Introduction (Cargese CNRS Interdisciplinary Seminar, Corsica – April 20th-24th, 2009)

The objective of this seminar was to develop generic and common methods, reusable, in the domain of evolutionary agents in virtual worlds. In physical and extended virtual worlds resources are considered to be limited. Lively, physical or computational systems have to track and optimize activity adapting structure. Agents are activity-aware of self and external resources. Their structure evolves according to fitness functions and objectives. The whole virtual world is constituted of activity-tracking and activity-aware systems, efficiently allocating simulation resources. “Activity” relates to structural and behavioral state changes of systems.

The number of participants has been limited to 20 by invitation only. Participants have been selected according to their competence domain and to the quality of their researches. The multidisciplinary origin of participants leads to the fact that they learned from each other according to their domain expertise.

The seminar was located at the CNRS-Università di Corsica Pasquale Paoli research center of Cargese (www.iesc.univ-corse.fr) in the island of Corsica. During the seminar, theory and application oriented workshops have been organized to compare and integrate contributions. A selected set of invited papers have been retained for inclusion in this book after peer-reviewed process. After a foreword where Bernard P. Zeigler presents a synthesis of the retained papers, we present the seminar program with all oral presentations (http://msdl.cs.mcgill.ca/conferences/Cargese/2009/15_Presentations).

Alexandre MUZY (General Chair)
& David R. C. HILL (Program Chair)

Foreword

This collection of papers represents an unusual cross-section of work in agent-oriented modeling and simulation in which agent concepts are examined from various perspectives ranging from practical to theoretical. The collection is unusual in that it spans contributions from various disciplines that normally do not interact with each other. This is in keeping with the multi-disciplinary nature of the Cargese Workshop, held in April 2009, a gathering that is reminiscent of the early days of cybernetics, before the extreme specialization and fragmentation of today rendered such meetings difficult to hold. Three themes emerge from the articles in this book: agents as artificial emulators of human intelligent activities, agents in modeling and simulation methodology, and activity concepts in complex agent systems. Let's look at the contributions from this point of view:

Agents as artificial emulators of human intelligent activities

- **Jean-Pierre Briot** discusses the software systems design of a virtual park manager that interacts with humans in planning to meet the park's multiple economic, social, and ecological objectives.
- In a first of its kind study, **Guillame Deffuant** demonstrates simple prototypes of agents that may be said to empathize with other agents in the sense that they can "imagine" what others are perceiving and planning.
- From a philosophical perspective, **Franck Varenne** analyzes the prospects for achieving a universal automated modeler agent, one that could develop models and simulations in the social sciences in the manner that only humans can at present.

Agents in modeling and simulation methodology

- **Levent Yilmaz** and **Bradley Mitchell** consider how an ensemble of models, each perhaps only partially representative of an uncertain environment, can support better decision making than a single authoritative one. Agents control simulation of the models and collaborate to allow the ensemble to evolve to better fit the environment.
- In his discussion of the universal automated modeler agent, **Franck Varenne** compares approaches based on system theory with those

based on a multi-level epistemology of modeling and simulation in social science. He finds that when critically examined, much is found to be in common between the two. However, his analysis also reveals insights into why researchers from the respective communities find it difficult to communicate with each other as well as why they will need to do so as the fields converge.

Activity concepts in complex agent systems

- **Xiaolin Hu** introduces discrete event modeling and simulation of pedestrian crowd behavior and shows it can exploit spatial and temporal heterogeneity of agent activity to speed up execution when compared to conventional time-stepped techniques.
- **Coquillard**, **Muzy** and **Wajnberg** first show that inclusion of spatial representations is important in agent-based models in ecology. However, simulations of such models encounter high orders of complexity, especially since stochastic processes are usually involved. Therefore, the authors develop a new approach to discovering and eliminating component processes that are irrelevant to objectives of the study.
- In a remarkable theoretical study, **James Nutaro** employs discrete event formalism to establish an isomorphism between self-clocked cellular automata and hybrid differential automata in order to carry over a theorem concerning the prevalence of limit cycles from the first to the second. Since self-clocked cellular automata are widely employed instances of asynchronous multi-agent systems, this result throws light on the unlikelihood of long term emergent behavior in such systems. Indeed, this study opens up a new area of theory that explores links between activity properties of multi-agent systems and the behaviors that they are capable of exhibiting. One might conjecture for example, that the greater the heterogeneity in space and time of agent activity, the more likely it is that truly “interesting” behavior continues to emerge without limit.

The reader, who looks to these articles as a source of new concepts, and new connections between familiar ones, will not be disappointed.

Bernard P. ZEIGLER (Honorary Chair),
June 2009

Monday April, 20th

18:00	Welcome of participants & schedule

Tuesday April, 21st

8:45	Schedule & planning **Alexandre Muzy** & **David Hill**
9:00-10:30	**Alexandre Muzy**: Activity tracking and awareness: Sketch for a transdisciplinar automatic framework
	Discussions
10:30-11:00	Coffee break
11:00-12:30	**Patrick Coquillard:** Activatability for tractable simulations of NP problems. Application to Ecology
	Olivier Michel: Domain-specific Language for the Modeling of Dynamical Systems with a Dynamical Structure
12:30-14:00	Lunch
14:00-16:30	**Xiaolin Hu:** Exploiting Spatial-temporal Heterogeneity for Agent-based Pedestrian Crowd Simulation
	Levent Yilmaz: Generative Multisimulation: Decision-Support under Uncertainty using Evolutionary Multimodels
	Luc Touraille: DML: a MarkupLanguagefor DEVS Models
16:30-17:00	Coffee break
17:00-19:00	**Hans Vangheluwe:** Exploring “Activity Tracking” a Language Engineering perspective
	Discussions

Wednesday April, 22nd

8:45	Introduction speech by **N. Maupertuis**
9:00-10:30	**Dominique Prunetti:** An interdisciplinary project between Economists and Computer Scientists
	Jean-Pierre Briot: A Computer-Supported Role-Playing Game for Participatory Management of Protected Areas: The SimParc Project + An Abstract Component-based Model for Constructing Operational Models (of Agent Behaviors) for Multi-Agent-based Simulations
10:30-11:00	Coffee break

11:00-12:30	**N. Lameta**: Genesis of behavior in economics: Application to an environmental issue
12:30-14:00	Lunch
14:00-16:30	**Philippe Caillou**: Modelling and analyzing activities on Rungis Wholesale Market: *Some methodological issues on Cognitive Agent Based Simulations*
	Guillaume Deffuant: The Empathon: an agent aiming at mimicking empathy
16:30-17:00	Coffee break
17:00-19:00	**Franck Varenne**: Framework for M&S with Agents (FMSA) in regard to Agent-based Simulations in Social Sciences: Emulation and Simulation
	Discussions

THURSDAY APRIL, 23rd

8:45	Schedule & planning: **David Hill**
9:30-10:00	**Bernard P. Zeigler** Synthesis (part I)
10:00-10:30	Coffee break
10:30-12:30	**Bernard P. Zeigler** Synthesis (part II)
	Discussions
12:30-14:00	Lunch @ Institute
14:00-17:00	Coffee break
17:00-20:00	Working groups

FRIDAY APRIL, 24th

8:45	Schedule & planning: **David Hill**
9:00-10:00	Working groups
10:00-10:30	Coffee break
10:30-12:30	Working groups and discussion of new articles
12:30-14:00	Lunch @ Institute
14:00-18:00	Local visit of the Calanche di Piana
18:00-19:00	Plenary synthesis

Design of an Artificial Decision Maker for a Human-based Social Simulation – Experience of the SimParc Project

Jean-Pierre BRIOT[1,2], Alessandro SORDONI[1], Eurico VASCONCELOS[2], Vinícius SEBBA PATTO[1], Diana ADAMATTI[3], Marta IRVING[4], Gustave MELO[4] & Carlos LUCENA[2]

Abstract – *This paper addresses an ongoing experience in the design of an artificial agent taking decisions in a social simulation (more precisely, a role playing game) populated by human agents and by artificial agents. At first, we will present our current research context, an ongoing research project aimed at computer-based support for participatory management of protected areas (and more specifically national parks) in order to promote biodiversity conservation and social inclusion. Our applicative objective is to help various stakeholders (e.g., environmentalist NGOs, communities, tourism operators, public agencies) to collectively understand conflict dynamics for natural resources management and to explore negotiation management strategies for the management of parks, one of the key issues linked to biodiversity conservation. Our approach combines techniques such as: distributed role playing games (serious games), geographical information systems, support for negotiation between players,*

1. Laboratoire d'Informatique de Paris 6, Université Pierre et Marie Curie – CNRS, Paris, France. Corresponding author: Jean-Pierre.Briot@lip6.fr.

2. Computer Science Department, Pontifícia Universidade Católica, Rio de Janeiro, RJ, Brazil.

3. Facultade de Tecnologia (FTEC), Caxias do Sul, RS, Brazil.

4. EICOS Postgrad Program, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brazil.

and insertion of various types of artificial agents (virtual players, decision making agents, assistant agents). In this paper, we will focus on the design of the decision making agent architecture for the park manager agent, the rationales for his decision and how he takes into account the preferences/votes from the players of the game and may justify/explain his decisions.

Keywords – Social simulation, serious games, participatory management, decision making, multi-agent, negotiation, argumentation, explanation.

Introduction

In this paper, we discuss the issue for an artificial agent to take decisions in a social simulation. The type of social simulation that we refer to here is a social simulation populated by humans. More precisely, we consider a role playing game ("serious game") where humans play some role and discuss, negotiate and take decisions about a common domain, in our case environment management decisions.

We are now in the process of inserting artificial agents into the human-based social simulation [Briot *et al.*, 2008]. One of the ideas is to possibly replace some of the human players by artificial players (artificial agents). The social simulation will therefore become hybrid, with human and artificial agents. A first motivation is to address the possible absence of sufficient number of human players for a game session [Adamatti *et al.*, 2007]. But this will also allow more systematic experiments about specific configurations of players profiles, because of artificial players' objective, deterministic and reproducible behaviors.

More precisely, we are considering three types of artificial agents: artificial park manager, artificial players and assistant agents. In this paper we focus on the design of the artificial park manager agent. Its objective is to take decision based on its own analysis of the situation and on the proposals by the players. The agent is also able to explain its decision based on its chain of argumentation.

In this paper, after introducing the SimParc project, its role playing game and its computer support, we describe the park manager agent objectives, architecture and implementation.

1. The SimParc Project

1.1. Project Motivation

A significant challenge involved in biodiversity management is the management of protected areas (e.g., national parks), which usually undergo various pressures on resources, use and access, which results in many conflicts. This makes the issue of conflict resolution a key issue for the participatory management of protected areas. Methodologies intending to facilitate this process are being addressed *via* bottom-up approaches that emphasize the role of local actors. Examples of social actors involved in these conflicts are: park managers, local communities at the border area, tourism operators, public agencies and NGOs. Examples of inherent conflicts connected with biodiversity protection in the area are: irregular occupation, inadequate tourism exploration, water pollution, environmental degradation and illegal use of natural resources.

Our SimParc project focuses on participatory parks management. (The origin of the name SimParc stands in French for "Simulation Participative de Parcs") [Briot *et al.*, 2007]. It is based on the observation of several case studies in Brazil. However, we chose not to reproduce exactly a real case, in order to leave the door open for broader game possibilities [Irving *et al.*, 2007]. Our project aim is to help various stakeholders at collectively understand conflicts and negotiate strategies for handling them.

1.2. Approach

Our initial inspiration is the companion modeling (ComMod) approach about participatory methods to support negotiation and decision-making for participatory management of renewable resources [Barreteau *et al.*, 2003]. Their pioneer method, called MAS/RPG, consists in coupling multi-agent simulation (MAS) of the environment resources with a role-playing game (RPG) [Barreteau, 2003]. Hence, stakeholders may understand and explore the consequences of their decisions. The RPG acts like a "social laboratory", because players of the game can try many possibilities, without real consequences.

Recent works proposed further integration of role-playing into simulation, and the insertion of artificial agents, as players or as assistants. Participatory simulation and its incarnation, the Simulación framework [Guyot and Honiden, 2006], focused on a distributed support for role-playing and negotiation between human players. All interactions are recorded for further analysis (thus opening the way to automated acquisition of behavioral models) and assistant agents are provided to assist and suggest strategies to the players. The Games and Multi-Agent-based Simulation (GMABS) methodology focused on the integration of the game cycle with the simulation cycle [Adamatti *et al.*, 2007]. It also innovated in the possible replacement of human players by artificial players.

1.3. Game Objectives

Current SimParc game has an epistemic objective: to help each participant discover and understand the various factors, conflicts and the importance of dialogue for a more effective management of parks. Note that this game is not (or at least not yet) aimed at decision support (*i.e.*, we do not expect the resulting decisions to be directly applied to a specific park).

The game is based on a negotiation process that takes place within the park council. This council, of a consultative nature, includes representatives of various stakeholders (e.g., community, tourism operator, environmentalist, nongovernmental association, water public agency…). The actual game focuses on a discussion within the council about the "zoning" of the park, *i.e.* the decision about a desired level of conservation (and therefore, use) for every sub-area (also named "landscape unit") of the park. We consider nine pre-defined potential levels (that we will consider as types) of conservation/use, from more restricted to more flexible use of natural resources, as defined by the (Brazilian) law. Examples are: *Intangible*, the most conservative use, *Primitive* and *Recuperation*.

The game considers a certain number of players' roles, each one representing a certain stakeholder. Depending on its profile and the elements of concerns in each of the landscape units (e.g., tourism spot, people, endangered species…), each player will try to influence the decision about the type of conservation for each landscape unit. It is clear that conflicts of interest will quickly emerge, leading to various strategies of negotiation (e.g., coalition formation, trading mutual support for respective objectives, etc).

A special role in the game is the park manager. He is a participant of the game, but as an arbiter and decision maker, and not as a direct player. He observes the negotiation taking place between players and takes the final decision about the types of conservation for each landscape unit. His decision is based on the legal framework, on the negotiation process between the players, and on his personal profile (e.g., more conservationist or more open to social concerns) [Irving, 2006]. He may also have to explain his decision, on players demand. Players and the park manager roles may be played by humans or by artificial agents.

1.4. Game Cycle

The game is structured along six steps, as illustrated in Figure 1. At the beginning (step 1), each participant is associated to a role. Then, an initial scenario is presented to each player, including the setting of the landscape units, the possible types of use and the general objective associated to his role. Then (step 2), each player decides a first proposal of types of use for each landscape unit, based on his/her understanding of the objective of his/her role and on the initial setting. Once all players have done so, each player's proposal is made public. In step 3, players start to interact and to negotiate on their proposals. This step is, in our opinion, the most important one, where players collectively build their knowledge by means of an argumentation process. In step 4, they revise their proposals and commit themselves to a final proposal for each landscape unit. In step 5, the park manager makes the final decision, considering the negotiation process, the final proposals and also his personal profile (e.g., more con-

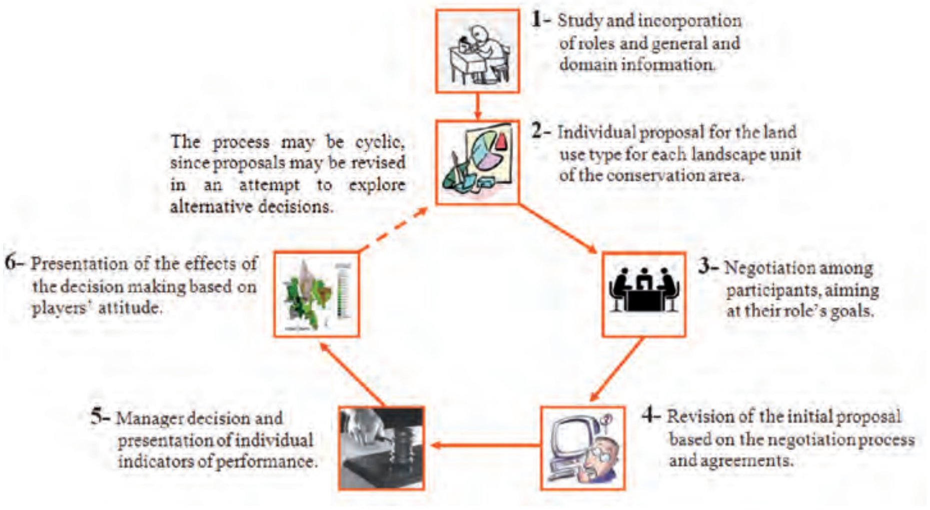

1- Study and incorporation of roles and general and domain information.
2- Individual proposal for the land use type for each landscape unit of the conservation area.
The process may be cyclic, since proposals may be revised in an attempt to explore alternative decisions.
3- Negotiation among participants, aiming at their role's goals.
6- Presentation of the effects of the decision making based on players' attitude.
5- Manager decision and presentation of individual indicators of performance.
4- Revision of the initial proposal based on the negotiation process and agreements.

Figure 1. The six steps of SimParc game.

servationist or more sensitive to social issues). Each player can then consult various indicators of his/her performance (e.g., closeness to his initial objective, degree of consensus, etc.). He can also ask for an explanation about the park manager decision rationales. The last step (step 6) "closes" the epistemic cycle by considering the possible effects of the decision. In the current game, the players provide a simple feedback on the decision by indicating their level of acceptance of the decision. A new negotiation cycle may then start, thus creating a kind of learning cycle. The main objectives are indeed for participants: to understand the various factors and perspectives involved and how they are interrelated; to negotiate; to try to reach a group consensus; and to understand cause-effect relations based on the decisions.

1.5. Towards Evaluating the Viability of Decisions

As described in previous section, the last step of the game "closes" the cycle by considering the possible effects of the decision. In the current game, players provide a simple feedback on the decision by selecting their level of acceptance of the decision. In a future project, we would like to introduce some technical evaluation of the quality and viability of the decision (e.g., considering the survival of an endangered species). Therefore, we plan to identify cases of usage conflicts (e.g., between tourism and conservation of an endemic species) and model the dynamics of the system (in an individual-based/multi-agent model or/and in an aggregated model). We would then like to explore the use of viability theory to evaluate the viability of the decision. Note that in our project current stage, we are concerned with credibility and not yet with realism because our objective is epistemic and not about producing an (hypothetical) optimal decision.

2. Computer Support for Role Playing Games

Our current prototype benefited from our previous experiences (game sessions and a first prototype) and has been based on a detailed design process. Based on the system requirements, we adopted Web-based technologies (more precisely J2E and JSF) that support the distributed and interactive character of the game as well as an easy deployment. Figure 2 shows the general architecture and communication structure of SimParc prototype version 2. In this second prototype, distributed users (the players and the park manager) interact with the system mediated internally by communication broker agents (CBA). The function of a CBA is to abstract the fact that each role may be played by a human or by an artificial agent. A CBA also translates user messages in http format into multi-agent KQML format and vice versa. For each human player, there is also an assistant agent offering assistance during the game session.

During the negotiation phase (see in Figure 3 an example of the corresponding user interface [Vasconcelos *et al.*, 2009]), players (human or artificial) negotiate among themselves to try to reach an agreement about the type of use for each landscape unit (sub-area) of the park. A Geographical Information System (GIS) offers to users different layers of information (such as flora, fauna and land characteristics) about the park geographical area. All the information exchanged during negotiation phase namely users' logs, game configurations, game results and general management information are recorded and read from a PostgreSql database.

The game and the supporting prototype have been tested through two game sessions by expert players (including a professional park manager) in January 2009 (see Figure 4).

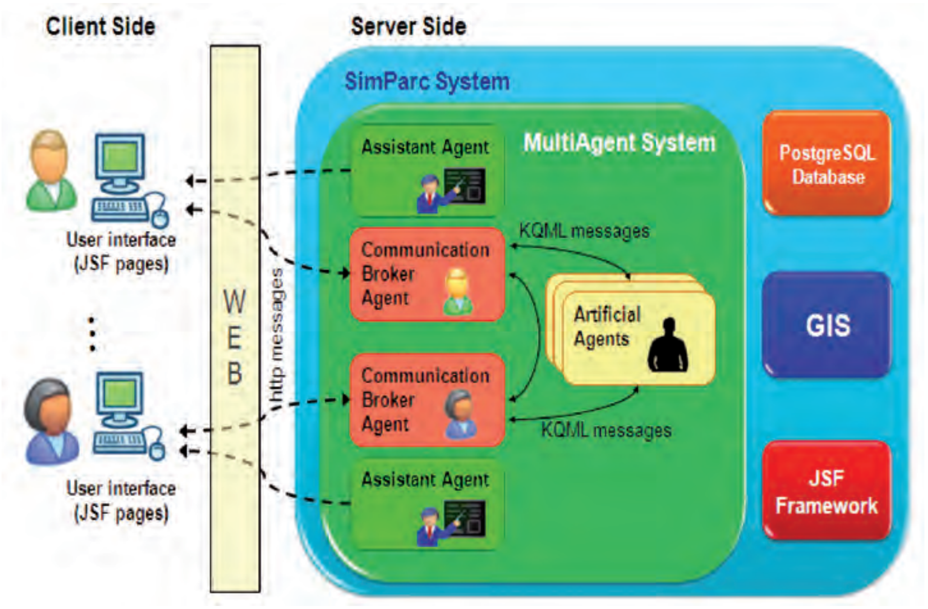
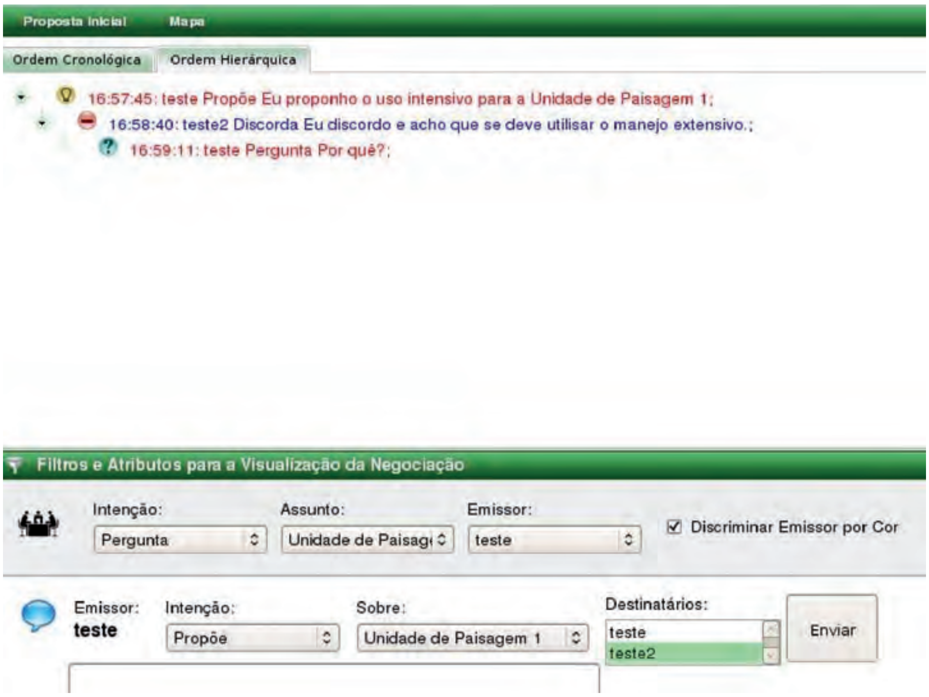


Figure 2. SimParc version 2 general architecture.



16:57:45: teste Propõe Eu proponho o uso intensivo para a Unidade de Paisagem 1;
16:58:40: teste2 Discorda Eu discordo e acho que se deve utilisar o manejo extensivo.;
16:59:11: teste Pergunta Por quê?;

Intenção: Pergunta
Assunto: Unidade de Paisag
Emissor: teste
☑ Discriminar Emissor por Cor

Emissor: teste
Intenção: Propõe
Sobre: Unidade de Paisagem 1
Destinatários: teste, teste2

Figure 3. Example of SimParc negotiation graphical user interface

Figure 4. SimParc session (January 2009).

3. Inserting Artificial Agents in the SimParc Game

We are currently inserting artificial agents into the prototype. We consider three types of artificial agents: the park manager, the artificial players and the assistants.

The park manager acts as an arbitrator in the game, making a final decision for types of conservation for each landscape unit and explains its decision to all players. He may be played by a human or by an artificial agent. We have implemented a prototype of an artificial park manager, based on 2 steps: (1) internal/individual decision by the park manager, based on some argumentation model; (2) merging of the decision by the manager with the votes by the players, based on decision theory (social choice). Traces of argumentation may be used for explaining the rationale of the decision. The artificial park manager architecture is detailed in next section.

Concerning artificial players, we refer to previous experience about virtual players in the ViP-JogoMan system [Adamatti *et al.*, 2007]. The idea is to possibly replace some of the human players by artificial players (artificial agents). The two main motivations are: (1) the possible absence of sufficient number of human players for a game session and (2) the need for testing in a systematic way specific configurations of players' profiles. The artificial players will be developed along park manager existing architecture. We plan to use its argumentation capabilities to generate the negotiation process. In parallel, we also explore using automated analysis of recorded traces of interaction between human players in order to infer models of artificial players. In some previous work [Guyot and Honiden, 2006], genetic programming had been used as a technique to infer interaction models, but we will also intend to use alternative induction and machine learning techniques, e.g., inductive logic programming.

The last type of artificial agent is an intelligent assistant agent. This agent is designed to assist a player by performing two main tasks: (1) to help participants in playing the game, e.g.: the assistant agent tells the player when he should make decisions; what are the phases of the game; what should be done in each phase; etc.; (2) to help participants during the negotiations. For this second task, we would like to avoid intrusive support, which may interfere in his decision making cognitive process. We have selected some objectives, e.g., to identify other players' roles with similar or dissimilar goals, which may help the human player to find possible coalitions or conflicts. An initial prototype implementation of an assistant agent has been implemented.

Further details about SimParc artificial players and assistant agents may be found in [Briot *et al.*, 2008] and in future publications. Some advanced interface has also been designed for human players dialogue and negotiation support and is detailed in [Vasconcelos *et al.*, 2009]. Now we will detail the rationale and architecture of our automated park manager who makes the final decision.

4. Park Manager Artificial Agent

In this section, we propose an agent architecture to implement park manager cognitive decision rationale. As we summarized before, our decision model is based on two mechanisms. These mechanisms could be viewed as modules of decision subprocesses. We believe that complex decision making is achievable by sequential organization of these modules. Before proceeding to the description of our agent architecture, we present some more detailed motivation for it.

4.1. Objectives

Participatory management aims to emphasize the role of local actors in managing protected areas. However, the park manager is the ultimate arbiter of all policy on devolved matters. He acts like an expert who decides on validity of collective concerted management policies. Moreover, he is not a completely fair and objective arbiter: he still brings his personal opinions and preferences in the debate. Therefore, we aim to develop an artificial agent modeling the following behaviors.

Personal preferential profile; the park manager decision-making process is supposed to be influenced by its sensibility to natural park stakes and conflicts. In decision theory terms, we can affirm that park manager's *preferential profile* could be intended as a *preference relation* over conservation policies. One of the key issues is to understand that we cannot define a strict bijection between *preferential profile* and *preference relation*. Agent's preference relation is partially dependent on natural park resources and realities. Moreover, this relation is not likely to be an order or a preorder. Hence, our agent must be able to generate its preference relation according with its preferential profile. We distinguish two preferential profiles:

i. *Preservationist,* aims to preserve ecosystems and the natural environment.
ii. *Socio-conservationist,* generally accepts the notion of *sustainable yield* – that man can harvest some forest or animal products from a natural environment on a regular basis without compromising the long-health of the ecosystem.

Taking into account stakeholders' decisions; a participatory decision-making leader seeks to involve stakeholders in the process, rather than taking autocratic decisions. However, the question of how much influence stakeholders are given may vary on manager's preferences and beliefs. Hence, our objective is to model the whole spectrum of participation, from autocratic decisions to fully democratic ones. To do so, we want the park manager agent to generate a preference preorder over conservation policies. This is because it should be able to calculate the distance between any two conservation policies. This way, we can merge stakeholders' preference preorders with manager's one to establish one participatory final decision. Autocratic/democratic manager attitude will be modeled by an additional parameter during the merge process.

Expert decision; the park manager's final decision must consider legal constraints related to environmental management; otherwise, non-viable decisions would be presented to the players, thus invalidating game's learning objectives. These constraints are directly injected in the cognitive process of the agent. Hence, the agent will determine a dynamic preference preorder over allowed levels of conservation (according to its preferential profile).

Explaining final decision; in order to favor the learning cycle, the park manager agent must be able to explain its final decision to the players. We can consider that the players could eventually argue about its decision; the agent should then defend its purposes using some kind of argumentative reasoning. Even if such cases will be explored in future work, it is our concern to conceive a cognitive architecture which provides a good basis for managing these situations.

4.2. Architecture Overview

Let us now present an architecture overview of the park manager agent. As depicted in Figure 5, agent's architecture is structured in two phases. We believe that sequential decision-making mechanisms can model complex cognitive behaviors along with enhanced explanation capabilities.

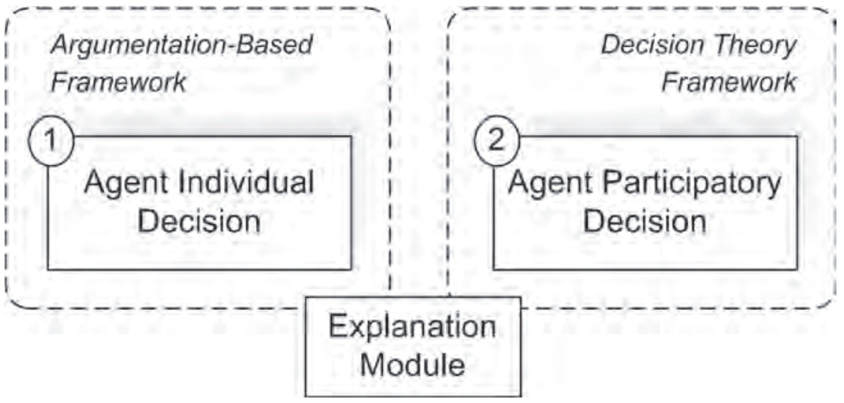

Argumentation-Based Framework
Decision Theory Framework
1
Agent Individual Decision
2
Agent Participatory Decision
Explanation Module

Figure 5. Park manager agent 2-steps decision process.

The first decision step concerns agent's individual decision-making process: the agent deliberates about the types of conservation for each landscape unit. Broadly speaking, park manager agent builds its preference preorder over allowed levels of conservation. An argumentation-based framework is implemented to support the decision making.

The next step of our approach consists of taking account of players' preferences. The result of the execution is the modified park manager decision, called agent *participatory* decision, according to stakeholder's preferences.

4.2.1. Agent Individual Decision

Recently, argumentation has been gaining increasing attention in the multi-agent community. Autonomous and social agents need to deliberate under complex preference policies, related to the environment in which they evolve. Generally, social interactions bring new information to the agents. Hence, preference policies need to be dynamic in order to take account of newly acquired knowledge. Dung's work [Dung, 1995] proposes formal proof that argumentation systems can handle epistemic reasoning under open-world assumptions, usually modeled by non-monotonic logics. Argumentation thus becomes an established approach for reasoning with inconsistent knowledge, based on the construction and the interaction between arguments. Recently, some research has considered argumentation systems capabilities to model *practical reasoning*, aimed at reasoning about *what to do* [Hulstijn and Torre, 2003; Amgoud and Kaci, 2004; Rahwan and

Amgoud, 2006]. It is worth noticing that argumentation can be used to select arguments that support available desires and intentions. Consistent knowledge can generate conflicting desires. An agent should evaluate pros and cons before pursuing any desire. Indeed, argumentative deliberation provides a mean for choosing or discarding a desire as an intention.

We could argue that open-world assumptions do not hold in our context. Agent's knowledge base isn't updated during execution, since it is not directly exposed to social interactions. Knowledge base and inference rules consistency-checking methods are, therefore, not necessary. However, one key aspect here is to conceive an agent capable of explaining its policy making choices; our concern is to create favorable conditions for an effective and, thus closed, learning cycle. We believe that argumentation "tracking" represents an effective choice for accurate explanations. Conflicts between arguments are reported to the players, following agent's reasoning cycle, thus enhancing user comprehension.

From this starting position, we have developed an artificial agent on the basis of Rahwan and Amgoud's work [Rahwan and Amgoud, 2006]. The key idea is to use argumentation system to select the desires the agent is going to pursue: natural park stakes and dynamics are considered in order to define objectives for which to aim. Hence, decision-making process applies to actions, *i.e.* levels of conservation, which *best* satisfy selected objectives. In order to deal with arguments and knowledge representation, we use first-order logic. Various inference rules were formulated with the objective of providing various types of reasoning capability.

For example, a simple rule for generating desires from beliefs, *i.e.* natural park stakes, is:

$$Fire \rightarrow Avoid\_Fires,\ 4$$

where *Fire* (fire danger in the park) is a belief in the agent's knowledge base and *Avoid_Fires* is the desire that is generated from the belief. The value *4* represents the intensity of the generated desire.

Examples of rules for selecting actions, *i.e.* level of conservation, from desires are:

$$Primitive \rightarrow Avoid\_Fires,\ 0.4$$

$$Intangible \rightarrow Avoid\_Fires,\ 0.8$$

where *Primitive*, *Intangible* represent levels of conservation and values *0.4, 0.8* represent their utilities in order to satisfy the corresponding desire.

4.2.2. Agent Participatory Decision

Despite participatory ideals, a whole spectrum of park managers, from autocratic to fully democratic ones, can be measured, depending on how more participatory and democratic decision-making is operationalized. We propose a method, fitted into the social-choice framework, in which participatory attitude is a model parameter.

In a real case scenario, a decision-maker would examine each stakeholder's preferences in order to reach the compromise that best reflects its participatory attitude. Our idea is to represent this behavior by weighting each player's vote according to the manager's point of view.

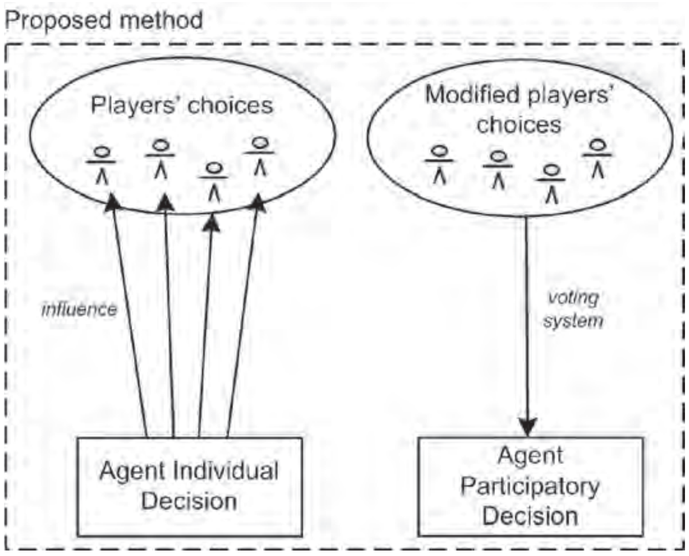

Proposed method
Players' choices
Modified players' choices
influence
voting system
Agent Individual Decision
Agent Participatory Decision

Figure 6. Park manager agent Participatory Decision.

This concept is illustrated in Figure 6. The process is structured in two phases. Firstly, the manager agent injects its own preferences into players' choices by means of an influence function describing agent's participatory attitude. Stronger influence translates into more autocratic managers. Secondly, modified players' choices are synthesized, using an aggregation function, *i.e.* Condorcet voting method. The result of the execution will be the agent participatory decision.

Example. Let the following be the players' choices, where > is a preference relation (a > b means "a is preferred to b") and *A*={*Intangible, Primitive, Extensive*} the candidates' set. The players' choices are converted into numeric vectors specifying the candidates' rank (each column corresponds to a candidate) for each vote:

player_1: Intangible > Primitive > Extensive, $\mathbf{v}_1 = (3, 2, 1)$

player_2: Extensive > Primitive > Intangible, $\mathbf{v}_2 = (1, 2, 3)$

player_3: Primitive > Extensive > Intangible, $\mathbf{v}_3 = (1, 3, 2)$

Let the Manager individual decision be:

manager_individual: Extensive > Primitive > Intangible, $\mathbf{v}_M = (1, 2, 3)$

Let the following be the influence function:

$$f(x,y) = \begin{cases} x & if\ x = y \\ x * 1/|x-y| & otherwise \end{cases}$$

Modified player's vectors will be:

$$\mathbf{mv}_1 = \langle f(\mathbf{v}_1(1), \mathbf{v}_M(1)), \langle f(\mathbf{v}_1(2), \mathbf{v}_M(2)), \langle f(\mathbf{v}_1(3), \mathbf{v}_M(3)) \rangle = (1.5, 2, 0.5)$$

$$\mathbf{mv}_2 = (1, 2, 3)$$

$$\mathbf{mv}_3 = (1, 3, 2)$$

In order to find manager participatory decision, we apply the Choquet integral C_μ [Choquet, 1953] choosing a symmetric capacity measure $\mu(S) = |S|^2 / |A|^2$, where A is the candidates set.

$C_\mu(Intangible) = 1.05$, $C_\mu(Primitive) = 2.12$, $C_\mu(Extensive) = 1.27$

The result of the execution will then be:

manager_participatory: Primitive > Extensive > Intangible.

Further details about architecture formal background and implementation are reported in [Sordoni, 2008].

4.3. Examples of results

The presented manager agent architecture and its first implementation were tested over different scenarios. Tests of the park manager agent were conducted offline in laboratory and have been validated by team experts. We are currently completing the integration of the park manager agent into the prototype. We will then soon conduct new session tests with human players and the park manager agent.

We report hereafter an example of explanation for manager's decision over a landscape unit. Let the manager individual decision be the following:

Manager_individual: Intangible > Recuperation

Arguments for *Intangible* are:

Endangered_Species & Tropical_Forest → Maximal_Protection

Intangible → Maximal_Protection

Arguments for *Recuperation* are:

Fire & Agriculture_Activities → Recover_deteriorated_zone

Recuperation → Recover_deteriorated_zone

4.4. Implementation framework

The architecture presented in this paper has been implemented in Jason multi-agent platform [Bordini and Hubner, 2007]. Besides interpreting the original AgentSpeak(L) language, thus disposing of logic programming capabilities, Jason also features extensibility by user-defined internal actions, written in Java. Hence, it has been possible to easily implement aggregation methods.

Conclusion

In this paper, we have presented the SimParc project, a role-playing serious game aimed at computer-based support for participatory management of protected areas. The lack of human resources implied in RPG gaming process acts as a constraint to the fulfillment of pedagogical and epistemic objectives. In order to guarantee an effective learning cycle, the park manager role must be played by a domain expert. Required expertise obviously narrows game's autonomy and limits its context of application. Our solution to this problem is to insert artificial agents into the game. In this paper, we focused on the architecture to implement the park manager cognitive decision rationale. It can justify its behavior and generate a participatory decision. Our argumentation system is based on [Amgoud and Kaci, 2004;

Rahwan and Amgoud, 2006]: conflicts between arguments can be reported thus enhancing user comprehension. Moreover, we presented a decision theory framework responsible for generating a participatory decision. To the best of our knowledge and belief, this issue has not yet been addressed in the literature. The game and the supporting prototype have already been tested through game sessions by expert players (including a professional park manager) in January 2009. The final integration of the complete SimParc prototype (with artificial agents) is under completion and we will soon start to test it. Besides the project specific objectives, we also plan to study the possible generality of our prototype for other types of human based social simulations. In the current architecture of the artificial park manager, only static information about the park and about the votes of the players are considered. We are considering exploring how to introduce dynamicity in the decision model, taking into account the dynamics of negotiation among the players (the evolution of player's decisions during negotiation).

Acknowledgments

We thank the other current members of the project: Isabelle Alvarez and Wei Wei for their current participation; and also: Ivan Bursztyn, Paul Guyot, Altair Sancho and Davis Sansolo for their past participation. This research has been funded by: the ARCUS Program (French Ministry of Foreign Affairs, Région Ile-de-France and Brazil) and the MCT/CNPq/CT-INFO "Grandes Desafios" Program (Project No 550865/2007-1). Some additional individual support has been provided by Alßan (Europe), CAPES and CNPq (Brazil) fellowship programs.

References

[Adamatti *et al.*, 2007] – Adamatti D. F., Sichman J. S. and Coelho H., Virtual Players: From Manual to Semi-Autonomous RPG. *In: Proceedings of the International Modeling and Simulation Multiconference (IMSM'07)*, Buenos Aires (Argentina), The Society for Modeling & Simulation International (SCS), February 2007, pp. 159-164.

[Amgoud and Kaci, 2004] – Amgoud L. and Kaci S., On the generation of bipolar goals in argumentation-based negotiation. *In:* Rahwan I., Moraitis P. and Reed C. (eds.), *Argumentation in Multi-Agent Systems: Proceedings of the First International Workshop (ArgMAS'04): Expanded and Invited Contributions,* volume 3366 of *Lecture Notes in Computer Science,* Berlin (Germany), Springer Verlag, 2005, pp. 192-207.

[Barreteau, 2003] – Barreteau O., The Joint Use of Role-Playing Games and Models Regarding Negotiation Processes: Characterization of Associations. *Journal of Artificial Societies and Social Simulation*, Vol. 6, No. 2, 2003, http://jasss.soc.surrey.ac.uk/6/2/3.html.

[Barreteau *et al.*, 2003] – Barreteau O *et al.*, Our Companion Modelling Approach. *Journal of Artificial Societies and Social Simulation*, Vol. 6, No. 1, 2003, http://jasss.soc.surrey.ac.uk/6/2/1.html.

[Bordini and Hubner, 2007] – Bordini R. and Hubner J., JASON: A Java-based AgentSpeak Interpreter used with Saci for Multi-Agent Distribution over the Net, available at http://jason.sourceforge.net/.

[Briot *et al.*, 2007] – Briot J.-P., Guyot P. and Irving M., Participatory Simulation for Collective Management of Protected Areas for Biodiversity Conservation and Social Inclusion. *In: Proceedings of the International Modeling and Simulation Multiconference (IMSM'07)*, Buenos Aires (Argentina), The Society for Modeling & Simulation International (SCS), February 2007, pp. 183-188.

[Briot *et al.*, 2008] – Briot J.-P., Vasconcelos E., Adamatti D., Sebba V., Irving M., Barbosa S., Furtado V. & Lucena C., A Computer-based Support for Participatory Management of Protected Areas: The SimParc Project. *In: Proceedings of XXVIIIth Congress of Computation Brazilian Society (CSBC'08) – Seminário Integrado de Software e Hardware "Grandes Desafios"*, Belem (PA, Brazil), July 2008, pp. 1-15.

[Choquet, 1953] – Choquet G., Theory of capacities. *Journal of Fourier Institute*, No. 5, 1953, pp. 131-295.

[Dung, 1995] – Dung P. M., On the acceptability of arguments and its fundamental role in nonmonotonic and n-person games. *Artificial Intelligence*, Vol. 77, No. 2, 1995, pp. 321-357.

[Guyot and Honiden, 2006] – Guyot P. and Honiden S., Agent-Based Participatory Simulations: Merging Multi-Agent Systems and Role-Playing Games. *Journal of Artificial Societies and Social Simulation*, Vol. 9, No. 4, 2006, http://jasss.soc.surrey.ac.uk/9/4/8.html.

[Hulstijn and Torre, 2003] – Hulstijn J. and Torre L. van der, Combining goal generation and planning in an argumentation framework. *In:* Heskes T., Lucas P., Vuurpijl L. and Wiegerinck W. (eds.), *Proceedings of the 15th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2003)*, Katholieke Universiteit Nijmegen, October 2003, pp. 155-162.

[Irving, 2006] – Irving M. (ed.), *Áreas Protegidas e Inclusão Social: Construindo Novos Significados*, Rio de Janeiro, Aquarius, 2006.

[Irving *et al.*, 2007] – Irving M. A., Briot J.-P., Bursztyn I., Guyot P., Melo G., Sancho A., Sansolo D., Sebba Patto V. and Vasconcelos E., Simparc: Computer Supported Methodological Approach for Constructing Democratic Governance in National Parks Management. *In: Proceedings of IUCN II Congreso Latinoamericano de Parques*

Nacionales y Otras Areas Protegidas, Bariloche (Argentina), September–October 2007.

[Rahwan and Amgoud, 2006] – Rahwan I. and Amgoud L., An Argumentation-based Approach for Practical Reasoning. *In:* Weiss G. and Stone P. (eds.), *5th International Joint Conference on Autonomous Agents & Multi Agent Systems, AAMAS'2006,* Hakodate (Japan) - New York (USA), ACM Press, 2006, pp. 347-354.

[Sordoni, 2008] – Sordoni A., Master's Degree Thesis. http://www-poleia.lip6.fr/~sordonia/master-thesis.pdf.

[Vasconcelos *et al.*, 2009] – Vasconcelos E., Briot J.-P., Irving M., Barbosa S. and Furtado V., A User Interface to Support Dialogue and Negotiation in Participatory Simulations. *In:* David N. and Sichman J. S. (eds), Multi-Agent-Based Simulation IX – International Workshop MABS 2008, Estoril, Portugal, May 12-13, 2008, Revised and Invited Papers, volume 5269 of *Lecture Notes in Artificial Intelligence*, Berlin (Germany), Springer Verlag, 2009, pp. 127-140.

The Empathon: Towards a Computational Agent Mimicking Empathy

Guillaume DEFFUANT[1]

Abstract – *This paper investigates a basic model simulating empathy. The model is abstracted from recent advances in neurobiology, such as the discovery of "mirror neurons" or of neurons coding for both visual recognition and connected action plans. In a very simplified computational environment, we define agents which associate action plans with visual stimuli, and are also able to "feel" such action plans when they observe another agent in a similar situation. We propose a first set of experiments involving only a pair of agents, with different action abilities.*

Keywords – Empathy, mirror neurons, computational agent.

Introduction

The simplest definition of empathy is "the ability to put oneself in the place of someone else". There are a lot of debates about more precise definitions. Several authors distinguish empathy from sympathy, considering that the latter is reduced to emotion contagion. Different theories (theory of the simulation, theory of the theory) back different hypotheses on the mechanism.

Whatever the precise mechanism or definition, a lot of considerable thinkers stressed the importance of empathy to understand specifically human

1. Cemagref, LISC, 24 rue des Liondards, 63172 Aubière, France (guillaume.deffuant@cemagref.fr).

behaviours and social dynamics. For instance, Max Scheler [1923] and Edith Stein [1917] see empathy at the root of religious experience [see also Deffuant, 1998]. Charles Darwin has stressed that empathy was the evolutionary basis of all social behaviours [see also Hoffman, 2000]. Adam Smith [1759] and Jean-Pierre Dupuy [1992] suggest that empathy can be intimately related with the autobiographic consciousness of human beings.

In the last decades of the 20th century, the scientific community showed a renewed interest for empathy. In psychology, a large panel of results is now available, often involving comparative tests between children of various ages, with or without deficiencies [Hill *et al.*, 2004; Baron-Cohen *et al.*, 1985]. A large variety of results is also available for different species of primates. Different theoretical models, in particular the "theory of the theory", and the "theory of the simulation" are in competition to interpret the results [Davies, 1994]. In economy and sociology, the concept of convention is related on multiple levels of empathy between the members of a community. Game theory and computer simulation approaches begin to address the problem of multi-level empathy. Several philosophers and thinkers turned back to A. Smith or J.-J. Rousseau, and revisited their ideas, mixing them with recent developments on complex systems.

Spectacular advances in neurobiology, which gave empathy its first neural evidences, played a major role in renewing the scientific interest for empathy. Indeed, the "mirror neurons", discovered in the 90ies, showed that putting oneself in the place of someone else is deeply wired in our neural system, and partly unconscious [see for instance, Gallese *et al.*, 1996; Gallese and Goldman, 1998; Decety *et al.*, 1997; Rizzolati *et al.*, 2001, among the numerous publications on the subject]. Previously, empathy tended to be considered only as the effect of an imagination effort, thus depending on conscious and high level cognitive functions. Moreover, the work on mirror neurons is deeply linked with a new view of perception / action neural system. This new view reminds the concept of "affordance" introduced by Gibson, and is grounded on the discovery of neurons coding jointly for shape and action planning. In other words, the perception of objects is intimately related with the actions they suggest.

As far as we know, agent based social simulation did not attempt to integrate these advances yet. Researches on trust and reputation [Conte *et al.*, 2008; Di Tosto *et al.*, 2007] often include some aspects related to empathy in the agents, but they do not ground them on perception. Other agent based models use agents which are inspired by game theory, including the prediction of the others next moves. But again, these models do not take into account the recent knowledge we have on the relations between perception and empathy. Hence defining agents with an abstract mechanism mimicking mirror neurons is a completely new line of research.

Yet, agent based approaches could provide very interesting complementary evidences about the role of the low level empathic abilities on social dynamics. Indeed, such models require to formalize clearly how empathy works, and its interactions with other cognitive functions, and then to test these hypotheses in a virtual laboratory. Ideally such tests would bring new arguments in the debate about the role of empathy in social dynamics as well as in some aspects of consciousness.

We propose a very first attempt in this direction, with artificial agents called "empathons", reminding Rosenblatt's "perceptron" [1958], which was seen by its author as a first artificial perceptive system. Similarly, we aim at defining first artificial empathic systems. We tried to keep the model as simple as possible, but not more than this. One needs to define the perception and action plans connected with objects first, and then to define the recognition of such situations in other individuals.

We first define the individual perception of empathon, and then the model using the observed state of others. Then we describe some simulation results, and finally, we propose a discussion.

1. Individual level: agents seeking targets and avoiding obstacles

1.1. *Overview of the setting*

We chose a quite common 2D environment with mobile agents which must reach targets and avoid obstacles. To simplify, all the objects (agent body, obstacles, targets) are circular. But the agents have a front (where are located the perceptive captors) and a back. The position of objects and agents is defined by continuous values. Agents can move only forward, thus, when they want to go in one direction, they need to turn first to aim the direction, and then they move. The movements of agents are confined inside a given 2D area because of a set of obstacles displayed in its boundary (see Fig. 1). The obstacles are fixed and their number is constant over the simulation. There is also a set of targets, which are located at random. A target disappears when it is reached by an empathon, and a new one is created with a random position (not too close to the obstacles or the empathons).

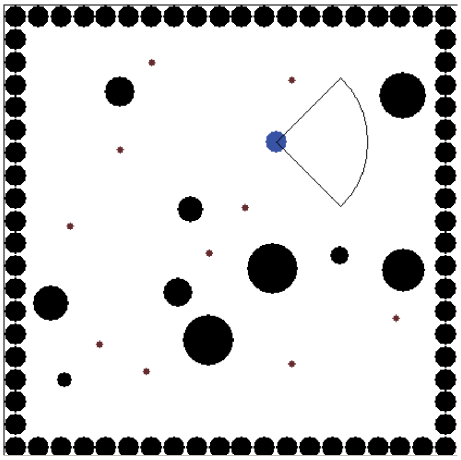


Figure 1. Example of setting with 10 obstacles and 10 targets. The empathon is in blue, and its local perceptive area is shown. Obstacles are in black, targets in brown.

1.2. Surrounding percept and action plans

At this level, the global algorithm ruling the agent is very simple:

- While no target in sight, explore and avoid obstacles (state: 'exploring').
- When a target is in sight, reach it, while avoiding obstacles (state: 'reaching target' or 'avoiding obstacle').

We now describe the model of percept and temporary memory, and then the actions plans.

Surrounding percept.

We suppose that an agent has a perception of its environment which is wider than its strict captor area. This corresponds to the idea that we have some perception of what is in our back, when we know the place where we are standing. To model this, we define a percept of the agent. This percept concerns a zone around the agent of size 3 times its vision length. This local zone is divided into a grid, and the agent keeps a temporary memory of whether it has watched this part recently or not (see Fig. 2). Moreover, it keeps the memory of the local position of targets, even if the agent is watching in another direction.

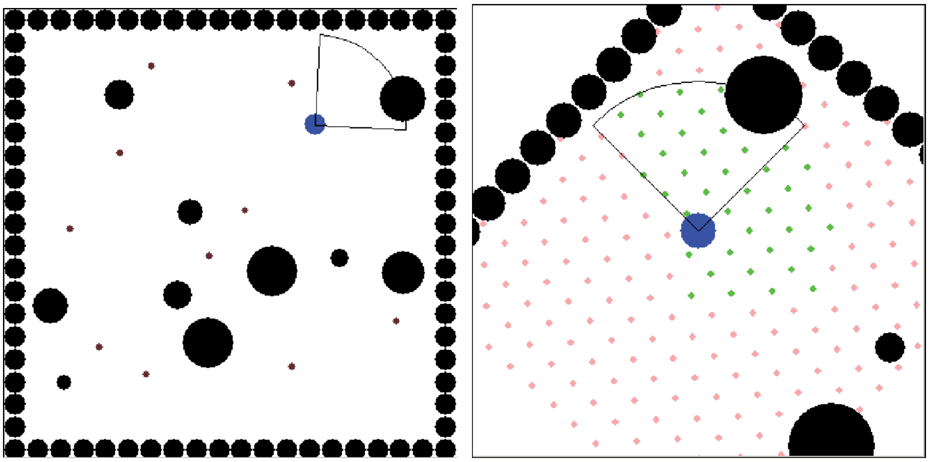


Figure 2. On the left, the global setting. On the right, the representation of the agent's percept. The agent is located in the center (local coordinates). The pink dots are in zones that the agent has not seen for long, whereas the green dots are zones which have been recently seen. We note that the obstacles appear in the percept, because we suppose the agent has a good knowledge of this permanent environment. However, the target (brown dot) does not appear because this part of the space has not been explored yet (pink dots).

In practice, the table of the dots associated with the surrounding percept is updated as follows:

The time step after the visit of the site, its associated value is 1,

Then, at each time step, it is decreased of a value $1/k$, k being the time span (in number of steps) of the temporary memory. After k steps, the agent has forgotten it has seen this zone.

Exploration and obstacle avoidance.

While it perceives no target, the agent implements an action plan to explore its environment and find a target. The principle is simple: at each time step:

1. the agent chooses at random a zone where it has not been for long (pink dot) as a direction of exploration.
2. The agent goes in this direction for one time step, with the maximum move allowed.
3. If an obstacle in on the path, then it stops before reaching the obstacle.

Target reaching and obstacle avoidance.

The agent records the location of the targets in its surrounding percept, while it has not eaten them. When several targets are perceived simultaneously, it chooses the closest as a goal to reach.

The obstacle avoidance process is necessarily more complicated than in the exploration. Indeed, in the exploration mode, the agent forgets about its initial direction as soon as it encounters an obstacle. When trying to reach a target, the agent must keep this goal. Moreover, it must determine a trajectory that goes around the obstacle and allows it to reach the target. We adopted a simple approach where the agent follows a tangent trajectory to the obstacle while the target cannot be reached directly. We skip the details, because this process is not at the core of our research questions.

If the avoidance action plan fails (the agent is blocked by the obstacles) then the target is abandoned. The agent puts itself in exploration mode or chooses another target if it has one in its surrounding percept.

Interactions with other agents.

At this level, others agents are simply considered as obstacles, except that their location is not fixed, and an agent tries to avoid only the agents which are visible. Therefore, agents can occasionally overlap each other. In this case, we simulate a shock, and each agent is pulled back a little, in the direction defined by both centers of the agents.

1.3. Results

As expected, a single agent manages to reach a set of targets rather easily, when all targets are accessible (not confined in a position where the obstacles prevent the agent to go). It alternates periods of exploration and of target reaching. When there are two agents in the same setting, each one behaves almost as if it was alone, except when it has to avoid the other, which is not so frequent.

2. First level of empathy: using the observed state of the other

2.1. Principles of the model

As a first attempt, we consider only two agents, and we suppose that one agent (called A_0) remains as in the first setting (considering other agents as simple obstacles), while the other agent (called A_1) has some access to the state of A_0 and uses this knowledge to get more targets. The main difference between A_1 and A_0 agents is that A_1 agents evaluate if there is another agent aiming at the same target, which is better placed. In this case they give up this target and choose another or get back to the state 'exploring'. The main hypotheses we make are as follows:

- Agent A_1 has a perfect access to the state (exploring, reaching target, avoiding an obstacle…) of agent A_0, if A_0 is directly visible. Indeed, we suppose that this state is directly visible (like emotion expressions in humans for instance). Moreover, we suppose that the direction of move of A_0 is also directly accessible to A_1, as well as its vision length and width.
- When A_1 perceives A_0 is aiming at a target (states 'reaching' or 'avoiding'), A_1 puts itself virtually in the place of A_0, tries to identify A_0's target. If A_1 is itself aiming at a target, it checks if A_0 is better placed for this target, and if it is the case, it drops for another one or for the state 'exploring'. Indeed, there is no point spending time pursuing it, because finally the other will get it.

2.2. Implementation

We describe now in more details the functions ruling agent A_1. When it has no direct sight on A_0, its behavior is the same as A_0. Hence we focus only on the case where A_0 is in A_1 direct vision zone.

The empathized agent, a "virtual" agent to anticipate the moves of the other

When A_1 perceives A_0, it creates an agent replicating A_0 (with the same position and state), and uses it to anticipate the moves of A_0. We call this "virtual" agent the empathized of A_0 by A_1, and note it $E_1(A_0)$. $E_1(A_0)$ replicates the characteristics of A_0 which are accessible to A_1. It may also include some a priori hypotheses on A_0. In this case, the main hypothesis is that A_0 is of A_0 type.

Setting the empathized agent's target

When A_0 is in the state 'reaching', A_1 gets the location of A_0's target by testing if each target currently present in its surrounding percept is located on the direction pointed by A_0. The target satisfying this test becomes E_1 (A_0)'s target. If none satisfies the test, it means that A_0's target is not present in A_0's surrounding percept, and A_1 considers that the target of $E_1(A_0)$ is "unknown".

When A_0 is in the state 'avoiding', A1 first considers the obstacles closer to A_0 than twice its radius, and selects the one for which the trajectory of A_0 is tangent. Then it tests if each of the targets present in its surrounding percept, is behind the obstacle for A_0. If several satisfy the test, it selects the closest as $E_1(A_0)$'s target. If none, then the target of E_1 (A_0) is set unknown.

Dropping the current target?

Once A_1 determined $E_1(A_0)$'s target, it decides whether it is worth keeping its own target or not. The decision depends on the respective states of the agents:

- If A_0's state is 'reaching' and A_1's state is 'avoiding' then if the target is less than twice closer to A_0, A_1 drops it, if the targets are the same. Indeed, it can be expected that avoiding an obstacle takes longer than the movement in right line.
- If both are in state 'reaching' then:
    - If the targets are the same, and if the distance of A_0 to target is smaller than the distance of A_1 to target, then A_1 drops the target
    - If the targets are different, then if A_0 is on the way to the A_1 target, then A_1 drops it.

When A_1 drops a target, it is ignored in its surrounding percept for a given number of times steps. If after this time, for some reason, A_0 has not caught the target, and if A_1 has it in its direct vision zone, it will consider it again.

Examples

In the following figures, we illustrate the rules applied by A_1 to drop or not its target. Fig. 3 shows a case where both agents share the same target and are in 'reaching state'. Fig. 4 and 5 show a case where the agents share the same target, and where A_1 drops the target and switches to the state 'exploring'. On Fig. 4, both agents are in reaching state, whereas on Fig. 5 A_1 is in 'avoiding' state.

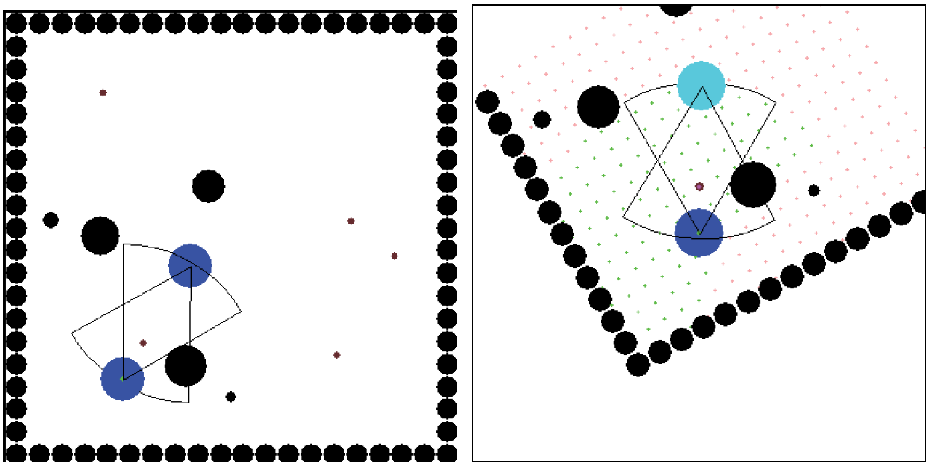


Figure 3. Example of configuration where agent A_1 (blue disc with a green dot at its centre) keeps its target. On the left, the global setting, and on the right the surrounding percept of A_1. The empathized agent $E_1(A_0)$ is represented in cyan color. Both agents share the same target, but it is closer to A_1, hence A_1 keeps its target.

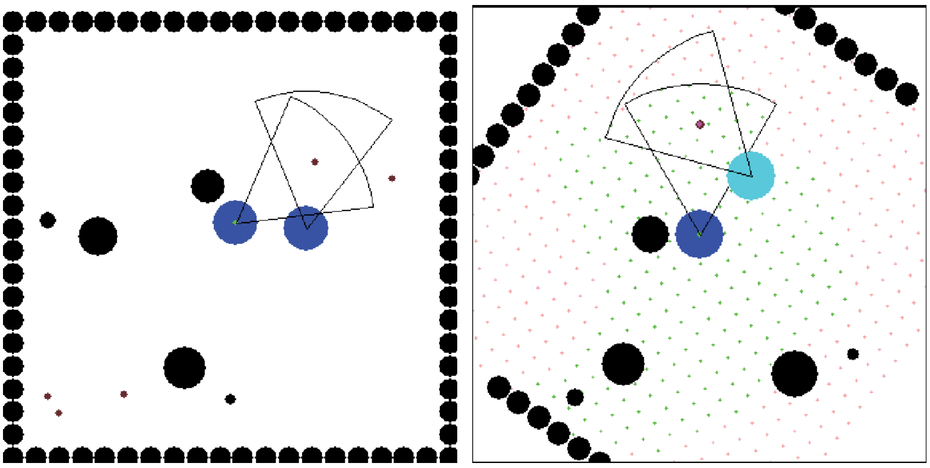


Figure 4. Example of configuration where agent A_1 (blue disc with a green dot at its centre) drops its target. On the left, the global setting, and on the right the surrounding percept of A_1. The empathized agent $E_1(A_0)$ is represented in cyan color. Both agents share the same target, but it is closer to A_0, hence A_1 drops its target. It will switch to the state 'exploring'.

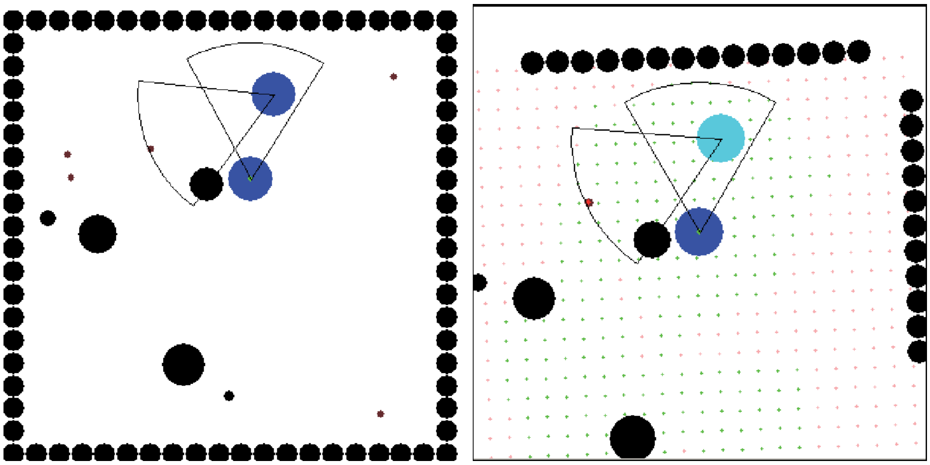


Figure 5. Example of configuration where agent A_1 (blue disc with a green dot at its centre) drops its target. On the left, the global setting, and on the right the surrounding percept of A_1. The empathized agent $E_1(A_0)$ is represented in cyan color. Both agents share the same target, it is closer to A_1, but A_1 is currently avoiding an obstacle. It judges that A_0 has better chances to get the target and thus drops it. It will switch to the state 'exploring'.

2.3. Simulation results

The graph of Fig. 6 shows the averaged number of caught targets after 1000 time steps for agent A_1 and A_0, over 50 replicas. Remember that the targets are regenerated automatically when caught by an agent: each time one target is caught another is added at random in the setting. Hence, the number of available targets is constant. Moreover, to simplify, we made this first test without any obstacle.

We note that in this configuration (vision length 0.5 and moving speed 0.005), agent A_1 performs better than A_0. The advantage is of a few percents, but it is robust. It is due mainly to the cases where A_0 follows A_1 on targets that A_0 has no chances to get the target. A_0 looses then a precious time. A_1 avoids this mistake with its strategies for dropping bad targets.

Note that this advantage is not always significant. For instance, when the vision length is smaller (0.3) and the moving is faster (0.025), the advantage of A_1 is not significant (see Fig. 7). Indeed, when the vision length is small, the cases of sharing targets are less frequent, and the path possibly avoided by dropping a target is smaller. This path is also relatively smaller when the

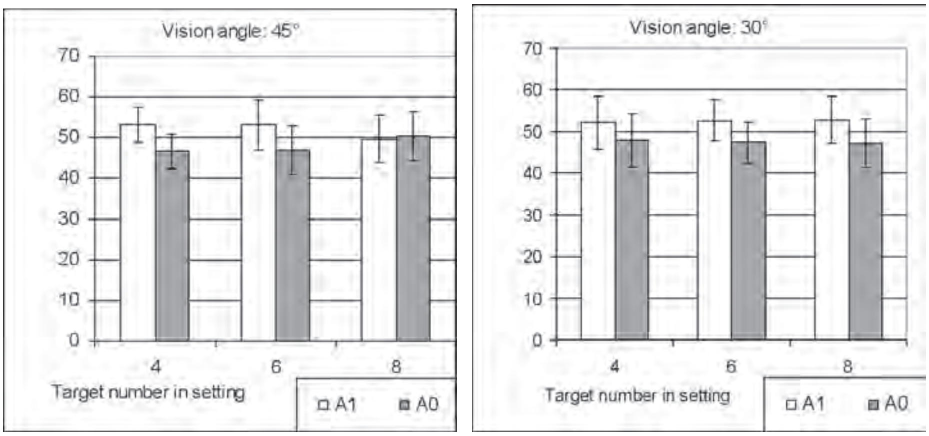


Figure 6. Percentage of targets caught by A_1 (level 1) and A_0 (level 0), during 1000 time steps, for different numbers of targets in the setting (the targets are constantly regenerated, and located at random in the setting, when caught by the agents). On the left graph, the vision angle is 45°, on the right graph, it is 30°. The vision length is 0.5 (for a square setting of size 1), the maximum speed is 0.005, and the number of obstacles in the setting is 0. The confidence intervals are standard deviations computed on 50 replicas.

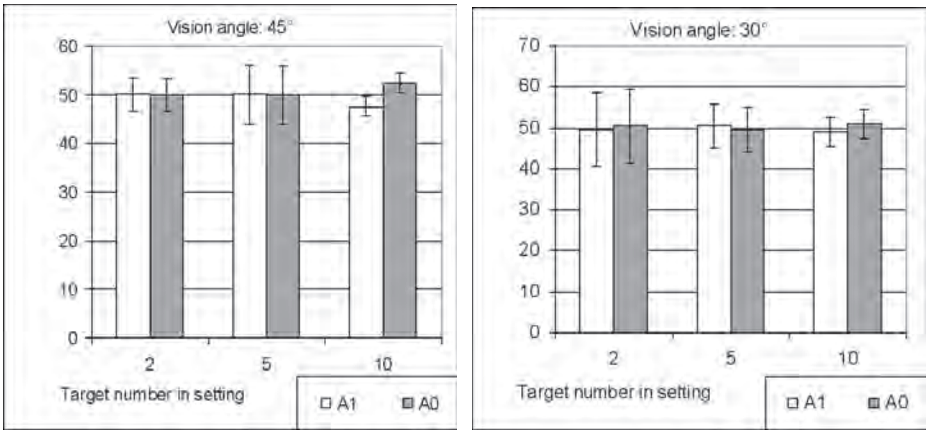


Figure 7. Percentage of targets caught by A_1 (level 1) and A_0 (level 0), during 1000 time steps, for different numbers of targets in the setting (the targets are constantly regenerated, and located at random in the setting, when caught by the agents). On the left graph, the vision angle is 45°, on the right graph, it is 30°. The vision length is 0.3 (for a square setting of size 1), the maximum speed is 0.025, and the number of obstacles in the setting is 0. The confidence intervals are standard deviations computed on 50 replicas.

moving step is higher. Therefore, the advantage to avoid useless moving for a bad target is smaller.

These results suggest that the capacity to exploit other's state is not always a decisive advantage in such games where one must reach targets before the other.

3. Discussion

The reported experiment should be considered as very preliminary. Indeed, in our view, its main interest is to illustrate an approach which opens a large set of possible experiments on abstract hypotheses on empathy. One could wonder to which extent this approach brings something different from existing ones. We first briefly discuss these differences with a few potentially competing approaches, and then we draw some perspectives.

Taking into account the likely move of an opponent is a typical problem of the theory of games [Von Neumann and Morgenstern, 1944; Nash, 1950; Maynard-Smith, 1982], which is common to our approach. The main difference is that we set this problem for an agent model with connected perception and action, taking place in a geometric space, whereas game theory agents have generally no perception of their opponents. To relate the perception of others to space and geometric shapes, and their likely moves, demands different mechanisms. The computation of the likely aimed target is a very preliminary example, mimicking in a simplistic way the dynamics of shared attention.

Similarly, one could consider that the empathon is a simple particular case of the general "belief, desire, intention" (BDI) architecture [Bratman, 1987; Wooldridge, 2000], because this agent has some belief about the world (its surrounding percept), a desire (to reach targets), and an intention (to make a move toward a target, to keep or drop it as a goal). However, BDI architecture generally does not address how to integrate the representation of others. And when it does, this is not related with a model of perception/action embedded in a geometric world.

Of course, a lot of other agent models evolve in a geometric world, which grounds their perception mechanism. "Sugarscape" [Epstein and Axtell, 1996], or social insect models [Dussutour *et al.*, 2004; Deneubourg and Goss, 1989], include populations of agents with a limited perception of the geometric territory in which they evolve, and choosing actions according to this perception. Hence, the empathon could be seen as yet another model of this type. In many respects, it is. The difference is also to include in the agent a representation of the perception of the other. Such a feature is generally not included in the models we cited.

This problem of representing the representation of the other brings immediately new questions and new difficulties that we only begin to uncover in this paper. We would like to conclude this discussion by mentioning some developments that we envisage in the future.

Our preliminary results tend to show that empathic capacities are not so important in the competition for reaching external targets. We expect them to be much more important in a game where the agents play together. One can imagine different settings, where some agents are seeking contacts of the others, and others are escaping them on the contrary. In this case, the mutual observation becomes crucial to manage to reach or to avoid the other.

We expect that agents will have to spend a part of their time to actively observe each other, in order to get information about their states and intentions. This implies that the representation of the other gets some persistence on several time steps, even if it is not actually in the direct vision zone, and sets the problem of modeling the other's moves while it's not directly perceived.

We intend to investigate more carefully a particular situation in these future experiments: when an agent conceives itself in the representation of the other. For instance, we can imagine games in which the agent would need to assess how it is seen by another agent (for instance as a threat or an opportunity). Such an investigation appears particularly important to challenge different theories about the role of the others in the design of a self [see Gopnik, 1993].

This approach will certainly progressively add more functions and variables to the agents. For instance, it will be interesting to introduce a more sophisticated memory, connected with a variable coding for 'pleasure' and 'pain'. It is indeed difficult to investigate the constitution of a self without some more complete treatment of the perception of time, and how this perception is modified by the presence of others.

Nevertheless, we consider important to begin with the simplest settings, to understand very well the role of each added feature in this progress to higher cognitive capacities.

References

[Baron-Cohen *et al.*, 1985] – Baron-Cohen S., Leslie A. and Frith U., Does the Autistic Child have a 'Theory of Mind'?. *Cognition*, 21, 1985, pp. 37-46.

[Bratman, 1987] – Bratman M. E., Intention, plans and practical reason, CSLI Publications, 1987 [1999], http://csli-publications.stanford.edu/site/1575861925.html.

[Conte *et al.*, 2008] – Conte R., Paolucci M. and Sabater Mir J., Reputation for Innovating Social Networks. *Advances in Complex Systems*, 11(2), 2008, pp. 303-320.

[Davies, 1994] – Davies M., The Mental Simulation Debate. *Philosophical Issues*, Vol. 5, *Truth and Rationaluty*, Ridgeview Publishing Company, 1994, pp. 189-218.

[Decety *et al.*, 1997] – Decety J., Grèzes J., Costes N., Perani D., Jeannerod M. *et al.*, Brain activity during observation of actions. Influence of action content and subject's strategy. *Brain*, 120, 1997, pp. 1763-1777.

[Deffuant, 1998] – Deffuant G., Les modèles cognitifs à l'épreuve des formes du religieux : proposition de directions de recherche centrées sur l'empathie. *Intellectica*, 1998/1-2, 26-27. pp. 89-109.

[Deneubourg and Goss, 1989] – Deneubourg J. L. and Goss S., Collective patterns and decision-making. *Ethology, Ecology & Evolution*, 1, 1989, pp. 295-311.

[Di Tosto *et al.*, 2007] – Di Tosto G., Paolucci M. and Conte R., Altruism Among Simple and Smart Vampires. *International Journal of Cooperative Information Systems*, 16(1), 2007, pp. 51-66.

[Dupuy, 1992] – Dupuy J.-P., *Introduction aux sciences sociales*, Paris, Ellipses, 1992.

[Dussutour *et al.*, 2004] – Dussutour A., Fourcassie V., Helbing D. and Deneubourg J. L., Optimal traffic organization in ants under crowded conditions. *Nature*, 428 (6978), 2004, pp. 70-73.

[Epstein and Axtell, 1996] – Epstein J. and Axtell R., *Growing Artificial Societies: social science from the bottom up*, The MIT Press, 1996.

[Gallese *et al.*, 1996] – Gallese V., Fadiga L., Fogassi L. and Rizzolatti G., Action recognition in the premotor cortex. *Brain*, 119, 1986, pp. 593-609.

[Gallese and Goldman, 1998] – Gallese V. and Goldman A., Mirror neurons and the simulation theory of mind reading. *Trends Cogn Sci*, 2, 1998, pp. 493-501.

[Gopnik, 1993] – Gopnik A., How we know our mind: the illusion of first person knowledge of intentionality. *Behavioral and Brain Sciences*, 16, 1993, pp. 1-14 et 90-100.

[Hill *et al.*, 2004] – Hill E., Berthoz S. and Frith U., Brief report: cognitive processing of own emotions in individuals with autistic spectrum disorder and in their relatives. *Journal of Autism and Developmental Disorders*, 34 (2), 2004, pp. 229-235.

[Hoffman, 2000] – Hoffman M. L., *Empathy and moral development: Implications for caring and justice.* Cambridge, University Press, 2000.

[Maynard-Smith, 1982] – Maynard-Smith J., *Evolution and the theory of games*, Cambridge University Press, 1982.

[Nash, 1950] – Nash J., Equilibrium points in n-person games. *In: Proceedings of the National Academy of Sciences of the United States of America*, 36 (1), 1950, pp. 48-49.

[Rizzolatti *et al.*, 2001] – Rizzolati G., Fogassi L. and Gallese V., Neurophysiological mechanisms underlying the understanding and imitation of action. *Nat. Rev. Neurosci.*, 2(9), 2001, pp. 661-670.

[Rosenblatt, 1958] – Rosenblatt F., The perceptron: a probabilistic model for information storage and organization in the brain. *Neurocomputing. Foundations of Research*, MIT Press, 1958 [rééd. 1988].

[Scheler, 1923] – Scheler M., *Nature et formes de la sympathie* [1923], Paris, Éditions Payot, rééd. 2002.

[Smith, 1759] – Smith A., *Theory of the moral Sentiments* [1759], Raphael D. D. and Macfie A. L. (éds.), Oxford University Press, 1976.

[Stein, 1917] – Stein E., *On the problem of empathy* [1917], Washington, ICS Publications, 1989.

[Von Neumann and Morgenstern, 1944] – Von Neumann J. and Morgenstern O., *Theory of games and economic behavior*, Princeton University Press, 1944.

[Wooldridge, 2000] – Wooldridge M., *Reasoning about Rational Agents*, The MIT Press, 2000.

Framework for M&S with Agents in regard to Agent Simulations in Social Sciences: Emulation and Simulation

Franck VARENNE [1]

Abstract – *The aim of this paper is to discuss the "Framework for M&S with Agents" (FMSA) proposed by Zeigler* et al. *[2000, 2009] in regard to the diverse epistemological aims of agent simulations in social sciences. We first show that there surely are great similitudes, hence that the aim to emulate a universal "automated modeler agent" opens new ways of interactions between these two domains of M&S with agents. E.g., it can be shown that the multi-level conception at the core of the FMSA is similar in both contexts: notions of "levels of system specification", "behavior of models", "simulator" and "endomorphic agents" can be partially translated in the terms linked to the "denotational hierarchy" (DH) and recently introduced in a multi-level centered epistemology of M&S. Second, we suggest considering the question of "credibility" of agent M&S in social sciences when we do not try to emulate but only to simulate target systems. Whereas a stringent and standardized treatment of the heterogeneous internal relations (in the DH) between systems of formalisms is the key problem and the essential challenge in the scope of Agent M&S driven engineering, it is urgent too to address the problem of the external relations (and of the external validity, hence of the epistemic power and credibility) of such levels of formalisms in the specific domains of agent M&S in social sciences, especially when we intend to introduce the concepts of activity tracking.*

Keywords – Framework of M&S with agents, Agent-simulation, Agent-based modeling, emulation, social sciences, epistemology, credibility of models, denotational hierarchy.

1. University of Rouen (France) and GEMAS (CNRS and University Paris-Sorbonne) (franck.varenne@univ-rouen.fr).

Introduction

Recent trends in the sciences of complex systems (integrative biology, cognitive economics, computational sociology, etc.) show the spreading of complex multi-level systems of models and simulations [Varenne, 2009a-b]. Due to multiple imbrications of types of symbols and of types of computations, the epistemic status of such complex simulations is most of the time problematic. New questions arise: for which reason, according to which criteria, can we decide that a given complex computer simulation is only a calculus of a model, or a conceptual exploration, or a credible world or a virtual experiment [Sugden, 2002]? It is probable that this status is not decidable only by looking at the types of the used models nor by looking at the type of simulator – or computational template – at stake [Phan and Varenne, 2008]. Facing some similar considerations on the increasing complexity of simulations, Winsberg [2009] claims that we have to adopt a deferetentialist epistemology: *i.e.*, we ultimately have to defer to the beliefs of the modeler and to his expertise in his domain in order to find and legitimate the epistemic status of each complex computer simulation (CS).

Although it surely is a cautious strategy to defer to specialists of the domain when modeling, this strategy is not systematically working when you have to work with many different disciplines *at the same time* (such as sociology, psychology, ecology and economics, as you can see now in some complex multi-level and multidisciplinary CS). Here, the problem relies on the diversity of regional – *i.e.* disciplinary – epistemologies of models and simulations. The problem is exactly this one: it does not suffice to have a common framework and ontology for your formalisms to have the possibility to find an agreement between the epistemological standpoints and commitments of various specialists on the epistemic status of the complex CS in question. Even when a meta-aspectual point of view is available (thanks to the availability of a common ontology), you cannot be sure that a common epistemological standpoint will automatically arise from this common ontology. Having a common – minimal – ontology does not guarantee that you will have a common – even minimal – epistemology. Both are largely independent. Hence, the question can be asked: if you adopt a deferentialist epistemology to evaluate the epistemic status of a complex multidisciplinary CS, which specialist will you have to defer to? Undoubtedly, the need for some new epistemological reflections reappears in this context of complex multimodeling and CS.

The first thing we can say is that the origin of the difficulty lies in the fact that the problem of the validation of models and simulations is not only a problem of *internal validity* between types of systems, nor only a problem of *external validity* of formalisms in regard to data. It is a mix of the two.

According to Guala (2003),

> The result of an experiment E is internally valid if the experimenter attributes the production of an effect B to a factor (or set of factors) A, and A really is the (or a) cause of B in E. Furthermore, it is externally valid if A causes B not only in E, but also in a set of other circumstances of interest, F, G, H, etc.
> […]
> Whereas internal validity is fundamentally a problem of identifying causal relations, external validity involves an inference to the robustness of a causal relation outside the narrow circumstances in which it was observed and established in the first instance.

In fact, in complex CS, there is a tremendous mix between the questions of external validity and the questions of internal validity. This is the reason why a deferentialist epistemology is partly right: implementers have to beware of the importance of that deference to experts of the domains to evaluate the epistemic status of their models & CS. But this is the reason why this epistemological strategy does not suffice either.

The aim of this paper is to introduce conceptual distinctions between the notions of model, simulation and emulation in relation to a hierarchical presentation of symbols so as to provide conceptual tools for facilitating the elucidation of this problem. In particular, by using the recent discriminating and referentialist interpretation of models and complex CS [Phan and Varenne, 2008] based on the concept of denotational hierarchy between symbols [Goodman, 1981], we will show that it is possible to reinterpret some conceptual tools of the Framework for M&S (FMS) described in [Zeigler *et al.*, 2000].

Accordingly, we will address the problem of the conception of a universal "automated modeler agent" [Zeigler *et al.*, 2009] by introducing a distinction between an emulation and a simulation. From this viewpoint, emulation will appear as a kind of simulation, not the only one. This generalizing interpretation enables to explain the partial connections between the interdisciplinary question of the epistemic statuses of complex agent models and simulations, especially in social sciences, and the project of emulating a universal automated modeler agent in the context of the FMS.

1. A referentialist epistemology of levels of symbols

Let’s first remind what we recently proposed to call a “referentialist and multi-level centered epistemology of complex M&S” [Phan and Varenne, 2008].

1.1. A definition of “Model”

Following Hill [2000], we first propose to base this open epistemology on the large definition of a model first given by Minsky [1965].

> To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A.

Note that this large definition is not large enough to take into account the non epistemic roles of models, *i.e.* those roles that are not primarily devoted to the acquisition of a specific knowledge (but to the acquisition of some know-how or some agreement). As noted by [Yilmaz *et al.*, 2006], models can be used in other contexts and for other purposes: training or entertainment, for instance.

Nonetheless, as far as epistemic dimensions of models and simulations are central both to the community working with models in the sciences of complex systems and to the community working with the system theory approach, and as far as these communities meet on this specific role of models, we can consider that this definition remains valid for our specific concern.

This pragmatic definition of epistemic models is interesting because it gathers three important features:

1. An object has not to be a representation to be a model. A model is not always a symbol or a system of symbols referring to something really subsisting. Here, I will take the term “symbol” as denoting any referring entity and the term “symbolization” as denoting any relation of referring or “standing for” [Goodman, 1981].
2. Although a model is not always representational, a model is not in itself a model. The property to be a model is pragmatically defined here because, according to Minsky, an object becomes a model only when related to an investigator and to a specific and contextualized investigation of this investigator. So, it is relatively to this investigation that an object becomes a model.

3. Nevertheless, a model is still characterized as an "object" by Minsky. Note that this does not imply that a model is necessarily a concrete and material object, of course. It can be an equation or an algorithm. But a model remains an "object" to the extent that it possesses an *ontological independency*: it is an independent entity in itself. It is not only a property of an autonomous entity. This "objectivity" of the model is what interests us mostly because it is what justifies the redirection of the questioning towards the model. As an independent entity, a model presents an autonomous behavior which can be investigated in itself.

This is the reason why most scientific models today are formal constructs possessing a kind of unity, formal homogeneity and simplicity. These unity, simplicity and homogeneity are chosen so as to satisfy a specific request (prediction, explanation, communication, decision, etc.).

Given all these listed features, the main function of a model appears more clearly: it is to facilitate the answering of some questions regarding a given investigated object.

1.2. Systems and Models

From this viewpoint, models can be seen as "systems" too, in the sense given by [Klir and Elias, 1985]. According to these authors, a system is a "set of some *things* and a *relation* among the things", *i.e.* an ordered pair S = (A, R) where A denotes the set of relevant things and R denotes a relation among them. Klir and Elias state that "the term 'relation' is used here in a broad sense to encompass the whole set of kindred terms such as 'constraint', 'structure', 'information', 'organization', 'cohesion', 'interaction', 'coupling', 'linkage', 'interconnection', 'dependence', 'correlation', 'pattern' and the like". From this system theory viewpoint, the simplicity which is sought for in every model lies essentially in the *uniqueness of the type of relation* at stake in the system-model.

In their book, Klir and Elias choose explicitly to focus on the *types of relations* and not on the *types of the related things*. By frankly choosing this basic approach for their subsequent conceptions of systems and their interrelations, they aim at freeing them from any interpretation, *i.e.* from any dependence to a particular scientific discipline or specialization. Accordingly, they define a "general system" as an "interpretation-free system chosen to represent a class of systems equivalent (isomorphic) with respect to some relational aspects that are pragmatically relevant". According to them, it follows that the entire practice of designing and processing models can be classified in the set of *theoretically – i.e. not empirically – based activities* [Klir and Elias, 1985].

But this conclusion is problematic when we see all the literature which on the contrary has taken seriously into account the empirical nature of modeling and, especially, of simulations[2]. In fact, it appears that the system theory approach of M&S is not always fine-grained enough for the case of complex simulations and for the analysis of the epistemic roles of simulations. But how is it possible to characterize a simulation today?

1.3. Simulations

Before the computer era, a simulation was defined as a kind of model. The simulation of a volcano's eruption through chemical reactions in a classroom was seen as a *phenomenological* model. That is, a simulation was a model that represents and mimics only the *behavior* (the performance) and not the functional *structure* of a real volcano.

In the 1940's, with the arrival of the first digital computers in nuclear physics, a numerical calculation of an intractable mathematical model was called a "simulation": first because analog computers were already called simulators (analog computers were mimicking the target system only through their measurable behavior but not through their physical/structural functioning), and second because a step-by-step discrete processing of symbols could be interpreted as a "behaviorist" – and not structuralist – processing of a formal model at a micro-level.

This common emphasis on the "behavior" can be recognized too in the characterization of a "simulator" by [Zeigler *et al.*, 2000]: "A simulator is any computation system […] capable of executing a model to generate its behavior". But a simulator is not a simulation. I will come back to this topic later.

Because most CSs were initially founded on the processing of a unique formal model, many papers characterize a computer simulation as a calculus of model. Simulation is presented as a kind of second order modeling, a temporal modeling of a model. Scholars, especially in physics, computer science and engineering sciences, are often used to say that "a simulation is a model in time".

2. See for instance: [Varenne, 2001; Mäki, 2002; Guala, 2002, 2003; Peck, 2004; Humphreys, 2004; Varenne, 2007; Phan *et al.*, 2007; Guala, 2009; Winsberg, 2009].

According to [Hill, 1996],

> Simulation is carried out by causing an abstraction of a real system (the action model) to evolve in real time in order to assist the understanding of the functioning and behavior of this system and to understand certain of its dynamic characteristics, and with the aim of evaluating different decisions.

Following this broadly accepted characterization, [Hartmann, 1996] states that:

> Simulations are closely related to dynamic models [*i.e.* models with assumptions about the time-evolution of the system] […] More concretely, a simulation results when the equations of the underlying dynamic model are solved. This model is designed to imitate the time evolution of a real system. To put it another way, a simulation imitates a process by another process.

For Parker (forthcoming work quoted by [Winsberg, 2009]), a simulation is:

> A time-ordered sequence of states that serves as a representation of some other time-ordered sequence of states ; at each point in the former sequence, the simulating system's having certain properties represents the target system's having certain properties.

However, as noted by [Phan and Varenne, 2008], it is not always true that the dynamic aspect of a simulation imitates the temporal aspect of the target system. Sometimes, a simulation imitates neither the dynamic aspect of the model nor the temporal aspect of the target system.

In the case of a rule-based CS, or in the case of what is often called a "model of simulation", a simulation of the model cannot imitate the dynamic aspect of the model because it is the simulation itself which is the dynamic aspect of the model, and nothing else. Moreover, in the case of a rule-based CS specifically designed to produce only a final picture of a complex dynamic object (such as a botanical plant) through a computational trajectory which is not mimicking the real trajectory of the real system, the simulation is neither mimicking any dynamic model nor the temporal aspect of the target system. For instance, it is possible to simulate the growth of a botanical plant sequentially and branch by branch (through a non-mimetic trajectory) and not through a realistic parallelism, *i.e.* burgeon by burgeon (through a mimetic trajectory), and to obtain the same resulting and imitating final image (see the case of the AMAPsim software presented in [Varenne, 2007]).

Therefore I have proposed to distinguish between CSs which are *mimetic in their results* from CSs which are *mimetic in their trajectory*. But of course, there exist CSs which are mimetic neither in their results nor in their trajectory. Such CSs are simulations only in that they are the calculation of a "model of simulation" and not because they are simulations of any target system (be it real or fictional).

For all these reasons, it seems no more relevant to see all simulations as "models in time". It is due to the fact that the meaning and the reference of the term "time" are problematic here. It is even more problematic than usually thought (when the sole distinction drawn is between the real time and the time of the simulation) in that the meaning of "time" depends itself of the kind of similitude we want for this simulation.

Through that, we understand too that the term simulation may either denote a *simulation of a model* or a *simulation of an external target system with the help of a model or a set of models*. In the former case, simulation remains an ancillary instrument for the model: the limited role of a simulation of model is to help the model generating some data that reveal the implicit behavior of the model. In the latter case, on the contrary, the model tends to become an ancillary instrument for the simulation of an external target system. So, to simulate *through* a model is not necessary to simulate *a* model, unless the term "simulation" changes its meaning in the same sentence.

Another problem with the traditional definition of a CS is that more and more simulations use *sets or systems of models* instead of a unique and mono-formalized *model*. Hence, it appears necessary to characterize a computer simulation *apart from a central reference to a unique model*. Seen from the viewpoint of the mathematical system theory, and particularly because of its property of *closure under composition*, this point can surely be overcome. In fact, this is one of the most powerful and interesting properties of the DEVS approach in M&S: a system of model is always assumed to be a model itself. But [Recanati, 2008], who is working on computer simulations of hybrid reasoning, has shown that it is not necessary to have a formalized metalanguage to enable computable interactions between ontologies. In this context, some iconic aspect of ontologies (e.g.: the form of the basis symbols) are used to bypass the recourse to any global metalanguage.

So, because questions on the kinds of "similitude" and "iconicity" of symbols at stake in a CS seem to persist and even re-emerge (e.g. if we want to clarify the meaning of "time" in this context, or if we want to determine whether a CS simulates *a model* or *an external target system*, or if we want to decide whether the simulation is a *unique model in time* or not), I suggest characterizing simulation apart from the notion of model. As a consequence, the temporal dimension of simulations itself will appear as an iconic aspect among others in any simulation: as we have seen, temporality can be defined through iconicity, but the converse is not true. So, the general term of *symbolization* ("denoting all cases of standing for" [Goodman, 1981] and itself implied by the notion of iconicity) seems to be a more fundamental term than "time" or "behavior" as far as a characterization of simulation – even of CS – is concerned.

1.4. A characterization of simulations

Two caveats: first, I will give a characterization and not a definition of a simulation; *i.e.*, it is possible to find processes which could be characterized that way without being properly considered as simulations. But my claim is that any simulation can be described this way. Second, note that this characterization *refers neither to an absolute similitude (be it formal or material) nor to a unique dynamical model*:

Generally and minimally speaking, a simulation can be characterized as *a strategy of symbolization taking the form of at least one step by step treatment*. This step by step treatment takes time. But the real or simulation time it takes does not necessary denote nor imitate a period of time whether from the model viewpoint or from the target system one. This step by step treatment proceeds at least in *two major phases*:

1. 1st phase (*operative phase*): *a certain amount of operations running on symbolic entities (taken as such) which are supposed to denote* either real or fictional entities, reified rules, global phenomena, etc.
2. 2nd phase (*observational phase*): an observation or a measure or any mathematical or computational *re-use* (e.g., in a CS, the simulated "data" taken as input data for a model or another simulation, etc.) of the result of this amount of operations *taken as given* through a visualizing display or a statistical treatment or any kind of external or internal evaluations.

For instance, in an analog simulation, some material or physical properties are taken as symbolically denoting other material or physical properties (be they of the same kind or not).

More specifically, a *CS (computer simulation) is a simulation for which we delegate (at least) the first phase of the step by step treatment of symbolization to a digital and programmable computer.*

In particular, a CS is a "calculus of a model" or a "model in time" when the symbolic entities which are operated upon during the operative phase can be presented (*i.e.* rewritten without informational loss) as a unique and formal construct possessing a kind of unity, formal homogeneity and simplicity. Seen from our larger characterization of simulations, it is not necessarily the case.

But there is one property of simulations which appears through this focus on symbolization: the changing of levels of symbols during the process. It is precisely on this point that we can fruitfully reconnect with the Framework for M&S.

1.5. Subsymbols and Iconicity in Simulations

In [Phan and Varenne, 2008], we suggested considering the changing *symbolhood* of symbols at stake in any strategy of simulation. In fact, during the observational phase, marks which were first treated as genuine symbols, *i.e.* as denoting entities, are finally treated as *sub-symbols*: so, they are treated at another level than the one they first operated. At the end of process, it is the result observed which gains a proper and new symbolic nature. And this is relatively to this new symbol or system of symbols that the first symbols become sub-symbols. Let's remind that, according to [Smolensky, 1988], subsymbols operate in a connectionist network at a lower level than the symbols. As such, they can be seen as constituents of symbols. Subsymbols "participate in numerical – not symbolic – computation": the kinds of operation on symbols (computations) are not the same at each level. In our context of reflections on simulations, it is not necessary to adopt the realist connexionist viewpoint of Smolensky to borrow him this term. Berkeley [2000, 2008], for instance, has shown that the subsymbols of Smolensky can be interpreted in regard to a larger range of levels and from a relativistic point of view.

In view of that, in the particular context of the strategy of a kind of symbolization which is specific for a simulation, we can say that some symbols are subsymbols *relatively* to the final symbol resulting from the second phase. Consequently, sub-symbolhood appears as a key feature of any simulation. But – what is most salient – subsymbolhood appears as a changing and relativistic – not definitive nor absolute – property of a symbol used in a simulation, during the simulation.

Through that, we can see that our characterization of simulation leads us to similar considerations as the ones presented by Zeigler *et al.* [2000] (chapter 1): simulation is a question of levels of symbols. But, is it necessarily or always a question of levels of systems – or even languages – strictly speaking? My suggestion can be now a little more substantiated and anticipated: characterizing a simulation as a relation between levels of systems is a particular case of characterizing it more largely as a relation between types of symbols through a given step by step treatment. In the former case, we have exact or approximate emulation in view (where emulation is defined as a particular case of simulation, as we will see). In the latter, we have the more general case of simulation in view.

First, let's clarify a bit the notion of iconicity and the correlative notion of subsymbolhood. In the 1960's, it was sometimes said that simulations were "iconic modeling" [Frey, 1961]: it was to be understood in the sense of *iconicity images can have*. *I.e.* simulations were seen to use the same – or similar – physical features as the ones possessed by the target system they were told to symbolize. The linguist Olga Fischer [1996] defines iconicity as "a natural resemblance or analogy between a form of a sign […] and the object or concept it refers to in the world or rather in our perception of the world". But she insists on the fact that *not all iconicities are imagic*. There are *diagrammatic iconicities*. For instance, there are relations of symbolization where the direct likeness between a signifier and a signified (such as in the onomatopoeia "miaow" for "sound made by cat") is missing: "instead there exists an iconic link between the horizontal relations on the level of the signifier and the horizontal relations on the level of the signified" [Nänny and Fischer, 1999]. It is the case in the sentence "veni, vidi, vici" where it is the order of events which is iconically denoted through the ordering of the verbs. But not all diagrammatic iconicity remains structural as is this last one: it can be much more indirect in that it can stem from the *semantics of the language in use*. There are *semantic diagrammatic iconicities* (*ibid.*). This is this apparently paradoxical indirection of iconicity which is often used in the creation of metaphors.

That is the reason why Fischer [1996] states that an *iconic semiotic relation* is first of all *relative to the standpoint of the observer-speaker-interpreter*. From these considerations, it follows that the most important is the property of an iconic relation to be – *relatively to a given language or vision of the world – less dependent of this language*. If we follow such a post-structuralist linguistics, iconity is no more univocally defined in terms of a superficial and implausible absolute resemblance between things and signs nor by an absolute homomorphism between pre-defined and pre-structured systems (the *system* of signs, on the one hand, and the *system* of things taken in a slice of the reality, on the other). But *iconicity* is more largely and more fundamentally defined in terms of *independence from a given language*.

Hence, if we want to focus on the epistemic power of complex simulations, the choice is no more only between interpreting it as a pure *material analogy* or as a pure *formal analogy*. The situation now is much more complicated.

But let's determine further these relational properties of iconicity and subsymbolhood:

- We'll say now that a symbol is *more iconic* than another symbol *in regard to a given language* in which it is inserted and used when its function of symbolization (its denotational power) is less dependent from the *conventional rules* of this given language.
- Correlatively, we'll say that a symbol S2 can be interpreted as a *subsymbol of another symbol* S1 *in regard to a given language* iff:
    1. S2 is more iconic than S1 in regard to this given language;
    2. There exists a computational operation (a step by step operation on symbols characterized by a weak *combinatorial power*) on S2 and other symbols of the same level which can produce a symbol of the type S1.

It could be objected that this step by step operation on elementary symbols is precisely of a conventional nature and that it is, as such, just as any other convention-based linguistic rules. The answer here would be relativistic too: this is a matter of degree. When we use iterated computations instead of a sophisticated intrications of grammars (*i.e.* when we use a computerized management of symbols instead of speaking or thinking), we have access to symbols for which only rules with weak *combinatorial power* are available.

More precisely, we can define:

1. The *combinatorial power* of a level of symbols as the measure of the *variety* (*i.e.* the number of different types) of combinations and operations on symbols which are available at this given level. In a CS, the weakness of the combinatorial power is compensated by the number of reiterated elementary computations.
2. The *degree of iconicity* as the (relative) measure of the degree of independency of the denotational power of a level of symbols relatively to the conventional rules of a neighboring level of symbols or language.

In a *denotational hierarchy*, we observe that the *degree of combinatorial power* of a level of symbols tends to be inversely proportional to the *degree of iconicity* regarding the neighboring level.

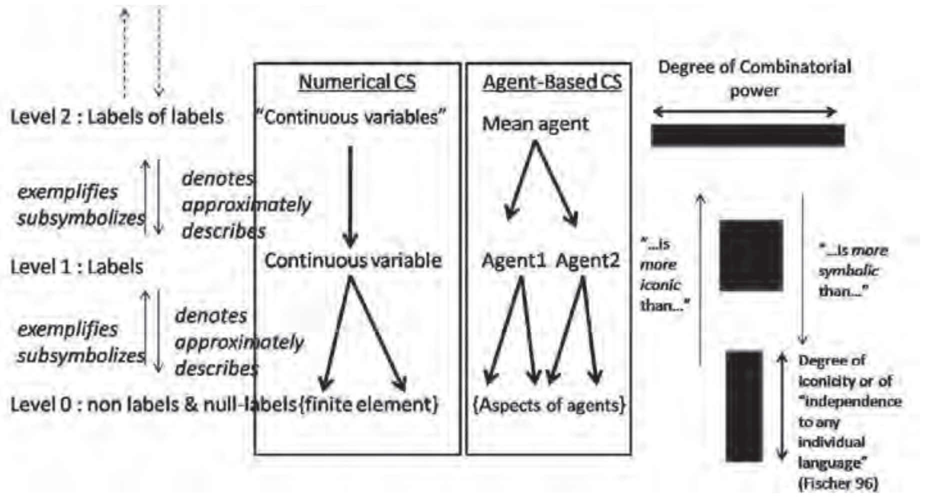


Figure 1. Denotational hierarchy and computer simulations.

In Figure 1, I represent the notion of the denotational hierarchy of [Goodman, 1981]. Then, I draw a parallel between the hierarchy of levels of symbols in such a hierarchy and the similar hierarchies in numerical simulations and in agent-based simulations. The relation of subsymbolization can be interpreted in terms of an exemplification whereas the relation of denotation can be interpreted in terms of an approximate description.

1.6. Simulations of Models and Simulations of Target Systems

From what has been said, one can explain why the term simulation can have different meanings in technical literature. According to [Ören, 2005; Yilmaz *et al.*, 2006], for instance, "simulation has two different meanings: (a) imitation and (b) goal-directed experimentation with dynamic models". In this section, we will show to what extent our conceptual analyses confirm and explain further this matter of fact.

It has been said earlier that the term simulation may either denote a *simulation of a model* or a *simulation of an external target system with the help of a* *model or a set of models*. The characterization just given can help us explain the things further.

First. We are right to say that a computer simulation is a “simulation of a model” when its specific strategy of subsymbolization essentially is taken as a strategy of *subsymbolizing* the dynamic of the model. From this viewpoint, a lapse of time taken in the dynamic of the model is *iconically denoted* by a lapse of time of computation in the CS. An iconic semiotic relation takes place here because *a lapse of time is denoted through another lapse of time*. This iconic relation is not an “imitation” in the proper sense, but it is what permits to characterize the second meaning of “simulation” – according to [Yilmaz *et al.*, 2006] – as a kind of experimentation. So, temporal iconic representations of dynamics of models such as “simulations of models” can be specifically characterized as “models in time” too. But this particular denotation of an aspect of a single model cannot be found in all simulations. So, it cannot be generalized. The well recognized fact that many CSs can be seen as “models in time” is more a regional consequence of the prevailing classical use of a certain kind of subsymbolization based on an iconic representation of time than the contrary. *I.e.*: this is not the fact that a given CS is a model in time which entails the presence of a kind of subsymbolizing in this CS, but the contrary. Surely, a minimal CS is often based on a *subsymbolizing* of the dynamic of a given model. But a CS has not necessarily to denote iconically the time elapsed in a dynamic of its related model or models or system of models to be a simulation.

$$\begin{tikzcd}
& M \arrow[dl, dashed] \arrow[d, dashed] \arrow[drr, dashed] & & & \\
Mt_0\text{-}t_1 \arrow[r] & Mt_1\text{-}t_2 & & Mt_2\text{-}t_3 \arrow[r] & {}
\end{tikzcd}$$

Figure 2. A computer simulation seen as a simulation of model.
Legend: M: Model; Mt_0-t_1: subsymbolization of the dynamic of the model between t_0 and t_1.

Second. A CS can be called a simulation for another reason: it can be seen as *a direct simulation of an external target system* and not as a simulation of model. Here, we find what [Yilmaz *et al.*, 2006] call the first meaning of simulation: imitation. In this case, it is implicitly assumed that symbols at stake in the simulations are entering in some direct iconic relations to some external properties of the external target objects.

From this viewpoint, contrary to what prevailed in the first case, lateral and external relations between symbols and target entities or target symbols or labels have to be taken into account.

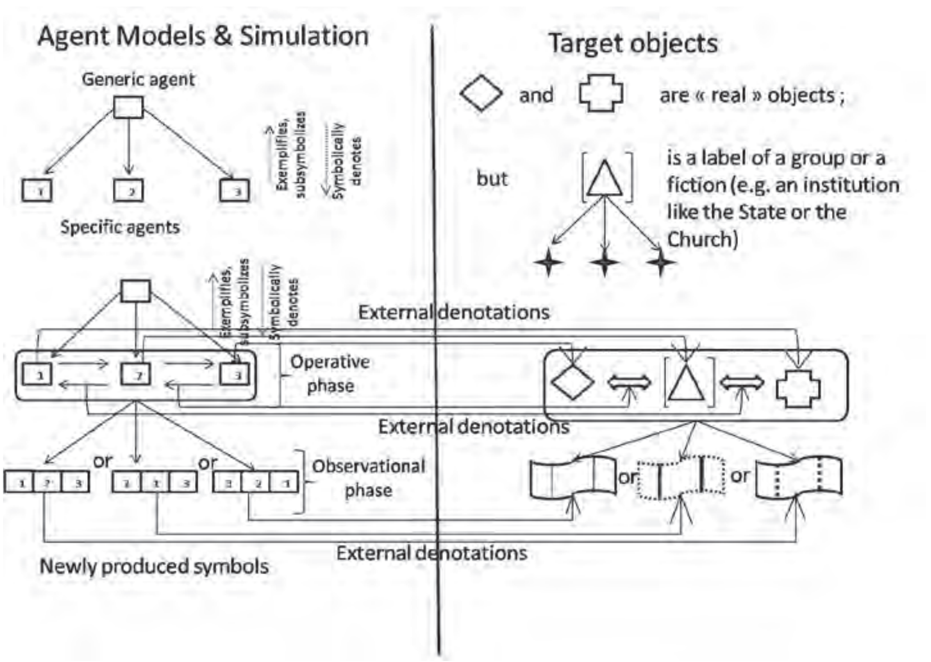


Figure 3. Denotational hierrarchy and external denotations.

In Figure 3, besides the internal relation of subsymbolhood between a symbol of the generic agent and the symbols of specific agents, the *external denotational relations* between these symbols and the target objects (be they real, constructed or fictional) are represented.

These external relations of denotation can be seen as iconic or as symbolic too. But this is not with the same meanings as the ones introduced in the previous case.

For instance, in Figure 3, the target objects ✦ are denoted by $\boxed{2}$ through a *symbolic external denotation*. This external denotation is symbolic because it goes through the intermediary symbol △ of which denotational properties are based on conventional rules (linguistic and social rules at the same time).

On the contrary, the target object ◇ is denoted by $\boxed{3}$ through an *iconic external denotation* because no such conventional intermediary is necessary: we can see iconicity in this case (*i.e.* a weak dependence to *any* language convention) in that there is a one by one connection between the specific symbol and the specific target object.

Because this iconicity is decided in regard to *any* or to a great number of languages or systems of symbols, it can be said to be an *absolute iconicity*. This is a great difference with the *internal iconicity* we presented first, which serves to characterize any simulation and which always remains relative to a given level of symbols or language. In fact, the latter takes place in the relations of simulation within a denotational hierarchy of levels of symbols, whereas the former denotes symbols or entities which may but have not to belong to any explicit denotational hierarchy. Externally denoted entities or symbols themselves have not to belong to any hierarchy (nor to the same hierarchy as the one of the simulation) to be denoted from a kind of symbol belonging to a model and simulation-oriented denotational hierarchy.

As a consequence, neither simple matching nor direct parallelism between the M&S-oriented DH and any real (or eventually consensual) hierarchy relevant for the target objects is necessary. Another way to coin this is to say that it is not necessary for the denotated target objects to form a *system* to be simulated in a complex CS.

In the next section, I will remind some of the key ideas of the Framework for M&S. Afterwards I will show how to interpret this conception of M&S with the help of the concepts recently introduced. Particularly, I will suggest seeing the FMSA as a specific conception of the practice of M&S with agents in that it is based on the relatively strong hypothesis that an integration of some *system* of *target objects* within the *denotational hierarchy* is always possible and/or relevant.

2. System Theory and Framework for M&S

From the standpoint of the theory of systems, the process of modeling and simulation and its variant can be interpreted in terms of relations not only between symbols and groups of target entities, nor even between levels of symbols, but always between *levels of system specifications.* As a consequence, the system of target objects (more briefly the *target system*) – or *observation frame* – is situated in an integrated *system-denotational hierarchy*. It takes place at the level 0 of this hierarchy.

2.1. The hierarchy of the epistemological types of systems

As noted by [Zeigler *et al.*, 2000], the hierarchy of *levels of system specifications* – *i.e.* of "*levels at which dynamic input/output systems can be described, known, or specified ranging from behavioral to structural*" [Zeigler *et al.*, 2009] – is very similar to the hierarchy of epistemological types of systems according to George Klir.

In 1985, after having defined the notion of system (see above), Klir gave a taxonomy and hierarchy of *epistemological types of systems*. This hierarchy is derived from the working of 3 primitive notions: "an *investigator* (observer) and his environment, an investigated (observed) *object* and its environment, and an *interaction* between the investigator and object" [Klir and Elias, 1985]. Each type of system in the hierarchy is determined by the kind of investigation at stake.

At level 0 are what Klir calls *source systems*, *i.e. systems* which are the "sources of empirical data regarding specific attributes of investigated objects". According to Klir, "systems on different higher epistemological levels are distinguished from each other by the level of knowledge regarding the variables of the associated source system". From this viewpoint, we see that right from the start, target objects and correlated empirical data are pre-structured in a *system*. This is the main reason why this level 0 can be integrated in the overall system hierarchy.

At level 1, are *data systems*, *i.e.* systems which provide the knowledge of *actual states* of the basic variables within the defined support set. At level 2, there exists "an overall support-invariant relation among the basic variables of the corresponding source system". This relation describes "an overall process by which suites of the basic variables are generated within the support set". Such systems are called *generative systems*. At level 3, systems are called *structure systems*. Each structure system is defined in terms of a set of generative systems or lower systems. These subsystems of a *structure system* interact in some way (e.g. they share variables…). After the level 4, the system begins to have the possibility to change its inner relation. At level 4, specifically, the characterization of the changes is itself support-invariant: such systems are called *metasystems*. At level 5, the characterization of the change can change too "according to a support-invariant higher level characterization". Such systems are called *meta-metasystems*. Finally, Klir claims that *metasystems of higher order* can be defined. From this viewpoint, note that a source system is included in each of the higher level systems: the hierarchy functions as a ladder of embedded systems.

2.2. Hierarchy of System Specifications

Similarly, as recalled in [Zeigler *et al.*, 2009], in the hierarchy of *system specifications*, the systems at level 0 provide only an *input and output interface*. Levels 1 (I/O behaviour) and 2 (I/O function) of this hierarchy correspond to Klir's level 1: *data systems*. In particular, systems of level 1 provide input/output pairs whereas systems of level 2 supplement these pairs by the knowledge of *an initial state*. Systems of level 3 specify further a *state transition*. In that, they correspond to the *generative systems* in Klir's hierarchy (Klir's level 2). Finally, *coupled component systems* form the level 4 of the hierarchy of system specifications. They correspond to *structure systems* in Klir's hierarchy (level 3). Note that, on the contrary to Klir's hierarchy, the hierarchy of *system specifications* does not explicitly take into account the possibility for the state transition to change. So there appear no higher levels of *system specification* than the one corresponding to the fixed structure systems of Klir.

According to [Zeigler *et al.*, 2000], the *central idea* of the hierarchy of Klir – which also applies to the *system specifications hierarchy* – is that "when we move to a lower level, we don't generate any really new knowledge – we are only making explicit what is implicit in the descriptions we already have".

Hence, due to the unique hierarchization and to the integration of all the target objects within the same hierarchy, a change of level can be seen as an explicitation of what is already there, but implicit. As a consequence, simulation cannot appear as anything else than a *simulation of model* as we defined it above (a *set of target objects* being always seen as a *system-model*). As underlined by [Zeigler *et al.*, 2000], "in the M&S context, one major form of systems analysis is computer simulation *which generates data under the instructions provided by a model*" (my emphasis). The authors object themselves that "one could argue that making something explicit can lead to insight, or understanding, which is a form of new knowledge". But they answer that "Klir is not considering this kind of subjective (or modeller-dependent) knowledge". Indeed, they conclude that "although no knowledge (in Klir's sense) is generated, interesting properties may come to light of which we were not aware before the analysis".

On the contrary, when we climb up the hierarchy (from a level n to a level n+m), we need to construct a higher detailed description of a system. In this case, we introduce some new knowledge as it appears in epistemic practices such as *system inference*, *system design* or *model construction*. That is: we try to find a *generative system* or *structure system* which can "recreate the observed data" of some *source system* [Zeigler *et al.*, 2000].

As we can see, from this viewpoint of system theory, simulation remains fundamentally an explicitation of mathematical structures (due in particular to: 1st the condition of closure under composition, 2nd the strong hypothesis of a unique denotational hierarchy). Simulation is always interpreted as a calculus of a model. As it appears for any mathematical construct (when compared to their numerical simulation), it is the model which is always considered as possessing a higher degree *of virtuality and cognitive power* in that it possesses a *higher – because a larger – power of possible denotation* through the supposedly unique *denotational hierarchy* (to which the target objects are all said to belong, at the source system level, in a well-suited systemic form).

3. Explaining different epistemic statuses of Models and Simulations

A problem is that practitioners of models and computer simulations in social sciences (computational economics, sociology, geography…) do not always agree on the fact that CSs are only calculus of models or that they only provide some insight of what is at stake in the hidden core of a unique model. As shown by a review of the literature made in [Phan and Varenne, 2008], models can be seen either as conceptual exploration or as experiment. Simulations can be seen as experiments on models or as direct virtual experiments or as "credible worlds" [Sugden, 2002].

3.1. Models as virtual experiments or as instruments

Founding its analyses on the notions of denotational hierarchy and iconicity presented above, [Phan and Varenne, 2008] have proposed to explain why and to what extent social scientists (and more generally practionners of M&S in the sciences of complex objects) are justified to say that a model has an empirical dimension in itself. In some cases, it is because *some causal factors are denoted in the model through symbols of which external iconicity (not internal) is patent and can be reasonably (consensually) recognized* as a sufficiently realistic conjecture.

On the contrary, it can be shown that models are *seen from a purely instrumentalist standpoint* (*i.e.* models are seen as inductive instruments abbreviating some real experiments) *when the modeler thinks that the measure of the external iconicity of the operating symbols is weak* and when this is *their combinatorial power at a high level in the denotational hierarchy which is mostly requested*.

3.2. Simulations as experiments on a model or as conceptual explorations

A simulation being minimally founded on some kind of internal subsymbolization, every CS of a model treats it at a sublevel "which tends to make its relation to the model analogous to the naïve dualistic relation between the formal constructs and the concrete reality" [Phan and Varenne, 2008]. This is because of this analogy between the *internal relations of subsymbolization* (within the DH) and the *external denotational relations* between symbols and target objects that *such a CS can be seen as an experiment on the model*. Conversely, if the goal of the investigation leads to focus on some residual but *external symbolic (not iconic) aspects* of used subsymbols, we are authorized to see such a CS of model as *a pure conceptual exploration*.

It follows that the *external validity* is no trivial question when we face a complex CS: with a complex CS, it is no more easy to have an overall viewpoint on the relations between symbols at stake or between symbols and objects. No overall viewpoint can univocally lead us to determine once for all the external validity of the CS as a whole. In fact, this external validity depends on the strength of the *alleged external iconic aspects*.

Note that if these *external iconic aspects* are extremely stabilized and characterized, the simulation can be compared to an *exemplification*. In this case, as noted by [Phan and Varenne, 2008], external *validity is not far from an internal one*. The difference is a matter of degree. Seen with the help of our conceptual distinctions made above, this case is precisely the one for which we are justified to make the strong hypothesis of an integration of the target objects not beside but *within* the denotational hierarchy. Moreover, the loose and polysemic relation of simulation becomes a particular one in this extreme case of an exemplification of a target through a simulation: simulation becomes a rigid relation of *emulation*. To emulate is not only to simulate but to *perfectly simulate*: *i.e.* presenting the property to generate the same behavior in any circumstances, even in those circumstances which have not been gathered and used to validate the simulation. This is the reason why, when Copeland [2004] is reminding the exact meaning of the Church-Turing thesis, he is explaining that *emulation* is not an imperfect imitation but a *perfect simulation* in that the simulating system *becomes* a system which proves to be equivalent to the simulated one.

3.3. Simulations as experiments in themselves

Some scholars claim that computer simulations are not real experiments, but *experiments* in themselves. But in what precise sense? After having paid attention to such scholars' claims and analyzed them in their own right, [Phan and Varenne, 2008] have shown that there are at least 4 criteria to decide whether a simulation is not only an experiment *on* the model but an experiment *in itself*.

First, when you see the CS as a *direct simulation of some target objects*, the empiricity of the CS comes from an *experiencing*, that is, from an observation and a comparison between the symbols at stake in the CS, on the one hand, and the target objects, on the other. External validity enters here in consideration. But there are two possible kinds of comparison. Either one can postulate an *external iconic relation* between the resulting symbols of the *observational phase* of the CS and some target objects, or one can postulate such an *external iconic relation* between the *elementary symbols* at stake in the operative phase and some other target entities. The former leads to an *empiricity of the CS regarding the effects* (of the computation), whereas the latter leads to an *empiricity of the CS regarding the causes* (of the computation).

Second, when you see the CS not as a *simulation of a model* (otherwise it still can be seen as an experiment *on* a model as we said above) but as *a simulation of a set of models*, it is not necessary that its empiricity be decided only from a direct comparison between the target objects and the symbols at stake in the DH. It can be decided from an *experimenting* on the internal interactions between levels of formalisms and levels of symbols within the (complex) DH. From this viewpoint, as shown by [Phan and Varenne, 2008], there are two kinds of empiricity: (1) the *empiricity due to the intrica-*

tion of the referential routes of symbols, and (2) the *empiricity due to the defect of any a priori epistemic status*.

Note that a CS borrows its empirical characteristic not from a complete substitutability with the target objects. It borrows it from a *partial substitutability* (in the two first cases) or even not from any substitutability at all, but from the *opacity* of the *intrication of symbols* in the DH (in the two last cases).

Now that we have distinguished between kinds of iconic relations (internal to a DH, external to a DH), and between types of epistemic statuses for a model or a simulation, it is time to determine some epistemological conditions which could be necessary for the formulation of a universal "automated modeler agent".

4. The FMSA and the search for a Universal Automated Modeler Agent (UAMA)

4.1. Agents, Endomorphic Agents and the universal "automated modeler agent"

As shown by [Zeigler *et al.*, 2009], the notion of endomorphic agent is central to the search for a first formulation of a UAMA. Briefly said, agents are objects (in the sense of object-oriented programming) that can have perceptions, beliefs, desire and intentions. Agents have been developed in distributed AI, but in philosophy of mind too, to match the first BDI models in the 70's – see the filiations of [Putnam, 1960] and [Fodor, 1975]. See [Ferber, 1999] too for a thorough presentation.

As noted by [Yilmaz *et al.*, 2006],

> Software agents are entities that (a) are capable of acting in purely software and/or mixed hardware/software environments, (b) can communicate directly with other agents, (c) are driven by a set of goals, objectives, and tendencies, (d) possess skills to offer services, (e) perceive their environment, and (f) can generate autonomous behavior that tends toward satisfying its objectives [Ferber, 1999].

In his definition, the social scientist Nigel Gilbert [2008] chooses to emphasize first on human characteristics such as "autonomy" and "social ability":

> Agents are conventionally described as having four important features:

1. **Autonomy**. There is no global controller dictating what an agent does; it does whatever it is programmed to do in its current situation.
2. **Social ability**. It is able to interact with other agents.
3. **Reactivity**. It is able to react appropriately to stimuli coming from its environment.
4. **Proactivity**. It has a goal or goals that it pursues on its own initiative.

But they are strong convergences between the two approaches: what is called Agent-based Modeling in the computational social sciences – see [Gilbert, 2008] – is quite the same as what is often called Agent-simulation in the modelers and computer scientists' community – see [Yilmaz *et al.*, 2006]. According to [Yilmaz *et al.*, 2006], Agent-based modeling or Agent-simulation can be defined as "the use of agents as design metaphors in developing simulation models".

In this context, it is assumed that "simulation models" are models specifically devoted to simulations understood as *imitations of target systems*. So beware that the meaning on this expression is not based on the general meaning of "simulation" but only on its first meaning (according to Ören and Yilmaz). Such a model can be a simple set of formal rules which can be unrealistic in themselves (in the sense of an external iconic relation) but which are conceived in such a manner that their common and interactive running leads to a realistic (hence imitative) result, once compared to the target system.

Whereas "Agent-based modeling" or "Agent-simulation" is devoted to an imitative role of simulations, what [Yilmaz *et al.*, 2006] call "Agent-based simulation" refers – on the contrary – to the instrumental role of agents formalisms.

> Agent-based simulation is the use of agent technology to generate model behavior or to monitor generation of model behavior. [Yilmaz *et al.*, 2006]

It is important to note that, in this case, the term simulation changes its meaning: it is no more to be understood as an imitation of a target system but as "a behavior of a model", as a "model in time" or as an "experimentation on a model".

We can explain this distortion by saying that, in such simulations, the emphasis is on the *internal iconic relations* and not on *the external ones*.

Now, what is an endomorphic agent?

An endomorphic agent is a particular agent “that contains models of itself and/or of other endomorphic agents” [Zeigler *et al.*, 2009].

When we search for a UAMA, we aim at formulating “models of mind” which could be incorporated in agents so that these agents could be said to *emulate* some of the human cognitive capacities (*ibid.*). In particular, the theory of the massive modularity of mind [Carruthers, 2006] – because offering the hope that an easy modeling of a multiplicity of simple modules in mind will soon be reachable – could be a way to give a first outline of a UAMA.

The necessity for a sufficiently evolved agent to construct in his mind – sooner or later – a “theory of the mind” of others and of himself can be simply and logically demonstrated [Zeigler *et al.*, 2009]. It has been largely recognized by evolutionary psychologists too.

From these considerations, we can infer that an endomorphic agent would meet the challenge to conciliate the two different kinds of Agent-directed simulation. It would have to conciliate the property to be an Agent-based model with the property to run an Agent-based simulation of itself (as agent). In the Agent, the prescribed Agent-based simulation of itself will give rise to a modular representation of itself (most of the times unrealistic from its viewpoint), whereas the Agent-simulation will – on the contrary – determine a representation of itself as a familiar and somewhat realistic agent (“realistic” compared to real external systems).

The problem of doubling the aspects (even when they are incompatible) on a same entity is not inescapable, of course; but it demands some careful attention to what kinds of different semiotic relations (internal, external, iconic, subsymbolic) are at stake in each case. This problem becomes all the more acute when we intend to build a modeler agent, moreover a “universal autonomous modeler agent”.

4.2. The universal autonomous modeler agent and the “modeler subjective knowledge”

We won’t enter here in the debate about the validity and significance of such assumptions in the general project of intelligence modeling. Our goal is more modest: it is to show one of the consequences of such an approach on the alleged epistemic role of models and simulations.

Thanks to our previous analyses and conceptual distinctions, we can understand that the formal construct of a universal endomorphic agent, which would construct by himself – at runtime – a theory of his mind-body, is a way for the FMSA to *guarantee the continuous integration* of the target objects in a *unique denotational hierarchy*, during the whole process of M&S. In fact, the system theoretic vision, the constraints of strict embedding between levels of symbols and the condition of closure under composition of systems authorize to take into account and integrate in the hierarchy of system specifications what Klir nevertheless rejected and called the "subjective (or modeler dependent) knowledge". This is the reason why it is justified to see a real promised land in this new project.

But we have shown above that the relations between *the levels of symbols within the DH* and the relations between *symbols of the DH and some target objects or target symbols* (these latter being based on the modeler dependent knowledge) are *not* of the same nature: in particular, the former are supposed to give rise (sometimes) to *relative internal iconicities* whereas the latter can give rise to *absolute iconicities*. So, the logical grammars of these iconicities – and then of these two types of relations – are *not the same*. If we neglect this difference, the diversity and the real coherence of the epistemological positions concerning the epistemic statuses of models and simulations among the different practices of M&S in complex sciences remain unexplainable.

So it appears that one of the greatest challenges for the search for a UAMA could be the careful formulation of this *distinction of nature and of standpoints* on symbols and on relations of symbols and target objects for any modeled cognitive process. Otherwise, it is not excluded that a simple partially auto-similar and auto-scopic agent (the formulation of which would be based on a rough homogeneization of the different kinds of relation of denotation) would be another pitfall in the quest for a greater unification of the tools and practices of M&S.

5. Emulation of Systems, Simulation of Agents

The last sections have shown that when we adopt the system theory approach for the design of agents which would be capable of modelling & simulating their world and other agents in a similar way as behavioral and social scientists – or even common people – do in their daily life, we have to make the hypothesis that external iconicities could be reduced to internal ones.

This hypothesis is strong. The problem it arises is not far from the one posed by Putnam [in 1991] when arguing against the computational view on mind (although Putnam himself had been one of the leader of this view in the 1960's): the denotational power of a symbol – or of a given level of symbol – not only depends on its insertion in a unique, closed and finite set – or hierarchy – of symbols but also on the physical and socio-linguistic context of this symbol or level of symbols in the real world.

But, according to the suggested approach here, this argument does *not* suffice to condemn us to any relativism or vitalism, nor to any refusal of the project of building a UAMA. On the contrary. It serves to make the challenge more precise and efficient.

Accordingly, in this last section, I will show how the use of the distinction between *external* and *internal iconicities* could help us to distinguish between an *approximate morphism* and an *imperfect simulation*.

5.1. Exact Morphisms, Approximate Morphisms and Kinds of Iconicity

As shown in [Zeigler *et al.*, 2000: chapter 12], from the FMS point of view, it can be useful to treat the horizontal relations between systems that belong to the same level of specifications. A relation which establishes "a correspondence between a pair of systems whereby features of the one system is preserved in the other" (*ibid.*) is called a *preservation relation* or *system morphism*.

In particular, [Zeigler *et al.*, 2000] introduces morphisms that are "such that higher level morphisms imply lower level morphisms": "this means that a morphism that preserves the structural features of one system in another system at one level also preserves its features at all lower levels" (*ibid.*). The existence of this possibility is coherent with two facts: 1) the fact that, from this viewpoint, going down the levels in the system specification hierarchy "corresponds to a simulation process, *i.e.* generating the behaviour of the model given its structure" [Zeigler *et al.*, 2000: chapter 14; 2] the fact that, when simulating, *i.e.* when going down the levels, our knowledge cannot increase at all, but only be rendered more explicit (see above).

In this context, a morphism is said to be *exact* when all the features of interest in the two systems are exactly preserved. In other words, a morphism is exact *iff* any of the two systems *emulates* the other. On the contrary, a morphism is said to be *approximate* when not all features of interest are preserved in this relation (*ibid.*).

Of course, there can be different kinds of simulations for the same system. In computational economics and social sciences, it is often said that agent-based simulations of social phenomena can be used to *explain*, at the micro-level, the *mechanisms* of the social phenomena whereas holistic models, working at a macro-level, are said to be phenomenological and of an instrumental nature. These mathematical and holistic models nevertheless can be simulated through discretization and other numerical tricks: but the finite elements which are the bases for such simulations of models are *not* in a relation of *external iconicity* with some target objects. On the contrary, in the case of individual-based simulations, simulated agents can be said to be related to such objects through an *external iconic relations*. Hence, it appears that a *simulation of agents* is not the same as an *emulation of systems*.

More precisely, it follows that the property to be a *relevant simulation* at a given level is *not an intrinsic property which could always be inherited only from a position in the hierarchy*. In particular, it cannot be inherited only by guaranteeing that a system at a higher level is in a morphism relation to another system at this higher level, this latter having a relevant simulation, at a lower level, for its own.

So, if we do not want to defer each time to a *subjective viewpoint* of the modeller, and if we want to implement endomorphic agents who would be automated modelling and simulating agents, there is a necessity to objectify and formalize this *external relation of denotation.* A way to do this could be to look for a metric suitable for an objective evaluation of the simulation error in the sense valid for a *simulation of a target objects*.

5.2. Towards a Metrics for Errors in Simulations in regard to external Iconicities

Now that we take into consideration the semantics of symbols at stake, it surely appears a challenge to find a *metrics* which would be appropriate for the formulation and the measure of the distance between a *desired simulation of a target* and the *simulation obtained*.

Such a metrics would be a useful tool as far as endomorphic agents are sought for [Zeigler *et al.*, 2009]: otherwise, how would it be possible to implement credible (for behavioral and social scientists) evolutionary endomorphic agents without having an idea of how they can assess their own performance in modelling and simulating? In this case, the necessity to implement the *modeler's knowledge and point of view* leads to the necessity to make a place, in such agents, to a sensibility to the *external iconicities* of the models they build, beside their sensibility to *internal ones* or to *isomorphisms*.

Another problem is that external iconicity is founded on a *weak dependence* of the denotational power of symbols of interest from any linguistics systems or any pre-established conventional rules. How can this independence be taken into account in a notational system?

But let's remind that this iconicity remains a *matter of degree* (even when this is seen as "absolute") and that the difference between external validity and internal validity remains a matter of degree too, as a consequence. So it could be a solution to introduce sets of symbols which could interoperate, but which could not be inserted in the overall denotational hierarchy of the M&S process. Such *floating sets of symbols* (seen as floating from the DH point of view) could be considered as "external patterns of reality" or as modules the activity of which has to be simulated from an external point of view. Algorithms of activity tracking [Zeigler *et al.*, 2009] could be used to simulate these external modules (modules taken as external to the DH) as it is not necessary to assume that these modules always belong to the same system or system of systems. More generally, it seems necessary to integrate explicitly the modelling of the *epistemic activity* of the UAMA (its "epistemology"). And a way to do this would be to track this epistemic activity and, moreover, to enable the UAMA to become *aware* of – and to react to – its own *epistemic activity*.

Conclusion

As noted by [Zeigler *et al.*, 2009], the human mind can be seen as the "behavior of the brain" [Carruthers, 2006]. The mind seems fascinating for the specialist in M&S in that it has solved for himself the problem of the System of Systems integration (SoS), *i.e.* the problem to integrate systems with specific functions into a more comprehensive and multifunctional system. Hence, it is perfectly understandable that the search for an endomorphic automated modeller agent seems so crucial today.

This paper has first presented an outline of a *multi-level referentialist epistemology of models* as far as complex M&S are concerned. It has shown that this multi-level epistemology leads to a very similar presentation of the M&S process as the system theory approach adopted for the FMS with Agents [Zeigler *et al.*, 2000]. Nevertheless, this distinct presentation has shown too that the difference between *external* and *internal denotations* helps to focus on the strong and specific hypothesis which is at the basis of the specific FMS approach: the possibility to integrate the target objects as a system in the denotational hierarchy of symbols – or system specification hierarchy – of the M&S process.

The fact that this strong hypothesis is *not* always assumed in the works done in the context of computational social sciences – and in computational complex sciences in general – explains the difference between the epistemological reflections of this community (which uses more and more M&S based on agents) and the epistemological reflections of the neighboring community concerned with multimodelling, DEVS formalisms and, more recently, the FMSA (FMS with Agents). In particular, our epistemological distinctions enable to explain why different epistemic statuses still can be attributed to their works with agent-based models and simulations by computational economists or sociologists, whereas these distinctions seem to have not much meaning in the other community.

In fact, this paper shows that these two communities will have to discuss more intensively in the coming years. In particular, it shows that the FMSA community will newly and explicitly have to deal with the "modeler's dependent knowledge" (*i.e.* with the modelling of his *epistemology*), once rejected (as non significant) in the first FMS approach.

Finally, it has been suggested that the challenge for the project of a formulation of a *universal automated modeller agent* (UAMA) has much to do with the search for a way to articulate the *internal and system theory approach* of levels of symbols and *the referentialist approach of the relations of denotation between symbols and external target objects*.

References

[Berlekey, 2008] – Berkeley I., What the … is a symbol?. *Minds and Machines*, 18, 2008, pp. 93-105.

[Berlekey, 2000] – Berkeley I., What the … is a subsymbol?, *Minds and Machines*, 10, 2000, pp. 1-14.

[Carruthers, 2006] – Carruthers P., *The Architecture of the Mind*, Oxford, Oxford University Press, 2006.

[Copeland, 2004] – Copeland B. J., Computation. *In:* Floridi L. (ed.), *The Blackwell Guide to the Philosophy of Computing and Information*, New York, Blackwell Publishing, 2004, pp. 3-17.

[Ferber, 1999] – Ferber J., *Multi-agent systems: An introduction to distributed artificial intelligence,* Harlow (UK), Addison-Wesley, 1999.

[Fischer, 1996] – Fischer O., Iconicity: A definition. *In:* Fischer O. (ed.), *Iconicity in Language and Literature*, Academic Website of the University of Amsterdam, maintained since 1996, http://home.hum.uva.nl/iconicity/.

[Fodor, 1975] – Fodor J., *The Language of Thought*, New York, Thomas Crowell Company, 1975.

[Frey, 1961] – Frey G., Symbolische und Ikonische Modelle. *In:* Freudenthal H. (ed.), *The concept and the role of the model in mathematics and natural and social sciences*, Dordrecht, Reidel Publ., 1961, pp. 89-97.

[Gilbert, 2008] – Gilbert N., *Agent-based Models*, Los Angeles, SAGE Publications, 2008.

[Goodman, 1981] – Goodman N., Routes of Reference. *Critical Inquiry*, Vol. 8, No. 1, Autumn 1981, pp. 121-132.

[Guala, 2009] – Guala F., Experimentation in Economics. *In:* Mäki U. (ed.), *Handbook of the Philosophy of Science*, Vol. 13: *Philosophy of Economics*, forthcoming (2009).

[Guala, 2003] – Guala F., Experimental Localism and External Validity. *Philosophy of Science*, 70, 2003, pp. 1195-1205.

[Guala, 2002] – Guala F., Models, Simulations, and Experiments. *In:* Magnani L. and Nersessian N. J. (eds.), *Model-Based Reasoning: Science, Technology, Values*, New York, Kluwer, 2002, pp. 59-74.

[Hartmann, 1996] – Hartmann S., The world as a process. *In:* Hegselmann R., Müller U. and Troitzsch K. (eds.), *Modelling and simulation in the social sciences from the philosophy of science point of view*, Kluwer, 1996, pp. 77-100.

[Hill, 2000] – Hill D. R. C., *Contribution à la modélisation de systèmes complexes, application à la simulation d'écosystèmes*, Mémoire d'habilitation à diriger des recherches, Université Blaise-Pascal, Clermont-Ferrand, 2000 (in french).

[Hill, 1996] – Hill D. R. C., *Object-Oriented Analysis and Simulation*, Addison-Wesley Longman, 1996.

[Humphreys, 2004] – Humphreys P., *Extending ourselves*, Oxford University Press, 2004.

[Klir and Elias, 1985] – Klir G. and Elias D., *Architecture of Systems Problems Solving*, New York, Plenum Press, 1985.

[Mäki, 2002] – Mäki U. (ed.), *Fact and Fiction in Economics*, Cambridge University Press, 2002.

[Minsky, 1965] – Minsky M., Matter, Mind and Models. *In: Proceedings of IFIP Congress*, 1965, pp. 45-49.

[Nänny and Fischer, 1999] – Nänny M. and Fischer O. (eds.), *Form Miming Meaning: Iconicity in Language and Literature*, Amsterdam / Philadelphia, Benjamins, 1999.

[Ören, 2005] – Ören T., Toward the Body of Knowledge of Modeling and Simulation. *In: Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, Orlando, Florida; paper 2025, 2005, pp. 1-19.

[Peck, 2004] – Peck S. L., Simulation as experiment: a philosophical reassessment for biological modeling. *Trends in Ecology & Evolution*, 19(10), 2004, pp. 530-534.

[Phan and Varenne, 2008] – Phan D. and Varenne F., Agent-Based Models and Simulations in Economics and Social Sciences: from conceptual exploration to distinct ways of experimenting. *In: Proc. of EPOS (Epistemological Perspectives on Simulation) – 3rd edition, October 2-3, 2008*, Lisbon, ISCTE, 2008, pp. 51-69.

[Phan *et al.*, 2007] – Phan D., Schmid A. F. and Varenne F., Epistemology in a nutshell: Theory, model, simulation and experiment. *In:* Phan D. and Amblard F. (eds.), *Agent Based Modelling and Simulations in the Human and Social Sciences*, Oxford, The Bardwell Press, 2007, pp. 357-392.

[Putnam, 1991] – Putnam H., *Representation and Reality*, Cambridge, The MIT Press, 1991.

[Putnam, 1960] – Putnam H., Minds and Machines. *In:* Hook D. (ed.), *Dimensions of Mind. A Symposium*, New York, 1960; reprinted *in*: Anderson A. R. (ed.), *Minds and Machines*, New York, Prentice-Hall, 1964, pp. 43-59.

[Recanati, 2008] – Recanati C., Hybrid Reasoning and the Future of Iconic Representations. *In: Artificial General Intelligence 2008 - The First AGI Conference*, Memphis, Tennessee, 2008, pp. 299-310.

[Smolensky, 1998] – Smolensky P., On the Proper Treatment of Connectionism. *The Behavioural and Brain Sciences*, 11, 1998, pp. 1-74.

[Sugden, 2002] – Sugden R., Credible Worlds: The Status of Theoretical Models in Economics. *In:* Mäki U. (ed.), *Fact and Fiction in Economics*, Cambridge University Press, 2002, pp. 107-136.

[Varenne, 2009b] – Varenne F., Models and Simulations in the Historical Emergence of the Science of Complexity. *In:* Aziz-Alaoui M. A. and Bertelle C. (eds), *From System Complexity to Emergent Properties*, Berlin, Springer 2009, pp. 3-21.

[Varenne, 2009a] – Varenne F., *Qu'est-ce que l'informatique ?*, Paris, Vrin, 2009.

[Varenne, 2007] – Varenne F., *Du modèle à la simulation informatique*, Paris, Vrin, 2007.

[Varenne, 2001] – Varenne F., What does a computer simulation prove?. *In:* Giambiasi N. and Frydman C. (eds.), *Simulation in industry – ESS 2001, Proc. of the 13th European Simulation Symposium*, Ghent, SCS Europe Bvba, 2001, pp. 549-554.

[Winsberg, 2009] – Winsberg E., *A Tale of Two Methods. Synthese*, forthcoming (2009).

[Yilmaz *et al.*, 2006] – Yilmaz L., Ören T. and Aghaee N. G., Intelligent agents, simulation, and gaming. *Simulation & Gaming*, Vol. 37, No. 3, September 2006, pp. 339-349.

[Zeigler *et al.*, 2009] – Zeigler B. P., Muzy A. and Yilmaz L., Artificial Intelligence in Modeling and Simulation. *In:* Meyers R. A. (editor-in-chief), *Encyclopedia of Complexity and System Science*, Heidelberg, Springer, 2009, pp. 344-368.

[Zeigler *et al.*, 2000] – Zeigler B. P., Praehofer H. and Kim Tag G., *Theory of Modeling and Simulation. Integrating Discrete Event and Continuous Complex Dynamic Systems*, New York, Academic Press,2000 (2nd ed.).

Generative Multisimulation: Decision-Support under Uncertainty using Evolutionary Multimodels

Levent YILMAZ[1] & Bradley MITCHELL[1]

Abstract – *Generative Multisimulation (GMS) is a generative simulation methodology, which introduces a symbiotic adaptive decision support capability for systems with shifting, ill-defined, uncertain environments. Rather than rely on a single authoritative model, GMS explores an ensemble of plausible models, which are individually flawed but collectively provide more insight than would be possible otherwise. A case study based on a UAV team search and attack model is presented to illustrate the potential of GMS. Preliminary results demonstrate the potential of GMS to produce a large degree of exploratory behavior, followed by increased exploitative search behavior as the physical system unfolds.*

Keywords – Generative multisimulation, evolutionary multimodels, autonomic simulation, decision-support systems.

Introduction

Strategy problems are typically characterized by significant uncertainty [Davis and Bigelow, 1988]. Proper simulation-based decision support methodologies that facilitate making decisions in field settings could improve modeling Course of Actions (COAs), simulating them faster than real time,

1. Auburn University, 3116 Shelby Center, Auburn, AL, 36849, USA. Corresponding author: yilmaz@auburn.edu.

and then performing COA analysis to improve robustness and resilience of decisions. Exploring the effectiveness of alternative COAs requires dynamic updating, branching, and simultaneous execution of simulations, potentially at different levels of resolution. Dynamic updating of simulation models is a key requirement for using simulation as a tool to improve systems in which information only becomes available once the system is in progress [Yilmaz, 2004; Yilmaz *et al.*, 2007]. In these types of systems, the initial conditions provide little or no insight into how the system may develop over time. Emergent behavior that arises dynamically is a primary source of information in these systems. Exploiting this information to enable robust decision-making in a timely manner requires the ability to observe the system in real-time and adapt useful characteristics for the system with as little computational effort as possible.

The notion of self-organization provides a useful metaphor for the design and development of next generation simulation infrastructures. The term self-organization has been used in different areas with different meanings, as is cybernetics, thermodynamics, biology, mathematics, computing, and information theory. A system can be described as self-organizing if its elements interact to dynamically achieve a globally desired behavior. The behavior is not imposed and determined in a top-down manner, rather it is achieved autonomously as elements of the system adapt and evolve to changing environmental conditions. Similarly, autonomic computing paradigm relies on perception and understanding of the environmental context to self-manage the computational infrastructure so that optimal operating conditions can be attained.

Symbiotic Simulation (S2) [Fujimoto *et al.*, 2002] involves the use of simulation systems that are synchronized with the physical systems to enable mutually beneficial adaptation. In S2, simulation outputs are examined and used to determine how the physical system may be optimized. Similarly, measurements from the physical system are used to validate the simulation. When uncertainty in the physical system is present, multiple what-if simulation experiments can be helpful in adjusting the physical system. However, since the number of what-if experiments that may be performed is limited by both computational and real-time constraints, the ability to conduct an efficient search of the model space is essential.

Generative Multisimulation (GMS) is intended as a self-organizing generative S2 technique appropriate for physical systems characterized by distributed, dynamic and uncertain conditions. It is heavily inspired by the fields of Multisimulation [Yilmaz, 2007], Exploratory Analysis [Davis and Bigelow, 2000], and Exploratory Modeling [Bankes, 1993], which involve the use of an ensemble of plausible models to provide insight in the absence of a single authoritative model. The salient feature of GMS is the use of evolutionary computation in terms of a Genetic Algorithm (GA) to evolve the model ensemble in response to changes in the physical system. With this feature, it is conjectured that an effective search of an uncertain model space would be possible, thus permitting synchronization with the physical system.

Motivation

Experimenting with evolutionary and/or contingency models in real-time on demand would be critical for decision support in unstructured problems with the characteristics of (1) deep uncertainty, (2)≈dynamic environments, and (3) shifting, ill-defined, and competing goals. The major challenges pertaining to decision-making in such asymmetric and irregular environments include the following [Yilmaz, 2007]:

- For most realistic problems, the nature of the problem changes as the simulation unfolds. Initial parameters, as well as models can be irrelevant under emergent conditions. Relevant models need to be identified and instantiated to continue exploration. Manual exploration is not cost effective and realistic within a large problem state space.
- Our knowledge about the problem being studied may not be captured by any single model or experiment. Instead, the available knowledge is viewed as being contained in the collection of all possible modeling experiments that are plausible given what is known and what is learned.
- Dealing with uncertainty is paramount to analyzing complex evolving phenomena. Adaptivity in simulations and scenarios is necessary to deal with emergent conditions for evolving systems in a flexible manner.

Strategy: The Basis for GMS

Addressing the above challenges requires a simulation system to cope with an unpredictable environment autonomously through flexibility and robustness. Flexibility can be achieved using different but closely related approaches:

- **Adaptation**. The system changes its behavior to cope with the change through learning and adaptation.

- **Anticipation**. An anticipatory system is a system whose next state depends on its current state as well as the current image(s) of its future state(s).

Coping with robustness and resiliency under changing conditions, on the other hand, requires capability to function in the face of perturbations. Proper mechanisms for dealing with uncertainty are abundant in nature. For instance, arthropod (insect) eyes which are called compound eyes, are made up of repeating units, the ommatidia, each of which functions as a separate visual receptor. Each ommatidium is pointed at just a single area in space and contributes information about only one small area in the field of view. The compound eye is excellent at detecting motion. As an object moves across the visual field, ommatidia are progressively turned on and off. Because of the resulting "flicker effect", insects respond far better to moving objects (e.g., situations) than stationary ones.

Similarly, when model excursions are viewed collectively, the behavior of plausible models can be informative despite the flaws of each individual model. Because of the presence of uncertainty, there may be many plausible models that could represent a system [Bankes, 1998]. Similarly, knowledge of the system constrains the set of plausible models. Multisimulation, Exploratory Analysis [Davis and Bigelow, 1999] and Exploratory Modeling experiment with ensembles of models as experimentation with a single plausible model would be just as likely deceptive as informative. Among a set of plausible models, variation occurs according to input uncertainty and structural uncertainty.

Decision-support under Uncertainty using GMS

GMS supports decision making by conducting a search through a potentially infinite space of models. The search works by evolving a set of potential system configurations for a dynamic set of environmental conditions. In this paper, we explore the following problems:

- What are different forms of uncertainty and how can they be modeled in such a way to facilitate exploration of alternatives and variation of configurations in an efficient manner?
- How can the key principles of CAS evolution be leveraged to design the evolution process of the decision support system so as to attain robustness under emergent order and unpredictable future state of the physical system?

Systems characterized by non-linear interactions among diverse agents often exhibit emergent behavior that may be very different from what the initial conditions of these systems would suggest. Traditional simulation techniques that rely on accurate knowledge of these conditions typically fail in these cases. The goal of GMS is to enable robust decision making in real-time for these problems. Rather than rely on a single authoritative model, GMS explores an ensemble of plausible models, which are individually flawed but collectively provide more insight than would be possible otherwise. The insights derived from the model ensemble are used to improve the performance of the system under study. Likewise, as the system develops, observations of emerging conditions can be used to improve exploration of the model ensemble. In essence, a useful co-evolution between the physical system and GMS occurs.

1. Generative Multisimulation: Methodology and Implementation

GMS behaves/evolves according to three key principles: order is emergent as opposed to predetermined [Zeigler, 1989], the system's history is irreversible, and the system's future is often unpredictable [Holland, 1975]. Agents are semi-autonomous units that seek to maximize some measure of goodness, or fitness, by evolving over time. Agents scan their environment and develop schema representing interpretive and action rules. Existing agent and model schema undergoes three types of change: first order change, where action is taken in order to adapt the observation to the existing schema; second order change, where there is purposeful change in the schema in order to better fit observations; and third order change, where a schema survives or dies because of the survival or death of its corresponding fitness. Schema changes through random mutation. Schema change generally has the effect of making the model ensembles more robust (it can perform in light of increasing variation or variety), more reliable (it can perform more predictably), or grow in requisite variety (in can adapt to a wider range of conditions). The fitness of the model ensemble is a complex aggregate of many factors, both local and global. The general health or fitness of the agents within a single model determines what the probability of change will be. Optimization of local fitness allows differentiation and novelty/diversity; global optimization enhances the coherence of GMS and induces long term memory. In general the probability of second order schema change is a nonlinear function of the fitness value.

1.1. Hybrid Exploration

In order to efficiently search a potentially infinite number of plausible models, GMS uses a hybrid exploration technique. As shown in Figure 1, uncontrollable inputs and controllable inputs are handled with an input analysis module and an output analysis module respectively. Measurements of the physical system's behavior are used to hypothesize distributions for uncontrollable inputs. As more details of the physical system's environment become known, the fidelity of these distributions with the actual values of the physical system should improve. Controllable input factors representing the configuration of the physical system are evolved using a genetic algorithm. A set of controllable inputs that completely describes a potential system configuration can be thought of as an individual. The controllable factors therefore, are considered to be the decision variables of a given problem while the uncontrollable factors determine the shape of a dynamic fitness landscape. An evolutionary algorithm enables the adaptation of individuals through a process of natural selection, with better performing individuals being more likely to survive and ultimately be used as the configuration settings for agents within the physical system itself.

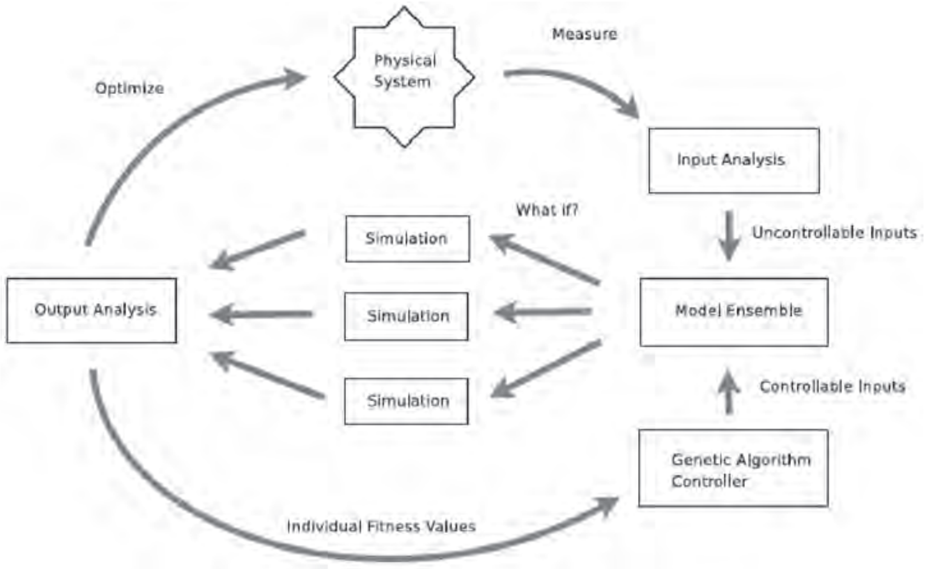


Figure 1. Symbiotic Simulation.

1.2. Partial Model Ensembles

Once a distribution has been hypothesized for each uncontrollable factor in the physical system, and the associated parameters for those distributions have been estimated, each factor is sampled multiple times. These samples are stored in a 2-dimensional array referred to as a Partial Model Ensemble (PME). Variates for each sampled distribution are associated on a per-sample basis and placed into the same row of an array. Each column of the array stores variates from a single input factor, and each row of the array is considered to be a partial plausible model of the physical system. Later, these partial models are combined with potential system configurations to create fully specified plausible models for simulation. To hypothesize distributions for uncontrollable inputs and estimate their parameters, a means of observing the physical system is required. As real-time symbiotic simulation is performed, observations of the physical system enable more accurate estimates of the input distribution parameters. With these improvements, the space of plausible models shrinks, allowing the system to simulate fewer models in greater detail.

1.3. Combined Model Ensembles

In GMS, a Combined Model Ensemble (CME) is a specification for conducting a series of simulation experiments involving a single individual from a population of potential system configurations. The goal of these experiments is to test an individual in multiple possible environments. Each simulation experiment examines the individual in the context of a set of uncontrollable factors representing a single possible environment. The fitness of the individual for a particular environment is obtained as an output from a simulation experiment. Resulting fitness values from all experiments with the individual in each respective environment are then averaged together to obtain an overall fitness for the individual, which is used to determine its probability for reproduction and survival within the genetic search.

The layout of a CME is shown in Table 1. The possible environments with which the individual is to be tested are determined by a PME of Uncontrollable Factors making up the left hand side of the table. Copies of the individual are paired with each row of the PME. A single row of the CME thus includes both a sampled set of values from the uncontrollable factor distributions and a copy of the individual which specifies the settings of the controllable model factors. Taken together, these two sets of elements form a single plausible model described by the entire row of values in the CME.

Uncontrollable Factors				Controllable Factors			
X_{11}	X_{12}	$\cdots$	X_{1m}	G_1	G_2	$\cdots$	G_l
X_{21}	X_{22}	$\cdots$	X_{2m}	G_1	G_2	$\cdots$	G_l
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
X_{n1}	X_{n2}	$\cdots$	X_{nm}	G_1	G_2	$\cdots$	G_l

Table 1. Combined Model Ensemble- A Combined Model Ensemble of *n* models is composed of a Partial Model Ensemble of *m* sampled variables, *X*, and an individual of *l* genes, *G*. Each row represents a single model for which one or more simulation replications should be performed. Note that the same individual is used in each model.

Note that for a given CME, the same individual is paired with each row of the PME. Furthermore, the same PME is combined with each individual in the population to create a set of unique CMEs, one for each individual. Since the same PME is used in each CME, each individual is evaluated using the same environmental conditions. In essence, the uncontrollable factor settings of the PME, which change each time a new PME is created, become a dynamic fitness landscape.

1.4. Local Adaptation: Particle Swarm Optimizer

Self-organization is a kind of aggregate behavior that is often associated with Complex Adaptive Systems (CAS). Self-organization as a property of an engineered system has been described as being one in which individual agents or units respond to local stimuli to achieve through a division of labor the efficient performance of some task. The collective efficiency of task performance must be greater than what could be accomplished individually. Additionally, self-organization involves the creation of an equilibrium state that arises from the local interactions of agents. This equilibrium may be achieved through either competition or cooperation.

In GMS, Particle Swarm Optimization (PSO) is used for implementing the learning element of adaptive agents. A broad introduction of PSO can be found in [Kennedy and Eberhart, 2001]. We use PSO as a continuous numeric optimization technique in which a potential solution to a problem is characterized as a point in some n-dimensional space, with the number of dimensions being equal to the number of decision variables. As the name suggests, PSO uses a population of potential solutions. These solutions "fly" through the problem space over time. As each particle moves through the problem space, it records the best solution, *pbest* that it has found so far as well as the best solution, *gbest* discovered by the other members of the swarm. Particles tend to gravitate toward these two positions over time as they search for better solutions [Bratton and Kennedy, 2007].

1.5. Global Adaptation: Genetic Search of Potential System Configurations

GMS evolves potential system configurations for use within the physical system. Unlike the creation of a PME, however, a search of the space of potential system configurations requires the exploitation of a different type of system information, namely the performance characteristics of the configuration. Evolutionary Algorithms exploit this type of information to improve a search as they are well suited for optimization problems. Among Evolutionary Algorithms, Genetic Algorithms have a feature useful to GMS, which is the distinction between genotypic and phenotypic representation. A genetic encoding of the characteristics of a potential system configuration allows the same system to be interpreted in different ways and in varying levels of detail.

The structure of a typical GA is shown in Algorithm 1. An initial population of *m* individuals is randomly created and the fitness values of the resulting individuals are evaluated. The algorithm then enters a loop in which successive generations of individuals are evolved. An inner loop continues until a group of *n* children are created. Two individuals are chosen to be parents. Crossover combines the characteristics of both parents to create a new child. A mutation operator is then applied which may modify the child with characteristics not possessed by either parent. The fitness of the new child is then evaluated.

Algorithm 1: A genetic algorithm

```
 1: pop[$m$] ⇐ createInitialPopulation ( )
 2: for $i$ = 1 to $m$ do
 3:         pop[$i$] ⇐ calculateFitness ( pop[$i$] )
 4: end for
 5: repeat
 6:         children[$n$]
 7:         for $i$ = 1 to $n$ do
 8:                 parents[2] ⇐ selectParents( pop )
 9:                 children[$i$] ⇐ crossover( parents )
10:                 children[$i$] ⇐ mutatation( children[$i$] )
11:                 children[$i$] ⇐ calculateFitness( children[i] )
12:         end for
13:         pop[$m$]  ⇐ selectSurvivors( pop, children )
14: until termination = true
```

Once the children have been created, the total population of individuals is then reduced back to *m*. Depending on the design of the GA, survivor selection may or may not involve direct competition between parents and children. The evolutionary process continues until some termination condition (such as a fixed number of fitness evaluations) is reached.

1.6. GMS Component Architecture

GMS implementation, designed to operate in mission critical environments, is based on an independent component architecture in which the individual components of the system could execute in parallel and communicate via message passing [Braude, 2004]. This could be especially useful for entities operating in the physical environment that would likely not have the hardware resources to process simulations. Rather, these entities would only need some means of measuring the physical system and sending those measurements to the components running the simulations. Figure 2 illustrates the essential components of the GMS system. Observations from the physical system are passed to an Input Exploration Component (IEC) responsible for conducting input analysis and selecting appropriate distributions for the uncontrollable model factors. Samples from these distributions are then used to create a PME, a copy of which is integrated with each individual to form one CME for each Agent-based Simulation Process (ASP).

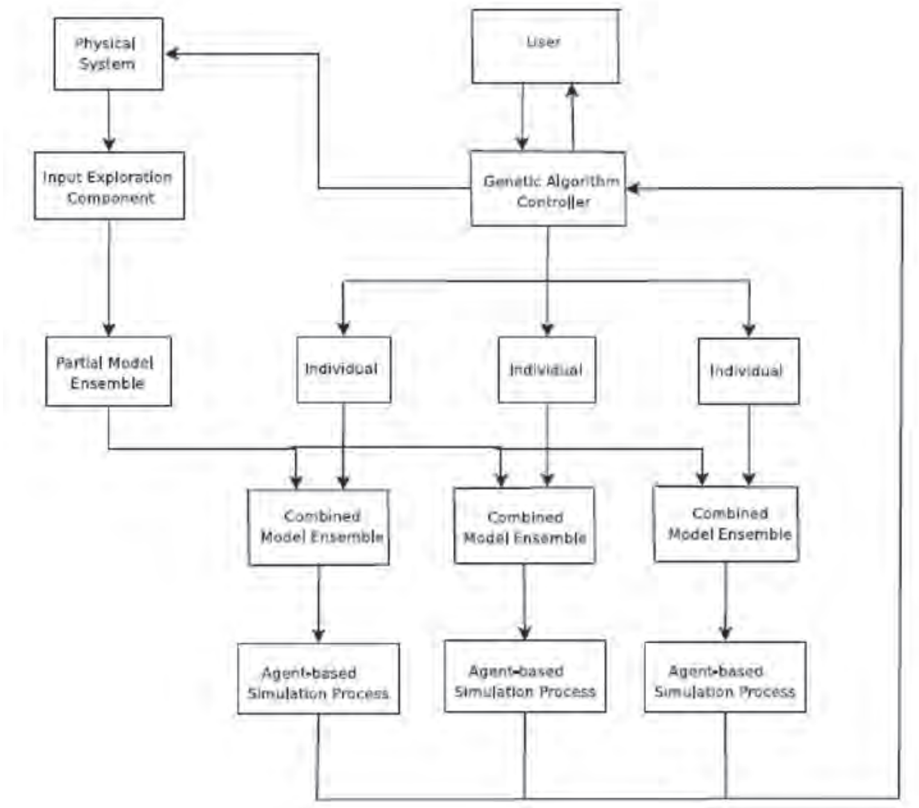

Physical System
User
Input Exploration Component
Genetic Algorithm Controller
Partial Model Ensemble
Individual
Individual
Individual
Combined Model Ensemble
Combined Model Ensemble
Combined Model Ensemble
Agent-based Simulation Process
Agent-based Simulation Process
Agent-based Simulation Process

Figure 2. Components of the GMS Framework

The Genetic Algorithm Controller (GAC) is responsible for evolving the population of individuals which are used to form the CMEs. and a number of Agent Simulation Instances (ASPs) to simulate CMEs in parallel. Ideally, each ASP is mapped to one or more CPU cores. If the population used by the GAC is large, or if hardware resources are limited, multiple ASPs can run on a single core. If there is an excess of hardware, a GMS implementation should be capable of offloading models within the CME to multiple CPU cores. Outputs from the ASPs take the form of objective fitness values of the individuals that have been averaged across all of the replications specified by the CME. These are passed to the GAC which then assigns the fitness values to the simulated individuals before continuing execution of the GA. Ideally, it should be possible for a user to interact with the GAC in real time to examine the individuals generated and possibly seed new configurations to the population.

The sequence of operations that occur within the main execution loop of a GMS implementation are shown in a UML sequence diagram in Figure 3. Active components and processes, displayed as boxes across the top of the diagram include the Genetic Algorithm Controller, Agent-based Simulation Processes (of which several run simultaneously), an Input Exploration Component and the Physical System. An initial message is passed from the GAC to the IEC requesting a new Partial Model Ensemble. The IEC takes observations from the physical system and uses them to produce uncontrollable factor settings which are incorporated into the PME. The PME is then passed to the GAC which uses them to create a CME for each ASP.

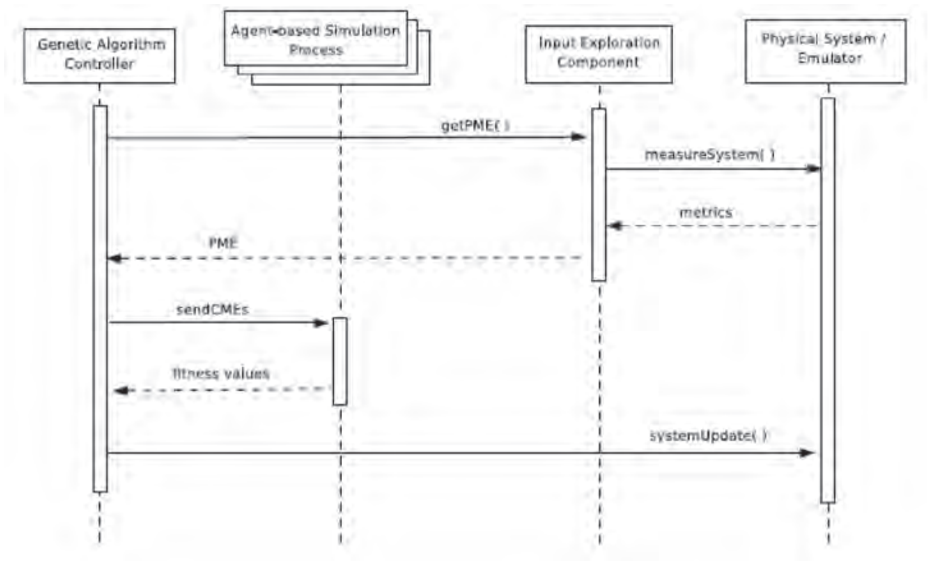

Genetic Algorithm Controller
Agent-based Simulation Process
Input Exploration Component
Physical System / Emulator
getPME( )
measureSystem( )
metrics
PME
sendCMEs
fitness values
systemUpdate( )

Figure 3. Component Interaction in GMS.

When created, the CMEs are passed to the ASPs and simulated in parallel. Simulation results are examined on a per CME basis so that fitness can be assigned to a given individual based on its performance against the environmental settings specified in the PME. After fitness values have been assigned to individuals, the GAC evolves the population. Individuals with higher fitness are given preference for producing offspring, which are created using genetic operators of crossover and mutation. Due to the need to maintain a naturally parallel structure for the algorithm, large numbers of children should be generated before selecting survivors.

In addition to evolving the existing population, the GAC also selects the most fit individual of each generation to update the physical system's configuration. This continual process of dynamically updating the physical system helps ensure that the physical system is responding correctly to observed changes in the environment.

2. Case Study: UAV Search and Attack Scenario

To perform an assessment of the potential of the GMS methodology, a model based on the features of Complex Adaptive Systems was integrated into the parallel application. The field of autonomous UAV cooperation provides a natural environment from which to create such a model. Toward this end, an agent-based model of an autonomous UAV team in a Search and Attack mission was developed. The UAVs in this model interact through local communication only. There is no global system of coordination or prior intelligence of targets. A rule-based approach for UAV movement is implemented. This set of movement rules, which is based on general knowledge of the problem domain, results in a robust performance element for UAVs capable of both independent and cooperative actions. These rules are implementations of the steering behaviors described in [Reynolds, 1999]. Similar implementations of these steering behaviors have been featured in a number of autonomous UAV studies including [Price, 2006] and [Crowther, 2004].

The Search and Attack scenario takes place in a 2-dimensional space of equal dimensions. UAVs begin play from a base located at the center of the map. Targets are distributed randomly across the map and their positions are initially unknown to the UAVs. Once they are launched, the UAVs must find and destroy all of the targets as quickly as possible. Figure 4 depicts the opening simulation steps in which UAVs are in the process of launching from the base (represented by a gray hexagon) and sweeping the map. Targets (represented by gray triangles) within the sensor envelope of a UAV have a chance of being discovered while those outside remain hidden.

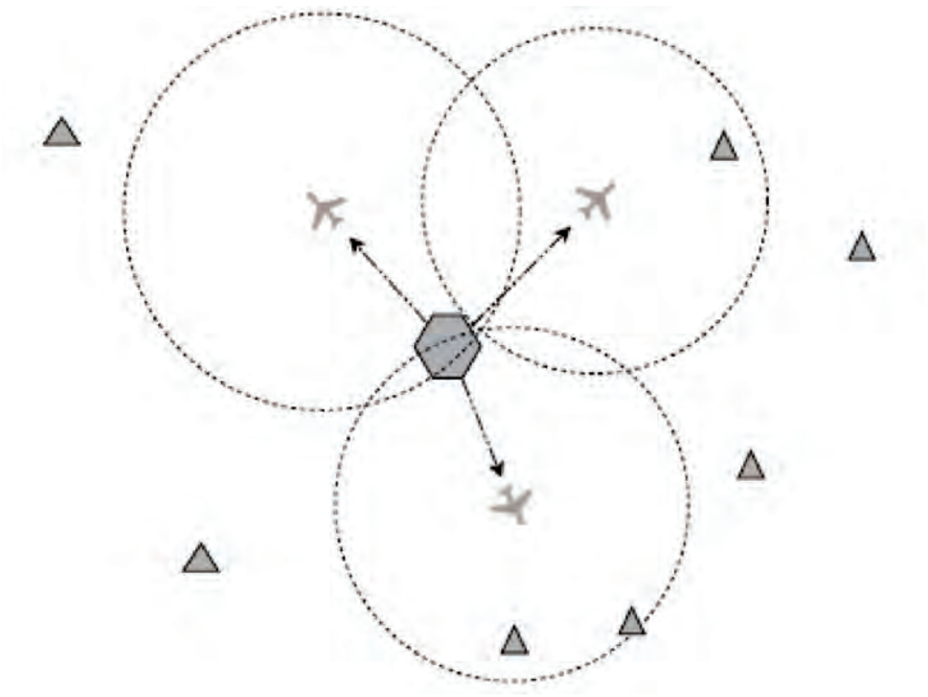


Figure 4. The Search and Attack scenario shortly after simulation start. UAVs depart from the centrally located airbase (represented by a hexagon with uniformly random initial movement vectors. The sensor envelopes of individual UAVs are represented by dotted circles.

Progression of the model is simulated through time-stepped execution, dividing simulation time into a number of equal size increments. During a simulation step, each UAV may act. The results of these actions are updated synchronously and may affect actions performed by other UAVs in the same time step.

3. Experiments with the Parallel GMS System

The goal of computational experimentation for this thesis was to provide an initial understanding of the potential of GMS as an S2 methodology.

Toward this end, it was hoped that successful experimentation would help to answer a central question:

- Can the use of symbiotic simulation through ensembles of plausible models improve a physical system's performance?}

3.1. Establishing Face Validity

One interesting phenomenon observed in some of the experiments is that the behavior of the UAVs' average Cooperation values due to the influence of the distributed PSO. Figure 5 illustrates a single replication from one set of experiments, which was not included in the original experimental design, but appears instructive. This case involved a high Communication Range setting of 60 and one leader UAV. As discussed earlier, the distributed PSO evaluates the PSO objective function and updates its decision variables (including Cooperation and Base Distance) once per 50 time steps. This period is referred to as the sampling interval, shown across the bottom of the plot. Thus, at sampling interval 10, 500 time steps have elapsed in the simulation. This plot tracks the change in the average of all Cooperation values within the UAV team across a single model replication. In this case, by about the 10th sampling interval, the distributed PSO had evolved the team away from cooperative behavior.

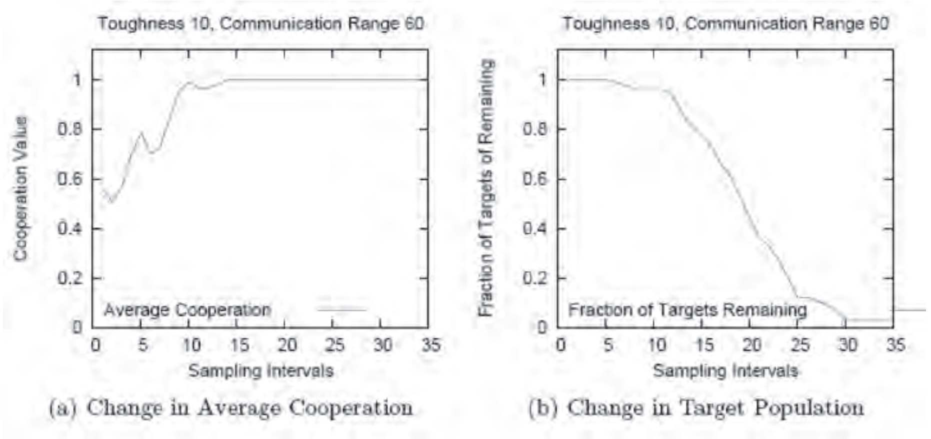


Figure 5. Effect of Cooperation on Target Population with High Communication Range.

The behavior shown in Figure 5 contrasts with what occurs when Communication Range is set to a low value. Figure 6 depicts that when Communication Range is set to 6, cooperation is more likely to occur. In this case, the average value of the PSO Cooperation value stayed relatively close to 0.5. The transitioning behavior to equilibrium in both cases is typical of self-organizing systems and helped us instill confidence in the learning mechanism.

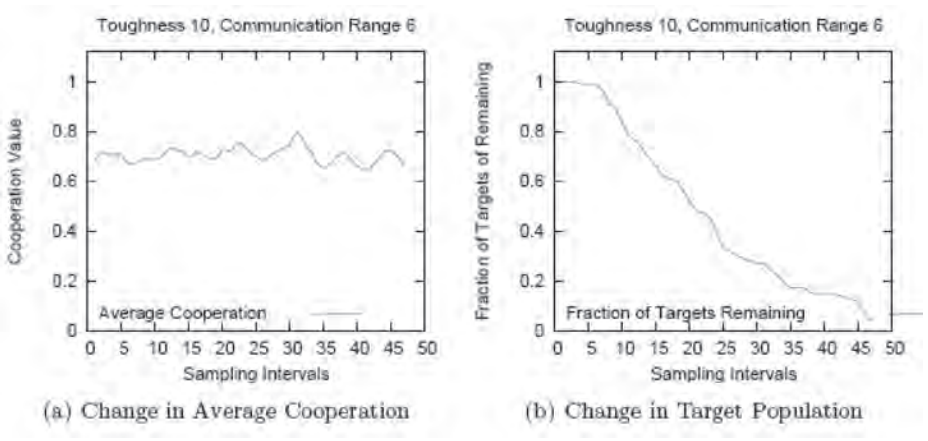


Figure 6. Effect of Cooperation on Target Population with Low Communication Range.

3.2. Emulator Objective and Individual Fitness

Experimentation with the Parallel GMS Application involved the use of a system emulator, rather than an actual physical system. This emulator is itself a simulation with the same structural model of autonomous UAV behavior used by the ASPs. It only differs in terms of the values used for its input factors and in that its controllable factor settings are not fixed for the duration of its execution. Similar to the model replications used in the stand-alone mode for face validity, the performance objective of the Emulator is to minimize the number of time steps required to eliminate all targets. Therefore, to synchronize the performance of the simulations executed in the ASPs with that of the emulator, the fitness of an individual is defined to be the average number of time steps required to eliminate all targets across all replications specified by that individual's CME.

The emulator is an ordinary UAV model simulation which receives controllable factor updates from GMS. Therefore, it was possible to examine the performance of the Parallel GMS Application by running an emulator with identical uncontrollable factor settings used in the stand-alone mode. Since the emulator and stand-alone models use the same model structure, their performance can be compared as long their uncontrollable factors and fixed model settings are identical. Our position is that the emulator's system would have an advantage over the system in a stand alone model. The controllable factors of Leader flags and Cooperation Thresholds can be modified by GMS to the benefit of the emulator, whereas the system in the stand alone model has controllable factor settings that are determined in advance and fixed for the duration of its execution. In order to compare the performance of the emulator with GMS against a stand-alone model, all fixed model settings such as Weapon Range, Average Weapon Effect, Weapon Effect Standard Deviation, Average Target Distance, and Target Standard Deviation were set to the identical settings used in the experiments described in the previous, as these settings were held constant across all treatments in that group. Furthermore, a subset of the treatments corresponding to one pair of settings of the uncontrollable factors is selected.

3.3. GA Design

One intent with GMS is for the Genetic Algorithm Controller to produce a large degree of exploratory behavior initially, followed by increased exploitative search behavior as the physical system changes. With this in mind, proportional selection was chosen as a parent selection operator for these experiments. The variation operators used for these experiments included one-point crossover, which obtains a single cut point that determines how the chromosomes will be divided and recombined to produce children. Each individual represented 20 UAVs and a non-overlapping survival model was selected so that all children survive while parents automatically die. This type of GA, which balances a relatively low selection pressure with low rates of variation from one-point crossover and low mutation, is suitable for encouraging exploration of controllable factors early in the physical system's development.

3.4. Emulator Results

The major limitation of these experiments was that the IEC was not yet implemented. Therefore, certain assumptions were made regarding emulator / physical system measurement and PME generation. The work of the IEC was simulated within the experiments. At run-time, the simulated IEC is initialized with uniform random distributions for three uncontrollable factors: Toughness, Communication Range, and Target Max Visibility. Note that Target Max Visibility was held constant across all treatments. The true values used for these factors within the emulator were Toughness 10, Communication Range 6, and Maximum Target Visibility 5.

The parameterized distributions within the simulated IEC for the uncontrollable factors are arbitrarily selected to be uniform distributions of length 30 about each factor's true setting (truncated by zero as a minimum parameter). This included U(0, 25), for Toughness, U(0, 20) for Maximum Target Visibility, and U(0, 21) for Communication Range. At each iteration, a PME of 30 rows and 3 columns (one for each uncontrollable factor) is created. When combined with an individual, each CME therefore had 30 rows and 43 columns (3 columns for the uncontrollable factors and 40 columns representing 2 genes for each of the 20 UAVs in an individual). Each row, representing a single parameterized model, runs for one replication resulting in 30 replications for each CME. A population of 20 individuals mapped to 20 ASPs was used. This resulted in 600 model runs during each iteration of the main loop. The emulator replications included on average, 16≈generations. Thus, on average, approximately 9600 model replications are performed by the ASPs for each replication of the emulator. More replications for each row of a CME and a larger population would have been desirable but were not used due to project time constraints.

For the experiment, 30 replications of the GMS Parallel Application on the Altix Supercomputer are run. As can be seen in Table 2, the performance improvement of the emulator is noticeable in terms of the number of time steps for mission completion compared to the best performing UAV team configuration among the stand-alone model tests.

Experiment	Average Time Steps	Standard Dev.
Emulator with SAMS	1597,5	164,34
Stand-alone mode 1	1777,24	129,68

Table 2. The results of the Parallel Application compared to the best performing Stand-alone Model. The Parallel Application provides significantly improved performance on average, but with higher variance.

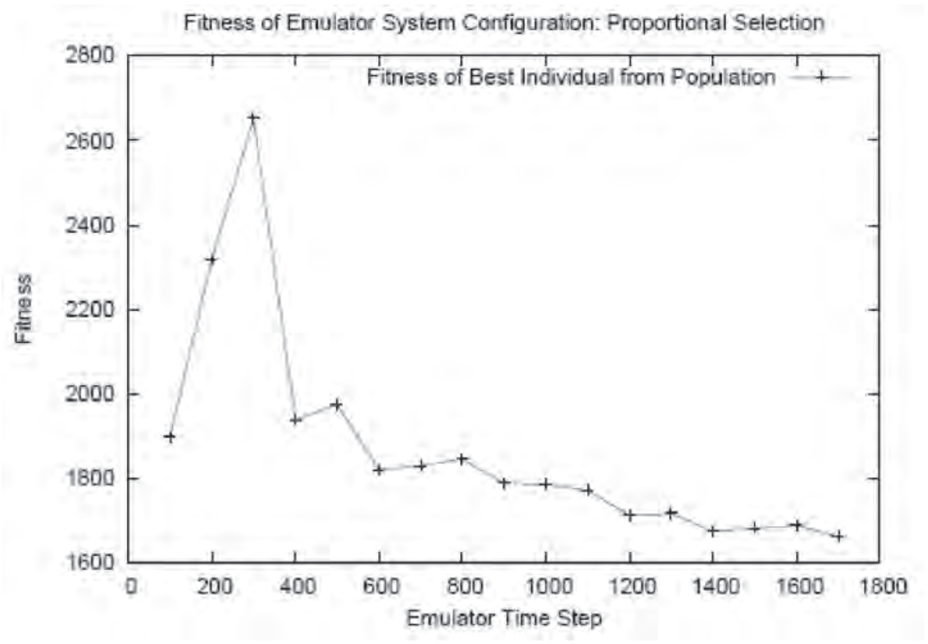


Figure 7. Fitness of Emulator System Configuration.

A possible explanation for the increased standard deviation in GMS may be that the number of replications performed for each CME were simply insufficient to properly assess the fitness of the CME's respective individual. This might also explain the initial decrease in fitness in Figure 7 as the initial iterations of the main loop involve significant uncertainty in the uncontrollable factors. While these experiments seem encouraging, more experimentation is needed to explore this issue in detail. In particular, experiments with larger numbers of model replications need to be performed. Secondly, the possibilities for modification of Genetic Algorithm used by GMS have barely been scratched. Much research is needed to identify what design features of a GA (or EA) contribute most to improved performance.

Conclusions

This research examines the need for dynamic model updating for Symbiotic Simulation of systems involving interacting agents with complex, non-linear behavior. Our position is that these systems may not be effectively studied with traditional simulation techniques that rely on valid, authoritative models of the physical system. Instead, techniques such as Multisimulation and Exploratory Analysis, which experiment with an ensemble of plausible models are developed to deal with these problems.

The proposed Symbiotic Adaptive Multisimulation approach involves a Hybrid Exploration strategy to study an ensemble of plausible models. When parameterized to account for input uncertainty in controllable and uncontrollable factors, GMS is able to dynamically update a system emulator resulting in improved performance. This benefit is realized with the help of a Genetic Algorithm that evolves potential system configurations over the lifetime of the system emulator and which can be used to update the emulator. These updates consist of adaptive strategies that are passed on to the agents operating within the emulator. This initial study of GMS as a simulation methodology has shown encouraging results, but has also left many problems unsolved. In particular, further experimentation with larger numbers of model replications is required. Also, experimentation with additional GA designs can be performed to understand the features that make an Evolutionary Algorithm suitable for GMS. The understanding gained could provide further improvements to the methodology in terms of speed and robustness. Other significant problems that remain are the incorporation of appropriate Multiresolution Modeling, input analysis for estimating uncontrollable factor distributions, and handling of structural uncertainty. Given these challenges, Autonomic Multisimulation appears to be a rich opportunity for further study.

References

[Bankes, 1998] – Bankes S., Policy analysis for complex and uncertain systems through computational experiments. *In: Proceedings of the 1998 IEEE Aerospace Conference*, 1998. pp. 125-132.

[Bankes, 1993] – Bankes S., Exploratory modeling for policy analysis. *Operations Research*, Vol. 43, No. 1, 1993, pp. 435-449.

[Bratton and Kennedy, 2007] – Bratton D. and Kennedy J., Defining a standard for particle swarm optimization. *In: Proceedings of the 2007 IEEE Swarm Intelligence Symposium*, Honolulu, HI: IEEE, 2007, pp. 120-127.

[Braude, 2004] – Braude J. E., *Software Design: From Programming to Architecture*, John Wiley & Sons, 2004.

[Crowther, 2004] – Crowther B., Flocking of autonomous unmanned air vehicles. *Aeronautical Journal*, 2004. Vol. 107, No. 10, pp. 111-124.

[Davis and Bigelow, 2000] – Davis K. P. and Bigelow J. H., Exploratory analysis enabled by multiresolution, multiperspective modeling. *In: Proceedings of the 2000 Winter Simulation Conference*, 2000, pp. 127-134.

[Davis and Bigelow, 1999] – Davis K. P. and Bigelow J. H., Experiments in multiresolution modeling (mrm), *RAND Technical Report*, 1999.

[Davis and Bigelow, 1988] – Davis K. P. and Bigelow J. H., The role of uncertainty in assessing the NATO-PACT central region balance. *RAND N-2839*, The RAND Corporation, Santa Monica (CA), 1988.

[Fujimoto *et al.*, 2002] –Fujimoto M. R., Lunceford D., Page E. and Uhrmacher A. M.. *Grand challenges for modeling and simulation*. Dagstuhl report, 2002.

[Holland, 1975] – Holland H. J., *Adaptation in Natural and Artificial Systems*, Ann Arbor (MI), The University of Michigan Press, 1975.

[Kennedy and Eberhart, 2001] – Kennedy J. and Eberhart C. R. C., *Swarm Intelligence*, San Francisco (CA), Morgan Kaufmann Publishers, 2001.

[Price, 2006] – Price C. I., *Evolving self-organized behavior for homogeneous and heterogeneous uav or ucav swarms*, Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH, 2006.

[Reynolds, 1999] – Reynolds C., Steering behaviors for autonomous characters. (Retrieved on October 31, 2008 from http://www.red3d.com/cwr/papers/1999/gdc99steer.html), 1999.

[Yilmaz, 2007] – Yilmaz L., Toward next generation simulation-based computational tools for conflict and peace studies. *Social Science Computer Review*, Vol. 25, No 1, 2007, pp. 48–60.

Exploiting Spatial-temporal Heterogeneity for Agent-based Simulation of Pedestrian Crowd Behavior

Fasheng QIU[1], Xiaolin HU[1]

Abstract – *Agent-based simulation is widely used to simulate pedestrian crowd behavior. These simulations typically are implemented in a discrete time manner, where each agent decides its movement in every time step, independent of the fact that agents may move in different speeds. The non-uniform movements of agents result in a crowd system's spatial and temporal heterogeneity, which can be better exploited using a discrete event model where an agent's decision making is triggered by the changes of its external environment and/or its internal states. Motivated by the above observation, this paper systematically studies the discrete time and discrete event models of agent-based crowd behavior simulation and compares their performance results. The experiment results show that the discrete event model is able to track the crowd system's "activities" in both space and time, and thus leads to more computation efficient simulations. This work builds a ground for performance analysis for large scale agent-based crowd behavior simulations.*

Keywords – Crowd behavior simulation, Spatial-temporal heterogeneity, space resolution, discrete event modeling, discrete time modeling.

1. Department of Computer Science, Georgia State University, 34 Peachtree Street, Suite 1450, Atlanta, GA 30303.

Introduction

Agent based simulation has been widely used in simulating complex social phenomena [Davidsson, 2000], like the crowd behaviors. Many pedestrian crowd models have been developed, see, e.g., [Helbing *et al.*, 2002; Qiu and Hu, 2010]. All of them are discrete time models and featured with a virtual environment and multiple pedestrian agents. The simulations proceed in a discrete time manner, where each agent performs a decision making to decide its next movement in each time step. Such a discrete time approach makes it difficult to exploit the crowd system's spatial and temporal heterogeneity resulting from agents' non-uniform movements. To give an example, consider two agents situated in a large environment where one agent moves 10 times slower than the other. In a discrete time simulation, both agents make a movement decision in every time step. This is independent of the agent's speed. However, since one agent moves much slower than the other, intuitively one would think that the slower agent does not need to make movement decisions as frequently as the fast one. Thus a more computation efficient way of simulating the two agents is to allow the fast agent to make movement decisions more frequently and the slow agent to make movement decisions less frequently. In pedestrian crowd behavior simulations with realistic human-like behaviors, agents typically have non-uniform movements due to different individual characteristics such as moving speed, personality, the psychological states (e.g., panic, non-panic), and other factors. These non-uniform movements result in spatial and temporal heterogeneity in terms of agents' movement and decision making. Thus it is desirable to explore such heterogeneity for more computation efficient simulations.

In this paper, we present a discrete event approach to model and simulate pedestrian crowd and compare it with the discrete time model. The discrete event model uses a concept of "space resolution", which defines the threshold of an agent's position change in the environment, to decide the frequency of an agent's decision making. With the space resolution, an agent's position change less than the space resolution threshold does not trigger its decision making. Nor does it trigger the message passing from the agent to others. As a result, agents that move slowly make movement decisions less frequently than the fast agents. This concept of "space resolution" is derived directly from the quantization and activity concepts presented in [Zeigler *et al.*, 2004]. The value of the space resolution has significant impacts on the crowd behavior simulation. On one hand, the larger the space resolution is, the less frequently agents make decisions, and thus the more efficient the simulation is. On the other hand, the space resolution means that an agent does not update its position until its position change bypasses the space resolution threshold. This introduces position errors in the crowd simulation. The larger the space resolution is, the larger the position errors are, and thus the less accuracy the simulation is. We point out that similar kind of relationships also exists in a discrete time model, whose efficiency and precision depend on the value of the time step. A main effort of this work is to establish a formal "fair-comparison" rule that quantifies the position errors of both the discrete time and discrete event models, and conduct experiments from different aspects to compare the two. Note that both the space resolution in the discrete event model and the time step in the discrete time model are global variables shared by all agents. In this work, we use the number of decision making as an indicator of simulation performance. This is based on the observation that an agent's decision making usually involves complex logics, and thus accounts for the most significant part of computation in a simulation. We carry out this work based on the DEVS [Zeigler *et al.*, 2000] modeling and simulation framework, in particular the DEVSJAVA environment[2]. The DEVS framework was chosen due to its formal formalism and its capability of modeling both the discrete time and discrete event models. Nevertheless, we note that the model design and the conclusions drawn in this research are general and do not rely on the DEVS framework.

The remainder of this paper is organized as follows. Section 1 introduces some related work of the pedestrian crowd simulation, and discrete event models. Section 2 presents the crowd system, the discrete event agent and discrete time agent models of the crowd system. Section 3 presents a quantitative analysis of the two models. Section 4 shows experiment results from three experiments. Finally we conclude this work and propose some future directions.

1. Related work

Pedestrian crowd simulation has been studied for a long time. A well known model is Helbing's physics and social force model [Helbing *et al.*, 2002] where the behavior is described as the vector addition of the separate force

2. www.acims.arizona.edu.

terms reflecting different environmental influences. This model has successfully simulated several important features of crowd behavior, such as lane formation in crowds with opposite walking direction, oscillations of the crowd passing direction at a bottleneck, alternating collective patterns of motion at intersections and so on. Kaup's work [Kaup and Clarke, 2007] extends Helbing's model to produce more realistic behavior of an individual pedestrian under panic or non-panic conditions.

Crowd behavior is also simulated for studying emergency evacuations and safe egress. [Pan *et al.*, 2006] developed a prototype system to study some emergent human and social behaviors, such as competitive, queuing, and herding behaviors, during emergency evacuations. The work of [Pelechano *et al.*, 2007] provided a High-Density Autonomous Crowds (HiDAC) system which can be used to simulate a wide variety of emergent behaviors such as lane formation, pushing behavior and so on, by applying a set of psychological and geometrical rules with a social and physical forces model. S.R. Musse [Musse and Thalmann, 2001] proposed a hierarchy crowd model with three different ways for controlling human behaviors. The work of [Qiu and Hu, 2010] provided a common framework to model the group structures in pedestrian crowds and several group structured are demonstrated. [Seck *et al.*, 2005] proposed a dynamic personality filter which is used to model more realistic human behavior.

Most of these crowd behavior models adopt a discrete time based simulation, which is featured with a series of discrete time steps. In each time step, each pedestrian agent makes a decision to decide the next movement, based on its internal states and external environment. Such a discrete time based approach treats the crowd a uniform entity, which makes it difficult to exploit the heterogeneity of the crowd system. Thus, this work proposes a discrete event based approach to achieve more computation efficient simulation. To our knowledge, there are several works focusing on building agent based social system on the discrete event based paradigm. Dubiel's work [Dubiel and Tsimhoni, 2005] integrated ABS with DES to model humans traveling freely through a real-world problem from the theme park industry. [Zaft and Zeigler, 2002] developed a Sugarscape-style artificial society based on the Discrete Event System Specification (DEVS) formalism. However, little literature has been reported about exploring the effect of spatial and temporal heterogeneity on the agents' decision making and communication. This is what we want to achieve in this paper.

2. Discrete Event and Discrete Time Model of Crowd Behavior Simulation

A crowd system contains a simulated virtual world and multiple pedestrian agents. Before introducing the details of the discrete event and discrete time agent models studied in this paper, we briefly describe the major parameters of agents and the environment. In the following sections, to save space, the discrete event based agent model and the discrete time based agent model will be stated as *DES model* and *DTS model* respectively.

A crowd consists of a virtual environment, obstacles and multiple pedestrian agents. In this paper, the virtual environment is a rectangle area which is measured with a *width* and *length*. There are two categories of agents, *Obstacles* and *Virtual pedestrians*. *Obstacles* are stationary rectangle objects in the environment that block agent's movement. *Virtual pedestrians* are a set of autonomous pedestrian agents that move in the environment while avoiding collisions with obstacles and other agents. Each pedestrian agent is featured with a perception model and behavior control model. The perception model defines an elliptical area in front of the agent where the agent can perceive obstacles and other nearby agents. The behavior control model uses a bio-inspired behavior control architecture which decides the agent's movement (see [Qiu and Hu, 2008] for more details). In this work, each pedestrian agent has two behaviors:

1. *Move:* This behavior is used to simulate the casual movement of each agent. An agent moves to pre-defined destinations or randomly generated destinations in a sequential order. The moving path is the shortest path from the current position to the destination. And when a destination is reached, the agent moves to the next destination. In this work, the sequence of destinations is defined explicitly to ensure both the *DES* and *DTS* models use the same set of destinations for the purpose of fair comparison between them.
2. *Avoid*: This behavior is used to simulate the obstacle avoidance in the movement. When an agent is within a predefined minimum distance from the nearest neighbor agent or obstacle, it will stay away from it. The action of this behavior is as follows: if the agent is on the left side of the avoiding object, it turns right with an angle; otherwise, it turns left. In this process, a basic "collision prediction" subroutine is used to predict if the current computing agent will collide with other agents

once the turn is finished. If the subroutine returns true, the agent will try other angles recursively.

Each pedestrian agent has an *ID* that defines the global unique identification of the agent, and *Speed* that is the agent's moving speed. Two more important parameters of each agent are *SpaceResolution* and *TimeStep.* For the *DES* model, *SpaceResolution* defines the position change threshold of the agents. As described above, the larger the space resolution is, the less frequent the agents' decision making is. For the *DTS* model, *TimeStep* defines the time step of the simulation. It also affects the decision making frequency of the agents. The larger the time step is, the less frequent the agents' decision making is. As will be discussed later, both the space resolution and time step introduce position errors in the crowd simulation.

2.1. The Discrete Time Model

The *DTS* model is straightforward to understand. At every time step, each agent checks its environment (for example, if a destination is reached or if there are other agents in nearby locations), makes a decision to decide its next movement (e.g., move forward, or move sideways to avoid collision), and then carries out the movement for this time step. This will change the agent's position. Thus at the next time step, the agent goes through the same sequence again to check its environment, make a decision, and carry out the movement.

Each agent is implemented as a *DEVS* atomic model and has two states "decision_making" and "update_position". At the "decision_making" state, the agent checks its environment and makes a decision to choose a movement action. After that, the agent transits to the "update_position" state where its position is updated. The above procedure is performed for each time step. The procedure is implemented in the internal transition function *deltint()* of an agent. The pseudo-code is shown in Fig. 1.

```
procedure deltint()
 1  if the agent's current state is "decision_making"
 2        check the environment;
 3        Action a ← perform a decision making;
 4        holdIn("update_position", Timestep);
 5  else if the agent's current state us "update_position"
 6        update position based on section a;
 7        holdIn("decision_making", 0.0);
 8  end if.
end procedure deltint.
```

Figure 1. Internal transition function of the DTS model.

2.2. The Discrete Event Model

Unlike the *DTS* model, an agent in the *DES* model does not make a decision at every time step. Instead, the decision making is based on the "change" of the environment and/or the "change" of the agent's own position (according to the space resolution). Whenever such a change happens, an agent checks its environment and makes a decision to choose a movement action. Based on that action, it calculates the duration (space resolution divided by the agent's current moving speed) of the action, and after that duration elapses it performs the action (*move* or *avoid*) to update its position. This means that an agent does not update its position unless the position change equals to the space resolution. When an agent updates its position, it also notifies its nearby agents because this represents a "change" in the environment for those agents. Once those agents receive the message, they carry out the same sequence, *i.e.*, check environment, make a decision, perform action, and notify nearby agents, as described above. Note that because agents constantly move, the couplings between an agent and its nearby agents is set up dynamically based on agent's positions.

Each agent is modeled as a *DEVS* atomic model which includes an external transition function, an internal transition function, a confluent transition function (internal transition function first and then the external transition function) and an output function. These functions are used to decide the agent movements and inter-agent communication. Figure 2 shows the state transitions of an agent. In the "active" state, an agent checks its environment and makes a decision. In "move or avoid" state, the agent performs the action and carries out the movement. While in "message" state, the agent notifies the neighborhood agents about its new position.

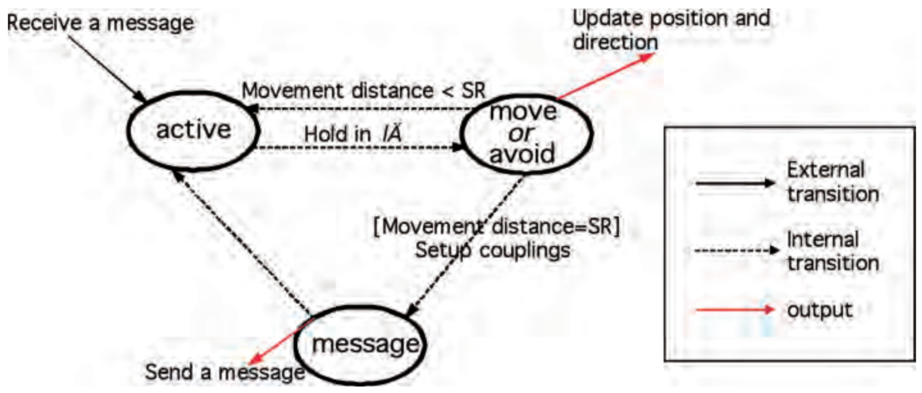


Figure 2. Agent state transition diagram.

Initially, the agent stays at the "active" state where the agent performs a decision making through the behavior control model to decide the next movement. The result is an action which indicates where the agent should move to. After holding in duration *δ*, the agent transits to "move or avoid" state where the agent moves to the new position. The agent then goes to the "message" state in order to send its new position information to the neighborhood agents if the movement distance is equal to the agent's space resolution *SR*. Before sending the position information, the couplings between the agent and the neighborhood agents are setup. And after the message is sent, the agent returns back to the "active" state. After each movement or when a new message is received from neighborhood agents, the agent performs a new decision making which continues the procedure mentioned above. The following functions implement this state transition diagram.

The external function is used to capture the messages sent from other agents. When a message is received, it transits to the "active" state. The procedure is shown in Fig. 3.

```
procedure deltext(e, x)
 1   if message received in x
 2           holdIn("active", 0.0);
 3   end if.
end procedure deltext.
```

Figure 3. External function.

In the internal function, the agent performs a decision making if the current state is “active”. Otherwise, it performs the movement, updates its position and sends the new position to the neighborhood agents if the movement distance is greater than its space resolution. The procedure is shown in Fig. 4.

```
procedure deltint()
 1  if the agent's current state is "active"
 2        update the position;
 3        check the environment;
 4        Action $a$ ← perform a decision making;
 5        holdIn(a.name, a.duration);
 6  else
 7        if the agent's current state is not "message"
 8              perform the action and update the position;
 9              if movement distance greater than $SR$
10                    setup couplings with the neighborhood agents;
11                    holdIn("message", 0.0);
12                    return;
13        else
14              holdin("active", 0.0);
15  end if.
end procedure deltint.
```

Figure 4. Internal function.

The output function is used to send messages to the neighborhood agents if the current state is “message”. The procedure is described in Fig. 5.

```
procedure output()
 1  if the agent's current state is "message"
 2        send a message to the coupled agents;
 3  end if.
end procedure output.
```

Figure 5. Output function.

3. Analysis of the DES and DTS Models

Both the *DES* and *DTS* model introduce imprecision, also referred to as *position error* in this paper, in modeling agents' position updates. For the *DTS* model, an agent's position will not be updated until the time step is reached. Within a time step, an agent's position is considered as unchanged. For the *DES* model, an agent's position will not be updated until the space resolution threshold is reached. Any position change within the space resolution threshold is not captured. This section analyzes the position error introduced by the *DES* and *DTS* models and studies their relationship. The goal is to build a ground for comparing the *DES* and *DTS* models and showing how the *DES* model can exploit the spatial-temporal heterogeneity of the crowd system.

We compare both the *DTS* and *DES* models with an analytic model for an agent's position update. Assume there are *n* agents in the crowd and the simulation is running over the time base [*t*1, *t*2], where *t*1 is the starting time and *t*2 is the ending time. Assume the position at time *t*1 is known of all agents. This position is also called *initial position*. Using the analytic model, an agent *j* ($1 \le j \le n$)'s position at time *t* ($t1 \le t \le t2$) is calculated through Eq. 1.

$$\vec{p}_{t,j} = \vec{p}_{t1,j} + \int_{t1}^{t} \vec{v}_{t,j}\,dt \qquad (1)$$

In both *DES* and *DTS*, an agent's position is updated discretely. Eq. 2 and Eq. 3 represent the position update of the *DTS* and *DES* models respectively, where Δt is the time step of the DTS simulation and *SR* is the space resolution of the DES simulation. In the *DTS* model, Eq. 2 shows that before the time step *t* is reached, the agent *j*'s position $\vec{p}_{t,j}$ is not changed. Thus, compared with the analytical model, between the time step *t* - 1and *t*, there is an error of agent *j*'s position. Similarly, in the *DES* model, Eq. 3 shows that an agent's position will not be updated until the space resolution threshold is reached. Note that in Eq. 3, P_{t-1} should be interpreted as the agent's previous position, instead of the position at time *t* - 1.

$$\vec{p}_{t,j} = \begin{cases} \vec{p}_{t-1,j} & \text{if } t < t-1+\Delta t \\ \vec{p}_{t-1,j} + \int_{t-1}^{t} \vec{v}_{t,j}\,dt & \text{if } t = t-1+\Delta t \end{cases} \qquad (2)$$

$$\vec{p}_{t,j} = \begin{cases} \vec{p}_{t-1,j} & \text{if} \left| \int_{t-1}^{t} \vec{v}_{t,j} dt \right| < SR \\ \vec{p}_{t-1,j} + \int_{t-1}^{t} \vec{v}_{t,j} dt & \text{if} \left| \int_{t-1}^{t} \vec{v}_{t,j} dt \right| = SR \end{cases} \qquad (3)$$

In order to make a fair comparison between *DES* and *DTS* models, the following condition should be satisfied: ***The maximum position error in both DES and DTS models is same.*** In *DTS* model, from Eq. 2 the maximum position error is $\int_{t-1}^{t-1+\Delta t} \vec{v}_{\max} dt$. Here $v_{\max}$ is the maximum moving speed among all agents. While in *DES* model, from Eq. 3 the maximum error is the space resolution *SR* of agents. Thus, Eq. 4 holds when the *DES* and *DTS* model are compared.

$$SR = \left| \int_{t-1}^{t-1+\Delta t} \vec{v}_{\max} dt \right| \qquad (4)$$

When the moving speed of an agent is constant during a time step, Eq. 4 can be simplified to Eq. 5 shown below. In the following, we use *TS* to represent the time step of the *DTS* model.

$$SR = V_{\max} * TS \qquad (5)$$

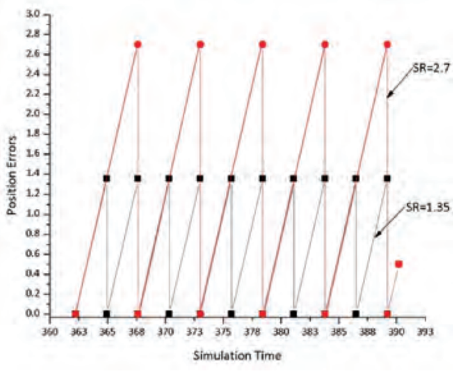
Eq. 5 is used as a basis in this work for comparing the *DES* and *DTS* models. In the next section, three experiments are used to compare the two models, and to show how the *DES* model can result in more efficient computation by exploiting the spatial-temporal heterogeneity of crowd behavior.

4. Experiment Results

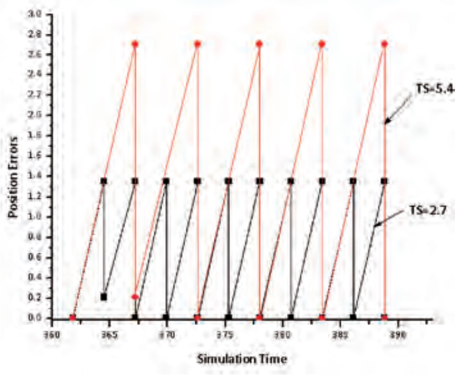
This section compares the *DES* and *DTS* models from different aspects, including position errors, number of decision makings and execution time. It intends to show that the *DES* model can achieve a more efficient simulation than the *DTS* model through comparing the execution time and number of decision makings of both models under the fair comparison condition described in section 3. The first experiment compares the maximum position errors of both models. The second experiment shows that the computation in *DES* model is less than that of *DTS* model if there is a non-uniform movement in the crowd. And the last experiment further shows that the *DES* model is more efficient than the *DTS* model if there is speed heterogeneity in the crowd.

4.1. Experiment 1

In this experiment, position errors of both *DES* and *DTS* models are compared, and the number of decision makings is explored for the simulation with one agent whose moving speed is 0.5. The agent moves through a series of destinations which are pre-defined. When all destinations are arrived, the simulation stops and position errors and the number of decision makings are calculated. Here, the position errors are based on the comparison between the *DES*/*DTS* model and the analytic model (see Eq. 1). Fig. 6(a) and Fig. 6(b) show the position errors of the two models under two space resolutions *SR* and two time steps *TS*. As described in Section 3, to ensure fair comparisons between the two models, for a specific space resolution SR in a DES simulation, the corresponding time step TS in a *DTS* simulation is calculated as TS = SR/v, where v is the agent's moving speed (0.5 in this experiment).



(a) DES model



(b) DES model

Figure 6. Position errors in *DES* model and DTS model (SR = 1.35 and 2.7, correspondingly TS = 2.7 and 5.4).

Fig. 6(a) shows the agent position errors in *DES* model under two space resolutions 1.35 and 2.7. X-axis represents the simulation time. Y-axis indicates the position error at different time. For *SR* = 1.35 the agent updates its position at time 2.7*N (N = 1, 2, 3…) since the moving speed is 0.5. And the position error is increasing linearly between two position updates. Similarly, for SR = 2.7, the agent updates its position at time 5.4*N (N = 1, 2, 3…). Fig. 6(a) shows that the greater the space resolution, the greater the position error the agent will have, and the less the simulation accuracy is. Fig. 6(b) shows the position errors in the *DTS* model under two time steps 2.7 and 5.4. It shows that the position error increases linearly between two time steps. For the time step 2.7, the maximum position error is 1.35. Note that when the agent approaches a destination (*i.e.* the time step 364.5), there exists position error because in our implementation if the agent is near the destination within a specified distance range, we consider that the agent has arrived at that destination. Fig. 6 confirms that both the *DES* and *DTS* model introduce position errors in agents' position update. To make a fair comparison, the maximum position error in both models should be the same. The *DES* model with *SR* = 2.7 and the *DTS* model with *TS* = 5.4 can be fairly compared since the maximum error in both models is same. Similarly, the *DES* model with *SR* = 1.35 and the *DTS* model with *TS* = 2.7 can be fairly compared because of the same maximum error.

Besides the maximum position error, the space resolution of a DES model also affects the number of decision makings the agent will perform. Fig. 7 shows the relationship between space resolution and the number of decision making the agent has performed in the *DES* model for different space resolutions. Here the number of decision makings calculated as the number of times the agent is in the "active" state since a decision making is performed whenever the agent is at that state (see Section 2 for more details).

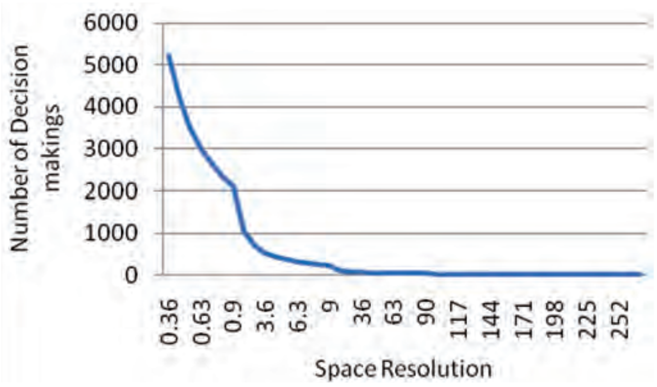


Figure 7. Relationship between space resolution and number of decision makings of *DES* model.

In Fig. 7, X-axis represents the space resolution and Y-axis represents the number of decision makings the agent has performed to traverse all predefined destinations. The number of decision makings decreases as the space resolution increases. This is because the greater the space resolution, the longer distance the agent can move in one step. Since the total distance the agent shall traverse is the same among different space resolutions, the longer distance the agent can move in one step, the fewer decision makings the agent needs to perform to finish all these destinations. As a result, one can increase space resolution to reduce the number of decision makings for more efficient computation. However, as shown by Fig. 6, the greater the space resolution, the greater the position errors are. Thus, one needs to consider the tradeoff between computation efficiency (number of decision making) and simulation accuracy (position error) and choose the space resolution in a balanced way.

4.2. Experiment 2

In experiment 1 the relationship between the number of decision makings and space resolution is explored for the simulation with one agent. Experiment 2 further compares the number of decision makings in both models for both *uniform* (*i.e.* agents with same moving speed) and *non-uniform* (*i.e.* agents with different moving speed) pedestrian crowds. The goal is to show that the number of decision makings in *DES* model is fewer than that of *DTS* model if the agents have *non-uniform* movements. In other words, the computation in *DES* model is less than that of *DTS* model, thus the *DES* model is more efficient than the *DTS* model. Otherwise, if the crowd has a *uniform* movement, the number of decision makings in *DES* model is the same as that of *DTS* model.

Three agents are used in this environment. The simulation time is 120000. Two cases are experimented. One is a *uniform* crowd where all agents have the same moving speed 0.5; the other is a *non-uniformed* crowd where agents have different moving speeds (0.005, 0.05 and 0.5 respectively). Fig. 8 shows the relationship between space resolution (*SR*) (or time step *TS*) and the average number of decision makings of the *uniform* crowd. The results of the *non-uniform* crowd are shown in Fig. 9. To ensure a fair comparison, *TS* is *2* SR* since the maximum moving speed of the agents is 0.5. The average number of decision makings is the total number of decision makings during the simulation divided by the number of agents.

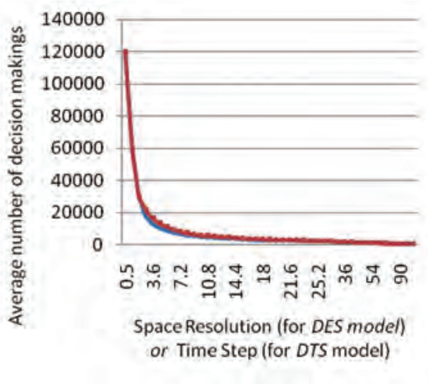


Figure 8. SR/TS and decision makings (same speed).

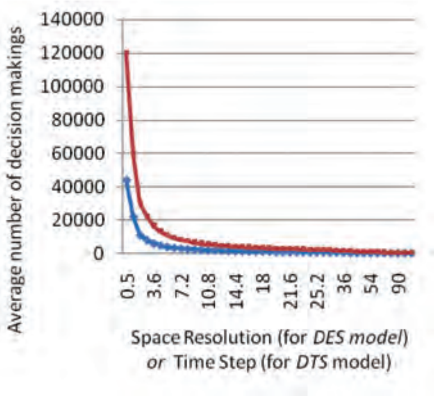


Figure 9. SR/TS and decision makings (different speed).

Fig. 8 shows that agents of *DES* and *DTS* models perform almost the same number of decision makings for all space resolutions. This is because in *DES* model, each agent in each time step will perform a decision making which is the same as the case in *DTS* model. However, for the *non-uniform* crowd (Fig. 9), *DES* model performs fewer decision makings than the *DTS* model. In other words, for the *non-uniform* crowd, under the fair comparison condition where two simulations have the same maximum position error, the *DES* model is computation more efficient than the *DTS* model. This is because in the *DTS* model, all agents need to make a decision in every time step. Thus a slow agent makes the same number of decisions as a fast agent. However, in a *DES* model the slow agent makes less number of decisions than a fast agent. To better see this, Fig. 10 shows in the case of *non-uniform* crowd, the relationship between space resolutions (or time steps) and the number of decision makings of each individual agent. Each agent in *DTS* model performs the same number of decision makings since each agent performs a decision making in each time step (shown by the curve indicated by "*DTS*"). The other three curves represent the three agents in *DES* model. As can be seen, the faster the agent moves, the larger number of decision makings the agent performs, since the faster agent performs decision makings more frequently than the slower ones. Fig. 9 and Fig. 10 show that in a *non-uniform* crowd, the *DES* model requires less computation than the *DTS* model. This will be further illustrated in Experiment 3.

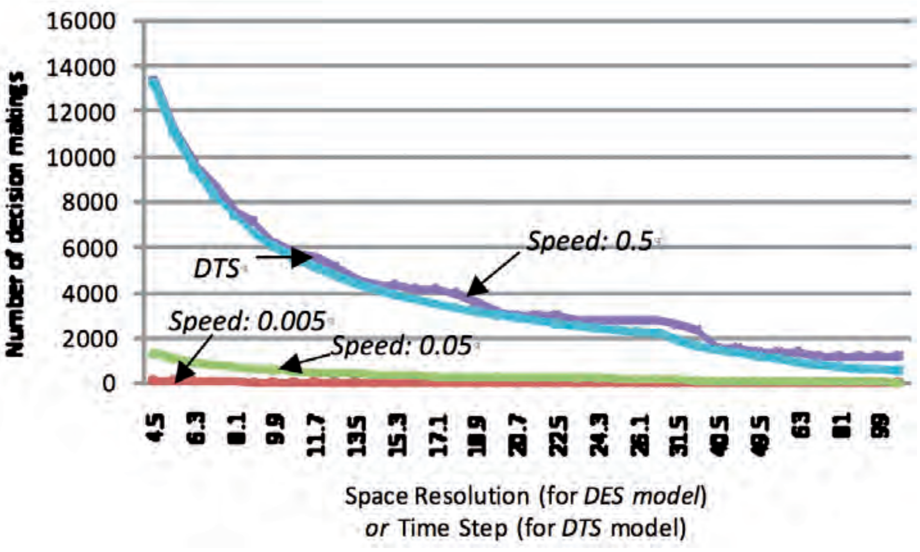


Figure 10. SR/TS and individual agents' decision makings (different speed).

4.3. Experiment 3

As an important research problem, *evacuation* of pedestrian crowds under emergent situations such as fire alarms has been studied for a long time [Helbing *et al.*, 2002]. Thus, in this experiment, we study the effect of space resolutions on decision makings and system "activities" under emergent situations. Here, activities refer to the distribution of the crowd system's decision makings in the environment. A snapshot of an emergent evacuation is shown in Fig. 11. In the simulation, agents move randomly in the environment and after a specified time, an emergency starts and all agents escape out of the environment with their speeds doubled. The system's spatial "activities" are shown in Fig. 12. As can be seen, more "activities" are distributed near the exit area since more decision makings are performed under the emergent situation and all agents escape from the exit with their speed increased.

Besides spatial heterogeneity, this system's activities also exhibit temporal heterogeneity. This is illustrated in Fig. 13, which shows the average number of decision makings of 64 simulation time intervals (*i.e.* interval *i* is [100**i*, 100*(*i*+1))) of a simulation. The simulation contains a non-uniform crowd with 20 agents where the moving speeds 0.00005 and 0.005

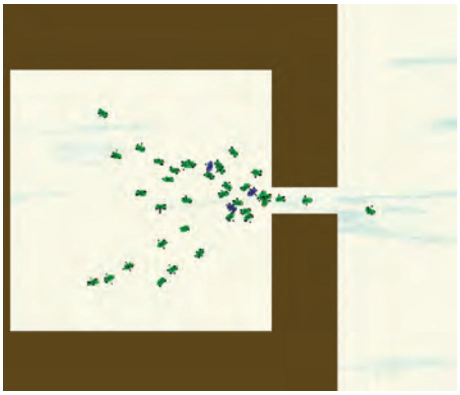


Figure 11. Simulation of emergent evacuation.

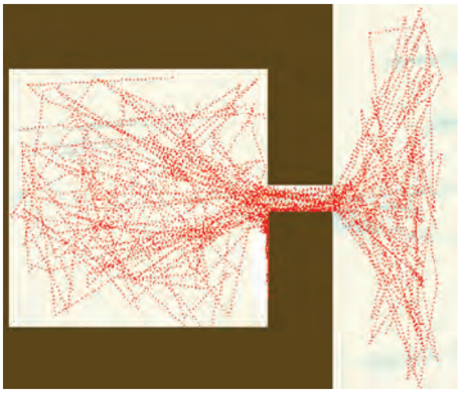


Figure 12. Illustration of system activities.

are distributed uniformly among 10 agents and the other 10 agents' moving speeds are 0.5. Note that these are agents' initial moving speeds. Agents' moving speed increase after emergency (described below). The simulation time is 6400 and the emergency start time is 1500.5. Space resolution *SR* is 0.5 and time step *TS* is 1.0. The average number of decision makings in each time interval is calculated as the total number of decision makings in that interval divided by the number of agents. Curve 1 shows the average number of decision makings of the *DTS* model for different time intervals. Since the number of decision makings of the *DTS* model dependents only on the time step, it is not changed for either the normal or emergent situations. Curve 2-4 show the distribution of the average number of decision makings of the *DES* model. Curve 2 and 3 are for the emergent cases where each agent gets a 100X and 50X speed increase respectively as along as its speed does not exceed 0.5. This constraint is used to keep a same maximum position error for both models, thus ensure a fair comparison. Curve 2 and 3 show the average number of decision makings increases after the emergency because of the speed increase. The more the speed increases, the larger number of decision makings. Curve 4 shows the average number of decision makings in each time interval for a simulation without emergency. In this case, the average number of decision making maintains the same throughout the simulation.

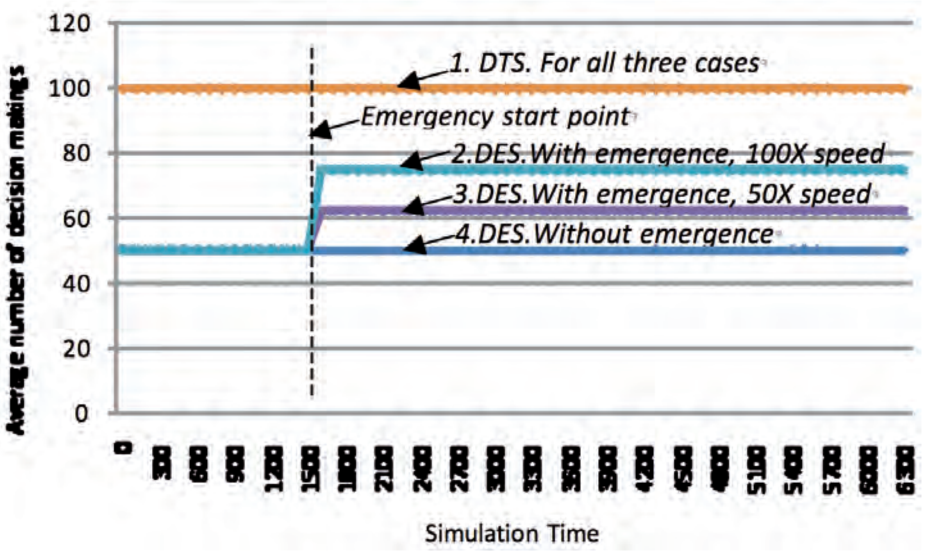


Figure 13. The distribution of the average number of decision makings.

To verify the computation advantage of the DES model as compared to the DTS model, we measure the execution time of both models for both the *uniform* and *non-uniform crowd*. We run crowd behavior simulation with 100 agents. Agents' initial speed in the *uniform crowd* is 0.05 and when an emergency occurs each agent doubles its speed. In the *non-uniform crowd*, the initial speed of each agent is generated randomly within the range [0.05, 0.5]. Similarly, when an emergency occurs each agent doubles its speed. The simulation time is 6400 and the emergency occurs at the time 500.5. Fig. 14 and Fig. 15 show the average number of decision makings and execution time for the *uniform crowd* and *non-uniform crowd* respectively. "*NoEmerg*" and "*Emerg*" stand for the normal case and emergent case respectively. "*DM*" stands for the average number of decision makings and "*Time*" stands for the execution time. The left-side Y-axis shows the execution time for each spatial resolution in milliseconds. And the right-side Y-axis shows the corresponding average number of decision makings. Fig. 14 shows that agents in both models perform almost same number of decision makings in either the normal case or the emergent case. Because of this, there is no big difference of the execution time of both models under both the normal and emergent cases. For the non-uniform crowd (Fig. 15), the *DES* model is more efficient than the *DTS* model in both cases since the execution time of the *DES* model is less than that of the *DTS* model, resulting from the less number of

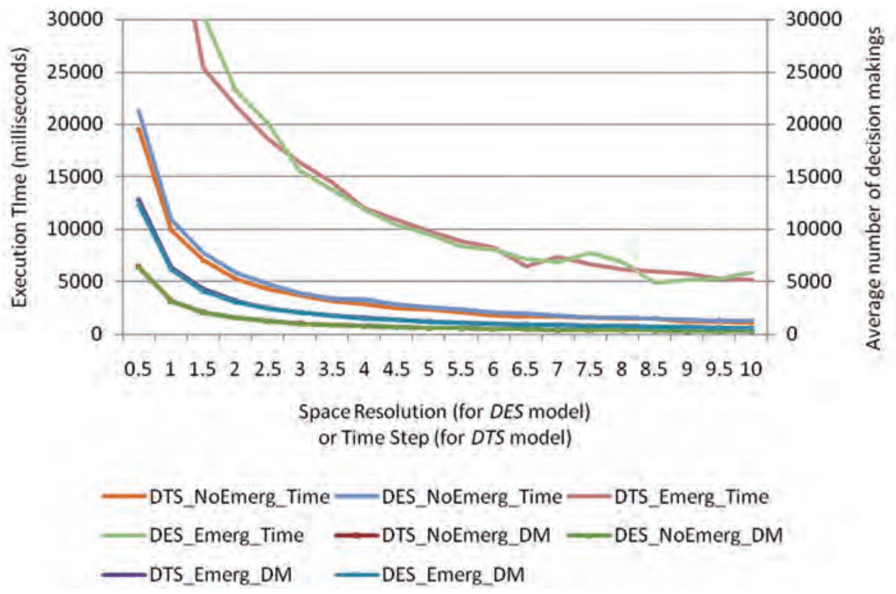


Figure 14. Execution time of *uniform crowd* in emergent evacuation.

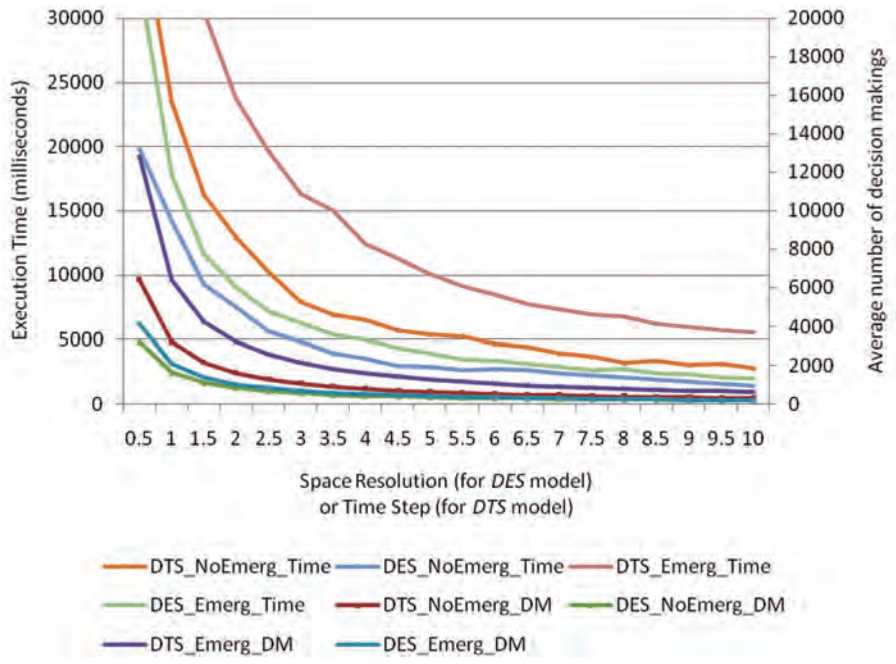


Figure 15. Execution time of *non-uniform crowd* in emergent evacuation.

decision makings performed in the *DES* model. Both Fig. 14 and Fig. 15 show that in the emergent case, agents perform more decision makings than the normal case. And thus, in the emergent case, the execution time is more than that of the normal case. This is due to the speed increase after the emergency starts. The faster the agents move the more decision makings they will perform (see section 2 for more details).

Conclusion and future work

This paper presents a discrete event model for simulating pedestrian crowd to exploit the crowd system's spatial-temporal heterogeneity resulting from agents' non-uniform movements. Both discrete event and discrete time models are considered and their performance results are compared. The experiment results show that the *DES* model can achieve better performance results (fewer decision makings and less execution time) than the *DTS* model for the non-uniform crowd. This is because the discrete event model is able to track the crowd system's "activities" in both space and time, and thus lead to more computation efficient simulations.

Future work will be carried out from several aspects. First, besides the decision makings, space heterogeneity also affects the frequency of agent interactions, such as the number of message passing between the agents. Thus one future work is to explore how the *DES* model affects the message passing among agents. Second, we will carry out more research to study how the *DES* model can affect the position error for more complex simulations with more agents. Third, we will study approaches of choosing an appropriate space resolution for a specific agent based simulation, and applying the proposed *DES* approach to other general agent based systems.

References

[Davidsson, 2000] – Davidsson P., Multi Agent Based Simulation: Beyond social simulation. *Multi Agent Based Simulation*, Vol. 1979, 2000, pp. 97-107.

[Dubiel and Tsimhoni, 2005] – Dubiel B. and Tsimhoni O., Integrating agent based modeling into a discrete event simulation. *In: Proceedings of the 2005 Winter Simulation Conference*, 2005, pp. 1029-1037.

[Helbing *et al.*, 2002] – Helbing D., Farkás I. J., Molnár P. and Vicsek T., Simulation of pedestrian crowds in normal and evacuation situations. *In:* Schreckenberg M. and Sharma S. D. (eds.), *Pedestrian and Evacuation Dynamics*, Berlin, Springer, 2002, pp. 21-58.

[Kaup and Clarke, 2007] – Kaup D. J. and Clarke T. L., Crowd Dynamics Simulation Research. *In: Proceedings of 16th Conference on Behavior Representation in Modeling and Simulation*, 2007, pp. 173-180.

[Musse and Thalmann, 2001] – Musse S. R. and Thalmann D., Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 7, No. 2, 2001, pp. 152-164.

[Pan *et al.*, 2006] – Pan X., Han C. S., Law K. H. and Latombe J. C., A Computational Framework to Simulate Human and Social Behaviors during Emergency Evacuations. *Blume Center News*, The John A. Blume Earthquake Engineering Center, Stanford University, No. 44, 2006.

[Pelechano *et al.*, 2007] – Pelechano N., Allbeck J. and Badler N., Controlling Individual Agents in High-Density Crowd Simulation. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation (San Diego, 2007)*, pp. 99-108.

[Qiu and Hu, 2008] – Qiu F. and Hu X., BehaviorSim: A Learning Environment for Behavior based Agent. *In: Proc. The 10th International Conference on the SIMULATN OF ADAPTIVE BEHAVIOR (SAB'08)*, 2008, pp. 499-508.

[Qiu and Hu, 2010] – Qiu F. and Hu X., Modeling Group Structure in Pedestrian Crowds Simulation. *In: Simulation Modeling Practice and Theory (SIMPAT)*, Vol. 18, issue 2, February 2010, pp. 190-205.

[Seck *et al.*, 2005] – Seck M., Frydman C., Giambiasi N., Ören T. I. and Yilmaz L., Use of a Dynamic Personality Filter in Discrete Event Simulation of Human Behavior under Stress and Fatigue. *In: Proceedings of the 1st International Conference on Augmented Cognition*, 2005.

[Zaft and Zeigler, 2002] – Zaft G. C. and Zeigler B. P., Discrete Event Simulation and Social Science: The XeriScape Artificial Society. *In: Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics*, 2002.

[Zeigler *et al.*, 2004] – Zeigler B. P., Jammalamadaka R. and Akerkar S. R., Continuity and Change (Activity) Are Fundamentally Related in DEVS Simulation of Continuous Systems. *Simulation and Planning AIS'04*, 2004, pp. 1-13.

[Zeigler *et al.*, 2000] –Zeigler B. P., Kim T. G. and Praehofer H.,*Theory of Modeling and Simulation*, Academic Press, 2000 (2 ed.).

Activatability for simulation tractability of NP problems: Application to Ecology

Patrick COQUILLARD[1], Alexandre MUZY[2], Eric WAJNBERG[1]

Summary – *Dynamics of biological-ecological systems is strongly depending on spatial dimensions. Most of powerful simulators in ecology take into account for system spatiality thus embedding stochastic processes. Due to the difficulty of researching particular trajectories, biologists and computer scientists aim at predicting the most probable trajectories of systems under study. Doing that, they considerably reduce computation times. However, because of the largeness of space, the execution time remains usually polynomial in time. In order to reduce execution times we propose an activatability-based search cycle through the process space. This cycle eliminates the redundant processes on a statistical basis (Generalized Linear Model), and converges to the minimal number of processes required to match simulation objectives.*

Keywords – Ecological modelling, Simulation activity, NP complexity class pollens, Generalized Linear Model, Entropy.

1. Écologie Comportementale et Moléculaire, UMR CNRS-INRA-Université de Nice, 400 route de Chappes, P.O. box 167, 06903 Sophia-Antipolis, France.

2. UMR CNRS LISA Università di Corsica - Pasquale Paoli – UFR Drittu, Scenzi suciali, ecunòmichi è di gestioni – 22, av. Jean Nicoli – BP 52 – 20250 Corti, France.

Introduction

Most of – if not all – biological and ecological systems are strongly influenced by spatial dimensions. Indeed, it is well established that, whatever the particular space scale the systems are considered, the analysis of interactions between organisms, or between organisms and physico-chemical components, is crucial to understand system behaviour and structure. Furthermore, such interactions may occur both in various ways (e.g., secretion of chemical compounds, contacts between individuals, competition for resource, gain or loss of matter and energy, etc.) and at various distances (*i.e.*, from immediate neighbouring to long distances). Finally, such interactions may occur in continuous or discrete modes in both space and time.

Powerful simulators in ecology actually take into account the spatial dimension of interactions [Wu and David, 2002; Ratzé *et al.*, 2007]. Several techniques are used, some of them allowing to involve both space and time, at a low level of details (e.g., Kendall process, stepping stone models, compartment models, etc.). Among these modelling techniques, the most powerful simulators in ecology belong to the class of "individual based models" (IBM hereafter; also denoted individually oriented models (IOM), [Fishwick *et al.*, 1998]) which allow integrating spatial interactions at a high level of details. The IBM approach completes the set of usual formal mathematical methods [Grimm, 1994; Sultangazin, 2004]. For instance, differential equations or partial differential equations are very efficient to give a coarse estimation of the evolution of large areas. However, (partial) differential equations are limited for simulating actual biological processes [Grimm and Uchmanski, 1994], particularly when the questions to be address require many details. Ecological modelling often has to account simultaneously for: (1) the diversity of individuals, (2) the spatial heterogeneity of the environment, (3) the changing interaction network (and changes of biotic structures), (4) the discrete and distant interactions between individuals, (5) the random processes and behaviours (*i.e.*, random spatial interactions or movements), etc.

IBMs are often implemented by object-oriented models [Coquillard and Hill, 1997] or by multi-agent models when there is a need to represent an autonomous social behaviour of individuals heading a common goal [Ferber, 1999]. In this case, IBMs are usually called agents. In addition, such modelling approach has the main following advantages [Hill and Coquillard, 2007]: (1) it allows theoretically the simulation of ecosystems with large sets of species harbouring different behaviours. Moreover, object classes can account for a part of mathematical modelling in order to obtain combined simulations if needed (mixing the discrete and continuous approach). (2) However, a lot of fieldwork always remains necessary as well as a deep knowledge of the modelled species. (3) It takes into account the spatial features of ecosystems that is difficult with partial differential equations (e.g., compartment models), or with the classical Markovian analysis. (4) It provides the possibility to manage, for each individual, the set of parameters the biologist decides to integrate in the model. The management of individuals, and correlatively of their physiological variations, enables model refinement to approach reality according to the detailed level wished by the user.

In a first part, we will show, through a simple example, that (1) introducing spatial relationships between individuals is a prerequisite to maintain a sufficient level of diversity and (2) that such operation requires a probabilistic approach. In a second part we will propose a statistically driven method for reducing the state of space for a model.

1. Reducing activity

Example of spatialization necessity

Let us have a look to the prisoner's dilemma which is the most emblematic problem in the game theory. The problem was initially formalized by W. Tucker. In its classical form, the dilemma is expressed as follows:

> Two suspects are arrested by the police. The police having separated both prisoners visit each of them to offer the same deal. If one testifies for the prosecution against the other and the other remains silent (cooperates), the betrayer goes free and the silent accomplice receives a full 10-years sentence. If both remain silent, both prisoners are sentenced to only six months in jail. If each betrays the other, each receives a five-year sentence. Each prisoner must choose to betray the other or to remain silent. Each one is assured that the other would not know about the betrayal before the end of the investigation. How optimally should the prisoners act?

In this game, the only concern of each prisoner is to maximize his payoff. Consequently, all rational players should play "testify" (Fig. 1) and cooperating is strictly dominated by defecting. As a consequence of such strategy, the game leads to the disappearance of cooperators. But many examples of coexistence of cooperation and selfish behaviours can be found in animal societies and economical situations. How is it possible? M. Nowak and R. May demonstrated in 1992-1995 that introducing spatial dimension in the dilemma, even in an elementary – and somewhat opened to criticism – form, makes such situation possible. They first reformulated the dilemma by introducing a sentence variable b ($b > 1$). Then, they distributed players on a grid in which each player has a probability to become a cooperator. This probability is function of states and gains of its immediate neighbours (see Fig. 2 for details). As a result of such a transformation, they obtained for some couples *(m, b)* the coexistence of both strategies (see Fig. 3). By doing that, however, they brought into the model some probabilistic compounds. Actually, this example illustrates clearly the usual way simulators reproduce spatial interactions.

If the other prisoner testifies:

If I remain silent, I will receive the full 10-years sentence;
But if I testify, I will only receive a 5-yeats sentence

If he does not:

If I remain silent, I will receive a 6-monthes sentence;
But if I testify, I will be free.

"Whatever his choice, I have interest to testify"

	Silent	Testify
Silent	(-0,5, -0,5)	(-10, 0)
Testify	(0, -10)	(-5, -5)

$\downarrow$

	Silent	Testify
Silent	(1 , 1)	(b , 0)
Testify	(0 , b)	(0 , 0)

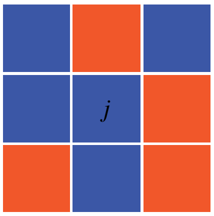
Figure 1. The original prisoner's dilemma (upper table) is reformulated by introducing $b > 1$ as a sentence variable (lower table).

$$P(j) = \sum_{i \in N_j} A_i^m s_i \Big/ \sum_{i \in N_j} A_i^m$$ = probability that *j* switches to a cooperator state

With $\begin{cases} s_i = 0 \text{ (selfish 🟥)} \\ s_i = 1 \text{ (cooperator 🟦)} \end{cases}$

N_i = Set of neightbours of *j*

A_i = Gains of cell *i* at time *t*



$m = 1$,	original stochastic game
$m \to \infty$,	the more the cell *j* will get a chance to adopt the more profitable strategy of its neighbours (deterministic game)
$m = 0$,	the gains are worthless, neutral derivate between both strategies

Figure 2. The spatialized dilemma (after M. Nowak and R. May). *P*(*j*) is the probability the prisoner *j* has to become a cooperator.

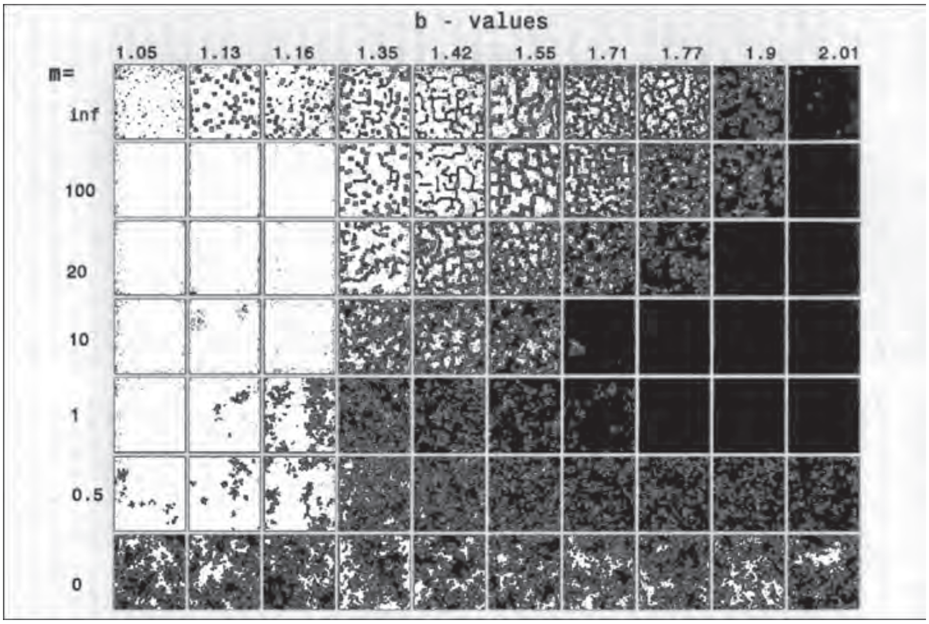


Figure 3. The spatialized prisoner's dilemma in the *(m,b)* plane. Each cell of the grid represents a game of 100 × 100 players. Large areas present the coexistence of cooperators (white) or selfish players (black). In gray colour, the players which have just changed of state. After M. Nowak and R. May (modified).

Obviously, integration of space into simulators leads to a better representation of reality. However, the level of details increases the number of parameters, computations and interactions between parameters. This results in an explosion of the state space and to the intractability of simulations. Therefore, solutions have to be found to reduce the state space and thus enhance tractability.

Space and NP problems

According to the previous example, we can conclude that embedding spatial interactions into simulators (designed to model ecological systems) requires, in most cases, a stochastic approach. In most cases, several processes are acting simultaneously in the course of runs to simulate spatial interactions (e.g., various types of competition, seed dispersal, migration, gene flows, chromosome shuffling, chromosomal crossovers, etc.) Considering that each of these processes act on several levels (that can be large), the number of possible trajectories of the system is an exponential function of the number n of processes varying on p levels, and the time required to find a particular trajectory of the system is $O(n^p)$. In other words, this problem belongs to the NP complexity class problem (see Fig. 4). This is the reason why ecologists have early decided to reformulate the statement: "finding a particular trajectory" into "finding the most probable trajectory".

Let us define *activity* of a system as *its number of transitions* and *activatability* as *the probability of transition activation*. If we consider p processes varying on $[1,\ldots,n]$ levels and that each level can be activated with a probability following a law $\boldsymbol{\pi}$, the activatability of the process i is (Fig. 4):

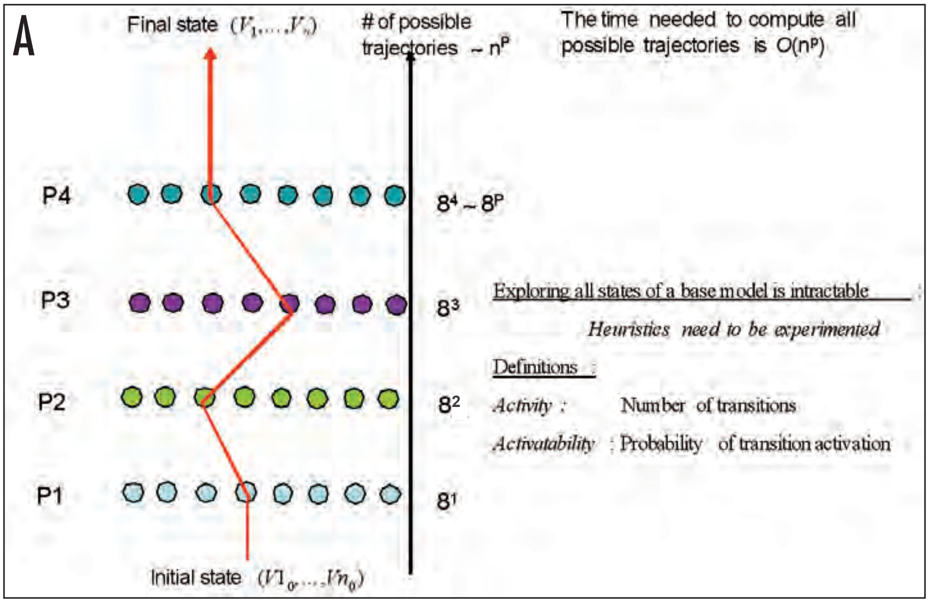
$$\Delta_i[\boldsymbol{\pi}_{\boldsymbol{i}}(s_{i1},\ldots,s_{in})],$$

The activity can be estimated as proportional to the number of replicates (R), the confidence interval (%) and the number of processes (p):

$$A\alpha(R,\%,p)^3$$

The number of replicates R depends on σ (the standard error of the response).

3. The confidence interval of a mean is calculated as: $\bar{x} - T_{\alpha/2;d}\frac{s}{\sqrt{R}} \le \mu \le \bar{x} + T_{\alpha/2;d}\frac{s}{\sqrt{R}}$, where $T_{\alpha/2}$ is given by the student law with $\alpha \le 0.05$ and d is the degree of freedom (df).


A
Final state $(V_1,...,V_n)$
# of possible trajectories ~ n^P
The time needed to compute all possible trajectories is $O(n^P)$
P4
P3
P2
P1
8^4 ~ 8^P
8^3
8^2
8^1
Exploring all states of a base model is intractable
Heuristics need to be experimented
Definitions :
Activity : Number of transitions
Activatability : Probability of transition activation
Initial state $(V1_0,...,Vn_0)$

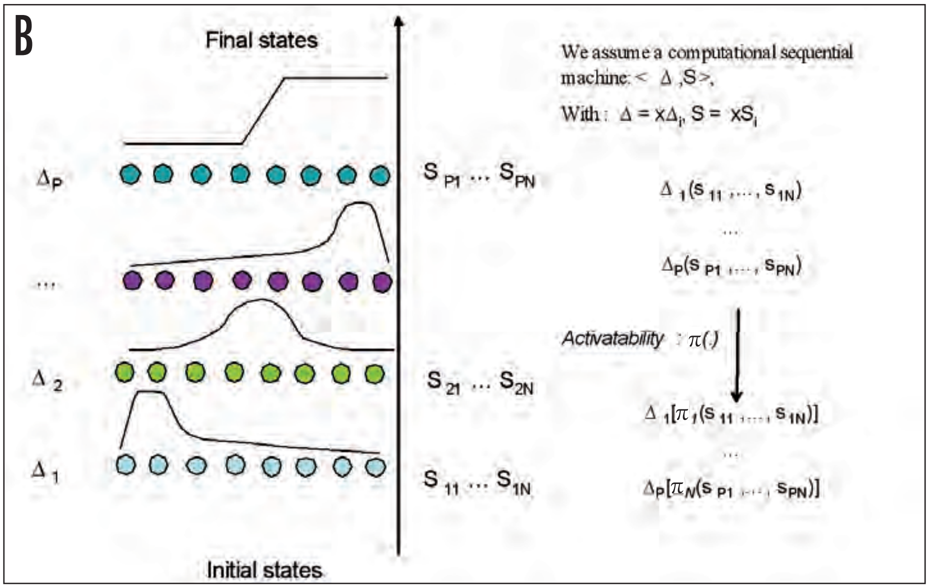

B
Final states
Δ_P
...
Δ_2
Δ_1
S_{P1} ... S_{PN}
S_{21} ... S_{2N}
S_{11} ... S_{1N}
We assume a computational sequential machine: < Δ ,S>,
With : $\Delta = x\Delta_i$, $S = xS_i$
$\Delta_1(s_{11},...,s_{1N})$
...
$\Delta_P(s_{P1},...,s_{PN})$
Activatability : $\pi(.)$
$\Delta_1[\pi_1(s_{11},...,s_{1N})]$
...
$\Delta_P[\pi_N(s_{P1},...,s_{PN})]$
Initial states

Figure 4. Activity, activatability and processes.

Reformulating the original question in "finding the most probable trajectory", we considerably reduced the computation time and escape from the NP-problem trap along with its heuristic solutions (the problem is now solvable in a polynomial time $O(k\log(p))$ with p processes). However, the problem of activating many stochastic processes is still relevant and some algorithms can be time consuming (for instance, the algorithm AKS which tests the primarity of a number is $O(\log(n)^{10.5})$).

These considerations lead to the following question: "How to reduce the number of processes?" Kleijnen and Groenendaal [1992] proposed that building of $2^{(n-1)}$ experimental designs in which each process is either active or inactive, give the same information than a 2^n protocol (*i.e.*, involving a half of process combinations). Thus, it is possible to test the effects of each process on the results eliminating redundant ones. The use of $2^{(n-k)}$ protocols is also possible but results in introducing confusion between some interactions and a confusion of the main effects with their interactions.

Proposal of an activatability cycle

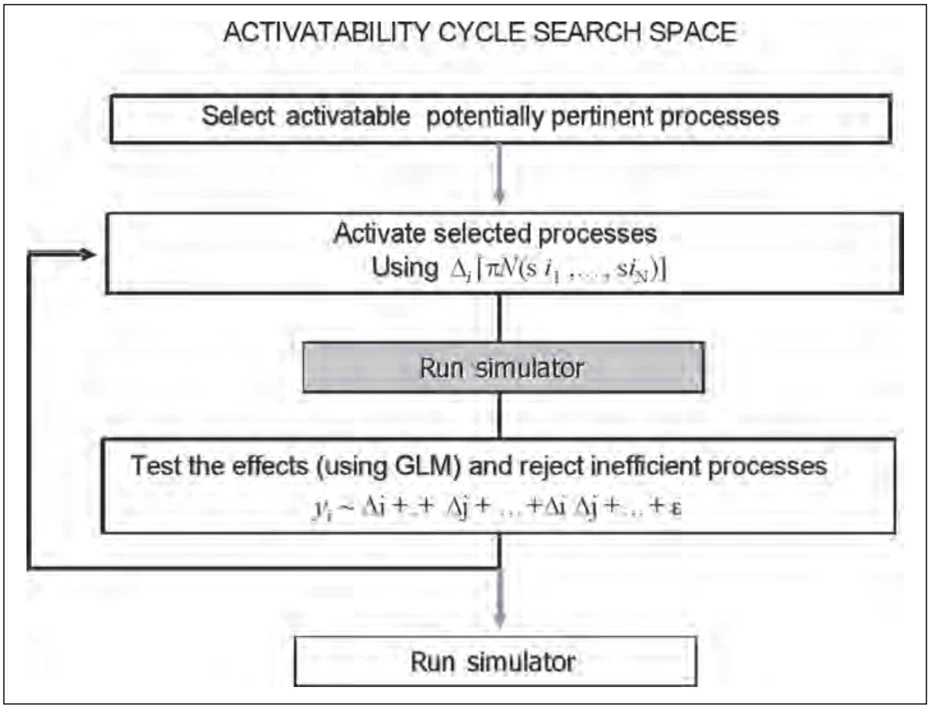

ACTIVATABILITY CYCLE SEARCH SPACE
Select activatable potentially pertinent processes
Activate selected processes
Using $\Delta_i[\pi N(s\,i_1,\ldots,si_N)]$
Run simulator
Test the effects (using GLM) and reject inefficient processes
$y_i \sim \Delta i + \_ + \Delta j + \ldots + \Delta i\,\Delta j + \ldots + \varepsilon$
Run simulator

Figure 5. The activatability cycle. Every loop, processes with no significant effects are eliminated.

In this section we will propose a method to build the most parsimonious model from a list of processes. Based on statistical evidences, the method is automatable and allows to embed into the simulator all the process which have a significant effect on the results of simulation, even those on which we have no a priori ideas about their pertinence in the model. However, the method cannot be seen as a validation process. In addition, the proposal must not be confused with the works of Hoffman [2005], who proposed an extended genetic algorithm based method "to accomplish simultaneously parameter fitting and parsimonious model selection" among a list of candidate models.

The proposed cycle of activatability (Fig. 5) is based on a complete design protocol (2^n protocol). At every cycle, main effects and their interactions on the response y, can be tested through a Generalized Linear Model (Nelder and Wedderburn, 1972).

$$E(y_i) = \beta_0 + \beta_1\Delta_1 + \ldots + \beta_i\Delta_i + \ldots + \beta_k\Delta_i.\Delta_j + \ldots + \varepsilon \quad (1)$$

where y_i is the dependant variable[4], Δ_i ($i \in [1..n]$) the principal effects (or independent variables), $\Delta_i.\Delta_j$ the interaction between the effects (sometimes called "product terms") and ε is a random error. Equation (1) is thus a linear regression. Quadratic effects can also be included in the regression (*i.e.* $(\Delta_i)^2$)). In a GLM, it is assumed that ε obeys to one function of the exponential family (Normal, Poisson, Binomial, etc.). The β_k parameters represent the variation of $E(y_i)$ when the kth variable move of one unit, the remaining variables being unchanged. Formally:

$$\beta_i = \frac{\partial E(y_i)}{\partial \Delta_i}$$

Equation (1) is solved by the usual matrix method for multiple regressions. In the general case, the resulting model is then tested against the y_i responses by means of an analysis of variance (ANOVA) which leads to: $R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$, called the correlation ratio, where $\hat{y}_i$ is the predicted response, $\bar{y}$ the average response and y_i the observed response. R^2 gives the amount of variation of the y_i which is explained by the model.

4. In addition, the dependant variable y can be transformed by means of a link function. This is usually the case when the y_i responses do not follow the Normal distribution. Note that GLM assumes that the y_i observations are independent.

In addition, it can be demonstrated that $F = \frac{R^2/k}{(1-R^2)/(n-k-1)}$ follows a Fisher distribution with *k* and *k*-*n*-1 degrees of freedom. In such conditions, we can reject the null hypothesis H_0: $\beta_1 = \beta_2 = \ldots = \beta_k = 0$, if $P(F \geq F_{calculated}) < \alpha$, with $\alpha = 0.05$. However, even if we reject the null hypothesis, this does not imply that all variables of the model have a significant contribution to the response *y*. To decide if a particular variable *j* has a significant contribution to the response we calculate $F = \frac{\beta_j^{\ 2}}{s^2(\beta_j)}$ and reject the hypothesis H_0: $\beta_j = 0$, if $F > F_{\alpha;1,n-k-1}$. To test successively each variable of the model, a stepwise[5] procedure eliminates and introduces the variables in the model of the response *y* (usually the *p*-value <*0.05* criterion is used).

Such a procedure takes advantages from allowing to embed into the simulator all the variables (= processes) the user wants to test – with no *a priori* exclusion. Another advantage is that processes can be aggregated into sets which can be treated as active units. In this case, the user attempts to measure the effects of some aggregated activities (sexual reproduction for instance) on a response (population dynamics involving both sexual and asexual reproduction of plants).

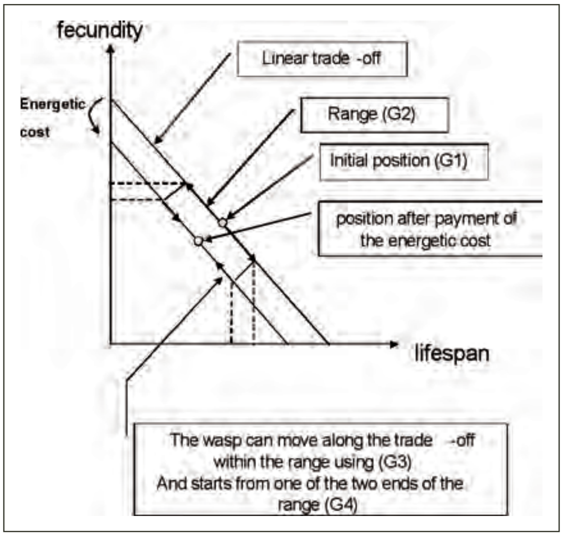

fecundity
Linear trade -off
Energetic cost
Range (G2)
Initial position (G1)
position after payment of the energetic cost
lifespan
The wasp can move along the trade -off within the range using (G3)
And starts from one of the two ends of the range (G4)

Figure 6. The life span-fecundity trade-off of a parasitoïd wasp.

5. Backward and forward methods are also used. Each of them introduces or eliminates step by step the variables into the model.

Application

In this application, four processes are represented by four real values coded on *16 bit*-structure called genes (see Fig. 6). The animal (a parasitoid[6] wasp) has an initial position along a trade-off (coded by gene *G1*) and can change its reproductive strategy throughout its life thanks to gene *G2*, which defines a range. The wider *G2*, the heavier the cost to pay (in term of energy). The cost reduces both fecundity and lifespan of the animal. The wasp can move within the range thanks to gene *G3*, which is a parameter of a Bayesian estimator re-evaluated each ten time steps of its life as follows:

```
Posterior = prior × G3 + (1-G3) × posterior;
Fecundity = Fecundity + G2 × posterior;
```

When a wasp encounters a patch of hosts, the number of hosts it attacks obeys to a saturation function. Thus, its velocity of attacking hosts decreases, following an exponential function. When its velocity has reached the average velocity calculated on the basis of the average environment richness, it leaves the patch and tries to find a new one. The cycle "foraging for hosts on patches and travelling between patches" is repeated until the wasp has reached the end of its life or has exhausted its potential fecundity.

The four genes are encapsulated into a single chromosome. Each wasp holds a single chromosome. The goal of the simulation consists in finding the vector {*G1,…,G4*} which maximises the score of the wasp, *i.e.* the number of eggs laid throughout its life. The score maximization is obtained by means of a genetic algorithm (GaLib, MIT, 1997-2007).

Basically, the four genes {*G1,…,G4*} are variables. However, each of them induces the call of several functions in the code and modifies the behaviour of the wasp. For instance, the gene G2 can strongly modify the phenotypic plasticity of the animal (*i.e.* its ability to adapt its fecundity/lifespan ratio to the environment characteristics) and influences its score. That is why we will now consider the {*G1,…,G4*} genes as processes instead of variables.

First cycle.

In this application, 2625 experiments (10 replicates each) were done. The results showed that there was no significant effect of *G4* on scores, whatever

6. An organism that lives at the expense of another (its host), impedes its growth and eventually kills it. Insect parasitoids, which are often very tiny, attack a single organism (plant or animal), from which they derive everything they need for their own growth and reproduction. One way a parasitoid does this is by laying its eggs in the body of the host insect (from Natural Canadian Research document).

the initial conditions in which wasps had to evolve. Consequently, the process directed by *G4* was dropped.

Second cycle.

Significant effects on {*G1, G2, G3*} were found, and the three processes clearly acted on animal scores (table I).

	G1		***G2***		***G3***	
Source of variation	*F* Value (calculated)	Pr > *F*	*F* Value (calculated)	Pr > *F*	*F* Value (calculated)	Pr > *F*
stability of environment (1)	263.81	**<.0001**	24.17	**<.0001**	2.04	0.0864
inter patch travel time (2)	222.31	**<.0001**	91.23	**<.0001**	3.22	**0.0121**
(1).(2)	7.97	**<.0001**	1.78	0.028	0.74	0.7534
energetic cost (3)	2.34	0.0532	56.46	**<.0001**	4.15	**0.0023**
(1).(3)	1.11	0.3409	2.96	**<.0001**	1.29	0.1968
(2).(3)	1.28	0.2031	4.85	**<.0001**	0.59	0.8926
averaged # hosts on patchs (4)	3045.64	**<.0001**	25.14	**<.0001**	10.95	**<.0001**
(1).(4)	20.85	**<.0001**	11.48	**<.0001**	1.19	0.2392
(2).(4)	56.15	**<.0001**	11.05	**<.0001**	2.05	**0.0019**
(3).(4)	1.25	0.1834	1.45	0.0741	1.26	0.1794
stochasticity (5)	24.37	**<.0001**	2.19	0.1126	1.66	0.1899
(1).(5)	4.27	**<.0001**	1.08	0.3755	1.14	0.3346
(2).(5)	1.47	0.1612	1.9	0.0553	0.52	0.844
(3).(5)	0.36	0.94	0.49	0.8676	1.38	0.202
(4).(5)	6.3	**<.0001**	3.07	**0.0003**	1.08	0.375

Table I. ANOVA test on the model obtained by the GLM procedure. Sources (factor) of variation (Δi) and interactions ($\Delta i.\Delta j$) of the linear model are indicated in the left column. Effects on {*G1,…,G4*}: *F* values and their probability to be greater than the theoretical values of the Fisher law are indicated for {*G1,…,G3*}; values for *G4* are omitted since probabilities were systematically ≥ 0.05. Significant effects are indicated in bold.

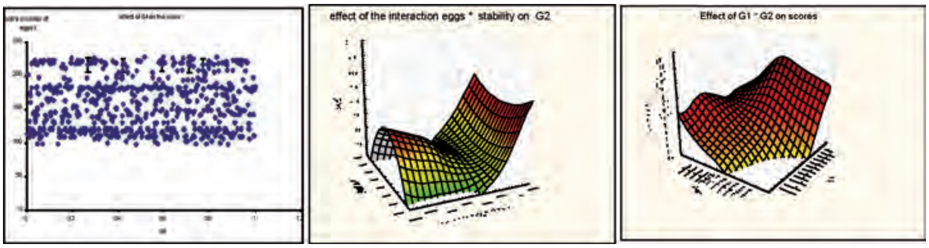


Figure 7. Simulation results (population size = *500*, number of generation = *300*, number of replicates = *50*). Left: no effect of *G4* on scores was detected according to the ANOVA results. Centre and right: after removal of *G4*, significant effects of initial conditions on {*G1, G2, G3*} and of {*G1, G2, G3*} on scores were identified.

	With GA4	Without GA4
Code size (compiled)	842 757	842 629
# functions	58	52
# max of calls	1 992 190	1 234 692
Virtual mem. (RES + swap)	3 380 000	3 380 000
RES	1 572 000	1 572 000
Execution time	8mn47.661	6mn59.502

Table II. Comparison between simulations embedding process *G4* or not (50 replications of a run initialized with a single combination of parameters).

Clearly, results (table I & Fig. 7) showed that process *G4* was redundant. Removing *G4* makes the simulations faster than simulations embedding the four processes, and gain is about *20%*. Most execution time reduction was a consequence of the reduction of function calls (*1 234 692* versus *1 992 190*). On the contrary, code size and memory size remained unchanged (Table II).

However, it is clear that if the activatability cycle was designed to select processes contributing significantly to the response of the simulator, it does not constitute a validation of the resulting model. Indeed, the validation phase must be engaged after the selection of processes has been achieved, as it is usually done in a classical approach. Lastly, if the resulting model can be validated by comparison with experimental data corresponding to the protocol design, the model has few chances to be validated when confronted to other data. In this case, the experimental design must be rebuilt and the activatability cycle reengaged.

2. Monitoring the activity of a simulated system

The Table I showed that the activatability cycle we used resulted in a substantial reduction of activity of the simulator. However, the activity itself was indirectly estimated through both computing time and number of functions called over the runs. Consequently, the monitoring of activity throughout time remains an opened question.

In the information theory, the entropy is considered as a measure of the disorder of the system. Let us consider four simple binary units {*G1,…,G4*} which can be in one of the two states: *Gi* = 0 or *Gi* = 1. We thus have 16 possible states:

G1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
G2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
G3	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
G4	1	0	0	1	1	0	1	0	1	0	1	0	1	0	1	0

If the states have equal probabilities, the probability of each state is: (1/(number of states)).2^{-N}, where *N* is the length of a state. Thus, we find the entropy of each unit: $s = \frac{1}{N}\log_2(2^N) = 1$. Consequently, if the 16 states are not of equal probability we find that the entropy *per unit* is smaller than 1. The entropy of the system can be computed as:

$S = -\sum p_i \log_2(p_i)$, where p_i is the probability of each state.

In such conditions, we can ascertain that the entropy S_t of the system at the instant *t* is limited to the range: S_{min} (*G*) < S_t (*G*) < S_{max} (*G*). Indeed, S_{min} represents the particular case in which there is only one possible state (*i.e.*

one possible combination of {*G1,…,G4*}) and S_{max} represents the case where all possible combinations have equal probabilities (S_{max} = 4 in this small example).

In a general case, the processes {*G1,...,G4*} can take several values as we saw in the application section. Because of the stochasticity of the simulation, one usually conducts simulations through replicates in order to obtain averaged values $\{\overline{G1},...,\overline{G4}\}$ and associated variances *at each time step of the runs*. It can be reasonability admitted that the probability density function of *G* conforms to a multidimensional Gaussian (in a majority of cases at least). A measure of the entropy can be obtained by establishing the variance-covariance symmetric matrix $\sum$ ($N \times N$) of the vector $\{\overline{G1},...,\overline{G4}\}$ and next by the calculation of its determinant[7]. This value gives the total amount of information diminished by the interactions between the processes. It is also an approximation of the number of possible combinations at each step of time, *i.e.* a hyper volume of the dispersal of the state space. This hyper volume corresponds to the activatability state space of the system. That is, at every time step, the states of the system can change. Considering a state change of the system as an activity (*i.e.*, constraining the definition of the *activity* term) of the system, its activatability corresponds to its possible state changes or activities. Under these assumptions, the entropy $S_t(Gi)$, at the *t* instant, of the process *Gi* is given by [Ahmed and Gokhale, 1989]: $S_t(G_i) = (1/2)\ln(2\pi e\sigma_i^2)$ where σ_i^2 is the *i*th element of the diagonal of $\sum$, and the entropy (or differential entropy) $S_t(G)$, is then given by:

$$S_t(G) \leq (1/2)\ln(2\pi e)^N \left|\Sigma\right.$$

S_{min} and S_{max} do not constitute some likely/sustainable situations. The former represents the case in which the system is fixed and is unable to adapt its behaviour to a fluctuant environment. The later characterises a system which is highly adaptable (it can face any situation), but with a too high cost of energy (e.g., for natural systems) or in terms of resources and computation time (virtual systems; see also the Figure 6)). We can thus ascertain that the environment in which a system evolves imposes some constraints to the system so that it must adopt a certain level of disorder (positive entropy). This disorder allows the system to face the fluctuations of the environment – within a fixed range – to the extent that it pays an energetic price corresponding to such flexibility.

7. Sometimes called generalized variance.

Conclusion

We conclude this paper in enumerating the following traits of the proposal in two short sections:

Advantages

- The proposal of the activatability cycle is automatable.
- The method allows embedding into the simulator all the processes wished by the user with no *a priori* exclusion.
- Processes are activated according to their statistical effects on results.

Drawbacks and troubles

- Reformulating the question, one loses the prediction (statistical results) but recognizes the dimension of complexity in the scientific explanation.
- The necessity of replicates strongly diminishes the information about spatial results. Thus, spatial trajectories of particular interest cannot be identified.
- The selection of processes must be a conservative operation (the internal coherence must be preserved).

Acknowledgment

We would like to warmly thank Pr. Thomas S. Hoffmeister who participated to the elaboration of the conceptual model of the fecundity-lifespan tradeoff of the parasitoïd wasp we used as an example of application.

References

[Ahmed and Gokhale, 1989] – Ahmed N. A. and Gokhale D. V., Entropy expressions and their estimators for multivariate distributions. *In: IEEE Transactions on Information Theory*, 35(3), 1989, pp. 688-692.

[Coquillard and Hill, 1997] – Coquillard P. and Hill D. R. C., *Modélisation et simulation d'écosystèmes*, Paris, Éd. Masson 1997.

[Ferber, 1999] – Ferber J., *Multi-Agent Systems: An introduction to distributed artificial intelligence*, Adisson Wesley, 1999.

[Fishwick *et al.*, 1998] – Fishwick P. A., Sanderson J. G. and Wolff W. F., A Multimodeling basis for across-trophic-level ecosystem modeling: The Florida everglades example. *SCS Transactions on Simulation*, 15 (2), 1998, pp. 76-89.

[Grimm, 1994] – Grimm V., Mathematical models and understanding in ecology. *Ecological Modelling*, 75-76, 1994, pp. 641-651.

[Grimm and Uchmanski, 1994] – Grimm V. and Uchmanski J., Ecological systems are not dynamic systems: some consequences of individual variability. *In:* Grasman J. and van Straten G. (eds.), *Predictability and Nonlinear Modelling in Natural Sciences and Economics*, Dordrecht (Germany), Kluwer Academic Publishers, 1994, pp. 248-259.

[Hill and Coquillard, 2007] – Hill D. R. C. and Coquillard P., Ecological modeling and simulation: from historical development to individual-based modeling. *In:* Fishwick P. A. (ed.), *Handbook of Dynamic System Modelling*, New York, Chapman & Hall/CRC, 2007, pp. 29-1/29-15.

[Hoffman, 2005] – Hoffman J. P., Simultaneous inductive and deductive modeling of ecological systems via evolutionary computation and information theory. *Simulation*, 82(7), 2005, pp. 439-450.

[Kleijnen and Groenendal, 1992] – Kleijnen J. P. C. and Groenendaal van W., *Simulation: a Statistical Perspective*, Chichester (U.K.), Wiley, 1992.

[Nelder and Wedderburn, 1972] – Nelder J. and Wedderburn R., Generalized Linear Models. *Journal of the Royal Statistical Society*, Series A (General) 135, 1972, pp. 370-384.

[Nowak and May, 1993] – Nowak M. A. and May R. M., The spatial dilemmas of evolution. *International Journal of Birfurcation and Chaos*, 3(1), 1993, pp. 35-78.

[Ratzé *et al.*, 2007] – Ratzé C., Gillet F., Müller J.-P. and Stoffel K., Simulation modelling of ecological hierarchies in constructive dynamical systems. *Ecological complexity*, 4, 2007, pp. 13-25.

[Sultangazin, 2004] – Sultangazin U. (ed.), Mathematical Modeling of Ecological Systems. *Mathematics and Computers in Simulation*, Special Issue, 67 (4-5), 2004, pp. 275-500.

[Wu and David, 2002] – Wu J. and David J. L., A spatially explicit hierarchical approach to modeling complex ecological systems: theory and applications. *Ecological modelling*, 153, 2002, pp. 7-26.

Are asynchronous cellular automata essentially periodic? A conjecture based of the theory of hybrid systems[1]

James NUTARO[2]

Anstract – *Asynchronous cellular automata, synchronous cellular automata, and a special class of hybrid automata are three, essentially interchangeable, forms of a single class of systems. This article expresses these seemingly distinct formalisms as mathematical systems, and from this basis shows that the former are homomorphic simplifications of a particular class of hybrid automata. It is further conjectured that, as a consequence of this relationship, every asynchronous cellular automaton is essentially periodic: like their synchronous counterparts, asynchronous automata appear to settle into a very regular, predictable trajectory if they are observed for a sufficiently long period of time. This conjecture is based on recent work in the theory of differential automata with constant derivatives. The paper concludes with a discussion of problems posed by the hypothesis and of its consequences for discrete event systems that are schedule preserving.*

Keywords – asynchronous cellular automata, hybrid systems, discrete event systems.

1. Research sponsored by Laboratory Directed Research and Development program of Oak Ridge National Laboratory(ORNL), managed by UTBattelle, LLC for the United States Department of Energy under contract no. DE-AC05-00OR22725. This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

2. Oak Ridge National Laboratory, Oak Ridge, TN (nutarojj@ornl.gov).

Introduction

Self-clocked cellular automata are similar to synchronous cellular automata, of which the Game of Life [Gardner, 1970] is best known, but differ in that each cell evolves at its own rate. The rate is expressed as a duration, which can be any real positive number, for the cell's state. A fixed duration is assigned to each cell, and the cell changes state at intervals equal to the duration. A transition rule, which is common to all cells, assigns the transitioning cell a new state that is determined by the current state of itself and its neighbors.

To be concrete, consider a one dimensional, self-clocked cellular automaton with n cells indexed 1, 2, …, n. The neighborhood of cells with index $1 < k < n$ are the cells at k-1 and k+1. The neighbors of the leftmost cell at k=1 are the cells at locations 2 and n; the neighbors of the rightmost cell are at locations n-1 and 1. The subscript k,l is used to denote the left neighbor of cell k and k,r the right. Three pieces of information are maintained for each cell: the time e_k that has elapsed since its last transition, its discrete state q_k, and its duration P_k. The elapsed time begins, and remains, in the interval $[0,P_k]$. The transition rule δ maps the state of a cell and its neighbors into a new state when $e_k=P_k$. A three step procedure is used to simulate the cellular automaton:

1. Find the time t_N until the next transition by finding the smallest P_k-e_k.
2. For each cell add t_N to e_k. If $e_k=P_k$ then calculate $\delta(q_k,q_{k,l},q_{k,r})$ and save the result to ν_k.
3. For each cell, if $e_k=P_k$ then set q_k to ν_k and set e_k to zero.
4. Go to step 1.

Self-clocked cellular automata can simulate asynchronous cellular automata that have a fixed update order. Without loss of generality, consider n cells that are updated from left to right. To simulate this update order, the duration of each cell is n and the elapsed time is initially e_k=n-k. The cell at position k has its first update at time k. Subsequent updates occur at times k+n, k+2n, … so that in every n time steps each cell is updated just once and in the desired order.

To illustrate this construction, consider an automaton with three cells. The discrete state of each cell is 0 or 1, the transition rule is

$$\delta(q_k,q_{k,l},q_{k,r})=\begin{cases}0 & \text{if } q_{k,l}=1,q_k=1,q_{k,r}=1\\ 0 & \text{if } q_{k,l}=1,q_k=1,q_{k,r}=0\\ 0 & \text{if } q_{k,l}=1,q_k=0,q_{k,r}=1\\ 1 & \text{if } q_{k,l}=1,q_k=0,q_{k,r}=0\\ 1 & \text{if } q_{k,l}=0,q_k=1,q_{k,r}=1\\ 1 & \text{if } q_{k,l}=0,q_k=1,q_{k,r}=0\\ 1 & \text{if } q_{k,l}=0,q_k=0,q_{k,r}=1\\ 0 & \text{if } q_{k,l}=0,q_k=0,q_{k,r}=0\end{cases}$$

and the initial values of the discrete states are $q_1=q_3=0$ and $q_2=1$. Table I shows six time steps of this cellular automaton; in each step, the cell that changes its state is enclosed in a box.

Time	k=1	k=2	k=3	Rule
0	q=0,e=2	q=1,e=1	q=0,e=0	R7
1	q=1,e=0	q=1,e=2	q=0,e=1	R2
2	q=1,e=1	q=0,e=0	q=0,e=2	R7
3	q=1,e=2	q=0,e=1	q=1,e=0	R2
4	q=0,e=0	q=0,e=2	q=1,e=1	R7
5	q=0,e=1	q=1,e=0	q=1,e=2	R2
6	q=0,e=2	q=1,e=1	q=0,e=0	R7

Table I. Six time steps of a self-clocked cellular automaton that simulates a left to right update order.

Self-clocked cellular automata and asynchronous automata with fixed update order are widely used to model multi-agent systems. Applications can be found in fields as diverse as ecology, biology, sociology, and economics. Asynchronous cellular automata are attractive for these applications because they can model asynchronous interactions that occur in the system of interest [Cornforth *et al.*, 2002; Green *et al.*, 2001].

When compared with its synchronously updated counterpart, an asynchronous cellular automaton can exhibit radically different behaviors [Bersini and Detour, 1994; Schönfisch and de Roos, 1999; Stark and Hughes, 2000]. It is surprising then that self-clocked cellular automata seem to share an important property of their synchronous relatives. A synchronous cellular automaton defined over a finite space is periodic because it has a finite number of states. A self-clocked cellular automaton, on the other hand, has an infinite number of potential states because of the elapsed time that is intrinsic to every cell. None the less, it is conjectured here that all self-clocked cellular automata must settle into one of a finite number of limit cycles, the particular choice depending on its initial state.

This conjecture emerges from the theory of hybrid systems, and specifically from the recent work by Matveev and Savkin [2000] that describes the limit cycles of differential automata. The argument begins by showing that self-clocked cellular automata are instances of a class of differential automata that have five critical properties. The theory of hybrid systems shows that these properties are sufficient for the automata to always converge to one of a finite number of limit cycles. The necessary conclusion is that self-clocked cellular automata converge eventually to a limit cycle. Moreover, because self-clocked cellular automata can simulate deterministic asynchronous cellular automata, it is concluded that all deterministic cellular automata – synchronous and asynchronous – are essentially periodic.

1. Brief review of differential automata

Differential automata are finite state automata that have a set of differential equations associated with each discrete state. Discrete events, which change the system's discrete state, occur when the automaton's continuous trajectory encounters an event surface. By changing the discrete state, the event causes the system to select a new set of differential equations which govern its motion away from the interrupting event surface. Figure 1 illustrates the trajectory of a differential automaton with a single continuous variable.

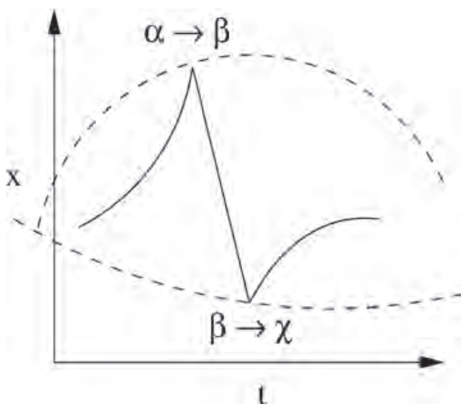


Figure 1. A trajectory of a differential automaton with a single continuous variable × which follows the solid curve, two event surfaces shown as dashed curves, and two discrete transitions $\alpha \rightarrow \beta$ and $\beta \rightarrow \chi$.

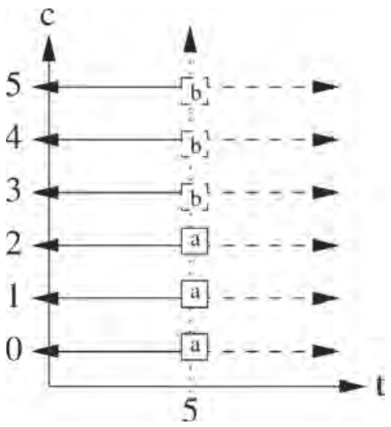


Figure 2. Illustration of the trajectory z that is defined by Equation 4. The value a is indicated with a solid line and b with a broken line.

Differential automata measure time in two dimensions. The first dimension is the real number line, and it measures time by seconds, minutes, and hours. The second dimension counts events that occur at an instant, neatly arranging instantaneous events by the order of their occurrence. Time is modeled by the set $\mathbb{R} \times \mathbb{N}$ and the advance operator $\triangleright$ moves a time (t,c) forward by (t',c') with the rule

$$(t,c) \triangleright (t',c') = \begin{cases} (t+t',0) & \text{if } t' \neq 0 \\ (t,c+c') & \text{if } t'=0 \end{cases} \tag{1}$$

The advance operator is not commutative and it is not associative. Time is ordered first by t and then by c: specifically,

$$(t_1,c_1) < (t_2,c_2) \Leftrightarrow ((t_1 < t_2) \vee (t_1 = t_2 \wedge c_1 < c_2)) \tag{2}$$

$$(t_1,c_1) = (t_2,c_2) \Leftrightarrow (t_1 = t_2 \wedge c_1 = c_2) \tag{3}$$

A trajectory of a hybrid system is a function from $\mathbb{R} \times \mathbb{N}$ to its set of states. The trajectory, by virtue of being a function, has a single value at each point in time; the second dimension of time allows for instantaneous changes in the discrete state of the system. For example, the trajectory

$$z((t,c)) = \begin{cases} a & \text{if } (t,c) \leq (5,2) \\ b & \text{if } (t,c) \geq (5,3) \end{cases} \tag{4}$$

has this property; it is illustrated in Figure 2.

A differential automaton is a system with a finite set Q of discrete states, a set $\mathbb{R}^m$ of continuous state vectors, and two functions that describe its dynamic behavior. The differential function $f : \mathbb{R}^m \times Q \rightarrow \mathbb{R}^m$ describes how the continuous variables evolve between discrete events. The transition function $\Phi : \mathbb{R}^m \times Q \rightarrow Q$ defines the event surfaces and their effect on the discrete state. The continuous state vector $\mathbf{x}$ satisfies

$$\frac{d}{dt}\mathbf{x}(t) = f(\mathbf{x}(t), q) \tag{5}$$

at each instance of real time for which q is constant.

When q changes at time (t,c), its subsequent value at time (t,c+1) is

$$q((t, c+1)) = \Phi(\mathbf{x}((t, c)), q((t, c))) \tag{6}$$

The discrete change in the differential function takes effect at time (t,c)+(0,1)=(t,c+1) and $\mathbf{x}$ evolves from its value at the transition. Because $\mathbf{x}$ does not change discontinuously, the trajectory $\mathbf{x}(t)$, which satisfies Equation 5 and is a function from $\mathbb{R} \rightarrow \mathbb{R}^m$, is equal to $\mathbf{x}((t,c))$, a function from $\mathbb{R} \times \mathbb{N} \rightarrow \mathbb{R}^m$, for all $\mathbf{c} \in \mathbb{N}$, and so the two technically distinct trajectories can be used interchangeably.

The *total* state transition function Δ_{da} takes the system from an initial state $(x(t_0),q)$ through an interval $[(t_0,c_0),(t_f,c_f))$. Defining first

$$g(\mathbf{x},q,h)=\mathbf{x}+\int_0^h f(\mathbf{x}(t),q)\,dt, \tag{7}$$

the function Δ_{da} is defined recursively by

$$\begin{aligned}
&\Delta_{da}((\mathbf{x},q),[(t,c),(t_f,c_f))) = \\
&\quad 1)\ (\mathbf{x},q) \text{ if } [(t,c),(t_f,c_f)) \text{ is empty} \\
&\quad 2)\ \Delta_{da}((g(\mathbf{x},q,t'-t),q),[(t',c'),(t_f,c_f))) \\
&\qquad \text{if } (\forall \hat{t} \in [t,t'))(\Phi(g(\mathbf{x},q,\hat{t}-t),q)=q) \\
&\qquad \text{where } (t',c') \text{ is the smallest in } [(t,c),(t_f,c_f)] \\
&\qquad \text{such that } \Phi(g(\mathbf{x},q,t'-t),q) \neq q \lor (t',c')=(t_f,c_f) \\
&\quad 3)\ \Delta_{da}((\mathbf{x},\Phi(\mathbf{x},q)),[(t,c)+(0,1),(t_f,c_f))) \\
&\qquad \text{if } \Phi(\mathbf{x},q) \neq q
\end{aligned} \tag{8}$$

This abstruse definition bridges a technical, but critical, gap between self-clocked cellular automata and the differential automata that are constructed in the next section; we will return to Equation 8 in Section 3.

2. Modeling self-clocked cellular automata with differential automata

Every self-clocked cellular automaton is a homomorphic image (see, e.g., [Zeigler, 2000]) of a differential automaton whose trajectories converge to a finite number of limit cycles. That self-clocked cellular automata are essentially periodic follows this fact. In this section, the relevant class of differential automata is constructed.

The discrete states of a binary cellular automaton can be numbered by treating its leftmost cell as the most significant bit in a binary number and its rightmost as the least significant. An automaton with n cells has 2^n discrete states. In addition to its binary state, each cell k has a clock τ_k that induces a change of state under two conditions: beginning from zero, τ_k grows until it reaches P_k, τ_k shrinks until it reaches 0. The direction of the clock is the cell's third and final state variable.

The pair (b_k,d_k) is the discrete state of cell k, where $b_k \in \{0,1\}$ is the binary state and $d_k \in \{1,-1\}$ is the direction of the clock; the set of discrete states is $Q=\{0,1\}\times\{1,-1\}$. The clock τ_k begins, and remains, in the interval $[0,P_k]$. The differential automata that models the cell is

$$\frac{d\tau_k}{dt}=d_k \tag{9}$$

$$\Phi_k(\tau_k,(b_k,d_k)) = \begin{cases} (\delta(b_k,b_{k,l},b_{k,r}),-d_k) & \text{if } (\tau_k = 0 \wedge d_k = -1) \\ & \vee(\tau_k = P_k \wedge d_k = 1) \\ (b_k,d_k) & \text{otherwise} \end{cases} \tag{10}$$

A cellular automaton with n cells is a differential automaton with the set of discrete states Q^n and the continuous state vector $\tau=[\tau_1\ \tau_2\ \dots\ \tau_n]$ for which each component τ_k is in $[0,P_k]$. The dynamic equations for this model are

$$\frac{d\tau}{dt}=[d_1\ d_2\ \dots\ d_n] \tag{11}$$

$$\Phi(\tau,(b_1,d_1),\dots,(b_n,d_n)) = ((b'_1,d'_1),\dots,(b'_n,d'_n)) \tag{12}$$

where $(b'_k, d'_k) = \Phi_k(\tau_k, (b_k, d_k))$ (13)

This differential automaton has five important properties: 1) it is deterministic; every initial state generates a single trajectory, 2) the model is legitimate (*i.e.*, non-Zeno), 3) the vector $d\tau/dt$ is constant between discrete events, 4) Q^n is a finite set, and 5) the event surfaces form hyper-cubes in its phase space. The fourth and fifth conditions confine the continuous trajectories of the cellular automaton to a subset K of $\mathbb{R}^n$. The first, second, and third conditions imply that the discrete trajectories are well behaved: the system is predictable, always moves forward on its real time line, and between events follows straight lines through $\mathbb{R}^n$.

Matveev and Savkin [2000] show that these properties have the following consequences: (i) there exists a limit cycle lying in K, (ii) the number of such cycles is finite, (iii) any limit cycle lying in K is regularly locally asymptotically stable in K, and (iv) any trajectory lying in K regularly converges to one of these limit cycles. Informally, every trajectory of the cellular automaton converges to a periodic trajectory as the real time t goes to infinity. Both the continuous and discrete variables are eventually periodic! Moreover, there are a finite number of these limit cycles, and so they act as distinct equilibrium trajectories for the system.

There are exactly two cellular automata with a single cell, and these give the simplest demonstration of the theory. The cell has a duration P. It is its own left and right neighbor, and so δ is entirely defined by its action on the triples (1,1,1) and (0,0,0); for brevity δ is written as a function of a single value. Two transition rules can be defined:

$$\delta_\alpha(b) = b \text{ and}$$

$$\delta_\beta(b) = \begin{cases} 1 & \text{if } b=0 \\ 0 & \text{if } b=1 \end{cases}$$

Both automata, the first with rule δ_α and the second with rule δ_β, have a pair of limit cycles. These are shown in Figure 3. The event surfaces are lines at t=0 and t=P. Beginning with a direction d=1, the clock moves up to P where a discrete event occurs and causes the direction to change; it then moves to 0 where the direction changes again; and the cell bounces back and forth between these two constraining surfaces. Both automata need two bounces to return to their initial states and so have a period of length 2P.

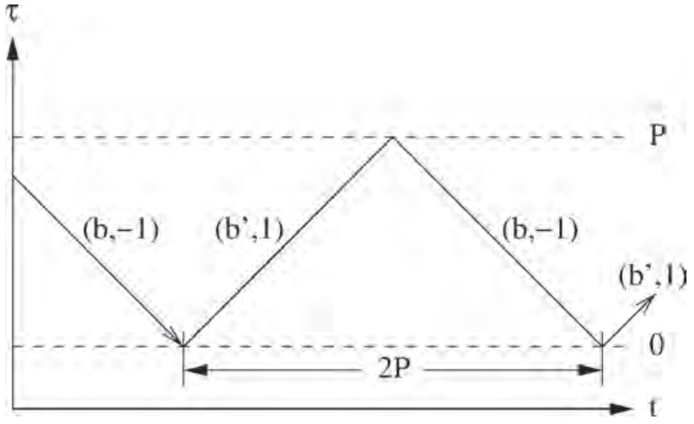


Figure 3. Event surfaces and periodic trajectories of two self-clocked cellular automata each with a single cell; in this drawing, $b'=\delta_\alpha(b)$ and $b=\delta_\alpha(b')$ for the first automaton, and similarly with respect to δ_β for the second automaton.

3. A homomorphism from differential automata to self-clocked cellular automata

The simulation procedure described to the Introduction can be reformulated as a recursive state transition function. A self-clocked cellular automaton with n cells has the set of states $S = \prod_{k=1}^{n}(\{0,1\} \times \mathbb{R})$, and in the state $s=(\ldots,(q_k,e_k),\ldots)$ each pair (q_k,e_k) describes a single cell. The real time remaining to the next state transition is given by the time advance function

$$ta(s) = \min_{k\in[1,n]} P_k - e_k \tag{14}$$

The state transition function that takes the system from the state s through an interval $[(t,c),(t_f,c_f))$ is

$$\begin{aligned}
&\Delta_{ca}(s,[(t,c),(t_f,c_f))) = \\
&\quad 1)\ s \text{ if } [(t,c),(t_f,c_f)) \text{ is empty} \\
&\quad 2)\ \Delta_{ca}((\ldots,(q_k,e_k + t'-t),\ldots),[(t',c'),(t_f,c_f))) \\
&\qquad \text{if } (\forall k)(e_k < P_k), \\
&\qquad \text{where } (t',c') \text{ is the smallest in } [(t,c),(t_f,c_f)] \\
&\qquad \text{such that } t'-t = ta(s) \vee (t',c') = (t_f,c_f) \\
&\quad 3)\ \Delta_{ca}((\ldots,(q'_k,e'_k),\ldots),[(t,c)+(0,1),(t_f,c_f))) \\
&\qquad \text{if } (\exists e_k)(e_k = P_k)
\end{aligned} \tag{15}$$

$$\text{where } (q'_k,e'_k) = \begin{cases} (q_k,e_k) & \text{if } e_k < P_k \\ (\delta(q_k,q_{k,l},q_{k,r}),0) & \text{if } e_k = P_k \end{cases} \tag{16}$$

The state transition function Δ_{ca} acts exactly like the iterative procedure from which it is derived: the elapsed times are incremented by the time remaining to next transition; the next state of the active cells is computed; and the process repeats.

With this formalism and the formalism of Equation 8, it can be shown that the self-clocked cellular automaton is a homomorphic image of the differential automaton constructed in Section 2. The state of the differential automaton is mapped into a state of the cellular automaton by

$$H((\ldots,(\tau_k,(b_k,d_k)),\ldots)) = (\ldots,(b_k,\varepsilon_k),\ldots) \tag{17}$$

$$\text{where } \varepsilon_k = \begin{cases} P_k - \tau_k & \text{if } d_k = -1 \\ \tau_k & \text{if } d_k = 1 \end{cases} \tag{18}$$

Using Equation 18, the time advance of the cellular automaton can be written

$$ta(H((\ldots,(\tau_k,(b_k,d_k)),\ldots))) = \min_{k\in[1,n]} \frac{1}{2}((P_k - \tau_k)(d_k + 1) - \tau_k(d_k - 1)) \tag{19}$$

To show that H is a homomorphism, it is enough to consider in turn cases 1, 2, and 3 in Equations 15 and 8. Case 1 is trivial. For cases 2 and 3, first observe that under H, the switching surfaces of the differential automaton define the time advance function of the cellular automaton. Specifically, the event surface that will be encountered next by the differential automaton is defined by

$$\min_{k\in[1,n]} \frac{1}{2}((P_k - \tau_k)(d_k + 1) - \tau_k(d_k - 1)) = 0 \tag{20}$$

and the time to reach this surface is $ta(H((\ldots,(\tau_k,(b_k,d_k)),\ldots)))$. Therefore, h in Equation 8 and ta(s) in Equation 15 are equal; the hybrid system in state γ and discrete event system in state $H(\gamma)$ undergo their next events at the same moment.

Keeping now in mind that h=ta(s), consider cases 2 and 3 in turn. For case 2, it is sufficient that simulation over the interval $\omega=[(0,0),(h,1))$ gives

$$H(\Delta_{da}((\ldots,(\tau_k,(b_k,d_k)),\ldots),\omega)) = \Delta_{ca}(H((\ldots,(\tau_k,(b_k,d_k)),\ldots)),\omega) \quad (21)$$

Now consider just the action of H, Δ_{da}, and Δ_{ca} on the single cell k. For the left side of Equation 21

$$H(\Delta_{da}((\tau_k,(b_k,d_k)),\omega)) =$$
$$H((\tau_k + hd_k,(b_k,d_k))) = \begin{cases} (b_k, P_k - \tau_k + h) & \text{if } d_k = -1 \\ (b_k, \tau_k + h) & \text{if } d_k = 1 \end{cases}$$

and for the right side

$$\Delta_{ca}(H((\tau_k,(b_k,d_k))),\omega) = \begin{cases} \Delta_{ca}((b_k, P_k - \tau_k),\omega) & \text{if } d_k = -1 \\ \Delta_{ca}((b_k,\tau_k),\omega) & \text{if } d_k = 1 \end{cases}$$

If $d_k=-1$ then

$$\Delta_{ca}((b_k, P_k - \tau_k),\omega) = (b_k, P_k - \tau_k + h)$$

and if $d_k=1$ then

$$\Delta_{ca}((b_k,\tau_k),\omega) = (b_k,\tau_k + h)$$

just as desired.

For case 3, it is sufficient that Equation 21 holds when ω is replaced by the interval $\omega_\phi=[(0,0),(0,1))$. For each cell there are three cases to consider. Case (i): if $0<\tau_k<P_k$ then the discrete state of the differential automaton does not change. Because h=ta(s), it is also true that $e_k<P_k$ and so the discrete state of the cellular automaton does not change. Therefore Equation 21 holds. Case (ii): If $0=\tau_k$ and $d_k=-1$ then the differential automaton changes its discrete state and, for H to be a homomorphism, the cellular automaton must as well. Again, because h=ta(s), $0=\tau_k$ and $d_k=-1$ is equivalent to $e_k=P_k$. Hence, Equation 21 holds. Case (iii): $P_k=\tau_k$ and $d_k=1$; the argument here is identical to Case (ii). Therefore, H is a homomorphism.

This is the main result. To summarize, the differential automata constructed in Section 2 are essentially periodic; every self-clocked cellular automaton is a homomorphic image of one of these differential automaton; therefore, every self-clocked cellular automaton is essentially periodic.

4. Discrete event simulation of self-clocked cellular automata

Every self-clocked cellular automaton can be written as a discrete event system in the terms of the Discrete Event System Specification (DEVS) and very efficiently executed using a DEVS simulation engine [Zeigler, 2000]. Each cell is an atomic model with states (b,b_l,b_r,ε) where $b,b_l,b_r\in\{0,1\}$ and $\varepsilon\in[0,P]$. The model's set of input is $\{l,r\}\times\{0,1\}$ where l is the left neighbor, r is the right neighbor, and $\{0,1\}$ is the neighboring state. The set of output is $\{0,1\}$. Letting $q=(b,b_l,b_r,\varepsilon)$, the dynamic behavior of a cell is defined by

$$\delta_{int}(q) = (\delta(b,b_l,b_r),b_l,b_r,0)$$
$$\delta_{ext}(q,e,x^b) = (b,b_l',b_r',\varepsilon+e)$$
$$\text{where } b_l' = \begin{cases} \hat{b} & \text{if } (l,\hat{b})\in x^b \\ b_l & \text{otherwise} \end{cases}$$
$$\text{and } b_r' = \begin{cases} \hat{b} & \text{if } (r,\hat{b})\in x^b \\ b_r & \text{otherwise} \end{cases}$$
$$\delta_{con}(q,x^b) = \delta_{ext}(\delta_{int}(q),0,x^b)$$
$$\lambda(q) = \delta(b,b_l,b_r)$$
$$ta(q) = P_k - \varepsilon$$

This model has the binary state b of the cell, the binary states b_l and b_r of its neighbors at their last transition, and the elapsed time ε. When the time advance ta expires, the model produces as output its next binary state, sets b to this new value, and resets the elapsed time. The cell can receive input from its neighbors at any time, and when this occurs it records their binary states and increments δ by the time e that has elapsed since its last event (*i.e.*, change of state by δ_{int}, δ_{ext}, or δ_{con}).

The clock of the differential automata can be recovered by adding the direction d and time t_L of the last event to the state variables of this model. Initially, d=1 and it is multiplied by -1 by the internal transition function. The time t_L of the last event is initially zero, and it is incremented by e in the external transition function and by the time advance in the internal transition function. The clock τ at any time t is then

$$\tau = \frac{1-d}{2}P + d(\varepsilon + t - t_L)$$

5. Illustrations of the theory

In a left looking cellular automaton, each cell takes its next state from its left neighbor. The transition function is

$$\delta(q, q_l, q_r) = q_l$$

and when P=1 the configuration of the cell space simply translates to the left at each step. Figure 4 shows 125 steps of this model beginning with alternating 1, colored black, and 0, colored white, cells. The periodicity of this model is immediately apparent.

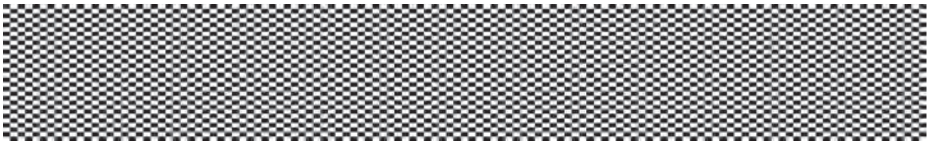


Figure 4. Discrete trajectory of the left looking automaton with P=1. The cells are arranged from bottom to top and time increases from left to right.

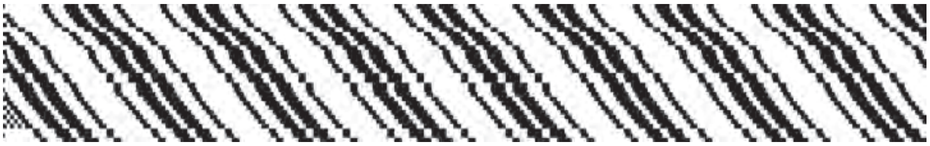


Figure 5. Discrete trajectory of the left looking automaton with P selected for each cell at random from 1/2, 1/3, and 1.

Figure 5 shows 125 units of time for the left looking model with P selected at random from 1/2, 1/3, and 1. The trajectory of this automaton initially appears to be irregular, but at the sixth set of bands settles into a recognizable pattern. The continuous phase space of this model has three dimensions: the first for cells with P=1/2, the second for cells with P=1/3, and the third for cells with P=1. Figure 6 shows planar cuts of the phase space, and these clearly depict the periodicity of the clock variables.

This same experiment was repeated with P selected randomly from $\sqrt{2}$, $\sqrt[3]{2}$, and 2. A recognizable pattern takes much longer to appear in this case. Figure 7 shows the discrete trajectory for the first 125 units of time. At first, the behavior of the automaton is quite erratic, but a structure seems ultimately to assert itself. This is illustrated in Figure 8, which shows the discrete trajectory from time 15000 to 15125.

The phase space of this automaton is more intricate than the previous two. The $\sqrt{2} - \sqrt[3]{2}$ plane is shown in Figure 9(a), where a regular pattern of diamonds is quite visible. In fact, there appear to be close spaced nets of diamonds, which produce a distinct pattern. These can be seen in the enlarged view of this plane shown in Figure 9(b). It should be noted, however, that these are not snapshots of a periodic trajectory. Because the clocks of the individual elements lack a common period, their trajectories will completely fill the phase plane.

Conclusions

Small systems, with components having rational durations, move quickly into their intrinsic limit cycles, but these may take a very long time to appear if the phase space of the model has many dimensions or if it has many discrete states. Patterns seen in short-term observations might reflect unique, transient effects or might be part of a long limit cycle.

For automata whose components have irrational durations, however, there is no genuinely periodic trajectory. As the demonstrations above show, the trajectory may appear more or less periodic after some time, and may in fact follow a more or less predictable path indefinitely. That is, it appears to be converging to some periodic trajectory, but if so, the theory developed by Matveev and Savkin [2000] (a brief summary is given in [Matveev and Savkin, 1999]) does not say what that trajectory is, stating only that it exists.

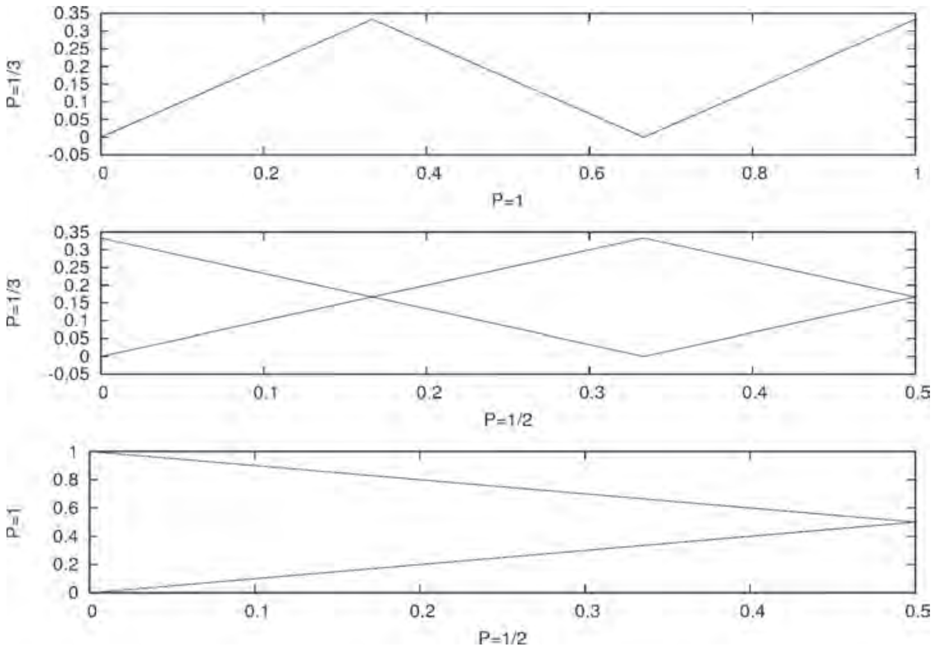


Figure 6. Planar cuts of the phase space corresponding to the discrete trajectory shown in Figure 5.

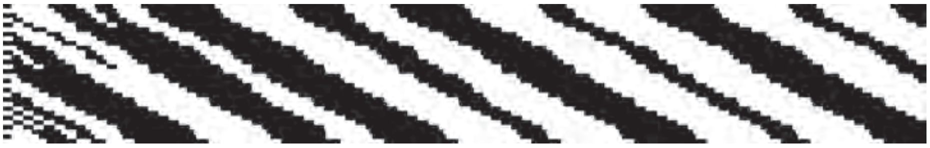


Figure 7. Discrete trajectory over the real interval [0,125] of the left looking automaton with P selected for each cell at random from $\sqrt{2}$, $\sqrt[3]{2}$, and 2.

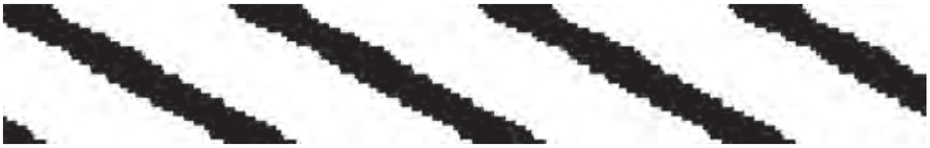
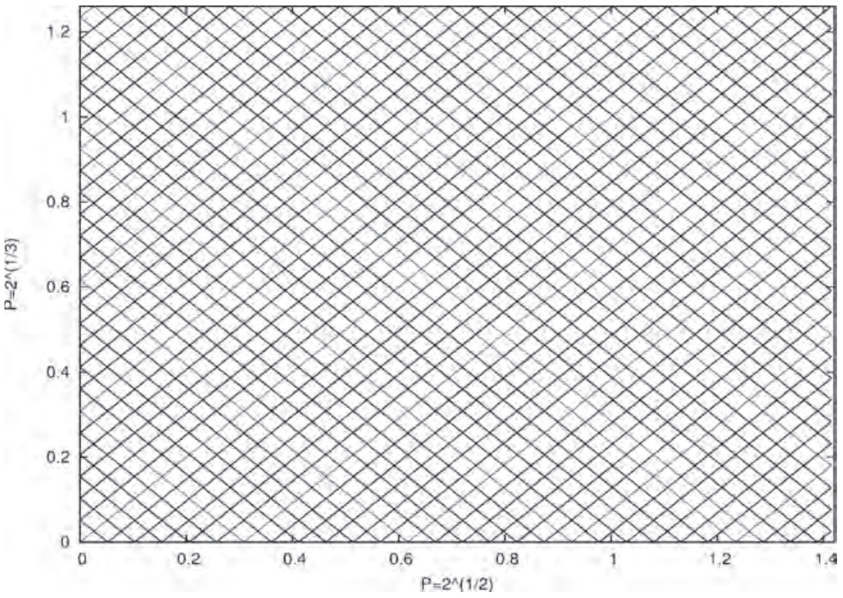
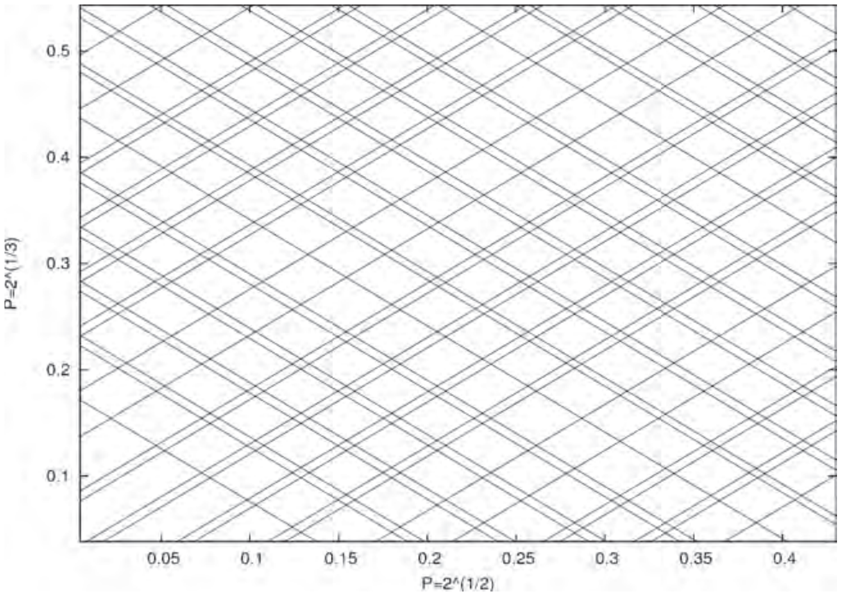


Figure 8. Discrete trajectory over the real interval [15000,15125] for the same automaton shown in Figure 7.



9(a) View of the entire phase plane.



9(b) Enlarged view.

Figure 9. The $\sqrt{2} - \sqrt[3]{2}$ phase plane corresponding to the discrete trajectory shown in Figure 8.

This seeming contradiction, between the absence of genuine periodicity and the claim of convergence to a limit cycle, raises three questions:

1. Do the asynchronous automata in fact satisfy the required assumptions, or has some critical point been overlooked?
2. Is an assumption missing from the underlying theory, which if invoked would rule out the asynchronous, cellular automata described above?
3. Is there in fact a limit cycle (defined in [Matveev and Savkin, 2000; 1999] as a class of periodic trajectories) that the system converges too and, if so, what are those trajectories?

With respect to questions 1 and 2, it can be observed that cellular automata are unusual amongst differential automata most often considered in that the real space occupied by distinct, discrete states overlap. This might provide a clue to either the missing assumption (*i.e.*, a positive answer to 2) or the point of divergence from the theory (*i.e.*, a positive answer to 1).

If the conjecture in this paper holds, it has further consequences for the class of Schedule Preserving DEVS (SP DEVS) described by Hwang [Hwang, 2005]. If a SP DEVS has a finite number of discrete states, then in each state the model has a fixed duration. It is likely, therefore, that the same homomorphism developed above for the cellular automata can be extended to this larger class of systems, and so it can be conjectured that all SP DEVS with a finite set of states are essentially periodic. Even if this broad conjecture fails by its encompassing irrational durations, the same conjecture may still hold if only rational durations are considered.

References

[Bersini and Detour, 1994] – Bersini H. and Detour V., Asynchrony induces stability in cellular automata based models. *In: Artificial Life IV*, 1994, pp. 382-387.

[Cornforth *et al.*, 2002] – Cornforth D., Green D., Newth D. and Kirley M., Do artificial ants march in step? Ordered asynchronous processes and modularity in biological systems. *In: Artificial Life VIII*, 2002, pp. 28-32.

[Gardner, 1970] – Gardner M., The fantastic combinations of John Conway's new solitaire game: Life. *Scientific American*, No. 223, 1970, pp. 120-123.

[Green *et al.*, 2001] – Green D., Newth D., Cornforth D. and Kirley M., On evolutionary processes in nature and artificial systems. *In: Proceeding of the Fifth Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, 2001, pp. 1-10.

[Hwang, 2005] – Hwang M. H., Tutorial: Verification of Real-time System Based on Schedule-Preserved DEVS. *In: Proceedings of 2005 DEVS Symposium*, San Diego, Apr. 2-8, 2005.

[Matveev and Savkin, 1999] – Matveev A. and Savkin A., Existance and stability of limit cycles in hybrid dynamical systems with constant derivatives. Part 1. General theory. *In: Proceedings of the 38th Conference on Decision & Control*, 1999, pp. 284-289.

[Matveev and Savkin, 2000] – Matveev A. and Savkin A., *Qualitative Theory of Hybrid Dynamical Systems*, Birkhauser. 2000.

[Schönfisch and de Roos, 1999] – Schönfisch B. and de Roos A., Synchronous and asynchronous updating in cellular automata. *Biosystems*, Vol. 51, No. 3, 1999, pp. 123-143.

[Stark and Hughes, 2000] – Stark R. and Hughes W., Asynchronous, irregular automata nets: the path not taken. *Biosystems*, Vol. 55, No. 1-3, 2000, pp. 107-117.

[Zeigler, 2000] – Zeigler B. P., Prauhofer H. and Kim T. G., *Theory of Modeling and Simulation*, Academic Press, 2000 (2nd ed.).

Conclusion

Currently, the art of creation of models benefits from tools coming from well established scientific disciplines. Some of these disciplines focus on the laws of life (e.g., Biology and Ecology), others focus on the laws of matter (e.g., Physics and Geology), some on the laws of societies (e.g., Sociology and Economics), others on the laws of human cognition (e.g., Artificial Intelligence and Psychology.) All these disciplines try, autonomously, to expand the explanatory boundaries of their models through simulation. At the same time, some computer scientists collaborate with specialists from different disciplines to confront and improve their own tools. What emerges from this current evolution is that computation-based Modeling & Simulation is today facing the challenge to improve modeling tools, toward increased multiphysical, multiformalism, multiperspective, multiscaled and/or integrated models. From this viewpoint, it clearly appears that there is a common need for more sophisticated and integrating modeling tools.

What should these tools be? First, they have to be abstract enough to become domain-specific. A well accepted common structure is the general system inherited from systems theory. The mathematical description (developed by Mesarovic, Klir, and Wymore) of the general system has proved to be worthy of specification through Zeigler's discrete-event systems. These computational systems are really the fundamental materials for the construction of computational models. However, if the discrete-event system specifications are the cells to build the organs, new plans and knowledge have to tell modelers how to connect these cells, how these cells can interact in an autonomous efficient way to constitute an organ, and how these organs can interact to constitute a body.

Systems are increasingly fine-grained and miniaturized. Hardware is perpetually improving its efficiency. Grids, distributed and parallel simulations are more and more massive. Disciplines use these interacting distributed systems, through more and more individual-based models and interacting parts of codes. However, tools are rare for the study of simulation results[1] and there is little information about how to link: existing powerful discipline models (e.g., statistical physics and Entropy, or Operational research and optimization tools) and generative systems. Moreover, although multi-agent systems contributed to the emergence of new locally interacting control mechanisms, to achieve a common goal, a broader framework embedding both autonomous and automatic control of systems needs to be developed. What should these new structures be? From which other structures can they be inspired?

It can be accepted that whatever our models of Reality, it will be difficult to discover the true nature of the laws of Reality. Hence, we can follow Hume, for instance, when he claims that every idea comes from empirical reality, through a process of abstraction. More than empiricism, the necessary shift from artificial intelligence to artificial life tells us that we now need to take into account, both the laws of life and the laws of matter simultaneously, to be able to manipulate and understand them. Therefore, models need to be more and more related to, inspired from, based on, and compared to Reality. This sounds like euphemism. However, precisely, our efforts should be concentrated on the forms, mechanisms, algorithms translating reality into models.

But which mechanisms characterize the physical and natural laws of Reality?

Here, are certainly some:

- Causality and contingency of events in time and space,
- Action and reaction (which relates to causality),
- There is no free lunch ;-): Every action has a cost and a possible benefit. Every physical and chemical reaction consumes and produces energy. For any one action, a trade-off exists between the matter (to be) consumed and the energy (to be) produced, the cost and the benefit of this action. - For sustainability, fidelity and efficiency, there exists an optimal allocation of resources between usages and resource availability within the simulation to reach its goal.

1. Simulators generate massive data through massive process interactions. Operational tools need to be developed to deal with the spatiality, the interactions, the correlations, as well as the pertinence of these data. The hidden meaning behind these data (to enhance our knowledge of the model) needs to be revealed.

At the beginning of this new numerical century, the challenge of the theory of modeling and simulation will definitely be to try integrating knowledge of all scientific disciplines to be more successful than the physico-mathematicians of the past century (such as Rashevsky or Rosen) in abstracting the main physical and natural mechanisms of Reality, integrating knowledge of all scientific disciplines. What a challenge!

Alexandre MUZY, Franck VARENNE,
Patrick COQUILLARD & David R. C. HILL