

AGENT BASED VIRTUAL TUTORSHIP AND E-LEARNING TECHNIQUES APPLIED TO A BUSINESS GAME BUILT ON SYSTEM DYNAMICS

Marco Remondino
Department of Computer Science
University of Turin
Cso Svizzera 185,
Italy
E-mail: remond@di.unito.it

KEYWORDS

Reinforcement Learning, Action Selection, E-Learning, software agents, Enterprise Simulation.

ABSTRACT

An advanced Business Game is presented in the paper, built on the methodology of System Dynamics. It can be used for cognitive learning and knowledge transmission in schools and Universities; it allows the learners to take decisions at each time step, after which it calculates the corresponding results, showing them according to the principles of double entry accounting.

An agent based framework is then discussed, which constitutes a form of virtual tutorship for the learners. The agents act as a decision support system for the decisions to be taken, and can explain some cause/effect relations. The agents themselves learn how the model work by practicing it, through some reinforcement learning techniques.

INTRODUCTION

Business Games (BG) can be considered a sort of role playing games, characterized by a managerial context. The players usually face some situations typical for enterprise management and must take various core decisions, mainly about marketing, logistics, production, research and development politics and so on. A very interesting feature of business games is that they can be employed as a teaching instrument; the students can learn some important concepts about enterprise management, by trying them on the field, instead of just studying them on books. This is regarded as "learning by doing" concept.

For this reason, in the following discussion, the players will be defined as "learners". The main didactic goals for BGs are to refine the decision capacities of the learners when facing situations of uncertainty, and above all their ability to take managerial decisions when there is a tradeoff between risk and profit. Besides, through a BG, some advanced managerial techniques can be taught, and so can be the interaction among the different enterprise functions.

The BG presented here implies that the learners must be organized in teams; each of them represents the manager of a single area (production, sales, research & development, manufacturing and so on) and they must coordinate themselves in order to achieve the best possible result. This

is ment to promote a collective knowledge, that is valuable in the real world.

The presented BG is built on the System Dynamics methodology (Forrester, 1961); this means that the mechanisms of the game are based on finite differences equations and curves defining the main parameters of the game itself.

An agent based framework is applied in the form of virtual tutoring system for the learners; the intelligent agents learn through a trial and error technique, based on Reinforcement Learning paradigms, by practicing the system. After this trial period, they form a model of how the simulation works (in particular of the cause/effect relations among the decisions and the observed results) and then they can be used as a decision support system for the human learners, during the game play.

A WEB BASED BUSINESS GAME

An existing simulation framework is described in this paragraph, used as a teaching support in some University courses at the Department of Computer Science, University of Turin, Italy; this will be the one to which the agent based paradigms described before will be applied, in order to obtain a virtual tutorship and a decision support system for the learners (the users). The simulation framework (<http://e-lab.di.unito.it/SimulazioneAziendale>), developed by prof. Gianpiero Bussolin and Dr. Marco Remondino at the University of Turin, Department of Computer Science, has been conceived as a teaching platform, used for transmitting such concepts as "double-entry accounting", and the way in which the decisions taken in a real enterprise affect the synthetic results, at the end of each period (month). The model, for now, is just in Italian, but a translation in English will soon be available.

In this model, the users have to take a number of core decisions at the beginning of every month; the system, based on Forrester's System Dynamics, generates a set of reports, typical for Management and Enterprise analysis. The users, by reading these reports, can track down the influence of the single decision – or even better the aggregate effects coming from two or more decisions – on the synthetic results, representing the monthly performance of the whole enterprise.

		(Range)
Lotto di acquisto materie prime (C)	150.00	(0 - 5000)
Lotto da mettere in produzione (C)	150.00	(0 - 5000)
Investimenti in impianti (euro)	32000.00	(0 - 500000)
Investimenti in ricerca e sviluppo (euro)	400.00	(0 - 5000)
Assunzioni mano d'opera diretta (p)	1.00	(0 - 250)
Licenziamenti mano d'opera diretta (p)	1.00	(0 - 250)
Prezzo unitario mercato nazionale (euro/pf)	566.00	(7 - 700)
Durata media del credito concesso ai clienti nazionali (gg)	30.00	(30 - 360)
Interesse annuo concesso ai clienti nazionali (%)	10.00	(1 - 20)
Promozione mercato nazionale (euro)	32.00	(0 - 5000)
Prezzo unitario mercato estero (euro/pf)	1000.00	(10 - 1000)
Durata media del credito concesso ai clienti estero (gg)	30.00	(30 - 360)
Interesse annuo concesso ai clienti estero (%)	10.00	(1 - 20)
Promozione mercato estero (euro)	32.00	(0 - 5000)
Tempo o dilazione richiesta per i pagamenti di fornitori (gg)	30.00	(30 - 240)
Prestiti richiesti alle banche (euro)	1.00	(0 - 500000)
Estinzione prestiti bancari (euro)	1.00	(0 - 500000)

Figure 1: a form for monthly decisions

Agents can have many roles in such a system; first of all, reactive agents can be used as a part of the system, in order to simulate customers or suppliers. These agents should be very simple, just reacting to some market curve. On the other hand, reactive agents could also be the production implants, with the possibility of being programmed by the users in some way, and then adapting themselves to the number of pieces to be produced, and so on. This kind of interactivity would make the model more realistic.

Cognitive agents may have different and more important roles in this kind of models used for e-learning. After a training period on the model itself, using the reinforcement learning methods discussed above, an agent can compute some strategies to be used to make profit in the simulation. That said, this agent could then be used both as a decision support system for human users – since it could foresee some results, based on its acquired experience – and as a virtual tutor, explaining the relations among certain variables (decisions) and the achieved results. This could help the learners to understand the cause/effect links.

The Inner Structure of the Model

The model is built using a structure based on the theory of System Dynamics.

The model itself is considered as an artefact, an interface between the *internal structure* (implemented in Java) and the *external environment*, i.e.: the physical one, in which the system itself is used by the learners, i.e.: the final users of the model.

There are six main subsystems, mutually connected, in the simulated enterprise: production, finance, emplants, research and development, marketing and sales. Some of these subsystems are divided into other subsystems, if needed (e.g.: national sales and sales to the rest of the World).

The model is a dynamic system and the temporal walkthrough in the system has been converted into a set of differential equations and laws that can generate the walkthrough itself. This description consists into a constant relation between the system status in a generic time T and the status after a brief time interval "delta T" (DT).

Two are the main variable types in the model: the stock type and the flow type (or rate). The latter are used to recalculate the formers after each DT.

Many of these flows are generated by the "actions" of the learners, i.e.: their decisions, in order to modify the states of the system. Not all the stats are modified by external actions, though. There exist some inner actions and regulations that can be considered as "internal implicit decisions" performed by the system, used to normalize the levels. The choice of the configuration and balance among the external decisions and implicit decisions identifies the nature and type of knowledge that has to be transferred to the learner in a direct or indirect way.

The external decisions are those that make it possible for the individual learners to know the object of their studies, since the object is directly "acted upon" by them. This kind of actions are simply referred to as "decisions", since they can be carried on by the learners. The other kind of decisions are those that make it possible to keep the system "alive" even when the learners (for a lack of knowledge) has not been able to lead the system.

The enterprise is part of a bigger external environment (or space) with which it continuously interacts. This environment is configured by some other sub-systems, like the banking system (able to supply the financial means for the developing of new technologies, new products and the enterprise itself), the market system (where the demand is generated in the form of orders for the enterprise), the technology system (that determines what kinds of technologies are available at a certain time step), the suppliers system and customers system (respectively simulating those sides) and the workforce system (determining the average wages, the work supply on the market and so on).

The equations in the model are in the form of:

$$SF_i = SS_i + (RI_i - RO_i) * DT \quad (1)$$

Where SF_i at the first member is the i-th Stock Variable at the end of a DT, while the SS_i on the right is the same variable at the beginning of the DT. RI_i and RO_i are respectively the Input Rate and Output Rate relative to the i-th stock variable.

The variation is then depicted as a difference among the Input Rate and Output Rate during the considered DT; this is summed to the previous stock value, to calculate the new one. It's important to notice that the algebraic difference among the two rates is to be weighted by the time in which that rates applied.

The units of measurement in the system directly derive from the above equation. The time is measured in *months* and the stocks are measured in *units*. The rates are then units/month and DT is again measured in months.

DT is a very brief time period; for simplicity, in the model is's set to be 1/100 of a month.

TWO AGENT BASED PARADIGMS

The term agent, deriving from the Latin *agens*, identifies someone (or something) who acts; the same word can also be used to define a mean through which some action is made or caused. In Nwana (1996) the studies on software agents have been divided into two main strands: the first one, starting in 1977, is based on the studies on Distributed Artificial Intelligence and gave origin to the cognitive agents, endowed with inner symbolic models. The second one, whose origins are to be found in the 90s, focuses on a wide range of agents, defined as reactive. They do not have any internal representation of their environment and the emphasis not on the way in which they decide what to do, but simply on when to act and what action to choose, basing on the stimuli from the environment. Reactive agents simply retrieve pre-set behaviors similar to reflexes without maintaining any internal state.

Both paradigms, though quite different, have some peculiar features that make them suitable for some given situations; the main problem with a purely cognitive agent, when dealing with real-time systems, is reaction time. For simple, well known situations, reasoning may not be required at all. In some real-time domains, minimizing the latency between changes in world state and reactions is important and so reactive agents can be successfully employed. On the other end, cognitive agents should be used when artificial learning or reasoning is concerned.

Reactive (sub-symbolic) Agents

This kind of agents may be regarded as any simple (not structured) software entities which interact among them and with the environment. A multi-agent context of this kind allows the emergency of complex behavior and self-organization, with no definition of a formal rule a priori. Apparent intelligent behavior is a product of the interaction among agents and environment, and of the interaction among many simple behaviors. 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 behavior 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.

In some situations, effective results can be obtained just by building simple, sub-symbolic agents, whose behavior 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 behavior. 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"

Cognitive (symbolic) Agents

These agents' behavior is goal-directed and reasons-based; i.e. is intentional action. The agent bases its goal-adoption, its preferences and decisions, and its actions on its Beliefs. In [8] we read that a software cognitive agent should feature the following properties:

- *autonomy*: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
- *social ability*: agents interact with other agents (and possibly humans) via some kind of agent-communication language;
- *reactivity*: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
- *pro-activeness*: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative." The Wooldridge and Jennings definition, in addition to spelling out autonomy, sensing and acting, allows for a broad, but finite, range of environments. They further add a communications requirement.

In general, cognitive agents have some sort of behavioral pattern, that can be described through modal logic, equations, evolutionary algorithms and so forth. In order to make the agents able to learn and improve themselves, some reinforcement algorithms can be embedded into their structure.

ACTION SELECTION AND REINFORCEMENT LEARNING FOR THE AGENTS

The action selection problem at time $t+1$, along with the goal selection, at a macro (aggregate) level, are central topics, when the agents must "learn" how the models works, by experimenting on it and being able to act as a decision support system for human users.

For action selection we do not simply mean the problem of choosing which action to take at a micro level (agent level), but also which one, among the possible goals, to select. The first level of detail is typical for reactive agents; in fact they don't have any goals, if not those imposed by the external environment. The cognitive agents feature both levels of detail. In this work, the action selection problem is crucial

for the formal definition of the involved agents, especially when they are employed as a decision support system, thus requiring some learning ability.

The goals could be combined to form higher level objectives or, on the contrary, be incompatible among them. In order to decide which actions to take, it's necessary to evaluate the utility for each of them; specific Reinforcement Learning (RL) algorithms are used for this purpose. These transform quantitative data (the payoff) in behavioural patterns for the agents.

An agent endowed with some RL algorithm, when in a particular state of the world (x), makes an action (a) and gets a payoff (r), calculated by a reward function based on the consequences of the action itself. Through a trial & error mechanism the agent learns what are the actions that maximise this numerical value. An effective RL algorithm is the one called Q-learning [Watkins, 1992]: an agent "lives" in a world modelled as a Markovian Decision Process (MDP), that's a set of states X , in which some actions from the set A can be done. For a state x , belonging to X , and an action a , belonging to A , there exist a probability function

$P_{xa}(y)$, determining the transaction probability towards a new state y . At the same time, for each couple of possible x and a , there exist another probability function $P_{xa}(r)$, determining the payoff – or reward – r , generated by the action. We obviously have that:

$$\sum_y P_{xa}(y) = 1 \quad (2)$$

And that:

$$\sum_r P_{xa}(r) = 1 \quad (3)$$

The Q-learning algorithm builds the so called Q-values, $Q(x,a)$, by considering not only the payoff of a singular action, but also the expected discounted sum of future payoffs obtained by taking action a from state x and following an optimal policy thereafter. So we have that in each time-step t :

$$R = r(t) + \lambda * r(t+1) + \lambda^2 * r(t+2) + \dots \quad (4)$$

where $\lambda \leq 1$ is the discount factor.

By defining policy (π) a set of action rules, given a state, we have that by following that policy, the total discounted reward at time t equals the previous formula with $E(r)$ instead of r , that's its expected value defined as:

$$E(r) = \sum_y r(x,a) * P_{xa}(y) \quad (5)$$

Besides acting as a decision support system for human learners experiencing with the model, artificial agents can be used to supervise the decision taken by learners, in order to interpret them in a cognitive way. For example, in the previously mentioned enterprise accounting model, some

users could immediately pursue an high profit, while others could be concerned first with the expansion of their enterprise on new markets. Others could choose to improve industrial plants, while others could want to differentiate production and invest on research & development and marketing. All these decisions are complex, since they are determined by the combination of many different variables. Sometimes the learners won't even realize that they are pursuing a strategy instead of another one, and they often won't foresee what the selected strategy could bring.

That's where the motivational model is employed; this is based on a function called wellbeing, where the intensity of the motivation to take a certain decision comes from the combination of two factors: internal drive and in a limited way, some external stimuli. We have that:

$$M_i = D_i + w_i \quad (6)$$

In order to give more relevance to the first term, an activation threshold can be used, such as:

$$\begin{aligned} \text{if } D_i \leq L_i &\Rightarrow M_i = 0 \\ \text{if } D_i > L_i &\Rightarrow M_i = D_i + w_i \end{aligned} \quad (7)$$

The wellbeing variable is calculated as the difference between the highest possible value and the sum of the motivational drives, weighted by a factor α , which represent in a cognitive way the personality of the human agent (the weight that the learner gives to the single decisions). We have that:

$$WB = WB_{\max} - \sum_i \alpha_i * D_i \quad (8)$$

The agents can also constitute some parts of the model itself; in the considered enterprise accounting model, some reactive agents can form the supply chain, or the warehouses, or even the competitors operating on the same market.

When dealing with reactive agents, the action selection problem is to be found at a macro (aggregate) level, i.e.: population level. If reactive agents are the competitors of human learners in the simulated world, they could have a fixed rule of behavior over time. Some evolutionary algorithms could be embedded in the agents, so that the best players on the market could merge, to form some other artificial players with an even better behavior.

In this way it's possible to start with a population of agents with a random behavior, facing the standard decisions in the model, and select – through the various "generations" – the best ones. So it's not the single agent that selects his behavior by updating its own policy (that remains the same, being the agent a reactive one), but the population that evolves over time, through the mechanism of reproduction and mutation.

This is an approach often used when the rules of the environment are given and the main task is to observe some emerging aggregate behavior arising from simple entities, i.e.: reactive agents. Since these agents does not feature a goal based – pro-active – behavior, the way they act tends to

be deeply dependent on the choices made by the designer. In order to design flexible systems, the aggregate behavior (at population level, i.e.: macro level) can be made self-adaptive through the implementation of an evolutionary algorithm (EA). In this case the agents will have a wired random behavior at the beginning, and evolve according to the environment in which they act, through a selection mechanism.

EA derive from observations of biological evolution. Genetic Algorithms (GA) [Holland, 1975] 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.

CONCLUSION AND FUTURE DIRECTIONS

A cognitive business game has been presented in this paper, used to form learners in the Universities and schools. The structure of the model is built on the theory of System Dynamics, by using the concept of stocks and rates, and considering the variations of the stocks as the difference among the input and output rates, multiplied by a delta T, a very short time interval.

The inner structure of the model has been briefly described in the paper, along the main sub-systems tied to form the whole.

The users of the system (called "learners") must take decisions at each time step, after which the system calculates the corresponding results, showing them according to the principles of double entry accounting.

Some agent based paradigms are then described as a future development for the system itself. The agent based framework will constitute a form of virtual tutorship for the learners. The agents act as a decision support system for the decisions to be taken, and can explain some cause/effect relations. The agents themselves learn how the model work by practicing it, through some reinforcement learning techniques, and are then able to assist the learners in the decision process.

ACKNOWLEDGMENTS

I would like to gratefully acknowledge the key support of *prof. Gianpiero Bussolin*, who originally designed and participated to the implementation of the Enterprise Simulator used as the starting point for many ideas described in this paper. I would also like to thank e-business LAB and in particular *prof. Marco Pironti* (University of Turin, Italy) for his precious support. I'd also like to acknowledge *the managerial board of Fontazione CRT*. Last but not least a big thanks to *Stefano Gabutti*, for his precious help and ongoing support.

REFERENCES

- Singh, H. (2003): *Building Effective Blended Learning Programs*, in Issue of Educational Technology, Volume 43, Number 6, Pages 51-54.
- Simon, H. A. (1996): *The Sciences of the Artificial*, (third ed.). Cambridge, MA, MIT Press
- Nwana, H.S. (1996): *Software Agents: an Overview*, in Knowledge Engineering Review, Vol. 11, N. 3, pp.1-40, Cambridge University Press.
- Sutton, R. S. (1998): *Reinforcement Learning: an Introduction*, MIT press
- Watkins, C. J. C. H., Dayan P. (1992): *Q-Learning*, Springer ed.
- Gadano S. C. (2003): *Learning behavior-selection by emotions and cognition in a multi-goal robot task*. The Journal of Machine Learning Research. Volume 4 Pages: 385 – 412. MIT Press Cambridge, MA, USA.
- Holland J.H. (1975): *Adaptation in natural and artificial system*, Ann Arbor, The University of Michigan Press
- Woolridge, M., and Jennings, N. (1995): *Intelligent agents: Theory and practice*. Knowledge Engineering Review 10(2). pp. 115-152

BIOGRAPHY

MARCO REMONDINO got his **Master Degree in Economics** at the beginning of 2001, with *110/110 cum Laude et Mentione*. Some months later, he started a **PhD in Computer Science**, during which he delved into theoretical studies about the structure of different software agents, namely the reactive and deliberative (BDI) ones. He completed his PhD in January 2005.

After that, he was awarded a **two-year research scholarship** from ISI Foundation, for the *Lagrange Project on Complex Systems*, during which he applied agent based paradigms to design several models in different scientific fields, namely Game Theory, Biology, Economics and Enterprise Simulation.

At present, he holds a **post-DOC research fellowship** from University of Turin, Department of Computer Science.

His main research interests are Agent Based Modelling, different paradigms for agents and their integration, computer simulation, Social Systems, emergent properties for Complex Systems, Social Networks, Action Selection, agent learning through Neural Networks, Genetic Algorithms, Classifier Systems, paradigms of Reinforcement Learning, Data Mining, validation of models and E-Learning.