Agent-based participatory simulation activities for the emergence of complex social behaviours

Stefano Cacciaguerra, Matteo Roffilli
Department of Computer Science, University of Bologna
Mura Anteo Zamboni 7, 40127 Bologna, Italy
{scacciag, roffilli}@cs.unibo.it

Abstract

Nowadays, social organizations (at macro-level) can be represented as complex self-organizing systems that emerge from the interaction of complicated social behaviours (at micro-level). Modern multi-agent systems can be employed to explore "artificial societies" by reproducing complicated social behaviours. Unfortunately, promoting interactions only among pre-set behavioural models may limit the capability to explore all possible evolution patterns. To tackle this issue, we aim at discovering emergent social behaviours through simulation, allowing human people to participate in the simulation environment, so that the range of possible behaviours is not pre-determined. In order to support this new approach, we propose a system architecture that is able to support an endless session level between a software agent and a human player (called participatory framework). In particular, while network faults or human low reactivity do not allow the human being to control his agent, this system architecture adopts a virtual player mechanism (called ghost player) that takes control of the agent driven by the user. The advanced version of such a ghost player relies on sub-symbolic Machine Learning techniques for mimicking the strategy of the off-line human being.

1 Introduction

Social organizations can be studied at many different levels of abstraction and analysis. Historically, in the analysis of organizational decision-making processes, a common strategy is to reduce a complex social activity to a single constrained optimisation problem that is solved by means of a (macro-level) function. Nowadays, social organizations can be approached as complex self-organizing systems that emerge from the interaction of complicated social behaviours (at micro-level) (Lomi *et al.*, Groningen 2003). This approach makes possible to explore the connection between the micro-level behaviour of individuals and the macro-level patterns that emerge from the interaction of many individuals (Lomi *et al.*, Notre Dame 2003). It is possible to effectively describe these behaviours as the actions of agents into an environment, where the agents are the individuals and the environment is the complex self-organizing system. We consider an agent as a computer system capable of independent actions in order to satisfy its planned objectives. In particular, to describe a complex self-organizing system we need several individuals, while to reproduce it, we need several agents. Along with this consideration, a multi-agent system can be successfully employed, in order to describe self-organizing systems. A multi-agent system is an environment that consists of a number of agents, which interact with one-another. Therefore, it is possible to reproduce social societies into a synthetic environment by creating "artificial societies". To successfully mimic real societies, the multi-agent systems make the agents interact thanks to their ability to cooperate, coordinate, and negotiate (Stone *et al.*, 2000). Nowadays, multi-agent systems are used for educational purposes. For example, a multi-agent system could be used as a computer-based learning environment to teach students of social and economic schools a number of central issues when studying organizational and decision-making processes, and the respective representation of problems (Chen *et al.*, 1993; Colella *et al.*, 1998). These "artificial societies" create a quasi-experimental observation-generation environment where it is possible to conduct tests. Modern multi-agent systems can be employed to explore multiple phenomena from natural to social ones by involving different disciplines: art, biology, chemistry, physics, computer science, earth science, games, mathe-

matics and social sciences.

Well-known modern multi-agent systems are: Swarm (Minar *et al.*, 1996), Repast (Collier *et al.*, 2003), Jas (Sonnessa, 2004), SPADES (Riley, 2003) and Netlogo (Wilensky, 1999). Swarm is a collection of (Objective-C) libraries that promotes the implementation of agent-based models. The Swarm code is Object-Oriented and facilitates the job of simulationists by supporting the incorporation of Swarm objects into their simulation programmes. Its programmes are hierarchical: the top level (called the "observer swarm") creates screen displays and the levels below them. These levels (called the "swarm model") implement the individual agents, schedule their activities, collect information about them and exchange it on the base of an "observer swarm" request. Swarm provides a lot of tutorials that share portions of code in order to facilitate the design of an agent-based model, for example: the management of memory, the maintenance of lists, the scheduling of actions. Jas and Repast are clones of Swarm originated from the translation of Swarm Objective-C sources into Java. In fact, they provide a (Java) library of objects useful to model, schedule, display and collect data from an agent-based simulation. Again, they allow the visualization of the data obtained from the simulation by means of histograms and sequence graphs. Further, they can show snapshots of the evolution of the simulated complex systems in a 2-dimensional (2D) "movie" format. SPADES (System for Parallel Agent Discrete Event Simulation) is a middleware system for agent-based distributed simulation. SPADES allows the simulationist to define the behaviour of agents (as remote processes) and the rules of the world where they live. Differently from the previous ones, it supports the distributed execution of the agents across multiple operating systems, while at the same time it runs distributed simulations regardless of network or system load, adopting a fair policy. NetLogo is a programmable modelling environment that allows the simulationists to give instructions to several passive (i.e. patches) and active (i.e. turtles) agents all operating at the same time. It also implements a classroom participatory simulation tool (called HubNet). HubNet connects networked computers to the Netlogo environment by helping each user control an agent during a simulation.

Typically (apart from Netlogo), a simulationist can interact with these multi-agent systems only during the configuration phase. This means that after a simulationist has chosen the initial conditions of the complex system, he simply becomes a spectator of it (simulated) evolution. If the estimation of the system variables does not critically affect the soundness of the simulative results, the above approach works right. In other cases, alternative approaches are needed to tackle this problem (ill-posed problem). One of them is called "participatory simulation" (Resnick *et al.*, 1998; Wilensky *et al.*, 1999). It provides a way to expand the capability of interactions with these systems at run time. Hence, during a participatory simulation, each single user can play the role of individual system entities and can see how the behaviour of the system as a whole can emerge from the individual behaviours. These synthetic environments promote the cooperation, coordination, and negotiation among the agents controlled by pre-fixed behavioural models (designed by a simulationist) and those driven by humans, all pursuing their own goals. The emergent behaviour of the model and its relation to the participation of humans can make the dynamics of the simulated system clearer. Therefore, these participatory role-playing activities result useful to understand how complex dynamic systems evolve over the time. This approach is very didactic because it promotes a deeper comprehension of the evolution of the simulated complex system. For example, consider a virtual stock exchange, where each player (investor) can play the role of a virtual buyer or seller who engages in the activities of the resulting share exchange dynamics.

The remaining part of this paper is organized as follows. In Section 2, we illustrate the main limits of the modern multi-agent systems, in general, and of agent-based participatory simulation activities, in particular. In Section 3, we present a new alternative approach that overcomes these limitations by adopting a ghost software mechanism and a participatory framework. Section 4 shows some results we obtained with a prototypic implementation of our system. Finally, Section 5 concludes our work with some hints for future developments.

2 Limitations of MASs

One of the main attractions of the above-described simulation environments is the easiness by which it becomes possible to statistically assess the validity of a model. Simulationists can simply explain their idea by writing some lines of code in natural language and then start the simulation. During the evolution they observe the values of some pre-fixed interesting variables and make decisions. Recent works permit to display, in real time, results of the simulation in 2-dimensional computer graphics (Repast; Jas; Netlogo). In our previous work (Cacciaguerra *et al.*, Las Vegas 2004), we improve these capabilities with a 3-dimensional (3D) computer graphics highlighting that this improvement allows to tackle a new class of problems from different

points of view. Recently, the last release of Netlogo environment promotes another 3D visualization confirming our insight (Wilensky, 2005). Nevertheless the multi-agent simulations presented up to now share a common feature: they carry out interactions only between pre-set software behavioural models. While this is extremely important for statistical assessments, we argue that it limits the generation of emerging complex behaviours in any simulation. Along with these considerations, we deem that there are two reasons for the limitation.

The first is related to the simplicity of the model assumed. Every model is defined as a hypothetical-deductive assumption related to some personal knowledge of the simulationist. In fact, the simulationist tries to describe his insight about target problem in a way that a deterministic machine can interpret. This approach is very sensitive to the level of accuracy when modelling the target problem. In fact, it results very difficult to accurately describe all the behaviours included in a model because of intrinsic complexity of social interactions. Then, to leave some degree of freedom, stochastic steps are often introduced causing a loss of sharpness in the analysis. In other cases, it is not possible to fully define a behavioural model because of the not-deterministic physical law behind it. Considering these expert design issues, the analysis are often performed only at standard time intervals: at starting point, at running and finally at asymptote. Obviously changing the starting conditions the simulation shows different behaviours, but asymptotically it reaches the same state-condition or the same periodical fluctuation. This approach guarantees the statistical soundness of the simulation results while it limits the capability to explore all possible evolution patterns.

The second reason is related to the bounded computational power. The current software is not able to handle large amounts of interactions in a timely way because of its engineering. In this case, as well as when facing typical problems related to physical simulations, the time constraint cannot be dealt with in a short period by making the experimentation of complex models impossible. In addition, the analysis of physical systems may result easier than the social one because of the rigid constraints and the proven theories behind it. Hence, it seems to be difficult to implement social simulations that are able to generate new and emergent behaviours. We argue that, by reducing the constraints for the statistical soundness, it is possible to overcome the two limitations (due to both the model accuracy and the time constraint) in an efficient way. To achieve this result, it is necessary for accurate behavioural models to be able to interact together quickly and for a sufficiently long time inside a synthetic environment. In particular, the following is needed:

- A common protocol (i.e. language) to exchange information,
- A high-bandwidth channel for managing communication and
- Large computation power to control behavioural models.

We believe that a cooperative game environment satisfies all the three requirements. A cooperative game is a special kind of game in which many people play together to reach some pre-set goals. The agent-based participatory simulation shows to be one of the best approaches for implementing a cooperative game. It is worth noting that according to our idea the game is the instrument for running a simulation and not the goal of the simulation. One of the main attractions of the transposition of the above problem from a pure software simulation into a cooperative game is that, in the transposed problem, humans can directly interact with the agents inside the synthetic environment by joining the game. Hence, any previous knowledge of the simulation toolkits and programming language is needed, making the simulation methodology widely accessible. Therefore, it becomes possible to use humans as complex and accurate behavioural models for the simulation. In fact, apart from general considerations about Artificial Intelligence (Penrose, 1994), we consider a human being as a very complex social behavioural model. Hence, in defining the objective of the game, we (implicitly) promote the human being to apply his own social model to a pre-fixed task. We argue that this is very similar to the mental process that the simulationist performs when writing a social model for a common simulation toolkit. In addition, humans obviously do not require additional computational power to interact together in a timely way. They also share a priori common language to perform interactions. In fact while a software simulation toolkit offers a hand-made protocol for exchanging information among agents, a game is self-explaining for humans. The 3D visualization (eventually extended with positional 3D audio) is the fastest way to perform interactions among people. In fact, it exploits human natural senses and it is of immediate comprehension. Hence, the cooperative game only demands to create and manage the shared environment to exchange information (that represents the game). In this way, the problem of time constraint is solved too. Further, the cooperative game shows other interesting properties. While solving key problems when running a simulation some questions about experimental design arise.

1. How can we analyse the behaviour of a hand-made behavioural model in such context?
2. Can we assume that providing a large number of participants and a long duration to the simulation will resume the lost statistical soundness?
3. And assuming this is right, how can we find such a large number of people that will play a simulation for an entire week?

3 New approach

In order to tackle these issues, we propose to populate the cooperative game with virtual agents that play together with human players in the same environment. Each virtual agent could be controlled by a software that implements hand-made behavioural models. Further, each human being is represented in the game by his digital avatar that can be fully controlled. Hence, we can think of the avatar as another agent that is driven by the human being instead of a software. In this way, no distinction is made between human beings and software players inside the game context. In line with this assumption, from a game perspective, it is easy to reach hundreds, thousands even up to millions of concurrent players.

Further, this approach offers interesting considerations. First, it becomes very difficult (if not impossible) to distinguish inside the game between software programs and human being-controlled agents using a priory or trivial information. The only way to distinguish them is to analyse the behaviour of each agent for enough time to classify it with some pre-fixed model of knowledge. In other words, a human being should evaluate the strategy (i.e. the pattern of behaviour) of another agent by using his thought of strategy. Along with this consideration, we can think to create an agent that makes this classification extremely difficult. Hypothetically, programming an agent so that no human being can recognize it as a software while playing with it for a long time should be possible. If this *mimic game* is successful, we could safely assert that this software has passed a new version of the Turing test (Turing, 1950). Designing such a software is a hard task and out of the scope of this work. Despite this consideration, promising technologies are emerging.

3.1 Ghost player

Nevertheless, maintaining a high number of human players for a long time is a hard job due to both physiological limits and technical issues. In fact, humans are quickly stressed by intense actions and briefly degrade their mental performances. In addition, depending on the modality of connection to the server (where the synthetic environment is accommodated), the played session can be broken by network faults. In any case, a good participatory simulative environment should not be affected by physiological limits and network faults. To this aim, we propose a preliminary adaptive mechanism (see Figure 1) to avoid these problems penalize the evolution of the complex system. The idea is the following: while a human player is gaming, a *ghost player* is joined to his agent. The ghost player has been previously programmed to run pre-fixed algorithms (a.k.a. behaviourist model) in order to achieve some goals during the game. The ghost player is endowed with an adaptive mechanism able to recognize when the human player is not controlling his agent during the played session. Exploiting this mechanism, when the human player is not able to send moves to his agent, immediately, the ghost player starts to control it avoiding interruptions and the slowdown of the game. When the human player will be able to send moves to his agent again, the ghost player comes, immediately, in the background leaving the control. Hence, this adaptive mechanism is able to keep the game session of a human player alive during the human rest and the network faults.

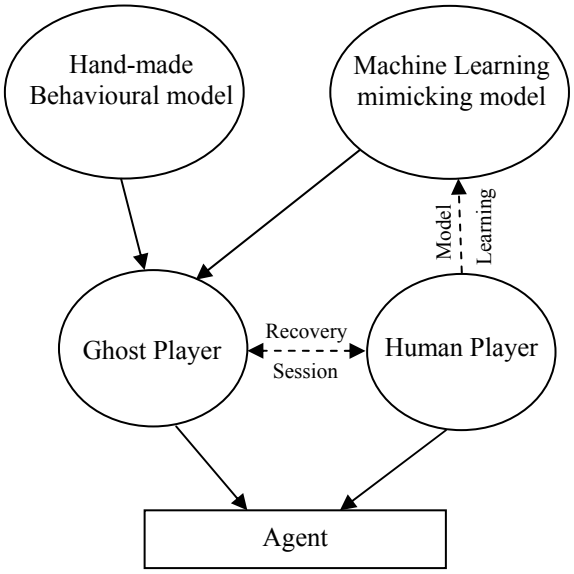

Figure 1: System architecture of our approach

Trying to keep the game alive this mechanism could partially corrupts its consistency. In fact, the ghost player might show a behaviour that is absolutely different from human beings' behaviours. If a lot of ghost players switch on and off intermittently this results in a high degree of unpredictability that potentially transforms the participatory game in a random game where no constructive interactions can be performed. In particular, a human player that is

not able to send moves for a short time, could take the control of his agent again in a situation that has destroyed his long period strategy.

Considering the previous considerations about the mimic game, we propose to replicate the strategy of the human player by providing the ghost player with mimic capabilities. The ghost player analyses the actions of the agent in background and seeks to fit its own pre-fixed behavioural model to the agent's behaviour. In addition, exploiting Machine Learning (Dietterich, 1997; Mitchell, 1997) techniques, it should be able, starting from an imperfect knowledge (i.e. noise-corrupted estimation of system variables) of the environment, to automatically construct a behavioural model resembling that of a human being. A preliminary mimic methodology could be the following: the ghost player knows the legal actions inside the game and it is programmed to consider only a sequence of n moves. Then, it statistically updates the probability of performing the action y knowing that n actions $x_1,\ldots, x_n$ were previously done. We are planning to substitute this simple Bayesian statistics in order to reach more accurate fitting and generalization. We are looking for some candidate methodologies gathered from the field of sub-symbolic Machine Learning. In particular we are evaluating Artificial Neural Networks (Bishop, 1995), ε-Machines (Shalizi *et al.*, 2000), and, especially, Support Vector Machines (Vapnik, 1998), which demonstrated good generalization power in hard tasks (Campanini *et al.*, 2004).

It is worth noting that our purpose is not to create an agent that learns to solve a given problem in an unknown environment and in unsupervised manner. This goal had been deeply analysed in the 90's and a bunch of symbolic algorithms were proposed to tackle it. Our aim is to teach an agent to replicate an existing behaviour starting from noise-corrupted knowledge. Thus, it is a sub-symbolic supervised Machine Learning task.

3.2 Participatory framework

According to the above-proposed approach, we develop a participatory framework that supports the management of the interaction between humans and agents into any participatory simulation. A user can make decisions (and then can act in the synthetic environment) in place of the behavioural model of an agent. More simply, a user can participate in the evolution of the (remote) simulated complex system. Therefore, this framework implements a connection between the user and the agent where a (ISO/OSI) session level is exploited. The user drives a specific agent by means of a client at application level (according to a client-server model architecture that recalls something similar to the Hubnet tool) that communicates over a network connection to the synthetic environment (see Figure 2). In particular, the session level becomes very useful if we are running a participatory simulation over an unreliable network. A typical multi-agent system architecture adopts a fair turn approach to evolve the synthetic environment. This means that each agent must act during each turn (also the NULL move is permitted). Therefore, agents driven by humans must act according to the turn approach too. In addition, the actions coming from a remote human player might slow down the whole serialization of the sequences of fair turns. For this reason, the participation of multiple (remote) users can slow down the evolution of the simulated complex system to unacceptable speed. This may be due to two possible reasons: i) an interruption of the communication and ii) a user high-delayed move. In the case i), an interruption might be two kinds, momentary and permanent, depending on the cause that has generated it. A momentary interruption might be due to network congestion or outages of the communication channel. Instead, a permanent interruption could be due to either a client or a server disconnection. In the case ii), the high-delayed move could be due to the low reactivity of a human player. Further, it could also happens that a human player does not want to sent a move leaving the control of his agent to the ghost player to rest him-self. Hence, the main goal of this framework becomes to maintain the evolution of the simulated complex system over a certain time threshold supporting the human playability. For this reason, if the human player is not able to participate in simulated system under this threshold, the framework guarantees the correctness of evolution within pre-fixed time constraints, by imposing on the slow agent to be driven by the ghost player.

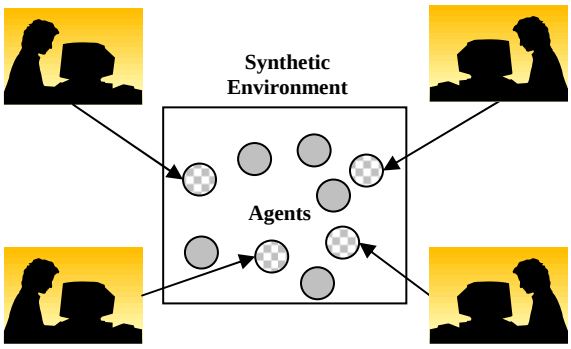

Figure 2: Client-Agent as client-server architecture

In addition, this framework manages the communication by means of a *session recovery* mechanism that allows the user to take the control of his agent again, after a disconnection from the participatory

simulative environment due to a permanent interruption or due to his own free will. In the meantime, the ghost player generates moves for the agent waiting for the re-connection of the human player. In this way, the simulationist can exploit a distributed simulation environment that takes advantage of a session level over the standard ISO/OSI stack (see Figure 3).

3.3 Implementation

We develop a participatory framework that implements a session level over the TCP/IP stack (see the Figure 3). This framework guarantees the correctness of the simulation evolution, and avoids the slowing down of its time performance by accurately managing a session mechanism between the human being and his agent. In particular, the participatory framework consists of a mechanism of session management and a communication management.

The *session management mechanism* guarantees that the human being can participate in the simulation by building his personal session. This means that a human player takes the control of an agent for a simulation run. Therefore, if the human player looses his connection (on purpose or against his own free will) with the agent, his participation in the simulation is guaranteed by the session management mechanism that gives the control to the ghost player. In the near future, if the human player connects to his agent again, the mechanism recovers the previously instantiated session by returning the control to the human being.

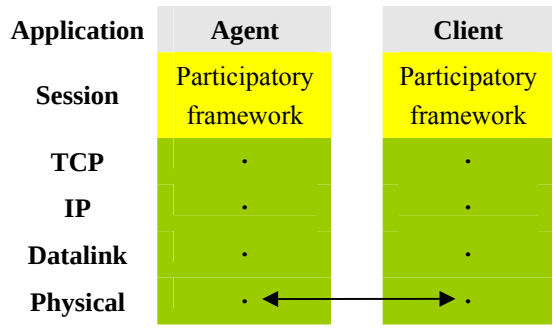

Figure 3: Participatory framework over TCP/IP stack

While the session management mechanism is in charge of managing long period problems due to a disconnection, the *communication management mechanism* handles imminent short period ones such as, human low reactivity, network congestion and outages. The communication management mechanism consists of an action timeout handler and a TCP timeout handler. The first is used by the communication management mechanism to avoid that a low reactivity from the human player slows down the evolution of the complex systems under a certain frequency. In particular, the action timeout handler monitors the responsiveness of the client (on which the human being plays). Hence, the simulationist can set the upper bound (called action timeout) to the responsiveness at a configured time. Obviously, above this bound, the action timeout handler imposes on the ghost player to drive the agent in place of the human being. For sake of completeness, we report that the agent periodically sends session acknowledgements to its client to confirm the responsiveness and to wait for the next move. Instead, the TCP timeout handler is used both at agent-side and client-side. This handler decides if the communication between the client (of the human being) and his agent is closed, based on statistical calculations. These statistics are related to the previous performance according to the agent responsiveness on client-side and human being responsiveness on agent-side. In particular, at agent-side, the TCP timeout handler sets the state of a communication as *broken* when a certain number (i.e. maximum consecutive action timeout configured by the simulationist) of consecutively lost interactions occurs. When the state of the communication is considered as broken, the TCP timeout handler closes it (with a shutdown). Instead, at client-side, the TCP timeout handler sets the state of a communication as broken, only after an amount of time (called TCP timeout) has passed without receiving any session acknowledgement from the agent. After a TCP timeout expiration, the TCP timeout handler at client-side closes the communication (with a shutdown). Finally, the participatory activity could be recovered by exploiting the session management mechanism. Therefore, it become possible to request a new connection to own personal agent exploiting the session management mechanism in active way, by clicking a button, or in passive way, by setting up the configuration file to automatically connect again agent after the expiration of a TCP timeout.

4 Results

In this section, we want to show some results that highlight the effectiveness of our approach. Along with this consideration, we implement a predator-prey artificial ecosystem (a.k.a. pursuit domain) as a model for participatory simulative environment that adopts our participatory framework and a ghost player. This simple biological model is the base for more complex systems. The predator-prey model randomly positions a variable number of preys and predators in a synthetic environment. Obviously, the

preys' goal is to escape, while the predator's is to pursue them. Once a predator reaches a prey, it eats this. Otherwise, if a long period of simulated time passes, the predator dies for starvation. In particular, in these preliminary tests, we focus on the escape trajectory of the prey-agent (green ball of Figures 4-6). The Figures 4, 5 and 6 summarize the video clip related to different runs of the artificial ecosystem (Cacciaguerra *et al.*, December 2004), where the clip frames represent the output of the predator-prey model executed on our prototype. The red balls report the previous positions of the prey-agent. According to this representation, the set of red balls represents the escape trajectory of the prey. In all the simulation runs, the prey-agent is driven by a human player during the initial period (see inset of Figure 4). After this period, the human player does not send the next moves, leaving the control of the prey-agent to the ghost player (in particular, after a maximum consecutive action timeout, the human being was disconnected; see the Figure 7). The pattern of moves related to the human being is similar to a stairway. In Figures 4 and 5, the ghost player adopts his mimic capabilities trying to reproduce a pattern of moves (i.e. a strategy) similar to that of the human being. This does not mean that the ghost player duplicates exactly the learned pattern in a periodical manner or in replicated copies. Instead, the ghost player has learned the way in which the human player drives his agent (to escape) and applies this abstract knowledge to mimic his behaviour (called generalization).

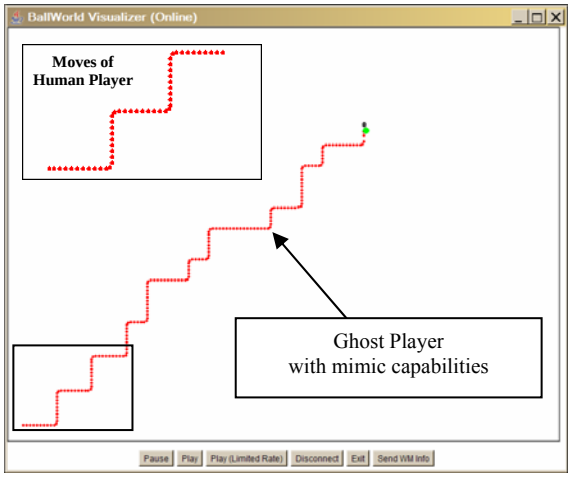

Figure 4: 2D visualization of the escape trajectory of the prey driven by ghost player with mimic capabilities (on Windows XP)

This becomes clear in Figure 5 where the ghost player, stressed to learn the same sequence of moves (see inset in Figure 4), shows a different but similar behaviour as in Figure 4. Obviously, if we look at Figure 6, where the ghost player was running adopting a non-mimic (i.e. random) algorithm, it is clear that the pattern of moves is very dissimilar.

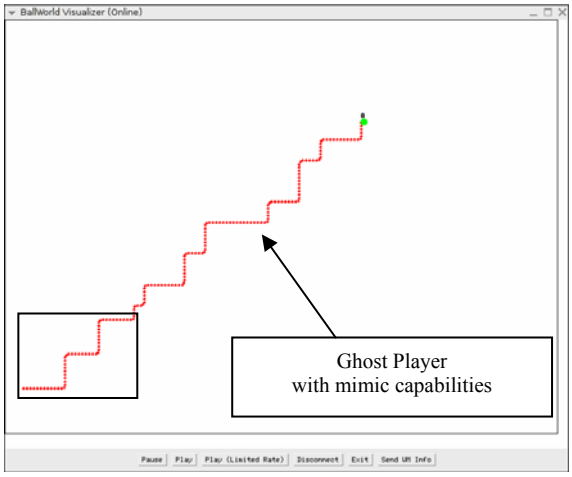

Figure 5: 2D visualization of the escape trajectory of the prey driven by ghost player with mimic capabilities (on Linux)

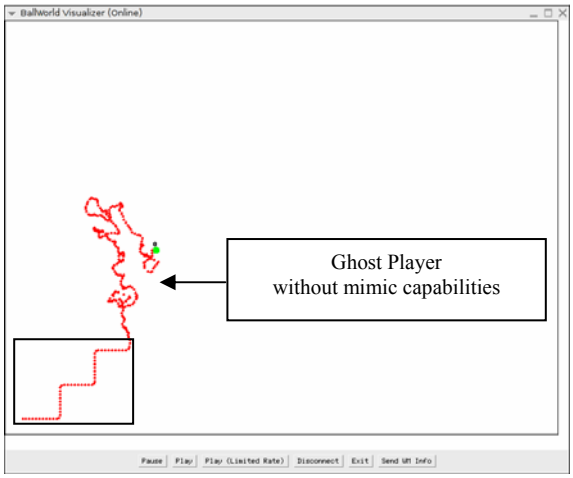

Figure 6: 2D visualization of the escape trajectory of the prey driven by the ghost player without mimic capabilities (on Linux)

Finally, Figure 7 shows the responsiveness of the prey-agent during the previously presented simulative run schemes. This graph illustrates the time spent by the prey-agent to insert it next move into the synthetic environment, showing three phases.

I. From 0 to 2600 simulated time, the agent is driven by the (remote) human player.
II. From 2601 to 5700 simulated time, the agent is driven by the (local) ghost player because the human being is not playing a move under the action timeout,
III. After 5701 simulated time till the end, the agent is driven by the (local) ghost player because a TCP timeout has expired.

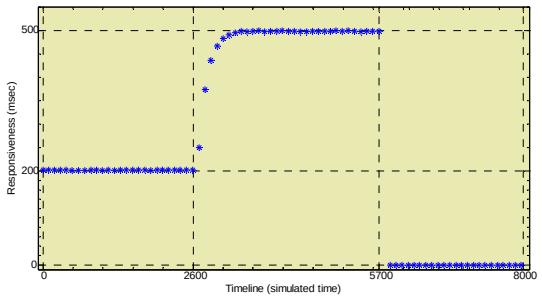

Figure 7: Responsiveness of the prey-agent before and after a user disconnection

5 Conclusions

We have designed and developed a software prototype able to support the execution of agent-based participatory simulative activities to discover the emergence of complex social behaviours. In particular, this prototype supports the participants with an endless session level that allows the human player to disconnect from the synthetic environment while a ghost player takes the control of his agent. A mimicking strategy has been developed to drive the ghost player by means of Machine Learning algorithms. Coupling our framework with smart mimicking capabilities makes it possible to engage agent-based participatory simulation activities with thousands of players dispersed in the world for a long time. The mimicking mechanism is fundamental to maintain a good level of coherence in the game during network faults and human rest. Some results confirm, by means of visual graphs, the efficacy of our approach. In particular, the movie (in mpeg format) of the simulation run reported in Figure 4 highlights the usefulness of our approach. We are designing our software prototype to pass to a new version of the Turing test using some methodologies gathered from the field of Machine Learning as Artificial Neural Networks, ε-Machines, and Support Vector Machines. Further, we are currently planning a massive experimental campaign to study the performance of our participatory framework. We hope this will demonstrate the emergence of complex social behaviours. In order to achieve these results learning behavioural models through imitation seems to be a key point. We wish to conclude this work by mentioning that these trained behavioural models may be very effective in other possible application fields such as digital cinema (Regelous, 2005), edutainment (Wilensky *et al.*, 1999), and multiplayer games (Ferretti *et al.*, 2003) where people can leave and come back.

Acknowledgements

Many thanks to Marco Pracucci for his contribution in the implementation of our idea.

References

C. M. Bishop. Neural Networks for Pattern Recognition. *Clarendon Press*, Oxford, 1995.

S. Cacciaguerra, A. Lomi, M. Roccetti and M. Roffilli. A Wireless Software Architecture for Fast 3D Rendering of Agent-Based Multimedia Simulations on Portable Devices. *In proc. of the IEEE CCNC 2004*, Las Vegas, USA, 2004.

S. Cacciaguerra and M. Roffilli. AISB 2005 movie website. http://wwwfn.csr.unibo.it/projects.html (Dec 2004).

R. Campanini, D. Dongiovanni, E. Iampieri, N. Lanconelli, M. Masotti, G. Palermo, A. Riccardi and M. Roffilli. A novel featureless approach to mass detection in digital mammograms based on Support Vector Machines. *In Phys. Med. Biol.* 49:961-975, 2004.

D. Chen and W. Stroup. General Systems Theory: Toward a conceptual framework for science and technology education for all. *Journal for Science Education and Technology*, 1993.

V. Colella, R. Borovoy and M. Resnick. Participatory Simulations: Using Computational Objects to Learn about Dynamic Systems. *In Proc. of the Computer Human Interface (CHI '98) Conference*, Los Angeles, 1998.

N. Collier, T. Howe and M. North. Onward and Upward: The Transition to Repast 2.0, *Proc. of the First Annual North American Association for Computational Social and Organizational Science Conference*, Pittsburgh, USA, 2003.

T. G. Dietterich. Machine Learning Research: Four Current Directions. *AI Magazine*, 18 (4), 97-136, 1997.

S. Ferretti and S. Cacciaguerra. A Design for Networked Multiplayer Games: an Architectural Proposal. *In proc. of the Euromedia'03*, Plymouth, UK, 2003.

A. Lomi and S. Cacciaguerra. Organizational Decision Chemistry on a Lattice. *In proc. of the 7th Annual Swarm Users/Researchers Conference)*, Notre Dame, USA, 2003.

A. Lomi and S. Cacciaguerra. The Emergence of Routines in an Organizational Decision Chemistry. *In proc. of the 1st European Social Simulation Association Conference (ESSA'03)*, Groningen, Netherland, 2003.

N. Minar, R. Burkhart, C. Langton and M. Askenazi. *The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations*, Santa Fe Institute Working Paper, 1996.

T. Mitchell. Machine Learning, *McGraw Hill*,1997.

R. Penrose. "Shadows of the Mind: A Search for the Missing Science of Consciousness", *Oxford University Press*, 1994.

S. Regelous. Massive Software. http://www.massivesoftware.com (May 2005).

M. Resnick and U. Wilensky. Diving into complexity: Developing probabilistic decentralized thinking through role-playing activities. *Journal of the Learning Sciences*, 1998.

P. Riley. SPADES: System for Parallel Agent Discrete Event Simulation. *AI Magazine*, 24(2):41–42, 2003.

C. R. Shalizi and J. P. Crutchfield. Pattern Discovery and Computational Mechanics. *In proc. of the 17th International Conference on Machine Learning*, Santa Fe Institute, 2000.

M. Sonnessa. The JAS (Java Agent-based Simulation) Library. *In proc. of the 7th Annual Swarm Users/Researchers Conference (SwarmFest'03)*, Notre Dame, Indiana, 2003.

P. Stone and M. Veloso. Multiagent Systems: A survey from a machine learning perspective, *Autonomous Robots, 8(3):345–383,* July 2000.

A. Turing. Computing machinery and intelligence, *Mind*, 1950.

V. Vapnik. Statistical Learning Theory, *John Wiley and Sons*, 1998.

U. Wilensky. NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL, 1999.

U. Wilensky. NetLogo 3-D Preview 1 released. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL, March 2005.

U. Wilensky and W. Stroup. HubNet. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL, 1999.