Simulation of Minority Game in TuCSoN

Enrico Oliva
Mirko Viroli
Andrea Omicini
ALMA MATER STUDIORUM—Università di Bologna
via Venezia 52, 47023 Cesena, Italy
E-mail:{enrico.oliva,mirko.viroli,andrea.omicini}@unibo.it

I. APPLICATION DOMAIN

Minority Game (MG) is a mathematical model that takes inspiration from the "El Farol Bar" problem introduced by Brian Arthur (1). It is based on a simple scenario where at each step a set of agents perform a boolean vote which conceptually splits them in two classes: the agents in the smaller class win. In this game, a rational agent keeps track of previous votes and victories, and has the goal of winning throughout the steps of the game—for which a rational strategy has to be figured out.

One of the most important applications of MG is in the market models: (2) use MG as a coarse-grained model for financial markets to study their fluctuation phenomena and statistical properties. Even though the model is coarse-grained and provides an over-simplified micro-scale description, it anyway captures the most relevant features of system interaction, and generates collective properties that are quite similar to those of the real system.

Another point of view, presented e.g. by (3), considers the MG as a point in space of a Resource Allocation Game (RAG). In this work a generalisation of MG is presented that relaxes the constraints on the number of resources, studying how the system behaves within a given range.

MG can be considered a social simulation that aims to reproduce a simplified human scenario. In principle, a logic-based approach based on BDI agent makes it easier to explicitly model a variety of diverse social behaviours.

As showed by (4), a multiagent system (MAS) can be used to realise a MG simulation—there, BDI agents provide for rationality and planning. An agent-based simulation is particularly useful when the simulated systems include autonomous entities that are diverse, thus making it difficult to exploit the traditional framework of mathematical equations.

In order to implement MG simulations we adopt the TuCSoN infrastructure for agent coordination (5), which introduces tuple centres as artifact representatives. A tuple centre is a programmable coordination medium living in the MAS environment, used by agents interacting by exchanging tuples (logic tuples in the case of TuCSoN logic tuple centres). As we are not concerned much with the mere issues of agent intelligence, we rely here on a weak form of rationality, through logic-based agents adopting pre-compiled plans called *operating instructions* (6).

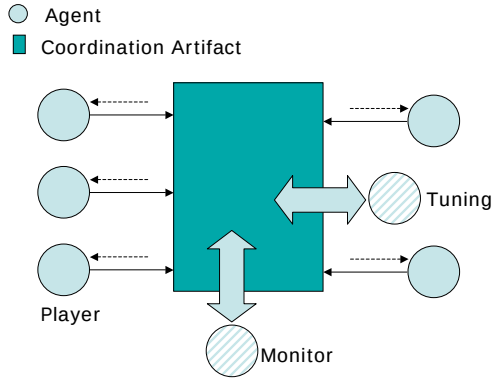

Fig. 1. TuCSoN Simulation Framework for MG

II. LOGIC ARCHITECTURE

The architecture proposed for MAS simulation is based on TuCSoN (5), which is an infrastructure for the coordination of MASs. TuCSoN provides agents with an environment made of logic tuple centres, which are logic-based programmable tuple spaces. The language used to program the coordination behaviour of tuple centres is ReSpecT, which specifies how a tuple centre has to react to an observable event (e.g. when a new tuple is inserted) and has to accordingly change the tuple-set state (7). Tuple centres are a possible incarnation of the coordination artifact notion (8), representing a device that persists independently of agent life-cycle and provides services to let agents participate to social activities.

In our simulation framework we adopt logic-based agents, namely, agents built using a logic programming style, keeping a knowledge base (KB) of facts and acting according to some rule—rules and facts thus forming a logic theory. The implementation is based on tuProlog technology[1] for Java-Prolog integration, and relies on its inference capabilities for agent rationality. Agents roughly follow the BDI architecture, as the KB models agent beliefs while rules model agent intentions.

Three kinds of agents are used in our simulation: player agents, monitor agents and tuning agents (as depicted in Figure 1): all the agents share the same coordination artifact. The agent types differ because of their role and behaviour: player agents play MG, the monitor agent is an observer

[1] http://tuprolog.alice.unibo.it

of interactions which visualises the progress of the system, the tuning agent can change some rules or parameters of coordination, and drives the simulation to new states. Note that the main advantage of allowing a dynamic tuning of parameters instead of running different simulations lays in the possibility of tackling emergent aspects which would not necessarily appear in new runs.

The main control loop of a player agent is a sequence of actions: observing the world, updating its KB, scheduling next intention, elaborating and executing a plan. To connect agent mental states with interactions we use the concept of action preconditions and perception effects as usual.

III. Minority Game Performance

To track the performance of an MG system, the most interesting quantity is *variance*, defined as $\sigma^2 = \overline{[A(t) - \overline{A(t)}]^2}$: it shows the variability of the bets around the average value $\overline{A(t)}$. In particular, the normalised version of variance $\rho = \sigma^2/N$ is considered. Figure 3 shows a typical evolution of the game.

Generally speaking, variance is the inverse of global efficiency: as variance decreases agent coordination improves, making more agents winning. Variance is interestingly affected by the parameters of the model, such as number of agents (N), memory (m) and number of strategies (s): in particular, the fluctuation of variance is shown to depend only on the ratio $\alpha = 2^m/N$ between agent memory and the number N of agents.

The results of observations suggest that the behaviour of MG can be classified in two phases: an information-rich *asymmetric* phase, and an unpredictable or *symmetric* phase. A phase transition is located where σ^2/N attains its minimum ($\alpha_c = 1/2$), and it separates the symmetric phase with $\alpha < \alpha_c$ from an asymmetric phase with $\alpha > \alpha_c$.

All these cases have been observed with the TuCSoN simulation framework described in next section.

IV. The Simulation Framework

The construction of MG simulations with MASs is based on the TuCSoN framework and on tuProlog as an inferential engine to program logic agents. The main innovative aspect of this MG simulation is the possibility of studying the evolution of the system with particular and different kinds of agent behaviour at the micro level, imposed as coordination parameters which are changed on-the-fly.

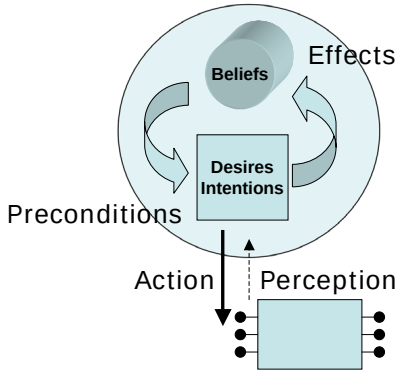

Fig. 2. Agent Architecture

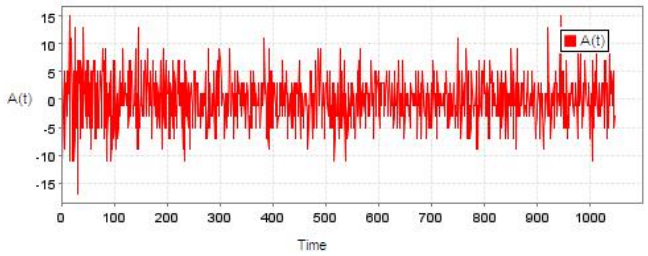

Fig. 3. Typical Time evolution of the Original MG with $N = 51$, $m = 5$ and $s = 2$

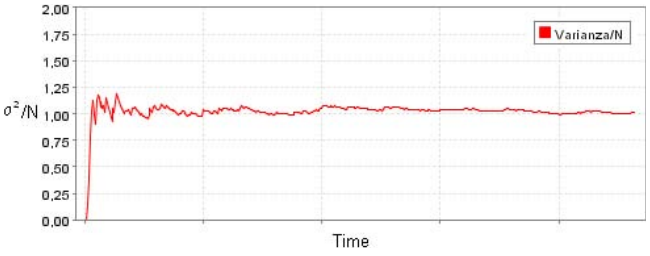

Fig. 4. Variance of the Game with 11 Random Agents

A. Operating Instructions

Each agent has an internal plan, structured as an algebraic composition of allowed actions (with their preconditions) and perceptions (with their effects), that enables the agent to use the coordination artifact to play the MG. This plan can be seen as Operating Instructions (6), a formal description based on Labelled Transition System (LTS) that the agent reads to understand what its step-by-step behaviour should be. Through an inference process, the agent accordingly chooses the next action to execute, thus performing the cycle described in Section 2.

Operating instructions are expressed by the following theory:

```
firststate(agent(first,[])).
definitions([
  def(first,[],...),
  def(main,[S],
       [act(out(play(X)),pre(choice(S,X))),
        per(in(result(Y)),eff(res(Y))),
        agent(main,[S])]
  ),
  ...
]).
```

The first part of operating instructions is expressed by term `first`, where the agent reads the game parameters that are stored in the KB, and randomly creates its own set of strategies.

In the successive part `main`, the agent executes its main cycle. It first puts tuple `play(X)` in the tuple space, where $X = \pm 1$ is agent vote. The precondition of this action `choice(S,X)` is used to bind in the KB X with the

value currently chosen by the agent according to strategy S. Then, the agent gets the whole result of the game in tuple `result(Y)` and applies it to its KB. After this perception, the cycle is iterated again.

B. Tuple Centre Behaviour

The interaction protocol between agents and the coordination artifact is then simply structured as follows. First each agent puts the tuple for its vote. When the tuples for all agents have been received, the tuple centre checks them, computes the result of the game—either 1 or -1 is winning—and prepares a result tuple to be read by agents.

The ReSpecT program for this behaviour is loaded in the tuple centre by a configuration agent at bootstrap, through operation `set_spec()`. The following ReSpecT reaction is fired when an agent inserts tuple `play(X)`, and triggers the whole behaviour:

```
reaction(out(play(X)),(
  in_r(count(Y)),
  Z is Y+1,
  in_r(sum(M)),
  V is M+X,
  out_r(sum(V)),
  out_r(count(Z))
)).
```

This reaction considers the bet (`X`) counts the bets (`Z`) and computes the partial result of the game (`V`). When all the agents have played, the artifact produces the tuple `winner(R,NS,NumberOfLoss,MemorySize,last/more)` which is the main tuple of MG coordination.

```
reaction(out_r(count(X)),(
  rd_r(numag(Num)),
  X=:=Num,
  in_r(totcount(T)),
  P is T+1,
  rd_r(game(G)),
  in_r(sum(A)),
  out_r(sum(0)),
  rd_r(countsession(CS)),
  in_r(count(Y)),
  out_r(count(0)),
  %%calculate variance
  in_r(qsum(SQ)),
  NSQ is A*A+SQ,
  out_r(qsum(NSQ)),
  %%calculate mean
  in_r(totsum(R)),
  NewS is R+A,
  out_r(totsum(NewS)),
  rd_r(numloss(NumberOfLoss)),
  rd_r(mem(MemorySize)),
  out_r(winner(A,P,CS,NumberOfLoss,MemorySize,G)),
  out_r(totcount(P))
)).
```

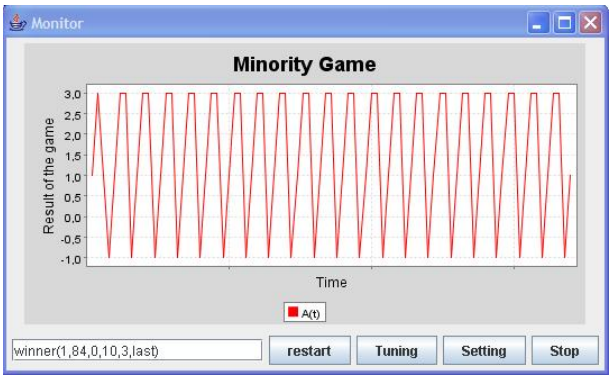

Fig. 5. Interface of the Monitor Agent

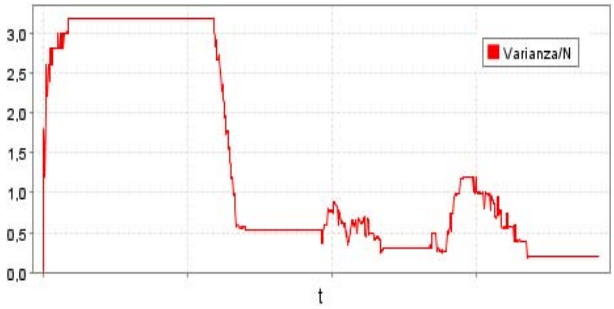

Fig. 6. Variance of the System with Initial Parameters $N = 5$ and $m = 3$

The `winner` tuple contain the result of game (`R`), the number of step (`NS`), two tuning parameters (`NumberOfLoss` and `MemorySize`) and one constant to communicate agents whether they have to stop or to play further (`last/more`). Figure 5 reports the graphical interface of the monitor agent that during its life-time reads the tuple `winner` and draws variance.

C. Tuning the Simulation

In classical MG simulation there are a number of parameters that can affect the system behaviour, which are explicitly represented in the tuple centre in form of tuples: the number of agents `numag(X)`, memory size `mem(X)`, and the number of strategies `numstr(X)`. In our framework, we have introduced as a further parameter the number of wrong moves after which the single agent should be recalculate own strategy, represented as a tuple `numloss(X)`. Such a threshold is seemingly useful to break the symmetry in the strategy space when the system is in a pathological state, i.e., when all agents have the same behaviour and the game oscillates from minimum to maximum value.

In our framework, it is possible to explore the possibility to dynamically tune up the coordination rules by changing `numloss` and `mem` coordination parameters, which are stored as tuples in the coordination artifact. The simulation architecture built in this way, in fact, allows for on-the-fly change of some game configuration parameters—such as the dimension of agent memory—with no need to stop the simulation and re-program the agents.

By changing the parameters, the tuning agent can drive the system from an equilibrium state to another, by controlling agent strategies, the dimension of memory, or the number of losses that an agent can accept before discarding a strategy. This agent observes system variance, and decides whether and how to change tuning parameters: reference variance is calculated by first making agents playing the game randomly—see Figure 4. The new value of parameters is stored in tuple centre through tuples `numloss(NumberOfLoss)` and

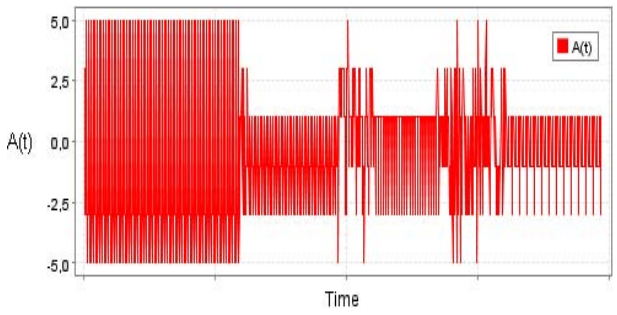

Fig. 7. System Evolution of the Variance in Figure 6

`mem(MemorySize)`, the rules of coordination react and update the information that will be read by the agents.

D. Simulation Results

The result of the tuned simulation in Figures 6 and 7 shows how the system changes its equilibrium state and achieves a better value of variance.[2] In this simulation the tuning agent is played by a human that observes the evolution of the system and acts through the tuning interface to change the coordination parameters, such as threshold of losses and memory, hopefully finding new and better configurations. The introduction of the threshold of losses in the agent behaviour is useful when the game is played by few agents: these parameters enable system evolution and a better agent cooperative behaviour.

V. Perspectives

In this paper, we aim at introducing new perspectives on agent-based simulation by adopting a novel MAS meta-model based on agents and artifacts, and by applying it to Minority Game simulation. We implement and study MG over the TuCSoN coordination infrastructure, and show some benefits of the artifact model in terms of flexibility and controllability of the simulation. In particular, in this work we focus on the possibility to build a feedback loop on the rules of coordination driving a system to a new and better equilibrium state.

We foresee some new perspectives in the use of the TuCSoN simulation framework in a industrial environment. The first one is to use the system to drive manufacturing in case of limited resources. In this scenario each agent is a half-processed item, whose production has to be completed as faster as possible, and whose access to the resources is regulated by dedicated resource artifacts. Another possible perspective is to evaluate the product demand and production in order to drive industry through market fluctuation. In our framework we could model a market scenario by minority rules, and then try to evaluate demand. Furthermore, all such applications would benefit from using a logic-based approach rather than an equation-based approach.

[2]In Figure 6, the first phase of equilibrium is followed by a second one obtained by changing the threshold parameter $S = 5$. Finally, a third phase is obtained changing the dimension of the memory to $m = 5$.

VI. Acknowledgements

The first author of this paper would like to thank Dr. Peter McBurney and the Department of Computer Science at University of Liverpool for their scientific support and their hospitality during his stay in Liverpool, when this paper was mostly written.

References

[1] W. B. Arthur, "Inductive reasoning and bounded rationality (the El Farol problem)," *American Economic Review*, vol. 84, no. 2, pp. 406–411, May 1994.

[2] D. Challet, M. Marsili, and Y.-C. Zhang, "Modeling market mechanism with minority game," *Physica A: Statistical and Theoretical Physics*, vol. 276, no. 1–2, pp. 284–315, Feb. 2000. [Online]. Available: http://dx.doi.org/10.1016/S0378-4371(99)00446-X

[3] H. V. D. Parunak, S. Brueckner, J. Sauter, and R. Savit, "Effort profiles in multi-agent resource allocation," pp. 248–255.

[4] W. Renz and J. Sudeikat, "Modeling Minority Games with BDI agents – a case study," in *Multiagent System Technologies*, ser. LNCS, T. Eymann, F. Klügl, W. Lamersdorf, M. Klusch, and M. N. Huhns, Eds. Springer, 2005, vol. 3550, pp. 71–81, 3rd German Conference (MATES 2005), Koblenz, Germany, 11-13 Sept. 2005. Proceedings. [Online]. Available: http://www.springerlink.com/link.asp?id=y62q174g56788gh8

[5] A. Omicini and F. Zambonelli, "Coordination for Internet application development," *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 3, pp. 251–269, Sept. 1999.

[6] M. Viroli and A. Ricci, "Instructions-based semantics of agent mediated interaction," in *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, Eds., vol. 1. New York, USA: ACM, 19–23 July 2004, pp. 102–109. [Online]. Available: http://portal.acm.org/citation.cfm?id=1018409.1018737

[7] A. Omicini and E. Denti, "Formal ReSpecT," *Electronic Notes in Theoretical Computer Science*, vol. 48, pp. 179–196, June 2001.

[8] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini, "Coordination artifacts: Environment-based coordination for intelligent agents," in *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, Eds., vol. 1. New York, USA: ACM, 19–23 July 2004, pp. 286–293. [Online]. Available: http://portal.acm.org/citation.cfm?id=1018409.1018752

[9] N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, Eds., *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*. New York, USA: ACM, 19–23 July 2004.