Adv Syst Sci Appl 2019; **01**; 141-149
Published online at http://ijassa.ipu.ru/index.php/ijassa/article/view/594

The application of graph decomposition to development of large-scale agent-based economic models

V.L. Makarov[1], A.R. Bakhtizin[1], E.D. Sushko[1], G.B. Sushko[1]
[1)] *Central Economics and Mathematics Institute CEMI RAS, Moscow, Russia*
E-mail: albert.bakhtizin@gmail.com

Received May 17, 2018; Revised March 13, 2019; Published April 15, 2019

Abstract: In this work we describe the application of the graph decomposition algorithms for the development of a scalable high-performance agent-based model of population of Russia described in terms of demography, migration and transport flows. The simulated system consists of agents representing individuals and sets of links to other agents, which represent the social interactions of individual. Individual agents in the model participate in several independent processes, for which different sets of social links is important such as family and neighbors. To perform a load balancing of agents between cluster computer nodes the METIS graph decomposition algorithm was used. These algorithms allow to split the graph of agents and links into parts of similar size with least possible number of links between them. A number of numerical experiments was carried out for test model to estimate the influence of the parameters of the model on scalability.

Keywords: agent-based modelling, numerical modelling, parallel computing, graph decomposition.

1. INTRODUCTION

The simulation of complex economic processes by means of agent-based approach requires a large number of agents to be used. One can expect the number of agents to be comparable to the population of the modelled society such as the city or a country i.e. up to 10^9 agents. High number of agents requires the use of supercomputer clusters to perform the simulation and requires a special programming technique to be applied in order to be able to run the model on such computer. A typical supercomputer nowadays is a cluster of multiprocessor multicore nodes with the shared memory inside the node and distributed memory between the nodes of the cluster. That distributed memory model requires the data describing agents to be evenly split between the nodes of the cluster. The paper describes the application of the METIS[1,2,3] spatial decomposition algorithm for conducting multi-agent simulations of the behavior of society using supercomputers.

There is a number of technologies and frameworks for development of ABM (Agent-Based Modeling) simulation software such as Microsoft Axum [4] and Repast HPC [5] and SWAGES [6]. These frameworks implement mechanisms of exchange of messages between agents but the problem of optimal distribution of agents between processes should be solved by the simulation code taking into account all types of interactions in the model. For example, a similar approach using METIS algorithm for a large-scale epidemiologic ABM using Repast HPC and METIS was implemented in paper [7]. Another example is the application of the METIS decomposition for modeling of railway scheduling in paper [8]. E ach of these codes used its own implementation of the decomposition code to transform the objects in the model into graph representation for METIS.

The simulated system consists of agents which represent individuals and a set of links to other agents, which represent the social interactions of individual. The individual agents in the model participate in several independent processes, for which different sets of social links is important such as family and neighbors.

The agents of the system participate in two processes: 1) the process of reproduction of the population, and 2) the process of migration. In the first process they use messages exchange to search for the partner to form a family. In the second process the message exchange mechanism is used to obtain information about available jobs in different regions to determine the direction of migration.

To conduct agent modeling of demographic processes with a large number of agents, it is necessary to create an initial state of the population of agents with a given distribution by region, age, sex, income and other relevant parameters. In the test case, reading of the data by region from the text file and the geometry of regions from the image file was realized. The data is used to construct a rectangular grid, each cell of which is connected to a certain region and parameters of agents in the cell are determined by the parameters of the region.

A random distribution of the number of resident agents between grid cells (uniform distribution of population within each region with the fixed total number of agents in the system) and calculation of grid cells decomposition by processors was implemented.

2. THE MODEL DESCRIPTION

The most common way of writing simulation programs for cluster computers is to use C++/Fortran language and MPI library which are available on all supercomputers. To simplify the development of the model the distributed ABM framework [9] was developed for use with the high-level Microsoft .Net platform. The use of the high-level programming language for model description requires the implementation of wrapper code to system-level functions for inter-process exchange using native MPI library. As most of modern supercomputers run on Linux operating system, we have decided to use Microsoft .Net Core and MONO [10] implementations of .NET platform available for this OS.

The choice of these technologies was determined by the following criteria:

1. The system has to be scalable across multiple computational cluster nodes (i.e. use resource of multiple nodes for speedup) therefore the multithreading calculation model was not suitable as it is limited to single cluster node.

2. The model should be easy to develop and maintain and therefore the high-level programming language C# was used.

3. The system should be efficient and therefore the native MPI library was used instead of TCP/IP Sockets or .Net libraries like Windows Communication Foundation as these technologies are not optimal for supercomputers and HPC applications. MPI libraries installed on each cluster computer are usually tuned for particular proprietary network system which is used on the cluster such as Infiniband.

4. The program should be portable and should be compatible with any operating system i.e. running on both developer workstations and on cluster computers.

5. The results of the simulation should not depend significantly on the number of processors used for the calculation. All interactions between agents on different processes on each simulation step should be taken into account. The simulation solution calculated on different number of processors can still be different due to the difference in the order of operations (due to the limited float point numbers precision) and the difference in the random number sequence (each process has its own independent random number sequence).

An efficient mechanism of message exchange between agents was implemented by means of message queue and native MPI collective operations. The message queue accumulates a buffer of messages to different processes and then uses MPI AllToAll exchange operation to deliver contents of messages. This operation delivers the buffers of arbitrary size from each MPI process to all other processes in most efficient way by splitting the buffer into chunks of optimal size for network transfer and hiding the latency of network operations by performing simultaneous several send and receive operations.

To use native operations with managed C# objects operations of binary serialization and deserialization of objects were implemented and C# wrappers for native functions were written.

2.1 The system decomposition algorithm

The modeled system consists of a large number of agents connected with each other by the number of social links. To perform an efficient simulation of the system it is necessary to split the set of agents between MPI processes taking into account their links with other agents. This distribution of agents between processes determines the parallel scalability of the program. The quality of the decomposition is determined by the balance of the agents count on different processes and the number of links between agents assigned to different processes.

The decomposition of a graph of agents with links can be performed by the application of graph decomposition algorithms such as METIS [1], Scotch [11] or Jostle [12]. These algorithms provide a computationally efficient way of decomposition of large graphs into parts of equal size with minimal borders between them. Such approach leads to a best possible system decomposition with a fine distribution of agents but it also leads to a number of drawbacks: 1) it requires the formation and a decomposition of a sparse matrix of size N, where N is the total number of agents, 2) each change of a number of agent due to agent's birth or death leads to a recalculation of the decomposition. In order to calculate the decomposition in a more efficient way the system was split by a rectangular grid covering all regions. The decomposition of cells is much faster due to smaller matrix and doesn't require recalculation of the decomposition after each time step as the distribution of population changes slowly with time. Due to the small size of the input matrix the decomposition can be performed using sequential version of METIS algorithm without sensible impact on the parallel performance of the simulation.

To calculate the decomposition of the grid, a graph METIS algorithm with a weighting was used (METIS_PartGraphRecursive). The METIS algorithm takes an input on a graph specified through the constraint matrix and an array of weights of the nodes of the graph and returns the optimal distribution of the graph by a given number of parts with minimizing the links between the parts.

The link matrix is given in CSR [13] format i.e. by the number of rows (N) and two arrays IA and JA. The array IA contains a sequence of partial sums of the number of non-zero elements in the matrix. This array is used to separate the elements of the JA array by strings. The JA array contains index lists of non-zero elements in all rows.

In the test example, the size of the matrix corresponded to the number of non-empty grid cells, the adjacent grid cells (not diagonally) were considered to be related. As the weight of the cell, the number of agents that should be created in the corresponding cell was used.

The calculation of the most optimal decomposition of such matrix is very computationally expensive. The METIS library implements a multilevel recursive coarse-grained algorithm for calculation of the reasonably good decomposition. On each level the original graph is transformed to a smaller graph using the coarse-graining algorithm, the decomposition of the smaller graph is performed and then a special refinement of the decomposition is performed.

As a result of calling the METIS algorithm for a given coupling matrix and weights and a given number of processors, an array of numbers is created that describes the optimal binding of the grid cells to the processors. These data are used to distribute cells and agents by processors during the calculation, but do not affect the aggregation procedure of the received calculation results. The results of the calculation are aggregated by region and should not depend on the decomposition used.

To test the decomposition algorithm a sample system was constructed using the data on the population of Russia. Russia can be a good example of a country with very different regions in terms of population. The total population of Russia is 144 million people and the average density of population is 8.58 person per square kilometer ranging from 4626 people/km^2 in Moscow to 0.07 people/km^2 in Chukotka Region. Large cities such as Moscow and St. Petersburg were considered as separate regions in order to distinguish the parameters of the region and parameters of the enclosing region because of the principal difference in population density, income and other parameters.

The figures below show the results of automatic grid decomposition using the METIS algorithm for 8 processors in two variants: without taking into account the number of agents in cells (the weight of each cell = 1) and taking into account the number of agents (cell weight equals to number of agents).

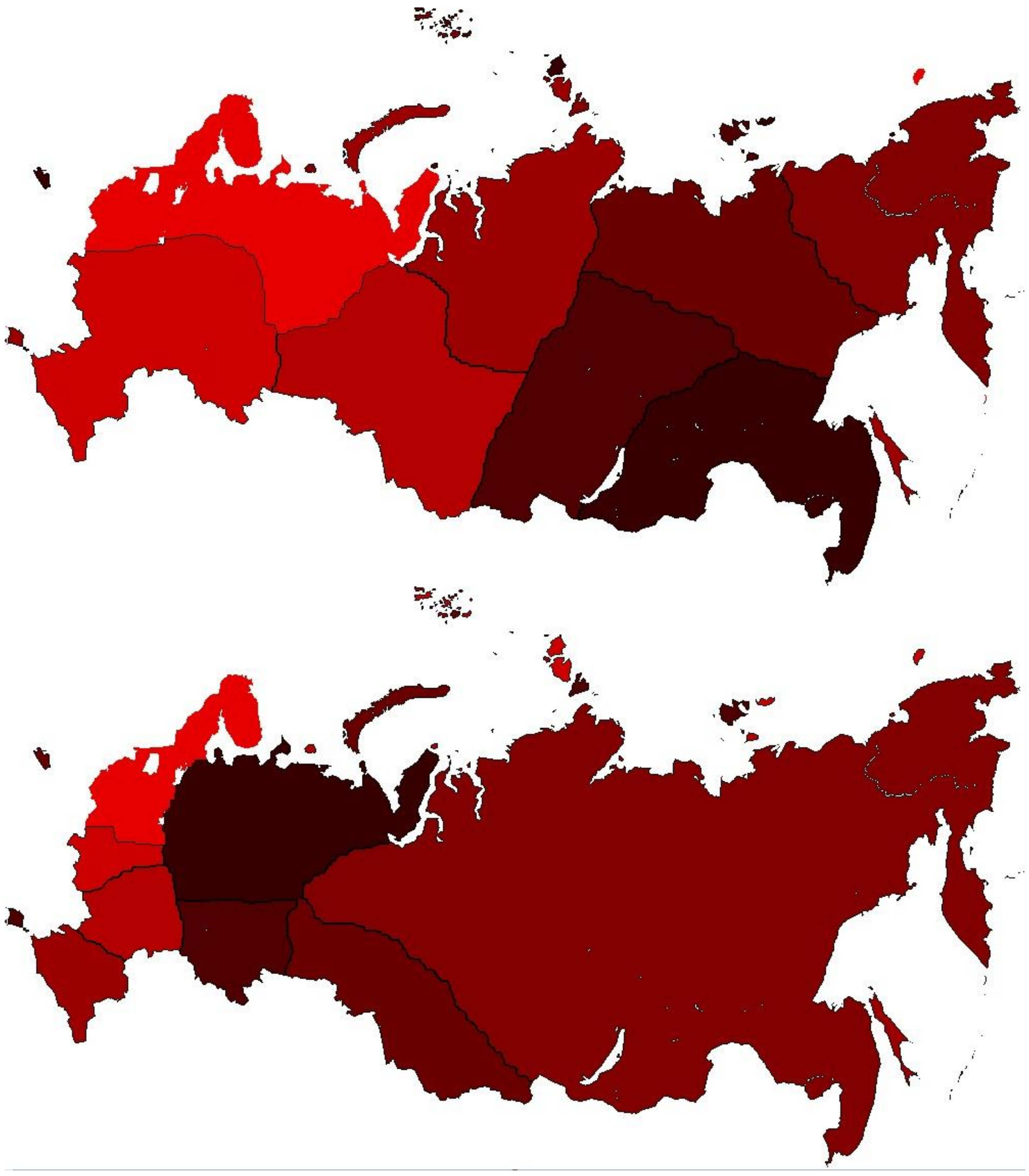

Fig. 1. The decomposition of a territory of Russia by 8 processors taking into account only the number of cells (top) and the number of agents in cells (bottom). One can notice much smaller regions in the western part of Russia on a second plot due to much higher population density.

Different shades of red show areas assigned to different processors. In the first case, the algorithm equalizes the number of grid cells on different processors with the minimum length of the boundary between them. In the second case, the number of agents attributed to the cells of a specific processor is equalized, while minimizing the length of the border.

One can see that in the second case the regions have different area because of the difference in population density. The difference in density between European part of the country (27 people per km^2) and Asian part (3 people per km^2) leads to much smaller regions in European part in terms of area and cell count. Another effect associated with the formulation of criteria for optimizing distribution is that the cells located on islands and exclaves are isolated and not directly connected to the nearest regions and in terms of decomposition can be linked to any processor.

The use of the sparse matrix of links between cells allows the further development of decomposition criteria which should in principle include the connectivity of cells which have no common border. These connections may arise due to air communication, railways and other ways of traveling which have to be taken into account for correct simulation of migration.

In the test example presented above, the total number of agents was set at 8400000. At the same time, the share of agents in each region was determined by the share of this region in the population of the country. Within the region, agents were distributed uniformly across cells. Such a calculation scheme allows us to change the total number of agents for test purposes, keeping their correct geographical distribution.

Using the algorithm described above, the distribution of the source system by the processors was obtained. In the implementation, it is important that such a distribution must be done before the agents are created, since the agents created must by initially correctly distributed among the nodes of the cluster to meet the limitations of the system memory.

To analyze the obtained distribution of agents by processors, one can use Amdahl's law, which connects the maximum achievable acceleration in parallel computations (S_p) with the number of processors (p) and the fraction of consecutive computations (a).

$$S_p = \frac{1}{\alpha + \dfrac{1-\alpha}{p}}$$

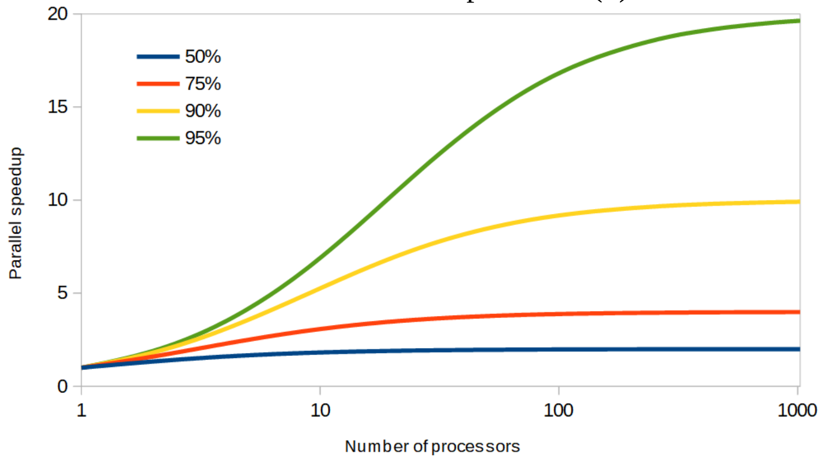

Fig. 2. The illustration of Amdahl’s law for description of parallel speedup on the number of processors and the value of parallel portion.

In our case, sequential computations arise due to an imbalance in the number of agents on the nodes, i.e. if we have two nodes and the first 400 agents, and on the second 300, then we can say that the first 300 agents are processed by each node in parallel, and the second 100 agents are processed sequentially by the second one, because the second node can’t perform calculations at this time.

The figure below shows the dependence of the parallel speedup on the number of MPI processes. The comparison of Amdahl’s law estimates for decomposition with and without cell weights show that decomposition with cell weights leads to a very balanced distribution of agents and potentially to a very good parallel speedup. The real parallel speedup of the test simulations is significantly lower than Amdahl’s law estimates but it still shows rather reasonable speedup.

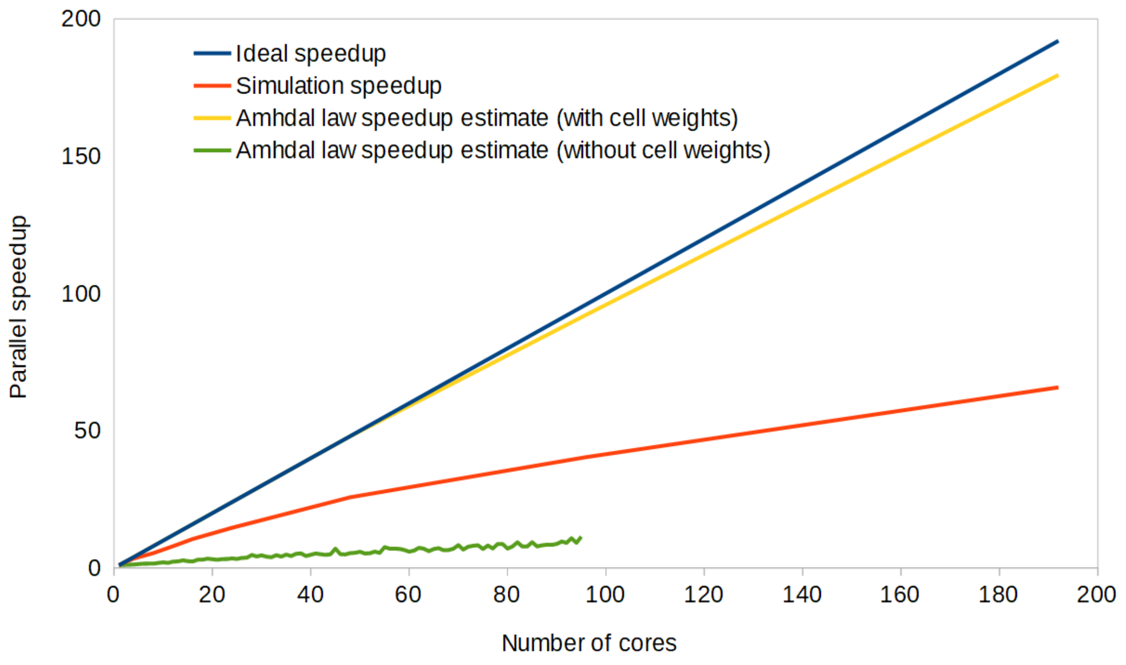

Fig. 3. The dependence of the parallel speedup on the number of MPI processes. The scalability of the test simulation is compared to the ideal speedup curve, and to the estimate speedup based on the Amdahl's law for the cases of decomposition with and without cell weights.

For the test calculation, a system consisting of 8400000 agents was selected. For each agent the number of neighbor agents was selected randomly (1..10), the neighbor agents were selected randomly within the same cell on the map and it's direct neighbors. The table shows the time of calculation of the step and the average number of agents on one processor.

Table. 1. The results of the simulation of 8.4 million of agents for different number of processors.

Number of cores	Time step simulation time, s	Parallel speedup	Number of agents per core	Fraction of remote message sends, %
1	55.683	1	8417590	0
2	28.872	1.92	4208795	0.15
4	16.114	3.45	2104397	0.39
8	10.262	5.42	1052198	0.68
12	7.006	7.94	701465	0.91
16	5.268	10.56	526099	1.10
24	3.796	14.66	350732	1.72
48	2.161	25.75	175366	2.26
96	1.378	40.38	87683	3.41
192	0.846	65.78	43841	5.56

The results shown in the table above show that the distribution of agents by processors as a result of the decomposition was fairly uniform, but the overall parallel speedup in this version is limited to a several dozen times. The table also shows the dependence of the fraction of remote message sends which were delivered to agent on another processor. The increase of the number of processors leads to increase of the number of borders between areas attributed to different processors, the number of remote sends and the ratio between network exchanges and local calculations within one processor which is another factor which affects the scalability of the simulation.

Uniform distribution of the number of agents by processors allows you to evenly distribute the load of RAM of each node of the cluster. As a test case, a multi-agent system was calculated with 148 million agents on 192 cluster processors. The main difficulty in this case was that with the total amount of RAM of the allocated nodes in 192GB on each node of the cluster there was only 8GB of RAM. Thus, it was necessary to distribute the agents evenly, so that on none of the cluster nodes the simulation processes exceed the memory consumption limit.

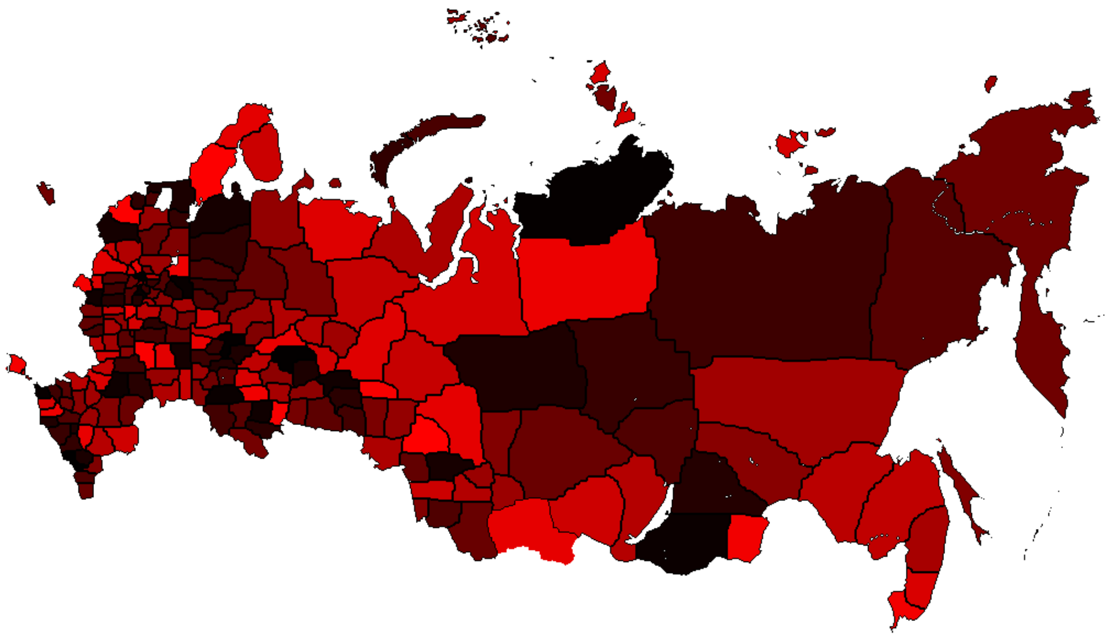

Fig. 4. Decomposition of the computational grid into 192 processors taking into account population density.

The figure above shows the distribution of the calculated grid for 192 processors, built taking into account the population density. In the distribution, the average number of agents per processor was 772000. The number of agents per processor ranged from 743711 to 787266, i.e. the spread of the number of agents on one processor was ~ 6% of the average. The calculation of one step in this simulation took about 15 seconds. In the future, the spread of the number of agents can be reduced by using a smaller calculation grid, which will improve the accuracy of the decomposition.

3.CONCLUSION

In this work a new framework for parallel calculations of agent-based models was presented and tested. The framework uses METIS graph decomposition algorithm to perform natural spatial distribution of agents in ABM-simulations to achieve high level of balance and parallel scalability. The decomposition algorithm based on cell distribution allows us to compute a balanced distribution of agents efficiently. Which can be used for both initial distribution of agents and redistribution of agents during simulation. The provided results of the test simulations show good scalability of the program across multiple computational nodes and possibility to perform ABM simulations with number of agents comparable to population of large countries and regions.

ACKNOWLEDGEMENTS

This work was supported by the Russian Science Foundation (grant # 14-18-01968). The possibility to perform computer simulations at the MVS-100K Joint Supercomputer Center and Tianhe 2 supercomputer is gratefully acknowledged.

REFERENCES

[1] Karypis, G. & Kumar, V. (1995). METIS-unstructured graph partitioning and sparse matrix ordering system, version 2.0.

[2] Karypis, G. & Kumar, V. (1999) Parallel Multilevel k-way Partitioning Scheme for Irregular Graphs. // SIAM Review, Vol. 41, No. 2, pp. 278 - 300

[3] LaSalle, D. & Karypis, G. (2013) Multi-Threaded Graph Partitioning // 27th IEEE International Parallel & Distributed Processing Symposium

[4] https://en.wikipedia.org/wiki/Axum_(programming_language)

[5] https://www.bsc.es/computer-applications/pandora-hpc-agent-based-modelling-framework

[6] Scheutz, M., Connaughton, R., Dingler, A., & Schermerhorn, P. (2006). SWAGES - An Extendable Distributed Experimentation System for Large-Scale Agent-Based Alife Simulations. In *Proc. of Artificial Life X*, pp. 412-419.

[7] Collier, N., Ozik, J., & Macal, C. M. (2015). Large-scale agent-based modeling with repast hpc: A case study in parallelizing an agent-based model. European Conference on Parallel Processing, pp. 454-465

[8] Salidol, M. A., Abril, M., Barber, F., Ingolotti, L., Tormos, P. et. al. (2006). Domain dependent distributed models for railway scheduling. In International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 163-176

[9] Bakhtizin A.R. et al. (2017) The Development of the agent-based demography and migration model of Eurasia and its supercomputer implementation. // Advances in Systems Science and Applications, [S.l.], v. 17, n. 4, p. 34-45

[10] http://www.mono-project.com

[11] Chevalier, C. & Pellegrini, F. (2008). PT-Scotch: A Tool for Efficient Parallel Graph Ordering. // Parallel Computing. 34 (6): 318–331. doi:10.1016/j.parco.2007.12.001

[12] Walshaw, C. & Cross, M. (2000). Mesh Partitioning: A Multilevel Balancing and Refinement Algorithm. // Journal on Scientific Computing. 22 (1): 63–80. doi:10.1137/s1064827598337373.

[13] Tinney W. & Walker J. (1967) Direct solutions of sparse network equations by optimally ordered triangular factorization. // Proceedings of the IEEE, 55(11):1801–1809