

MULTI-AGENT BASED MODELLING: FROM SOCIAL SIMULATION TO REAL TIME STRATEGY GAMES

Marco Remondino
Department of Computer Science
University of Turin
10149 Torino, Italy
E-mail: remond@di.unito.it

KEYWORDS

Intelligent agent, simulation, genetic algorithm, classifier system, strategy game

ABSTRACT

Simulation has been regarded as the third way to represent social models, alternative to other two symbol systems: the verbal argumentation and the mathematical one. Simulation can be processed by a computer and is particularly suited for complex systems, in which the aggregate behaviour is not necessarily the sum of the single parts. Agent Based Modelling (ABM) is for sure the most interesting and advanced approach for simulating a complex system: in a social context, the single parts and the whole are often very hard to describe in detail; by using intelligent agents as basic building blocks, there are formalisms which allow to study the emergency of social behaviour through the creation of models, known as "artificial societies". This paper deals with an hybrid agent based methodology, borrowed from the social sciences application field, which could be successfully applied to real-time strategy games; this would create a realistic environment and a less deterministic behaviour, thanks to the AI technology embedded in the hybrid approach.

INTRODUCTION

According to (Ostrom 1988), simulation can be considered a third way to represent social models; in particular, it can be a powerful alternative to other two symbol systems, the verbal argumentation and the mathematical one. The former is, of course, a non computable way of modelling, though a highly descriptive one. As to the mathematical argumentation, everything can be done with equations, in principle, but the complexity of differential equations increases exponentially as the complexity of behaviour increases. Describing complex individual behaviour with equations often becomes intractable. Simulation has a great advantage over the other two, which is to be found in its high portability on a computer, through a program or a particular tool, and in the possibility of describing complex behaviour starting from simple interacting entities. Computer programs can then be used to model either quantitative theories or qualitative ones. Since real time strategy games are more and more complex, and take place in dynamic worlds in which complex decisions, often based on partial knowledge must be made,

the artificial intelligence behind them can be modelled with agent base techniques, already used in social simulation. In particular, a hybrid methodology will be discussed, which is particularly fitted for those situations in which some parts of the environment are strictly deterministic, while others must act basing their decisions on the interaction among them and the environment itself.

DIFFERENT KINDS OF AGENTS

Agent Based Modelling (ABM) is for sure the most interesting and advanced approach for simulating a complex system: in a social context, the single parts and the whole are often very hard to describe in detail; by using intelligent agents as basic building blocks, there are formalisms which allow to study the emergency of social behaviour through the creation of models, known as "artificial societies". Thanks to the ever increasing computational power, it's been possible to use such models to create software, based on such intelligent agents, which aggregate behaviour is often complex and difficult to predict, and which can be used in open and distributed systems. A software agent can be described as a flexible system, capable of dynamic, autonomous actions in order to meet its design objectives, that is situated in some environment. The main features for a software agent are: situatedness, that is ability to perform actions according to a particular input received from outside, which can, in turn, change the environment itself; autonomy in performing actions, without intervention of humans; flexibility and adaptability. Some particular agents can also be proactive, which means they are goal-directed, and social, in the way they can interact with other artificial agents, robots, and humans. Such an intelligent agent can be referred to as a *Belief-Desire-Intention* (BDI) one. There are many agent based paradigms that can be applied to simulation:

- *Symbolic*: highly structured agents, described through expressions of modal logic. This is perfect when there is only a single agent, which must interact with the environment, but it's not versatile when used to simulate big communities
- *Sub-symbolic*: many simple (not structured) agents which interact among them and with the environment. A multi-agent context of this kind allows the emergency of complex behaviour and self-organization. Intelligent behaviour is a product of the interaction among agents and environment, and of the interaction among many

simple behaviours. It can be really hard to describe the real world under every aspect: some fundamental macro-actions can thus be defined on single agents, which allow cooperation with the environment and with other agents. The concept of Multi Agent System for Social Simulations is thus introduced: the single agents have a very simple structure. Only few details and actions are described for the entities: the behaviour of the whole system is a consequence of those of the single agents, but it's not necessarily the sum of them. This can bring to unpredictable results, when the simulated system is studied.

- *Hybrid Architectures*: at the lower levels, we find reactive agents, like the ones described above, while at the upper levels there are more complex and structured agents. In this way, we can combine reactive capabilities with planning.

In some situations, effective results can be obtained just by building simple, sub-symbolic agents, whose behaviour is randomly determined or is built by applying fixed pre defined reaction rules; this is the case, for instance, of Heatbugs, one of the canonical Swarm demonstrations (www.swarm.org):

"It's an example of how simple agents acting only on local information can produce complex global behaviour. As we read on Swarm main site, each agent in this model is a heatbug. The world has a spatial property, heat, which diffuses and evaporates over time. In this picture, green dots represent heatbugs, brighter red represents warmer spots of the world. Each heatbug puts out a small amount of heat, and also has a certain ideal temperature it wants to be. The system itself is a simple time stepped model: each time step, the heatbug looks moves to a nearby spot that will make it happier and then puts out a bit of heat. One heatbug by itself can't be warm enough, so over time they tend to cluster together for warmth"

EVOLUTIONARY METHODS

This is a useful approach when we wish to simulate situations in which we give the rules of the environment and we want to observe some emerging aggregate behaviour arising from simple entities; of course, the way the agents will act tends to be deeply dependent on the choices made by the programmer. As an alternative we can choose to create agents with the ability to compute rules and strategies, and evolve according to the environment in which they act; in order to model them, we can use some methods derived from the studies on artificial intelligence (AI), such as artificial neural networks and evolutionary algorithms. While the former is a collection of mathematical functions, trying to emulate nervous systems in the human brain in order to create learning through experience, the latter derives from observations of biological evolution. Genetic Algorithms (GA) are inspired by Darwin's theory of evolution, often explained as "survival of the fittest": individuals are modelled as strings of binary digits and are the encode for the solution to some problem. The first generation of individuals is often created randomly, and then some fitness

rules are given (i.e. better solutions for a particular problem), in order to select the fittest entities. The selected ones will survive, while the others will be killed; during the next step, a crossover between some of the fittest entities occurs, thus creating new individuals, directly derived from the best ones of the previous generation. Again, the fitness check is operated, thus selecting the ones that give better solutions to the given problem, and so on. In order to insert a random variable in the genetic paradigm, that's something crucial in the real world, a probability of mutation is given; this means that from one generation to the next one, one or more bits of some strings can change randomly. This creates totally new individuals, thus not leaving us only with the direct derivatives of the very first generation. GA have proven to be effective problem solvers, especially for multi-parameter function optimization, when a near optimum result is enough and the real optimum is not needed. This suggests that this kind of methodology is particularly suitable for problems which are too complex, dynamic or noisy to be treated with the analytical approach; on the contrary, it's not advisable to use GA when the result to be found is the exact optimum of a function. The risk would be a convergence to some results due to the similarity of most the individuals, that would produce new ones that are identical to the older ones; this can be avoided with a proper mutation, that introduces in the entities something new, not directly derived from the crossover and fitness process. In this way, the convergence should mean that in the part of the solution space we are exploring there are no better strategies than the found one. It's crucial to choose the basic parameters, such as crossover rate and mutation probability, in order to achieve and keep track of optimal results and, at the same time, explore a wide range of possible solutions.

Classifier Systems (CS) derive directly from GA, in the sense that they use strings of characters to encode rules for conditions and consequent actions to be performed. The system has a collection of agents, called classifiers, that through training evolve to work together and solve difficult, open-ended problems. They were introduced in (Holland 1976) and successfully applied, with some variations from the initial specifics, to many different situations. The goal is to map if-then rules to binary strings, and then use techniques derived from the studies about GA to evolve them. Depending on the results obtained by performing the action corresponding to a given rule, this receives a reward that can increase its fitness. In this way, the rules which are not applicable to the context or not useful (i.e. produce bad results) tend to loose fitness and are eventually discarded, while the good ones live and merge, producing new sets of rules.

FROM REAL MODELS TO STRATEGY GAMES

Strategy games are those in which the player must manage and control military units, workers, resources and so on, and is generally charged with choices and tasks (construction, conquest, organization, etc.) in order to reach a main objective. There is of course an environment, which sometimes can be changed by the actions taken by the player himself, and other actors, that can also be human players or, more generally, artificial entities, managed by some form of

AI. There are mainly two classes of strategy games: *Turn Based* and *Real Time*. While the former category is not very interesting for this paper, the latter is the one in which many actions (issued by different entities) take place in parallel. That's where many similarities arise with real time simulators of real world situations; we may think, for example, to some stock market simulations: by observing the general trend of the artificial stock market created with some basic rules, one can be amazed, by seeing that it resembles in many ways a real one. The market can be simulated by creating some different types of intelligent agents, which follow inner rules; some of them can simply act randomly, while others will "study" the trend before acting. Some of them could even use advanced techniques, such as stop loss. We can now set up things such as of one of these agents is indeed a human player, and we have a sort of real time strategy game, in which the main objective is to become richer and richer, while other computer driven entities try to pursue the same target.

Other interesting examples can be found into the enterprise simulation field; here we have mainly three techniques to model enterprises:

- *Process Based*: used to model a very well structured and known situation, in order to perform a what-if analysis: it's used to create models of parts of enterprises or mechanical/electronic systems. Its greatest advantage is that it starts from a basic scheme, often derived from existent documents, through which it becomes very easy to bring a real situation into a process simulator: usually, a model to be used for process simulation looks like a flow chart, in which a token passes from one box to another one, in a deterministic way, on the basis of the given rules. This kind of approach is widely spread and allows to deeply analyze a part of a whole, studying the expected behaviour of a system, when some change is operated. This is why process simulation is a great support to decisions; the simulator can answer to many questions and what-if problems, that would require big efforts in the real environment; for example, a part of a manufacturing plant can be simulated, by dividing it into its main processes, and then it will be possible to check what would happen on the final output if some change occurs.
- *Agent Based*: when the system to be simulated has a complex aggregate behaviour, not easy to describe just studying and modelling the single entities, agent based simulation is the only usable approach. In complex systems the sum of the parts is often not enough to describe the whole, and usually from the interaction of many simple entities a complex behaviour emerges. So, if we want to model an enterprise in which also the human factor is present, or we want to consider also the influence of the environment, it will be impossible to do that with a process based approach, thus leaving agent based simulation as the only feasible method. While in process simulation the stress is on the function of the single parts, which are deeply modelled as resembling the reality, in agent based simulation the most important side is interaction among entities, which creates the aggregate behaviour.
- *Hybrid (Agent-Process based)*: according to (Remondino, 2003), combining the two approaches, we can have a detailed model of the whole enterprise, with its production units, sales, purchases and account departments, logistics, warehouses and so on, modelled with a process based approach, and the environment, customers and sellers behaviour simulated using agent based technology. This approach, called Agent Based Process Simulation, allows to model machineries and the production units of an enterprise; the most difficult part to simulate, but probably also the most interesting for which regards the emergence of aggregate behaviour and self organization, is the human factor. For this reason, propositional logic is used to model the deterministic parts of the enterprise, while GA and CS constitute the mind of the agents involved.

When one of the agents involved (plausibly the director, the disposer or a manager), is a human player which must take decision to pursue a main objective (e.g. obtain profit, overcome competitors, etc.), the enterprise simulator built with an agent based, or even better a hybrid approach could be regarded as a complex and realistic real time strategy game. The environment and the various entities involved (customers, competitors and so on) are governed by the computer, according to AI rules, using GA and CS. The deterministic parts are simply modelled using logic based formulas, which can be easily translated into if-then conditions.

AGENT BASED PROCESS SIMULATION

Usually, since processes can be modelled as deterministic flows, my proposal is to use both Propositional and Modal Logic to describe their structure. In (McCartney 2001) we read that:

"The basis for most current systems of formal logic is Propositional Logic, also known as Propositional Calculus or PC. PC describes truth-based rules using the fundamental ideas of not and or, and derivations of the concepts of and, implication, and strong implication. A common extension to PC is predicate logic. Predicate logic includes variables as well as non-truth-based validity; or mapping variables into values other than the Boolean true or false. Another non-truth based logic is modal logic, which is based on PC and introduces the concepts of necessity and possibility. Modal logic is closely related to PC and predicate logic, but is able to describe states that would be indescribable in either of these languages"

In order to model a deterministic process, the Propositional Logic could be enough, since it allows to create truth tables of the single sub-processes. Modal Logic allows having a more versatile environment, allowing to determine if a proposition is true for sure, false for sure or sometimes true and sometimes false (i.e. it's possible). In my framework I will only suppose the use Propositional Logic, to model simple processes: this allows to describe a process, create a model of it and simplify the transition to programming code required to port it into a working simulation. A sub-block of a process produces output_1 if the logic formula is True, or

output_2 if it's False; one of the two outputs can be simply Void. In this way, a part of a whole process can be like exemplified in Figure 1.

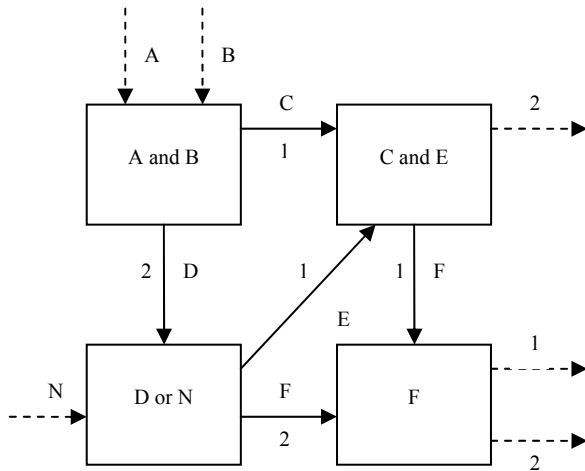


Figure 1: Propositional Logic Based Sub-block

Passing from this kind of representation to a programming language is a very easy step, since all the single boxes can be represented with if-then functions. In this way a very complex deterministic process can be modelled starting from very simple building blocks. Modal Logic can even add concepts of probability and necessity, so that a particular output going out from a basic building block can always occur or it's possible that it occurs. In this case a probability function can be given, representing the views on the possible modal worlds, to specify how often an output can be produced, given the initial rule.

This approach allows to model machineries, production units and all the parts based on a deterministic or stochastic behaviour in a real time strategy game. Agent Based Process Simulation is a way to model deterministic structures, made up of single processes, divided into Propositional Logic based building blocks, and having them interact with agents belonging to the sub-symbolic paradigms. This allows to simulate situations in which not only the deterministic structure, but also unpredictable situations could arise, caused by the environment or the human factor are important. In this way all the agents that require to have a human-like behavior could be modeled using AI derived approaches, thus creating a realistic behaviour in the game. At the same time, these agents will deal with deterministic (or stochastic) structures, thus learning how to interact with them.

ARTIFICIAL SOCIETIES AND INTERACTION

A very important feature for Real-time Strategy games is multiplayer capability, so that in the same environment many different human player can interact and create their own world. While this has been made relatively easy thanks to the internet, the artificial intelligence behind the computer driven players is still lagging behind, thus creating the impression that the actions performed by the artificial players

are somewhat predictable and fixed. In my opinion, a good single-player mode, that means a good AI, is really important for a game and people appreciate it.

While creating a working framework employing the proposed methodology is beyond the purpose of this paper, I'll try to give an example of what could be achieved with it. In the present games, the evolution of a computer driven society is often following precise patterns, set by the programmers themselves. Usually, The computer "knows" how to accomplish certain types of strategies that are common. In games in which the player has to build his own party, often the members act in a mechanical (and not intelligent way); the only commercial example of a learning AI is, in my opinion, the game Creatures (and its sequels), in which "heterogeneous" neural networks are used; the developer (T. Simpson, 2002) describes this technique saying that:

"Heterogeneous as in not harmonious. The neurones are divided up into lobes which serve different purposes, although the neurones in each lobe are the same. Things such as leakage rate, dendrite migration and so forth can be set for particular lobes without simply having a collection of the same old neurone as it would be in a "normal" net. This is the way mother nature does it, etc. As for what they actually do, well, they act like real living brains, only somewhat smaller than our own right now."

The problem about this game is that it was too much CPU intensive and, above all, it was not really a strategy game, but rather a virtual "pet" to grow up. Using the technology I'm proposing in this paper, it would be possible to create a storm of intelligent agents able to learn from one generation to the next one, by using the GA and CS paradigms, yet not charging the CPU that much, since all the parts that can be represented through processes will be simple deterministic structures. In this way, the player will have the opportunity to play into a self organizing world and at each round the game would be different, according to the actions taken by the human players themselves and the other artificial agents involved in the game.

CONCLUSIONS

The number of degrees of freedom in modern strategy games makes them a perfect field of application for agent based techniques, which can often exploit the complex aggregate behaviour even when applied to real situations, like social, anthropological and economical simulations. Besides modern games take place in dynamic complex worlds in which complex decisions, often based on partial knowledge must be made. This is exactly what happens, for example, in a real enterprise, a stock market or, in general a society.

According to (Fairclough et al., 2001), the actual trend in AI for games is to use schedule based finite state machines (FSMs) to determine the behaviour of the player's adversaries. Although this has been achieved to very good effect, FSMs are by their nature very rigid and, behave poorly when confronted by situations not dreamt of by the designer. That's why an agent based approach could deliver more realistic and less deterministic behaviour: agents could

self organize, producing intelligent aggregate behaviour, able to puzzle the human player and, at the same time, presenting different paths of evolution at every match. Besides, using hybrid approaches (agent based and process based) would allow the intelligent agents to self organize according to the deterministic structures, just giving simple rules; this would cause the behaviour to be consistent with the one that could be observed in the real world and would be quite independent from the choices of the programmers. In this way, the game could also deal with unforeseeable situations that were not implemented as possible ways of evolution. The main drawback of using such methods for simulating the AI in a game is that these techniques are quite hungry of CPU resources; though, in the last seven years, we have witnessed to the rise of dedicated graphics hardware (3D cards) which now, thanks to the integrated transform and lighting, pixel and vertex processing and so on, leave to the CPU just the management of the basic computing functions and of AI.

REFERENCES

- Holland, J.H. 1976, "Adaptation", In R. Rosen and F. M. Snell, editors "Progress in theoretical biology", New York: Plenum
- Fairclough C. et al., 2001 "Research Directions for AI in Computer Games", TCD-CS 2001
- Ostrom T. 1988, "Computer simulation: the third symbol system", Journal of Experimental Social Psychology, vol. 24, 1998, pp.381-392.
- Remondino M. 2003, "Emergence of Self organization and Search for Optimal Enterprise Structure: AI Evolutionary Methods Applied to ABPS", ESS03 proceedings, SCS Europ. Publish. House
- Simpson T. 2002, in "Games Making Interesting Use of Artificial Intelligence Techniques", the web

AUTHOR BIOGRAPHY



MARCO REMONDINO was born in Asti, Italy, and studied Economics at the University of Turin, where he obtained his Master Degree in March, 2001 with 110/110 cum Laude et Menzione and a Thesis in Economical Dynamics. In the same year, he started attending a PhD at the Computer Science Department at the

University of Turin, which will last till the end of 2004. His main research interests are Computer Simulation applied to Social Sciences, Enterprise Modeling, Agent Based Simulation and Multi Agent Systems. He has been part of the European team which defined a Unified Language for Enterprise Modeling (UEML). He is also participating to a University project for creating a cluster of computers, to be used for Social Simulation.