# Use of High Performance Computing in Agent-Based Social Simulation: A Case Study on Trust-Based Coalition Formation

Luciano M. Rosset, Luis G. Nardin and Jaime S. Sichman
Laboratório de Técnicas Inteligentes – EP/USP
Av. Prof. Luciano Gualberto, 158 – trav. 3
05508-970 – São Paulo – SP – Brasil
{luciano.rosset,luis.nardin}@usp.br, jaime.sichman@poli.usp.br

*Abstract*—**Computer models based on agents have shown to be very useful in the field of social simulation, especially for its versatility and ease to model complex systems. In the context of agent simulations, Nardin and Sichman developed a model for the study of coalition formation among agents, based on trust. Later, the need of more efficient ways for simulating the model was acknowledged in order to explore large-scale scenarios. The solution found was the adoption of a high performance agent-based computing platform. This article intends to explore the use of such platform on the Nardin and Sichman's model by migrating its code to the Repast HPC tool, which is executed on the Blue Gene/P supercomputer.**

## I. Introduction

In the last decades, computer simulation has proved to be a viable approach for science, in addition to the traditional deductive and inductive approaches [1]. According to Banks [2], computer simulation consists on the reproduction of real systems through computer models, making the study of these systems' dynamics possible without interfering on them. The application of this approach to the study of social science problems enables the reproduction of social systems behaviours through the use of computational methods, which is called social simulation.

Adequate modelling of social systems is essential for obtaining a useful simulation. Among the available modelling paradigms, agent-based modelling has many adequate abstractions for representing such systems: the main one is the agent abstraction. A comprehensive definition proposed by Ferber [3] states that an agent is a physical or virtual entity who (a) is capable of acting in an environment; (b) is capable of communicating with the others; (c) is driven by a set of tendencies (individual goals to achieve or a satisfaction utility to optimize); (d) has its own resources; (e) is capable of perceiving the environment (limited perception); (f) has (eventually) a partial representation of this environment; (g) has competences and offers services; (h) may eventually reproduce itself; and (i) tends to behave in order to satisfy its goals using the available resources and competences and taking into consideration its internal perceptions, representations and the received communication.

According to Davidsson [4], the intersection between social simulation and agent-based computing brings forth a new area denoted Agent-Based Social Simulation (ABSS), whose main purpose is to provide models and tools for simulating social phenomena.

In the context of ABSS, Nardin and Sichman proposed an agent-based simulation model, named *Trust and Coalition* (or simply T&C), that integrates the notions of coalition formation and trust in order to enable the analysis of the impacts of trust on the formation of partnerships among autonomous agents [5]. This model was implemented using NetLogo [6], which is an educational agent modelling and simulating environment. Based on this implementation, several experiments were carried out identifying that the use of trust is relevant for the formation of partnership in fully heterogeneous scenarios in which the agents have high levels of trust intolerance and volatility [7].

Although these simulations were successfully performed and provided data for analyses concerning the correlation between trust and coalition formation, there is an uncertainty about the influences that different environment features, such as scale and topology, may cause on such correlation. Thus, in order to analyse such influences, the execution of simulations considering larger populations, such as millions of individuals, is required. However, since the T&C model execution is highly computational demanding, simulations comprising a larger agent population would require an unmanageable amount of time for execution using conventional Agent-Based Modelling and Simulation (ABMS) tools, e.g. NetLogo. For instance, it takes about 1 hour to run a simulation considering 2,500 agents in an Intel i5 2.5 GHz and 4 GB RAM.

High Performance Computing (HPC) is a good path to follow in order to fulfil the needs of large-scale simulations [8]. Therefore, in a first step towards an analyses of the T&C model in a large-scale environment, we decided to migrate its previous NetLogo implementation to Repast for High Performance Computing (Repast HPC) [9], and this article aims to present some details of such implementation.

The remainder of the article is organized as follows. In section II, we briefly present Repast HPC and its main features, as well as the motivation for selecting this tool. An overview of the *Trust and Coalition* simulation model is presented in Section III and its implementation using Repast HPC is described in Section IV. Since this is a ongoing work, we describe in Section V some of our intended future work.

## II. Repast HPC

Some ABSS models demand great computational capabilities and consume an unmanageable amount of time and memory when simulated monolithically, i.e., the model's simulation is performed using a single process. The single process has not only strict processing restrictions, but may also be overwhelmed by memory needs. In order to overcome the monolithic approach limitations and decrease the simulation execution time, some approaches were proposed. Among these different approaches, the most common ones are (i) *parallel computing* in which one computer composed of several processors execute the simulation in parallel, and (ii) *distributed computing* in which several different computers connected via a network process the simulation in parallel [10]. The former is known to be the fastest option as it has a reduced communication overhead, but no software tool capable of executing agent models based on this approach is available. Based on the *distributed* approach, several software tools were proposed lately, such as SWAGES [11], FLAME [12] and Repast HPC [9].

Among these tools, we chose to use Repast HPC in this work because it is the one that presents the greatest advantages concerning flexibility and general use, due to its easy structure based on contexts and projections [9]. Moreover, it handles transparently all required inter-process communication and synchronizes agents status in different processes, when needed, in order to optimize cross-process information sharing. Additionally, Repast HPC is the only available, tested and supported platform that runs in a Blue Gene/P supercomputer, which is the target machine for performing our large-scale experiments.

Repast HPC is a cross-platform C++ based environment for large-scale ABMS. It was developed to run large-scale agent models which complexity or number would overwhelm a single process. Its focus is on enabling distributed runs over multiple processes that communicate and share agents using Message Passing Interface (MPI). Each individual process is responsible for executing the behaviours of a subset of all agents in the simulation. Therefore, each process has a scheduler, a context and the projections associated with the context. Repast HPC core components are:

- *AgentId* – A number that uniquely identifies an agent in the simulation. It is composed of the agent's *identity number*, *process rank* and *type*. The *identity number* is a number unique in a specific *process rank*. The *process rank* is the process number the agent is associated to, being unique for each process. The *type* is a number that specifies a class of agents, with the same behaviour.
- *Context* – It is a simple container that groups agents of the same type together; the developer accesses the agents properties through the context.
- *Projection* – It imposes a structure in which the agents are organized. The structure defines relationships among the agents using the semantics of a projection. Three projections are provided: *network*, which consists of a set of nodes and links between them, defining a graph; *grid*, where topological information is included, and agents may occupy a one, two or three-dimensional matrix, that eventually may be wrapped (a 2D wrapped grid works as a torus); and *continuous space*, which basically is a grid which coordinates are floating-points.
- *Scheduler* – It allows agents to schedule events and avoids processes beginning new tasks before other processes end their current ones, which guarantees a consistent simulation execution.
- *Data Collection* – It gathers agents information and write them into files/structures. It allows to produce output files, composed of aggregated or non-aggregated data from all simulation processes.

All simulations in Repast HPC are managed by a set of controllers, one per process, which manages all components such as *Contexts*, *Projections*, *Data Collections* and *Scheduler*. In the beginning of the simulation, each controller creates agents, assigns to each one an *AgentId* and associates them with a *Context*. *Projections* are then created and associated to the *Context* for further positioning of the agents. After this initial setup, the controllers create several events, which trigger actions execution. The sequence of the organization and execution of the events is performed by the *Scheduler*. These events generation and actions execution are performed in steps named *ticks* and the simulation runs until a predefined condition is met or a specified number of ticks is reached. Besides executing the agents behaviours, Repast HPC may also gather useful information during the simulation, through its *Data Collection* feature, writing them into output files.

## III. Trust and Coalition Model

The simulation model used in this work is the *Trust and Coalition* (T&C) proposed by Nardin and Sichman [5]. The simulation model proposes a spatial Prisoner Dilemma (PD) game that integrates the notions of coalition formation and trust, which purpose is to enable the analyses of the impacts of trust on the formation of partnerships through coalitions.

The simulation model is composed of an environment represented by a grid and a group of agents, each of them located at one position of the grid. It runs iteratively for a specified number of cycles. At each cycle, each agent interacts with its neighbours (von Neumann or Moore neighbourhood) playing a 2-players PD game, with the following payoff matrix values: *T*emptation=5, *R*eward=3, *P*unishment=1, and *S*ucker=0.

At each cycle, each agent is in either one of the possible states: *independent*, *coalition leader*, or *coalition member*. When *independent*, the agent is not associated to any coalition and chooses between cooperating or not with its neighbours based on its own strategy. A strategy is randomly assigned to the agent at the beginning of the simulation, and there are 3 different possible strategies: Tit-For-Tat (TFT), Probabilistic Tit-For-Tat (pTFT), or Random. When the agent is a *coalition leader* or a *coalition member*, it always cooperates with the neighbours of the same coalition and does not with other neighbours, representing its commitment to the group and its trust on its leader.

Based on the agent's state, its payoff is calculated differently. The *independent* agent payoff is the sum of all the payoffs obtained by playing the PD game against its neighbours.

The *coalition leader* agent payoff is a tax calculated over the sum of all its *coalition members* payoffs. The *coalition member* payoff is defined as follows: (i) the sum of all the *coalition members* payoff is calculated, (ii) the value paid as tax to the *coalition leader* is subtracted from this value, and (iii) each *coalition member* receives the even division of this remaining value by the number of *coalition members*.

After the payoff is calculated, each agent can change its state. The *independent* agent decides to associate to a coalition if its payoff is the smallest amongst all its neighbours. In this case, the agent associates with the coalition with greatest payoff. If both agents are *independent*, then a new coalition is formed and the agent with the greatest payoff becomes the *coalition leader* and the other the *coalition member*. On the other hand, the *coalition member* remains or leaves a coalition based on a *trust* value in its *coalition leader*. If its payoff is not the greatest amongst its neighbours, it decreases the trust in the *coalition leader*, otherwise it increases it. When the *trust* value decreases to a value below a *trust threshold*, then the agent leaves the coalition and becomes *independent*, otherwise it remains in the coalition. The *coalition leader* becomes independent only when its coalition disppears.

## IV. Implementation

As the T&C model has only one *Type* of agent, our Repast HPC model uses a single *Context* to hold $n_x x n_y$ agents. A two-dimensional grid *Projection* is created, and each agent is located in one cell of grid. The grid is then divided into $m_x$ by $m_y$ processes ($n_x$ and $n_y$ must be respectively multiple of $m_x$ and $m_y$). Since the agents positioned in the border of the grid require to interact with agents running on other processes, there is a component handled by Repast HPC, called *buffer*, that synchronizes this exchange of data. Therefore, each controller accounts for $n_x/m_x * n_y/m_y$ agents (grid cells, in fact) plus $2(n_x/m_x + n_y/m_y) + 4 * 2^{(size(buffer)-1)}$ agents taking into account the buffered ones.

The sequence of methods execution divides the events into smaller ones, corresponding for agent's decisions, payoff calculation and coalition management, each of them scheduled to run simultaneously by all processes.

The only limitation we found in Repast HPC was the data collection feature, which does not allow gathering data independently from individual agents, but only by process. Therefore, we will use HDF5[1], since it allows data collection of individual agents similarly to what was done in [8].

## V. Future Work

Since this is an ongoing project, we are still migrating the model to Repast HPC and no results are currently available. At the moment, we have performed some preliminary tests using Repast HPC and migrated part of the simulation model. However, as soon the migration is completed, we shall perform some further analyses considering:

- *larger populations* – This study has a social approach and one very important point to tackle is the effect of different population sizes. HPC allows us to study the behaviour of populations with the size of entire cities;
- *different topologies* – We want to analyse the effect of narrower rectangular grids, wrapped grids (torus) and other neighbourhoods;
- *more detailed parameter sweep* – Earlier NetLogo simulations were made with a very large variation of the parameters. One example is the *tax* paid by *coalition members* that ranged from 0% to 100% varying by 25%. HPC will allow swifter simulations, therefore making it possible to perform a larger number of simulations.

[1]http://www.hdfgroup.org/HDF5/

## Acknowledgements

Luciano M. Rosset and Jaime S. Sichman are partially supported by CNPq/Brazil. We would like to acknowledge the computing time provided on the Blue Gene/P supercomputer supported by the Research Computing Support Group (Rice University) and Laboratório de Computação Científica Avançada (LCCA-CCE, Universidade de São Paulo).

## References

[1] R. Axelrod, "Advancing the art of simulation in the social sciences," *Complexity*, vol. 3, no. 2, pp. 16–22, 1997.

[2] J. Banks, Ed., *Handbook of Simulation : Principles, Methodology, Advances, Applications, and Practice*. New York: John Wiley & Sons, 1998.

[3] J. Ferber, *Les Systèmes Multi-Agents: Vers une Intelligence Collective*, ser. Informatique, Intelligence Artificielle. Paris: InterEditions, 1995.

[4] P. Davidsson, "Agent based social simulation: A computer science view," *Journal of Artificial Societies & Social Simulation*, vol. 5, no. 1, p. 7, 2002. [Online]. Available: http://jasss.soc.surrey.ac.uk/5/1/7.html

[5] L. G. Nardin and J. S. Sichman, "Simulating the impact of trust in coalition formation: A preliminary analysis," *Advances in Social Simulation, Post-Proceedings of the Brazilian Workshop on Social Simulation*, pp. 33–40, 2011.

[6] U. Wilensky, *NetLogo*, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, 1999. [Online]. Available: http://ccl.northwestern.edu/netlogo/

[7] L. G. Nardin and J. S. Sichman, "Trust-based coalition formation: A multiagent-based simulation," in *Proceedings of the 4th World Congress on Social Simulation*, Taipei, TW, 2012.

[8] J. T. Murphy, "Computational social science and high performance computing: A case study of a simple model at large scales," in *Proceedings of the 2011 Computational Social Science Society of America Annual Conference*, Santa Fe, 2011. [Online]. Available: http://computationalsocialscience.org/conferences/17-2/csssa-2011-papers

[9] N. Collier and M. North, "Parallel agent-based simulation with repast for high performance computing," *SIMULATION:Transactions of the Society for Modeling and Simulation International*, pp. 1–21, 2012.

[10] R. M. Fujimoto, *Parallel and Distributed Simulation Systems*, 1st ed., ser. Wiley series on parallel and distributed computing. New York: John Wiley & Sons, 2000.

[11] M. Scheutz, P. Schermerhorn, R. Connaughaton, and A. Dingler, "SWAGES: an extendable distributed experimentation system for large-scale agent-based ALife simulations," in *Proceedings of the 10th International Conference on the Simulation and Synthesis of Living Systems*, 2006.

[12] S. Coakley, M. Gheorghe, M. Holcombe, S. Chin, D. Worth, and C. Greenough, "Exploitation of high performance computing in the FLAME agent-based simulation framework," in *High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*, 2012, pp. 538–545.