УДК 573.22

HIGH-PERFORMANCE SIMULATIONS OF POPULATION-GENETIC PROCESSES IN BACTERIAL COMMUNITIES USING THE HAPLOID EVOLUTIONARY CONSTRUCTOR SOFTWARE

© 2014 г. **Z.S. Mustafin[1], Yu.G. Matushkin[1, 2], S.A. Lashin[1, 2]**

[1] Institute of Cytology and Genetics SB RAS, Novosibirsk, Russia,
e-mail: lashin@bionet.nsc.ru;
[2] Novosibirsk National Research State University, Novosibirsk, Russia

Received October 17, 2014. Accepted for publication November 27, 2014.

Three high-performance versions of the Haploid Evolutionary Constructor program are presented (http://evol-constructor.bionet.nsc.ru). The software was designed for simulating the functioning and evolution of microbial communities. These high-performance versions are to be run on systems with shared and distributed memory, using CPU and/or GPU. Almost linear acceleration has been achieved on clusters and multi-core CPU. On GPU systems, the simulation time was reduced to several minutes (dozens of hours on CPU).

Key words: Microbial community, parallelization, simulation, evolution, optimization.

INTRODUCTION

Simulation of evolutionary processes occurring in bacterial communities is one of the vital tasks of modern systems biology. Bacteria perform a vast majority of processes in nature. Many bacterial species are utilized to meet human needs. However, some bacterial communities reach such a high genetic diversity and huge population size that they cannot be investigated under laboratory conditions. Theoretical studies including mathematical modeling and simulation may be helpful in these cases. Information technology development has led to the appearance of a variety of programs devoted to modeling and simulation of various aspects of bacterial communities' life. Evolution and functioning of such communities in certain conditions (including infeasible for laboratories) can be modeled for purposes of medicine, fundamental and applied science.

In recent years, many papers on modeling and simulation of various features of bacterial communities have been published. Some papers are focused on the biological sense of simulation results, such as interconnections between individual and populational features of communities and their members (Kutalik *et al.*, 2005) or mechanisms of biodiversity sustainability in various fitness landscapes and at various mutation rates (Beardmore *et al.*, 2011). Others studied mathematical and programming features (Ashlock, McEachern, 2011; Bihary *et al.*, 2012). Applicability and advantages/disadvantages of agent-based (DeAngelis, Mooij, 2005) approaches, or cellular automata (Esteban, Rodrнguez-Patyn, 2011) have been also analyzed and compared with classical ODE and PDE equations. In spite of the multitude of modeling methods and software packages for simulation of bacterial communities, most of them consider a system under study at only one level of biological organization. Furthermore, few of them use modern technologies for high-performance computations.

This study is dedicated to the development of high-performance methods for simulating the functioning and evolution of bacterial communities (more generally, communities of unicellular haploid microorganisms). The method has been implemented as part of the Haploid Evolutionary Constructor software package (hereafter referred to as the HEC, http://evol-constructor.bionet.nsc.ru). HEC models are multiscale, and they include submodels describing different levels of biological organization: genetic, metabolic, population, and ecological (Lashin *et al.*, 2011; Lashin, Matushkin,

2012). Such composite models consume a lot of computational resources, especially in the case of communities of extremely broad genetic diversity (about 10^8 various allelic combinations in a population considering 10–100 model genes), which results in long simulation time. The paper also presents high-performance algorithms for HEC and test results. Three high-performance implementations have been made: OpenMP (http://openmp.org), MPI (http://www.open-mpi.org), and CUDA (http://www.nvidia.com/object/cuda_home_new.html).

HEC software package

HEC uses the multiscale modeling approach (Ayton *et al.*, 2007; Martins *et al.*, 2010). Four layers of biological organization are considered: genetic, metabolic, populational, and ecological (Lashin, Matushkin, 2012). Models of every layer can be implemented with various mathematical techniques (differential equations, automata, graphs, etc.). For each layer, libraries of submodels are released (Lashin, Matushkin, 2012). Notably, models of gene networks can also be implemented as HEC plugins. Such a multilayered approach allows users to study various aspects of a bacterial community within an integral framework.

Polymorphic population is described via the "*generalized population genome*" and the genetic spectrum technique (Lashin *et al.*, 2010), which affords a valuable decrease of the computational time as compared to classical agent-based approaches. This method also ensures comparable accuracy. An organism (cell) is characterized by a set of traits, each of which determines the process of either synthesis or utilization of a particular metabolite (substrate). In HEC, the whole network of those processes is assumed to be a "gene network" of the cell. Such a "gene network" can be formally implemented by using, for example, differential equations. Parameters of such a gene network are assumed to be *genes*, whereas particular values of these genes are assumed to be *alleles*. Cells belonging to the same population may possess different allelic combinations (ACs). The total number of ACs in a population characterizes its genetic diversity. Various ACs may be differently efficient in substrate synthesis and/or utilization, which results in different fitnesses and reproduction rates of subpopulations in the entire polymorphic population. HEC allows simulation of mutations, horizontal transfer of genes and gene loss. The last two change the set of metabolic reactions and, thereby, the gene network of a cell, generating a new strain/species. This feature allows HEC to model speciation and evolution of biodiversity in the community, which can be simulated either in complete-mixed or in spatially distributed environments.

The variation in reproduction rates depending upon both genetic and environmental factors allows us to simulate a wide range of evolutionary modes including neutral evolution. Other features of HEC are the simulation of phage infections (Lashin *et al.*, 2011) and gene networks (Lashin, Matushkin, 2012). Integration of the gene network concept with the HEC opens exciting possibilities for investigation of gene network evolution at the over-genetic and over-organism levels of biological organization, such as populational or ecological.

The genetic diversity of a community impacts the computational time

The most time-consuming procedure in the HEC computational process is the simulation of the reproduction of populations. When a broadly diverse (10^6–10^8 unique ACs) community is simulated, almost all computational time is consumed by this function (Fig. 1).

Figure 2 shows an example of a generalized population genome in HEC. It is just a multidimensional distribution of allelic frequencies for all genes present in cells of this population. This

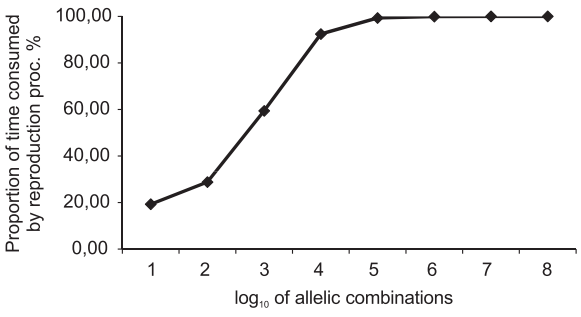

Fig. 1. Proportion of time consumed by the reproduction procedure in relation to the overall computational time.

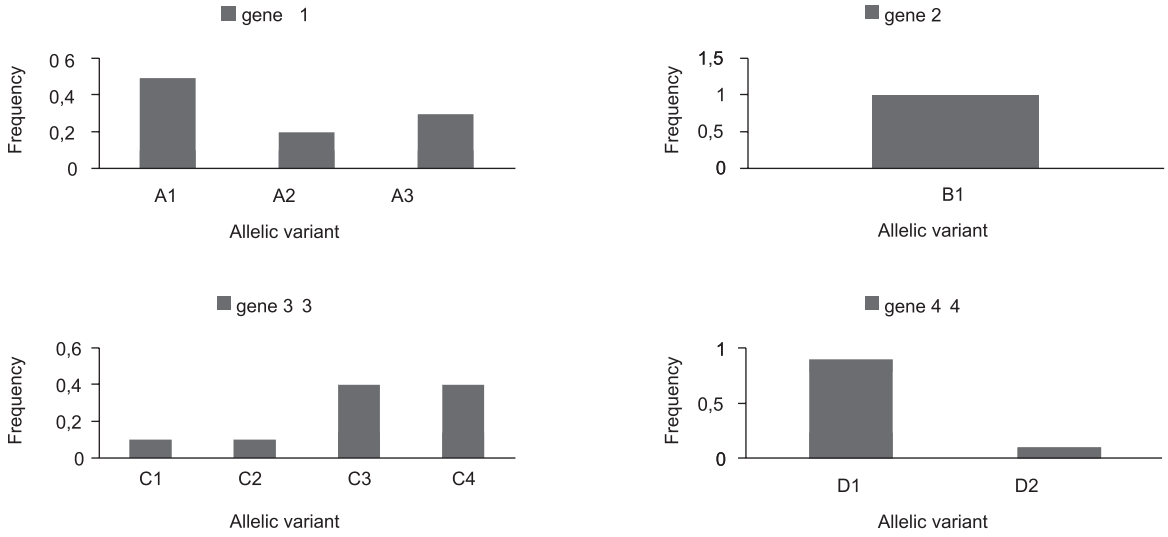

Fig. 2. Example of a generalized population genome in the HEC:

A1(0.5) A2(0.2) A3(0.3)	distribution of alleles for gene 1
B1(1)	distribution of alleles for gene 2
C1(0.1) C2(0.1) C3(0.4) C4(0.4)	distribution of alleles for gene 3
D1(0.9) D2(0.1)	distribution of alleles for gene 4

generalized genome contains four genes, which have three, one, four, and two possible allelic variants present in the population, respectively. By multiplying the numbers of allelic variants, we obtain the total AC number (for example, $3 \times 1 \times \times 4 \times 2 = 24$, as shown below).

In order to calculate the total population size change, it is necessary to calculate it for each AC subpopulation (cells with identical genome), and then rearrange new allelic frequencies in the population (Lashin *et al.*, 2010). This routine requires cycles and cycles of the same function calls where only the AC changes. Thus, it can be and should be parallelized.

High-performance versions of the HEC

The initial reproduction procedure used a recursive algorithm for iteration over allelic combinations, which was unsuitable for parallelization. The iteration algorithm and internal data representation were modified (Mustafin *et al.*, 2012), which resulted in an elegant parallelization scheme (fig. 3) upon which high-performance implementations could be made. Several high-performance versions of the reproduction procedure were developed and tested: OpenMP, MPI, and CUDA ones.

The OpenMP version has been developed for the desktop version of HEC primarily along with a graphic user interface. It is effective in modeling mid- and high-diverse communities (> 100 AC). Computations for models of low-diverse communities (< 100) themselves take less time than data exchange between processes. It is ineffective to parallelize such models. The optimal number of parallel threads for this version should be divisible by the number of populations. It is also desirable that simulated populations should have roughly equal levels of genetic diversity in order to obtain the optimal thread load. The OpenMP version is suited for high frequency/few-core processors (In contrast with MPI, OpenMP gives an acceleration even when the number of threads exceeds the number of processor cores.)

An MPI version to be used with console versions of HEC is being developed. It is effective when it comes to models of genetic diversity more than 100 ACs. Communities of any number of populations with any genetic diversity can be simulated with minimal efficiency loss (as the genetic diversity increases, the data exchange time defies evaluation). Models with high genetic diversity (> 10^5 ACs) show linear efficiency growth. The tendency breaks only when the software is run with

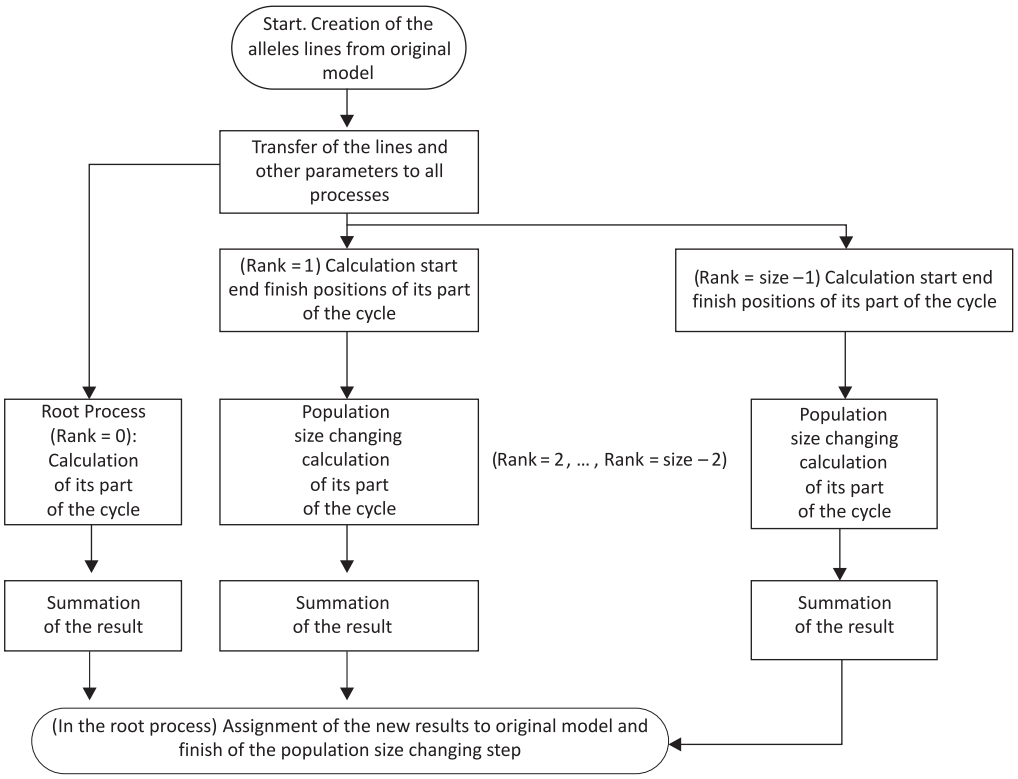

Fig. 3. Parallelization scheme for the reproduction procedure.

The MPI version implies that each thread receives its portion of allelic combinations, performs calculations, and returns data to the root node (MPI_BROADCAST and MPI_REDUCE are used). The CUDA version saves data to the memory of the video card, and then GPU performs calculations and data return to the main process.

very high numbers of parallel threads (typically for supercomputer clusters). The MPI version is suited for supercomputer clusters and personal computers with several core processors. It can also replace the OpenMP version, while in most cases it shows just as good results as OpenMP.

The CUDA version is developed to run on computers with NVidia CUDA graphic accelerators. It is effective only on models of high and extremely high genetic diversity. If AC is less than 10^5, the version is awfully ineffective due to memory allocation (on a video accelerator) and copying data takes more time than computation procedures. An obvious advantage of the CUDA version is the possibility to use extremely high numbers of threads (more than 1000-fold compared to OpenMP or MPI). With this version, models of genetic diversity exceeding 10^6 ACs can be calculated much faster than with any other. Simulation results and tests are presented in the following sections (see also tables 1–4).

Test calculations

Test simulations were performed on the AMD Phenom II ×6 1055T (6-core) processor and the NVIDIA GTX 570 video accelerator. The MPI version was also tested on the NKS-30T supercomputer cluster (http://bioinformatics.bionet.nsc.ru/). The tests were performed on the set of test models published previously (Lashin *et al.*, 2010; Lashin, Matushkin, 2012). Furthermore, special load tests were used.

Table 1 shows the test results of the MPI version. Simulation time depended on several parameters, the main of which were AC number (as the measure of community genetic diversity) and the number of iterations (i.e. generations) per simulation run. Parallelization efficiency also depends on these parameters. In low genetic diversity (1–100 ACs) models, parallelization is ineffective, as the execution time of the reproduction procedure is low with respect to the overall execution time. In the range from 100 to 1000 ACs, the paralleliza-

Table 1

Test results for the MPI version on AMD Phenom II ×6 1055T (2.8 GHz)

Number of allelic combinations	Iterations (generations)	Average simulation time, s; parallelization efficiency, % Number of parallel threads					
		1	2	3	4	5	6
10^3	25 000	53 s	45 s – 58 %	44 s – 40 %	44 s – 30 %	45 s – 23 %	52 s – 17 %
5×10^3	10 000	73 s	46 s – 79 %	38 s – 64 %	34 s – 54 %	33 s – 44 %	37 s – 33 %
10^4	5 000	68 s	39 s – 87 %	31 s – 73 %	27 s – 63 %	25 s – 54 %	27 s – 42 %
10^5	500	85 s	48 s – 89 %	35 s – 81 %	28 s – 76 %	24 s – 71 %	23 s – 62 %
10^6	50	162 s	88 s – 92 %	61 s – 89 %	48 s – 84 %	40 s – 81 %	37 s – 73 %
10^7	4	199 s	101 s – 99 %	69 s – 96 %	55 s – 90 %	46 s – 87 %	44 s – 75 %

The number of generations per simulation decreases with the growth of genetic diversity. It was made in order to make the test last for several minutes. The values are rounded up or down to the nearest integer.

Table 2

Test results for the OpenMP version on AMD Phenom II ×6 1055T (2.8 GHz)

Number of allelic combinations	Iterations (generations)	Average simulation time, s; parallelization efficiency, % Number of parallel threads			
		1	2	4	8
10^3	25 000	49 s	39 s – 63 %	37 s – 33 %	35 s – 18 %
5×10^3	10 000	65 s	41 s – 80 %	29 s – 56 %	28 s – 29 %
10^4	5 000	59 s	35 s – 84 %	23 s – 64 %	21 s – 35 %
10^5	500	75 s	49 s – 77 %	23 s – 82 %	21 s – 45 %
10^6	50	147 s	81 s – 91 %	45 s – 82 %	34 s – 54 %
10^7	4	186 s	99 s – 94 %	73 s – 64 %	48 s – 48 %

Table 3

Comparison of parallelization efficiency for the OpenMP and MPI versions

Number of allelic combinations	Iterations (generations)	Parallelization efficiency, %			
		2 threads		4 threads	
		MPI	OpenMP	MPI	OpenMP
10^3	25 000	58	63	30	33
5×10^3	10 000	79	80	54	56
10^4	5 000	87	84	63	64
10^5	500	89	77	76	82
10^6	50	92	91	84	82
10^7	4	99	94	90	64

Table 4

Comparison of the best computational times obtained using various high-performance versions of HEC

Number of allelic combinations	Iterations (generations)	Minimal computational time, s (optimal number of parallel threads)		
		MPI	OpenMP	CUDA
10^2	25,000	23 (1)	22 (1)	318 (1000)
10^3	25,000	43 (3)	35 (8)	335 (500)
5×10^3	10,000	33 (5)	28 (8)	128 (1000)
10^4	5000	25 (5)	21 (8)	65 (500)
10^5	500	23 (6)	21 (8)	12 (1000)
10^6	50	37 (6)	34 (8)	3 (10,000)
10^7	4	43 (6)	47 (8)	3 (5000, 10,000, 50,000)
10^8	1	×	×	5 (50,000, 100,000)

× Failure with AMD Phenom II ×6 1055T

tion effect becomes apparent and depends on the number of iterations per simulation. In models of high-average genetic diversity (1000–10,000 ACs), parallelization is more effective, and the iteration number exerts next to no effect on efficiency. In models with AC > 10,000, efficiency does not depend on the number of iterations. With AC increase, the total efficiency approaches unity, but the more parallel threads run, the less efficiency is kept, although the computation time decreases continuously.

Test results for the OpenMP version are presented in Table 2. The main difference from MPI is that OpenMP loses more efficiency when the number of threads grows. Comparison of these two versions is shown in Table 3.

Finally, the test results for the CUDA version were compared with OpenMP and MPI. We compared minimal obtained times for each version (Table 4). In the table, the optimal number of threads for each table cell is shown in parentheses. Thus, the CUDA version can significantly speed up simulations in computationally costly (AC>10^5) tasks.

Choosing the optimal parallel version

We classified parallel versions described above according to the most suitable simulation tasks for each model. They are listed below:

(1) Models of communities with genetic diversity less than 100 ACs are better simulated with the use of non–parallel HEC because the reproduction procedure in such a simple case takes only a small amount of the total time. Sometimes parallel versions even increase the simulation time. Only the OpenMP version works with the same efficiency (no data exchange), while the MPI version takes 15–20 % more time. We strongly recommend not using the CUDA version in simple tasks, as it may increase the running time.

(2) Models of communities with genetic diversity of 100–10,000 ACs are better simulated with the use of either MPI or OpenMP versions. The efficiency of both models grows in proportion to the increase of AC. The efficiency of the CUDA version grows even more, but its overall computational time is longer than in the above case.

(3) Models of communities with genetic diversity of 10^5–10^6 AC are suitable for all three versions.

(4) Finally, communities of extremely high genetic diversity (> 10^6 AC) are better simulated with the CUDA version. It takes much less time for simulation than with MPI or OpenMP.

CONCLUSION

In this paper, we present several high-performance versions of the HEC software packages, available at our web site. Under appropriate conditions (models of "optimal" genetic diversity), they provide nearly linear acceleration. Parallel versions are classified according to the most suitable situations for their usage. The non-parallel version is the best for simple models of low genetic diversity. All the three parallel versions are appropriate for models of intermediate genetic diversity. Finally,

the CUDA version is the best for extremely diverse communities. We think that the last case is the most interesting for large-scale theoretical studies, and we hope that high-performance versions of HEC presented in this paper will allow users to investigate more complex and diverse microbial communities and will produce new interesting biological results.

ACKNOWLEDGMENTS

The study was supported by the following grants: RFBR 12-07-00671-a, project VI.61.1.2, and interdisciplinary SB RAS project 47.

LITERATURE

Ashlock D., McEachern A. A simulation of bacterial communities // IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB). IEEE. 2011. P. 1–8.

Ayton G.S., Noid W.G., Voth G.A. Multiscale modeling of biomolecular systems: in serial and in parallel // Curr. Opin. Struct. Biol. 2007. V. 17. No. 2. P. 192–198.

Beardmore R.E., Gudelj I., Lipson D.A., Hurst L.D. Metabolic trade–offs and the maintenance of the fittest and the flattest // Nature. 2011. V. 472. No. 7343. P. 342–346.

Bihary D., Kerenyi A., Gelencser Z. *et al*. Simulation of communication and cooperation in multispecies bacterial communities with an agent based model // Scalable Comput. Pract. Exp. 2012. V. 13. No. 1. P. 21–28.

DeAngelis D.L., Mooij W.M. Individual-based modeling of ecological and evolutionary processes // Annu. Rev. Ecol. Evol. Syst. 2005. V. 36. No. 1. P. 147–168.

Esteban P.G., Rodríguez-Patón A. Simulating a Rock – Scissors – Paper Bacterial Game with a Discrete Cellular Automaton // New Challenges on Bioinspired Applications, Lecture Notes in Computer Science / Eds. Ferrández J.M., Álvarez Sánchez J.R., De la Paz F., Toledo F.J. Berlin Heidelberg: Springer, 2011. P. 363–370.

Kutalik Z., Razaz M., Baranyi J. Connection between stochastic and deterministic modelling of microbial growth // J. Theor. Biol. 2005. V. 232. No. 2. P. 285–299.

Lashin S.A., Matushkin Y.G. Haploid evolutionary constructor: new features and further challenges // In Silico Biol. 2012. V. 11. No. 3–4. P. 125–135.

Lashin S.A., Matushkin Y.G., Suslov V. V., Kolchanov N.A. Evolutionary trends in the prokaryotic community and prokaryotic community-phage systems // Russ. J. Genet. 2011. V. 47. No. 12. P. 1487–1495.

Lashin S.A., Suslov V.V., Matushkin Yu.G. Comparative modeling of coevolution in communities of unicellular organisms: adaptability and biodiversity // J. Bioinform. Comput. Biol. 2010. V. 8. No. 3. P. 627–643.

Martins M.L., Ferreira S.C., Vilela M.J. Multiscale models for biological systems // Curr. Opin. Colloid Interface Sci. 2010. V. 15. No. 1–2. P. 18–23.

Mustafin Z.S., Matushkin Y.G., Lashin S.A. Haploid Evolutionary Constructor: parallelization and high performance simulations of evolution of prokaryotic communities // Russ. J. Genet. Appl. Res. 2012. V. 16. No. 4/1. P. 825–829.