Conjugative Simulator Report

1 Purpose

The purpose of the conjugative simulator is to simulate the behavior of conjugation-based computing experiments within bacteria E. Coli. These experiments form the core idea of the project and they are based on interplay of three types of plasmids: pA, pB and pC. We use two strains of E. Coli: the *producer strains* which carry either the plasmid pA or pB, and the *computing strain* which originally carries the plasmid pC. Producer cells may contain different alleles of pA or pB. These plasmids are transferred to computing cells via the process of horizontal gene transfer so that most of the cells would end up containing three plasmids pA, pB and pC. The computing strain also contains a biochemical device called *comparator*. Its purpose is to evaluate whether the combination of alleles of pA and pB together with pC present in the cell satisfies certain pre-defined condition. In the negative case the comparator produces protein TetR which makes the plasmids pA and pB unstable. They are eventually lost and replaced by different alleles and the process is repeated. In the affirmative case the plasmids become stable and allowed to be replicated. We hide all the implementation details, helper plasmids and other supporting mechanisms at this abstract level of description.

The role of the conjugative simulator is to find optimal initial conditions to get the fastest system convergence to a stable solution. We mainly want to address these two questions:

- What is the optimal starting ratio between plasmids Pa and Pb?
- What is search capability of the system – how many plasmids Pc do we need for the system to converge?

2 Entities, state variables and scales

Bacteria - Represents one living bacteria of size about 1um. Its state variables are:

- coordinates x, y (integer, maximal values given by dish size)
- length (1 to 10, corresponds to real size of 0.1um to 1um), initial value is 5
- isStarving (true or false) – set to true when the bacterium has not consumed any nutrient for a certain period of time
- isTransconjugantPa (true or false)
- isTransconjugantPb (true or false)
- isTransreplicantPa (true or false)
- isTransreplicantPb (true or false)

Plasmid - Represents DNA entity that is contained in bacteria

Site - One spot for bacterium. One site can contain only one bacterium regardless of its size.

isEmpty – no bacterium is in a site

nutrientAmount – how many pieces of nutrient is in a site

Nutrient – Represents bacterial food. Nutrient is located in site and can be stacked (there can be more than one unit of nutrient)

Event – An object that represents a potential action. Each event defines a source and the destination (meaning source site and the destination site). An event type can be any of: Elongation, Conjugation, Reproduction, ReproductionWithPushing PlasmidLoss, Consumption, Movement or Death.

The List of events – This list contains events that are about to be processed during current simulation step.

The List of pushing – This list contains events of type ReproductionWithPushing.

3 Process overview and scheduling

3.1 Space (Grid, Petri Dish)

One bacterium can occupy one site. There can be only one bacterium per site, regardless of size of occupying bacteria. Each site is considered to be square shaped and has 8 neighboring sites. Distance from the center of one site to the center of neighboring site is considered to be equal for all neighboring sites. Each site can contain nutrient pieces. Therefore, the simulation uses discrete space.

3.2 Time

The simulation time is divided into time steps. One simulation step is roughly 2 minutes of real time. Each bacterium does its actions in each time step. Therefore, the simulation runs in discrete time.

3.3 Event Groups

All the events that can occur during simulation are divided into two groups. Description of events itself is located in section 3.5, Features – Simulation events.

Free Event Group – Any event or all events from this group could happen in each simulation step for each bacterium. Events in this group are:

- Bacterial Elongation
- Nutrient Consumption
- Movement
- PlasmidReplication
- Death

Exclusion Event Group – There can be only one event for each bacterium in every simulation step. They are mutually exclusive. Events in this group are:

- Reproduction (with or without pushing other bacteria)
- Conjugation

Other events

- PlasmidLoss (can happen only in during reproduction)

3.4 Simulation step

One simulation step consists of a sequence of six phases. Phases are executed sequentially in the order in which they are mentioned.

I. Generating events

Input: the environment (the dish)
Output: the List of events, the List of pushing

In this phase we calculate the probability of every potential event/action for each bacterium in the system.

Note that this calculation is a submodel and it is described in section 7.1 Calculating bacterial rates and shown in Image 3 to Image 8.

Then we select for each bacterium (with a respect to previously calculated probability) either reproduction or conjugation and add this selected event into the List of events. Some bacteria are unable to reproduce because there is no empty site available. If there is some empty space nearby, bacterium could push other bacteria to make some space for reproduction. That would produce a special event ReproduceWithPushing which is added to the List of pushing. This decision schema is shown in Image 1.

There is also a chance that a PlasmidLoss event is generated during reproduction. PlasmidLoss event is added into the List of events.

Finally we decide which other events are going to happen according to their rates – elongation, consumption, movement or death. All selected events are also added into the List of events.

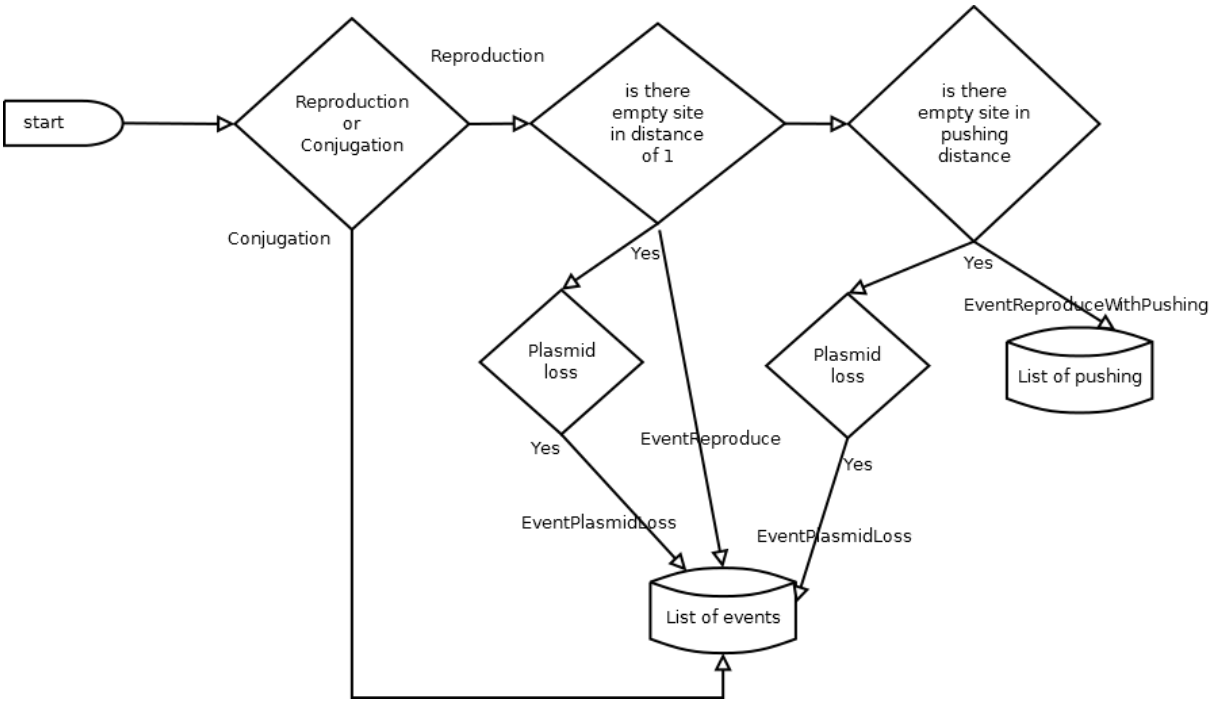

Image 1: Decision schema of reproduction and conjugation

II. Conflict resolution

Input: the List of events
Output: filtered version of the List of events

The list of Events now contains description of what would every bacteria like to do. Some events are, however, conflicting. Conflicting events are those events that cannot happen in the same site. An example of conflict would be the movement of two bacteria into the same site.

In this phase we iterate through the List of events and identify conflicting events. Once we identify a group of conflicting events, we randomly select one event from that group that we keep and then discard (remove from the List of events) the other conflicting events.

After this phase the List of events does not contain any conflict.

III. Event prioritization

Input: filtered version of the List of events
Output: sorted and filtered version of the List of events

It is important to sort events in the List of events in the way that we will be able to use them all. Events sorting can prevent situations like 'bacterium dies and then reproduce' to happen.

Correct order of events is (with Consumption happening first): Consumption, Reproduction, PlasmidLoss, Conjugation, Movement, and Death.

In this phase we sort all events in the List of events according to the priorities mentioned in the previous paragraph.

IV. Event execution

Input: sorted and filtered version of the List of events
Output: updated environment

In this phase we execute each event in the List of events (except for ReproduceWithPushing event). Execution means that we actually update the environment that was static until this moment.

V. Pushing (shoving)

Input: the List of pushing
Output: updated environment

In this phase we execute each event from the List of pushing. Execution means that we push bacteria from event source site towards nearest empty site and thus making one empty site right next to the bacterium that is ready to reproduce. Bacterium then immediately reproduces into the empty site.

VI. Generating output

Input: the environment
Output: image + semicolon delimited list of statistical data

This phase produces a graphical and statistical output. Graphical output visualize current situation in the dish. As shown in the Image 2, there are different colorings for bacteria carrying different plasmid combinations. Gray color represents available nutrient, green color is a bacteria without any plasmid, red is bacteria with plasmid Pa or Pb, orange is bacteria with Pa and Pb and yellow is bacteria with all of Pa, Pb and Pc. Actual coloring is about to be changed in future versions of simulator.

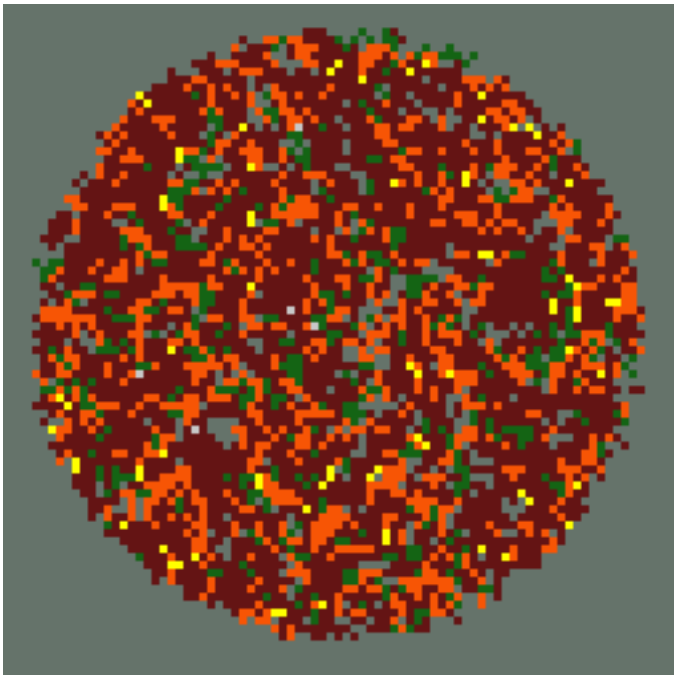

Image 2: An example graphical output generated during one simulation step

Statistical output is a semicolon delimited list containing counts of bacteria:

- all bacteria in the dish

- carrying only plasmid Pa
- carrying only plasmid Pb
- carrying only plasmid Pc
- carrying plasmid Pa or Pb
- carrying plasmid Pc and one of Pa or Pb
- carrying all three plasmids – Pa and Pb and Pc

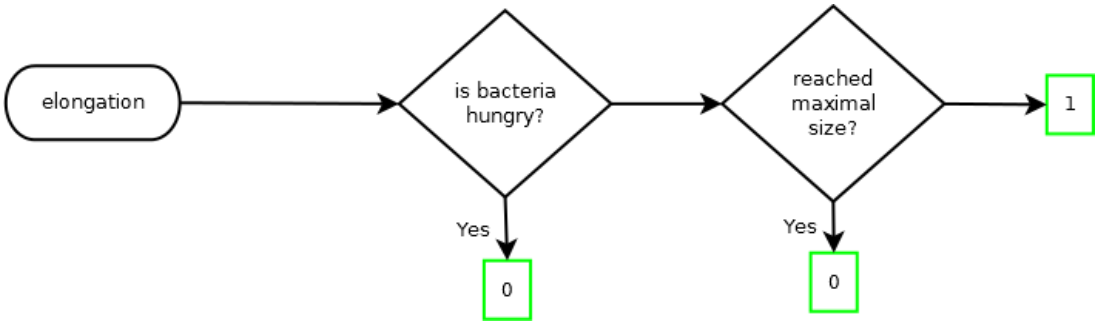

Image 3: Calculating elongation rate

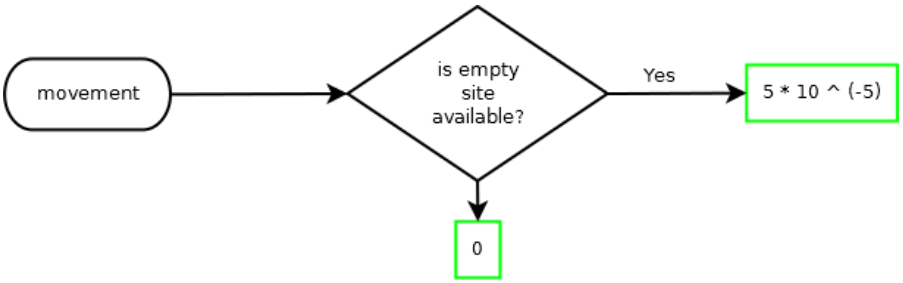

Image 4: Calculating movement rate

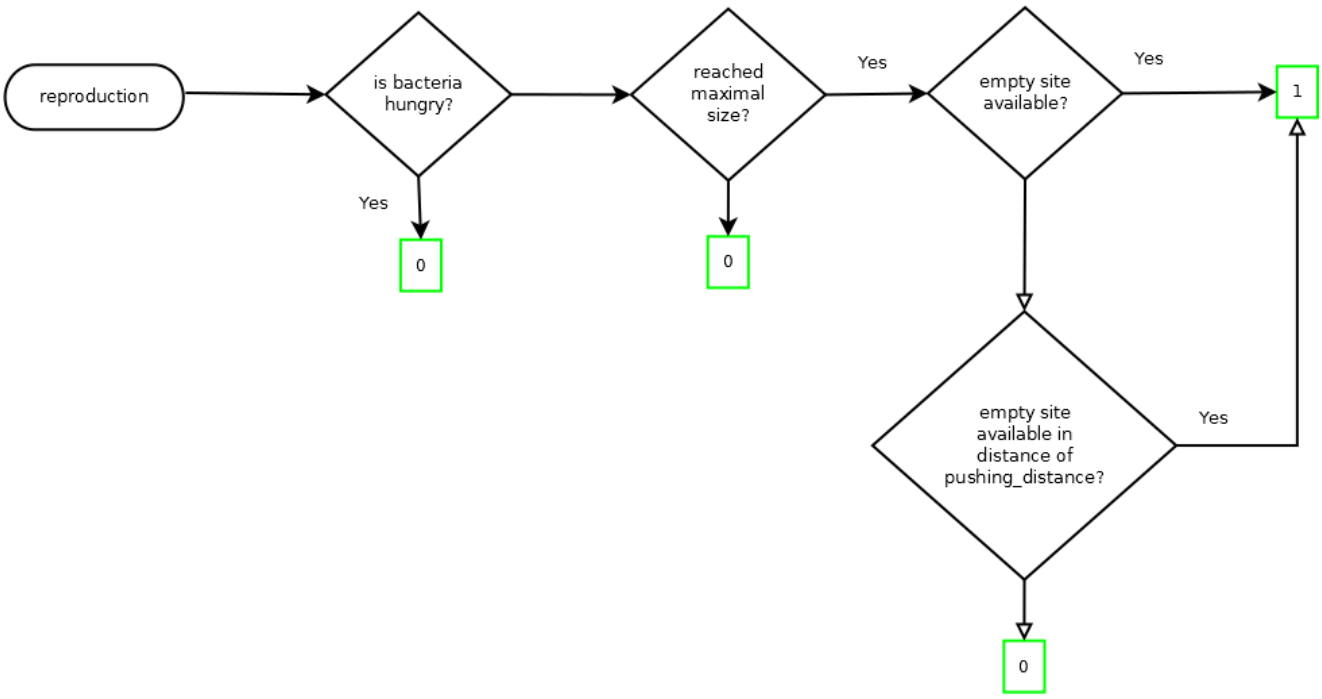

Image 5: Calculating reproduction rate

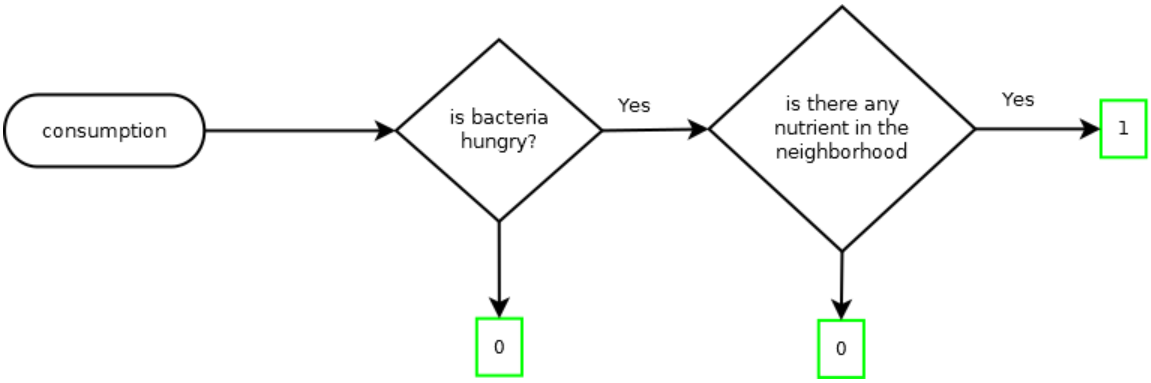

Image 6: Calculating consumption rate

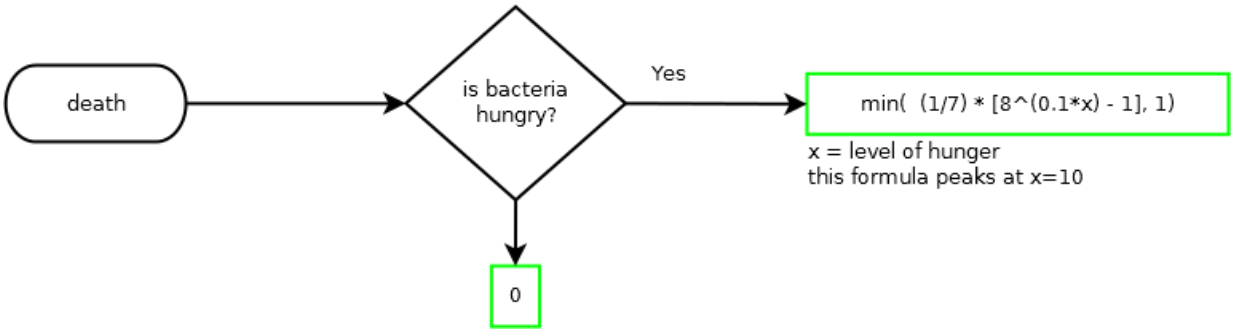

Image 7: Calculating death rate

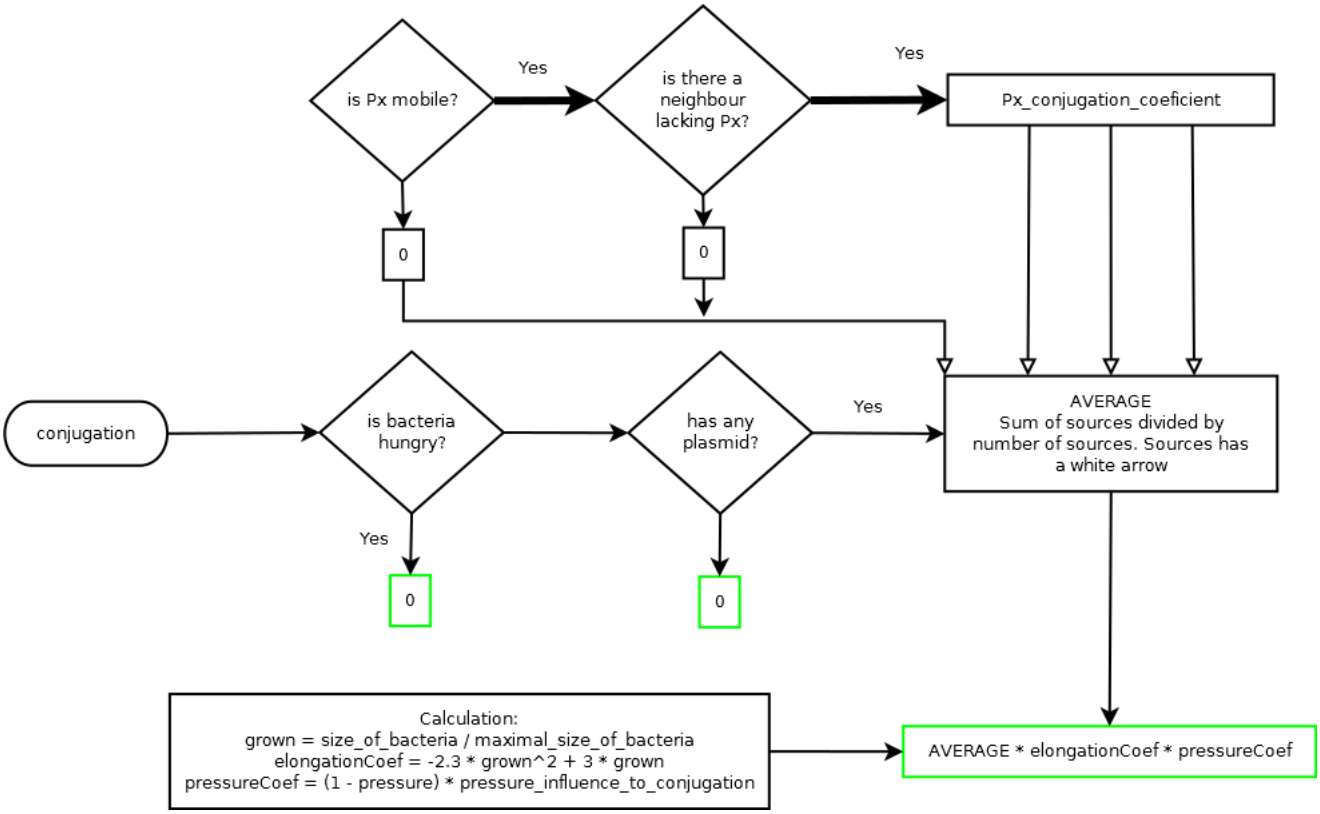

Image 8: Calculating conjugation rate

3.5 Features – Simulation events

Nutrient consumption – All bacteria must consume nutrients to regain its energy and building blocks for reproduction. Consumption of a nutrient is implemented such that each bacterium is looking for a piece of nutrient in its neighborhood. If there is one and the bacterium is starving, it consumes one piece. Starving bacteria have higher probability to die. Neighborhood is (by default) the set of sites at distance 1 from the bacteria. The distance is a parameter that can be tuned during the simulator validation. We do not expect, however, an important influence of this parameter on the population behavior.

Bacterial elongation – Each fed (non-starving) bacterium slightly elongates in each time step. Although this is not shown in visual representation, the length attribute is used as an input for calculation of action probabilities. Each reproducing bacterium halves its length. Starving bacteria do not elongate at the particular step. Starving bacteria without access to nutrient cannot elongate at all. Bacteria of length 10 are unable to elongate any more. All the bacteria have length 5 at the beginning of simulation. The maximum length constant 10 was chosen to cover the estimated time granularity and ratio of frequency of different events, based on the published experimental data.

Bacterial reproduction – Each fed bacteria that elongated at least 5 times (the constant again chosen due to the time granularity of events) has a high probability to reproduce (divide). An offspring is always located in neighboring sites. If there is no empty neighboring site, the bacteria pushes the other bacteria towards the empty site and thus creates a space for its offspring. The search for a near empty site is limited by a certain radius (under Euclidean metric) which can be fine-tuned during the model validation. We do not expect the radius to exceed the value of 15 sites, maybe even lower value would be suficient. For instance, for radius 6 the neighborhood contains already 112 sites. Let

us choose a random set of bacteria in a certain area. If there are no empty sites in their neighborhoods, it means that more than 99% space is occupied in that area. Therefore, the pressure must be rather high and the possibility to push a long column of bacteria is very unlikely. The remaining less than 1% of free sites should not affect the simulation results. The value of this parameter should be also verified by optical observation of growth experiments. Each offspring receives approximately one half of plasmids of each type, due to a certain random distribution.

Bacterial movement – Each bacteria has a certain (small) probability to move. The movement is implemented as swapping of two sites and it does not inflict metabolic burden. A swap of two occupied sites is not allowed, only a swap of an occupied site and an empty site is possible.

Bacterial death – Each starving bacteria has a certain probability (to be tuned during the model validation) to die. Dead bacteria are discarded from the grid.

Plasmid conjugation – Each bacterium can carry an unlimited amount of plasmid types. Each plasmid type is present in a number of copies ranging from 1 to the copy number. The copy number is the maximum amount of copies of the given plasmid in bacteria and it is a property of plasmid. One plasmid type is randomly selected during a conjugation event and one copy of that plasmid is transferred into neighboring bacteria. It is impossible to transfer a plasmid of the type already contained in the receiving bacteria. The probability of conjugation depend on further parameters as pressure and elongation (there are known relations between the division and conjugation). The conjugation takes one simulation step.

Plasmid loss – The plasmid loss event can occur during reproduction. There is a chance that an offspring does not get any copy of selected plasmid, due to the random distribution of plasmids.

4 Design concepts

Basic principles – the model is a designed as an IBM – Individual-Based Model. It works in discrete space (grid) and time. Each bacterium is represented as an independent entity with a set of state variables. Bacteria can form higher level entities – *colonies*. For all bacteria, possible events for each simulation step are first calculated (including conflict resolution) and then executed. Although these actions are performed sequentially, they can be considered as a simulated parallelism as each bacterium acts independently (except for conflict resolution). On a highly parallel machine the true parallelism can be easily employed.

The simulator contains an implementation of logic for conflict resolution. We use known rules of individual behavior to observe and measure emergent properties of the system.

Emergence – We want to observe plasmid infection spreading across bacterial colonies.

Adaptation – We reproduce the observed behavior using probability values for each possible action. Probabilities are being modified according to current bacteria state (starving / fed) and the environment in neighborhood (nutrient available / unavailable).

Objectives – Bacteria have simple objectives – to consume whenever possible, to reproduce as fast as it can and to conjugate whenever is possible (due to its internal state) to infect another uninfected bacteria.

Learning – Bacteria do not learn.

Prediction – We do not use any prediction models.

Sensing – Bacteria can sense nutrient in its neighborhood (directly neighboring site) and it can also sense uninfected bacteria in its neighborhood.

Interaction – Bacteria-bacteria interactions (competing for nutrient, pushing to reproduce, donor-receiver conjugation) and plasmid-plasmid interactions (entry exclusion: not reinfection of the same bacteria with the same type of plasmid Pa or Pb is allowed).

Stochasticity – Any simulated event has some probability to happen. This probability is either previously known constant value (such as reproduction rate) or a formula that contains variables whose evaluation depends on the environment and state variables.

Collectives – Bacteria are organized in colonies, but this is rather emergent property of the system. Bacteria itself are acting on their own but sometimes also exhibit altruist behavior. For example, in biofilms bacteria expel an oligosaccharide that is good for the colony but costly for each individual bacterium.

Observation – We render an image for each simulation time step. This image shows a current dish layout and colony growth. We also observe how many bacteria of each kind (according to contained plasmids and transconjugancy of the bacteria) the system currently contains.

5 Initialization

Initial state of the system consists of a set of colonies, nutrient and individual bacteria, together with a set of global numerical and logical parameters. These descriptors can be saved and loaded as a *project*.

Colony is defined as a mixture of bacteria in predefined ratios. Colony has its known location in the dish, its concentration of bacteria (measured in pct), and its radius (measured in number of sites). The individual colony composition and location of each bacterium is generated at the simulation start.

We also can define **individual bacteria** that do not belong to any colony yet. Such a bacterium is always at the same location and is always of the same type with same state variables.

Nutrient is defined by its location, radius and concentration. Actual nutrient amounts and distribution in defined location is generated at the simulation start due to a random distribution with these given parameters.

Due to stochasticity of the model, two runs of the same project could lead to different results. This behavior is desirable, for instance to confirm that the results of the simulator do not vary much in similarly defined initial conditions (and, hence, they are not sensitive to random noise).

6 Input / output

A project file and simulation attributes are the only input for the simulator. The project files are generated in the integrated Project designer that is part of user interface. Project files contains following data:

Colonies

- position, shape, ratios (how many bacteria of what type are in a colony)
- density (how much empty space is between bacteria in a colony)

Bacteria

- type (according to a plasmid that given bacteria carry)
- position in the dish (or assignment to a colony)

Plasmid

- Definition of which plasmid types are in the simulation. While it is possible to define any type of plasmid, plasmids Pa, Pc and Pc has special meaning in the simulation (such as specific coloring in the graphical output)
- Mobility (whether or not a plasmid is capable of being transferred using conjugation)

Nutrient

- position of nutrient area
- radius of nutrient area
- density (how many pieces are on an average in one site)

7 Submodels

There are no submodels in the proper sense. Events defined in Section 3 are executed as simple actions. There is however some logic to compute event probabilities.

7.1 Calculating bacterial rates

Reproduction – if bacteria are not starving and already reached its maximal size and are neighboring with an empty site, rate is 1 and Reproduce event is generated. If bacteria is not neighboring with an empty site but there is an empty site nearby, rate is 1 and ReproduceWithPushing event is generated. Otherwise rate is 0.

Movement – if bacteria are neighboring with an empty site, rate is 0.00001. Otherwise rate is 0.

Death – if a bacteria is starving, rate is smaller value from either $\frac{1}{7}\ (8^{0.1X} - 1)$ or 1 where X represents number of last simulation steps in a row when was a bacteria starving.

Consumption – if a bacteria is starving and there is a nutrient available in the neighborhood, rate is 1. Otherwise rate is 0.

Elongation – if bacteria is not starving and hasn't reached its maximal size, rate is 1. Otherwise rate is 0.

PlasmidLoss – depends on plasmid type. Rate is 0 % for Pa and 0,4 % for Pb due to recent experimental results in the Work package 4 of the project. Also a new plasmid Pd might be used in the future with the rate of 3,6 %.

Conjugation – An individual conjugation rates for Pa and Pb are attributes that can be defined in user interface as Pa_conjugation_coefficient and Pb_conjugation_coefficient. The final rate is average value of conjugation rates of each plasmid type contained in bacteria ($average\_of\_individual\_rates$) multiplied by elongation coefficient *elongCoef* and multiplied by pressure coefficient *pressureCoef*.

Formula used for conjugation:

$$grownPercent = \frac{sizeOfBacteria}{maximalSizeOfBacteria}$$
$$elongationCoef = -2.3 \times grownPercent^2 + 3 \times grownPercent$$
$$pressureCoef = (1 - pressure) \times pressure\_influence\_to\_conjugation$$
$$\boldsymbol{rate} = average\_of\_individual\_rates \times elongCoef \times pressureCoef$$

where *pressure* is estimated as the amount of neighbors in the neighborhood and *pressure_influence_to_conjugation* is a simulation attribute that can be defined in the user interface.

8 Experimental validation

The simulator contains a set of attributes that can be used to validate it with experimental results. These attributes are:

bacteria_max_length – how many growth cycles would be needed to reach maximum size of bacteria from the minimal size 1. Current default value is 10. However, in the model the default initial size of bacteria is 5, hence 5 steps are necessary to reach the maximum size. Assuming that mature bacteria has real maximal size of 1um, each unit of simulated bacterial length is then 0.1um.

bacteria_nutrientSeachRadius – how far can bacteria reach for a nutrient. Default value is 1 meaning that bacteria can consume nutrient from its own site and from neighboring sites in a distance of 1.

nutrientCoef – this is a multiplier of nutrient amounts. Default value is 1, meaning that this attribute has no efect. Changing it to higher values causes increasing amounts of initial nutrient in the dish.

pressure_influence_to_conjugation – how much does pressure influence conjugation. Higher values mean that conjugation will be suppressed or slowed down because of pressure conditions.

PushingDistance – How far are bacteria able to push the other bacteria to make empty site available for reproduction. The default value is 15 for performance reasons. This value was not validated yet.

plasmidPaConjugateCoef –this coefficient modify balance in conjugation speed between plasmids Pa, Pb and Pc. Increasing this value causes plasmid Pa to conjugate faster. Each of plasmid types Pa, Pb and Pc has its proper coefficient ranging from 0, unlimited from above. The probability that a

particular plasmid type is chosen during the conjugation event is calculated as its coefficient divided by the sum of coefficients of all plasmid types present in the cell. Default value for Pa is 1.

plasmidPbConjugateCoef – this coefficient modify balance in conjugation speed between plasmids Pa, Pb and Pc. Increasing this value causes plasmid Pb to conjugate faster. Default value is 1.

plasmidPcConjugateCoef – this coefficient modify balance in conjugation speed between plasmids Pa, Pb and Pc. Increasing this value causes plasmid Pc to conjugate faster. Default value is 0 because plasmid Pc is immobile and thus unable to being transferred during conjugation.

writeImageEvery – this attribute affects how often output images are generated. The default value is 1 which means that every simulation step generates output images. Setting this value higher increases speed of simulation and reduces amount of generated graphical data.

Further adjustable coefficients and settings are described in Sections 3 and 7. Some of these attributes and internal processes have been validated experimentally. In a sequence of experiments we tuned these attributes to match the behavior observed experimentally in a Petri dish. Some other attributes / processes have been taken directly from the literature. For example, we already incorporated results from [1] showing that bacteria are more susceptible to receive a plasmid at advanced stages of their growth cycle. This finding was incorporated into elongCoef coefficient that is part of formula for calculating the conjugation rate. We also use the plasmid loss coefficients for plasmids Pa and Pb that were described in [2]. These coefficients are used during each successful reproduction event and they are applied to each plasmid present in the reproducing bacteria. In the case of plasmid loss event, some of the daughter bacteria receive only those plasmids that were not marked as lost.

Some attributes are designed ad-hoc by observing results of experiments, mostly published in [1], [2], as the *bacteria_max_length*.

Finally, there still remain a few attributes to be introduced and examined when more experimental data are available. For instance, the speed of tetR blocking the plasmid replication and its subsequent loss is essential for the evaluation module. This parameter has only recently been estimated in lab experiments.

9 Results

The results obtained so far can be divided into three groups:

1. Design, construction and implementation of the simulator software based on the careful review of recent results by top research groups studying and modeling conjugation on surfaces, and the study of methodologies they use. Of course, the resources do not include only [1] and [2] but many other research papers and online sources. Based on this knowledge we decided to design the simulator as the IPS (Interacting Particle System) working in discrete space and time. Some parameters, constants and solutions (see, e.g, Section 7) have been inspired by these reviewed sources which speeded up both the simulator development and its validation. The simulator can be downloaded at the address http://dl.dropbox.com/u/35736140/bactosim-source-code.zip

2. Experimental validation of key simulator parameters using results published in [1], [2] has been done and key parameter values have been estimated for a particular experiment setup. More detailed insight into these parameters is described in Section 8. The validation process, however, cannot be considered completely finished yet as a new cross-validation should be performed after more experimental data are available.
3. Prediction simulations and their results, aiming to answer (partially) the questions imposed in Section 1. This research is in its early stage and only preliminary results are available yet. For example, these preliminary results confirm that most of the conjugation events happen at the edges of the colony where pressures are not so high. Also re-plating and remixing a sample into a new agar plate increase overall conjugation speed compared to leaving bacteria to grow on the original plate as noted in [2]. We also got better system convergence (more bacteria carrying plasmid Pc being infected with plasmid Pa or Pb) with initial ratios of bacteria carrying plasmid Pc being substantially higher than ratios of bacteria carrying plasmids Pa and Pb. More experiments must be performed to specify the optimal ratio under various initial conditions.

10 Future work

Repeated validation: Validation of the model is not a one-time process. After initial validation which has been done using the experimental data [1], [2], further validation will take place when more experiments are performed. It is possible that different settings of attributes can lead to the same results in one experiment, but to different results in another experiment. Therefore, after obtaining more data, the parameters will be cross-validated and eventually re-tuned. The same holds also for the prediction phase of the simulation – possible differences between the predicted and observed behavior of the population will be used to fine-tune the parameters.

Prediction use of the simulator has already started but much of the work remains to be done. It consist in an extensive sequence of simulated experiments with different initial setup of attributes, sizes of colonies, nutrients etc. Setup of the most promising simulations will be then verified in lab experiments and the obtained results will be compared with the simulation prediction. Possible differences will be used again to fine tune some simulation parameters. It is possible that several iterations will be necessary until stable and credible results are obtained.

Interpretation and analysis of the obtained results forms an important phase. When the model is correctly set, it can provide a large amount of fast and cost-effective simulated experiments. These experiments would allow to study the influence of various attributes and parameters of the experimental setup and their mutual relations. They can also lead to changes in some known formulas from the literature and to a development of new formulas describing relations between attributes of the conjugation processes. The most interesting results of this kind will have to be verified in the lab.

11 References

[1] Jose Seoane, Tatiana Yankelevich: An individual-based approach to explain plasmid invasion in bacterial populations. FEMS Microbiol Ecol. 75 (2011) 17-27.

[2] Irene del Campo, Raúl Ruiz, Ana Cuevas, Carlos Revilla, Luis Vielva, Fernando de la Cruz: Determination of conjugation rates on solid surfaces. Plasmid, in press, available online.

[3] http://dl.dropbox.com/u/35736140/bactosim-source-code.zip