Mathematical Biology and Bioinformatics. 2014. V. 9. № 2. P. 585–596.
URL: http://www.matbio.org/2014/Lashin_9_585.pdf.

========== **MATHEMATICAL MODELING** ==========

УДК: 573.22

HEC 2.0: Improved Simulation of the Evolution of Prokaryotic Communities

Lashin S.A.[*1,2], Klimenko A.I.[1,2], Mustafin Z.S.[1], Kolchanov N.A.[1,2], Matushkin Yu.G.[1]

[1]*Department of Systems Biology, Institute of Cytology and Genetics, Novosibirsk, 630090, Russia*
[2]*Faculty of Natural Science, Novosibirsk State University, Novosibirsk, 630090, Russia*

Abstract. Modeling and simulation of prokaryotes and prokaryotic communities are important for the development of modern fundamental, medical and biotechnological researches. Previously, we had developed a software package "Haploid evolutionary constructor", which models simultaneously describe several layers of biological organization: genetic, metabolic, population, ecological. Here we present a new version of the program, which includes the following major improvements: graphic user interface, parallel version of the computational core, and support of user-defined plugins. Plugins describe either changes of prokaryotic population size or cellular metabolism (gene networks). The graphic user interface components for the "Haploid evolutionary constructor" provide convenient visualization of data, model construction, setting up and control. High-performance versions of our software have been implemented using OpenMP and MPI technologies, and can be run at both desktops and MPI clusters. The software is available at the website http://evol-constructor.bionet.nsc.ru/ along with documentation and example models. The "Haploid evolutionary constructor" provides researchers the convenient tool for simulation of bacterial communities' evolution. The models of prokaryotic communities constructed with this software may be used for studying the fundamental principles of evolution, connecting various levels of biological organization, from genetic to ecological ones. It may also be used as an educational tool for the illustration of fundamental biological laws.

Key words: *simulation software, bacterial evolution, prokaryotic community*.

INTRODUCTION

Prokaryotes communities play a significant role in all major biogeochemical cycles. Typically, they live and evolve in communities: bacterial mates, biofilms or another complex spatially distributed multilayer structures [1, 2]. A number of prokaryotic species are uncultivated outside their natural communities. That is why mathematical modeling and computer simulations are of great concern in research of prokaryotes living and evolution. These theoretical tools are important for the development of modern fundamental, medical and biotechnological researches.

An agent-based modeling is a major approach in simulation software of prokaryotic organisms. Under this approach, populations are simulated as systems, which contain agents – separate models of individual organisms or groups of organisms possessing certain traits.

[*]lashin@bionet.nsc.ru

Each agent has its own unique history of interactions with environment and other agents. Agent-based modeling is widely used in ecological modeling, social dynamics, and simulation of evolutionary processes [3–5]. The models may show how behavior of single individuals within the bounds of local rules leads to formation of complex patterns including spatially distributed, like fish shoals, bird flights, swarms of insects and other biological communities [6–8].

The advantage of this approach is the fact that it allows to map the variety of features of a separate individual in the most flexible way, and, at the same time, it describes clearly the interactions between different individuals at micro level. Moreover, while such evolutionarily significant events as mutations and recombination are discrete and individual-oriented, they are naturally described via agent-based methods. The main disadvantage of these methods of simulations is a high computational complexity. However they require quite a lot of experimental data to describe a "portrait" model. In this regard, we proposed recently a multi-layer methodology for simulations of prokaryotic communities, and the software package "Haploid Evolutionary Constructor" (HEC) (http://evol-constructor.bionet.nsc.ru/) for the purpose of the reduction of computational complexity in simulations of high-diverse microbial communities as well as the dependency on amounts of experimental biological data.

Currently, there are varieties of software that simulate different aspects of prokaryotic dynamic life. For example, BacSim [9] intends simulating bacterial biofilms formation as a result of cellular processes. I.e., it integrates cellular processes into generalized population model. The basic object in BacSim model is a bacterial cell living in a spatially heterogeneous environment. While natural bacterial colonies may have a huge size (billions and more cells), they may be impossible to be simulated (e.g. in [9] they simulate population of about 4000 cells). Besides, only one specie cells are considered, that constrains simulation of genetic diversity of a community.

Micro-Gen Bacteria Simulator [10], simulates the lifecycle of cultivated bacteria and their interactions with various molecules, primarily consider ecological, population and metabolic levels of organization, while the genetic processes and heredity are not described. The software may be used at high-performance clusters as the used model is scalable. It provides simulations of high-concentrated bacterial communities ($> 10^7$ bacteria/ml, which is close to natural concentrations).

In contrast the AEvol and RAEvol programs [11, 12], highlight genome evolution processes in bacterial populations, and do not deal with population or ecological processes like e.g. substrates exchange. However, one of the major challenges in the modern science is the conversion from the reductionism in considering biological objects and processes to their system understanding.

The AgentCell software [13] was developed to simulate stochastic fluctuations at cell-cell interactions. Using this software, authors simulated chemotactic response to a linear gradient of attractant of individual swimming bacteria in 3D environment. In this model, each bacterial cell is an individual object including its own chemotaxis gene network, motor and flagellum.

While the majority of the programs for simulation of bacterial communities are focused on the description of separate levels of biological organization, our software, Haploid Evolutionary Constructor (HEC) provides modeling and simulation of communities at the following levels: genetic, metabolic and population. The approach underlying the HEC is focused on the simulation of population-genetic, metabolic and ecological factors, and attempts to substitute the reductionism ideas by synthetic ones, which is the mainstream of the systems biology.

IMPLEMENTATION

The initial Haploid evolutionary constructor program was published in 2007 [14]. Since then we have been adding new functions to the program. In particular, bacteriophages [15] and gene networks [16] layers have been added. In version 2.0 we present the following

improvements in the HEC: support of high performance computations, graphical user interface, and support of the plugin system. The computational core is implemented in the C++ programming language with the use of Boost [17] and OpenMP [18] libraries (for the high performance version). The GUI is implemented in the Java programming language (Java Development Kit 1.6 Standard Edition). Test models are located in the "scripts" directory of the distribution.

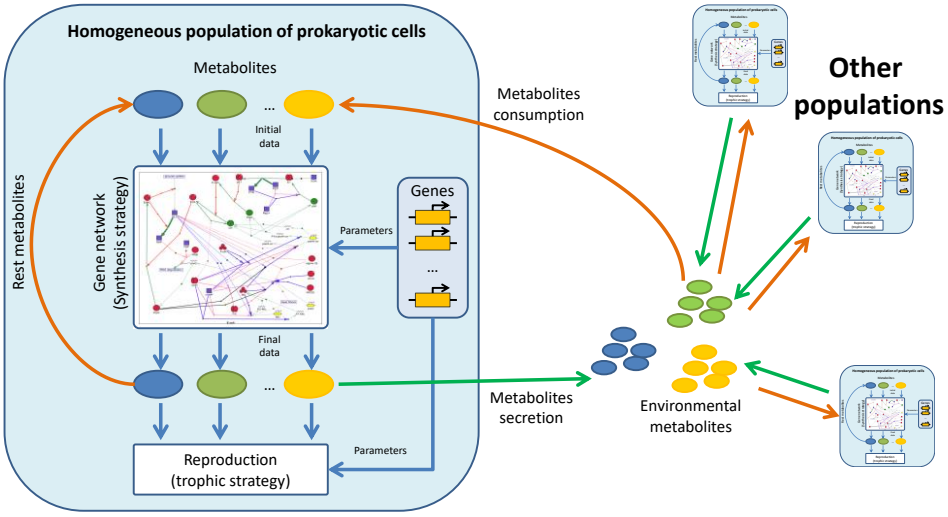

Fig. 1. Principal diagram of the HEC model.

The main concept of the HEC model is presented in Fig. 1. The key object of the model is a population of prokaryotic cells. Cells belonging to the same population have identical metabolic function – all of them consume the same substrates and produce the same products. In HEC, the metabolic function of cells in a population is defined by two submodels: the gene network of metabolites synthesis (in HEC, it is called *synthesis strategy*), and the rule of population reproduction and alleles' competition/selection via metabolites utilization (so called *trophic strategy* in HEC). The numerical parameters of those submodels are assumed as *genes* in our methodology. In the case of synthesis strategies, genes determine the rate constants for certain processes in a gene network; in trophic strategies genes determine utilization rate constants for metabolites and parameters for their influence on population size. Hence, the genome of a separate cell (which we may assume to be a generalized genome of *monomorphic population* of cells) is represented uniformly as a set of constants, the first part of which concerns to gene network, the second part – to trophic strategy. In addition to the monomorphic state (all cells are genetically identical), a population may be in *polymorphic* state. In this case there are several allelic variants for some (or for all) genes. The detail description of the modeling of genetic polymorphism via so called *genetic spectra arithmetics* was published in our previous studies [19]. Population-environment interactions are realized via metabolites transport in-/outside cells (Fig. 1).

Working with the HEC includes two actions: the description/setup of a model and the simulation runs of a model.

1. Model setup

At this stage, a user defines the model structure and sets up parameters. The main objects of the HEC model are prokaryotic population and environment.

Population is characterized by its size, metabolic functionality and genetic diversity. The metabolic functionality is determined by sets of substrates utilized and synthesized by a population along with the rules of utilization (and cells reproduction) and synthesis. In the HEC, these rules are called trophic strategies and synthesis strategies, respectively. The

current version of the HEC has six trophic strategies and two synthesis ones. Users may also develop their own strategies via plugins and the HEC open API (see the section “Extension with user-defined plugins” below). Utilization and synthesis efficiencies are defined as numerical parameters, each of which is assumed to be a “gene”. The set of distribution for all genes variants (alleles) in a population is called “generalized population genome”.

Its further evolution is described with the use of so called genetic spectra arithmetics [19]. The user can also set up the general metabolism parameters of a population (mean utilization/ synthesis rates), death and flow parameters etc. The environment is characterized by its volume, concentrations of environmental and inflow substrates (specific and nonspecific), rate of flow and presence of populations. The resulted model of prokaryotic community may be schematically represented via graph of trophic interactions (section “Graphic user interface”, Fig. 3). Default parameter values were estimated on the base of *E. coli* data [20] and described in detail in [16]. The user can edit the additional probability parameters for stochastic simulation of evolution (frequencies of mutations, horizontal gene transfers and losses of genes).

2. Simulation process

The following regular actions are simulated per iteration (Fig. 2):

— consumption of environmental substrates by populations;
— substrates utilization and reproduction of populations;
— substrates synthesis and secretion into environment;
— environmental flow.

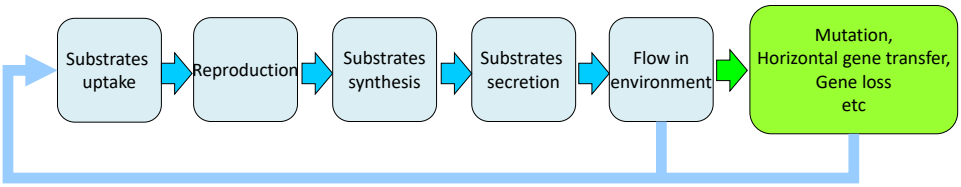

Fig. 2. Simulation steps per iteration.

After the completion of iteration, the user may simulate additional action (or some actions at a time):

— mutation in a gene in some cell(s) of a population;
— horizontal transfer of gene(s) from a cell of donor population into a cell of acceptor population. It may lead to the origin of a novel population [19];
— loss of gene(s) in some cell(s) of a population (also may lead to the origin of a novel population);
— change of substrates inflow concentration and/or change of flow rate.

There are two modes for model simulations – general and stochastic. In general mode changes of model structure (see above) can be only defined by user, in stochastic mode they may occur randomly, according to probabilities set up.

3. Graphical user interface

There is a graphical user interface (GUI) in the HEC v.2 package. GUI provides visualization of all characteristics for every object of the HEC model: population size dynamics, environmental concentrations of substrates, trophic interactions in a community, and alleles’ distributions for every gene in a population. GUI is distributed as standalone application and consists of the following frames/windows (Fig. 3):

1. frame displaying structure of a prokaryotic community, includes the tabs:
   a) community structure as a graph of substrate-products interactions (Fig. 4,a);

b) community structure as a graph of interpopulation trophic interactions (Fig. 4,b).

2. frame displaying model dynamics and current model state, includes the tabs:
   a) population dynamics (Fig. 5,a);
   b) dynamics of environmental metabolites concentrations (Fig. 5,b);
   c) distributions of alleles for all genes in populations (Fig. 5,c).
3. frame displaying a model written in special HEC script language (which described in detail in [16]), includes the tabs:
   a) script describing only the state of the current model (only declare part);
   b) script describing the whole model from its beginning including all calculation commands.
4. control frame for the simulations, includes buttons performing actions described above (section “Simulation process”).

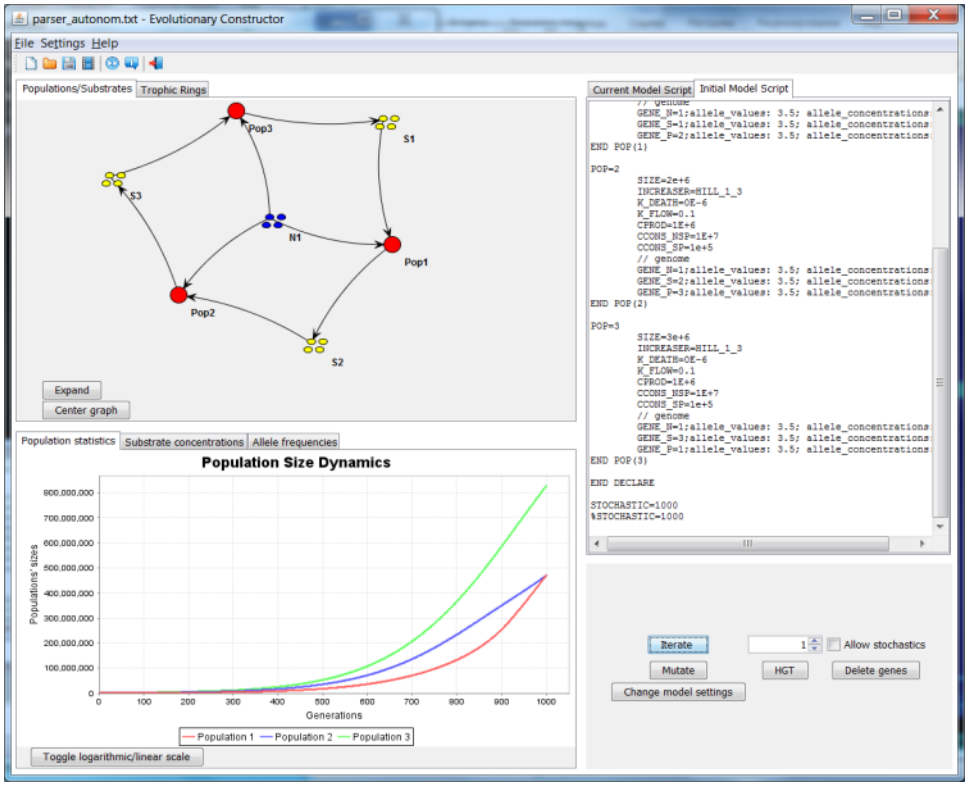

Fig. 3. Overview of the HEC graphic user interface.

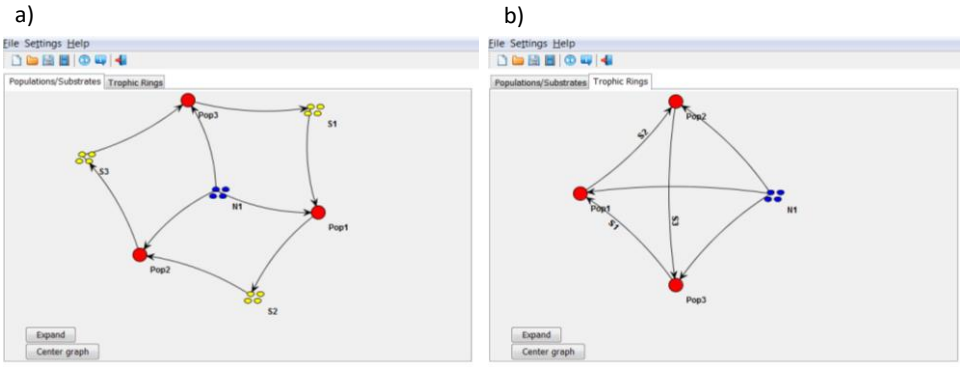

Fig. 4. Structure of prokaryotic community displayed as: a) graph of substrate-product interactions; b) trophic graph.

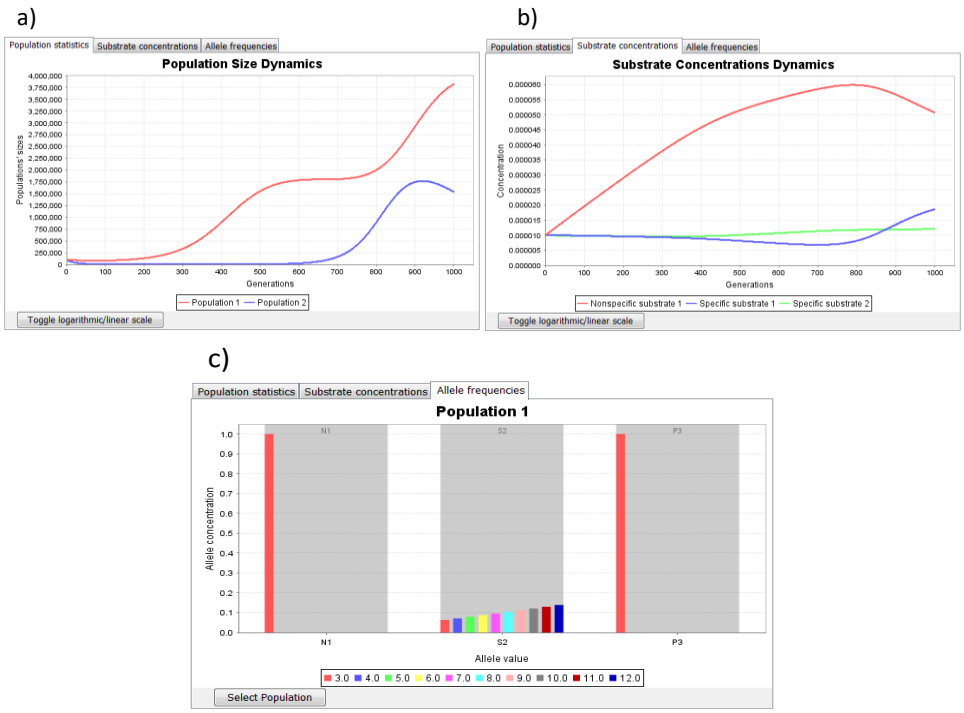

Fig. 5. a) Population dynamics of the community; b) substrates dynamics; c) alleles frequencies of the Population 1.

The final model and the simulation scenario can be saved in a script-file using the HEC script language.

4. Extension with user-defined plugins

At present, there are two types of plugins in the HEC: trophic strategy plugins and synthesis strategies plugins. Plugins should be built as dynamic libraries (*.dll on Windows, *.so on Linux). In order to make personal plugins, one should develop subclasses of the abstract classes `Increaser` (for trophic strategies) and `SynthesisStrategy` (for synthesis strategies). The detail plugin instructions are available at the HEC web-site (http://evol-constructor.bionet.nsc.ru) along with necessary include files and *.dll/*.so templates.

4.1. Implementing new trophic strategy

The `Increaser` class is located in the “Increaser.h” file:

```
class Increaser {
public:
   virtual double calculate(double curPopSize,
         int nGNum,
         int cGNum,
         int dGNum,
         const vector<float>& curMonoGenome,
         const vector<int>& curMonoCodes,
         const vector<double>& nSubstrates,
         const vector<double>& sSubstrates,
         float deathCoef, float flowCoef,
         double totalPopSize) = 0;
};
```

A subclass of the `Increaser` class should implement the method `calculate` defining the new size of a monomorphic population. For this purpose, one can operate the following parameters:

- `double curPopSize` – size of the current monomorphic population,
- `int nGNum` – number of utilization efficiency genes for nonspecific substrates,
- `int cGNum` – number of utilization efficiency genes for specific substrates,
- `int dGNum` – number of synthesis efficiency genes for specific substrates,
- `const vector<float>& curMonoGenome` – alleles values (rate constants) for utilization and synthesis efficiencies indexed in the following order (`[0,nGNum-1]` – nonspecific substrates utilization rates, `[nGNum,nGNum+cGNum-1]` – specific substrates utilization rates, `[nGNum+cGNum, nGNum+cGNum+dGNum-1]` – specific substrates synthesis rates,
- `const vector<int>& curMonoCodes` – substrates indices in correspondence with `curMonoGenome` (explanation in Fig.6),
- `const vector<double>& nSubstrates` – nonspecific substrates amounts (molecules) consumed by all cells of a population,
- `const vector<double>& sSubstrates` – specific substrates amounts (molecules) consumed by all cells of a population,
- `float deathCoef` – death coefficient,
- `float flowCoef` – flow coefficient,
- `double totalPopSize` – size of the whole community.

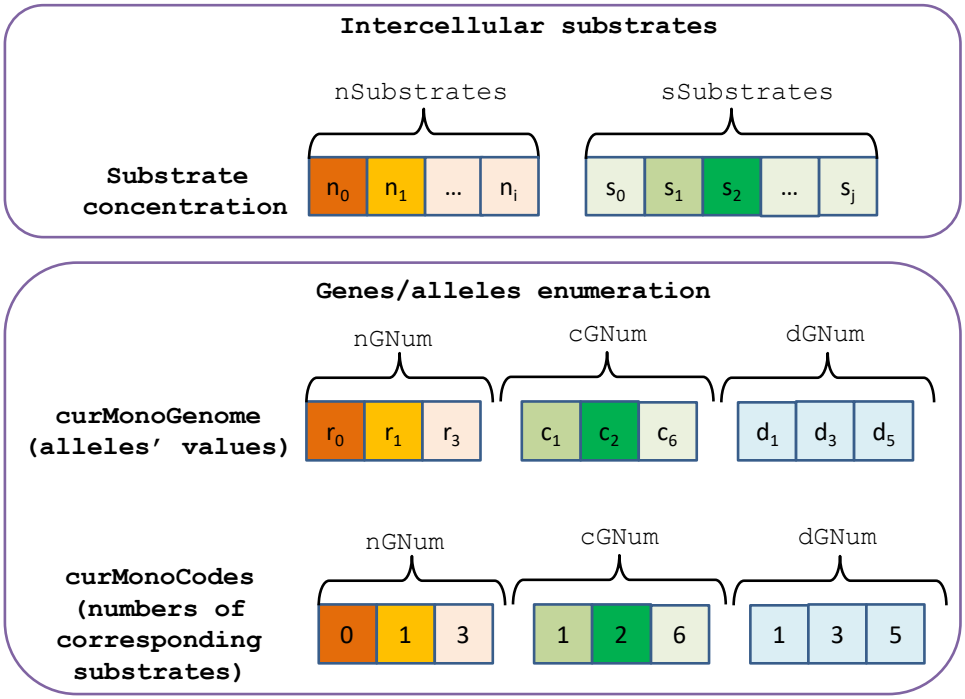

Fig. 6. Correspondence between allele values (`curMonoGenome`) and substrate numbers (`nSubstrates`, `sSubstrates`) via substrate indices (codes at `curMonoCodes`). For example, the utilization rate for nonspecific substrate N1 (which concentration is n_1) is r_1, as `curMonoGenome` [`curMonoCodes` [1]] == r_1 and `nSubstrates` [`curMonoCodes` [1]] == n_1;

The return value of the method is the new population size (`double`).

4.2. Implementing new synthesis strategy

The `SynthesisStrategy` class is located in the "SynthesisStrategy.h" file:

```
class SynthesisStrategy {
public:
   virtual void synthesize(_PolymorphPopulation* p) = 0;
};
```

A subclass of this class should implement the method `synthesize`, defining the new values of synthesized specific substrates. The class `_PolymorphPopulation` (located in "_PolymorphPopulation.h" file) provides the functionality for getting necessary population data and setting new values of the synthesized substrates:

```
class _PolymorphPopulation
{
public:
   virtual ~_PolymorphPopulation(void){}
virtual std::vector<GeneticSpectrumVector>
getProdSpecGenes() const = 0;
virtual std::vector<double> getSSCaptured() const = 0;
virtual std::vector<double> getNSCaptured() const = 0;
virtual double getMeanCellProductivity() const = 0;
virtual double getPopulationSize() const = 0;
virtual void setSynthesizedSubstrate
(int subNum, double value) = 0;
};
```

The only nontrivial data type here is the class `GeneticSpectrumVector` (located in "GeneticSpectrumVector.h" file), which describes the distribution of alleles in a population. Simple synthesis strategy, previously described in our studies [19], is exampled in the file "SynthesisStrategySimple.cpp".

5. High performance version of the computational core

Using the profiler Intel Parallel Amplifier [21], we have found the only function to be excessively time-consuming in simulations of communities of high genetic diversity (10^6–10^8 unique allelic combinations). This is the function of population size change. The main idea of this function is to look over all allelic combinations in a population (stored in genetic spectra) to calculate specific growth for each combination.

At present, there are two high-performance versions of the HEC: first one uses MPI (only available for the console version), the second uses OpenMP (available for both console and GUI versions). MPI version uses `MPI_Bcast` and `MPI_Reduce` functions to share computational tasks. All processes have almost equal load (minor variations occur if number of possible allelic combinations is aliquant to the number of processes). In OpenMP version, the whole populations are distributed in parallel threads, that minimizes number of memory reads/writes but additionally requires number of parallel threads to be aliquot to the number of populations in a community (otherwise, some threads may stand idle).

The MPI version may also be used on shared memory systems. On Microsoft Windows, it is required to install Microsoft HPC Pack. In order to run this version, one should use the command prompt and type `mpiexec -n N hec_mpi.exe script.txt` command, where `N` is the number of parallel processes; `script.txt` is the name of model script.

We measured parallelization efficiency on Novosibirsk cluster supercomputer NCS 30-T [22] (6-core processes X5670 2.93 GHz (Westmere)) for the MPI version. OpenMP version was measured on laptop (Amd Phenom II x6 1055T). Model scripts used for load testing ~10^6–10^8 allelic combinations in a community are listed in Additional file 1. Results of load testing are shown in Table 1 (NCS 30-T) and Table 2 (laptop).

Table 1. Measures of parallelization efficiency on Novosibirsk cluster supercomputer NSC 30-T. Standard deviation was rounded up

Number of processes	Calculation time (mean), hh:mm:ss	Parallelization efficiency (mean)	Acceleration (mean)	Standard deviation, sec
1	8:02:26	1	1	249
2	4:10:24	0.9633	1,9266	65
4	2:05:20	0.9623	3,8492	188
8	1:02:03	0.9718	7,7744	4
16	0:31:06	0.9695	15,512	3
24	0:20:48	0.9664	23,1936	3
36	0:13:46	0.9734	35,0424	1
64	0:07:52	0.9582	61,3248	2
96	0:05:13	0.9633	92,4768	1
144	0:03:32	0.9481	136,5264	1
264	0:02:03	0.8914	235,3296	2

Table 2. Measures of parallelization efficiency (MPI) on 6-core laptop (Amd Phenom II x6 1055T)

Number of allelic combinations	Number of generations	Computational time (sec.) – parallelization efficiency Number of parallel threads					
		1	2	3	4	5	6
1000	25000	53 sec	45 sec – 58%	44 sec –4 0%	44 sec – 30%	45 sec – 23%	52 sec – 17%
5000	10000	73 sec	46 sec – 79%	38 sec – 64%	34 sec – 54%	33 sec – 44%	37 sec – 33%
10000	5000	68 sec	39 sec – 87%	31 sec – 73%	27 sec – 63%	25 sec – 54%	27 sec – 42%
100000	500	85 sec	48 sec – 89%	35 sec – 81%	28 sec – 76%	24 sec – 71%	23 sec – 62%
1000000	50	162 sec	88 sec – 92%	61 sec – 89%	48 sec – 84%	40 sec – 81%	37 sec – 73%
10000000	4	199 sec	101 sec – 99%	69 sec – 96%	55 sec – 90%	46 sec – 87%	44 sec – 75%

For those load tests, we have obtained almost linear acceleration in the MPI console version. Using the NCS 30-T cluster, we have reduced the computational time from 8 hours (1 process) to 2 minutes (264 processes on 22 computational nodes, see Additional file 2).

In OpenMP version, the acceleration may be observed even for communities of relatively low genetic diversity (~100 allelic combinations and more) (Table 3).

Table 3. Measures of parallelization efficiency (OpenMP) on 6-core laptop Amd Phenom II x6 1055T

Number of allelic combinations	Number of generations	Computational time (sec.) – parallelization efficiency Number of parallel threads			
		1	2	4	8
1000	25000	49 sec	39 sec – 63%	37 sec – 33%	35 sec – 18%
5000	10000	65 sec	41 sec – 80%	29 sec – 56%	28 sec – 29%
10000	5000	59 sec	35 sec – 84%	23 sec – 64%	21 sec – 35%
100000	500	75 sec	49 sec – 77%	23 sec – 82%	21 sec – 45%
1000000	50	147 sec	81 sec – 91%	45 sec – 82%	34 sec – 54%
10000000	4	186 sec	99 sec – 94%	73 sec – 64%	48 sec – 48%

Increasing genetic diversity of a community, we get more and more efficient parallelization tending to linear. It should be noted that actual acceleration on OpenMP may exceed number of physical processors. For instance, using 6-core processor laptop, we ran OpenMP version with 10 threads and obtained 8-fold acceleration.

RESULTS AND DISCUSSION

As demonstration model, we have constructed the “poisoner-prey” model which is the variation of the classic predator-prey model [23]. In this model, the community consists of two populations P1 and P2, producing substrates S2 and S1, respectively. S1 activates growth of P1 while S2 inhibits P2 (Fig. 7).

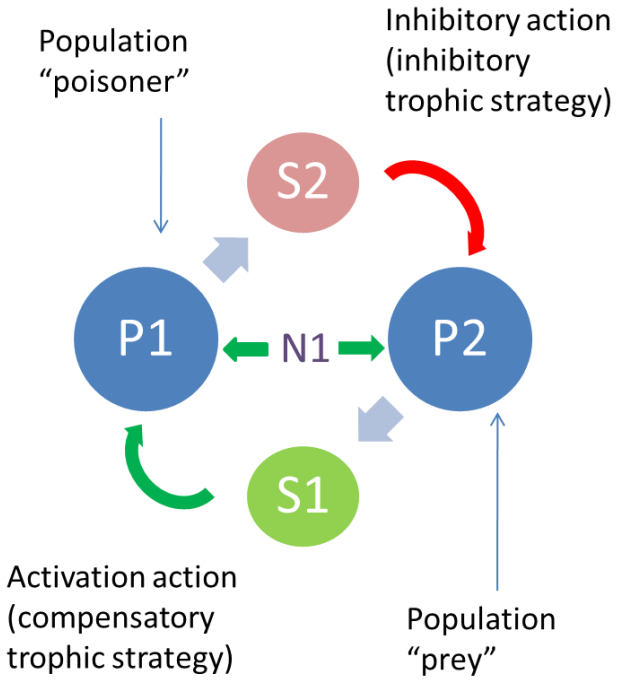

Fig. 7. The structure of trophic relations in the community. See text for details.

Nonspecific substrate N1 is utilized by both populations. Depending on relations between utilization efficiency for S1 in cells of P1 and sensitivity to S2 in cells of P2, there may be various dynamics of the community: stable/decay oscillations, stationary states and death of the whole community. If the populations are genetically polymorphic (gene for S1 utilization in P1 cells and/or gene of S2 sensitivity in P2 cells), one may observe directional selection: mean utilization efficiency for S1 is being increased in P1 cells, while mean sensitivity for S2 is being decreased in P2 cells. The creation of the model is described in detail in Additional file 3, while the script with the same model – in Additional file 4.

The models of prokaryotic communities constructed with the HEC may be used for studying the fundamental principles of evolution, connecting various levels of biological organization, from genetic to ecological ones. The HEC may also be used as an educational tool for the illustration of fundamental biological laws. Finally, we hope that in future we might use the HEC for the tasks of microbial biotechnology.

We have performed optimization of computational algorithm of the HEC. The higher genetic diversity has a prokaryotic community, the more effective the optimization is (almost linear acceleration was obtained). We assume such communities to be most interesting for studiers, and hope that the HEC and its high-performance versions presented in this paper will allow users to explore more complex and variable models of biological processes in prokaryotic communities, and therefore, to get advances in evolutionary biology. One can also use the HEC to study competition and evolution in gene networks populations, including adaptations of gene networks parameters to particular environmental or cellular conditions. Model of a gene network may be described in the HEC in terms of synthesis strategies, while criterion of optimality and selection mechanisms – in terms of trophic strategies.

The study was supported by the following grants: Budget Project VI.61.1.2, RFBR 13-04-00620.

REFERENCES

1. Webb J.S., Givskov M., Kjelleberg S. Bacterial biofilms: prokaryotic adventures in multicellularity. *Current opinion in microbiology*. 2003. V. 6. № 6. P. 578–585.

2. Stoodley P., Sauer K., Davies D.G., Costerton J.W. Biofilms as complex differentiated communities. *Annual Reviews in Microbiology*. 2002. V. 56 № 1. P. 187–209.
3. Grimm V., Berger U., Bastiansen F., Eliassen S., Ginot V., Giske J.,Goss-Custard J., Grand T., Heinz S., Huse G. et al. A standard protocol for describing individual-based and agent-based models. *Ecological modelling*. 2006. V. 198 № 1. P. 115–126.
4. Macy M., Willer R. From factors to actors: Computational sociology and agent-based modeling. *Annual review of sociology*. 2002. V. 28. P. 143–166. URL: http://www.jstor.org/stable/10.2307/3069238 (accessed 18.11.2014).
5. DeAngelis D.L., Mooij W.M. Individual-based modeling of ecological and evolutionary processes. *Annual Review of Ecology, Evolution, and Systematics*. 2005. P. 147–168.
6. Hoare D., Couzin I., Godin J-G., Krause J. Context-dependent group size choice in fish. *Animal Behaviour*. 2004. V. 67. № 1. P. 155–164.
7. Grimm V., Revilla E., Berger U., Jeltsch F., Mooij W.M., Railsback S.F., Thulke H-H., Weiner J., Wiegand T., DeAngelis D.L. Pattern-oriented modeling of agent-based complex systems: lessons from ecology. *Science*. 2005. V. 310. № 5750. P. 987–991.
8. Olfati-Saber R. Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory. *Automatic Control, IEEE Transactions on*. 2006. V. 51. № 3. P. 401–420.
9. Kreft J.U., Booth G., Wimpenny J.W.T. BacSim, a simulator for individual-based modelling of bacterial colony growth. *Microbiology*. 1998. V. 144. № 12. P. 3275–3287.
10. Murphy J.T., Walshe R. Modelling Antibiotic Resistance in Bacterial Colonies Using Agent-Based Approach. In: *Understanding the Dynamics of Biological Systems*. Eds.: Dubitzky W., Southgate J., Fuß H. New York: Springer Science & Business Media, 2011. P. 131–154.
11. Knibbe C., Mazet O., Chaudier F., Fayard J-M., Beslon G. Evolutionary coupling between the deleteriousness of gene mutations and the amount of non-coding sequences. *Journal of Theoretical Biology*. 2007. V. 244. № 4. P. 621–630.
12. Beslon G., Parsons D.P., Sanchez-Dehesa Y., Peña J-M., Knibbe C. Scaling laws in bacterial genomes: a side-effect of selection of mutational robustness? *Biosystems*. 2010. V. 102. № 1. P. 32–40.
13. Emonet T., Macal C.M., North M.J., Wickersham C.E., Cluzel P. AgentCell: a digital single-cell assay for bacterial chemotaxis. *Bioinformatics*. 2005. V. 21 № 11. P. 2714–2721.
14. Lashin S.A., Suslov V.V., Kolchanov N.A., Matushkin Y.G. Simulation of coevolution in community by using the "Evolutionary Constructor" program. *In Silico Biology*. 2007. V. 7. № 3. P. 261–275.
15. Lashin S.A., Matushkin Y.G., Suslov V.V., Kolchanov N.A. Evolutionary trends in the prokaryotic community and prokaryotic community-phage systems. *Russian Journal of Genetics*. 2011. V. 47. № 12. P. 1487–1495.
16. Lashin S.A., Matushkin Y.G. Haploid evolutionary constructor: new features and further challenges. *In Silico Biology*. 2012. V. 11. № 3. P. 125–135.
17. *Boost C++ Library*. URL: http://www.boost.org/ (accessed 18.11.2014).
18. *The OpenMP® API specification for parallel programming*. URL: http://openmp.org/wp/ (accessed 18.11.2014).
19. Lashin S.A., Suslov V.V., Matushkin Y.G. Comparative Modeling of Coevolution in Communities of Unicellular Organisms: Adaptability and Biodiversity. *Journal of Bioinformatics and Computational Biology*. 2010. V. 8. № 3. P. 627–643.
20. Sundararaj S., Guo A., Habibi-Nazhad B., Rouani M., Stothard P., Ellison M., Wishart D.S. The CyberCell Database (CCDB): a comprehensive, self-updating, relational database to coordinate and facilitate in silico modeling of *Escherichia coli*. *Nucleic acids research*. 2004. V. 32. № suppl 1. P. D293–D295.

21. *Intel Parallel Amplifier*. URL: http://software.intel.com/intel-parallel-studio-xe (accessed 18.11.2014).
22. *Novosibirsk cluster supercomputer NCS 30-T*. URL: http://www2.sscc.ru/HKC-30T/HKC-30T.htm (accessed 18.11.2014).
23. Lotka A.J. Contribution to the Theory of Periodic Reactions. *Journal of Physical Chemistry*. 1910. V. 14. № 3. P. 271–274.

Received December 04, 2014.
Published December 29, 2014.