

# Minority Game: A Logic-Based Approach 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

**Abstract**—Minority Game is receiving an increasing interest because it models emergent properties of complex systems including rational entities, such as for instance the evolution of financial markets. As such, Minority Game provides for a simple yet stimulating scenario for system simulation.

In this paper, we aim at presenting a logic approach to the Minority Game whose goal is to overcome the well-known limits of the equation model in the verification of the system behaviour. We realise the social system simulation using a novel MAS meta-model based on agents and artifacts, where the agent rationality is obtained using a BDI architecture.

To this end, we adopt the TuCSoN infrastructure for agent coordination, and its logic-based tuple centre abstractions as artifact representatives. By implementing Minority Game over TuCSoN, we show some of the benefits of the artifact model in terms of flexibility and controllability of the simulation.

A number of parameters can affect the behaviour of Minority Game simulation: such parameters are explicitly represented in the coordination artifact, so that they can be tuned up during the simulation. In particular, experiments are shown where memory size and number of wrong moves are adopted as the tuning parameters.

## I. INTRODUCTION

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. Several researches showed that, although very simple, this model takes into account crucial aspects of some interesting complex systems coupling rationality with emergence: e.g. bounded rationality, heterogeneity, competition for limited resources, and so on. For instance, MG is a good model to study market fluctuation, as an emergent property resulting from interactions propagating from micro scale (agent interaction) to macro scale (collective behaviour).

As showed by (2), 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.

The Minority Game is a social simulation that aims at reproducing a simplified human social scenario. A (human)

society is composed by different kinds of people with different behaviours, and its composition affects the progress of the game. In principle, a logic-based approach based on BDI agent makes it easier to explicitly model a variety of diverse social behaviours. Also, in this scenario, argumentation theory (3) is useful to model the information exchange and sharing between humans/agents so as to improve the agent reasoning abilities, as well as to provide a more realistic simulation of a society.

In this paper we proceed along this direction, and adopt a novel MAS meta-model based on the notion of artifact (4). The notion of artifact is inspired by Activity Theory (5): it represents those abstractions living in the MAS environment that provide a function, which agents can exploit to achieve individual and social goals. The engineering principles promoted by this meta-model makes it possible to flexibly balance the computational burden of the whole system between autonomy of the agents and the designed behaviour of artifacts.

In order to implement MG simulations we adopt the TuCSoN infrastructure for agent coordination (6), 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* (7).

By implementing MG over TuCSoN, we can experiment with flexibility and controllability of the artifact model, and see if and how they apply to the simulation – in particular, artifacts allow for a greater level of controllability with respect to agents. To this end, in this paper we show how the model allows some coordination parameters to be changed during the run of a simulation with no need to stop the agents: this can be useful e.g. to change the point of equilibrium, controlling the collective behaviour resulting by interactions propagated from the entities at the micro level.

The remainder of this paper is organised as follows. First, we introduce the general simulation framework based on agents and artifacts. Then, we provide the reader with some relevant details of the Minority Game. Some quantitative results of MG simulation focussing on system dynamics and run-time changes are presented, just before final remarks.

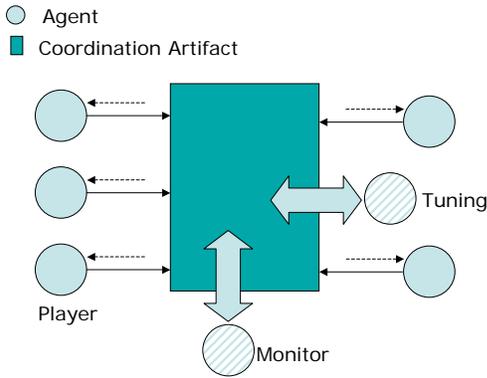


Fig. 1. TuCSoN Simulation Framework for MG

## II. THE TuCSoN FRAMEWORK FOR SIMULATION

The architecture proposed for MAS simulation is based on TuCSoN (6), 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 (8). Tuple centres are a possible incarnation of the coordination artifact notion (9), 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<sup>1</sup> for Java-Prolog integration, and relies on its inference capabilities for agent rationality. Agents roughly follow the BDI architecture (as showed in Figure 2), as the KB models agent beliefs while rules model agent intentions.

To coordinate agents we take inspiration from natural systems like ant-colonies, where coordination is achieved through the mediation of the environment: our objective is to have a possibly large and dynamic set of agents which coordinate each other through the environment while bringing about their goals.

Externally, we can observe overall system parameters by inspecting the environment, namely, the tuple centres agents interact with. In this way we can try different system behaviours changing only the coordination behaviour of the environment. Furthermore we can change, during the simulation, some coordination parameters (expressed as tuples in a tuple centre), programming and then observing the transition of the whole system either to a new point of equilibrium or to a divergence.

Three kinds of agents are used in our simulation: player agents, monitor agents and tuning agents (as depicted in

<sup>1</sup><http://tuprolog.alice.unibo.it>

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 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 (perception), updating its KB (effects), scheduling next intention (precondition), elaborating and executing a plan (action). This structure is depicted in Figure 2. Moreover, in order to connect agent mental states with interactions, we use the concept of action preconditions and perception effects as usual.

## III. MINORITY GAME

MG was introduced and first studied by (10), as a means to evaluate a simple model where agents compete through adaptation for finite resources. MG is a mathematical representation from ‘El Farol Bar’ problem introduced by (1), providing an example of inductive reasoning in scenarios of bounded rationality. The game consists in an odd number  $N$  of agents: at each discrete time step  $t$  of the game an agent  $i$  takes an action  $a_i(t)$ , either 1 or  $-1$ . Agents taking the minority action win, whereas the majority loses. After a round, the total action result is calculated as:

$$A(t) = \sum_i^N a_i(t)$$

In order to take decisions agents adopt strategies. A strategy is a choosing device that takes as input the last  $m$  winning results, and provides the action (1 or  $-1$ ) to perform in the next time step. The parameter  $m$  is the size of the memory of the past results (in bits), and  $2^m$  is therefore the potential past history that defines the number of possible entries for a strategy.

The typical strategy implementation is as follows. Each agent carries a sequence of  $2^m$  actions, called a strategy, e.g.

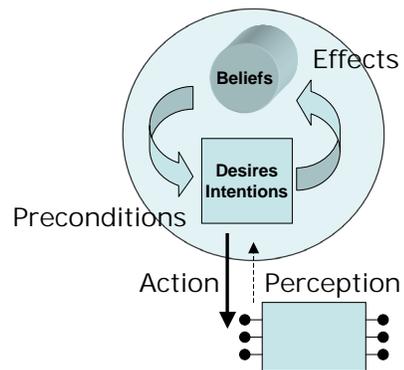


Fig. 2. Agent Architecture

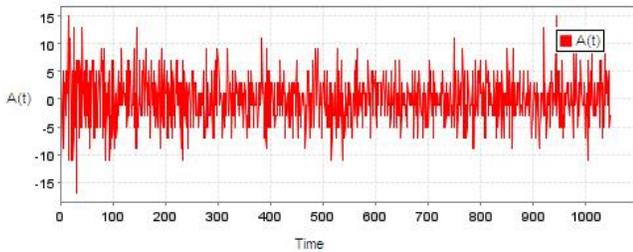


Fig. 3. Typical Time evolution of the Original MG with  $N = 51$ ,  $m = 5$  and  $s = 2$

$m = 3$   $2^3$  actions =  $[+1, +1, -1, -1, +1, -1, +1, +1]$ . The information on past  $m$  wins is stored considering the success of  $-$  group if  $A(t) > 0$  or  $+$  group if  $A(t) < 0$ . Such a past history is mapped on the natural number that results by considering  $-$  as 0 and  $+$  as 1. Such a number is used as position in the sequence of the next action to take: for instance, if  $[-, +, -]$  is the past winning group, we read it as 010 (that is, 2), and accordingly pick the decision in position 2 inside  $[+1, +1, -1, -1, +1, -1, +1, +1]$ , that is  $-1$ .

Each agent actually carries a number  $s \geq 2$  of strategies. During the game the agent evaluates all its strategies according to their success, and hence at each step it decides based on the most successful strategy so far. Figure 3 shows a typical evolution of the game.

One of the most important applications of MG is in the market models: (11) 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 (12), 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.

#### A. MG Logic-Based Approach

MG can be considered a social simulation that aims to reproduce a simplified human scenario. Each (human) agent, in this scenario, must do a choice under the minority global rule. In order to study the system composed by different kinds of players with different behaviours, we here adopt a logic-based approach to build the players. In this way, it is possible to observe particular social behaviours which would otherwise remain hidden in the approximation of the mathematical model.

A more recent paper (2) observes that MG players could be naturally modelled as agents with a full BDI model, and adopts a new adaptive stochastic MG with dynamically evolving strategies in the simulation. We can then apply our simulation framework, with Logic Agents and Coordination Artifacts,

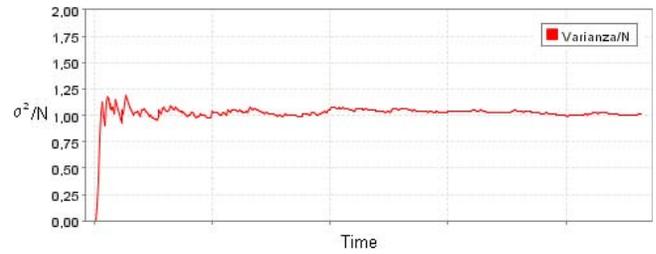


Fig. 4. Variance of the Game with 11 Random Agents

to test the MG from a logic-based point of view, and to experiment with some dynamic tuning strategy.

The next step is to consider players as in an argumentation scenario (3), where agents have the ability to exchange arguments with the purpose to make their own choice or to persuade others to change theirs.

#### B. MG Performance

In order to track the performance of an MG system, the most interesting quantity is *variance*, defined as  $\sigma^2 = [A(t) - \overline{A(t)}]^2$ : 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.

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.

For large values of  $\alpha$ —the number of agents is small with respect to the number of possible histories—the outcomes are seemingly random: the reason for this is that the information that agents observe about the past history is too complex for their limited processing analysis.

When new agents are added, fluctuation decreases and agents perform better by choosing randomly, in this case  $\rho = 1$  and  $\alpha \approx 1/2$ , as visible in the results of our simulation in Figure 4—the game enters into a regime where the losing group is close to  $N/2$ , hence we might say coordination is performing well.

If the number of agents increase further, fluctuations rapidly increase beyond the level of random agents and the game enters into the crowded regime. With a low value of  $\alpha$  the value of  $\sigma^2/N$  is very large: it scales like  $\sigma^2/N \approx \alpha^{-1}$ .

The results of other 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  $\sigma^2/N$  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.

##### 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 (7), a formal description based on Labelled Transition Systems (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 II.

Operating instructions are expressed by the following theory:

```
% pre=Preconditions
% eff=Effects
% act=Action
% per=Perception
firststate(agent(first, [])).
definitions([
  def(first, [], ...),
  %definition of the main control loop
  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)), (
  %read the last value of count
  in_r(count(Y)),
  Z is Y+1,
  %calculate the partial result
  in_r(sum(M)),
  V is M+X,
  out_r(sum(V)),
  %store the new value of count
  out_r(count(Z))
  %this action will be catch
)).
```

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(Result, Turn, NumberOfLoss, MemorySize, last/more)* which is the main tuple of MG coordination.

```
reaction(out_r(count(X)), (
  %check if all agents have already played
  rd_r(numag(Num)),
  X:=Num,
  in_r(totcount(T)),
  Turn is T+1,
  rd_r(game(G)),
  %read the result of the game
  in_r(sum(Result)),
  %reset the sum value
  out_r(sum(0)),
  rd_r(countsession(CS)),
  in_r(count(Y)),
  %reset the count value
  out_r(count(0)),
  %calculate variance
  in_r(qsum(SQ)),
  NSQ is Result*Result+SQ,
  out_r(qsum(NSQ)),
  %calculate mean
  in_r(totsum(R)),
  NewS is R+Result,
  out_r(totsum(NewS)),
  rd_r(numloss(NumberOfLoss)),
  rd_r(mem(MemorySize)),
  % put out the tuple with the result
  out_r(winner(Result, Turn, NumberOfLoss,
  MemorySize, G)),
  out_r(totcount(Turn))
)).
```

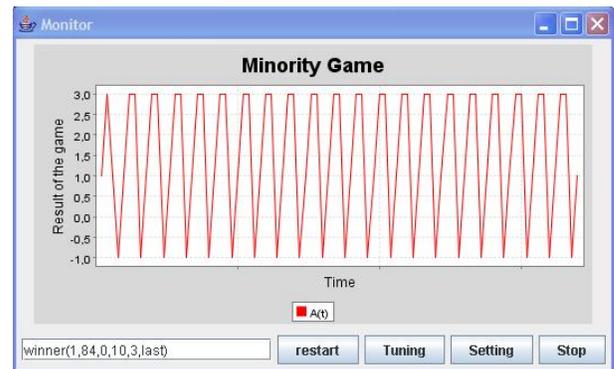


Fig. 5. Interface of the Monitor Agent

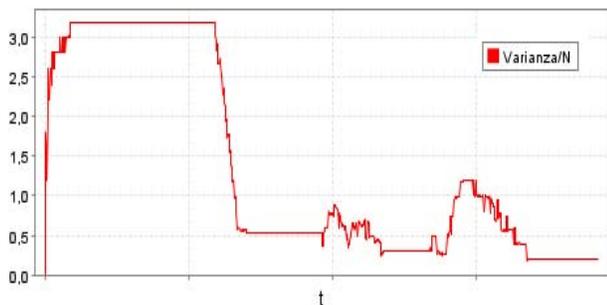


Fig. 6. Variance of the System with Initial Parameters  $N = 5$  and  $m = 3$

The winner tuple contains the result of the game (Result), the number of steps (Turn), 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  $\text{numag}(X)$ , memory size  $\text{mem}(X)$ , and the number of strategies  $\text{numstr}(X)$ . 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  $\text{numloss}(X)$ . 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  $\text{numloss}$  and  $\text{mem}$  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  $\text{numloss}(\text{NumberOfLoss})$  and  $\text{mem}(\text{MemorySize})$ , the rules of coordination react and update the information that will be read by the agents.

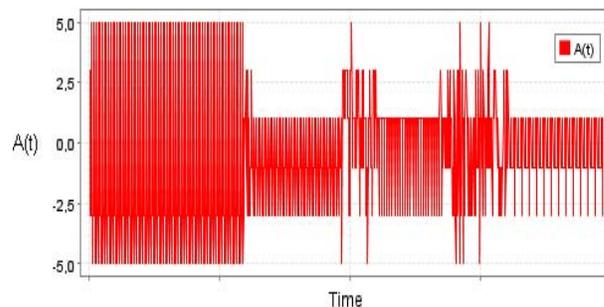


Fig. 7. System Evolution of the Variance in Figure 6

### 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.<sup>2</sup> 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. CONCLUSION

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. Many related agent simulation tools actually exist: as this paper is a starting point, we plan to perform a systematic comparison of their expressiveness and features. In the future, we are interested in constructing an intelligent and adaptive tuning agent with a BDI architecture, substituting the human agent in driving the evolution over time of the system behaviour.

## VI. ACKNOWLEDGEMENTS

The first author of this paper, Enrico Oliva, would like to warmly 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.

<sup>2</sup>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  $m = 5$ .

## 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] 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>
- [3] S. Parsons and P. McBurney, "Argumentation-based communication between agents," in *Communication in Multiagent Systems*, ser. Lecture Notes in Computer Science, M.-P. Huget, Ed., vol. 2650. Springer, 2003, pp. 164–178.
- [4] A. Ricci, M. Viroli, and A. Omicini, "Programming MAS with artifacts," in *Programming Multi-Agent Systems*, ser. LNAI, R. P. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, Eds. Springer, Mar. 2006, vol. 3862, pp. 206–221, 3rd International Workshop (PROMAS 2005), AAMAS 2005, Utrecht, The Netherlands, 26 July 2005. Revised and Invited Papers. [Online]. Available: <http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=3862&page=206>
- [5] A. Ricci, A. Omicini, and E. Denti, "Activity Theory as a framework for MAS coordination," in *Engineering Societies in the Agents World III*, ser. LNCS, P. Petta, R. Tolksdorf, and F. Zambonelli, Eds. Springer-Verlag, Apr. 2003, vol. 2577, pp. 96–110.
- [6] 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.
- [7] 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>
- [8] A. Omicini and E. Denti, "Formal ReSpecT," *Electronic Notes in Theoretical Computer Science*, vol. 48, pp. 179–196, June 2001.
- [9] 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>
- [10] D. Challet and Y.-C. Zhang, "Emergence of cooperation and organization in an evolutionary game," *Physica A: Statistical and Theoretical Physics*, vol. 246, no. 3–4, pp. 407–418, Dec. 1997. [Online]. Available: [http://dx.doi.org/10.1016/S0378-4371\(97\)00419-6](http://dx.doi.org/10.1016/S0378-4371(97)00419-6)
- [11] 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](http://dx.doi.org/10.1016/S0378-4371(99)00446-X)
- [12] H. V. D. Parunak, S. Brueckner, J. Sauter, and R. Savit, "Effort profiles in multi-agent resource allocation," in *1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, C. Castelfranchi and W. L. Johnson, Eds. Bologna, Italy: ACM, 15–19 July 2002, pp. 248–255.
- [13] 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.