A Grid Based Simulation Environment for Parallel Exploring Agent-Based Models with Vast Parameter Space

Chao Yang[1,2(✉)], Isao Ono[2], Setsuya Kurahashi[4], Bin Jiang[2,3], and Takao Terano[2]

[1] Business School, Hunan University, Changsha, China
{yangchao,jiangbin}@trn.dis.titech.ac.jp
[2] Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology, Meguro, Japan
{isao,terano}@dis.titech.ac.jp
[3] College of Computer Science and Electronic Engineering, Hunan University, Changsha, China
[4] Graduate School of Business Sciences, University of Tsukuba, Tsukuba, Japan
kurahashi@gssm.otsuka.tsukuba.ac.jp

Abstract. Agent-based simulation models with large experiments for a precise and robust result over a vast parameter space are becoming a common practice, where enormous runs intrinsically require highly intensive computational resources. This paper proposes a grid based simulation environment, named Social Macro Scope (SOMAS) to support parallel exploration on agent-based models with vast parameter space. We focus on three types of simulation methods for agent-based models with various objectives: (1) forward simulation to conduct experiments in a straightforward way by simply operating sets of parameter values to obtain sets of results; (2) inverse simulation to search for solutions that reduce the error between simulated results and actual data by means of solving "inverse problem", which executes the simulation steps in a reverse order and employs optimization algorithms to fit the simulation results to the desired objectives; and (3) model selection to find optimal model structure with subset of parameters and procedures, which conducts two-layer optimization to obtain a simple and more accurate simulation result. We have confirmed the practical scalability and efficiency of SOMAS by a case study in history simulation domain.

Keywords: Agent-based simulation · Grid computing · Forward simulation · Inverse simulation · Model selection

1 Introduction

Social simulation is a research method by which researchers construct artificial societies in their computers to explore problems in social science [1]. Traditional statistic ways set up social simulation models based on the real dataset in a top-down manner, which show the relationship of parameters within a model but fail to clarify

Q. Zu et al. (Eds.): HCC 2014, LNCS 8944, pp. 534–548, 2015.
DOI: 10.1007/978-3-319-15554-8_44

the causations. In the recent literature, agent-based simulation (ABS) has become a powerful computational modeling paradigm. ABS approximately represents members of organizations, corporations or societies as autonomous agents, and builds a model of social system as interactions among agents along their learning, adaption, and evolutionary process. ABS implements the model in a bottom-up type, through which, we are able to explain the causations from micro-level conditions to a macro-level emergence, and then understand the corresponding social phenomena. ABS has been widely applied in the research of social simulation field [2].

However, on the other hand, ABSs, especially those employ evolutionary algorithms and reinforcement learning methods for solutions, explore an enormous parameter space, and it is generally difficult to choose an appropriate parameter set and determine their influence degrees in the simulation model. In addition, it is also difficult to decide whether a combination of parameter values works well to achieve the desired objectives. Furthermore, large experiments required by ABSs over a vast parameter space always consume long execution time and intrinsically require highly intensive computational resources far exceeding the capability of one computer processor [3, 4]. In order to solve such problem, distributed computing such as grid technologies and cloud computing are employed to obtain the simulation result within a short CPU time.

Therefore, we propose a grid based simulation environment for supporting agent-based models with vast parameter space, based on a general classification of simulation approaches of agent-based models. The main objectives of this paper are to (1) provide a grid based simulation environment with three functional components to support parallel execution of ABS models through forward-, inverse-simulation and model selection methods; (2) implement a common programming interface for easy and smooth execution of ABS experiments through forward-, inverse-simulation and model selection methods, without necessary to modify the source code of ABS models. Such a grid based simulation environment is especially beneficial for those social scientists they are not familiar with recent computer usages, because, using the SOMAS library, they do not need to modify their own simulators but only prepare sets of property files to run. This paper then reports one case study in history simulation domain, through which demonstrates the effectiveness of SOMAS.

The rest of this paper is organized as follows: Section 2 makes a brief review of the related work. Section 3 proposes and implements SOMAS. Section 4 conducts one case study on SOMAS, and evaluates the effectiveness of SOMAS by analysis of the simulation results. Finally, Section 5 concludes the paper and proposes some ideas for future work.

2 Related Work

There have been many contributions on the implementation of large experiments of ABS models on a distributed computing environment. In this paper, we distinguish these tasks into two categories: one is large-scale distributed ABS model, which simulates a large system with thousands to millions of agents; the other is small ABS

model with vast parameter space, which executes enormous trials of the ABS model with various parameter sets. The former are more challenging because of managing one trial of ABS model with interactions among a huge number of agents across the boundary of computational nodes, while node boundary does not need to be considered in the latter because there is no communication across each run of ABS.

Among the contributions in the first category, Yamamoto et al. propose a Java-based framework named ZASE to support large-scale massive multi-agent based simulation (MMABS) with thousands to millions of agents [5]. ZASE makes use of computer clusters over the Internet to conduct distributed computation in parallel, and is applied to analyze the large-scale city transportation system in Japan. Another work is HLA_Grid_Repast, which is a middleware platform to allow users to execute large-scale distributed Repast simulations [6]. HLA_Grid_Repast presents Repast models as one level of service and facilitates the execution of large-scale agent federations, following a mechanism of parallel simulation and modeling over a grid environment at the same time. Besides, some popular toolkits such as Netlogo not only offer users abundant functions for agent-based modeling, but also support distributed experiments of agent-based models on clusters.

Of the works in the second category, Pignotti et al. create a virtual machine environment named SimulationBox to support ABS models using Swarm, Repast and Mason on a grid environment. SimulationBox provides semantic workflow mechanism to select models that meet specified macro criteria [7]. Spot-Oriented Agent Role Simulator (SOARS) by Deguchi et al. support both large-scale ABS models with vast number of agents and small ABS models with vast parameter space on a grid environment. There also exists system like MEME, QosCosGrid which support some kind of grid computing. However, the usage of these tools is not wide spread so far.

Besides, among the studies of running simulation models distributed on a grid environment, grid-oriented genetic algorithm framework (GOGA) has been developed for solving large-scale genetic algorithm (GA) optimization problems [8]. Especially, Ono et al. has proposed GOGA 2 to support parallel computational tasks of GA on a grid environment, and has confirmed its practicability on a grid test bed constructed by a cluster of multiple public clusters [9]. Although the main purpose of GOGA 2 is numerical optimization, not suitable for social simulations, we can utilize its libraries to distribute and implement large experiments of agent-based models with multiple simulation trials on a grid environment, through forward-, inverse-simulation and model selection methods.

3 Proposal and Implementation of SOMAS

In this section, we propose a grid based simulation environment, named Social Macro Scope (SOMAS), in order to meet with various simulation requirements for agent-based models over a vast parameter space. In the following, we will discuss the design and implementation of SOMAS based on a general classification of ABS

methods. Then, we describe the procedure of using SOMAS to implement distributed experiment execution of ABSs on a grid environment.

3.1 Simulation Methods of Agent-Based Model

There are many types of simulation methods to implement agent-based models, for instances, conventional ways execute the simulation steps in a straightforward way: all micro-level parameters and initial conditions are firstly set, then the simulation steps are executed, and finally the macro-level results are observed as outputs. Such a process is called "forward simulation" in this paper. By changing the parameter values, forward simulation collects sets of results and uses them for sensitivity analysis or landscape prediction. When using forward simulation method, plural trials are always executed to examine stochastic variations of outcomes by operating sets of parameter values, such as the Axelrod Cultural Model (ACM) [10], and other examples can be found in [11].

In contrast, inverse techniques execute the simulation steps in the reverse order by means of solving "inverse problem": a macro-level objective function is firstly set, the simulated worlds are then evolved to fit to the objectives, and finally the micro-level agent parameters are optimized [12, 13]. Generally, in "inverse simulation", optimization techniques such as Genetic Algorithms (GAs) are employed to search for solutions that reduce the error between the simulated results and objective values, examples can be found in Yang's research work [4]. She has developed ABS models to analyze a particular family line which continued to produce successful candidates in civil service examination in imperial China over five hundred years, and finally inferred the successful family strategies by systematic parameter optimization via inverse simulations. Other contributions can be found in [3].

However, there always comes the request to coordinate the complexity of a model structure when applies inverse simulation to analyze complex social phenomena. To solve such a problem, we propose model selection method, which selects subset of variables and procedures as features, and uses them to find more accurate agent-based models to describe the desired problem. We implement model selection by using feature selection in machine learning [14]. When we apply model selection to ABS field, each candidate parameter and procedure of the ABS model is managed as a feature, and each model structure is determined based on a subset of selected variables and procedures as features. The procedure can be described by if-else rules and represented by "1" or "0" values in feature selection process. The model is evaluated by how precise the selected feature set is able to reproduce the desired results as macro outcomes. The complexity of the model is decided based on the number of features in the evaluation results. Within a certain error limitation, model selection can also provide plural feature sets for multiple solutions to a same macro problem.

Fig.1 summarizes and compares the general different execution processes among forward-, inverse-simulation and model selection methods. As shown in Fig.1, a lot of simulation trials are necessary when applying the three types of methods to conduct

ABSs. Most ABS applications reduce the number and range of parameters and procedures in order to finish the experiments within a limited evaluation time, while a meta-level problem of how to decide a reasonable execution time for evaluation occurs. Parallel exploration on vast parameter space is therefore considered as a solution to speed up the execution time of ABS experiments.

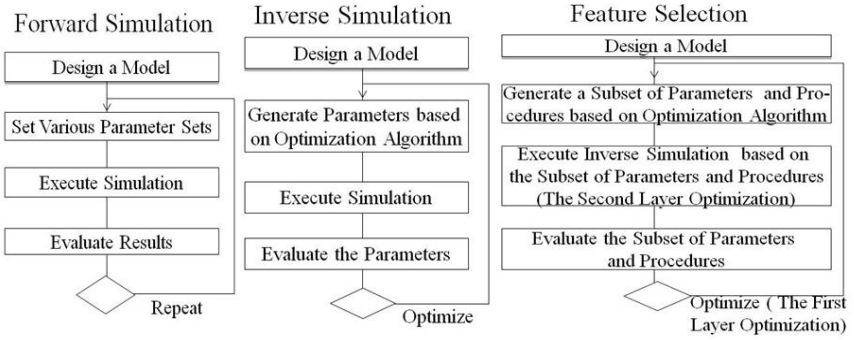

Fig. 1. Three types of simulation methods of agent-based models

3.2 Proposal of SOMAS

Against such a background, a grid based simulation environment is required to support parallel exploration on ABS models with vast parameter space, which releases social scientists from a lot of time to implement optimization algorithms and learn domain knowledge on grid programming. In this paper, we consider a general scenario of agent-based models through various simulation methods as follows: 1) sets property files to use a grid-based simulation environment; 2) determines one type of simulation method to implement the agent-based model; 3) sets system parameters which required by the selected simulation method; 4) generates a number of random seed to determine the random sequence for executing optimization algorithms in inverse simulation or model selection methods; 5) executes the simulation model by performing appropriate logging at the same time; 6) repeats the actions from step (2) to (4) until the simulation reaches the predefined number of simulation trials;7) handles the log files properly to create a summary of the experiments. According to the above mentioned scenario, the parallelism of ABS models with different simulation methods is feasible through the following two points: 1) parallelism of trials, and 2) parallelism of optimization algorithm. Because multiple trials of ABS models and the evaluation process of optimization algorithms are independent, we can implement parallelism by assigning them to multiple computation nodes.

On the other hand, the difficulties of implementation of SOMAS mainly include: 1) a common programming interface for easy and smooth execution of agent-based models through various simulation methods; 2) programming libraries to meet with requirements from parallelism experiment execution, such as security authentication,

file transfer and acceptance, communication between the nodes of in and out of the computation cluster, task scheduling, flexibility to add/delete computation nodes to/from a grid environment, and convenient interfaces for users to operate a grid environment; 3) programming libraries for operations required by optimization algorithms.

3.3 Design of SOMAS

SOMAS is designed as a Java-based simulation framework, which utilizes GOGA 2's four libraries:1) Java-based Simple Grid Framework (JSGF) to provide grid setup operations, 2) Java-based Master- Worker library (JMW) to provide parallel computing operations, 3) Java-based GA Framework (JGAF) to provide GA-related operations, and 4) Java-based GA-Gridfying library (JGAG) to implement parallel GA on a grid environment. Further, SOMAS design and implement Java-based Simulation Method library (JSM) to enable three types of simulation methods on a grid. In addition, SOMAS provide a common programming interface to connect simulation methods and user program. Although user program is considered as ABS model by default, other types of social simulation models with multiple independent trials can also be explored. SOMAS also define a set of interface for logger output and experimental result summary. Table 1 compares the main functional difference between SOMAS and GOGA2.

Table 1. Comparison of main functional difference between GOGA 2 and SOMAS

Framework	JSGF	JMW	JGAF	JGAG	JSM
GOGA 2	○	○	○	○	×
SOMAS	○	○	○	○	○

3.4 Grid Environment Description

Grid technologies connect geographically dispersed computation resources such as PC clusters via Internet, in order to obtain the simulation results within a short CPU time by benefiting from a parallelism [15]. In recent years, the computer centers of universities and IT companies have started fare-paying services to rent the nodes of their PC clusters. A grid constructed based on a set of public PC clusters by rental service from universities or IT companies is called a Grid of Multiple Public Clusters (GMPUC). By using the rental services, users can dynamically construct a grid environment whenever they need computational resources, without necessary to handle complex operations for administrating PC cluster systems. Fig.2 describes a grid environment constructed based on GMPUC, which consists of user site and multiple geographically dispersed remote clusters.

User site is the site where user terminal belongs to. If user terminal has a private IP, and cannot be directly accessed from the outside, it is assumed that, as shown in Fig.2, the relay node with a global IP enables indirectly access from the user terminal

to the outside node. Otherwise, if the user terminal has a global IP and can be directly accessed from the outside, a relay node is not necessary. For the user terminal, it is assumed that the Java execution environment is installed. And, for the relay node, it is assumed that the SSH server has been installed with SSH port forwarding service, and other nodes can login by means of using the SSH public key encryption.

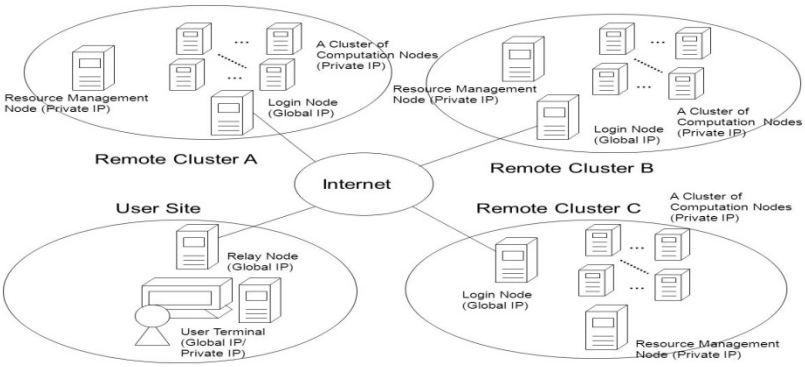

Fig. 2. Grid Computation Environment

Remote cluster is assumed to be composed of a login node, a resource management node and multiple computation nodes, where NFS, NIS is running. Login node has a global IP, which can be directly accessible from the outside. Login node can be login through SSH public key encryption authentication, and the process on the computation nodes can be invoked through local scheduling system of resource management node. Resource management node has a private IP, and is assumed to be able to communicate with login node, and multiple computation nodes. Resource management node runs local scheduling system. In this paper, Sun Grid Engine is assumed for the local scheduling system, which has been used in many GMPUC applications. Computation nodes have either private IPs or global IPs. Computation nodes in the same PC cluster can communicate with each other. The Java runtime environment is assumed has been installed on computation nodes. Besides, it is assumed that any computation node can finish a callback connection with either user terminal or relay node.

3.5 Implementation of SOMAS

SOMAS is developed and implemented based on the GMPUC environment, which utilizes the master-worker library of GOGA 2 to distribute plural trials of ABS models on a grid, through forward-, inverse-simulation and model selection methods, respectively, shown in Fig.3, Fig.4 and Fig.5. These pictures consist two parts: the user site and the remote PC clusters. In general, *user terminal* at user site manages the processes such as trials generation, and communicate with *trials worker nodes* of

the remote PC cluster. The PC cluster consists of plural remote nodes act as *trials worker* or *GA master* or *GA evaluation worker*, which performs differently when employing forward-, inverse-simulation and model selection method. Although there is only one node acts as the *user terminal*, we can control the number of remote nodes of the PC clusters by setting the values of arguments of the grid property file (explain later). We introduce the implementation of three types of simulation methods below.

Forward Simulation Function. Fig.3 describes the constitution of computation nodes and the process of implementation of forward simulation function. In forward simulation, the computation nodes include a *trials master* (*user terminal*) and plural *trials worker nodes* (remote node).

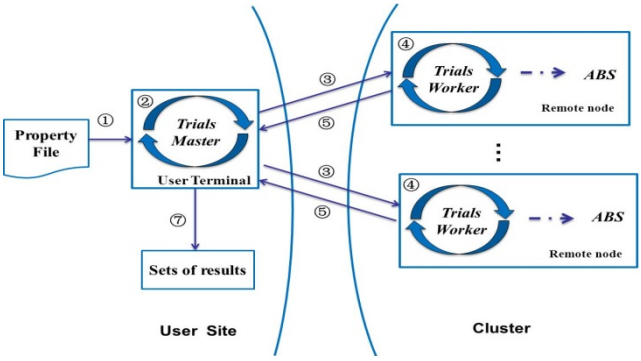

Fig. 3. The implementation of forward simulation function

As shown in Fig.3, forward simulation is carried out as follows: 1) Trials master reads experimental settings based on the property files, 2) According to the experimental settings, Trials master generates various parameter sets such as (A, B, C, …), and then 3) Trials master sends parameter sets to worker nodes via resource management node (which finds worker nodes available for computation tasks from the PC clusters and then determines the assignments of parameter sets to worker nodes), next 4) Worker nodes execute ABSs based on the received parameter sets, 5) when ABSs finish, worker nodes return the simulation results to trials master, the workflow from step (3) to (5) is repeated until all the parameter sets are handled, finally, 6) Trials master summarizes the simulation results and we obtain sets of result by forward simulation.

Inverse Simulation Function. Fig.4 gives the constitution of computation nodes and the process of implementation of inverse simulation function. We employ GAs for optimizations. In inverse simulation, the remote PC cluster consists of GA master and plural GA evaluation worker nodes for parallel computations. GA master manages GA operations other than individual evaluation tasks. GA evaluation worker nodes execute the ABS models based on the received parameter values and return the simulation results to GA master. The ABS models are executed similar to the forward simulation.

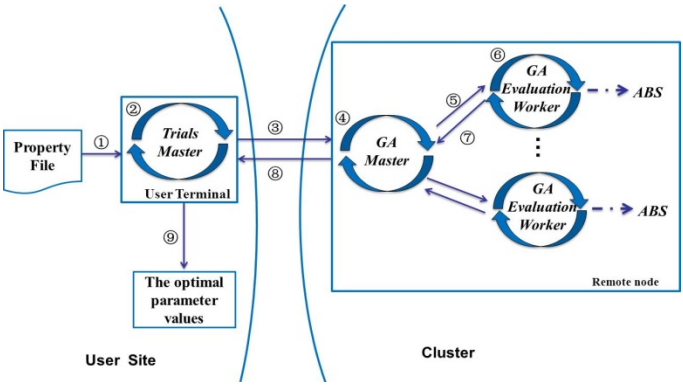

Fig. 4. The implementation of inverse simulation function

As shown in Fig.4, inverse simulation is carried out as follows: 1) Trials master reads the experimental settings based on the property files, 2) According to the experimental settings, trials master generates a trial of inverse simulation with specified optimization algorithm such as GA, 3) when GA is employed, trials master invokes a GA master on the remote PC clusters and send the GA task to it, 4) GA master manages GA operators such as initial population generation, crossover, mutation and generation alternation, 5) when the evaluation of an individual is necessary by a process of GA, GA master invokes GA evaluation worker nodes, and sends multiple trials of evaluation tasks to the worker nodes, 6) each GA evaluation worker node executes the ABS based on the parameter values of each individual, and evaluates it, 7) as a result of simulation, the evaluation value which expresses how well the desired objectives are able to be achieved is returned to GA master, the workflow from step (5) to (7) are repeated until inverse simulation is finished, then 8) the simulation result is returned to trials master, and finally, 9) an optimal parameter values are obtained by GA result. Through such a procedure, we obtain a model structure by systematic parameter optimization through inverse simulation, which represents the desired objectives.

Model Selection Function. Fig. 5 shows the constitution of computation nodes and the process of implementation of model selection function. Model selection is implemented through a two-layer optimization mode: the first-layer optimization selects a subset of parameters and procedures as features and the second-layer optimization employs it as input variables to execute inverse simulation. The results of objective function of inverse simulations are used as external criterions to evaluate the efficiency of selected feature sets. As shown in Fig.5, Trials master on the user terminal manage GA operations of feature selection; GA masters and GA evaluation worker nodes of the remote cluster execute a group of inverse simulations based on the selected feature sets.

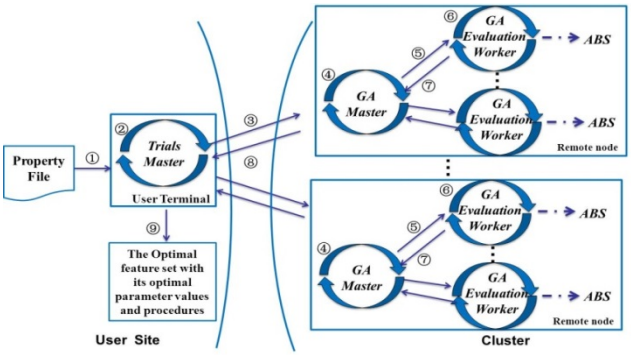

Fig. 5. The implementation of model selection function

Taken GA as the specified optimization algorithm, model selection is carried out as follows: 1) *Trials master* reads the experimental settings based on the property files, 2) According to the experimental settings, *trials master* executes the first-layer GA for feature selection, each subset of parameters and procedures is managed as a feature set and expressed as an individual of GA, *trials master* generates multiple inverse simulation tasks based on these feature sets, and then 3) *Trials master* invokes *GA masters* on the remote cluster, and sends feature sets to them, from step (4) to (7) the second layer GA of inverse simulations are executed based on the parameters values and procedures within the feature sets, 8) the evaluation results obtained from inverse simulations based on the feature set are returned to the *Trials master*, and finally, 9) the optimal feature set with its optimal parameter values and procedures is achieved. Through such a process, we obtain a set of candidate model structures by systematic parameter and procedure optimization through two-layer GA, which meet with the desired objectives.

Property Files of SOMAS. Three types of property files are prepared to use SOMAS: (1) grid property file, (2) SOMAS property file, and (3) simulation property file. *Grid property file* is a shell file which determines a grid-based simulation environment and the total number of computation nodes. This file is made based on the settings of the remote cluster side to use. *SOMAS property file* sets the simulation method and the constitution of computation nodes of the remote cluster. For the simulation methods, forward-, inverse simulation or model selection can be chosen. *Simulation property file* is implemented as forward simulation property file, inverse simulation property file, or model selection property file (which consists of feature selection property file and inverse simulation property file), depending on the simulation method chosen to conduct the agent-based models. Forward simulation property file sets the configurations to generate plural combinations of parameter values from a predefined data file and based on these parameter values to conduct simulations. Inverse simulation property file sets the operators of GA. In model selection, the feature selection property file sets the GA operations for feature selection, and the inverse simulation property file is set up as same in inverse simulation.

3.6 Procedure of Using SOMAS

Because the necessary software to run SOMAS on PC cluster are only Linux, SSH and Java (which are all readily available and the installation is easy to finish), if Job Management System, such as Sun Grid Engine, Oracle Grid Engine, Torque, PBS Pro, etc., is installed, then users can share the system efficiently. The general procedure of using SOMAS to conduct ABS on GMPUC are described as follows: 1) first, we build the agent-based model by implementing the common java programming interface, 2) second, we set grid property file to determine the sitemap and the number of computation nodes on GMPUC, and then 3) we set SOMAS property file to decide forward-, inverse-simulation or model selection method to conduct ABS, and we also set the constitution of computation nodes of the remote clusters, 4) under each case, we set configurations and parameters in forward-, inverse simulation or model selection property file, and finally 5) we conduct ABS experiments through forward-, inverse-simulation or model selection method, and summary the simulation results.

4 Experiments and Discussions

This section describes one case study of history simulation domain on SOMAS. The case study presents computational and methodological extensions of civil service examinations, family lines, and cultural capitals in imperial China. The motivation is to demonstrate the applicability and efficiency of SOMAS on parallel exploration on ABS models with vast parameter space, through forward-, inverse-simulation and model selection methods on a grid-based simulation environment. We build a grid test bed of GMPUC type, which consists of computation nodes from two public clusters: DIS cluster in Suzukakedai campus, and TSUBAME cluster in Ookayama campus, both located at Tokyo Institute of Technology. We assume in this study that each run of the simulation models does not need to communicate with any of other runs. It would therefore seem justifiable to distribute multiple runs of ABS to plural computation nodes on a grid-based simulation environment.

4.1 Case Study: Analysis of Family Strategy in Cultural Capital Reproduction

Description of History Simulation. In this case study, an agent-based model is designed to investigate the role of parental relationships and intergenerational reproduction of cultural capital to understand the long-term professional success of an elite family line during the Ming and Qing dynasties in imperial China [4][16]. In the model, cultural capital is reproduced by different family strategies, and the variables as parameters which characterize the strategies determine the result of cultural capital reproduction along with the family line. The parameters are the rate of cultural transmission from family roles, such as great-grandfather, grandfather, etc. to child, and the coefficient between knowledge cultural capital and artistic cultural capital [4] [16].

In the following, we implement the history simulation model on SOMAS, and compare the simulation results through forward-, inverse simulation and model selection methods.

Forward Simulation Test. In forward simulation, we set combinations of different values of parameters and execute 729/4, 096/15, 625 trials. The square error of two types of cultural capitals between simulated result and actual data has been used for evaluation, tallying by each agent. The smaller the square error is, the better the result. By running multiple trials of ABS models with respective parameter sets on plural computation nodes, we choose the best parameter set with its values to describe the successful family norm in civil service examinations in imperial China.

The result is: the grandfather, the father, the uncle and the mother have great influence to pass on cultural capitals to the child.

Inverse Simulation Test. In this case study, inverse simulation uses a real-coded GA through MGG [17] and UNDX [18] to optimize the parameters. The experimental configurations of history simulation through GA-based inverse simulation method are: selection is by best and rank-based roulette, the crossover of MGG is set to 100, the number of initial societies is 200, and the maximum generation is 5, 000. The value of alpha and beta in UNDX were set according to the suggestion in [18].

The result is that a combined influence of the school, the grandfather, the father, the mother, the uncle and the aunt works well to pass on cultural capitals to the child.

Model Selection Test. In model selection, we reexamine the model with subset of parameters and candidate procedures in order to obtain a more accurate model structure. The experimental configurations of first-layer GA for selecting feature set are: selection by best and rank-based roulette, uniform crossover, mutation rate is 0.05, the number of initial populations is 50, the number of child per each generation is 200, and the maximum generation is set to 500. Moreover, the configurations of second-layer GA for inverse simulation are: selection by best and rank-based roulette, crossovers of MGG is 100, the number of initial societies is 200, and the maximum generation is set to 5, 000.

By model selection, we sort the obtained feature sets by their fitness values in an ascending order and the top five patterns are: 1) school, grandfather, mother, uncle and aunt, 2) school, great grandfather, grandfather, mother, uncle and aunt, 3) school, great grandfather, grandfather, father, mother, uncle and aunt, 4) school, great grandfather, father, mother, uncle and aunt, 5) school, grandfather, father, mother, uncle and aunt. The result of inverse simulation (the fifth pattern) can also be found by model selection, while the first pattern with more accurate fitness values is never found in inverse simulation. Such a pattern explains the same macro phenomenon by a simpler model structure with a subset of parameters and candidate procedures, that is, besides school education, the combined influence of grandfather, mother, uncle and aunt are important to transmit cultural capitals to the child.

Speed Test. The experimental results on speed test of history simulator are summarized from Table 2 to Table 4, which suggest that SOMAS works well. However, because heterogeneous nodes of the grid test bed vary in their computing capabilities, the execution time does not show a linear speed up rate when adding nodes to the grid test bed.

Table 2. The experimental results on speed test by forward simulation (628 agents)

The number of nodes	Execution time		
	3^6	4^6	5^6
1	0° 04′ 55″	0° 31′ 40″	1° 56′ 14″
50	0° 00′ 14″	0° 01′ 09″	0° 04′ 12″
100	0° 00′ 12″	0° 01′ 03″	0° 03′ 48″

Table 3. The experimental results on speed test by inverse simulation

The number of nodes	Execution time		
	326 agents	628 agents	997 agents
1	1° 25′ 26″	4° 37′ 08″	5° 56′ 34″
10	0° 09′ 03″	0° 27′ 17″	0° 33′ 22″
100	0° 02′ 17″	0° 04′ 41″	0° 05′ 39″

Table 4. The experimental results on speed test by model selection (326 agents)

The number of nodes(the number of GA master × the number of evaluation node)	Execution time
1	N/A (*1)
10 (1×10)	41° 55′ 22″
100 (10×10)	10° 14′ 48″
500 (10×50)	4° 05′ 14″

(*1) The calculation did not finish in three days.

4.2 Scalability Test of SOMAS

Furthermore, we carry out experiments to confirm the scalability of SOMAS on the grid test bed, through inverse simulation method. The experiments set the number of child per each generation of MGG to 200 and use Parallel MGG algorithm (PMGG), which equips parallel generation replacement and evaluation mechanisms at the same time [18]. Because heterogeneous computation nodes on the grid test bed vary in their computational capabilities, we select the longest time consumption of one trial of history simulation as a standard unit time, and then evaluate all execution times again bases on this standard unit time for normalization. Fig.6 shows the normalized results, which confirms that the rate of speed up is almost linear for up to 100 CPUs.

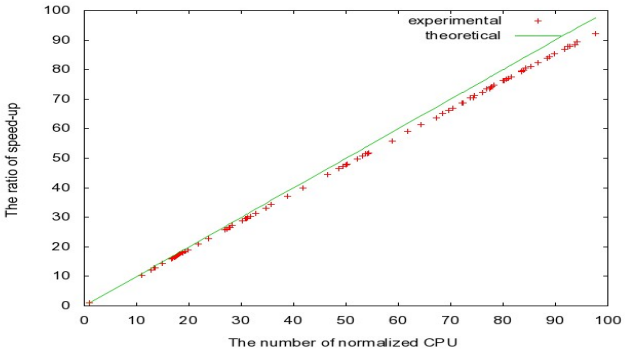

Fig. 6. The result of scalability test of SOMAS

5 Concluding Remarks

In this paper, we have proposed a grid-based simulation environment, named Social Macro Scope (SOMAS) for agent-based model with vast parameter space. The main contributions of SOMAS are: 1) the simulation method libraries which enable running agent-based models through various simulation methods, so as to meet with different simulation objectives; 2) smooth execution of three types of simulation methods of agent-based models (forward- and inverse-simulation as well as model selection) through a common Java interface, especially, a new capability of model selection to find simpler and more accurate model structure; 3) distributed experimental execution processes to search for a vast parameter set and candidate procedure set by employing grid technologies; furthermore, 4) SOMAS has successfully developed one ABS case study on history simulation domain, we have obtained the simulated result within a short CPU time, new findings have been found by model selection method.

Intensive experiments have confirmed the practicability and effectiveness of SOMAS for large experiments with multiple runs of ABSs through various simulation methods, independent of the agent-based model itself. We have also confirmed that the proposed SOMAS framework has a good scalability up to 100 CPUs on a grid test bed.

Our future work is to develop more popular libraries and more case studies on SOMAS, to confirm its robustness and maximize its usability. We are also going to implement convenient libraries for automatic data analysis of simulation results.

References

1. Takahashi, S., Sallach, D., et al.: Advancing Social Simulation: the First World Congress. Springer (2007)
2. Demazeau, Y., Ishida, T., Corchado, J.M., Bajo, J. (eds.): PAAMS 2013. LNCS, vol. 7879. Springer, Heidelberg (2013)

3. Terano, T.: Exploring the Vast Parameter Space of Multi-Agent Based Simulation. In: Antunes, L., Takadama, K. (eds.) MABS 2006. LNCS (LNAI), vol. 4442, pp. 1–14. Springer, Heidelberg (2007)
4. Yang, C., Kurahashi, S., Kurahashi, K., Ono, I., Terano, T.: Agent-Based Simulation on Women's Role in a Family Line on Civil Service Examination in Chinese History. Journal of Artificial Societies and Social Simulation **12**(25) (2009)
5. Yamamoto, G., Mizuta, H., Tai, H.: A Platform for Massive Agent-based Simulation and its Evaluation. The First International Workshop on Coordination and Control in Massively Multi-Agent Systems (2007)
6. Chen, D., Theodoropoulos, K.G., Turner, T.S., Cai, W.T., Minson, R., Zhang, Y.: Large scale agent-based simulation on the grid. Future Generation Computer Systems **24**(7), 658–671 (2008)
7. Pignotti, et al.: A semantic workflow mechanism to realise experimental goals and constraints. In: Proceedings of the 3rd workshop on Workflows in support of large-scale science, Works-08, Austin, Texas (2008)
8. Imade, H., Morishita, R., Ono, I., Ono, N.: A grid-oriented genetic algorithm framework for bioinformatics. New Generation Computing **22**(2), 177–186 (2004)
9. Ono, I., Terano, T., Okamoto, M.: A proposal of grid-oriented genetic algorithm framework 2 (in Japanese). In: Proceedings of the 35th SICE Symposium on Intelligent Systems (SICE 2008), pp. 154–159 (2008)
10. Axelrod, R.: The Dissemination of Culture: A Model with Local Convergence and Global Polarization. Journal of Conflict Resolution **41**(2), 203–226 (1997)
11. Huet, S., Edwards, M., Deffuant, G.: Taking into Account the Variations of Neighbourhood Sizes in the Mean-Field Approximation of the Threshold Model on a Random Network. Journal of Artificial Societies and Social Simulation **10**(1) (2007)
12. Kurahashi, S., Minami, U., Terano, T.: Why not multiple solutions: agent-based social interaction analysis via inverse simulation. In: IEEE International Conference on System, Man, and Cybernetics, (SMC99), 2048 (1999)
13. Terano, T., Kurahashi, S.: Inverse simulation: genetic-algorithm based approach to analyzing emergent phenomena. In: Proceedings of the International Workshop on Emergent Synthesis (IWES 1999), pp. 271–276 (1999)
14. Liu, H., Motoda, H.: Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series) (2007)
15. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (2004)
16. Yang, C., Kurahashi, S., Kurahashi, K., Ono, I., Terano, T.: Pattern-Oriented Inverse Simulation for Analyzing Social Problems: Family Strategies in Civil Service Examination in Imperial China, Advances in Complex Systems **15**(7) (2012)
17. Sato, H., Ono, I., Kobayashi, S.: A New Generation Alternation Model of Genetic Algorithms and Its Assessment. Journal of the Japanese Society for Artificial Intelligence **12**(5), 734–735 (1997)
18. Ono, I., Kita, H., Kobayashi, S.: A real-coded genetic algorithm using the unimodal normal distribution crossover. In: Ghosh, A., Tsutsui, S., (eds.) Advances in Evolutionary Computing (2002)