an example of a random walk performed by 1 and 3 molecules, while Figure 6 an example of a the dynamic behaviour of DNA occupancy.

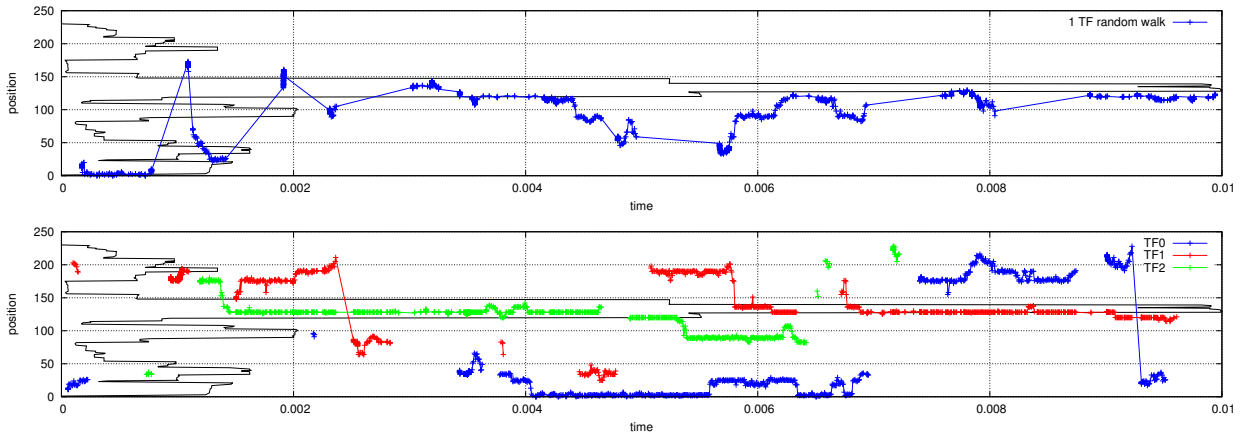

Figure 5: *Dynamic Behaviour of TF molecules.* We consider a random 250 *bp* DNA and TF molecules which can bind/unbind, hop, jump, slide left/right. (**Top**) 1 TF molecule (**Bottom**) 3 TF molecules. The black line on the y-axis represents the affinity for that position of a TF molecule.

4.9 Cumulative Simulated Matrix file

The Cumulative Simulated Matrix file contains information about the 3D contact maps that were simulated and the ones that were used as input (see Figure **??**). The coordinates for two interacting bins are given by `BIN_X` and `BIN_Y`, each specifying chromosome, start and end position of the bin (specified using `chr:start..end` format). Next, the Hi-C score that was used as input for the two bins is provided and then the total time during the simulation when the two bins interacted. The `PROPORTION_TIME` is the two bins interacted during the simulation.

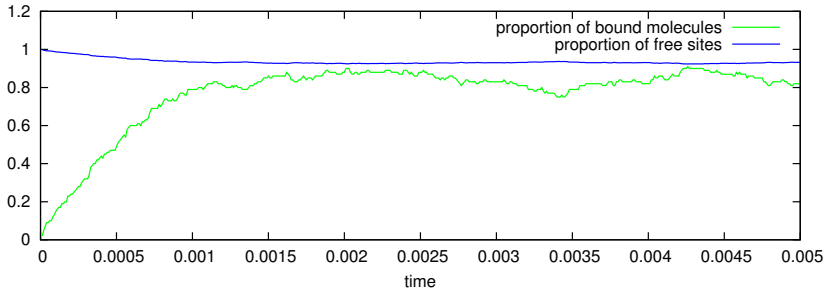

Figure 6: *DNA occupancy*. We consider a random 46000 *bp* DNA and 100 non-cognate TF molecules which can bind/unbind, hop, jump, slide left/right.

"BIN_X"	"BIN_Y"	"HIC_SCORE"	"CUMULATIVE SIMULATION"	"PROPORTION TIME"
3R:25951000 ..25951500	3R:25951000 ..25951500	71.1010458	2063736.825	0.01
3R:25951000 ..25951500	3R:25951500 ..25952000	0	0	0.01
3R:25951000 ..25951500	3R:25952000 ..25952500	0	0	0.01
3R:25951000 ..25951500	3R:25952500 ..25953000	3.8528042	-4148.749944	0.01
3R:25951000 ..25951500	3R:25953000 ..25953500	3.367547	72103.12254	0.01
3R:25951000 ..25951500	3R:25953500 ..25954000	3.5182811	163894.426	0.01
3R:25951000 ..25951500	3R:25954000 ..25954500	0	0	0.01

5 Backup of the simulation

The command line interface also provides backups of the simulations at regular time intervals. Note that there is no backup system for the graphical interface. The backup system is started if the system is at an intermediary point and there was no backup done in the last `BACKUP_AFTER` seconds (or in the last second if this value is not specified in the command line). Once the backup is started the entire system state is saved into a file which name starts with `backup_` and that has the extension `.bkp`. In addition, a file with the same name and the extension `.txt` stores on each line the following informations:

1. the name of the file where the backup was last saved
2. the number of the current set of simulations
3. the intermediary point where the backup was performed

4. the number of intermediary steps

These files can be large (up to a few GB) and can take one to a few minutes to be saved.

While saving the file, the old backup is not modified and the new one starts in a new file which has the prefix `tmp`. After the new backup is finished, the old backup is deleted and the new one is renamed to have the prefix `backup`. In addition, the information of the new backup is saved in the `.txt` file. Once the simulation is started again, the simulator will start the first backup file that it can find in the current directory (the one that starts with `backup`). Thus, to ensure that the simulation restarts from time zero, one needs to delete all backup files in the current directory. At the end of the simulation, the simulator will automatically remove the last created backup file.

Note that setting `BACKUP_AFTER` equal to -1 results in no backup being created.

6 General comments

Multi-threading could possibly increase the simulation speed. Nevertheless, usually, one needs several runs to do statistics when measuring the time to reach the target site or the amount of time the target site was occupied and, consequently, several simulations are needed for each set of parameters.

If a user has more CPUs, then he/she can run in parallel several simulations. While running multiple independent simulation does not require any communication between processes, multi-threading would require that multiple threads communicate, which results in an overhead. For example, if one has 8 CPUs and requires 8 replicates of the simulation, then running 8 parallel non-multi-threaded simulations would finish sooner than running 8 multi-thread simulations run serial. This is the main reason for which we did not find the multi-threading as an essential option (at least at this time).

7 Acknowledgements

N.R.Z. would like to acknowledge financial support from Medical Research Council through the MRC Bioinformatics Training Fellowship program.

References

Berg, O. G. and von Hippel, P. H. (1987). Selection of DNA binding sites by regulatory proteins statistical-mechanical theory and application to operators and promoters. *Journal of Molecular Biology*, 193(4):723–750.

Blainey, P. C., Luo, G., Kou, S. C., Mangel, W. F., Verdine, G. L., Bagchi, B., and Xie, X. S. (2009). Nonspecifically bound proteins spin while diffusing along dna. *Nature Structural & Molecular Biology*, 16(12):1224 – 1229.

Elf, J., Li, G.-W., and Xie, X. S. (2007). Probing transcription factor dynamics at the single-molecule level in a living cell. *Science*, 316:1191–1194.

Gerland, U., Moroz, J. D., and Hwa, T. (2002). Physical constraints and functional characteristics of transcription factor-DNA interactions. *PNAS*, 99(19):12015–12020.

Gillespie, D. T. (1976). A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361.

Gillespie, D. T. (2007). Stochastic simulation of chemical kinetics. *Annu Rev Phys Chem.*, 58:35–55.

Riley, M., Abe, T., Arnaud, M. B., Berlyn, M. K., Blattner, F. R., Chaudhuri, R. R., Glasner, J. D., Horiuchi, T., Keseler, I. M., Kosuge, T., Mori, H., Perna, N. T., Plunkett, G., Rudd, K. E., Serres, M. H., Thomas, G. H., Thomson, N. R., Wishart, D., and Wanner, B. L. (2006). Escherichia coli k-12: a cooperatively developed annotation snapshot - 2005. *Nucleic Acids Research*, 34(1):1–9.

Stormo, G. D. (2000). DNA binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23.

Weindl, J., Hanus, P., Dawy, Z., Zech, J., Hagenauer, J., and Mueller, J. C. (2007). Modeling DNA-binding of Escherichia coli σ^{70} exhibits a characteristic energy landscape around strong promoters. *Nucleic Acids Research*, 35(20):7003–7010.

Wunderlich, Z. and Mirny, L. A. (2008). Spatial effects on the speed and reliability of protein-DNA search. *Nucleic Acids Research*, 36(11):3570–3578.