

AISB '03
Cognition
in Machines
& Animals

**Proceedings of the
AISB '03 Third Symposium on
Adaptive Agents and Multi-Agent Systems
(AAMAS)**

7th – 11th April 2003

The University of Wales, Aberystwyth

EPSRC



Prifysgol Cymru
Aberystwyth
The University of Wales



The AISB'03 Convention
COGNITION in MACHINES and ANIMALS

7th - 11th April 2003
University of Wales, Aberystwyth

www.aishb.org.uk

**Proceedings of the Third Symposium
on Adaptive Agents and Multi-Agent
Systems**

Published by

**The Society for the Study of Artificial
Intelligence and the Simulation of
Behaviour**

www.aisb.org.uk

ISBN 1 902956 31 5

Printed at the University of Wales, Aberystwyth, SY23 3DB, Wales

The proceedings of the five symposia in the AISB'03 Convention are available from SSAISB. They are entitled as follows:

The Second International Symposium on Imitation in Animals and Artifacts
ISBN 1 902956 30 7

The Third Symposium on Adaptive Agents and Multi-Agent Systems
ISBN 1 902956 31 5

Scientific Methods for the Analysis of Agent-Environment Interaction
ISBN 1 902956 32 3

Biologically-Inspired Machine Vision, Theory and Application
ISBN 1 902956 33 1

Artificial Intelligence and Creativity in Arts and Science
ISBN 1 902956 34 X

due to all the symposia Chairs for all their time and hard work: Kerstin Dautenhahn, Chrystopher Nehaniv, Dimitar Kazakov, Daniel Kudenko, Eduardo Alonso, Ulrich Nehmzow, Gregor Schöner, Horst Holstein, Fred Labrosse, Pablo Gervas and Simon Colton. Special thanks also to our Technical Programme Assistant, Joanne Walker, and, for typesetting expertise, to Fred Labrosse. Finally, thanks to all the authors for their contributions, which form the essential core of the event. I am looking forward to the programme and I wish everyone a stimulating and productive meeting.

Mark Lee (Convention Host and Programme Chair)

Contents

AISB Keynote Paper	1
Artificial Intelligence and the Natural Selection; Decision Trees and the Network Forest. <i>S. Solomon, E. Shir and S. Kirkpatrick</i>	3
AAMAS Invited Paper	13
When IT Gets a Mind of IT-Self Emergence of Complexity in Artificial Systems. <i>S. Solomon and E. Shir</i>	15
Contributed Long Papers	21
An Agent Managed Ant-Based Network Control System. <i>D. Legge and P. Baxendale</i>	23
Developing Committee of Explorers for Reinforcement Learning Agents. <i>F. T. Sunmola and J. L. Wyatt</i>	31
SMART (Learning Model Acquisition with Reinforcement) Learning Agents: A Preliminary Report. <i>C. Child and K. Stathis</i>	38
Learning Multi-agent Search Strategies. <i>M. J. A. Strens</i>	46
Learning to Coordinate Using Commitment Sequences in Cooperative Multi-Agent Systems. <i>S. Kapetanakis, D. Kudenko and M. Strens</i>	54
Advice-Exchange between Evolutionary Algorithms and Reinforcement Learning Agents: Experiments in the Pursuit Domain. <i>L. Nunes and E. Oliveira</i>	62
Evolving Strategies for Agents in the Integrated Prisoner's Dilemma in Noisy Environments. <i>C. O'Riordan</i>	74
Experiments in Subsymbolic Action Planning with Mobile Robots. <i>J. Pisokas and U. Nehmzow</i>	80

Contributed Short Papers	89
Adaptive Behaviour Based on Evolving Thresholds with Feedback. <i>T. De Wolf and T. Holvoet</i>	91
Coordinated Exploration in Stochastic Common Interest Games. <i>K. Verbeeck, A. Nowe and K. Tuyls</i>	97
Emergence of Organisations, Emergence of Functions. <i>D. Capera, J-P. George, M-P. Gleizes and P. Glize</i>	103
Real-time simulation for Flood Forecast: an Adaptive Multi-Agent System STAFF. <i>J-P.George, M-P. Gleizes, P. Glize and C. Regis</i>	109
Shortening of Evolutionary Agent Learning Using Sequential Pattern Methods. <i>J. J. Korczak and P. Lipinski</i>	115
Autonomous Itinerary Planning for Mobile Agents. <i>C. Erfurth and W. Rossak</i>	120
Adaptive Firms and Emergent Organizational Forms. <i>Z. Guessoum, L. Rejeb and R. Durand</i>	126

Artificial Intelligence and the Natural Sciences Decision Trees and the Network Forest

by Geoffrey I. Hinton & Elman S. Elman

Department of Psychology, University of Toronto

100 St. George Street, Toronto, Ontario, Canada M5S 1A5

AISB Keynote Paper

We are pleased to present this keynote paper at the AISB conference. The paper discusses the relationship between Artificial Intelligence and the Natural Sciences, focusing on the use of Decision Trees and the Network Forest. The paper is divided into two main parts: the first part discusses the use of Decision Trees in the Natural Sciences, and the second part discusses the use of the Network Forest in the Natural Sciences. The paper concludes with a discussion of the future of Artificial Intelligence and the Natural Sciences.

1. The History of the AISB

The AISB was founded in 1963 as the British Association for the Advancement of Artificial Intelligence. It was the first of a series of international associations that have since been founded, including the American Association for Artificial Intelligence (AAAI), the Canadian Association for Artificial Intelligence (CAAI), and the European Association for Artificial Intelligence (EAAI). The AISB has a long and distinguished history, and has played a major role in the development of Artificial Intelligence in the United Kingdom and elsewhere.

The AISB has a number of committees and subcommittees, and has published a number of journals and books. It also organizes a number of conferences and workshops, and has a number of other activities. The AISB is a non-profit organization, and its activities are funded by a number of sources, including the British government, the European Commission, and private industry.

The AISB is a member of the International Federation for Artificial Intelligence (IFAI), and is also a member of the European Association for Artificial Intelligence (EAAI). The AISB is a member of the British Psychological Society (BPS), and is also a member of the British Society for Artificial Intelligence (BSAI). The AISB is a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI), and is also a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI).

The AISB is a member of the British Psychological Society (BPS), and is also a member of the British Society for Artificial Intelligence (BSAI). The AISB is a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI), and is also a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI).

The AISB is a member of the British Psychological Society (BPS), and is also a member of the British Society for Artificial Intelligence (BSAI). The AISB is a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI), and is also a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI).

The AISB is a member of the British Psychological Society (BPS), and is also a member of the British Society for Artificial Intelligence (BSAI). The AISB is a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI), and is also a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI).

The AISB is a member of the British Psychological Society (BPS), and is also a member of the British Society for Artificial Intelligence (BSAI). The AISB is a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI), and is also a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI).

The AISB is a member of the British Psychological Society (BPS), and is also a member of the British Society for Artificial Intelligence (BSAI). The AISB is a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI), and is also a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI).

The AISB is a member of the British Psychological Society (BPS), and is also a member of the British Society for Artificial Intelligence (BSAI). The AISB is a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI), and is also a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI).

The AISB is a member of the British Psychological Society (BPS), and is also a member of the British Society for Artificial Intelligence (BSAI). The AISB is a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI), and is also a member of the British Society for the Philosophy of Artificial Intelligence (BSPAI).

Artificial Intelligence and the Natural Selection; Decision Trees and the Network Forest

S. Solomon, E. Shir and S. Kirkpatrick

Hebrew University of Jerusalem

sorin@cc.huji.ac.il, eran@cogniview.com, kirk@cs.huji.ac.il

Abstract

We present the Internet (and the overlying Web) as a complex system with several layers of emergence. We suggest the various ways in which one can deduce the increasing levels of complexity and (self-)organization one from the other. An effort is made to give enough technical detail to make our approach plausible. Yet the emphasis is on the challenges that the web poses in principle to our understanding. The connection with some previous studies of similar (but simpler) emergent complex systems is made.

1. The Identity Crisis of the Single Computer

When Turing defined the conceptual framework for information processing machines he limited their scope to the context of rational, deterministic, closed systems. For a long while the computer engineers and computer scientists were proud of the absolute conceptual and practical control that they had on their algorithms and machines. Even when randomness appeared it was under well defined rules that still allowed rigorous proofs and rigorous control. Most of the project managers spent most of the time to eliminate any possibility that the system would do something else than what explicitly prescribed by the user specifications.

As long as the main requests of the society from the computing machines were just to compute, this perfect order, predictability, reproducibility and their closed self-contained nature constituted the very backbone of the product integrity. However, at some stage, computers became so useful, cheap, powerful and versatile that the customers required their massive integration in real life situations. Upon throwing the computers into the real life one opened a Pandora box full of surprises:

- computers proved to be inexplicably poor in tasks that the AI pioneers have (recurrently!) predicted to be quite at hand:

Example : Turing stated more than 50 years ago the famous Turing test: a woman communicates in writing with a computer and respectively with another woman. The objective of the computer is to masquerade as a woman. Turing's hope was (Turing 1950):

“... in about fifty years' time ... interrogator will not have more than 70 percent chance of making the right identification...”

Obviously the prediction is far from being fulfilled at the present time.

- humans either, turned out to be not-so-good at being ... humans.

Example: Men fail in proportion of 80% the above Turing test. Therefore a computer which would succeed 70% as predicted by Turing, would be in fact super-human (or at least super-man) (Adam and Solomon 2003).

- computers could do better than people, task that humans have claimed to themselves as intrinsically and exclusively human.

Example: By mimicking on computer the common structural elements of past successful ideas one was able to construct idea generating algorithms. The computer generated ideas were ranked (by humans) higher than ideas generated by (other) humans (Goldenberg et al 1999).

- computers turned out to do the same tasks that human do in very different ways.

Example: By analyzing mathematically psychophysical observations one was able to infer the approximate, quick-fix ad-hoc algorithms that the visual system uses to reconstruct from 2 D image sequences the 3 D shapes. They are totally different from the ideal mathematically rigorous 3D reconstruction algorithms. In fact using this knowledge one was able to predict specific visual illusions (that were dramatically confirmed by subsequent experiments) (Rubin et al 1995).

2. The Social life of computers

Beyond the surprises in the behavior of the individual computers, much more far-reaching surprises lurked from another corner: already 30 years ago, Physics Nobel Prize laureate Phil Anderson (1972) wrote a short note called "More is Different". The main point was to call attention to the limitations of thinking about a collection of many similar objects in terms of some "representative individual" endowed with the sum, or average of their individual properties. This "mean field" / continuum / linear way of thinking seems to fail in certain crucial instances. In fact there are exactly those instances that emerge as the conceptual gaps between various disciplines. Indeed, the great conceptual jumps separating the various sciences and the accompanying mysteries connected to the nature of life, intelligence, culture arise exactly when "More Is Different". It is there that life emerges from chemistry, chemistry from physics, conscience from life, social conscience/ organization from individual conscience etc.

The physical, biological and social domains are full of higher functions that emerge spontaneously from the interactions between simpler elements. The planetary computers infrastructure has acquired a large number of elements by now. It is therefore not unexpected that this bunch of man-made artifacts would develop a mind of itself. In fact we do perceive now days that the very basic rules under which computers were supposed to function are being affected. *Are we facing a "More is Different" boundary?* It might be too early to decide, but let us make a list of the ways the rules of the game have changed:

-Instead of being closed systems, the computers came to be exposed to a rapidly changing environment: not just some environment dynamics, but a continuous change in the very rules that govern the environment behavior.

-Some of the environment consists of other computers, so the potential infinite endogenous loop of adaptation, self-reaction and competition is sparked.

-Instead of a deterministic dynamics, the system is submitted to noise sources (Shnerb et al. 2001) that do not even fulfill the right conditions to allow rigorous proofs (e.g. in many of the cases, far from being "normal" the noise probability distribution doesn't even have a finite standard deviation).

-Instead of worrying about the integrity of one project at a time, one has to face the problems related with the evolution in parallel of many interacting but independently owned projects, carried out by different teams.

-Instead of being clustered in a well-defined, protected, fixed location (at fixed temperature :-), the computers started to be placed in various un-coordinated locations and lately, some of them mounted on moving platforms.

-Instead of a single boss that decided which projects go ahead, with what objectives and with what resources, the present environment is driven by the collective behavior of masses of users whose individual motivations, interests and resources are heterogeneous and unknown. Unfortunately psychology was never one of the strengths of the computer people...

In conclusion **the main change** to which we are called to respond is the passage of computer engineering from deterministic, up-to-down designed and controlled systems to stochastic, self-organizing systems resulting from the independent (but interacting) actions of a very-large number of agents.

Main conceptual shifts related to this:

- The fast evolution of rules /players behavior make classical AI and game theory tools inapplicable => Relying on adaptive co-evolution becomes preferable (even though in general there are no absolute mathematically rigorous success guarantees).

- Instead of well defined asymptotic states obtainable by logical trees etc., the pattern selection depends on spatio-temporal self-organization, frozen accidents, and a Nash equilibrium might not exist or be unreachable in finite time.

- inhomogeneity and lack of self-averaging (Solomon 2001) make usual statistics in terms of averages and standard deviations unpractical: resulting (Pareto-Zipf power) distributions (Levy and Solomon 1996) might not have strictly speaking an average (and in any case not close to the median) and neither finite standard deviation.

- systems display phase transitions: dramatic global system changes triggered by small parameter changes.

The situation seems really hopeless. But a ray of light appears:

In their encounters with the real world, CS people encountered occasionally quite accomplished (and sometimes attractive) entities which were not designed (let alone rationally) by anybody.

So may be there is another way to create things beside constructing an exhaustive logical tree that prescribes in detail the system action in each given situation. For lack of another name one could call it "natural evolution".

There is long list of large prices to pay by giving up the "old way": the system might not fulfill well defined user specifications, might not be fault free and might not be predictable, etc etc

3. Terrestrial Intelligence Search

Almost everybody heard of SETI@Home. SETI@Home is a project aimed to use the spare CPU power of computers all over the world in order to analyze the enormous amounts of data constantly aggregated from radio telescopes and search for evidences of extraterrestrial messages. This project, initiated after the funding for dedicated computer power to the project has dried out, has become a huge success and is now considered to be one of the strongest super-computers ever created.

SETI@Home essence is to get a hold of as much as possible CPU power, and their architecture is a client/server one (the client gets a packet of data to analyze from the server and send the results back to the server. However there is no interaction between peers, no collaborative effort. The SETI@Home phenomenon shows that the psychology and social behavior of the Internet citizens it is sometimes a more stable and on-going force than the usual internet evolution rules.

GOOGLE and AKAMAI are essential for access and use of global information networks in today network. GOOGLE attempts to find anything that is currently on the Web. AKAMAI pushes information as close as is justified to its consumers to ensure good performance.

Both GOOGLE and AKAMAI are highly centralized, or depend upon specially selected server complexes which are themselves centrally planned, deployed, and managed. The ultimate goal must be the emergence of fully distributed, invisibly- or self-managed facilities which take full advantage of a storage, computation, and communication hierarchy that spans the range from

central, physically secure services enjoying ultra high bandwidth to the incredible capacity that will exist out at the edges of the network, with intermittent, lower bandwidths and a division of labor between personal and community activities.

GNUTELLA is a first attempt at opening up information sharing and access in a purely peer to peer structure. Overlay networks are the mechanism by which such new functions are brought into use in the previously existing networks. GNUTELLA exemplifies the new situation in which the CS problem becomes intimately intertwined with the social and psychology problems: the various social groups connected to GNUTELLA or clustered around various (We)blogs could not exist without the internet medium, but on the other hand have an independent dynamics and influence back on the present usage and future development of the web. The Web develops collective clusters that have their own emergent identity, interests, adaptive behavior, tastes: in short- a mind of themselves.

Could it be that GNUTELLA and its grand-children may pass the SETI intelligence tests?

Could it be that upon appropriate study and using enough empathy, we could recognize the Web as an entity on its own that fully deserves the label terrestrial intelligence?

If you jump to say no, recall that less then 100 years ago, entire nations and races denied human status to their peers belonging to other races / nations.

In order to understand the planetary computer infrastructure and the way it may develop in the future we have to first realize that we are still at the stage at which

we do not know even the basic facts about the internet !

Sure, we know - or can find out readily - everything about every single element of the system but does the knowledge of all the electric currents in your TV teaches you anything about the currently running TV program?

4. The Problems with Studying the Internet

This basic knowledge on the Internet can be achieved only by enhancing very dramatically the scope and

precision the measurements of the network topology and traffic. Moreover, one needs good measurements of the short and long time-scale changes of these network characteristics.

Only on the basis of this experimental knowledge one will be able to

- develop and validate relevant models,
- test ideas on current networks and
- extrapolate potential architectures, protocols and control strategies for the future.
- search for cognitive emergent functions.

There are **crucial difficulties** related to Internet studies:

- The lack of knowledge of the details of the network (and the lack of knowledge of which details are relevant vs. the negligible ones).
- Centrally measuring the Internet's topology is an impossible task because of "hidden connections" (basically, a node A can acquire by measurement knowledge on links leading from A to some other nodes B, C... , but, in general, not on links between B and C).
- The impossibility to represent the entire network in one machine:

Creating in a lab a relevant, truthful simulation of either today's or tomorrow's Internet is an impossible task due mainly to the following problems :

A. Consistency - It is impossible to create artificially a replica of the Internet, since the growth of the Internet is an extremely collaborative, distributive and complex process, which in itself changes rapidly.

Currently there are two approaches which are used in creating **models of the Internet**.

- **The first** and most popular approach is to "grow" an artificial Internet using a heuristic algorithm which aims to capture certain global characteristics of the Internet such as a scale free structure of the graph topology .Such an approach is less than sufficient since we still lack full understanding of the topological nature of the Internet, the current understanding is in fact misleading in some important aspects .
- **Another** potential approach is to use the results of a central topology measurement project and create a model of the real Internet. Again, what is now widely accepted is that extracting the Internet structure using a central or semi-central experiment architecture has significant drawbacks, and the model created deviates substantially from reality. In any case, while a rough, still ,boolean picture of the topology may be taken, it is

obvious that the message dynamics cannot be mimicked centrally.

B. Scale - Creating a large enough model becomes a resource issue when building it on a machine, or even a cluster of machines. It becomes impossible and uneconomical from a certain scale. Thus most researchers compromise and test their ideas on a less than adequate simulation model. This approach is clearly in the danger of missing the essential "More is Different" effect.

C. Adaptability - As the Internet evolves, so should its model. However, understanding explicitly each meaningful change and implementing it into the model may become a Sisyphean task .

D. Unknown parameters : connections bandwidths , switches speed stacks capacity and software parameters of each relay.

5. The solution: The Introspective Internet

THE IDEA

The idea is having the Net measure itself, simulate itself and foretell its own future.

Instead of using a single machine to study the Internet one will use (potentially) all the machines on the Internet

The study of the Internet gives us a first opportunity to use the object of study itself, as the simulation platform

It is told on one emperor of China, that he had asked his consuls to create for him a detailed map of his vast kingdom.

After numerous attempts that did not satisfy his desire for details and accuracy, eventually the consuls took the emperor to the most accurate model of all, real China which lies behind the walls

This story is often told to stress the fact that every model is a simplification of the object it aims to model. However, our claim is that by using the Internet itself as its model, we will be able to bridge this gap between the real object and its model.

THE METHOD

We propose to use the Internet as the main infrastructure that will simulate itself and its future generations:

recruit millions of nodes spread over the entire Net that:

- Perform local measurements of their Internet neighborhood
- Participate as message routing/processing elements in collective Internet experiments

One can hope to harness into the task hundreds of thousands and even millions of users and nodes in all hierarchy levels, that will contribute some of their CPU power, Internet bandwidth, and most importantly, their topological/ geographical whereabouts on the net, to the effort of monitoring and studying current days Internet on the one hand, and simulate and study its future on the other.

Our goal is to create the largest computer simulator ever, one that will be many scales larger than the second largest one. But this simulator will not merely be a very large one. It will also be extra-ordinary since each of its elements will be unique:

each of the elements will bring not only generic resources such as cycles ,memory or bandwidth, it will be also a true representative of its local net neighborhood. For that matter, it will create a strong connection between the simulator and its object of simulation, i.e. reality .

The Tool: DIMES

(Distributed Internet MEasurement and Simulation)

We propose the creation a distributed platform that will enable:

- Global scale measurement of Internet graph structure, packet traffic statistics, demography
- Simulation of Internet behavior under different conditions.
- Simulation of the Internet future.

We propose creating a research infrastructure to which programmable experiment nodes could be added spontaneously. Using this layer, large scale statistics aggregation efforts will be conducted. We will conduct simulations that will study both:

- the reaction of the current Internet to various irregular situations and

- the usability of concrete new ideas, algorithms and even net physical growth schemes.

Building the research infrastructure itself is a worthwhile and important project, since it will take the already known and established success of *distributed computing* projects such as SETI@Home to a new level, namely, *collaborative computing*. Whereas *distributed computing* distributes small pieces of work among many machines which interact only with the central "Work Manager", *collaborative computing* also enables interaction algorithms between the machines. On this collaborative computing platform, collaborative algorithms can be studied and complex phenomena such as the emergence of global behavior from local interaction can be exhibited in the most realistic manner .

DIMES Set Up

The backbone of DIMES@Home will be the distributed network of copies of the **basic** (recursive) **software agent**. These **software agents** will form a virtual Internet layer that will sit on top of the physical Internet .

Similarly to the real nodes of the Internet graph, i.e. routers, the **basic agents** will have virtual routing capabilities, which will enable them to forward messages on the virtual layer.

Another important feature of the **basic agent** will be that it will have the ability to simulate inside it other instances of **itself** . This will enable us to integrate seamlessly between large agents that simulate and represent large internet neighborhoods and reside on strong machines and thin agents which simulate a single node and reside on a home pc on the bottom of the process priority list .

The agent will be spread in the world using a multi layer marketing strategy:

- We will address home users by providing them with both concrete benefits (such as visualization of their Internet neighborhood, troubleshooting capabilities of Internet problems) and psychological ones (such as competition, community and a larger than life goal) .

- We will also address ISPs and Internet infrastructure providers, as well technology vendors, by providing them with concrete advantages that only such a research project can provide .

Let us now get into greater details of what we aim to achieve.

6. DIMES@Home: the Distributed Internet MEasurement and Simulation project

DIMES@Home as a research support infrastructure will allow simulation and testing of ideas on:

- Internet graph growth and traffic patterning by technology and society (find optimal physical resource allocation policies: Where is the best place on my network to put another router?).
- The robustness and flexibility of the Internet under anomalous conditions, such as congestion waves, K-core connectivity loss and catastrophes and under alternative hardware / routing policies.
- Predict the impact of new emergent phenomena on the entire Internet. For example, if DIMES existed few years ago, we would be able to study flooding effects of first generation peer-to-peer protocols much before they have become a substantial issue which consumes large portions of the Internet bandwidth .
- Provide the Large Scale Networks research community with a universal ,ubiquitous experimental platform

These goals are related to a long list of concrete research questions. These research questions are divided to two main groups .

The **first group** are the research questions concerned with the optimal build up of a large scale collaborative computing and networking system. These questions will push the edge of current knowledge and technological skills in issues such as GRID computing and peer-to-peer protocols, as it will be the first attempt to create a complex system of these proportions. Issues such as autonomous computing and emergent hierarchies will have to be addressed and studied in new levels.

The **second group** of questions are research questions which focus on the attributes and abilities of the Internet of today, as well as on the potential directions it will take tomorrow. Issues such as dynamic routing, market inspired mechanism ,optimization in stochastic environments and multicast will be the focus of much study .

7. The Internet as a complex emergent system

If we would be in the situation of studying the human intelligence, one way to organize our efforts would be along an axis starting at the most physical / "hardware" aspects and ending at the most "soft", "humanistic" ones. Thus this study would fall within 3 different disciplines:

1. biophysics: studying neurons and their interactions network.
2. psychophysics: studying perception and information processing.
3. psychology: studying cognitive and motivational processes.

The table below attempts to display the planned study of the internet by DIMES@Home along similar principles. The **three rows** of the table (labeled 1., 2., 3.) capture the three levels at which one can characterize the Internet:

1. physical,
2. information flow and
3. emergent / cognitive.

In other words,

1. **The lowest row** studies the topology of the internet connections and packet dynamics and the consequences thereof for robustness and traffic.
2. **The next layer (row)** studies the links created by overlay networking and the information flow.
3. **The highest layer** concerns the emergent institutions connecting people / sites in information, business and social communities.

The horizontal axis represents time: **present (column A) , future (column C) and in-between (column B)**. Each of these three columns relies on a different aspect supported by the DIMES@Home platform (**measurements, simulation and theory** respectively - see the bottom of the table):

- A. The study of the Internet **present** relies mainly on observations / **measurements**.
- C. The study of the Internet **future** relies on / **simulation** / design.
- B. The gap between present and future is bridged by new ideas, **theoretical concepts** and new technology.

TIME ->	A.	B.	C.
Conceptual Layers V	Internet Now Observatory	New Ideas; Analytical tools	Future Internet Mock-up
Cognitive / Social Layer Web Self-Organization	Content based relationships Space- Time dynamics of Communities	Emergent Collective Entities Emergence of Complex Institutions from local interactions Market mechanisms	Collective Objects, Their Personality, Interests, Strategies Predict emergence / success / failure of social / technological Web waves
Information Layer Overlay networking	GRID and peer-to-peer, Information transmission measurements Web dynamics	Information flow distributed control Local protocols Self-healing	Study / Simulate Statistical stability and guarantees? Can trust be distributed?
Hardware Infrastructure Layer communication lines packets and routers	Find hidden net elements and connections Packets Traffic Spatio-Temporal Fluctuations	Autocatalytic Agents; Self-Regulated growth Power Laws, Small Worlds K-core Percolation	Hardware Evolution: Mobile nodes, ad-hoc connectivity etc. Simulate reliability and traffic changes upon hardware /protocol modifications
	Mesurements	Theory	Simulation
	DIMES@Home Platform		

Table 1 Levels of the study of the present and future of the Internet

Let us describe in more detail the table contents:

Row 1. Hardware Infrastructure layer (closest to the Dimes@Home foundation) represents the most basic study of the **internet physical infrastructure and packets dynamics**.

1.A Present Internet (physical) infrastructure topology and dynamics:

- discover and measure inter-country and local links not previously observable, along with information on the rate of change of these characteristics

- measure with high precision the packets dynamics and its spatio-temporal fluctuations. Characterize spatial as well as temporal extent of bottlenecks and delays.

1.B. Theoretical ideas for describing Internet infrastructure evolution:

- relation between topology and dynamics (Ben-Av and Solomon 1990); scaling and critical slowing down (Solomon 1995).

- biologically inspired network growth (Hershberg et al. 2001) mechanisms: autocatalytic agents (Louzoun et al. 2001), self regulation (Levy et al 1996) , Lotka-Volterra systems (Solomon and Richmond 2001).

- small world, power laws (Blank and Solomon 2000) , other global topological features (Weisbuch and Solomon 2002).

- link and K-core percolation (Shalit et al 2003) ; multigrid (Stenhill et al 1994) and cluster (Perski et al 1996).

1.C The Future of the Internet infrastructure geometry / functionality:

- guess the future voluntary / emergent trends: mobile nodes, ad-hoc nets, active nets, novel routing patterns and distributed resource allocation.

- stochastic networks dynamics (Adi et al 1994).

- finding solutions for systematic, recurrent failure of elements (without replacing them) (Kirkpatrick).

- simulate the effect of changes on net reliability, prompt response etc.

- Clusters (Ben-Av et al. 1992) and Multigrid / Multiscale dynamics (Evertz et al 1991).

Row 2. Information Layer is expected to teach us where in the web the information is / should be generated, localized, processed, and transmitted.

2.A The present Web (overlay networks) dynamics:

- emergence and evolution of peer-to-peer networks including their cooperative mechanisms for information storage, retrieval, processing and transmission.

- a question analog to the brain anatomy-physiology problem (Rubin et al. 1995b, Adi-Japha et al. 1998) is finding the relation between web geometry, micro-dynamics on one side vs. informational relevance on the other: what are the most relevant nodes/ links? the ones with the most connectivity, traffic?

- what are the evolutionary pressures put on the Web by that this geometry - functionality connection? Frustrated systems , spin glasses and cluster algorithms (Persky and Solomon 1996).

2.B The theoretical studies of the Web:

- One should be able to produce distributed / local protocols for information transmission (Goldenberg et

al 2000) with a certain level of adaptive / self- healing features and reasonable security (Weisbuch and Solomon 2000).

- Characterize the stability of overlay network local protocols in terms of their typical case in the statistical mechanics style (Solomon and Levy 2003) is quite different then worst-case analysis that dominated the mostly Game Theory-based treatments until now (when sitting in the corner of a room, there exists the danger that all the air molecules in the room will spontaneously cluster in an other corner. There are no documented cases of this worst case actually taking place).

- It is important to establish to which degree departures from the (stochastically) stable regime are restored (Persky et al 1995).

2.C The future of the Web:

- The simulation approach (Solomon 1995) will permit testing the success of distributed control algorithms, which manage typical behavior, and distributed trust, privacy, and other socially valuable "qualities" of service, whose underlying graph theories are far more complex. The success of heuristic strategies for forming peer-to-peer networks and services which accomplish both economical communications and local data provisioning can only be tested by this sort of simulation.

- The practical application of such analyses to security issues will result in optimal "immunization"/"security" policies consistent with the "openness"/"information freedom" that peer-to-peer networks need.

Row 3. Social / Cognitive Layer studies higher emergent web faculties:

3.A. Present collective features:

- On top of the peer-to-peer content based connections (**Row 2**), collective self-organizing communities appear quite spontaneously. As in the case of other spontaneous condensation phenomena, the exact place and time at which the percolation transition (Weisbuch et al 2000) takes place may be unpredictable. Yet one can try to measure first experimentally the space-time evolution of these communities and then get increasing understanding and control of their dynamics (**3.B.**).

- One would hope to characterize in terms of these collective emergent phenomena (Solomon et al 2000) the present connections within and between communities (possibly using the links within Web content clusters provided by today's search engines).

- Dynamics of Networks of Concepts and the emergence of meaning (Stolov et al. 2001).

3.B. Theoretical Ideas on Web emergent features:

- One should explore the potential of market mechanisms (Levy et al. 1994, 1996, 2000) to design and manage the web-communities "social" / "resources" needs. This is a very open field at present. Both proofs (3.B.) of desirable characteristics and simulations (3.C.) of market behaviors (Solomon 1999) would be useful.

- The application of simple agent-oriented models to complex collective phenomena has already certain predictive successes (Louzoun et al 2000, Muchnik 2003).

- The self-organization potential of the market mechanisms may explain the emergence and many of the properties of the net institutions (Louzon and Solomon 2001).

3.C. Predicting the future social trends and cognitive features of the Web

- One should endeavor to design network / web relationships not to **be optimal** but (Mar Or et al. 2003) to **evolve** according to protocols / by-laws that self-organize them in **optimal adaptive communities** (don't endow the system with a fish from the beginning: teach it how to fish).

- One should be able to predict the conditions in which certain global "phase transitions" may take place and the ways to trigger, prevent, accelerate, stop or modify them (Weisbuch and Solomon 2000).

Such a program will require the integration between several distinct research communities. In fact such a group, including scientists from the communications, network measurement, distributed computation, and physics of complex systems communities is being established. As the program expands in the later years, we expect to also involve workers from information theory and multi-agent dynamics

References

R. Adam and S. Solomon , Do Men Pass The Turing Test? in preparation.

E. Adi, M. Hasenbusch, M. Marcu, E. Pazi, K. Pinn and S. Solomon. Monte Carlo Simulation of 2-D Quantum Gravity as Open Dynamically Triangulate Random Surfaces. *Phys. Lett. B* 320: 227-233, 1994.

E. Adi-Japha, I. Levin and S. Solomon. Emergence of Representation in Drawing: The Relation Between Kinematic and Referential Aspects. *Cognitive Development* 13: 25-51 , 1998.

P.W. Anderson. More is Different. *Science* 177: 293, 1972.

R. Ben-Av and S. Solomon. Topology and Lattices. *Int. J. Mod. Phys. A* , 5 (2): 427-437, 1990.

R. Ben-Av, J. Kinar and S. Solomon. Cluster Simulation of the Ising Model on Dynamically Triangulated Lattices. *Int. J. Mod. Phys. C* 3: 279-295, 1992.

A. Blank and S. Solomon. Power laws in cities population, financial markets and internet sites (scaling in systems with a variable number of components). *Physica A* 287 (1-2): 279-288 , 2000.

H-G Evertz, M. Hasenbush, M. Marcu, K. Pinn and S. Solomon. Stochastic Cluster Algorithms for Discrete Gaussian (SOS) Models. *Phys. Lett. B* 254: 185, 1991.

J. Goldenberg, D. Mazursky, and S. Solomon . Creative Sparks. *Science* 285: 1495-1496, 1999.

Jacob Goldenberg, David Mazursky, and Sorin Solomon. Meme's the Word. *Science* 286: 1477, 1999.

J. Goldenberg, B. Libai, S. Solomon, N. Jan and D. Stauffer. Marketing percolation. *Physica A*, 284 (1-4): 335-347 ,2000.

U. Hershberg, Y. Louzoun, H. Atlan and S. Solomon. HIV time hierarchy: winning the war while, loosing all the battles. *Physica A*: 289 (1-2): 178-190, 2001.

M. Levy, H. Levy and S. Solomon. A microscopic model of the stock market; Cycles, booms and crashes. *Economics Letters* 45 : 103-111, 1994.

M. Levy, N. Persky and S. Solomon. The Complex Dynamics of a Simple Stock Market Model. *Int. J. of High Speed Computing* 8: 93 , 1996.

M. Levy, H. Levy, S.Solomon, *Microscopic Simulation of Financial Markets; From Investor Behavior To Market Phenomena* Academic Press, New York, 2000.

M. Levy and S. Solomon. Power Laws are Logarithmic Boltzmann Laws. *International Journal of Modern Physics C* , 7 (4): 595 , 1996.

- Y. Louzoun, S. Solomon, H. Atlan and I. R. Cohen. Modeling complexity in biology. *Physica A*, 297 (1-2): 242-252, 2001.
- E. Mar Or, E. Shir and S. Solomon. Solving Traffic Jams: Human Intervention or Self-Organization? To appear in *International Journal of Modern Physics*.
- L. Muchnik, F. Slanina, and S. Solomon. The Interacting Gaps Model: Reconciling Theoretical and Numerical Approaches to Limit-Order Models. To appear in *Physica A*.
- N. Persky and S. Solomon. Collective Degrees of Freedom and Multiscale Dynamics in Spin Glasses. *Phys. Rev. E*, 54: 4399, 1996.
- N. Persky, R. Ben-Av and S. Solomon. Dynamical Scaling from Multi-Scale Measurements. *Phys. Rev. B*, 51: 6100-6103, 1995.
- N. Persky, I. Kanter and S. Solomon. Cluster Dynamics for Randomly Frustrated Systems with Finite Connectivity. *Phys. Rev. E*, 53: 1212, 1996.
- N. Rubin, S. Hochstein and S. Solomon. Restricted Ability to Recover 3D Global Motion from 1 D Motion Signals: Theoretical Observations. *Vision Research* 35: 569-578, 1995.
- N. Rubin, S. Hochstein and S. Solomon. Restricted Ability to Recover 3D Global Motion from 1 D Motion Signals: Psychophysical Observations. *Vision Research* 35: 463-476, 1995.
- A. Shalit, S. Kirkpatrick and S. Solomon. Scaling, K-Cores and Networks, in preparation.
- N. M. Shnerb, Y. Louzoun, E. Bettelheim, and S. Solomon. The importance of being discrete: Life always wins on the surface. *Proc. Natl. Acad. Sci. USA*, 97 (19): 10322-10324, 2000.
- S. Solomon. The Microscopic Representation of Complex Macroscopic Phenomena. *Annual Reviews of Computational Physics II* pp 243-294, editor D. Stauffer, World Scientific 1995.
- S. Solomon. Stochastic Lotka-Volterra Systems of Competing Auto-catalytic Agents. in *Decision Technologies for Computational Finance*, edited by A.-P. Refenes, A. N. Burgess, and J. E. Moody, Kluwer Academic Publishers, 1998.
- S. Solomon. Behaviorally realistic simulations of stock markets; Traders with a soul. *Computer Physics Communications* 121-122 : 161, 1999.
- S. Solomon, Generalized Lotka Volterra (GLV) Models of Stock Markets in "*Applications of Simulation to Social Sciences*", pp 301-322 Eds: G Ballot and G Weisbuch; Hermes Science Publications 2000.
- S. Solomon. Self-Organization and Non-Self-Averaging Effects in Systems of Discrete Auto-Catalytic Elements pp 351-360 in *Multiscale Computational Methods in Chemistry and Physics* eds. A. Brandt, J. Bernholc and K. Binder, IOS Press 2001.
- S. Solomon and P. Richmond. Stability of Pareto-Zipf law in non-stationary economies in: *Economics with heterogeneous interacting agents*, page 141. ed. by A. Kirman and J.B. Zimmermann, Springer, Berlin-Heidelberg 2001
- S. Solomon, G. Weisbuch, L. de Arcangelis, N. Jan and D. Stauffer. Social percolation models. *Physica A* 277 (1-2): 239-247, 2000.
- I. Stenhill, S. Solomon and K. Wolowelsky. Dynamical Algebraic multi-grid simulations of free fields on Random Triangulated Surfaces. *Computer Physics Communications* 83: 23-29, 1994.
- Y. Stolor, M. Idel, S. Solomon. What Are Stories Made Of? Quantitative Categorical Deconstruction of Creation. *International Journal of Modern Physics C* 11 (4): 827-835, 2000.
- A.M Turing, Computing Machinery and Intelligence, *Mind* 49 (1950) 433.
- G. Weisbuch, S. Solomon. Self-organized percolation and critical sales fluctuations. *Int. J. Mod. Phys. C* 11: (6) 1263-1272, 2000.
- G. Weisbuch, S. Solomon and D. Stauffer. Social Percolators and Self-Organized Criticality in: *Economics with heterogeneous interacting agents*, p. 43 ed. by A. Kirman and J.B. Zimmermann, Lecture Notes in Economics and Mathematical Systems, Springer, Berlin-Heidelberg 2001.
- G. Weisbuch and S. Solomon. Social Percolators and Self Organized Criticality. in *Handbook of Graphs and Networks: From the Genome to the Internet*, pp. 113-132 eds. S. Bornholdt and H. G. Schuster, Wiley-VCH, Berlin, 2002.

When IT gets a mind of its own Emergence of Complexity in Artificial Systems

Simon Bakker and Bram Stuurman

AAMAS Invited Paper

Abstract

Artificial systems are designed to solve specific tasks and are often designed to be highly predictable and controllable. However, in this paper we argue that artificial systems can exhibit complex, emergent behavior that is not predictable or controllable.

In this paper, we argue that artificial systems can exhibit complex, emergent behavior that is not predictable or controllable. We discuss the emergence of complexity in artificial systems and the challenges this poses for designers and researchers. We also discuss the potential benefits of understanding and harnessing this complexity.

1. Emergence of Complexity in Artificial Systems

Artificial systems are designed to solve specific tasks and are often designed to be highly predictable and controllable. However, in this paper we argue that artificial systems can exhibit complex, emergent behavior that is not predictable or controllable. We discuss the emergence of complexity in artificial systems and the challenges this poses for designers and researchers. We also discuss the potential benefits of understanding and harnessing this complexity.

The emergence of complexity in artificial systems is a phenomenon that has been observed in a wide range of domains, including robotics, artificial intelligence, and multi-agent systems. This complexity arises from the interactions between the components of the system and the environment, and it can lead to the emergence of new, unpredictable behaviors.

When IT gets a mind of IT-self

Emergence of Complexity in Artificial Systems

Sorin Solomon and Eran Shir
Hebrew University of Jerusalem

Sorin@cc.huji.ac.il , eran@cogniview.com

Abstract

We describe some ways in which the emergence and self-organization may help the design and development of computing and information based complex artifacts

The physical, biological and social domains are full of complex systems that emerge spontaneously from the interactions between the simpler elements existing in nature or in society. In recent years, though, with the proliferation of man-made artifacts, information technology (IT) and networking, many artificial systems have acquired a large number of elements and started to emerge collective complex phenomena. For people unaware of the complexity emergence mechanisms, it came as a surprise that a bunch of man-made artifacts would develop a mind of themselves to the level that their behavior couldn't be described, understood and utilized with the usual engineering methods.

Some examples of behavior in such systems are:

- growth and information flow dynamics in computer networks,
- the multi-agents ecology,
- the world-wide-web contents evolution,
- commerce arenas filled with negotiation agents and
- traffic networks.

All of these phenomena have in common the following facts:

- They are all built from or operate on multitudes of similar components which interact intensely with each other.
- The technological/research field in which the components were initially designed and developed is not adequate to analyze, study and manipulate the overall system. For example, when a computer network grows so much that it can be looked at as a statistical ensemble, understanding the computer's

hardware architecture or its operating system design is just not relevant to the task of identifying emerging collective qualities of the network.

- While there is immense difference in the detailed structure of the elementary components composing each of these systems, their complex collective features can be shown to be analogous. In fact, sometimes it doesn't even matter whether the basic elements are man-made or natural ones.

1. Designed to Emerge; Bottom-up design of self-organized complex systems

In spite of this analogy with natural complex systems, the systems made of man-designed elements offer an unprecedented opportunity. While the complexity emergence in biology and society took place unintentionally, often as a result of chance, in man-made objects the emergence can be planned and influenced through the way in which the elementary objects are designed and displayed. The convergence of complexity and IT paradigms enables for the first time the deliberate development of a new breed of complex applications and systems:

Systems designed so that the simple interactions between their elementary components create collectively a desired global behavior.

Using the methodologies developed to study complexity phenomena, one can design complex behavior in artificial systems. This paradigm has the potential of transforming entire fields of IT applications. It may boost the development of directions that otherwise would reach a dead-lock, or would be increasingly limited by a mere logarithmic

growth with the computer power. For certain applications, e.g. network routing, orthodox, global, hierarchical design methodologies can be proved to be inapplicable, even if the hardware capabilities will continue to grow exponentially. Some examples of such fields and applications are:

- Car traffic control optimization
- Texts translation
- Information retrieval and knowledge extraction
- Network routing, content routes optimization
- Adaptive collaborative software development
- Real time air traffic scheduling adaptation
- Autonomous UAV (Unmanned Aerial Vehicle) flocks
- Bioinformatic research algorithms and tools optimization
- Integrated robot flocks (or "dividuals", see below)

Below, we look further into some of these examples, but first let us discuss the generic advantages and problems of such "bottom-up" designing of complex systems. The most important are their

1. freedom of scale
2. adaptability and robustness
3. ease in design, implementation and maintenance

Let us elaborate these 3 points:

1. In algorithms theory the $O(N)$ computational complexity measure is the corner stone of the entire science. The functioning of many of the "bottom-up" designed systems is often "simple" in this respect: all of the work is done locally and the issue of the cost at the level of the entire system does not even arise. Nobody has yet submitted the next year budget for the world web. Similarly, one may add autonomously traffic lights, install new routers, or add additional UAVs to a flock as much as one wants, and the robustness or cost of the rest of the system will not be significantly affected. The local adaptive-agent based control algorithms become unavoidable in applications where in principle there are possible interactions between any arbitrary elements of the system. In such examples, the global hierarchical algorithms have to deal with an exponentiation in the number of alternatives to be considered. The local adaptive agents are usually dealing well with avoiding to even considering the cases that do not actually appear during their current functioning. Usually this is a great

relieve as most of the computationally "worst cases" never appear in practice.

2. The high non-linearity of certain systems, like the control of traffic lights in a large city, makes it virtually impossible to find the optimized global algorithm for these systems. However, using methodologies developed in the study of complex phenomena such demands as optimality and predictability become possible, and in addition the systems themselves become understandable.

It is extremely hard, and in many cases plain impossible, to build a global logic system with good reflexes. By good reflexes we mean the ability to rapidly adapt to changes in an adequate manner. The need to tell in a top-down manner to each element what to do next makes it extremely difficult to react fast in the correct way. However, when one gets a collective behavior as an emergent character of a multitude of elements, adaptation comes naturally, and only in regions where it is needed.

3. Another important advantage of "designing for emergence" is the fact that it makes the applications easy to design, program and maintain. This is crucial for keeping-up the expansion rate of computer uses in domains with low programming literacy. There are simply not enough programmers around to design, implement and maintain classical software tools for every user.

The big disadvantage, but truly the big challenge, is that there exists no standard theory and practice for designing systems that emerge self-organized complexity bottom-up. It is obvious that this will require not only technological progress but also scientific effort. On the other hand its results might fare much beyond efficiency and impinge upon the frontiers of human knowledge.

Let us look now on several promising examples of IT applications that will benefit extremely from applying complexity concepts in the development

2. Specific Applications

Agent based and collaborative applications

There are two approaches, about the agents cooperating in the management of self-organization of

complex tasks.

- The more traditional approach is making the agents more and more sophisticated, with AI core etc.
- The second and more promising approach aims "collaborative applications" i.e. applications that encompass multiple agents in a collaborative manner in order to fulfill a task.

The main conceptual limitation of the current collaborative application approach is that it tries to divide the global task into small independent segments performed by independent agents. This external division and stream-lining of the tasks preserves some of the rigidity and weaknesses of the logical-tree, centralized methods. The challenge is to design the right interaction protocols and feedback mechanisms that to insure the self-organization of the work in an optimal way.

Traffic Lights on the Spot

Centralization was a major paradigm for control theory since its incubation. And for good reasons. A centralized control system is deterministic, and can be developed in a straightforward manner. But currently we are in a convergence point of several processes that suggest a coming paradigm shift. First of all, the amount of computer power which can be squeezed into a portable computerized device has grown exponentially. Even more important is the exponential growth in communication abilities, which makes both land based and wireless communication technology a mature, cheap and accessible option. These technological developments are met with the emergence of new research paradigms described above, aiming to study the complex systems behavior that emerges from the interaction of the system's components. Among control applications most fit for turn over, traffic management looks especially promising. It is composed of many identical components, which need to adapt rapidly to changing circumstances. A centralized system, wishing to adapt to a car accident will need to check all of the global ramifications of a change, while a local rules system will be built so that such a process will occur naturally.

Goals of the research:

- Create a simulation tool of a distributed traffic signal network, which manage the flow of traffic in a city.
1. Develop local rules of conduct for the traffic signals, depending on information gathered from local environment (local congestion and pressure

information, current rules of near by traffic signals). Study different measures of 'good flow'.

2. Develop a methodology of transforming centralized policy into distributed local rules.
3. Study optimization methods of the local rules.

Ultimately, the very concept of traffic light may come under revision. The same information, instructions, regulations and signals which are currently communicated through road signs might be transmitted and may be even enforced at the level of the cars: your car will slow and stop at "red lights" unless you explicitly choose to override the order it receives from the "traffic lights" system.

Self-Organized Unmanned Aerial Traffic

The concept introduced above: emergence of self-organized traffic is particularly fit for Unmanned Aerial Vehicles (UAV) navigation:

- in aerial traffic there are less intricate obstacles than in ground transportation,
- the danger for hurting humans is less direct in unmanned vehicles
- humans do not have a head start in 3D navigation skills compared to computers (as opposed to hundreds of thousands of years of 2D navigation).
- The 3D geometry allows for the formation of large flocks that form, join and split according to the individuals initial and destination points.

Relaxing the demand for a central control center that communicates with all of the UAVs in real time and directs them what to do, will enable creating large flocks of UAVs, that will be able to travel much further from their home. In addition it will enable to enlarge significantly the set of possible tasks such flocks will undertake, due to the fact that their reaction time scale will be reduced dramatically. In the case of Cruise Missiles Flocks the possibility to share their location and visual information, may result in a dramatic improvement of their navigation and target identification skills.

From Integrated Robot Flocks to Dividuals

The idea above that a collection of objects sharing information can be more efficient than a more

intelligent single object has tremendous potential far beyond the realm of self-organized navigation.

As opposed to humans, robots can share and integrate directly visual, and other non-linearly structured information. They do not suffer from the humans need to first transform the information in a sequence of words. Moreover, the amount, speed and precision of the data they can share are virtually unlimited.

Like in the story of the blinds feeling an elephant, at the best the communication channels typical to humans are not sufficient to insure fast, precise and efficient integration of their knowledge / information / intelligence. By contrast, robots with their capability to determine exactly their relative position and to transmit in detail the raw data that they collect are perfectly fit for the job perfectly.

So, in such tasks, while

1 human > 1 robot

one may have

100 humans < 100 robots.

This may lead to the concept of Integrated Robot Flocks which is much more powerful than the biologically inspired ants nest metaphor because the bandwidth of information sharing is much more massive. Rather than learning from biology, we may learn here how to avoid its limitations.

For instance, rather than thinking of the communicating robots as an integrated flock, one can break with the biology (and semantics) inspiration and think in terms of the **divided individual** (may we call them **"dividuals"**):

Unlike biological creatures, the artificial ones do not have to be spatially connected: it may be a great advantage to have a lot of eyes and ears spread over the entire hunting field. Moreover, one does not need to carry on the reproduction organs when the teeth (mounted on legs) go to kill the pray (the stomach too can be brought in only later on, in case of success).

Of course, a good idea is to steal from time to time the control on somebody's else wings (as a form of Non-Darwinian evolution). Contrast it to the usual way real animals exploit the other: they are constrained by their biological reality to first degrade the wings to simple molecules at which stage, it is too late to use them for flying.

The anthropomorphic- biological grounding here is a liability that one has to free oneself from, rather than a source of creative inspiration.

Encounters of the Web kind

The emergence of a "thinking brain" by the extension of a distributed computerized system to an entire planet is a recurring motif in science-fiction stories and as such a bit awkward for scientific consideration. Yet, if we believe that a large enough collection of strongly interacting elements can produce more than their sum, one should consider seriously the capabilities of the web to develop emergent properties much beyond the cognitive capabilities of its components. As in the case of the Integrated Robots Flocks, the relative disadvantage of the individual computer vs. the individual human is largely compensated by the "parapsychological" properties of the computers: any image perceived by one of them at one location of the planet can be immediately shared as such by all. Moreover they can share their internal state with a precision and candor that even married couples of humans can only envy.

A serious obstacle in recognizing the collective features emerging in the web is the psychological one: people have a long history of insensitivity to even slightly different forms of "intelligence". In fact various ethnic / racial groups have repeatedly denied one another capabilities in the past. Instead of trying to force upon the computers the human version of intelligence (as tried unsuccessfully for 30 years by AI), one should be more receptive to the kind of intelligence the collections of computer artifacts are "trying" to emerge.

An useful attitude is to approach the contact with web in the same way we would approach a contact with a extraterrestrial potentially intelligent being. A complementary attitude is to study the collective activity of the web from a cognitive point of view, even to the level of drawing inspiration from known psychological processes and structures.

Making the Net Work

Network routing is probably the most natural application for the complexity approach. First, routers are junctions of a network (in itself a central concept in complexity). Secondly, there is no conceivable practical way in which the network routing problem can be solved by a hierarchical, global algorithm. Thus routing is solved through the use of local routers that

know where to send the information that is flowing through them. However, due to the fact that the development of routing algorithms still resides exclusively in the realm of computer engineering and computer science, the routing algorithms themselves do not pass the line of triviality. Billions of dollars are lost every year in damage due to bottlenecks, congestions, Denial Of Service caused by malicious attacks, negligence or simply by mistake or mis-design. Many other applications and businesses do not get transferred to the Internet due to these problems. Building systematic algorithms, which utilize the fact that the Internet has grown big enough to be thought of as a statistical ensemble, promise to create a more flexible, vibrant, trustworthy Internet.

Bringing the Rosetta Stone into the Chinese Room

The translation capability became one of the stumbling stones of AI. The belief that translation capability is tantamount to artificial understanding leaves one with a very bleak prospect for achieving it soon or easily. Yet the proliferation and globalization of knowledge makes massive online translation and processing of documents, especially on the Internet, an unavoidable and ubiquitous task.

The problem is wider than simple translations from one national language to another. One may equally refer to various technical jargons as technical "languages" and to writing executive abstracts as translations from verbose texts to their terse versions. "Translations" may also be the transformation of plain unstructured text into some structured canonical template (code to be used as input to further processing by specialized programs).

One possible approach is the bootstrap one: using the abundance of already translated documents as a resource for new translations. The bulk of documents written in a given "language" can be used as an operative definition of that language. Documents expressing in different "languages" the same thing can be used as operative models for "translations" between those "languages". This would be the automatic version of the techniques applied at the time for translating the old Egyptian hieroglyphs using the Rosetta stone [On that stone, the same text appeared written in hieroglyphic script and in parallel, in Greek and Demotic].

Rather than designing heavy AI machines with large databases and sophisticated "intelligent" algorithms, all one has to do is to let a host of adaptive Internet agents to adapt the chunks of text of the already existing "translations" to new texts. If the result is not satisfactory

(may be due to the lack of a large previous pool of "translations" between the two "languages"), the human corrections necessary to improve it, will insure that the current document can be used in the future as a basis for better performance (supervised learning). Of course, the agents may be endowed also with the capabilities to use transitivity: given a set of translations from a "language" A to a "language" B and a set of "translations" (in roughly the same subject) from B to C, they will produce automatic "A-to-C translations".

Instead of having to repeat each time the "translation" process, one would use the Internet with all the previously created documents as a global intelligent Chinese Room populated by many cooperating adaptive agents and furnished with a huge Rosetta stone.

An Agent-Managed Ant-Based Network Control System

David Legge; Peter Baxendale

Centre for Telecommunication Networks,

School of Engineering,

University of Durham

d.n.legge@dur.ac.uk; peter.baxendale@dur.ac.uk

Abstract

An Ant-Based Routing System has been constructed and it has been shown to find optimum routing strategies under static conditions. With the introduction of dynamic traffic situations onto the network, the ants do not always display optimum behaviour. It has been shown that parameters cannot be set to a single value to satisfy both steady state and transitory conditions optimally. It is therefore proposed to use an Agent on each node to adapt to changing conditions on the network. This can be achieved by manipulating the parameters with which the ants operate. Since the agent is classifying different states and the associated actions best suited to them, methods of model-free Reinforcement Learning, such as Temporal Difference and Q-Learning are discussed, along with the problems of a continuous domain.

1 Introduction

There are many papers describing Ant-Based Routing in a telecommunications network, such as Dorigo [1997], but many of these systems use empirically-found parameter values, such as Schoonerwood [1996]. It has been shown that there is a trade-off in setting these parameters, between system-stability and quickness of response; as there is in any control system. It is also the case that parameters have been used in simulation, that would be unobservable in real networks; for example the synchronisation of absolute time over a network is by no means a trivial task, especially to the precision and accuracy required to measure the propagation of ants across a link.

It is proposed that if an agent could learn to distinguish between different "states" the network might exhibit (e.g. Stability, Start-up, etc) then these parameters could be adapted to minimise undesirable events and states (e.g. Data loss, delays, instability).

A simple example of this is that Schoonerwood [1996] uses an initial training period to allow the network to find equilibrium. This, it could be argued, is unrealistic, as once a network is initialised, an administrator would want to admit traffic to it immediately; especially if it is a commercial network recovering during working hours. In the proposed solution, a reasonable route, whilst not necessarily optimal due to the constraint of bounded rationality, is found very quickly, at which point traffic could be accepted.

The time required for this initial transitional period - before equilibrium has been found - should be as short as possible. It would be advantageous to be able to set relatively high *gains* at startup, but then when the system has settled reduce these gains, as they are not so suitable for steady state operation. This function could be taken on by an agent, supervising the ants and learning strategies to cope with different circumstances.

It is important to note that traffic transversing an integrated telecommunication network¹ (to which there has been a large shift in the recent decade) has both stochastic and deterministic elements, so while prediction is possible in some circumstances, in many cases it is not.

This inherently random nature, almost chaotic, makes a single generic optimal - or near-optimal - strategy impossible. Attempts to apply Artificial Neural Networks to the prediction of traffic levels on a network have met with little success; a strategy is learnt that is then too simplistic to cope with any change in circumstance.

Since prediction is essentially a fruitless task, the role of an agent employed to adapt the control system of a network, must be to react as quickly as possible to situations, once they become apparent.

Ant-based systems are used for their self-organisational properties, but of course this is a form of unsupervised learning, implying only a reactive role. They are

¹Meaning one that carries data, voice, multimedia etc.

used to determine the best route by testing all routes constantly, and representing all the routes internally as a probability. The route nominated by the ants is the one with the highest probability. This route will ideally be the instantaneous best route, but there is inevitably some delay in the response. Some consideration must also be given to stability - constant switching of routes can cause problems in reordering data at the receiving end if it arrives out of sequence. To counter this, if a route is switching rapidly, an extra threshold can be introduced.

Consider two nodes connected by two different routes, (a) a direct link and (b) a circuitous route, involving many hops. Under normal loading all traffic will be sent the direct route, and rightly so; in the ant-scheme this will be exhibited by the short route accumulating a large probability at the expense of the longer route.

Now consider when this route becomes congested; under normal operation it will take a long time for the ant probabilities to change sufficiently for a route change to take place; in this case it would be advantageous for the ant-parameters to be adapted; and this requires Intelligent Control.

Assume the shorter route described above had probability of 80% compared to 20% for the longer one. If each Ant has the affect of changing the probability by two or three percent, it will take ten to fifteen ants to equalise the routes probabilities. During this time traffic is likely to be discarded, which can cause even more congestion through retransmission schemes.

From a different perspective, the agents can be considered from the view of independent "blind" game-playing agents, analogous to those discussed in Kapetanakis [2002]. In this paper, some probabilistic characteristics are used to challenge the agents into making risk-based decisions. In our situation the game being played is a number of magnitudes more complex, and with greater degrees of freedom.

Indeed, the situation outlined here is of a continuous nature, not a well-defined discrete domain. This inherently requires some mapping to enable it to be tackled by a digitally-based computer. A difference also exists in that success is not so well defined, and has a delay in response - known as deadtime.

Appointed to further work is both the situation in which the agents observe the decisions of other agents by examining the parameters embedded in the ants produced by other nodes, and also where all the agents have direct lines of communication.

2 Aim

The aim of this research is to show that an intelligent agent can be used to modify the parameters associated

with the Ant-system to make both the convergence from start-up, and the response to changing conditions within the network, quicker.

The parameters are changed on a per-node basis - i.e. an agent on each node is in control of the parameters used on that node. This means that, at this stage, the agents are acting independently from each other, although the ant system is an implicit form of communication. This can be described as a fully-distributed control system, although there is this inherent communication between the nodes, in the form of ants.

An agent could observe how other agents are acting by checking the parameters carried by ants originating from other nodes. A further stage would be to enable direct communication between the agents, but this is left to Further Work.

3 Current Work

A significant amount of research has investigated the self-organisational properties of Ant-Based systems on telecommunications - either for load-balancing and/or for routing. This type of approach was developed by Dorigo [1997], and also taken up by Schoonerwood [1996].

These techniques are extended in many others, including Vittori [2001], in which Q-learning is employed to update the probabilities.

However, these papers use single-valued parameters, such as rate of production of ants, rate of deposition of pheromones etc, that are empirically obtained. These are not changed at runtime, or indeed off-line. There is little description of how these values are obtained, and they are unlikely to be suitable for different sizes of networks.

The Vittorri paper utilises Q-learning to update the routing tables and found performance gains in doing so. In comparison this paper utilises Q-learning to modify the parameters used by the ants. The paper goes on to compare a "standard" ant-based system, an ant-based system with Q-Learning, and finally a mobile-agent based agent system.

The agent decisions are made without conferring with the other agent(s), indeed the agent is not aware of the presence of other agents.

As previously referred to, Kapetanakis [2002] discusses agents participating in games with other agents, where the reward gained at the end of the round is a function of the decisions made by all the agents. e.g. Agent 1 decides on action A, while Agent 2 opts for action B. The games are taken further by the addition of a probabilistic affect, so that the the reward is $f(A, B, P)$.

The agents use Q-learning, and some are also given an optimistic outlook (Where $\text{MAX}(Q)$ is used instead of just Q). The agents are able to find joint strategies, which take into account the Expected Value² rather than the maximum value of the reinforcement.

4 Ant-System

This section describes the implementation of the Ant-System used. It is broadly based on the system discussed in Schoonerwood [1996].

The basis for the ant system is many simple-acting entities displaying complex behaviour on the macro-scale. To this end the ants are small packets of data that are kept as simple as possible. Carried within the ant are: Source, Destination, Amount of Pheromone and Deposition rate. Rather than carrying code with every ant, the processing is performed by the node. This minimises the amount of control traffic required - an important consideration on a real network.

The Rate of Ant Production is a self explanatory parameter, and at generation the ant is given a uniformly-random destination, chosen from all the known nodes on the network. Each ant is also branded with an initial amount of Pheromone and a Deposition Rate (which together describe its longevity). Once generated, and at each node along its journey, it selects its next-hop by the generation of a uniformly random number and the consultation of the local probability table associated with the ants final destination.

On arrival at a node the ant deposits some pheromone, having ensuring the ant has not returned to its source. If the ant has looped back to its source, it is discarded (or squashed to continue the overall analogy). The amount of pheromone left by the ant is a simple percentage (defined by the Deposition Rate) of the current amount the ant has. Before this is added to the probability table (for pheromone table) it is multiplied by the (relative) available size of the traffic queue for the link just used by the ant. The resultant amount of pheromone is added to the local table regarding the ant's *source*, i.e. the ant reinforces the *return* route, *not* the route it is currently taking. The tables are then normalised to sum to unity; this has the affect of rewarding the route just used, but also punishing all the routes *not* just used. Note that this is functionally equivalent to the process of pheromone evaporation, used in other Ant-Based Routing implementations. If this wasn't the case, the level of pheromone would tend to infinity.

A Minimum Probability is applied to ensure *all* routes are tested periodically; since the system is dynamic, constant exploration is required. As shown later, a greedy-agent is likely to perform badly. If the ant has reached its destination it is then discarded.

²Expected Value in its probabilistic sense, i.e. Sum of (reward x probability).

The output of the ant-colony is the best-next-hop for each known node. This is the highest probability link, except for when an extra threshold is applied - i.e. a link must exceed the probability of the current best route by some value for the recommendation to change.

The effect of the ant parameters on the probabilities have been investigated with and without traffic. It has been shown that it is possible to have a very quick response, but with a relatively noisy steady-state condition, or a smoother steady-state but a slower response. The Rate of Production also has an affect on the response.

An assumption is made in Ant-Based Routing that each and every link is bi-directional; the discussion of the validity of this assumption is beyond the scope of this paper. In Vittori [2001] routes are reinforced in both forward and backwards directions; although this may aid speed of convergence, the forward reinforcement has no logical basis - the feedback may be incorrect, and lead to problems with sub-optimal routes being continually reinforced - thereby exacerbating any congestion on that route.

The next section describes the agent used to strategically manipulate the parameters that control the ants to improve their performance.

5 The Agent

An agent is present on each node and monitors the ant colony on that node. The agent receives reports from the colony, both regularly and when a route change occurs.

The Agent is informed of all the parameters associated with the ants (Rate of Production, and those carried in the ants body), as well as the threshold value for route change, and is able to adjust all of them.

The Agent is also told the current probability of the preferred route, as well as its mean, standard deviation and gradient. These statistics are designed to show the current state of the network as well as some history - it also gives information about the performance between the reports received from the ant colony. Obviously, the windowing function for calculating the mean, etc, is an important property and is set to highlight long-term trends, but average out short term fluctuations caused by individual ants arriving. An opposing consideration is the delay caused by an overly long window, making the response more sluggish.

The Ant-System is a separate entity and can operate with or without the Agent and is reactive in nature. The agent provides a supervisory role, and is thus more proactive in nature - complementing the purely firefighting nature of the ants.

To compare the success or failure of the agent, a comparison can be made between the performance of the Ant-system with and without the supervision of the agent.

The type of learning strategies that could be employed by the agent are discussed in the next section.

6 Agent Learning Strategies

Since the agent is operating in a complex domain, which is difficult to predict due to its stochastic nature, a scheme that does not require a model would be beneficial. In terms of Reinforcement learning, this discounts Dynamic Programming. However, there remain Monte Carlo, Temporal Difference, and Q-Learning techniques.

A complicating factor of all these techniques, is the continuous domain in which the agent must operate. This requires an Analogue to Digital Conversion role in the system. This is because by adjusting parameters at run-time, there is a potentially infinite variety of actions, as well as an infinite number of states.

Q-Learning is where an agent learns state-action values, and is a potentially profitable area for research, as is Temporal Difference Learning.

7 Experimentation

The aim of the experimentation is to show that there is an improvement in performance when the parameters governing the ants are changed dynamically at run-time. Thus taking account of changing conditions on the network.

Because of the inherent complexity involved in testing a network, two relatively simple simulations are used. Firstly, a network where there is a number of equally long routes between the two nodes of interest and one of them becomes congested; while the second is where there is an obvious shortest route which then becomes congested (with at least one other alternative route).

Of interest in the test, is the reaction of the ant-system (both managed and un-managed) to congestion; the speed of response is of importance, as is the amount of data-traffic lost (the ants, by their nature, are expendable as information is gained by them not arriving). A further concern is that the routing should not constantly change between two or more routes.

For both scenarios, the Managed Ants performance must be compared with that of the Unmanaged Ants. As an intermediate step, a predetermined strategy, with *a priori* knowledge of the traffic conditions, is implemented that will also improve the performance. This

will give a further benchmark with which to contrast the performance of the full system.

7.1 The Simulator

The simulator being used is Network Simulator - *ns*. It is designed to model TCP/IP³ networks. These networks are connectionless; that is to say packets of data are transmitted without setting up a definite route first. This means that at each hop, packets are routed according to the instantaneous best route. Packets from the same flow do not necessarily follow the same path. Consideration of Circuit-Switched and Connection-Oriented networks are left to Section 9.

7.2 Network One

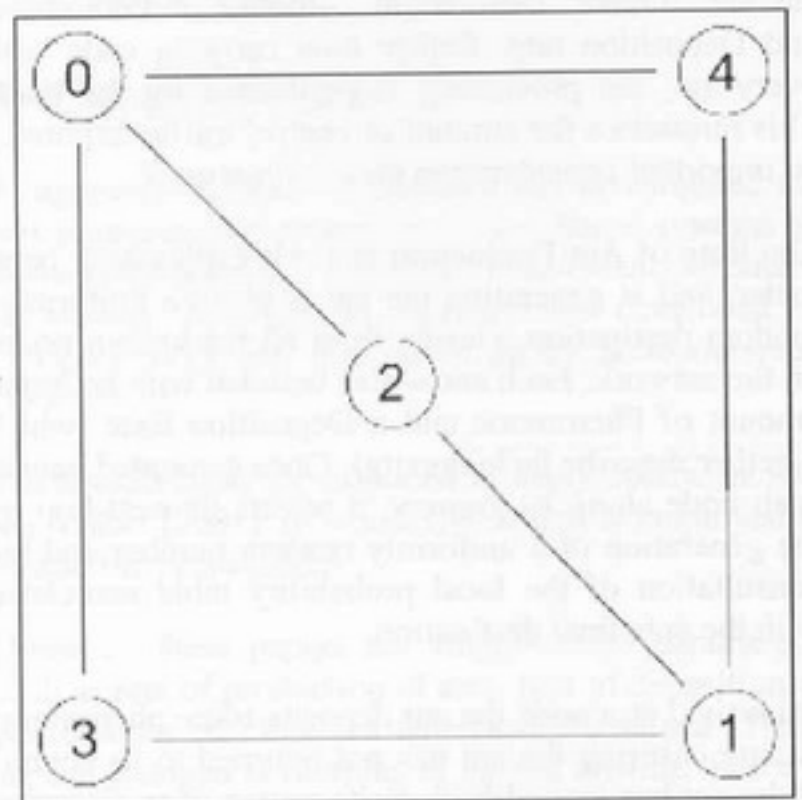


Figure 1: Diagram of network with numerous equally shortest routes

The network to be tested is shown in Figure 1. It is a simple five node network. The important feature is that there are three equally-long routes between Node 0 and Node 1, one of which, as described above, will at times be congested.

7.3 Network Two

This scenario exists when there is a distinct shortest-path between two nodes as well as longer alternative routes, not sharing any links. The topology is shown in Figure 2.

Where the shortest route becomes congested, the ants must react as quickly as possible to the congestion. They must overcome a large amount of inertia represented by the high probability built-up by the shortest

³Transport Control Protocol / Internet Protocol, as well as UDP, User Datagram Protocol, a lighter weight version of TCP

route- this inertia is greater than that of the previous scenario.

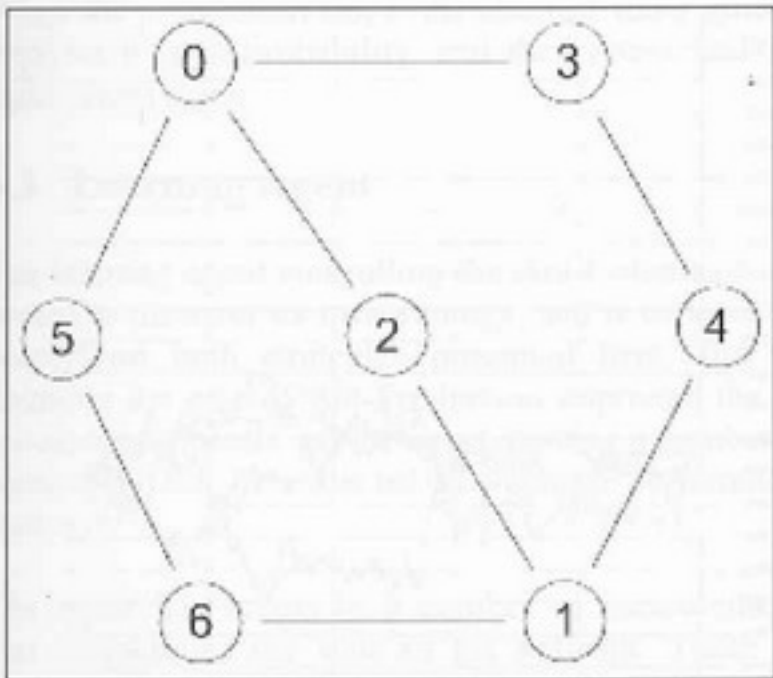


Figure 2: Diagram of network with a definite shortest route

7.4 The Traffic Scenario

Both network scenarios will be required to transmit some user-data, which we shall describe as real-time, loss-sensitive data. This shall be transported from Node 0 to Node 1, in both network topologies. The traffic shall be offered to the network continuously for a duration of 25s, starting after 15s.

Problems would be caused by the network dropping the user-data, excessive delay or excessive jitter - variation in the inter-arrival time of packets, often caused by switching of routes.

The situation under which the network will be tested is as follows. The traffic causing congestion, shall represent other network load and will be set at a level that will not, by themselves, cause packets to be dropped, other than ants.

The simulation will start with no data-traffic on the network. The user-data starts after fifteen seconds, but after twenty seconds two more traffic flows start.

Firstly, a constant-bit-rate (CBR) flow, that uses the majority of the bandwidth available on the link between Node 0 and Node 2. Secondly, a smaller Variable-Bit-Rate (VBR) flow also uses the same link.

This combination of traffic will result in intermittent congestion during the time when both flows are present on the network. During congestion traffic will require storage in queues, waiting to be transmitted, and ant-packets will be dropped - having been assigned a lower priority than traffic.

These two flows both stop after thirty five seconds, with a further fifteen seconds to allow re-stabilisation before the end of the simulation.

8 Results

The results presented here are a comparison of different solutions, based on the use of Ant-Based Routing. The control test uses no adaptation of the parameters – they remain constant throughout the simulation. Secondly, a hardcoded manipulation of the parameters is used, for which the traffic-levels are known beforehand, these should show that improvements to the Ant Control are possible. Thirdly, an agent would be allowed to change the parameters as it sees fit at runtime, and according to strategies it learns.

The results are shown in the form of graphs of the probabilities for each next-hop from the source node of user-traffic. On the same graph is the mean probability of the ant-nominated route. The sample window (one sample being an ant-arrival) has a length of 250, and is a cyclic buffer. This mean value, as well as the Standard Deviation, is to be used as a characteristic of the performance of the network by the agent.

8.1 Ant-Based Routing

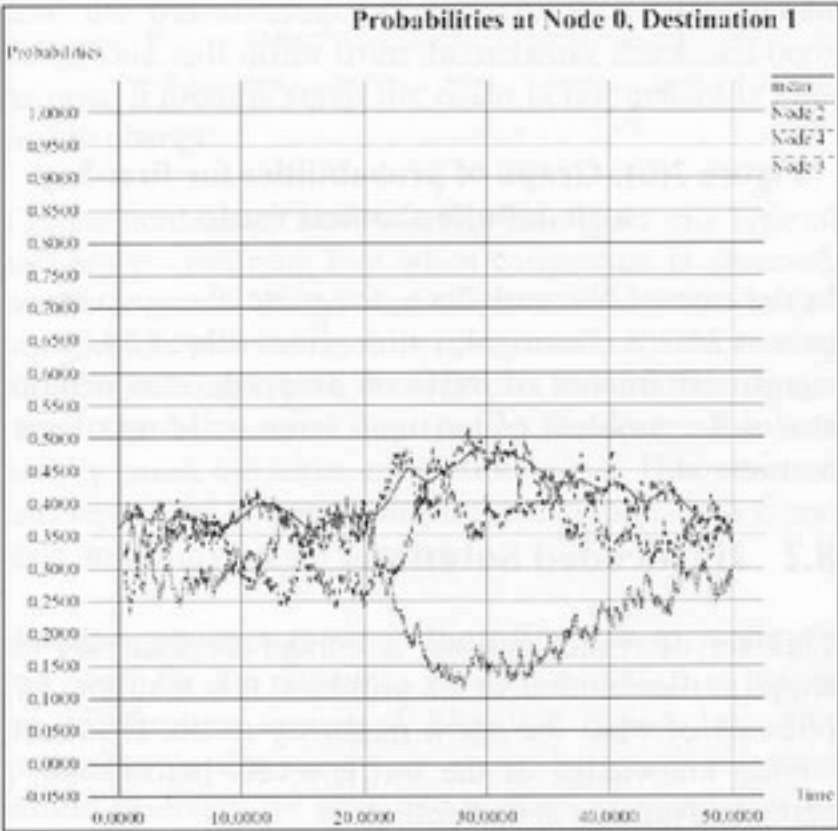


Figure 3(a): Graph of probabilities for first-hop for numerous equally shortest routes

This run uses “standard” Ant-Based Routing. The network stabilises, almost immediately, since there is nothing to choose between the three routes; in this scenario one route must be chosen, however arbitrarily, for the sake of network stability. When the traffic conditions change, at 20s, the ants take a little time to react. The simplest strategy is the un-managed Ant-

Basted Routing, and as such, it is expected that the other two stratagems out-perform it.

Figures 3(a) and 3(b) show the probability for each route from Node 1 to Node 0 for Network One and Two respectively.

In the case of Network One, the control traffic is not using the congested route when the traffic starts (at 20s), however, it can be observed that the probability of the congested route falls sharply, from this time. After 35s, the route starts to recover, showing constant exploration.

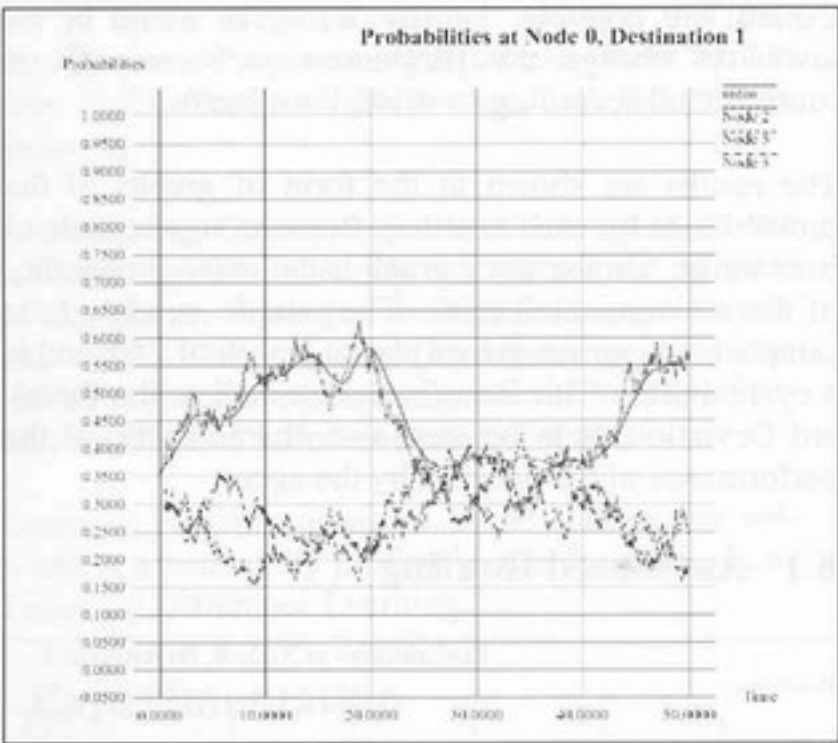


Figure 3(b): Graph of probabilities for first-hop with definite shortest route

In the case of Network Two, the route changeover occurs at 24.30s. During this time, from 20s to 24.30s, a significant amount of traffic is dropped. This demonstrates the problem of having a large build-up of system inertia.

8.2 Hardcoded Solution

This set of results utilises a human-programmed attempt at the solution to the problem. It is a simple hypothesis of what the agent might try to do. However, having knowledge of the traffic-levels beforehand, a certain advantage is evident.

In both scenarios, shown is Figures 4(a) and 4(b), the rate of production is doubled, to promote exploration of other routes, and to emphasise the congestion present on the network - since ants in excess of the capacity will be discarded.

Again on Network One, shown in Figure 4(a), the congested route is not being used by the user-traffic when the congestion starts, but it can be seen that the probability of the congested route drops off more sharply

compared to the ordinary ant-based routing, but then recovers when the congestion is removed.

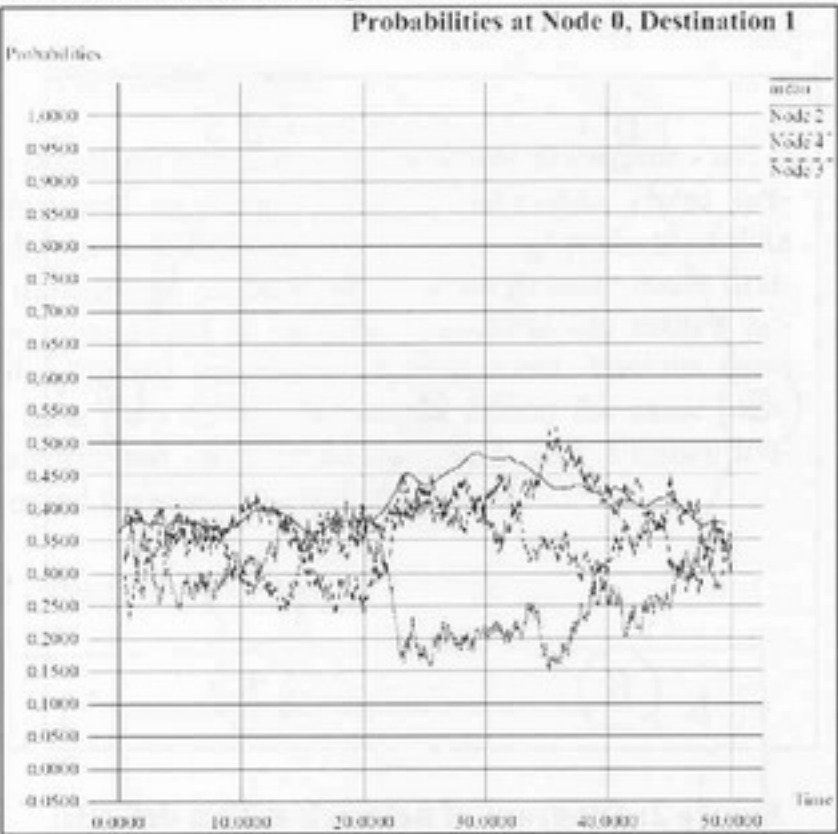


Figure 4(a): Graph of probabilities for first-hop for numerous equally shortest routes under hardcoded strategy

Because the routes are equiprobable from the beginning, no settling time is required by the network, but the ants change the best route frequently to begin with.

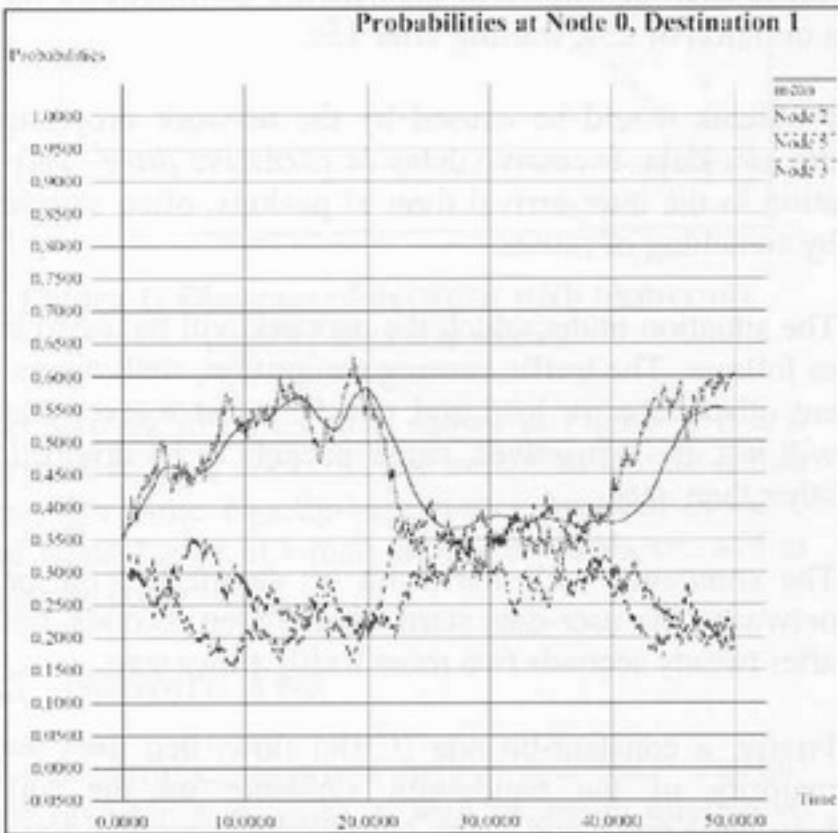


Figure 4(b): Graph of probabilities for first-hop with definite shortest route under hardcoded strategy

For Network Two, Figure 4 (b), an improvement in the response is shown by the change of route occurring at 23.60s. This an improvement of more than a second to

the ants response - almost a 20% reduction in time to decisively respond.

When the congestion stops, the shortest route quickly recovers a large probability, and the system finds its equilibrium again.

8.3 Learning Agent

The learning agent controlling the Ant-Colonies is expected to discover its own strategy, and is expected to outperform both strategies presented here. Just by doubling the rate of Ant Production improved the response significantly, so the agent varying a number of parameters can be expected to improve performance further.

The agent has access to a number of measurements that characterise the state of the network. These include the instantaneous probabilities of each route (at the time of each report), the mean over the last 250 samples, and the Standard Deviation (SD) from that mean. Figure 5, which is the SD from the standard Ant-Based Control on Network One, shows that the SD has characteristics that could be detected and used as a symptom of change on the network. Comparing this with Figure 3(a) it can be seen that the large spike corresponds to the fall in probability caused by the congestion.

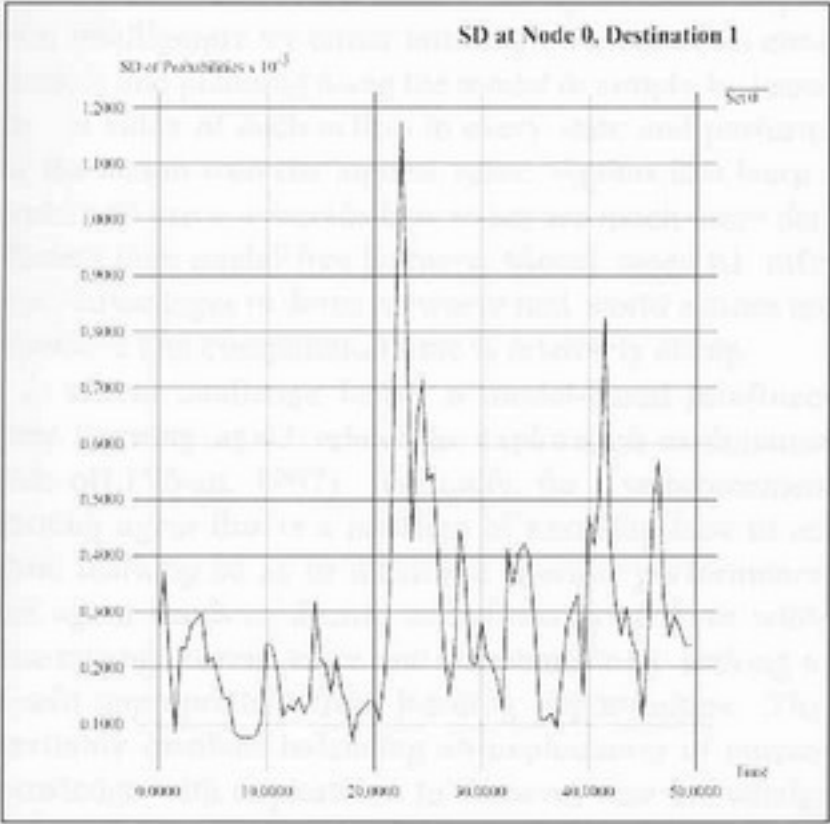


Figure 5: Graph of the windowed Standard Deviation for Network One under Ant-Based Control

The agent also has a record of the number of ants dropped by each queue on each node, and the time at which the route last changed. It has control of the parameters used to control the ants, such as their production rate, and their pheromone levels. It is also able to raise the threshold required to change the route.

9 Further Work

There is potential for the agents on different nodes to communicate to improve their effectiveness. This includes both direct and indirect communication.

The first stage is simply indirect communication. It would merely require the agent to observe the parameters being used by other agents, and embedded in the ants. Since the ants pass through the nodes anyway, little work is required to change the implementation to allow the agent access to the information. This technique is also known as stigmergy. Investigation into using this information is the greater part of the work.

The second stage will be to add the ability of agents to communicate directly across the network using a standard Agent Communication Language (ACL), such as a FIPA implementation. In the work described here, the agents have been working independently. Due to the finite propagation time over the network in real-time, the agents will *always* have to act in an isolated manner to some degree.

On the Telecommunications side, there is the consideration of controlling networks running different types of protocols. The main consideration is of connection-oriented networks; those that explicitly setup a route before the transmission of any data (e.g. a telephone call). This will differ from the network discussed here, as once a route is setup the route is not generally subject to change.

Connectionless network routing is subject to a type of *self-noise* - meaning that when congestion is detected, traffic is routed away from the area so it is no longer congested. After some time (determined by the routing algorithm being used), the traffic is moved back again, since the congestion has gone. However, this can simply cause the same congestion again. This situation can repeat *ad infinitum* causing oscillations and therefore instability on the network.

Another problem is when the offered load is simply too great for the network. Whichever link it is placed on will become swamped. This will require proactive reasoning to deal with, as a purely reactive system would probably be sent into a state of instability attempting to deal with it.

Both these effects are inherent to the use of per-hop distributed routing schemes. It would require a holistic, intelligent control regime to avoid such deficiencies.

The use of Ant-Based Routing must, to some extent, assume symmetrical (i.e. Bi-directional) links. This is due to the mapping of a continuous, distributed real-world landscape (in which real Ants can roam around in), to a discretised node/link model, where decisions on direction can only be taken at certain points, as can

pheromone accumulation. In the real world, pheromones can be left as a constant trail, and a trail can split off at any point.

The validity of this assumption is subject to debate, but *is* dependent on the specific network. What is clear is that both directions along a single link must be considered as a coupled pair.

10 Conclusions

The Ant-Based System has been shown to find optimal routing solutions to static conditions, but has been found to show undesirable characteristics when put under the strain of dynamic loading. A solution has been sought that utilises a software agent to adapt to these changing conditions by changing the various parameters that will affect the operation of the ant-system.

By demonstrating both the performance of an ordinary Ant-Based Routing/Load Balancing System, and one subsumed by an agent with a more proactive role, it has been shown that there is potential for an agent to improve dramatically on the results - indicated by network stability, traffic loss, and other communication measures (i.e. jitter etc).

Both of these schemes have been compared with that of a human devised strategy with complete-knowledge of the situation, which would be deemed optimal. However, given the constraints of bounded rationality and realisability an agent could not be expected to obtain this; it might however produce a more subtle scheme that also performs well.

With a strategic agent on each node, this represents a fully distributed network control system. There is still the potential for communication to be investigated, and it is likely to improve further on the results presented here. In line with the model of subsumption architectures, this could be performed at a higher layer, but could also be performed by the agents proposed here. This would represent a hybrid between centralised and distributed control which is often the optimum solution.

Acknowledgments

The Network Simulator, ns, is available and detailed at www.isi.edu/nsnam/ns

References

Dorigo M., Di Caro G., "AntNet: A Mobile Agents Approach to Adaptive Routing", Technical Report IRIDIA 97-12, 1997

Kapetanakis S., Kudenko D., "Improving on the Reinforcement Learning of Coordination in Cooperative Multi-Agent Systems", Proc. AAMAS-II, pp 89-94, 2002

Schoonerwood, R., Holland, O. & Bruten, J., "Ant-Like Agents for Load Balancing in Telecommunications Networks" Proc. 1st International Conference on Autonomous Agents, Marina del Rey, pp209-216, ACM Press, 1997

Vittori K., & Araújo F.R., "Agent-Oriented Routing in Telecommunications Networks", IEICE Transactions on Communications, E84-B(11):3006-3013, November 2001

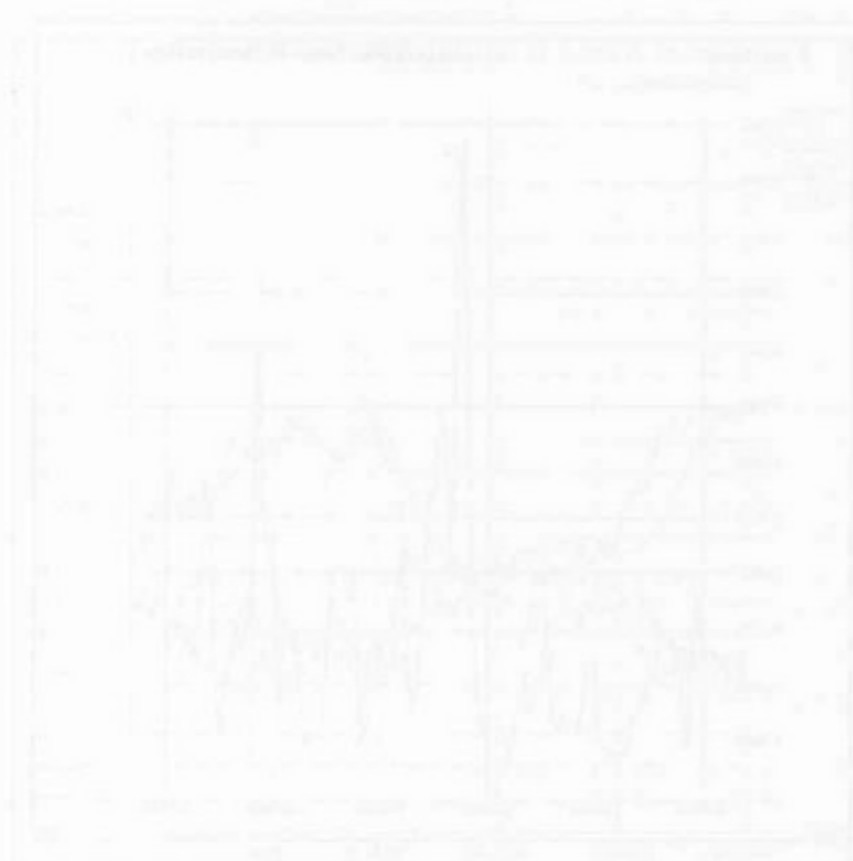


Figure 8: Graph of the network stability. The graph shows two data series over time. The x-axis is labeled 'Time' and ranges from 0 to 100. The y-axis is labeled 'Stability' and ranges from 0 to 1.0. One series, represented by a solid line, starts at 1.0 and remains relatively stable until approximately time 40, where it begins to fluctuate and generally trends downwards towards 0.5. The second series, represented by a dashed line, starts at 0.5 and remains relatively stable until approximately time 40, where it begins to fluctuate and generally trends upwards towards 1.0. Both series show significant fluctuations throughout the time period.

Developing Committee of Explorers for Reinforcement Learning Agents

Funlade T. Sunmola*

*School of Computer Science
University of Birmingham
Birmingham
B15 2TT, England
fts@cs.bham.ac.uk

Jeremy L. Wyatt†

†School of Computer Science
University of Birmingham
Birmingham
B15 2TT, England
jlw@cs.bham.ac.uk

Abstract

In this paper, we describe an architecture and algorithms for reinforcement learning using a population of exploration control units. The control units, collectively known as committee of explorers, allow a reinforcement-learning agent to act while learning by choosing actions to perform through a combination of value predictions of the individual committee members. We discuss reasons for using committee of explorers and use results of an empirical study to provide insights to the performance of the committee approach.

1 Introduction

A reinforcement learning (RL) agent learns to behave by iteratively performing actions in the world and using resulting experiences to decide future actions. The experiences encapsulate amongst other things a reward or reinforcement on the actions taken. The agent learns to perform intelligently by either building a model of its environment and planning using the model or simply by learning the value of each action in every state and performing the action with the highest value. Agents that learn a model and use it to decide how to act are much more data efficient than model-free learners. Model-based RL offer many advantages in domains where real-world actions are expensive and computation time is relatively cheap.

A salient challenge facing a model-based reinforcement learning agent relates to exploration-exploitation trade off (Wyatt, 1997). Typically, for a reinforcement learning agent this is a problem of knowing how to act while learning so as to maximise lifetime performance. The agent needs to decide on actions to perform while maximising current value and simultaneously seeking to benefit appropriately from learning opportunities. This inevitably involves balancing an exploitation of current knowledge with exploration to discover new knowledge that might lead to better performance in the future.

In Markov Decision Processes (MDPs)¹ the optimal Bayesian solution to the trade off problem is well known, but intractable (Martin, 1967; Bellman, 1961). Many approximate ways of dealing with the trade off in RL have been proposed. A new heuristic method was recently introduced (Wyatt, 2000) that brings together ideas from several recent approaches (Kaelbling, 1990; Wiering and Schmidhuber, 1998; Kearns and Singh, 1991). The

heuristics provides a framework for exploration control in reinforcement learning based on optimistic model selection (OMS) from a density over possible models. The heuristics centred on the use of a single control unit and it was shown to outperform existing methods when optimised.

In this paper, we extend the heuristics framework to incorporate a simultaneous use of multiple control units. We refer to the multiple control units as a committee of explorers. In committee of explorers, an ensemble of control mechanisms is employed in the learning process and value prediction of the committee is generated in form of a combination of the predictions of the individual committee members.

Founded on the concept of committee machines (Tresp, 2001), a committee of explorers approach can be useful in many ways. First, committee might exhibit exploration performance unobtainable by an individual committee member on its own. The reason is that the errors of the individual committee members cancel out to some degree when their value predictions are combined. A second reason for using committee of explorers is modularity and exploitation of process structure. When there is structure, as it is naturally the case in some environments, it may be useful to focus individual explorers on particular regions of the information space.

In general terms, committee machines are learning methods that make decisions by combining the results of several individual learners. The learners may be neural networks, probabilistic networks, decision trees, or any other algorithm. The underlying philosophy of the committee machine approach is to fuse knowledge acquired by individuals, basically experts, to arrive at an overall decision that is supposedly superior to that attainable by any one of them acting alone. Most studies on committee

¹Reinforcement Learning addresses unknown MDPs

machines have centred on neural networks (Hansen and Salamon, 1990) and statistical methods (Dalal and Hall, 1983; Lemm, 1999). In this study, our interest in the committee approach relates to the exploration performance of agents when learning from reinforcement.

This paper is structured into five main sections. In section 2 we present a formulation of the exploration control problem. A committee of explorers approach to the problem is presented in Section 3. Empirical study we carried out on the committee approach is presented with a discussion in section 4. Finally, the paper ends in section 5 with conclusions and areas of future work.

2 Exploration Control Problem

An explicitly represented, state based, MDP is simply a 4-tuple (S, A, P, R) . In the tuple, S is a set of N distinct states, A is a set of actions, $p_{ij}^a \in P$ is a transition function that captures the probability of reaching state j after we execute an action a at state i such that $i, j \in S$, and $r(i) \in R$ is a reward function mapping S into real-valued rewards. The state transition coefficients p_{ij}^a obey standard stochastic constraints hence they have the following properties $0 \leq p_{ij}^a \leq 1$ and $\sum_{j=1}^N p_{ij}^a = 1$. The decision problem for an agent in the MDP is to find a policy π that optimise the expected discounted total reward $V = E(\sum_{t=1}^{\infty} \gamma^{t-1} r_t)$, where r_t is the reward received t steps into the future and $\gamma \in [0, 1]$ is a discount factor. For our reinforcement learning agent, the transition function is unknown.

In Bayesian formulation of the problem, we assume that there is a space $\mathcal{P}$ of possible transition functions (parametric models) for the MDP and that there exists a prior probability density over this space. Given that a state $i \in S$ in the process has N possible succeeding states when an action a is taken, then the transition function from that state action pair is a multinomial distribution over the outcomes:

$$\vec{p}_i^a = \{p_{i1}^a, p_{i2}^a, \dots, p_{iN}^a\} \quad (1)$$

The possible transition functions from i, a are the possible $\vec{p}_i^a$.

A convenient, natural conjugate, choice of prior distribution over the $\vec{p}_i^a$ is Dirichlet density:

$$f(\vec{p}_i^a | \vec{m}_i^a) = \frac{\Gamma(\sum_{j=1}^N m_{ij}^a)}{\prod_{j=1}^N \Gamma(m_{ij}^a)} \prod_{j=1}^N (p_{ij}^a)^{m_{ij}^a - 1} \quad (2)$$

and the density is parameterised by the $m_{ij}^a > 0$ for all j .

We observe process transitions. From our choice of likelihood function and prior distribution, it follows directly that the ensuing posterior distribution will be Dirichlet too with parameters $m_{ij}^{a''}$. For a single transition $i \xrightarrow{a} j$ the update rule is $m_{ij}^{a''} = m_{ij}^{a'} + 1$. The density for the multi-state case follows directly from this since the densities over the one step transition functions for all

state action pairs are independent. The density $f(P|M)$ for a possible transition function $P \in \mathcal{P}$ for the MDP is therefore simply the product of the $f(\vec{p}_i^a | \vec{m}_i^a)$ over all i . The density is parameterised by the matrix $M = [m_{ij}^a]$ where $M \in \mathcal{M}$. In a Bayesian framework, we choose a prior matrix M' which specifies our prior density over the space of possible models. The additional information from a sequence of observation is captured in a count matrix F . The posterior density given these observations is therefore simply parameterised by $M'' = M' + F$.

Given the usual squared error loss function, the Bayesian estimator of expected return under the optimal policy is the expectation of the value function $\tilde{V}_i$ in our MDP with unknown transition probabilities:

$$V_i(M) = E[\tilde{V}_i | M] = \int_{\mathcal{P}} V_i(P) f(P|M) dP \quad (3)$$

where $V_i(P)$ is the value of i given the transition function P . We know from the central result of both Bellman and Martin that when this integral is evaluated we transform our problem into one of solving an MDP with unknown transition probabilities, defined on the information space $\mathcal{M} \times \mathcal{S}$:

$$V_i(M) = \max_a \left\{ \sum_j \bar{p}_{ij}^a(M) (r_{ij}^a + \gamma V_j(T_{ij}^a(M))) \right\} \quad (4)$$

in which, for convenience, the transformation on M due to a single observed transition $i \xrightarrow{a} j$ is denoted $(T_{ij}^a(M))$, $\bar{p}_{ij}^a(M)$ is the marginal expectation of the Dirichlet, and r_{ij}^a is the reward associated with the transition $i \xrightarrow{a} j$. This shows how the Bayesian estimate of value elegantly incorporates the value of future information. The optimal solution to our exploration-exploitation trade off problem is thus to act greedily with respect to the Bayes Q-values. The solution to the problem is however intractable because it involves dynamic programming over a tree of information states. Approximate solutions may be found by either simply using the certainty equivalent (CE) estimate constructed by replacing $T_{ij}^a(M)$ with M in (4), or we could approximate the integral by random sampling (Strens, 2000).

The OMS method (Wyatt, 2000) integrates idea from a popular family of approximate approaches to the exploration-exploitation trade off which typically use some instantiation of the heuristic 'be optimistic in the face of uncertainty' (Kaelbling, 1990; Wiering and Schmidhuber, 1998; Meuleau and Bourgin, 1999). OMS integrates the instantiation idea with the Bayesian view of exploration by selecting an optimistic model P_{opt} from $\mathcal{P}$ using probability intervals calculated based on $f(P|M)$. Wyatt suggests two ways of selecting P_{opt} , which were termed simple and full OMS. In simple OMS we are optimistic only about hypothesised transition to an imaginary terminal state. In full OMS we can be optimistic about transitions to other states too.

3 Committee of Explorers

Broadly, our goal is to use a population of model-based explorers to improve the exploration performance of an agent when learning from reinforcement. The population constitutes a parallel interacting exploration processes that jointly produce indicators for appropriate actions to take when acting to accomplish a task in an environment. An outline of the committee approach is presented in Table 1.

-
1. **Begin**
 2. Set $t = 0$, generate an initial population P of E explorers and initialise aggregate exploratory value function ξ
 3. **loop**
 4. get current state.
 5. choose an action based on value function ξ .
 6. do the action and observe transition.
 7. update explorers' beliefs and adjust their exploration value functions.
 8. combine the individual value functions.
 9. set $t = t + 1$.
 10. **end loop**
 11. **end**
-

Table 1: An outline of the committee of explorers' approach

The above outline of the committee approach (table 1) is fairly general with flexibility in the choice of specific implementation. For example, it is possible to use heterogeneous exploration controllers. In what follows, we specify an implementation of the approach by using an homogenous population of optimistic explorers. The explorers are considered homogenous in terms of the similarity in exploration control algorithms they use. Each individual in the population employs a fully optimistic model selection algorithm to guide its exploration of the environment. We present in section 3.1 an overview of full OMS and in section 3.2 we describe the committee approach composed of homogenous optimistic explorers.

3.1 Full OMS

Following an initialisation step (phase 1), the main loop (phases 2-4) of an OMS-oriented solution approach to the exploration-exploitation trade off problem is organised as follows:

Phase 1: initialise process values and model parameters.

Phase 2: select action based on the current value estimate, interact with environment and observe transition.

Phase 3: update parameter matrix in standard way, based on the observed transition.

Phase 4: select an optimistic model P_{opt} and do value backup.

In the full OMS, P_{opt} is selected as follows. Given state action pair i, a we order its successors by the current estimate of the value function, in descending order. We then calculate the lower and upper bounds of the $(1 - \alpha)$ probability interval for each transition. We construct an optimistic transition function by allocating the maximum probability mass to states early in the ordering while keeping all probabilities within their lower and upper bounds. The state action pair i, a is then backed up using the optimistic one step transition function that results. The relevant Bellman equation is:

$$\xi_i^a(M) = \sum_j p_{opt,ij}^a(M)(r_{ij}^a + \gamma \max_{a'} \{\xi_j^{a'}(M)\}) \quad (5)$$

where $p_{opt,ij}^a(M)$ are the transition probabilities according to P_{opt} . The agent selects the action with the highest optimistic value ξ_i^a .

3.2 Committee of Optimistic Explorers

In implementing the committee of explorers approach, we use a set of E OMS-based exploration controllers in phases 3 and 4 (see section 3.1.). Each member of the committee is initialised with a prior parameter matrix and each keeps its own individual transition probability and value function estimates. We extend the organisation by adding a new phase for aggregating the value predictions of the committee members. The implementation is illustrated in Figure 1.

An outline of the committee approach in an asynchronous real time dynamic programming (ARTDP) setting is shown in Table 2.

There are three principal design issues for the committee of explorer approach. Firstly, the approach requires us in line 2 (table 2) to choose prior parameter matrix for each committee member. If we have a knowledge of the transition function P we can use the knowledge to specify our prior for the committee members. Secondly, in line 12 (table 2) we select optimistic models using OMS. We may use OMS in the committee approach in a number of ways. We suggest two principal ways that we term *individualised optimism* and *shared optimism*. For individualised optimism, committee members are optimistic only on the basis of their individual value predictions. For shared optimism, the committee members can be optimistic on the basis of predictions of other committee members as well. Our focus in this paper is on individualised optimism. In the individualised optimism approach, we construct an individual's optimistic transition function for action pair a, i by ordering successors on the basis of the individual's value estimates $\xi_{e,i}^a$. Thirdly, aggregated value estimates

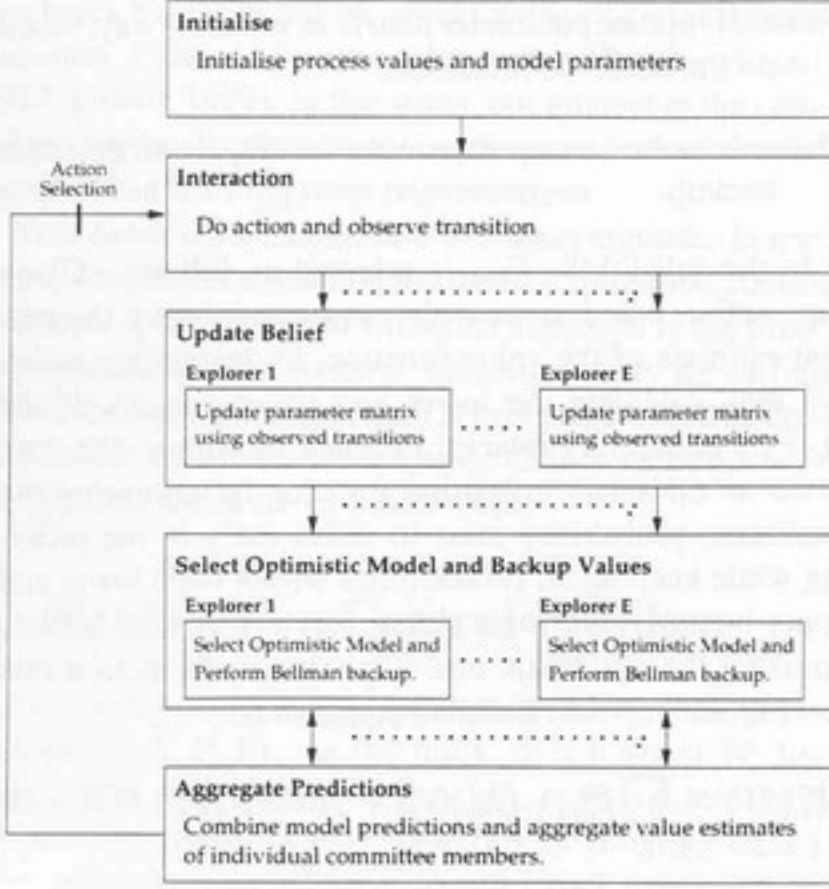


Figure 1: Phases of the committee of explorers' approach.

are updated in line 14 (table 2) through the use of a combination operator C . In this study, we augment the Bellman value function of equation (5) to obtain the aggregated exploration value function.

1. **Procedure CERL**
2. **Initialise** prior parameter matrix M_e , optimistic values $\xi_{e,i}^a$ and aggregate values ξ_i^a $\forall e \in E, \forall i \in S$ and $\forall a \in A$
3. **loop**
4. get current state i .
5. set $a = \arg \max_b \{\xi_i^b\}$, break ties randomly.
6. observe transition $i \xrightarrow{a} j$.
7. update $m_{e,ij}^a = m_{e,ij}^a + 1, \forall e \in E$.
8. **loop until your ARTDP algorithm stops**
9. **for each explorer $e \in E$**
10. choose j'
11. **for each action b**
12. find $p_{e,opt,j'}^b$ using OMS
13. update $\xi_{e,j'}^b$ using Eq. (5)
14. set $\xi_{j'}^b = C(\xi_{1,j'}^b, \dots, \xi_{e,j'}^b, \dots, \xi_{E,j'}^b)$
15. **end for**
16. **end for**
17. **end loop**
18. **end loop**

Table 2: Main loop of the committee of explorers' approach

The augmented Bellman equation is as follows.

$$\xi_i^a = \max\{\xi_{1,i}^a, \dots, \xi_{e,i}^a, \dots, \xi_{E,i}^a\} \quad (6)$$

where $\xi_{e,i}^a$ is the exploration value function of explorer e for state action pair a, i . $\xi_{e,i}^a$ is obtained by invoking equation (5) for each explorer $e \in E$.

4 Empirical Study

We study the committee of explorers using a simple but exploration challenging task to provide insights into its performance. The task and setup of the experiment is described in section 4.1. We present results and its discussion in section 4.2

4.1 Task and setup

The environment is a Meleau's deceptive MDP of Figure 2. The deceptive MDP we used is a two action chain

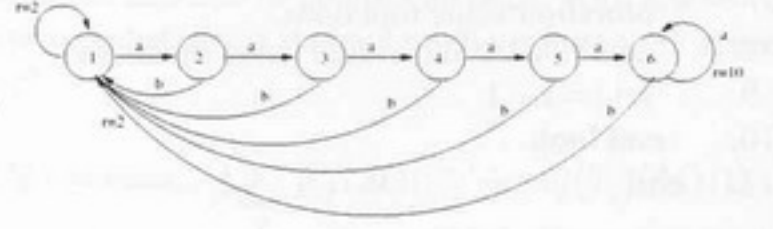


Figure 2: Task: A two action chain model.

model in which reward is a deterministic function of state and γ is set at 0.99. The actual transition functions for the MDP allows actions a and b to have a chance of 80% success when executed. The agent remains in its current state when the action fails. We carried out tests in two stages. First, the OMS approach was applied under the following four prior settings.

Non-informative ('flat') prior under action b + Non-informative ('flat') prior under action a	OMS flat-flat
Informative ('inf') prior under action b + Informative ('inf') prior under action a	OMS inf-inf
Non-informative ('flat') prior under action b + Informative ('inf') prior under action a	OMS flat-inf
Informative ('inf') prior under action b + Non-Informative prior ('flat') under action a	OMS inf-inf

Second, we test the committee approach using two members and set the priors for the committee as follows.

Non-informative ('flat') prior under action b + Informative ('inf') prior under action a	Member 1
Informative ('inf') prior under action b + Non-Informative prior ('flat') under action a	Member 2

In addition we compared results to a baseline algorithm in which actions were based on the optimal policy, i.e. action a was executed at each time step.

Each run of an algorithm was 5000 time steps and results were averaged over 30 runs. All the algorithms were run using Wiering and Schmidhuber's version of prioritised sweeping with a threshold for the priority queue $\epsilon = 0.001$, and the maximum number of backups per step set at 80. For selecting optimistic model, α was set to 0.05.

4.2 Results and discussion

Figures 3-9 show the results of the experiments. In Figure 3, we present the cumulative discounted rewards for the OMS algorithms under the set of four different priors. In addition, a comparison is made with the cumulative discounted rewards obtained for the optimal policy.

We see from Figure 3 that the cumulative discounted rewards vary for each of the five tests. As expected, OMS inf-inf and the optimal policy (Opt.) are close exhibiting a superior performance when compared to others. The OMS inf-inf has a higher cumulative discounted reward at the initial time steps. The cumulative rewards accrued over 500 time step periods are shown in Figure 4.

In Figure 5 we plot the cumulative discounted rewards for the committee and show how it compares with OMS inf-flat and OMS flat-inf. The CERL approach appears to cause the learner to obtain higher accumulated reward during the initial stages of learning. This performance was however not sustained with time. We observe similar trend when we measured the cumulative rewards accrued over 500 time step periods (see Figure 6).

We measured the model error at each time step. Model error was measured simply as the log of squared difference between the maximum likelihood model of the explorer and the actual transition model. The results for action a , action b and the average for both actions a and b are shown in Figures 7, 8 and 9. As expected, the optimal policy (Opt.) did not explore the environment and it is unable to learn the transition model under action b .

Finally, we measured the similarity in the belief of the explorers over the learning period. We measure similarity simply as the log of squared difference between the maximum likelihood models of the explorers. The result of the comparison is illustrated in Figure 10. The result shows that the explorers' posteriors converge during the early stages of the learning process. This result is not surprising if we draw upon the seminal results of Blackwell and Dubins (1962) that convergence will obtain if agents' priors are continuous.

5 Conclusions and Future Work

We have presented a committee approach to the exploration-exploitation trade off problem in reinforcement learning through the use of a population of opti-

mistic explorers. The performance of the committee approach is found to compare favourably to that of the single OMS explorer at the initial stages of the learning period. A central issue is how a knowledge of the problem domain can be used to effectively guide the choice and evolution of committee members. This issue amongst several others provide opportunities for future work. We are studying the use of evolutionary methods especially recombination operations in the committee approach. We are also working on the generalisation of this approach to factored MDPs.

References

- R. E. Bellman. *Adaptive control processes: A guided tour*. Princeton University Press, 1961.
- D. Blackwell and L. Dubins. Merging of opinions with increasing information. *Ann. Math. Statist.*, 38:882–886, 1962.
- S.R. Dalal and W. J. Hall. Approximating priors by mixtures of natural conjugate priors. *Journal of Royal Statistical Society, series B*, 45, 2:278–286, 1983.
- L. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.
- L. P. Kaelbling. *Learning in embedded systems*. Ph.D.Thesis, Dept. of Computer Science, Stanford, 1990.
- M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Proceedings of the Fifteenth International Conference on Machine Learning*, 12:993–1001, 1991.
- J. C. Lemm. Mixtures of Gaussian process priors. pages 8–10, 1999.
- J. Martin. *Bayesian decision problems and Markov chains*. Wiley, New York, 1967.
- N. Meuleau and P. Bourgin. Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, 35:117–154, 1999.
- M. Strens. A Bayesian framework for reinforcement learning. pages 943–950, 2000.
- V. Tresp. *Committee machines*. In *Handbook on neural Network Signal Processing*, CRC Press, 2001.
- W. Wiering and J. Schmidhuber. Efficient model-based exploration. In *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behaviour*, 1998.

J. L. Wyatt. *Exploration and inference in learning from reinforcement*. Ph.D.Thesis, University of Edingurgh, Dept. of Artificial Intelligence, Edingurgh University, 1997.

J. L. Wyatt. Exploration control in reinforcement learning using optimistic model selection. *International Conference on Machine Learning (ICML)*, 2000.

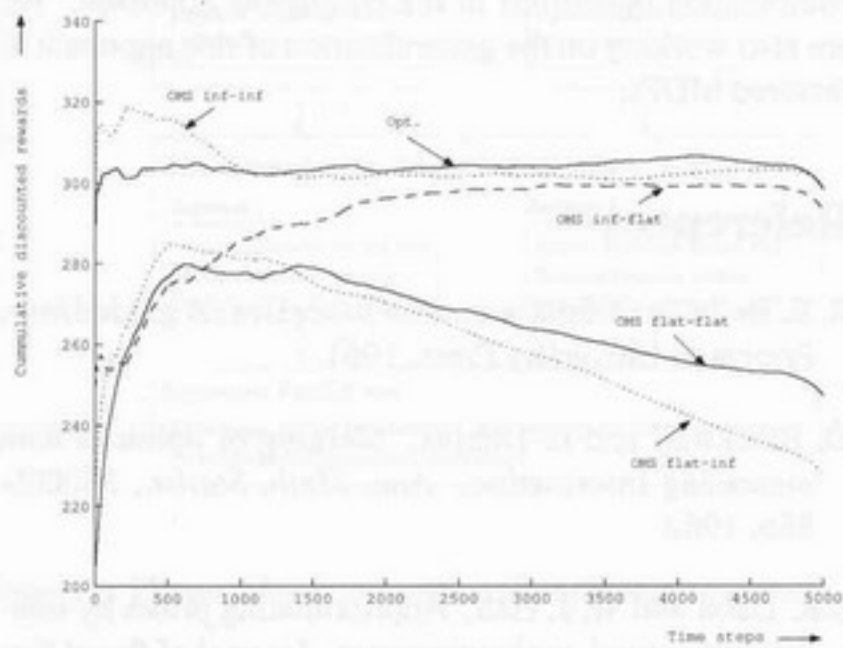


Figure 3: Cumulative discounted rewards (normalised) for OMS and Opt.

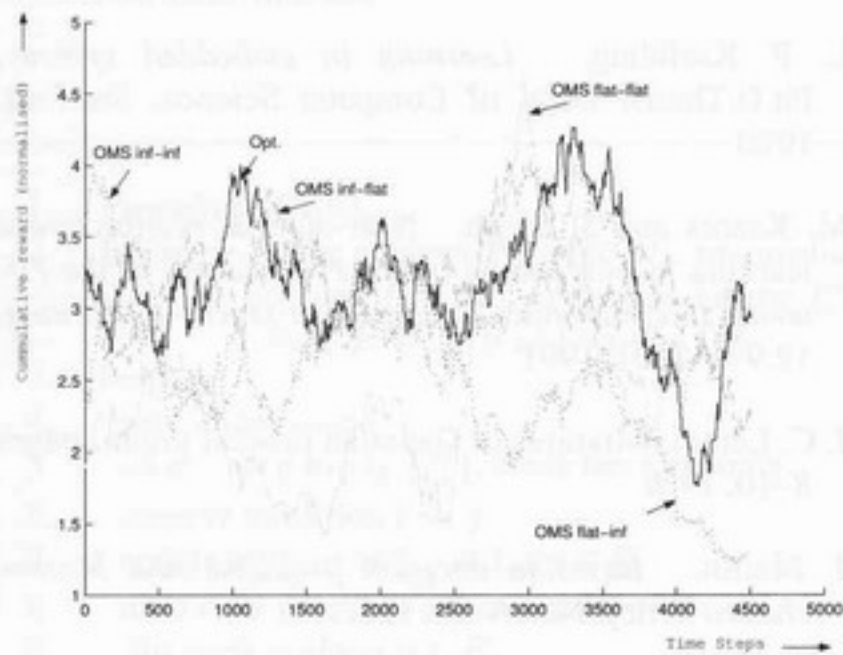


Figure 4: Cumulative reward (normalised) for OMS and Opt.

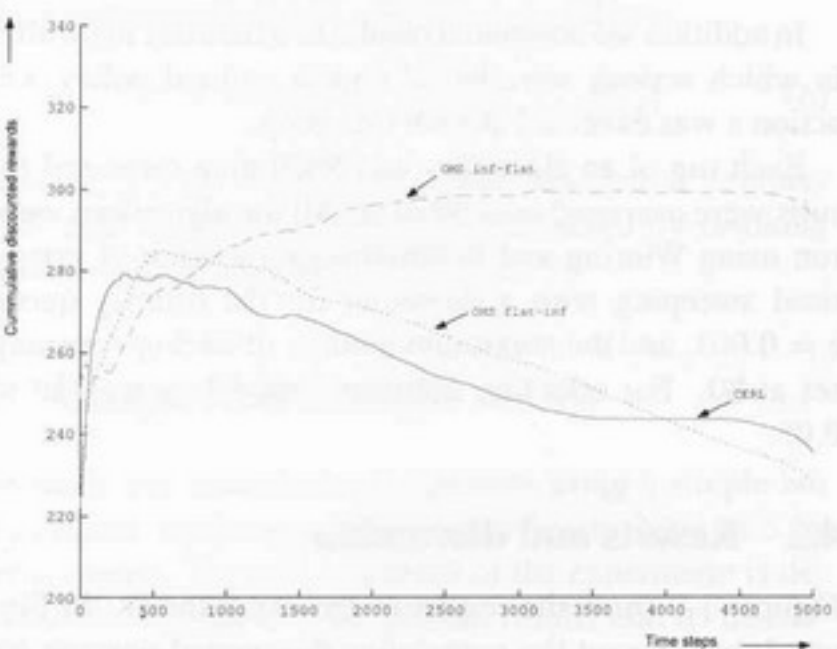


Figure 5: Cumulative discounted rewards (normalised) for CERL and OMS.

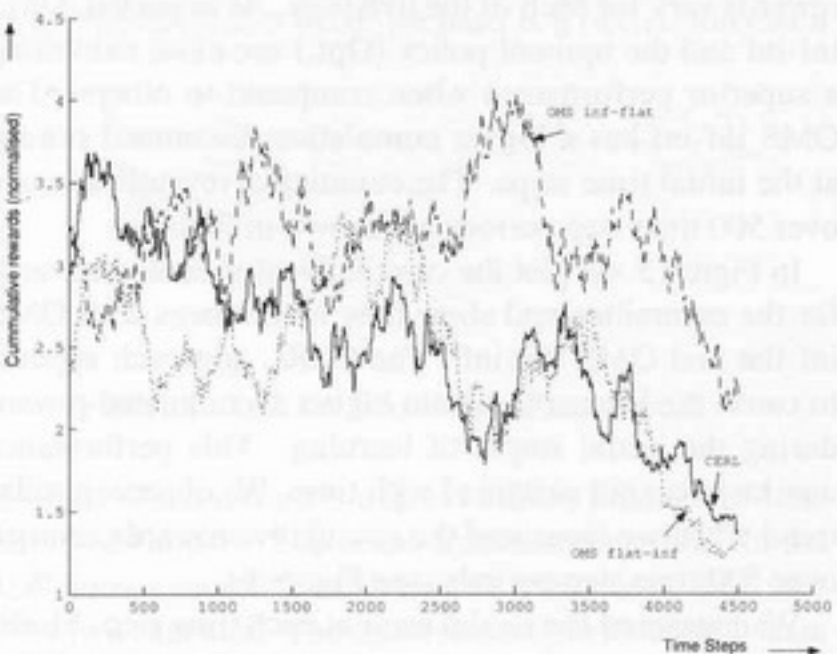


Figure 6: Cumulative reward (normalised) for CERL and OMS.

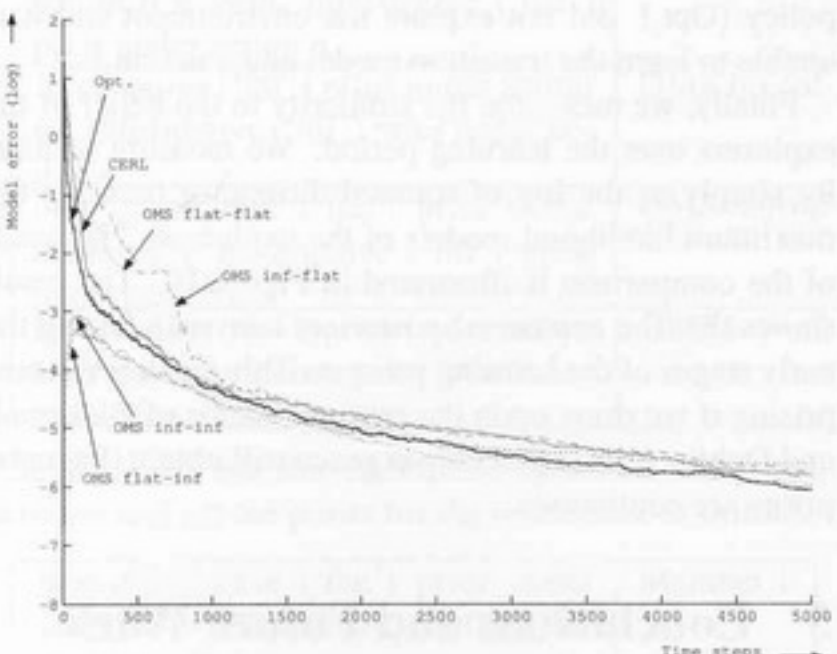


Figure 7: Comparisons of model error under action a.

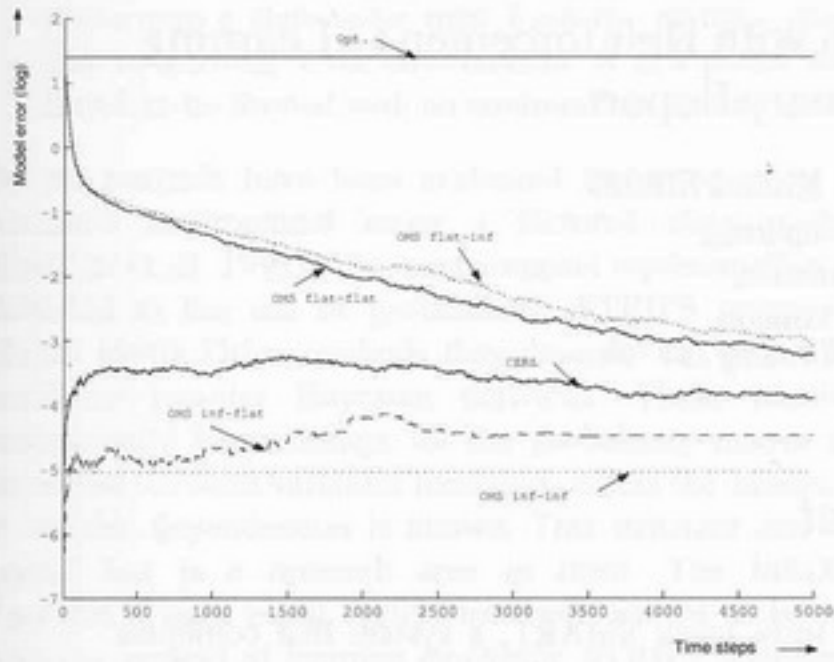


Figure 8: Comparisons of model error under action b.

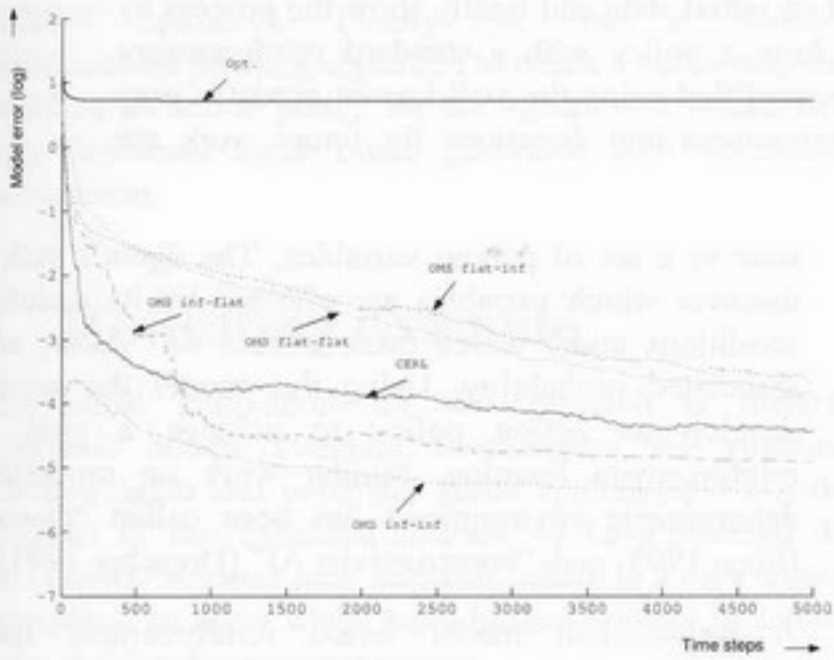


Figure 9: Comparisons of model error under actions a and b.

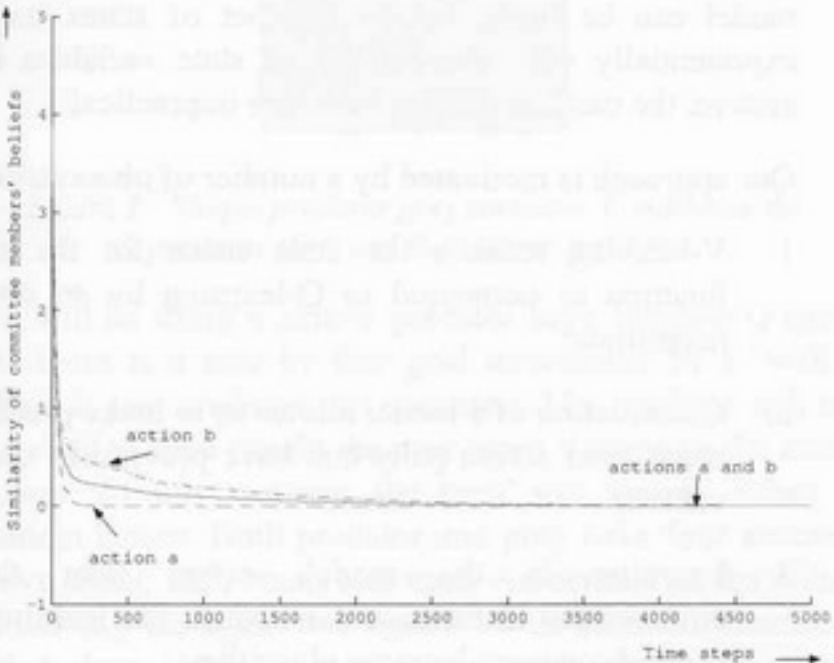


Figure 10: Similarity of committee member's beliefs.

SMART (Stochastic Model Acquisition with Reinforcement) Learning Agents: A Preliminary Report

Christopher Child* and Kostas Stathis

Department of Computing,

School of Informatics,

City University, London

{c.child,k.stathis}@city.ac.uk

Abstract

We present a framework for building agents that learn using SMART, a system that combines stochastic model acquisition with reinforcement learning to enable an agent to model its environment through experience and subsequently form action selection policies using the acquired model. We extend an existing algorithm for automatic creation of stochastic strips operators (Oates et. al 1995) as a preliminary method of environment modelling. We then define the process of generation of future states using these operators and an initial state and finally show the process by which the agent can use the generated states to form a policy with a standard reinforcement learning algorithm. The potential of SMART is exemplified using the well-known predator prey scenario. Results of applying SMART to this environment and directions for future work are discussed.

1 Introduction

Reinforcement learning has been shown to be a powerful tool in the automatic formation of action policies for agents (Kaelbling et. al. 1996). There are two main approaches: Value learning (V-Learning) which assigns a value to each state in the system and Q-Learning, which assigns a value to each state action pair (Sutton and Barto 1998). V-learning assigns state values by propagating back rewards received in future states after taking an *action* according to a *policy*. V-learning is limited in application because it requires a model of the world in order to predict which state will occur after each action. Q-Learning is more widely applicable because it assigns rewards to a state action pair. The agent is therefore not required to predict the future state and does not require a model. Often it is impossible for the designer of an agent to provide a model of the environment. Even if the environment has been designed in software, the transition from state to state may be impossible for the designer to predict due to factors such as the action of other agents.

In this paper we investigate "stochastic model acquisition". We define this to be any system which enables an agent to acquire a model of its environment from the environment. To be more precise we are modelling the agent's perception of the environment because this is all the information which it has access to. Initially the agent has knowledge of the actions it can perform, but not the effects, and has a perceptive function that maps a world

state to a set of percep variables. The agent's task is to discover which variables are effected by its actions, the conditions under which these effects will occur, and an associated probability. Using this model the agent can develop an action policy to achieve a goal using reinforcement learning. Similar work on modelling in deterministic environments has been called "discovery" (Shen 1993) and "constructivist AI" (Drescher 1991).

A well-studied model based reinforcement learning architecture is Dyna-Q (Sutton & Barto 1998). The algorithm suffers from the disadvantage that it relies on statistical measures of entire state transitions rather than state variable transitions. These require multiple visits to each state in a stochastic environment before an accurate model can be built. As the number of states increases exponentially with the number of state variables in the system, the method quickly becomes impractical.

Our approach is motivated by a number of observations:

1. V-learning reduces the state space for the reward function as compared to Q-learning by an order of magnitude;
2. Construction of a model allows us to make predictions about state action pairs that have previously not been visited;
3. Learning in the model, rather than through environment interaction can reduce the learning time of reinforcement learning algorithms.

* Corresponding author.

- Changes in the agents goals (reward function) result in re-learning a state-value map from the model, rather than re-learning in the environment. A new policy can therefore be formed with no environment interaction.

Several methods have been evaluated for representing a stochastic environment using a factored state model (Boutilier et. al. 1999). The most compact representation is identified as the use of probabilistic STRIPS operators (Hanks 1990). Other methods they describe are generally based on two-tier Bayesian networks. These require exponentially large storage for the probability matrix as the number of state variables increases, unless the structure of variable dependencies is known. This structure can be learned but is a research area in itself. The MSDD algorithm (Oates et. al. 1995) has been shown to be an effective method of learning stochastic STRIPS operators and will therefore be used in this work.

Having generated a set of stochastic STRIPS operators, the next stage is to generate expected world states using these STRIPS operators. Finally we use a standard reinforcement learning algorithm to create a value map and therefore an action policy for the agent. This results in a fully functional agent mind, generated with no human intervention.

2 Motivating Example

The broad motivation for this research is towards automatic action selection mechanisms for robotics, software agent and computer game applications. For the purposes of this extended abstract we have selected the well known predator prey scenario, which is a very simple example of an agent which has a limited number of actions and a restricted perceptual system.

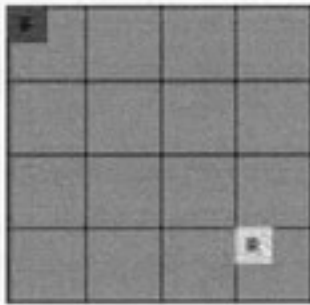


Figure 1 : Simple predator prey scenario. *F* indicates the predator (fox) and *R* the prey (rabbit).

We will be using a simple predator prey scenario (Figure 1). There is a four by four grid surrounded by a "wall". There is *one* predator and *one* prey. The predator will be assumed to have caught the prey when it lands on the same square. In this instance the prey will simply select a random action. Both predator and prey have four actions: move north, east, south and west. An action has the effect of moving the agent one square in the selected direction, unless there is a wall, in which instance there is no effect. The predator and prey move in alternate turns. The agent's percept gives the contents of the four squares around it and

the square it is on. Each square can be in one of three states: empty, wall or agent. For example a predator agent which is situated in the north west corner of the grid with a

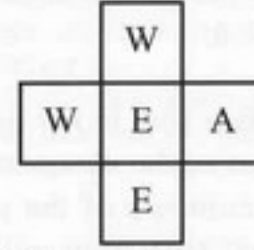


Figure 2: The predator agent percept

prey to the east would have the percept $\langle \text{WALL}, \text{AGENT}, \text{EMPTY}, \text{WALL}, \text{EMPTY} \rangle$ corresponding to the squares to the north, east, south, west and under respectively, as shown in figure 2.

3 Framework

The stages of the agent system are as follows: stochastic model acquisition, state generation and policy generation.

3.1 Stochastic Model Acquisition

3.1.1 Stochastic STRIPS operators

The STRIPS planning operator representation has, for each action, a set of preconditions, an "add" list, and a "delete" list (Fikes and Nilsson 1971). The STRIPS planner was designed for deterministic environments, with the assumption that actions taken in a state matching the operator's preconditions would consistently result in the state changes indicated by the operators add and delete lists. In a non-deterministic environment a less restrictive view is taken, allowing actions to be attempted in any state. The effects of the action then depend on the state in which it was taken and are influenced by some properties external to the agents perception which appear random from the agent's perspective.

We follow the format for stochastic STRIPS operators used by Oates & Cohen (Oates and Cohen 1996). A stochastic STRIPS operator takes the form:

$$O = \langle a, c, e, p \rangle$$

where a specifies an action, c specifies a context, e the effects and p the probability. If the agent is in a state matching the context c , and takes the action a , then the agent will observe a state matching the effects e with probability p .

Contexts and effects of operators are specified as a list of tokens representing the percept and the action of the agent in the order described in section 2, with the addition of a wildcard symbol (*) denoting irrelevance, which matches any token.

As an example of the use of wildcards, consider the following operator:

<MOVE EAST, (* WALL * * *), (* WALL * * *)> Prob:1.0

In this example: a is MOVE EAST, c is (* WALL * * *), e is (* WALL * * *) and p is 1.0.

This operator specifies that if the agent chooses to move north and the contents of the square to the north is detected as WALL then the contents of the square to the north of the agent on the next time step will still be a wall, with probability 1.0. The wildcards specify that this condition is irrelevant of anything else that the agent observes¹. The percep order is as specified in section 2.

3.2 Learning STRIPS operators

We have chosen to use Multi-Stream Dependency Detection (MSDD) (Oats et. al. 1995) to learn the STRIPS operators in this context. Although inductive logic programming (ILP) (Muggleton 2000) is powerful in its domain area of learning predicate logic rules, it is still an open research area to generate reliable stochastic logic rules with the system. MACCENT is another inductive logic system specifically designed for learning stochastic rules (Dehaspe 1997), but is not well suited to large rule sets. MSDD has been chosen for its suitability to deal with the domain area, and has previously been shown to be able to generate stochastic STRIPS operators from data (Oates and Cohen 1996).

3.2.1 MSDD

Formal statements of both the MSDD algorithm and its node expansion routine are given in the algorithms below. MSDD is a batch algorithm and uses H , the precep data observed by the agent in the preconditions-effects format shown in section 3.1.1. The function f evaluates the best node to expand next and typically counts the co-occurrence of the node's preconditions and effects. This requires a complete pass over the data set H .

MSDD (H , f , $maxnodes$)

1. $expanded = 0$
2. $nodes = \text{ROOT-NODE}()$
3. while NOT-EMPTY($nodes$) and $expanded < maxnodes$ do
 - a. remove from $nodes$ the node n maximising $f(H, n)$
 - b. EXPAND(n), adding its children to $nodes$
 - c. increment $expanded$ by the number of children generated in (b)

EXPAND (n)

1. for i from m down to 1 do
 - a. if $n.preconditions[i] \neq '*'$ then return children
 - b. for $t \in T_i$ do
 - i. $child = \text{COPY-NODE}(n)$
 - ii. $child.preconditions[i] = t$
 - iii. push $child$ onto children
2. repeat (1) for the effects of n
3. return children

¹ Note that the agent's percep does not include the agent itself.

This algorithm does not specify which children should be generated before others, but does ensure that each dependency is explored only once, and facilitates pruning of the search. For example, all descendants of the node

<*, (WALL * * * *), (WALL * * * *)>

can be pruned because there is no need to explore rules with a wildcard in the action position, and all descendants of this node will be expanded from the rightmost wildcard resulting in children with no action.

We have made two changes to standard MSDD. The first is in EXPAND (b.iii). We check that the generated child matches at least one observation in the database before adding it to children. For example, MSDD can generate the rule <*, (WALL WALL WALL * *), (* * * * *)>, but in our environment the agent can only observe a maximum of two walls (when it is positioned in the corner of the map). A check against the data set will reveal that the generated rule has no matches and can be eliminated from the node list, along with it's children as a consequence. This prevents the generation of a large number of incorrect rules, which would have been eliminated in the "filter" stage of MSDD.

The second change is that the effect part of the rule is only allowed to have one non-wildcard element. We have made this change because a very large number of rules are generated by standard MSDD. Combining individual effect fluents can generate complete successor states. This has the disadvantage that illegal states can be created, such as <WALL WALL WALL * *>. These can, however, be eliminated using constraints (section 3.3).

3.2.2 Filter

The second stage of MSDD is the "filter" process, which removes specific rules already covered by more general ones. For example, < $C_1 C_2 * * *$ > is a more specific version of < $C_1 * * * *$ >. If the condition C_2 has no significant effect on the probability of the rule then it is unnecessary. For example, MSDD could generate two rules as follows:

<MOVE NORTH, (WALL WALL * * *), (* WALL * * *)> Prob: 1.0
 <MOVE NORTH, (* WALL * * *), (* WALL * * *)> Prob: 1.0

Both of these rules tell us that, if the agent moves north and there was a wall to the east, it will observe a wall to the east on the next move. The extra information that there was a wall to the north does not affect the agent's subsequent observation. More general operators are preferred because a reduced number of rules can cover the same information. For operators which do not have a probability of 1 we are testing, for an operator $O = \langle a, c, e, p \rangle$, whether $\text{Prob}(e | c_1, c_2, a)$ and $\text{Prob}(e | c_1, a)$ are significantly different. If not, the general operator is kept the specific one discarded.

```

FILTER (D, H, g)
1. sort D in non-increasing order of generality
2. S = {}
3. while NOT_EMPTY(D)
  a. s = POP(D)
  b. PUSH (s, S)
  c. for d ∈ D do
    if SUBSUMES(s, d) and G(s, d, H) < g then
      remove d from D
4. Return S

```

Where *D* is the set of dependency operators generated by MSDD. *H* is the history of observations made by the agent. SUBSUMES(*d*₁, *d*₂) is a Boolean function defined to return true if dependency operator *d*₁ is a generalisation of *d*₂. *G*(*d*₁, *d*₂, *H*) returns the *G* statistic to determine whether the conditional probability of *d*₁'s effects given its conditions is significantly different from *d*₂'s effects given its conditions. The parameter *g* is used as a threshold, which the *G* statistic must exceed before *d*₁ and *d*₂ are considered different².

For example, dependency *d*₁ below returns a very low *G* statistic when compared with *d*₂:

*d*₁: <MOVE NORTH, (WALL * EMPTY * AGENT), (* * * * AGENT)>
*d*₂: <MOVE NORTH, (WALL * * * AGENT), (* * * * AGENT)>

The additional condition that there is an EMPTY square to the south does not effect the probability of an agent being present on the square it inhabits after a move north (which will be the same square because there is a wall to the north). The predator agent already has the information that the prey is in the same square and that there is a WALL to the north, so the square to the south will either be the prey or be empty irrelevant of the empty square to the south. For an explanation of calculation of the *G* statistic see Oates et. al. (1995).

3.2.3 Rule Complements

When running MSDD on data collected from the predator prey scenario, we observed that the “filter” process occasionally filters rules and not their complements. For example the filter process could filter *d*₂ below, but leave *d*₁:

*d*₁: <MOVE NORTH, (WALL WALL * * AGENT), (* * * * AGENT)>
*d*₂: <MOVE NORTH, (WALL WALL * * AGENT), (* * * * EMPTY)>

This would cause a problem in the state generation (section 3.3), because the rule generates only states with agents present. To avoid this we have added a further step to the algorithm in order to search through all generated rules checking that their complement rules are present. This process is less problematic in our system because rules only have effects in one fluent.

```

AddRuleComplements (D, H)
for d ∈ D do
  f = d.effect
  for fValue ∈ possibleValues(f)
    if fValue not equal to f.value
      newRule = copy of d with d.effect set to
                  fValue
      if newRule is not present in D
        if newRule has match in H
          add newRule to D

```

The above algorithm goes through all rules in the learned dependencies, *D*, checking that all possible values of its effect fluent are either present in *D* already or do not match any observations in the history *H*. If a missing rule is found it is added to *D*.

3.3 Generating States from learned rules

The state generator function of the SMART learning process generates all possible next states, with associated probabilities, for an action that the agent could take in a given state. These states are generated using the rules learned by MSDD from the history of observations. This state generation process is necessary for the agent to generate a state value map using reinforcement learning (section 3.3.5).

Our modified implementation of the MSDD algorithm generates a set of rules with only one fluent in the effects part in order to reduce substantially the number of rules that must be evaluated. When we match these against some initial conditions, such as:

<MOVE EAST, (EMPTY WALL WALL EMPTY EMPTY)>

The subset of the generated rules matching these conditions is shown in Table 1.

Table 1: Subset of the generated dependency rules matching the condition <MOVE EAST, (EMPTY WALL WALL EMPTY EMPTY)>. E = EMPTY, W = WALL and A = AGENT. The table shows the rule set after removing general rules (section 3.3.1).

Conditions						Effects					Pr
Action	N	E	S	W	U	N	E	S	W	U	
EAST	*	W	*	*	*	*	W	*	*	*	1.0
EAST	*	*	W	*	*	*	*	W	*	*	1.0
EAST	E	W	*	E	*	*	*	*	A	*	0.07
EAST	E	W	*	E	*	*	*	*	E	*	0.93
EAST	E	W	*	*	E	A	*	*	*	*	0.03
EAST	E	W	*	*	E	E	*	*	*	*	0.97
EAST	E	*	W	E	*	A	*	*	*	*	0.06
EAST	E	*	W	E	*	E	*	*	*	*	0.94
EAST	E	W	*	E	E	*	*	*	*	E	1.0

² A value of 3.84 for *g* tests for statistical significance at the 5% level.

Notice that the rule set correctly tells us that moving east when there is a wall to the east results in us still observing a wall to the east. Also moving east with a wall to the south results in a wall to the south irrespective of any other contextual information. The rule set captures all the information we require for the given state and action and has the advantage that rules are also applicable to many other states.

The generated rule set for a given situation can provide several rules for each output fluent. To generate individual states from these rules we have to decide which rules are more relevant in the given situation. The following sections describe the process by which we choose rules, and why the decision was made to process the rules in this way. The stages of the state generator are:

1. Remove general rules covered by specific.
2. Remove rules with less specific effects.
3. Generate possible states and probabilities.
4. Remove impossible states using constraints and normalise the state probabilities.

The output of the state generator is a table of states and associated probabilities.

3.3.1 Remove General Rules Covered by Specific

The subset of the generated rules that apply to a specific state action pair may contain rules with different conditions applicable to the same effect fluent. If the environment has not been well explored by the agent, a general rule may be more reliable because a specific rule will not have been encountered as often in the history of observations made by the agent. If, however, the agent has been allowed to explore the environment extensively, as was the case in our experiments, we can assume that a specific rule is likely to contain more relevant information than a general one. An expansion to the system would be to choose less specific rules depending on the statistical confidence measure of each rule. This is a subject for further research.

General rules are removed by searching through the rest of the dependencies for other nodes with an effect for the same fluent. If another rule is found which is more specific the general rule is removed. Rules with equal specificity are not removed.

```

RemoveRulesCoveredBySpecific(rules)
sort rules in non-increasing order of generality
for rule ∈ rules
  for testRule ∈ rules after rule
    if effectFluent of rule is same
      position as effectFluent of testRule
      if rule.wildcards > testRule.wildcards
        remove rule from rules

```

Table 2 gives an example of rules that were removed to give the rule set in Table 1. The rules were removed because a more specific rule was available for the same effect fluent.

Table 2: The first two rules in the above table were removed because the specific rule (in bold) covered them.

Conditions						Effects					Pr
Action	N	E	S	W	U	N	E	S	W	U	
EAST	*	*	*	E	*	*	*	*	*	A	0.04
EAST	*	*	*	E	*	*	*	*	*	E	0.96
EAST	E	W	*	E	E	*	*	*	*	E	1.0

3.3.2 Remove Rules With Less Specific Effects

MSDD generates rules with equally specific conditions but less specific results. Rules which are more specific in the effects part are providing us with the information that the given effect cannot arise in this context. We can therefore eliminate the less specific effects. The rules shown in italics in Table 1 are removed by this part of the state generation process, because other rules exist with equal specificity of effect.

```

removeLessSpecificOutcome(rules)
for rule ∈ rules do
  effectP = position of effect in testRule
  NumEffects = countRules(rule.conditions,
                          effectP)
  for other ∈ matchingNodes do
    if other has same effectP and same
      conditions
      otherNumEffects
      = countRules(other.conditions,
                    effectP)
    if otherNumEffects >= NumEffects
      removeRules(other.conditions, effectP)
    else if otherNumEffects > 0
      removeRules(rule.conditions, effectP)

```

If we do not remove rules with less specific effect in this way the state generator can produce states which could not occur in the real environment. Consider for example the rules in Table 3.

Table 3: Removing rules with less specific outcomes. Rules in Italics are removed because their outcome is less specific than the other rules with the same effect fluent.

Conditions						Effects					Pr
Action	N	E	S	W	U	N	E	S	W	U	
NOR	E	E	W	*	E	E	*	*	*	*	0.94
NOR	E	E	W	*	E	A	*	*	*	*	0.06
<i>NOR</i>	<i>E</i>	<i>E</i>	<i>*</i>	<i>E</i>	<i>E</i>	<i>E</i>	<i>*</i>	<i>*</i>	<i>*</i>	<i>*</i>	<i>0.07</i>
<i>NOR</i>	<i>E</i>	<i>E</i>	<i>*</i>	<i>E</i>	<i>E</i>	<i>A</i>	<i>*</i>	<i>*</i>	<i>*</i>	<i>*</i>	<i>0.05</i>
<i>NOR</i>	<i>E</i>	<i>E</i>	<i>*</i>	<i>E</i>	<i>E</i>	<i>W</i>	<i>*</i>	<i>*</i>	<i>*</i>	<i>*</i>	<i>0.88</i>

The rules removed by this process, shown in italics, state that a move north could result in a wall to the north. In fact there is a wall to the south in the original state. A move north could, therefore, only result in an empty square or an agent, as there is no situation in which the agent can move

from one wall to another opposite in a single move in the environment.

3.3.3 Generate Possible States

The possible states are generated by:

1. Creating a new state from each combination of effect fluent values in the remaining rules.
2. Multiply the probability of each effect rule to generate the probability of each state.

The states generated from the rules in Table 1 are as follows:

Table 4: Possible after states for the given state and action generated from the rules in Table 1.

NORTH	EAST	SOUTH	WEST	UNDER	Pr:
EMPTY	WALL	WALL	EMPTY	EMPTY	~0.91
EMPTY	WALL	WALL	AGENT	EMPTY	~0.07
AGENT	WALL	WALL	EMPTY	EMPTY	~0.02
AGENT	WALL	WALL	AGENT	EMPTY	~0.001

There were two rules for the “north” fluent with results EMPTY and AGENT, and two rules for the “west” fluent with results EMPTY and AGENT. The other rules had one result each resulting in a total of: $2 * 1 * 1 * 2 * 1 = 4$ possible states. The probabilities are found by multiplying together the probabilities of the rules which resulted in these fluents.

3.3.4 Remove Impossible States with Constraints

Some of the states generated could not occur in the domain area. For example in the predator prey scenario the operators may generate a percept with two agents, when there is only one agent in the world. We would ultimately like our agent to generate its own constraints that tell it which world states are impossible. A rule such as IMPOSSIBLE (* * AGENT AGENT *) would allow elimination of the impossible world states generated. For the purposes of this paper, we will be using a user-defined set of constraints. If we do not use these constraints the erroneous generated states can propagate to create world states where there are five prey agents, or walls surround the predator agent, and the model becomes meaningless as it is too far detached from the real world states. Currently our system simply removes impossible states by checking that each generated state does not contain more than one agent, or walls opposite each other. This method breaks the principle of an autonomous learning agent that learns a model of its environment without human intervention. A constraint generator will therefore be the subject of future research.

After elimination of illegal states the probabilities of remaining states are normalised by dividing the probability of each state by the total probability of all generated states

to give the final states. The state in italics in Table 4 is removed by this process because it contains two agents.

3.3.5 Removing Unused Rules

In the present system the state generator has to continually search through the dependency list and remove general rules which are covered by specific ones. We could reduce both the size of the rule set, and the time taken to generate states by removing general rules which are never used to generate states from the rule set as follows:

Starting from the most general rule, in order of increasing specificity, we check to see if there is a set of more specific rules entirely covering the possible observed values of each wildcard fluent. If, for example, the rule set below exists:

- d_1 : <MOVE NORTH, (*** AGENT), (*** AGENT) >
 d_2 : <MOVE NORTH, (*** AGENT), (*** EMPTY) >

The following more specific rule set would cover the above rules and cause them to be unused:

- d_1 : <MOVE NORTH, (WALL *** AGENT), (*** AGENT) >
 d_2 : <MOVE NORTH, (WALL *** AGENT), (*** EMPTY) >
 d_3 : <MOVE NORTH, (EMPTY *** AGENT), (*** AGENT) >
 d_4 : <MOVE NORTH, (EMPTY *** AGENT), (*** EMPTY) >

Notice that the more specific rule set covers WALL and EMPTY, but does not have an AGENT value. In our predator prey scenario it is not possible to have two agents in one percep. These rules therefore cover all fluent values with the remaining value finding no matches in the database. This feature has not been implemented in the current system.

3.4 Generating Policies

We use standard reinforcement learning techniques to generate a policy for the agent. The acquired model allows us to use value iteration, which is a simple and efficient method of generating a value map for the agent (Sutton & Barto 1998). The update equation for value iteration is given by:

$$V_{k+1}(s) = \max_a \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V_k(s')]$$

The value of state s on pass $k + 1$ of the value iteration is calculated by taking the maximum valued action. The value of the action is equal to the sum for s' of the probability of action leading from state s to s' multiplied by the reward plus the discounted value of state s' on pass k . In order to generate a value map, we start with a state generated by a random initial position of the predator prey scenario and add this to the value map. A single entry in the value map is as follows:

State, Value, Reward

State is a percep (e.g. <WALL, WALL, EMPTY, EMPTY, EMPTY>). *Reward* is set to a positive value if there is an AGENT in the last position of the percep (corresponding to the predator catching the prey) and zero or a negative value otherwise. *Value*, for each state is then generated by repeated application of the following algorithm, with the next state to refine picked at random from those generated in the `getStateValue` step of the algorithm below.

```

refineValue(state, actions, valueMap)
maxUtility = 0
for action in actions do
    nextStatesAndProb = generateStates(state,
                                     action)

    actionUtility = 0
    for afterState in nextStatesAndProb
        getStateValue(afterState)
        actionUtility += afterState.probability *
            (afterState.value *  $\gamma$  + afterState.reward)
    if (actionUtility > maxUtility)
        maxUtility = actionUtility
setStateValue(valueMap, state, maxUtility)

```

4 Empirical Results

Performance of the policy generated by SMART learning was assessed by tested against a standard Dyna-Q algorithm (Sutton and Barto 1998). Dyna-Q is a model based learning system, which uses a state map generated by recording the frequency with which each action in each state leads to the next state. Dyna-Q has previously been applied to the predator prey scenario presented in this paper (Varsy 2002). We therefore repeat the test conditions used for this work using the SMART framework, for the purposes of comparison.

4.1 Test Conditions

An experimental run consisted of a sequence of trials or episodes that end after the prey has been captured. In the first trial of each run, the predator and prey are given the start position as indicated by Figure 1. The result of an experiment was the number of steps required for the predator to catch the prey averaged over 30 runs. Each run was 2000 steps, giving 1000 moves for both the predator and prey agents.

In our test environment we allowed the agent to gather a perceptual history (*H*) from 10,000 iterations of the environment. Our extended MSDD algorithm was subsequently run on the observation history to generate stochastic STRIPS operators. The state generator and reinforcement learning algorithm were then applied for 2,000 iterations to generate a value map. *Reward* was set to 1.0 for a state where the prey agent is “under” the predator, and -0.1 otherwise. The discount factor (γ) was 0.9. The environment was then run according to the test conditions. On each environment update the predator agent picked the highest utility action. This was achieved using a similar process to the selection of highest utility action in the “refineValue” algorithm (section 3.4).

Table 5: Average life-span of prey under test conditions

Action Method	Prey life-span
Random moves	16.37
Dyna-Q	7.04
SMART learning	9.72

Both predator and prey taking random moves resulted in the predator being on the same square as the prey every 16.37 moves. This result is expected. There are 16 squares in the grid and the predator will randomly occupy the same square roughly once every 16 moves. The extra fraction is probably due to the start positions being at opposite ends of the grid.

Using Dyna-Q the predator caught the prey in an average of 7.04 moves. Dyna-Q was required to learn a policy during the run, and, as the predator and prey take alternate turns, half the moves are taken by the prey and although random, are likely to result in the prey evading the predator.

SMART learning resulted in the predator catching the prey in an average of 9.72 moves. Initially this might appear to be a low score when compared to Dyna-Q, because the agent enters the world with a fully formed policy. An ideal state action policy should result in the predator being able to move onto the prey’s square every time it moves once it has caught it the initial time. This would result in a prey life-span of approximately 2 moves. The SMART learner, however, is a state-value method, and does not have access to the immediate effect of its action as it perceives just

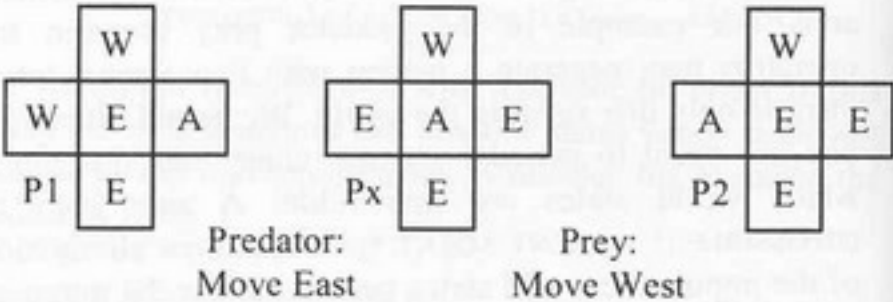


Figure 3: Order of perceptions in predator-prey scenario

before it moves and not just after. The prey takes a move between these two perceptions, so the predator is only able to learn by an approximate method.

Figure 3 shows order of perceptions from the perspective of the predator. The predator initially has the perception P1. The move east action takes the predator onto the agent’s square. The predator, however, does not observe the percep Px, because it is the prey’s turn to move. When the predator observes again at P2 the prey has moved out of the capture square. Methods such as Dyna-Q suffer less from this problem, because reward is associated with the state, action pair (e.g. <P1, Move East>) and the predator

is therefore able to associate rewards with the actions which produced them.

Despite this problem, SMART learning enabled the agent to form a good policy in the predator prey environment and the failure to form an optimum policy owes more to the experimental conditions than a problem with the learning technique itself.

5 Conclusions and Future Work

We have presented a framework for stochastic model acquisition, which promises to be a powerful extension to the reinforcement learning paradigm. We have shown how to overcome some of the problems encountered when attempting to generate next states from automatically acquired STRIPS operators and demonstrated that action policies can be developed using a model represented by these operators.

The ability to learn a model of the environment through experience automates an otherwise difficult or impossible process for an agent designer. Future experiments aim to demonstrate that the agent can keep important experience when its goals are changed and the designer is able to change the reward structure and learn a new policy without the need for expensive interaction with the environment.

The use of a rule learning method to acquire a model provides an accessible format for a human designer. If the system is not performing as the designer wishes the rules can be investigated and anomalies spotted more easily than with a black box learning system such as a neural network.

Subjects of further research will include:

- Testing the system in an environment more suited to model based learning, in which the results of action are immediately perceivable by the agent.
- Evaluation of rule learning methods in environments with greater independence between state variables, where stochastic STRIPS operators are likely to more efficiently compress the environment model.
- The addition of a parameterised value learning system to estimate state values, allowing compression of the state value map (Tesauro 1994)
- Investigation of stochastic predicate logic rule learning methods to learn stochastic situation calculus rules (Muggleton 2000). This would also require the use of a relational reinforcement learning method (Dzeroski et. al. 2000) to learn state values.

6 Acknowledgements

Chris Child would like to acknowledge the support of EPSRC, grant number 00318484.

References

- Boutilier, C. Dean, T. Hanks, S. 1999. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research* 11: 1-94.
- Dzeroski, S. and De Raedt, L. and Blockeel, H. 1998. Relational Reinforcement Learning. *International Workshop on Inductive Logic Programming*.
- Drescher, G.L. 1991. *Made-Up Minds, A Constructivist Approach to Artificial Intelligence*. The MIT Press.
- Fikes, R.E. and Nilsson, N.J. 1971. STRIPS: a new approach to the application of theorem proving to problem-solving. *Artificial Intelligence* 2(3-4): 189-208.
- Hanks, S. 1990. Projecting plans for uncertain worlds. Ph.D. thesis, Yale University, Department of Computer Science.
- Kaelbling, L. P. and Littman, H.L. and Moore, A.P. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* 4: 237-285.
- Muggleton, S.H. 2000. Learning Stochastic Logic Programs. *Proceedings of the AAAI2000 Workshop on Learning Statistical Models from Relational Data*, L. Getoor and D. Jensen, AAAI
- Oates T., Schmill, M.D., Gregory, D.E. and Cohen P.R. 1995. Detecting complex dependencies in categorical data. Chap. in *Finding Structure in Data: Artificial Intelligence and Statistics V*. Springer Verlag.
- Oates, T. and Cohen, P. R. 1996. Learning Planning Operators with Conditional and Probabilistic Effects. *AAAI-96 Spring Symposium on Planning with Incomplete Information for Robot Problems*, AAAI.
- Sutton, R.S., and A.G. Barto. 1998. *Reinforcement Learning: An Introduction*. A Bradford Book, MIT Press.
- Shen, W. 1993. Discovery as Autonomous Learning from the Environment. *Machine Learning* 12: 143-165.
- Tesauro, G.J. 1994. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation* 6, 2: 215-219.
- Varsy, R. 2002. Extending Planning and Learning Through Reinterpretation of World Model. M.Sc. thesis, City Univesity.

Learning multi-agent search strategies

Malcolm J A Strens*

*Future Systems Technology Division, QinetiQ
1032/X107, Cody Technology Park, Farnborough, Hants. GU14 0LX. U.K.
mjstrens@qinetiq.com

Abstract

We identify a specialised class of reinforcement learning problem in which the agent(s) have the goal of gathering information (identifying the hidden state). The gathered information can affect rewards but not optimal behaviour. Exploiting this characteristic, an algorithm is developed for evaluating an agent's policy against *all possible* hidden state histories at the same time. Experimental results show the method is effective in a two-dimensional multi-pursuer evader searching task. A comparison is made between identical policies, joint policies and "relational" policies that exploit relativistic information about the pursuers' positions.

©QinetiQ Ltd, 2003.

1 Introduction

We address the reinforcement learning problem (Sutton and Barto, 1998) for episodic tasks in partially observable environments. These tasks are characterised by the presence of 'hidden state' which is not observable by the agent, although it may be revealed over the course of a trial.

For example, suppose the task is to search an area of the ground using a robotic vehicle (the *pursuer*, under the learning agent's control). Initially, the pursuer may only know that the evader (another vehicle perhaps) is within some given uncertainty area. The pursuer may also have available a dynamics model that determines how the evader could move over the course of a trial. The goal is simply to bring the evader within detection range of the pursuer's sensors (and at this point the trial ends). Therefore the only use that the pursuer can make of its sensor measurements during the course of the trial is to eliminate regions of the search area: *i.e.* the pursuer must reason about *where the evader could be*. This is an extreme example of a *dual control* problem: an agent must take actions to gather information on the way to its goal.

If the means available to the agent for observing the environment (*e.g.* sensors) do not allow the agent to gather instantaneously all the information that is relevant to its decision-making, then the task is a *partially observable* (PO) one. In PO tasks, reactive policies based on only the instantaneous observations are rarely effective: it is necessary for the agent to fuse information over time to make best use of the observations (and initial information). The searching task is a good example: the instantaneous observations are identical at every time step until the evader is found, and so no useful memoryless policy could be found. However, if the agent uses the observa-

tions to modify a longer-term memory representing where the evader could be at the current time, it can use this memory as the foundation for decision-making.

A standard model for reasoning about PO environments is the Partially Observable Markov Decision Process (POMDP). This models the transition dynamics of the (full) environment state, the expected return for every (state, action) pair, and a stochastic function mapping states to observations. Most success for large POMDP problems has come from restricting the complexity of learning by choosing the agent's policy from a parameterised family (our approach), rather than estimating state-action values for every information state. The learning problem becomes one of finding appropriate policy parameters. However, even with parameterised policies there remains a difficulty in how to define the 'inputs' to the policy. In fully observable problems, these inputs should be a compact representation of the state (a *feature vector*) that carries with it all information relevant to decision-making. Similarly, in PO problems the inputs can be *information features* that encode aspects of the belief relevant to decision-making. For example, in the detection task where the belief is an evader uncertainty area (EUA), candidate sets of features might include informative points in the EUA (center of mass, extrema of boundaries) or spatial moments of the distribution.

Section 2 gives formal descriptions for a POMDP and the proposed specialisation and *information gathering problem*. Section 3 introduces recursive Bayesian filters for tracking beliefs, including the particle filter used in the experiments. Section 4 describes a set of appropriate information features for the searching task, and three types of policy parameterisation (individual, joint, and relational). Section 5 describes a fast way of learning in information gathering problems that exploits conditional independence between the true hidden state and the optimal behaviour, given the belief. Section 6 describes direct search methods for finding effective policy paramete-

ters. Section 7 describes evaluation of the approach using a multi-pusuer evader task, and section 8 concludes.

2 A special class of POMDP

Before describing a specialised version of the POMDP for information gathering problems, we give a formal description of the existing MDP and POMDP models. An MDP is a discrete-time model for the stochastic evolution of a system's state, under control of an external input (the agent's actions). It also models a stochastic reward that depends on the state and action.

Definition A Markov Decision Process is given by $\langle X, A, T, R \rangle$ where X is a set of states and A a set of actions. T is a stochastic transition function defining the likelihood the next state will be $x' \in X$ given current state $x \in X$ and action $a \in A$: $P_T(x'|x, a)$. R is a stochastic reward function defining the likelihood the immediate reward will be $r \in \mathbb{R}$ given current state $x \in X$ and action $a \in A$: $P_R(r|x, a)$.

A POMDP is a general-purpose discrete-time mathematical model for reasoning about single-agent interaction in the presence of partial observability. The POMDP assumes that the agent receives observations that do not necessarily convey the full hidden state of the environment.

Definition A POMDP $\langle X, A, T, R, Y, O \rangle$ builds upon a MDP $\langle X, A, T, R \rangle$ by adding a set of observations Y and an observation function O that generates stochastic observations from the hidden state according to $P_O(y|x)$.

2.1 Optimal behaviour in POMDPs

Reinforcement learning for partially observable problems is generally more difficult than for fully-observable ones. The reason for this can be seen by analysing optimal behaviour in a POMDP (assuming all parts of the POMDP are known¹). Optimal behaviour is given not by a policy that maps observations to action probabilities, but by a policy that maps *beliefs* to action probabilities. The agent's belief is the estimate for $P(x|H)$ where H is the complete interaction history. This belief is a probability distribution over X . It is possible to find the agent's optimal policy by constructing a MDP from the POMDP but with a much larger state space corresponding to the set of possible beliefs. This larger MDP can, in theory, be solved by standard RL methods. In practice, however, the belief space is usually so large that the methods are intractable without some kind of approximation. For example Thrun et al. (1999) estimated state-action values

¹If the observation, transition and reward functions of the POMDP are not known *a priori* an even more difficult learning problem is encountered.

for a discrete set of belief 'exemplars', each corresponding to a particle filter estimate of the belief. The KL divergence was used as a distance-measure between information states, to allow the nearest neighbor exemplar to be found for each information state encountered during learning. In the worst case, the belief space has size exponential in the number of time steps of interaction.

2.2 Information gathering problems

Now we investigate special cases where part of the state is *fully* hidden. By *fully* hidden we mean that observations that depend on this state do not affect optimal behaviour (and are therefore excluded from the model). Firstly, suppose that the fully hidden state represents the location of a physical entity (e.g. a vehicle) and that the agent's *only* goal is to discover this information. When the information is discovered the trial ends. In this case the agent knows its own physical state, and observations convey no information about the hidden state until the end of the trial. Therefore optimal searching behaviour is not dependent on these observations. More generally, consider an agent that aims to discover some part of the hidden state, but passes the information on to another agent for action. The trial does not end, but from that point on the agent has "no interest" in that part of the state. An example would be a remote surveillance system (e.g. a relocatable satellite) that must plan its own course to gather imagery at a set of locations, and report this information for interpretation by another system (e.g. a ground station). This is an example of a *continuing* information gathering problem (IGP) for which the following definition is proposed:

Definition An IGP is given by $\langle X, S, A, T^h, T^s, R \rangle$. The fully hidden part of the state $x \in X$ evolves according to a transition function $P_{T^h}(x'|x, s, a)$. The observable part of the state $s \in S$ evolves separately according to a transition function $P_{T^s}(s'|s, a)$. R defines the stochastic immediate reward $P_R(r|x, s, a)$ which may depend on both the hidden part and the observable part of the state.

Note that there is no observation function because the observations are identical to the observable part of the state (s). The observable state could incorporate the physical state of the agent(s) and any other observable scenario information. A stationary policy for the agent is now a stochastic function of (i) the belief $b \equiv P(x|H)$ over the fully hidden state and (ii) the fully observable state:

$$\pi(b, s, a) \equiv P(a|b, s)$$

The benefit of this new formulation is that the belief b evolves independently of the actual fully hidden state x . (There are no observations to convey information about it.) The belief b is obtained simply by a recursive Bayesian filter that has no information update step.

3 Recursive Bayesian filtering

In many applications it is feasible to estimate the belief (even when is not feasible to enumerate the space to represent a value function). Using the POMDP notation, the full state of the environment is a vector x_t representing the position and motion of one or more objects. The belief is a probability density for x_t given the interaction history $H_t \equiv (A_{t-1}, Y_t)$ where $A_{t-1} \equiv (a_1, \dots, a_{t-1})$ is the action history and Y_t is the observation history, and the initial information $P(x_0)$. The recursive Bayesian filter makes use of a dynamics model $P_T(x'|x, a)$ and an observation model $P_O(y|x)$ to recursively estimate the belief:

$$\underbrace{P(x_{t+1}|A_t, Y_t)} = \int_{x_t} P(x_t|H_t) P_T(x_{t+1}|x_t, a_t) dx_t$$

$$P(x_{t+1}|H_{t+1}) \propto P_O(y_{t+1}|x_{t+1}) \underbrace{P(x_{t+1}|A_t, Y_t)}$$

There are many possible implementations. The Kalman filter approximates the belief by a multivariate Gaussian. Particle filters use a ‘cloud’ of samples as the representation, allowing multi-modal distributions to be represented. Grid-based filters discretise the state space and store a probability at each grid point.

3.1 The particle filter

Our experiments make use of the particle filter (Doucet et al., 2001) because it is very easy and convenient to work with: it represents the belief as a sum of weighted hypotheses $\{(x_t^i, w_t^i)\}$:

$$P(x_t|Z_t) = \sum_{i=1}^N w_t^i \delta(x_t^i, x_t)$$

The update can be implemented (separately for each particle) by importance sampling, using a some proposal density $q(x_{t+1}^i|x_t^i, H_{t+1})$ then re-weighting the particles according to:

$$w_{t+1}^i \propto w_t^i \frac{P_O(y_t|x_t^i) P_T(x_{t+1}^i|x_t^i, a_t)}{q(x_{t+1}^i|x_t^i, H_{t+1})}$$

Over the course of time, the weights of the particles may become imbalanced causing the set of particles to be a poor representation of the true belief. To overcome this problem, *resampling* is usually necessary. Resampling makes all weights equal, but there tend to be more copies in the new population of the particles that had the largest weights.

3.2 Particle filter for fully hidden state

The full recursive Bayesian filter is not required in information gathering problems. In particular, there are no observations conveying information about the hidden state,

and so the belief depends only on the initial distribution $P(x_0)$ and the dynamics expressed as a transition model $P_T^h(x_{t+1}|x_t, a_t)$ (now using IGP notation). The update becomes a single step:

$$P(x_{t+1}|H_{t+1}) = \int_{x_t} P(x_t|H_t) P(x_{t+1}|x_t, s_t, a_t) dx_t$$

where $H_t \equiv (A_{t-1}, S_t)$ and $S_t \equiv (s_1, \dots, s_t)$ is the observable state history. We will be use this “history” filter for two different purposes: representing an agent’s beliefs and representing an ensemble of scenarios for evaluation.

A detection event in the pursuer-evader problem will lead to the weight of the corresponding hypothesis being set to 0. If resampling were then to take place, these zero-weight hypotheses would be replaced in the filter by duplicates of other hypotheses; i.e. the computational effort of the whole particle filter would be focussed on cases where the evader would not yet have been intercepted. However, there is a strong argument for not resampling². Given that no computational effort is expended on zero-weight particles, the processing requirement will be proportional to the likelihood that the evader has not yet been detected. This implies that, without resampling, computational load is matched to the maximum change in total return from the trial. Therefore, in the pursuer-evader problem, the particle filter has been reduced to a very simple update for each hypothesis, implementing the evader’s instantaneous motion.

4 Policy parameterization

The uncertainty in the evader’s location is given at any instance in time by a belief consisting of $N_H = 256$ hypotheses (particles in the filter). To pose the problem as one of direct policy search, a set of information features is required that summarises the set of particles adequately for robust decision-making. This process is sometimes called *belief compression* (Roy and Gordon, 2002) The features described here make use only of the position part of each particle’s state.

4.1 Information features

Let (x_i, y_i) be the position vector for hypothesis i expressed in a coordinate system of a pursuer of interest³. To obtain a compact set of information features, we integrate over the particles using a small number of basis

²However, resampling is required if the transition function does not assign similar probabilities to all feasible outcomes.

³This coordinate system has its x-axis parallel to the pursuer’s velocity vector. A nonlinear transformation is also applied: a displacement of (u, v) in the pursuer’s coordinate system is mapped to $(x, y) \equiv (u/\{R_0 \sqrt{|u|}\}, v/\{R_0 \sqrt{|v|}\})$, in order to provide more spatial resolution close to the agent. R_0 is a constant that will determine the effective spatial extent of the basis functions, and is chosen equal to 512 in our evaluation.

functions ϕ_{jk} for $j \in \{1, 2\}$ and $k \in \{1, 2, 3, 4\}$. A suitable set is given by the regularised spatial derivatives:

$$\phi_{jk}(x, y) = H(j, 2k - 2)$$

where:

$$H(m, n) = \frac{\partial^m}{\partial x} \frac{\partial^n}{\partial y} \exp\left(-\frac{x^2 + y^2}{2}\right)$$

These basis functions are all derivatives of a Gaussian distribution placed at the origin (of the pursuer's coordinate system). They are mutually orthogonal and (multiplicatively) separable in the two axis directions. Only even derivatives are selected in the y-direction; the odd ones are of no interest as a result of symmetry in the problem. This yields a set of information features:

$$z_{jk} = \frac{1}{N_H} \sum_i \phi_{jk}(x_i, y_i)$$

We then define an information-state value function:

$$V_\alpha(x_i, y_i) = \sum_{j,k} \alpha_{jk} C_{jk} z_{jk}$$

C_{jk} is a constant that ensures the magnitudes of the information features are balanced:

$$C_{jk}^{-1} \equiv \int_x \int_y \phi_{jk}^2(x, y) dy dx$$

The 2×4 parameter matrix α will be the target for learning. We make use of the smooth dynamics to avoid computing V explicitly. Instead, each pursuer selects its action (turn left or turn right) according to the sign of the gradient of V with respect to the angle θ of its velocity vector. The gradient is given here without proof:

$$\frac{\partial V}{\partial \theta} \equiv \sum_i \sum_{j,k} C_{jk} \{x_i H(j, 2k - 1) - y_i H(j + 1, 2k - 2)\}$$

Since only the sign of the gradient is used to determine the pursuer's action, an arbitrary scaling of α will not affect its behaviour. This redundancy is eliminated by requiring $\|\alpha\| = 1$.

4.2 Joint and relational policies

In general, every pursuer need not be given the same policy. Instead, the problem can be regarded as a search for the joint policy $(\alpha_1, \dots, \alpha_{N_P})$ which has $8N_P$ dimensions. We will evaluate whether the extra representational freedom (and complexity) of a joint policy can be exploited.

In order to allow any number of pursuers to *cooperate* without the policy size increasing, a simple relational policy has also been designed. Every pursuer has a copy of the policy which consists of two components: α (as before) and β which contains information about the relative

positions of the UAVs. β affects the value function in the same way as α except that the basis functions are summed over pursuer locations instead of particle locations:

$$z_{jk}^{rel} = \frac{1}{N_P - 1} \sum_l \phi_{jk}(x_l, y_l)$$

$$V_\beta(x_l, y_l) = \sum_{j,k} \beta_{jk} C_{jk} z_{jk}^{rel}$$

where (x_l, y_l) is the location of pursuer l expressed in the coordinate system of the pursuer for whom the decision is to be made. Using relational policies, it is the gradient of the combined value function $V_\alpha + V_\beta$ that determines each pursuer's action. The dimension of this policy is 16 no-matter how many pursuers are present. (Also, the number of pursuers taking part in the search could change dynamically during a trial.) This approach will be evaluated to determine whether pursuers are aided by taking account of each other's locations.

5 Dual particle filter method

One major benefit of identifying the problem as an "information gathering process" rather than (the more general) POMDP is that specialised solution methods can be derived. In the pursuer-evader problem, the pursuers' locations are the fully observable state (s) and the evader's location is the hidden state. When the evader comes within detection range of a pursuer, the trial ends. Therefore, the hidden state is never used to determine future actions: i.e. observation of the hidden state conveys no information about optimal behaviour (pursuer policy). The implication of this is that *any proposed policy can be tested against all possible hidden state histories in parallel*. The outcome of a parallel trial is not success or failure (evader detected or not) but instead the *proportion* of the histories in which the evader was detected.

The method proposed here uses one particle filter to represent the pursuers' beliefs about the location of the evader, and a separate particle filter to represent hidden state histories for parallel evaluation. Although it may not be immediately obvious, the two particle filters are estimating the same distribution: the possible state of the evader at each time step. Nevertheless, it is essential that they are kept separate because if the approximation errors (inevitable in a particle filter) were correlated, a grossly overoptimistic estimate of the performance of a policy would be obtained.

We found that 256 particles in each filter were sufficient, whereas using a single filter for both estimation tasks means that over 4000 particles are required for similar performance. Each trial does not end until the evader has been detected in every parallel scenario, or a maximum duration has expired. This approach is also very efficient compared with the naive alternative: running 256 separate trials; one for each possible trajectory taken by the evader.

6 Direct Policy Search

Suppose that the agent's policy has a parameter vector w of size m ; for example $w \equiv \alpha$ for pursuers with identical policies; $w \equiv (\alpha_1 :: \dots :: \alpha_{N_p})$ for joint policies (where $::$ indicates concatenation of vectors); or $w \equiv \alpha :: \beta$ for relational policies. A single simulation trial in which w defines the pursuers' behaviours will lead to a stochastic return. The aim of learning is to find a w that maximises the expected return. *Direct policy search* methods attempt to optimise expected return without reference to performance gradient information. Some methods use a large number of trials for each proposed w to obtain a near-deterministic evaluation, then apply a deterministic optimization procedure (e.g. downhill simplex method). However, here we use a method that can work with (noisy) policy comparisons (Strens and Moore, 2001).

6.1 Differential evolution

Our approach is a variant of *differential evolution*, an evolutionary method that operates directly on a population of real-valued vectors (rather than binary strings) (Storn and Price, 1995). Proposals are obtained by linear combination of existing population members.

Initially, the population is chosen randomly from some prior density on w . To generate each proposal, one candidate in the population is chosen (systematically) for improvement. Then the vector difference between two more randomly⁴ chosen members, weighted by a scalar parameter, F , is added to a third randomly chosen 'parent'. Crossover (see below) takes place between this and the candidate, to obtain a proposal point. The proposal is compared with the candidate by running a small set of new simulation trials for each. To reduce variance in this comparison, the same set of scenarios⁵ is used for every comparison. According to the outcome *either* the proposal replaces the candidate *or* the population remains unchanged. In either case, a new candidate for replacement will be selected on the next iteration.

Crossover is implemented here by selecting each element of the proposal vector from either the candidate or the new vector with equal probability. Crossover helps to prevent the population from become trapped in a subspace. For each proposal, $\log_2 F$ was chosen uniformly from the range $[-10, 0]$ and the result was scaled by a value $F_{max}(t)$ that decreased with time. ($\log_2 F_{max}(t)$ was reduced uniformly from 0 to -10 during learning.) This ensures convergence of the population.

⁴ All vectors used in the process of generating a proposal are mutually exclusive.

⁵ A scenario is an initial configuration for the pursuers' locations

6.2 Illustration

Figure 1 illustrates this process in more detail: the weighted difference between two population members (1,2) is added to a third population point (3). The result (4) is subject to crossover with the candidate for replacement (5) to obtain a proposal (6). The proposal is evaluated (using a number of trials) and replaces the candidate if it is found to be better. Note that the proposal could be identical to (4) or (5), depending on the outcome of crossover.

This form of DE has a very useful property; replacing any one population member due to an occasional incorrect comparison is not catastrophic. It suffices that the comparison be unbiased, and correct with probability only slightly better than chance (0.5) in large populations (Strens and Moore, 2002). This means that it is possible to use only a small number of simulation trials (4 in this case) per proposal.

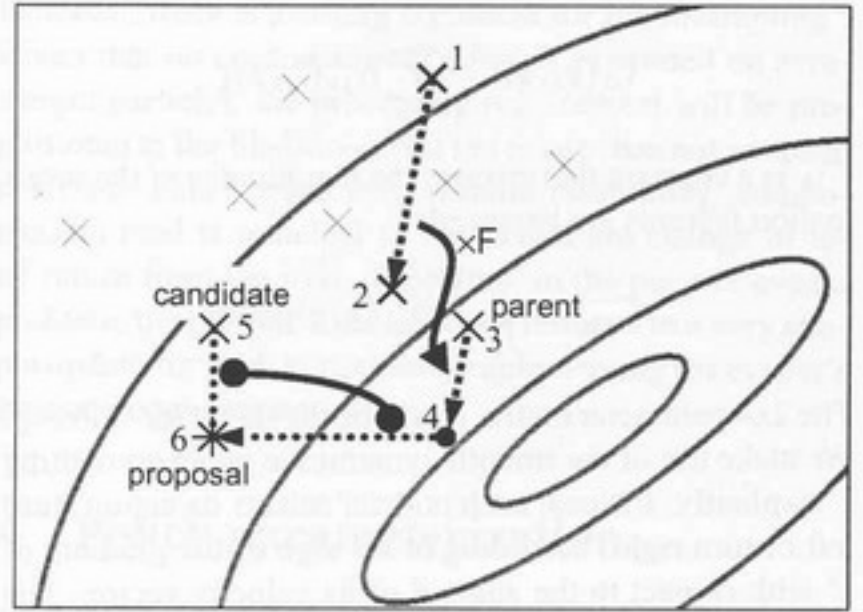


Figure 1: Obtaining a new proposal in differential evolution.

7 Evaluation

We perform a comparison between individual, joint and relational policies for different numbers of pursuers. The aim is to demonstrate the effectiveness of the dual particle filter method and to collect some evidence about the best way to structure a policy for this type of task.

7.1 Multi-pursuer evader task

We consider a multi-pursuer evader task in which the observable state is the location and velocity of each pursuer, and the hidden state is the location and goal direction of an evader. The pursuers must cooperate to detect the evader before the uncertainty in its position becomes too large.

- **States.** Each pursuer's state is a location on the 2-dimensional plane and a motion direction. The evader's state is a location in the plane and a preferred direction; therefore each particle filter's will be a 3-vector.
- **Scenarios.** The initial belief for the evader's location is an uncertainty area given by a Gaussian distribution with standard deviation 100, located at the origin. The pursuers' all start at a distance from the origin chosen uniformly from [2000, 3000]. Their angular separation (subtended from the origin) is 30 degrees. Note that rotational and translational invariance of our formulation will mean that the policy obtained will work just as well for any rotated or translated versions of this set of scenarios.
- **Dynamics.** The pursuers can move at a speed of 24 (length units per time step) and the evader at 1. The evader can instantaneously change its direction whereas the pursuers have a maximum turn angle of $\pi/8$ at each time step. In our implementation, each pursuer selects either $+\pi/8$ or $-\pi/8$ according to the information-state value function's gradient. It remains possible for a pursuer to follow an almost straight course by alternating between these two actions. The evader's behaviour is given by a stochastic policy: with equal probability it either moves in its preferred direction or moves in a direction that takes it away from the nearest pursuer (ensuring the problem is non-trivial⁶). If the evader's position is within detection range (24 units) of a pursuer.
- **Termination.** The maximum trial duration is 256 steps. With a naive implementation in which only one evader trajectory (history) is used in each trial, detection would often be completed within this time limit. However, using the dual particle filter method, the trial will not end until detection takes place in *every* hypothesis for the evader's trajectory. Therefore, often the maximum duration will be reached (but with a diminished number of active particles in each filter.)
- **Returns.** The return at the end of a simulation trial is the proportion of the original probability mass within the evaluation filter that has been eliminated during the course of the trial. This represents the expected return over the 256 parallel scenarios described above, and can be interpreted simply as the probability that the evader is found within the trial duration. *The differential evolution method uses this return as the (stochastic) evaluation of the weight vector associated with that trial.*

Figure 2 shows the trajectories of 3 pursuers during a trial as sequences of connected circles. The radius of each circle is the area within which the evader can be detected. The plotted markers indicate the state of each of the particle filter at the end of a trial. In the colour version of this paper, the central mass of green points indicates particles that have been detected (these are frozen in the locations at which they were detected, for display purposes). The outer (black and blue) points⁷ represent hypotheses for the current location of an evader that has not yet been captured. A more detailed view of the central circle (radius 400 units) is also given, for a different learning trial. It shows a strategy has been learnt for the 3 pursuers that covers the central part of the space very well. By this stage in the trial the pursuers have moved back out toward the edge of this central circle, because the evader could have reached these areas.

7.2 Configuration of learning system

Differential evolution used a population size of 16 and F was chosen as described in section 1. 256 particles were used in each filter. 4 trials were performed for each policy evaluation (with pursuers at distances of 1125, 1375, 1625 and 1875 from the origin). Therefore 8 trials were needed for each policy comparison, and so 512 comparisons were possible in the total budget of 4096 trials. (This is a relatively small number of function evaluations for an evolutionary algorithm.) The best policy within the population was tested at the end of learning, using a set of 512 scenarios, in which the initial pursuer positions were uniformly distributed in the interval [2000,3000]. A baseline performance was obtained using a very simple policy in which $\alpha_{11} = 1$ and all other elements are 0. This causes each pursuer to turn towards the centre of probability mass of the evader's position distribution, taken under the weighting function $H(0, 0)$.

7.3 Results

Table 1: Success rate (%) for different policy types.

N_P	1	2	3	4
Baseline	46 $\pm$ 4	64 $\pm$ 3	75 $\pm$ 4	81 $\pm$ 3
Identical	47 $\pm$ 7	72 $\pm$ 3	80 $\pm$ 2	88 $\pm$ 2
Joint	NA	73 $\pm$ 6	79 $\pm$ 3	85 $\pm$ 2
Relational	NA	72 $\pm$ 6	86 $\pm$ 3	93 $\pm$ 1

Table 1 shows the performance of the baseline policy and three types of learnt policy as a percentage (likelihood of detecting evader), for different number of pursuers. The error bounds given are at one standard deviation.

⁶In particular it ensures there is a dependency between the pursuers' actions and the evolution of the hidden state; otherwise it would be possible to pre-compute evader trajectories.

⁷The difference between black and blue is that the points are taken from the 2 particle filters. If there is little overlap between the distributions this provides evidence that more particles are required.

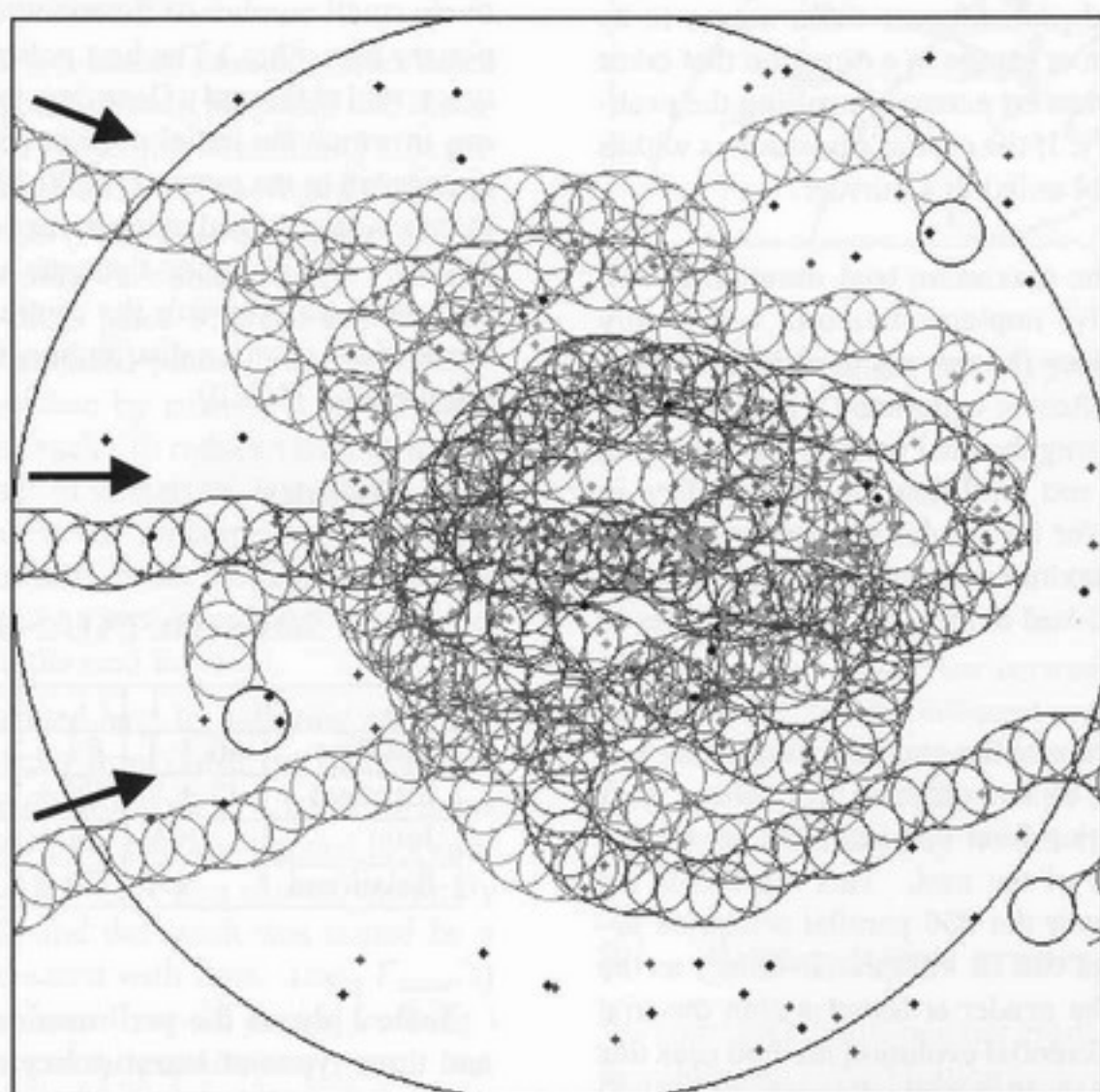
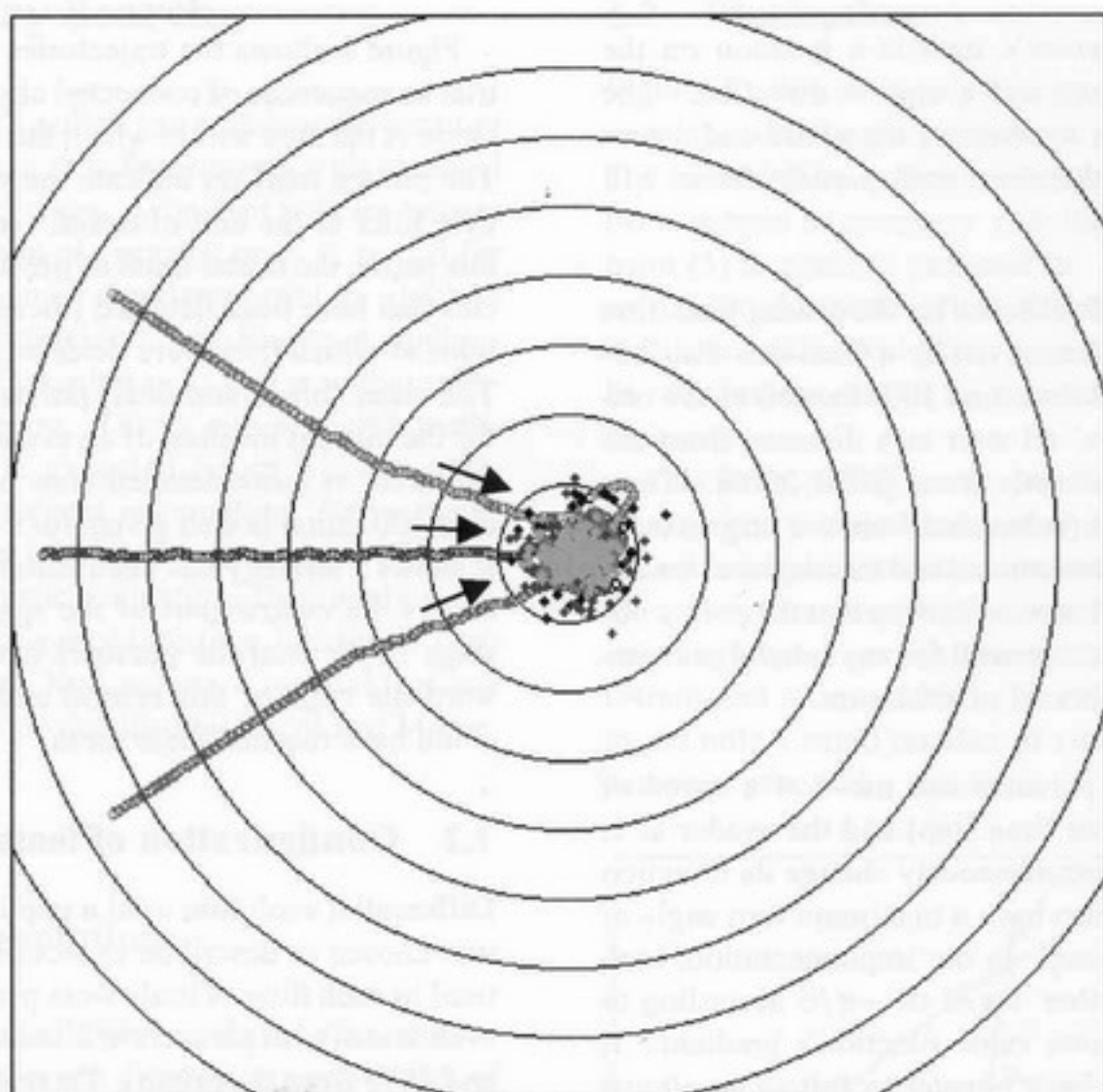


Figure 2: Pursuer trajectories and final particle states in a simulation trial. Full trial (top); detail from a different trial (below).

ation: the standard errors ($n = 500$) are about 20 times smaller so a difference of 2% is significant, using Gaussian statistics.

The task is sufficiently difficult that no method is able to obtain 100% success, even with 4 pursuers. This is not surprising, because the uncertainty in the evader position expands at a rate that increases rapidly with time. The baseline strategy (no learning) provides performance that increases with the number of pursuers: this indicates that the pursuers do not all converge onto the same trajectories even though they are acting independently according to a simple strategy. Introducing learning improves performance (significantly) for 2 or more pursuers even though the pursuers continue to ignore each others' positions in their decision-making. The learnt joint policy, in which each pursuer has a separate set of policy parameters performs very similarly: it seems there is no great advantage in allowing each to have a different strategy. There is actually a performance decrease with 4 pursuers, probably because the joint strategy has 32 parameters and so could be expected to take much longer to learn than the identical (8 parameter) strategies.

The learnt relational policies are equally effective (compared with the other learnt strategies) for 2 pursuers, but show major benefits with 3 or more pursuers. This indicates pursuers that are aware of each others' positions can "divide and conquer" the search problem in a more systematic way. The gains that have been obtained are very significant: with 4 pursuers the change that the evader escapes has been halved (compared with the other learnt policies) and reduced by nearly 3 times compared with the baseline policy.

8 Conclusions

Some information gathering problems such as searching tasks have special structure. Although there is hidden state, and beliefs must be tracked, the actual hidden state does not affect optimal behaviour. In other words, optimal behaviour is conditionally independent of the true hidden state given the belief. Therefore it is possible to evaluate a policy against any number of hidden state trajectories in a single learning trial.

The implementation described here matches the computational burden of agent reasoning (belief revision) and simulation (generating hidden state histories) exactly: a particle filter can be used for each of these processes. Furthermore, these two particle filters are performing the same estimation task. They are kept separate in the learning algorithm only because there are benefits if errors are not correlated. When the learnt strategy is transferred into a real environment (not a simulation) only the particle filter representing agent beliefs would be retained: the second filter is part of the learning algorithm rather than the agent itself.

When learning policies for multiple homogeneous

agents, it is not necessary to learn separate policies for each agent (the joint policy option); however it is important that the agents make use of relational information (e.g. relative position) in order to divide the workload effectively. The "relational policy" achieved this by applying the same set of basis functions (used for obtaining evader information features) to represent the relative positions of the other pursuers. The deterministic structure of the pursuers' dynamics was also exploited in the policy formulation, that might otherwise have required a state-action value function to be estimated.

Acknowledgements

This research was funded by the UK Ministry of Defence Corporate Research Programme in Energy, Guidance and Control.

References

A. Doucet, J. F. G. de Freitas, and N. J. Gordon (Eds.). *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. A novel approach to nonlinear and non-gaussian bayesian state estimation. *IEE Proceedings on Radar and Signal Processing*, 140:107–113, 1993.

N. Roy and G. Gordon. Exponential family pca for belief compression in pomdps. In *Advances in Neural Information Processing Systems*, 2002.

R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, 1995.

M. J. A. Strens and A. W. Moore. Direct policy search using paired statistical tests. In *Proceedings of the 18th International Conference on Machine Learning*, pages 545–552. Morgan Kaufmann, San Francisco, CA, 2001.

M. J. A. Strens and A. W. Moore. Policy search using paired comparisons. *Journal of Machine Learning Research*, 3:921–950, 2002.

R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.

S. Thrun, J. C. Langford, and D. Fox. Monte Carlo hidden Markov models: Learning non-parametric models of partially observable stochastic processes. In *Proceedings of the 16th International Conference on Machine Learning*, pages 415–424. Morgan Kaufmann, San Francisco, CA, 1999.

Learning to coordinate using commitment sequences in cooperative multi-agent systems

Spiros Kapetanakis*, Daniel Kudenko*

*Department of Computer Science
University of York, Heslington, York, YO10 5DD, UK
{spiros,kudenko}@cs.york.ac.uk

Malcolm Strens†

†Future Systems Technology Division, QinetiQ,
Ively Road, Farnborough, Hampshire, GU14 0LX, UK
mjstrens@qinetiq.com

Abstract

We report on an investigation of the learning of coordination in cooperative multi-agent systems. Specifically, we study solutions that are applicable to *independent* agents i.e. agents that do not observe one another's actions. In previous research (Kapetanakis and Kudenko, 2002) we have presented a reinforcement learning approach that converges to the optimal joint action even in scenarios with high miscoordination costs. However, this approach failed in fully stochastic environments. In this paper, we present a novel approach based on reward estimation with a shared action-selection protocol. The new technique is applicable in fully stochastic environments where mutual observation of actions is not possible. We demonstrate empirically that our approach causes the agents to converge almost always to the optimal joint action even in difficult stochastic scenarios with high miscoordination penalties.

1 Introduction

Learning to coordinate in cooperative multi-agent systems is a central and widely studied problem, see, for example Lauer and Riedmiller (2000); Boutilier (1999); Claus and Boutilier (1998); Sen et al. (1994); Weiss (1993). In this context, coordination is defined as *the ability of two or more agents to jointly reach a consensus over which actions to perform in an environment*. We investigate the case of *independent* agents that cannot observe one another's actions, which often is a more realistic assumption. This generality distinguishes our approach from alternatives (e.g., Wang and Sandholm (2002)) that require complete mutual observation of actions.

In this investigation, we focus on scenarios where the agents must learn to coordinate their actions through environmental feedback. In previous research (Kapetanakis and Kudenko, 2002) we have presented a reinforcement learning technique (called FMQ) for independent agents, that converged to the *optimal* joint action in scenarios where miscoordination is associated with high penalties. Previous Q-learning approaches failed to solve such scenarios with independent agents, and the FMQ technique was the first to do so. However, the FMQ approach failed in fully stochastic environments, where the rewards associated with joint actions are randomised.

In this paper, we present a novel technique that is based on a shared action-selection protocol (called *commitment sequence*) that enables the agents to estimate the rewards for specific joint actions. We evaluate this approach experimentally on a number of stochastic versions of two especially difficult coordination problems that were first introduced in (Claus and Boutilier, 1998): the *climbing*

game and the *penalty game*. The empirical results show that the convergence probability to the optimal joint action is very high, in fact reaching almost 100%.

Our paper is structured as follows: we first introduce a common testbed for the study of learning coordination in cooperative multi-agent systems: stochastic cooperative games. We then introduce a basic version of the novel commitment sequence technique that uses simple averaging and discuss the experimental results. Finally, we present an extension using Gaussian estimation that improves both the probability of convergence to the optimal joint action and the convergence speed. We finish with an outlook on future work.

2 Stochastic cooperative games

A common testbed for studying the problem of multi-agent coordination is that of stochastic cooperative games (Fudenberg and Levine, 1998). In these games, the agents may have common, conflicting or complementary interests. In this work, we have concentrated only on cooperative games i.e. games where the agents are rewarded based on their joint action and all agents receive the same reward. At present, we only consider single-stage stochastic games. In these games, every agent chooses an action from its action space for every round of the game. These actions are executed simultaneously and the reward that corresponds to the joint action is broadcast to all agents. In multi-stage stochastic games, the execution of a joint action yields not only a common reward but also the transition of the agents to a new state.

Table 1 describes the reward function for a simple co-

operative single-stage game. For example, if agent 1 executes action b and agent 2 executes action a , the reward they receive is 5. Obviously, the optimal joint-action in this simple game is (b, b) as it is associated with the highest reward of 10.

		Agent 1	
		a	b
Agent 2	a	3	5
	b	0	10

Table 1: A simple cooperative game reward function.

Our goal is to enable the agents to learn optimal coordination from repeated trials in cases where the game matrix is not known to the agents. To achieve this goal, one can use either *independent* or *joint-action* learners. The difference between the two types lies in the amount of information they can perceive in the game. Although both types of learners can perceive the reward that is associated with each joint action, the former are unaware of the existence of other agents whereas the latter can also perceive the actions of others. In this way, joint-action learners can maintain a model of the strategy of other agents and choose their actions based on the other participants' perceived strategy. In contrast, independent learners must estimate the value of their individual actions based solely on the rewards that they receive for their actions. In this paper, we focus on individual learners, these being more universally applicable.

In the present study, we analyse a number of coordination problems, all of which are descendants of the *climbing game* and the *penalty game*. We show why these problems are of interest to us and why they are hard problems. The climbing game is representative of problems with high miscoordination penalties and a single optimal joint action, whereas the penalty game is representative of problems with miscoordination penalties and multiple optimal joint actions. Both games are played between two agents and their reward functions are shown in Tables 2 and 3:

		Agent 1		
		a	b	c
Agent 2	a	11	-30	0
	b	-30	7	6
	c	0	0	5

Table 2: The climbing game.

		Agent 1		
		a	b	c
Agent 2	a	10	0	-10
	b	0	2	0
	c	-10	0	10

Table 3: The penalty game.

In the climbing game, it is difficult for the agents to converge to the optimal joint action (a, a) because of the negative reward in the case of miscoordination. For example, if agent 1 plays a and agent 2 plays b , then both will receive a negative reward of -30. Incorporating this reward into the learning process can be so detrimental that both agents tend to avoid playing the same action again. In contrast, when choosing action c , miscoordination is not punished so severely. Therefore, in most cases, both agents are easily tempted by action c . The reason is as follows: if agent 1 plays c , then agent 2 can play either b or c to get a positive reward (6 and 5 respectively). Even if agent 2 plays a , the result is not catastrophic since the reward is 0. Similarly, if agent 2 plays c , whatever agent 1 plays, the resulting reward will be at least 0. From this analysis, we can see that the climbing game is a challenging problem for the study of learning coordination. It includes heavy miscoordination penalties and "safe" actions that are likely to tempt the agents away from the optimal joint action.

Similarly, the penalty game is a hard problem as it not only has heavy penalties for miscoordination but also includes multiple optimal joint actions. If agent 1 plays a expecting agent 2 to also play a so they can receive the maximum reward of 10 but agent 2 plays c (perhaps expecting agent 1 to play c so that, again, they receive the maximum reward of 10) then the resulting penalty can be very detrimental to both agents' learning process. In this game, b is the "safe" action for both agents since playing b is guaranteed to result in a reward of 0 or 2, regardless of what the other agent plays.

Because of the difficulties discussed above, regular Q-learning agents failed to converge to the optimal joint action in both the climbing game and the penalty game. A Q-learning variant that solves these games for independent agents has been presented in (Kapetanakis and Kudenko, 2002) using an approach known as the FMQ heuristic. This heuristic works in a reinforcement learning setting and it allows two agents that have no communication capabilities or shared knowledge to jointly reach the optimal joint action in single-stage games.

However, the FMQ heuristic cannot distinguish adequately between miscoordination penalties and reward variance in stochastic games. Therefore, it fails to solve more complex games such as the *stochastic climbing game* which is shown in Table 4. The stochastic version of the climbing game differs from the original in that each joint action now corresponds to two rewards instead of just one. These two rewards are received with probability $1/2$. If the two agents were to commit to playing a specific joint action indefinitely, the reward they would accumulate over time would converge to the same value as in the original game. In this respect, the stochastic climbing game is equivalent to the original. This equivalence is maintained across all variations of the climbing and penalty game that we introduce in this work.

The difficulty in solving the stochastic climbing game

		Agent 1		
		<i>a</i>	<i>b</i>	<i>c</i>
Agent 2	<i>a</i>	10/12	5/-65	8/-8
	<i>b</i>	5/-65	14/0	12/0
	<i>c</i>	5/-5	5/-5	10/0

Table 4: The stochastic climbing game table (50%).

with the FMQ heuristic stems from the fact that the heuristic is designed to deal with one type of uncertainty, namely that which arises from sampling heavily penalised actions. This uncertainty is due to the inability of one agent to observe the other agent's actions. The FMQ heuristic filters out the noise from failed coordination attempts to help the agents reach the optimal joint action. The difference in the stochastic version of the climbing game is that there are now two sources of uncertainty in the game, the other agent's actions (as before) and the multiple rewards per joint action.

We will show two ways to tackle the fully stochastic climbing game using the idea of commitment sequences. We also presently introduce two new variants of the game so as to show the full potential of our methods. The first variant is the *three-valued climbing game* in which that there are three rewards corresponding to any joint action. This game is shown in Table 5.

		Agent 1		
		<i>a</i>	<i>b</i>	<i>c</i>
Agent 2	<i>a</i>	16/22/-5	4/6/-100	10/20/-30
	<i>b</i>	4/6/-100	25/0/-4	10/5/3
	<i>c</i>	8/12/-20	10/20/-30	4/5/6

Table 5: The three-valued stochastic climbing game.

The probability of receiving any one of the three rewards that correspond to each joint action is $1/3$. If the two agents were to play the action profile (b, b) indefinitely, they would accrue an average reward of 7 as in the original game.

The next variant of the climbing game that we introduce is the *variable-probability climbing game*. There are again two rewards for each joint action. However, they are now received with non-uniform probabilities. Equivalence with the original game is again maintained. The variable-probability climbing game is included in Table 6. The notation $(\pi)n$ signifies that the probability of getting a reward of n for playing that joint action is π . For example, the probability of getting a reward of 5 for joint action (c, b) is 0.8 whereas the probability of getting a reward of 10 for the same joint action is 0.2.

Finally, we introduce a stochastic variant of the penalty game. This game is called the *stochastic penalty game* and is shown in Table 7.

		Agent 1		
		<i>a</i>	<i>b</i>	<i>c</i>
Agent 2	<i>a</i>	(0.4) 6.5 (0.6) 14	(0.25) -36 (0.75) -28	(0.6) 4 (0.4) -6
	<i>b</i>	(0.25) -36 (0.75) -28	(0.8) 5 (0.2) 15	(0.8) 5 (0.2) 10
	<i>c</i>	(0.7) 3 (0.3) -7	(0.6) 4 (0.4) -6	(0.8) 4 (0.2) 9

Table 6: The variable-probability stochastic climbing game.

		Agent 1		
		<i>a</i>	<i>b</i>	<i>c</i>
Agent 2	<i>a</i>	8/12	-3/3	-8/12
	<i>b</i>	-3/3	0/4	-3/3
	<i>c</i>	-8/-12	-3/3	8/12

Table 7: The stochastic penalty game (50%).

3 Reward estimation

In games with stochastic rewards, such as all the climbing game variants and the stochastic penalty game, it is difficult to distinguish between the two sources of variation in the observed reward for some action. It would be useful to have a protocol that allows 2 or more agents to select the same joint action repeatedly in order to build up a model for the stochastic reward distribution. This section describes a novel approach for achieving this.

The basic principle is that agents follow a common action selection policy that enables them to estimate the potential reward for each joint action. The action selection policy is based on the following idea: if an agent chooses an action at time i , then the agent is required to choose the same action at specific future time points. The only assumption that this approach makes is that all agents share the same global clock and that they follow a common protocol for defining sequences of time-slots.

3.1 Commitment sequences

A *commitment sequence* is some list of *time slots* $(t_1, t_2, \dots)$ for which an agent is committed to taking the same action. If two or more agents have the same protocol for defining these sequences, then the ensemble of agents is committed to selecting a single joint-action for every time point in the sequence. Although each agent does not know the action choices of the other agents, it can be certain that the observed rewards will be statistically stationary and represent unbiased samples for the reward distribution of *some* joint action. In order to allow an arbitrarily high number of joint actions and consequently commitment sequences to be considered as the agent learns, it is necessary that the sequences are finite *or* have an exponentially increasing time interval $\delta_i \equiv t_{i+1} - t_i$ between successive time slots. A sufficient condition is that

$\gamma\delta_{i+1} \geq \delta_i$ where $\gamma > 1$ for all $i > i_0$ (for some pre-defined constant i_0). In the results given here, sequences are infinite with $\gamma = 5/4$.

Here, the successive increments are generated by the function:

$$\delta_{i+1} = \left\lfloor \frac{c\delta_i + c - 2}{c - 1} \right\rfloor$$

where $c > 1$ is the increment factor and $\lfloor \cdot \rfloor$ indicates rounding down to an integer value. For example, if the increment factor is 5 (i.e. the increment ratio γ is $5/4$) the rule becomes: $\delta_{i+1} = \lfloor (5\delta_i + 3)/4 \rfloor$. The first such sequence starts with (1, 3, 6, 10, 15, 22, ...). The second sequence therefore starts at time slot 2. To prevent any 2 sequences from selecting the same time slot, each sequence excludes the existing ones from counting. Hence the second sequence starts with (2, 5, 9, 14, 20, 28, ...).

For time i suppose the agents chose actions $(a_1^i, a_2^i, \dots, a_m^i)$ (where m is the number of agents). Then an estimate of the value of this joint action is available as the average reward received during the part of the sequence that has been completed so far. Longer sequences provide more reliable estimates. Initially, we evaluate the simplest approach possible where the agents maintain a running reward average for all the active commitment sequences. Later, we will explore a method that takes into account the stochasticity in the rewards and the length of sequences within the framework of Gaussian estimation.

3.2 Action selection policy

Each agent must choose an action at the start of each sequence. A new sequence starts whenever no existing sequence is active in the current time slot. There are two obvious ways to select the new action: either explore (select the action randomly and uniformly) or exploit (select the action currently considered optimal). The simple approach used here is to choose randomly between exploration and exploitation for each sequence. In order to prefer longer sequences (more reliable estimates), we maintain statistics about the commitment sequences that are active. One such statistic is the length of the sequence until the current time point. Agents only consider a commitment sequence to yield a reliable reward estimate if its length becomes greater than a threshold value, N_{min} . In these experiments, N_{min} was set to 10.

For a 2-agent system, we chose the exploration probability p to be 0.9. As an exception, the first N_{init} sequences (where $N_{init} \geq 1$) must be exploratory to ensure that an exploitative action can be calculated. The algorithm followed is shown in Figure 1. In the results below, $N_{init} = 10$.

The exploit function simply returns the action that corresponds to the commitment sequence with the highest observed average reward among those whose current length is at least N_{min} .

```

if number of sequences  $< N_{init}$  then
    explore randomly and uniformly
else
    with probability  $p$  : explore randomly and uniformly
    with probability  $1 - p$  : exploit
end if

```

Figure 1: The average reward estimation algorithm.

3.3 Parameter analysis

In this section, we analyse the influence of the algorithm's parameters on the learning. These parameters are: the minimum number of commitment sequences that must have been started before the agents can exploit (N_{init}), the minimum length that a sequence must reach before it is considered for exploitation (N_{min}), the increment factor (c) and the exploration probability (p).

N_{init} was set to 10 in all our experiments. Since none of our experiments have less than 10 commitment sequences, N_{init} has no role in the learning other than to make sure that, upon selection of an exploitative action, there are *some* sequences among which to choose.

N_{min} affects the learning as follows: In experiments where only short commitment sequences are created (either short experiments or long ones with a small increment factor), learning performance improves by setting N_{min} to a reasonably high value since our confidence is higher for longer commitment sequences. In longer experiments, N_{min} plays no part in the learning other than to decrease learning performance if set too high. This is because it is possible that an agent never explores as no commitment sequences reach the required length. Figure 2 illustrates this effect. We have plotted the convergence probability of the learners in the stochastic climbing game after 1000 moves. Since no commitment sequences of length greater than 20 have been created, the learning performance decreases for values of $N_{min} > 20$.

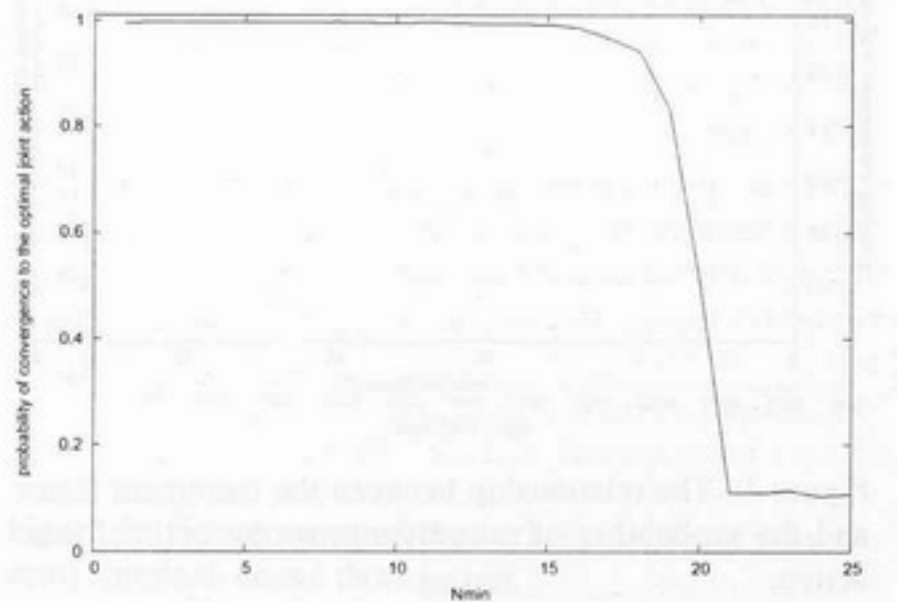


Figure 2: Number of successful experiments out 1000 runs for $N_{min} \in [1, 25]$.

The increment factor c defines how quickly a commitment sequence is visited again. Informally, the higher the value of the increment factor, the greater the time between two successive updates of a commitment sequence. Consequently, the total number of commitment sequences reduces monotonically with the increment factor for any length of the experiment. To illustrate this relationship, Figure 3 shows the total number of sequences for an experiment of 1000 moves with increment factor $c \in [2, 29]$.

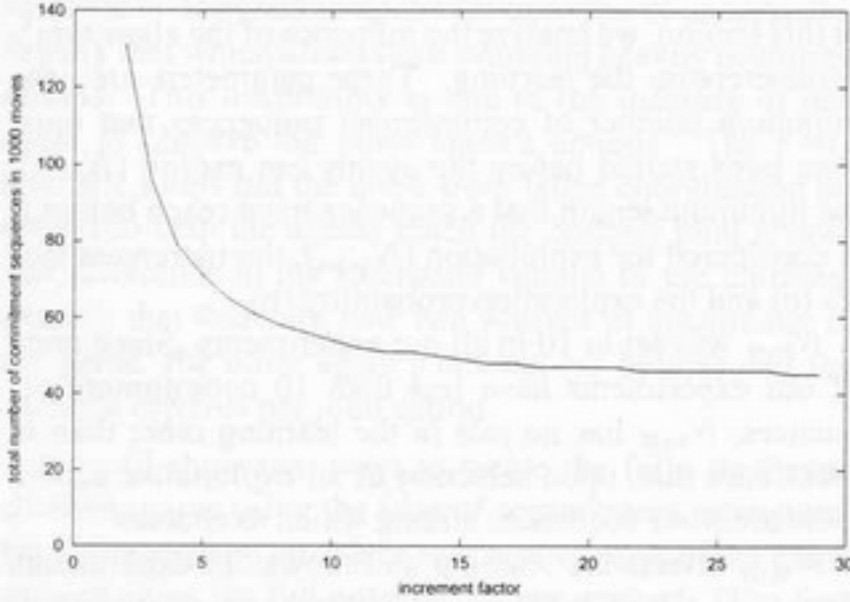


Figure 3: The relationship between the increment factor and the total number of sequences.

The performance of the learners for different increment factors is also plotted in Figure 4. This plot was generated for the stochastic climbing game with $N_{min} = 10$ and experiment length 1000.

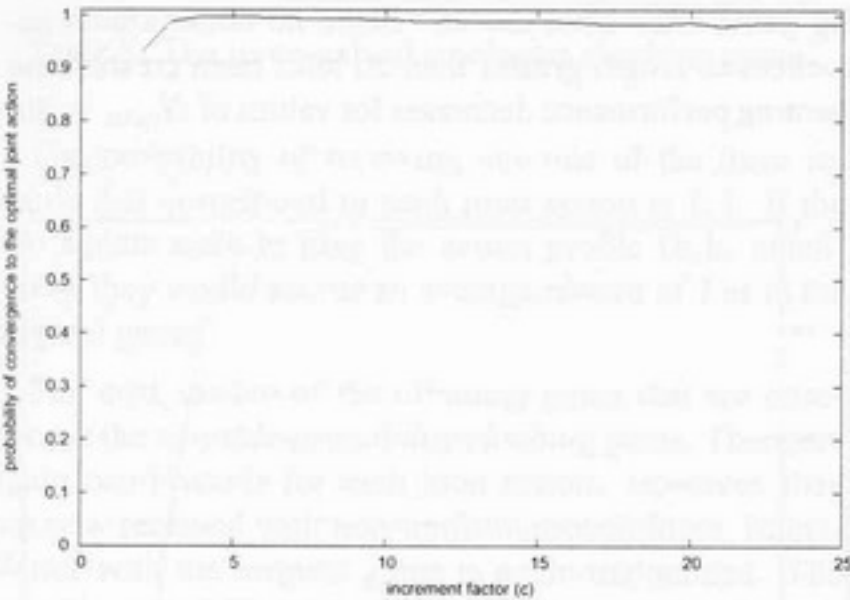


Figure 4: The relationship between the increment factor and the probability of convergence to the optimal joint action.

Finally, in order to understand the influence of the exploration probability p on the learning, we have plotted its effect on the learning performance in Figure 5. The

exploration probability has been varied from 0 (always exploit) to 1 (always explore), N_{min} and N_{init} were set to 10 and the experiment was 1000 moves long. This plot was created for the stochastic climbing game.

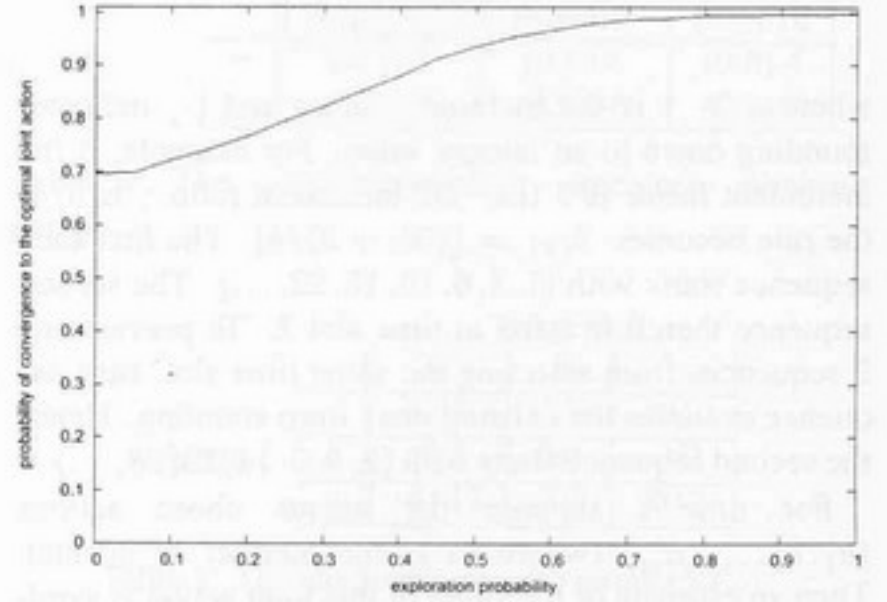


Figure 5: The relationship between the exploration probability and the probability of convergence to the optimal joint action.

From Figure 5, we can see that 100% exploration is optimal in this case. However, we have identified some games where one agent's exploitative behaviour can help the other agent to learn. If performance during learning is an issue, a low exploration probability is also desirable. In our experimental evaluation, the exploration probability will be 0.9.

3.4 Experimental results

This section contains the experimental results for the basic approach in all three versions of the climbing game and in the stochastic penalty game. As before, we repeated each experiment 1000 times to obtain high confidence in the results. The number of moves was varied from 500 to 3000 and the parameters N_{init} and N_{min} were both set to 10. In all experiments, we chose $\gamma = 5/4$. The results for the climbing game variants are plotted in Figure 6.

In Figure 6, SCG stands for Stochastic Climbing Game, TVSCG stands for Three-Valued Stochastic Climbing game and VPSCG stands for Variable-Probability Stochastic Climbing Game. From Figure 6, we can see that the probability of convergence to optimal eventually reaches over 90% for all cases, with some reaching over 95% even for relatively short experiments. The most difficult game is the three-valued climbing game as there is more variance in the rewards that correspond to each joint action.

The stochastic penalty game is solved much quicker by this approach. The probability of convergence to the optimal joint action reaches over 95% even for very short

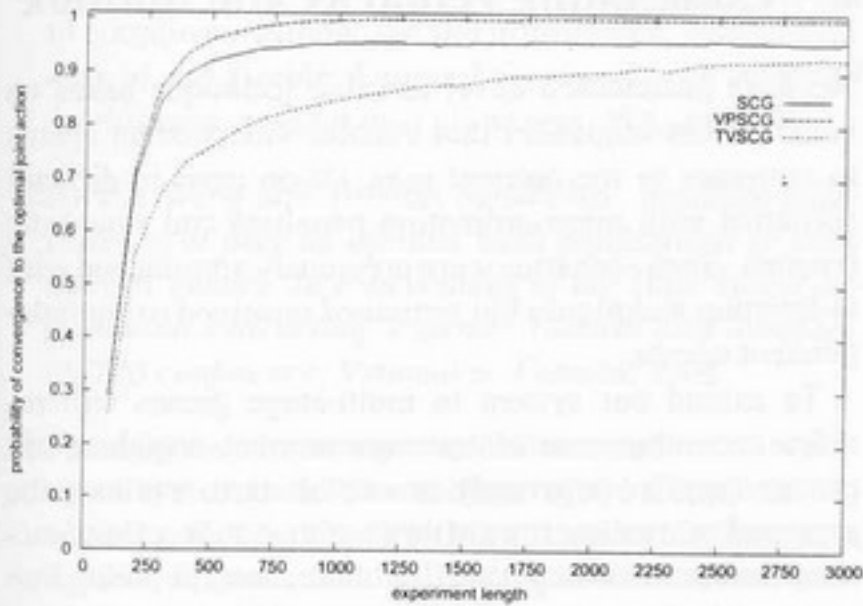


Figure 6: Convergence of the basic approach on the three games.

experiments (500 moves). This is because the variance in the rewards is small and the method is impervious to the existence of multiple optimal joint actions. Regardless of how many optimal joint actions there are in the game, one will always have a higher estimate than others¹ so that choosing an action for exploitation is not affected by the number of optimal joint actions.

4 Variance-sensitive approach

The basic approach using averaging performs fairly well. However, there are cases (e.g. the three-valued stochastic climbing game) where convergence only reaches above 90% for very long experiments. This section outlines an extended approach using a Gaussian estimator that improves both the convergence speed and the probability of convergence to the optimal joint action.

4.1 Finding the exploitative action

As in the basic approach, if the agents chose actions $(a_1^i, a_2^i, \dots, a_m^i)$ (where m is the number of agents) at time i then an estimate of the value of this joint action is available as the average reward received so far for this sequence. For the extended approach, we attempt to reason about the *true* expected reward. To do this, we must make some assumptions about the possible form of the reward for each joint action, e.g. that it must have finite variance.

Here we use a Gaussian model and estimate its mean and variance from the observations. If n rewards are observed with empirical average m and sum of squares S , we obtain estimates for the population mean μ and its variance σ_μ (estimates of a quantity x are denoted by $\hat{x}$):

$$\hat{\mu} = m$$

¹ If multiple estimates are equal, the agents will choose the one that corresponds to a longer commitment sequence.

$$\hat{\sigma}_\mu^2 = \frac{S + \sigma_0^2}{n^2} - \frac{m^2}{n}$$

σ_0 is a parameter to the algorithm and should be based on the expected variance of rewards in the game. In order to prefer longer sequences (more reliable estimates), a pessimistic estimate $\hat{\mu} - N_\sigma \hat{\sigma}$ is used to provide a lower bound on the expected return for each sequence. At any given time, the exploitative behaviour for an agent is to choose the action corresponding to the sequence with the greatest lower bound. Large values of N_σ reduce the risk that an optimistic bias in the reward estimate from a short sequence will affect the choice of action. However, smaller values may give faster initial learning. In the results below, $N_\sigma = 4$.

4.2 Exploration policy

In the variance-sensitive approach, the agents choose randomly between exploration and exploitation for each sequence. For a 2-agent system, we chose the exploration probability to be 0.9 as before. We have maintained the N_{init} parameter ($N_{init} \geq 1$) but have eliminated N_{min} . We simply allow N_{init} commitment sequences to start and then only use the variance-sensitive estimator to find the exploitative action. In the results below, $N_{init} = 10$.

4.3 Experimental results

Figure 7 depicts the convergence performance of the variance-sensitive approach for the three variants of the climbing game. In these experiments, σ_0 was set to 50 for the three-valued stochastic climbing game and to 10 for the rest. In Figure 7, SCG, VPSCG and TVSCG stand for Stochastic Climbing Game and its Variable-Probability and Three-Valued variants.

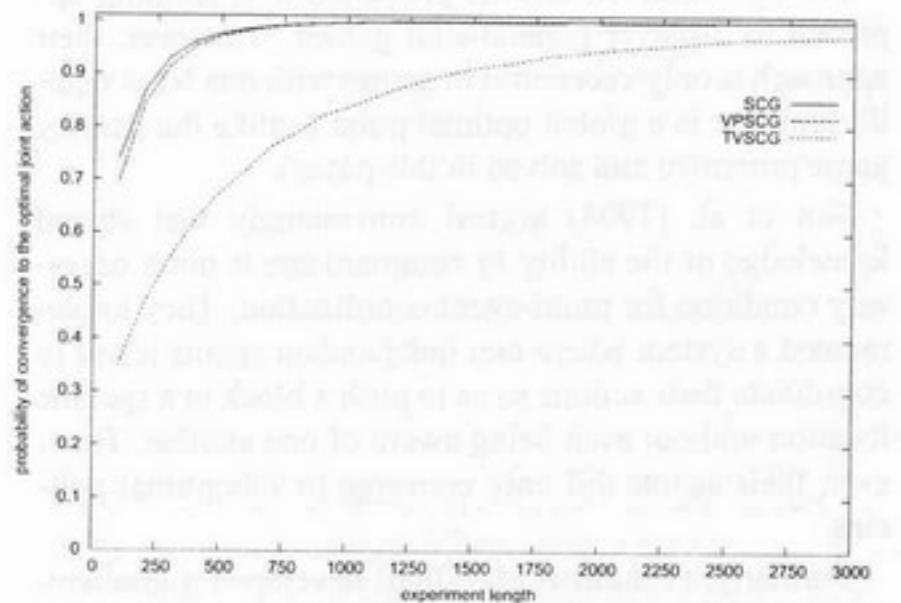


Figure 7: Convergence of the variance-sensitive (Gaussian) approach on the three games.

As we can see from Figure 7, the variance-sensitive approach outperforms the averaging approach in all cases. In fact, the probability of convergence to the optimal

joint action consistently reaches over 98%. For the three-valued game, we chose $\sigma_0 = 50$ because the variance in the stochastic rewards is higher. The probability of convergence to the optimal joint action in this game reached over 90% for longer experiments i.e. after 2000 moves.

Finally, the stochastic penalty game was once again much easier to solve. Here, the probability of convergence to the optimal joint action exceeded 95% for 200 moves and reached 100% after only 800 moves. This clearly illustrates the ability of the variance-sensitive approach to avoid the problem of choosing between multiple optimal joint actions.

5 Related work

There are two main paradigms for the learning of coordination, one using independent agents and another using joint-action learners. While joint-action learners are able to observe one another's actions, their independent counterparts can not. Therefore, approaches using independent learners (such as ours) are more general and more universally applicable.

Claus and Boutilier (1998) used joint-action learners and fictitious play in their approach to learning coordination in cooperative multi-agent systems but reported a failure to solve problems where miscoordination is heavily penalised. Later, Boutilier (1999) developed an extension to value iteration that allowed each agent to reason explicitly about the state of coordination, i.e. whether the group of agents are in a coordinated or non-coordinated state. Wang and Sandholm (2002) developed a learning algorithm for joint-action learners that provably converges towards the optimal Nash equilibrium.

Hu and Wellman (2002) presented a Q-learning approach to 2-player general-sum games. However, their approach is only successful in games with one Nash equilibrium that is a global optimal point (unlike the penalty game presented and solved in this paper).

Sen et al. (1994) argued convincingly that shared knowledge or the ability to communicate is not a necessary condition for multi-agent coordination. They implemented a system where two independent agents learnt to coordinate their actions so as to push a block to a specific location without even being aware of one another. However, their agents did only converge to suboptimal policies.

Similarly, Peshkin et al. (2000) developed a gradient-descent policy-search algorithm for cooperative multi-agent domains that is guaranteed to find a local optimum in the space of factored policies but may not always find an optimal Nash equilibrium.

Nowé et al. (2001) used social agents that employ a periodical policy to tackle learning in single-stage 2-player games. In contrast to our approach, these agents rely on communication to achieve coordination.

6 Concluding remarks and outlook

We have presented a novel learning technique based on commitment sequences that enables independent agents to converge to the optimal joint action even in difficult scenarios with miscoordination penalties and stochastic rewards. Such scenarios were previously approached with Q-learning techniques but remained unsolved using independent agents.

To extend our system to multi-stage games will require a combination of the commitment sequence approach (applied separately in each state to evaluate the expected immediate reward) with state-action value functions that estimate expected discounted returns taking into account the state-transition function. Value functions of this kind may be associated with sequence-action pairs and/or state-action pairs. A hybrid method that makes use of the FMQ heuristic (Kapetanakis and Kudenko, 2002) will be considered first.

References

- C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 478–485, 1999.
- Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, 1998.
- Drew Fudenberg and David K. Levine. *The Theory of Learning in Games*. MIT Press, Cambridge, MA, 1998.
- Junling Hu and Michael P. Wellman. Multiagent q-learning. *Machine Learning Research*, 2002.
- Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI'02)*, 2002.
- Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference in Machine Learning*, 2000.
- Ann Nowé, Johan Parent, and Katja Verbeeck. Social agents playing a periodical policy. In *Proceedings of the 12th European Conference on Machine Learning*, Freiburg, Germany, 2001.
- Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Kaelbling. Learning to cooperate via policy search. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, Seattle, WA, 1994.

Xiaofeng Wang and Tuomas Sandholm. Reinforcement learning to play an optimal nash equilibrium in team markov games. In *Proceedings of the 16th Neural Information Processing Systems: Natural and Synthetic (NIPS) conference*, Vancouver, Canada, 2002.

Gerhard Weiss. Learning to coordinate actions in multi-agent systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, volume 1, pages 311–316. Morgan Kaufmann Publ., 1993.

Advice-Exchange between Evolutionary Algorithms and Reinforcement Learning Agents: Experiments in the Pursuit Domain

Luís Nunes

^{*}ISCTE/FEUP/LIACC-NIAD&R

ISCTE, Av. Forças Armadas, 1649-026 Lisbon, Portugal

Luis.Nunes@iscte.pt

Eugénio Oliveira

[†]FEUP/LIACC-NIAD&R

FEUP, Av. Dr. Roberto Frias, 4200-465, Porto, Portugal

eco@fe.up.pt

Abstract

This research aims at studying the effects of exchanging information during the learning process in *Multiagent Systems*. The concept of *advice-exchange*, introduced in (Nunes and Oliveira, 2002b), consists in enabling an agent to request extra feedback, in the form of episodic advice, from other agents that are solving similar problems. The work that was previously focused on the exchange of information between agents that were solving detached problems is now concerned with groups of learning-agents that share the same environment. This change added new difficulties to the task. The experiments reported below were conducted to detect the causes and correct the shortcomings that emerged when moving from environments where agents worked in detached problems to those where agents are interacting in the same environment. New concepts, such as *self confidence*, *trust* and *advisor preference* are introduced in this text.

1 Introduction

The question that drives this research is: “(How) can learning agents cooperate during the learning process?”. This question is broad, and parts of it were already addressed by other authors. For this reason, as well as for others that concern the applicability of the solution and the authors’ research interests, we have drawn some restrictions to the type of problems and learning agents used in the experiments. In this work we will focus on agents with simple communication skills that are solving problems with the same structure, but possibly different local dynamics. The agents have only partial knowledge of the state-space and each learns using an algorithm of its choice, unknown to others. Agents may exchange information with peers that are cooperating to solve the same problem (its *partners*), or with agents that are solving similar problems in other environments with the same structure. For the moment we only consider non-malicious agents that always try to help others to the best of their knowledge. The environment in which they act does not provide supervision information, only a measure of the quality of each action (or sequence of actions).

The approach taken in this work aims at improving the individual and global learning performance by providing learning-agents (hereafter referred to as “agents” for the sake of simplicity) with more feedback than the environment can give them. Agents, during their learning process, gather expertise in how to deal with the particular problem they are facing as well as information about the quality of their own actions. The knowledge of the performance achieved by others in similar problems can be used by an agent to detect how good its solution is. Also, the

knowledge of how an agent would react to a given state of the environment may contain information that can improve the learning results of another agent, if the structure of the environment is similar enough.

The groups of agents referred in this paper are heterogeneous in the sense that they may use different learning techniques. The interest in cooperation amongst groups of heterogeneous agents lies in having different exploration techniques that may reach different solutions, or at least not become trapped in the same local-maxima, thus being able, by exchanging information, to help each other during the learning process. The main difficulty in having heterogeneous groups is that internal parameters cannot be exchanged, because agents are not aware of which learning algorithm their peers are using. This fact restricts communication capabilities to knowledge that has meaning outside the agents’ internal context. The simplest types of data with these characteristics are the ones generated by the environment itself, which all agents must understand in order to sense and act. This data can be either the state of environment, actions, or values of the quality function that agents are attempting to maximize. The exchange of other types of data, such as “trust agent X”, was also found useful and it requires that all agents share a (small) common vocabulary. We will further assume that the agents are working in domains in which the problem has the same structure and is presented in the same way to the agent, i.e. the meaning and order of the variables that compose both state and actions, are known to be the same for all environments used in an experiment. Nevertheless, different environments in the same experiment may reward the same state-action pair differently and/or have different state transition dynamics.

The key questions of this research are: how to select, incorporate, and benefit-from all the available information without re-processing all the data gathered by each of an agent's peers (which is time and band-width consuming). The objective is to gain knowledge based on the synthesis that was done by others without exchanging the entire solution and adopting it blindly (which in some circumstances may even be a very poor option, if the dynamics of the advisee's environment are different enough from the advisor's, or if the agents are using different learning algorithms). To the best of the authors' knowledge there is no work that specifically addresses the problem of exchange of information during learning in heterogeneous groups of agents that use *reinforcement-based* learning algorithms. The term "*reinforcement-based*" is used here to refer to learning algorithms that are "*based on a scalar quality/utility feedback*", as opposed to *supervised learning*, which requires a desired response as feedback.

The work reported in this paper focuses on exchanging episodic information consisting of state-action pairs and statistics on the performance of other agents. An agent requests information whenever it perceives that the current performance is relatively low when compared to that of its peers or when the agent is in a state for which the choice of the following action is unclear.

The amount of information exchanged is controlled by a *self-confidence* parameter in each agent. This parameter is used in the decision of whether or not an agent should get advice for each new state of the problem. The selection of where to collect this information is based on knowledge regarding other agents' results and on the development of *trust* relationships between an agent and its peers. The exchanged information is used either as a desired output for *supervised learning* or as a different source of reinforcement in a process that was labelled "*learning from a teacher*" by Clouse (1997).

The experiments reported here aim at testing solutions for the difficulties reported in (Nunes and Oliveira, 2002a) concerning the disturbances on the learning process caused by the interaction between agents and the exchange of advice during learning in social environments. In this work we compare the performance of four groups of agents in several versions of a task (the pursuit problem), thus providing the basis for an assessment of the advantages and drawbacks of *advice-exchange*. The discussion, based on these results, should provide clues on how techniques of this type can be improved to share information between agents efficiently. The four groups of agents used in the experiments are: Individual; Homogeneous cooperative groups; Heterogeneous cooperative groups and Heterogeneous cooperative groups plus Heuristic experts. The experts are a group of handcrafted agents that use an optimal (although individualistic) heuristic. This group of agents was introduced in the experiments to estimate the performance that can be obtained by agents that use optimal individual actions and verify the gain that can be achieved by the agents when they have near-optimal

teachers.

In the following section the reader can find a summarized review of the related work. Section 3 presents the learning algorithms used by the agents. In section 4 the *advice-exchange* algorithm is explained, followed by the description of the experiments (section 5). In section 6 the results are presented and discussed, and finally, the last section presents the conclusions and some guidelines for future work.

2 Related Work

Quite a few researchers have contributed to answer the question stated above since the early 90's, especially in what concerns the transfer of learned information between *Q-Learning* agents (QL-agents). Without being exhaustive, due to the extent of the research that covers this and related topics, and trying simply to contextualize the work that will be presented below, the next paragraphs summarize the main contributions in the past decade. The work on information exchange between QL-agents, started in the early nineties with the contributions of Whitehead (1991); Clouse and Utgoff (1991, 1992); Lin (1992) and Tan (1993).

Whitehead made several experiments with a cooperative learning architecture labelled *Learning By Watching* (LBW). In this architecture, the agent learns by watching its peers' behavior (which is equivalent to sharing series of state-action-quality triplets). In LBW an agent will use other agents' episodes as if they were its own. This work proves that the complexity of the search mechanisms of LBW is inferior to that of standard *Q-Learning* for an important class of state-spaces.

The results presented in (Clouse and Utgoff, 1991, 1992) are reviewed and expanded in Clouse's Ph.D. thesis (Clouse, 1997). This important contribution reports the results of a technique labelled *Ask for Help* (AH), in which QL-agents learn by asking their peers for suggestions and then performing the suggested actions. This was labelled *learning from a teacher*. Clouse proposes two different approaches for the problem of deciding when to ask for help: in the first approach the advisee asks for help randomly for a given percentage of the actions it has to take; in a second approach the agent asks for help only when it is "confused" about what action to take next. Confusion is defined as having similar quality estimates for all possible actions that the agent can perform at a given stage.

Lin (1992) uses an expert trainer to teach lessons to a QL-agent that is starting its own training. He experimented with several architectures, the most interesting for the current subject are: AHCON-T (*Connectionist Adaptive Heuristic Critic with Teaching*) and QCON-T (*Connectionist Q-Learning with Teaching*). These architectures record and replay both "taught" and "learned lessons", i.e. sequences of actions that led to a success.

Some of these sequences (the “taught lessons”) are given by an expert trainer (human or automatic), whilst others are recorded by the student agent as it explores the problem (the “learned lessons”). The student only replays the actions and observes the reward. This work reports that the “*advantages of teaching should become more significant as the learning task gets more difficult*” (Lin, 1992), (section 6.4). Results in variants of the maze problem show that teaching does improve learning performance in the harder task, although it seems to have no effect on the performance of the agents on the easier task.

Tan (1993) addressed the problem of exchanging information between QL-agents during the learning process. This work reports the results of sharing several types of information amongst agents that are working in the pursuit (predator-prey) problem. Experiments were conducted in which QL-agents shared policies (internal solution parameters), episodes (series of state, action, quality triplets), and sensation (observed states). Although the experiments use solely *Q-Learning* in the pursuit domain, the author believes that: “*conclusions can be applied to cooperation among autonomous learning agents in general*”, (Tan, 1993), (section 8). Tan concludes that “*additional sensation from another agent is beneficial if it can be used efficiently*”, (Tan, 1993) (in Abstract). The results reported by Tan also appear to point to the conclusion that sharing episodes with peers is beneficial and can lead to a performance similar to that obtained by sharing policies. Sharing episodes volunteered by an expert agent leads to the best scores in the presented tests, significantly outperforming most of the other strategies in the experiments.

Training of novice Q-learning agents by *implicit imitation* of pre-trained experts was studied and successfully applied by Price and Boutilier (1999, 2000). This work has some very interesting characteristics: the student agent has no knowledge of the actions done by the expert, it can only observe its state transitions; there is no explicit communication between the expert and the student and the goals and actions of both agents can be different. The student agent uses an augmented Bellman equation to calculate the utility of each state. This equation incorporates the information regarding the state transitions made by the expert. The technique was tested in several maze problems with very good results.

The “*Agent Teaching Agent*” framework (Sen and Kar, 2002) relies on an expert to teach a student agent a certain concept by selecting the training examples to use on each epoch based on the errors the student has made in the previous epochs. In this work student and teacher use different learning algorithms.

3 Agents’ Learning Algorithms

In this section we will briefly describe the variations on standard learning algorithms that were used in the experiments. Due to space constraints a full presenta-

tion of Q-Learning (QL) (Watkins and Dayan, 1992) and Backpropagation (BP) (Rumelhart et al., 1986) is omitted. The version of QL used is the standard one-level Q-Learning with Boltzmann exploration and decaying temperature, described in (Tan, 1993; Kaelbling et al., 1996). Backpropagation is also a standard version with decaying learning rate. A fully detailed description, of the learning algorithms used along with their parameters can be found in (Nunes and Oliveira, 2003).

The variant of *Evolutionary Algorithms* (EAs) (Holland, 1975; Koza, 1992) used in these experiments deserves further attention. EAs are a well known learning technique, with biological inspiration where the solution parameters are interpreted as a specimen (or phenotype), and its performance in a given problem as its fitness. An EA controls a population of specimens. In each generation-loop of the algorithm the fitness of all specimens in the population is evaluated and the best are selected for breeding. Each specimen is encoded in a genotype. The encoding is usually a transformation that serves the purpose of translating a specimen into a set of symbols, for which mutation and crossover have some desired effect. This effect is usually dependent on the representations of the problem and hypothesis. The processes of encoding, translation and creation of specimens can provide a bias to these learning algorithms, by using some knowledge of the structure of the problem or of the subset of hypothesis that is more suitable to solve it. The best strategies for the encoding of genotypes, selection of the best specimens, mutation and crossover, for different types of problems, have been source of debate since the introduction of this learning technique. We shall not dwell in this debate, but the interested reader can find two interesting approaches and further references on the subject of combining EAs and BP in (Salustowicz, 1995) and (Yao, 1999).

The variant of EAs used in these experiments works mainly based on mutation of parameters and a simple crossover strategy. This variant was based on the work presented in (Glickman and Sycara, 1999) that achieved good results in the task of navigation in a difficult maze problem. The main characteristics of this variant are the following: there is no encoding of specimens to genotypes, the genotype is simply the set of real-valued matrixes that constitute an Artificial Neural Network (ANN) of fixed size; all specimens are ANNs of the same size; the selection strategy is elitist, keeping a certain number of the best specimens untouched for the following generation; the remainder of the population is generated by mutation and crossover of randomly chosen specimens selected amongst the ones that achieved the best performances in the previous generation. Contrary to (Glickman and Sycara, 1999) this variant does not use tournament selection. Mutation is done by disturbing all the values of the parameters with random noise with null average, and normal distribution. A mutation-rate parameter (m) controls the variance of the noise added to a

neural network's weight when it is mutated. This parameter is decayed during training according to $m_{t+1} = m_t * (t_F - t) / t_F$, until it reaches a, pre-established, minimum value. In the previous equation t represents the current epoch and t_F is a parameter of the algorithm that determines the epoch at which the learning rate should drop to zero. The crossover strategy consists on choosing two parents from the selected pool and copy randomly from each of the parents sub-sets of parameters (in this case the allowed subsets are ANN's sub-trees). Crossover is scarcely used because the approaches that were proved effective imply complex encoding and decoding schemes and their effectiveness is highly dependent on the problem being addressed. The crossover procedure applied in this case requires that the ANNs have the same structure. Crossover is done simply by choosing each unit (and all its incoming weights) randomly from one of the two selected parents.

Each agent contains a population of specimens, and exchanges its active specimen after a given number of epochs (3 in this case). After the evaluation of all the specimens, the selection process takes place, followed by mutation and crossover of the selected specimens to generate a population of the same size as the previous one.

4 Advice-exchange

The first version of this algorithm was introduced in (Nunes and Oliveira, 2002b) where it was evaluated in solving a simplified traffic-control problem. In these experiments, agents were working on similar but detached problems. Recent work on this subject can also be found in (Nunes and Oliveira, 2002a, 2003). *Advice-exchange* consists in presenting the current state of the environment, as seen by a given agent (the advisee), to another agent that has better performance in similar problems (the advisor) and in using the action proposed by the advisor as desired response for a form of *supervised learning*. This will enable an agent to learn, both, from the environment's reinforcement and from its more successful partners, which will act as teachers/critics.

The algorithm, described in table 1 is based on the knowledge of two quantities for all peers involved in *advice-exchange*: the *relative current average quality* (cq) and the *relative best average quality* (bq). These values are calculated as follows: let $aq_{i,t}$ be the average quality of the actions of agent i in epoch t , then $cq_{i,t+1} = (1 - \alpha)aq_{i,t} + \alpha cq_{i,t}$, with $\alpha \in]0, 1[$, and $bq_{i,t+1} = \max(aq_{i,t}, \beta bq_{i,t})$, with $\beta \in]0, 1[$. These measures of the quality of the actions of an agent proved to be preferable to the absolute, current and best average quality values because they are less subject to oscillation due to spurious results.

The *self-confidence* parameter is used in the decision of whether or not to get advice. This parameter was included to deal with the problem of the sudden withdrawal

Table 1: Summary of the *advice-exchange* algorithm in the perspective of the advisee agent i .

While not train finished
Broadcast:
cq_i : relative current quality
bq_i : relative best quality, for $i \in \text{Agents}$
While not epoch finished
1. Get state s for evaluation.
2. If <i>best quality not good enough</i> or <i>current quality not good enough</i> or <i>uncertain/confused</i> concerning state s
2.1 Select the best advisor (k).
2.2 Request advice to agent k for state s
2.3 Agent k : process request of agent i producing advised action (a)
2.4 Process advised action (a)
3. Evaluate state s and produce response (r)
4. Receive reward for action taken
End epoch loop
Update $cq_i, bq_i, trust_{ij}$ and sc_i (<i>self-confidence</i>).
End train loop

of advice and to prevent agents from getting unnecessary advice. When an agent is deciding if it needs advice it will compare its own quality statistics (cq and bq) with those of other peers. The *self-confidence* parameter is used as a weight, applied to its own quality values. If *self-confidence* is higher than 1.0 (meaning that in the recent past the agent was frequently one of the most successful) it will overrate its own performance. If, on the other hand, it is lower than 1.0 (meaning that in the recent past the agent was frequently one of the least successful) it will underrate its own performance.

This parameter can compensate the oscillations in performance, typical of dynamic domains. An agent that has recently been successful will not ask for advice (and thus, change its strategy) because of one (or a few) bad epoch(s). The time it takes for self-confidence to decay to values lower than 1.0 allows an agent to recover from spurious bad results and only consider getting advice if the performance drops consistently for several epochs. Another problem that was mitigated by the introduction of *self-confidence* was that of sudden withdrawal of advice. When a low-performance agent is learning from a peer that obtains better results, the training ends when their performances reach the same level. It was observed that after advice was withdrawn some agents experienced a sudden drop in performance and had to restart the *advice-exchange* process. The analysis of this behaviour, allowed us to conclude that the agents, after just a few epochs, had lost some of the capabilities that they were recently taught. The introduction of *self-confidence* gives the agent some time to solidify the lessons taught by others because it takes several epochs for a low-performance

agent to regain its self-confidence when its performance improves.

Several variants of the update of *self-confidence* parameters were tested. In general the self-confidence (sc_i) for agent i is updated as follows: let hb_i be the number of agents for which $bq_j < bq_i$, for all agents $j \neq i$; hc_i the number of agents for which $cq_j < cq_i$, for all agents $j \neq i$; nc the total number of agents in the environment and k the agent with best cq . Then,

$$sc_{i,t+1} = \begin{cases} sc_{down} * sc_{i,t} & : C1 \text{ is true} \\ sc_{up} * sc_{i,t} & : C2 \text{ is true} \end{cases} \quad (1)$$

where $C1 = (hb_i > \gamma_{down}nc \vee cb_i > \gamma_{down}nc) \wedge cq_i < \rho_{down}cq_k$; $C2 = hb_i < \gamma_{up}nc \vee cb_i < \gamma_{up}nc \vee cq_i > \rho_{up}cq_k$; $sc_{down} \in]0, 1[$ and $sc_{up} \in]1, 2[$. The value of sc_i is not allowed to be greater than sc_{max} or smaller than sc_{min} taking up those values if the update has surpassed the maximum or minimum values, respectively. The values of γ_{up} , γ_{down} , ρ_{up} , ρ_{down} , sc_{min} and sc_{max} were, respectively (1/2, 3/4, 0.9, 0.8, 0.5, 2.0).

The *trust* parameters are used in the selection of the most suitable advisor for a given agent. It was observed that in certain situations the agent with the best quality statistics was not the best advisor for all its peers. The performance as an advisor depends not only on the performance of an agent, but also on other factors, such as: similarity of the dynamics of the advisor and advisee's environment and capability of the advisee to learn the advised behaviour adequately. The introduction of the *trust* parameter, as a measure of "how much has the advice from agent j helped to improve a given agent's performance in the recent past", helps an agent to seek the most appropriate advisor for its own situation. The update of the *trust* parameters is done in a similar way as for the *self-confidence*. An agent will increase or decrease the trust on a given peer proportionally to the number of advice it has received from this peer in the past epoch. Trust will be increased if the relative current quality of the advisee increased in the last epoch ($cq_{i,t} > cq_{i,t-1}$) and decreased otherwise.

Trust in a certain advisor can also be influenced by a partner. When a partner changes advisors or when an advisor has proved successful an agent issues a message to its own partner telling it to trust the partner of its own advisor. This enables agents to learn joint strategies used by other teams by influencing each of their partners to seek advice with a different member of a successful team. The update of the trust agent i has in agent j is simply: $trust_{ij,t+1} = trust_{up} * trust_{ij,t}$, where $trust_{up} > 1.0$.

The decision, made by a given agent i , on whether or not to get advice is based on three conditions:

1. Best quality not good enough: $sc_i bq_i < bq_j$, for some $j \neq i$,
2. Current quality not good enough: $sc_i cq_i < cq_j$, for some $j \neq i$,

3. Confused about current state: An agent is said to be confused concerning a given state if its N best rated options differ in their probability of been chosen by less than a given number ϵ . In these experiments the values used were $N = 4$, $\epsilon = 0.1$.

An agent will request advice if any of these conditions is true. If an agent uses EAs it will not get advice in the first epoch in which a specimen is being tried. This prevents it from getting advice based on the performance achieved by other specimens. When an agent (i) decides to request advice it selects the peer with the highest product $cq_j * trust_{ij}$, $j \neq i$, and sends its current state to the selected agent. An agent will give preference of choice to the last advisor when this agent's score is similar to the best. By similar, we mean that the quality-trust product ($cq_j * trust_{ij}$) is lower by less than, for instance, 20% of the same value for the agent with the highest quality-trust product. The introduction of the preference given to the last advisor was necessary to avoid agents from exchanging advisor too frequently. This problem of frequent advisor switch led to serious disturbances of the learning process because advisees were unable to concentrate in learning one strategy and in some cases their performance dropped sharply when obtaining conflicting advice.

If the advisor is an EA-agent it will reload the weights used in the epoch when the best score was achieved and run the state provided by the advisee as if it was experiencing it in order to produce its best response to the request. If the selected peer is a QL-agent it will run the state with the quality-table that produced the best result in a previous epoch and without using Boltzmann exploration in the action-selection process. This way it will choose the action with highest quality according to the values used in the epoch in which it has achieved the best performance. The process of incorporating the received knowledge in an agent's current hypothesis is also different for both types of agents. When the advisee is an EA-agent it will backpropagate the advice as desired response using standard online *Backpropagation* on the weights of the ANN. The selection process will work on the specimens after they have been changed by the *Backpropagation* algorithm, passing the knowledge acquired by *supervised-learning* to the next generation. This can be interpreted as *Lamarckian* learning.

When the advisee is a QL-agent it will give a bonus of b_{up} to the advised action and of b_{down} to all other actions starting at that state, with: $b_{up} > 0$, $b_{down} < 0$ and $b_{up} \approx 10b_{down}$. A similar technique, labelled *Biasing-Binary*, that uses the same absolute quantity both for positive and negative feedback is reported in (Whitehead, 1991).

It was observed that EA-agents took longer to incorporate the advised information into their current hypothesis than QL-agents. This is due to the particular characteristics of the learning algorithm and the fact that an EA-agent evaluates specimens in sequence. To overcome this problem it was necessary to store advice, which can be replayed before the beginning of a training epoch. The

advice given during a training epoch is stored after being used, up to a limit of n_{stored} patterns (in this case we have used $n_{stored} = 450$). When the storage-space is filled, incoming advice will replace the oldest advice stored. At the beginning of a training epoch, if the last advisor is still the one that has the highest trust-quality product and the EA-agent still requires advice, the advice is replayed to the specimen in evaluation prior to the taking of any actions. If the last advisor has lost the trust of the advisee the stored advice is thrown away. The next time an advisor is chosen the specimen that receives advice is re-initialized before replaying the advice. Replaying advice consists in performing *backpropagation* using the stored advice as desired response. This process is repeated until a given number of epochs is reached (usually 100) or the *Backpropagation* error falls below a certain threshold. The use of *advice-replay*, although it allows to incorporate knowledge gathered by *advice-exchange* more rapidly, is still under study because it has the unwanted side-effect of reducing the diversity of the specimens by possibly pre-training an entire population with a similar set of examples. Another disadvantage of this approach is that online *backpropagation* is a time-consuming procedure, and training time may increase considerably when using this approach.

5 Experiments

The experiments this work refers to were conducted in the Pursuit Problem (a.k.a. Predator-Prey Problem).

This problem was first introduced in (Benda et al., 1985), although the version presented here has been inspired on several variations of this environment presented in (Tan, 1993) and (Haynes et al., 1995).

The problem faced by the predator consists in learning to catch a prey in a square arena. A predator is said to have caught the prey if at the end of a given turn it occupies the same position as the prey. The arena is toroidal (both in the horizontal and vertical directions), and contains two agents (predators) and one prey. At the beginning of each turn all agents receive the state of the environment which consists solely on the relative position of the prey. The toroidal shape of the grid is taken into account when computing the relative position of the prey as well as when moving and the predator receives as input the vector with smallest norm from the several possible vectors between its position and the prey's in a toroidal world. The accuracy of the state received by the predator depends on its visual range. The predator perceives the correct position of the prey up to a limit defined by the visual range parameter (the value of this parameter in these experiments was 3). If the prey is at a distance greater than the visual range its relative position is disturbed (by Gaussian noise with null average) proportionally to the distance between predator and prey. The vision field is toroidal in the same way movement is.

Each predator has to choose between nine possible actions in each turn, i.e. it chooses either to move in one of the eight possible directions (four orthogonal and four diagonal), or to remain in its current position. Each predator moves one step in one direction in each turn. The prey moves before the predators. To decide in which direction to move, the prey detects the presence of the closest predator (closest in the sense of being at the shortest Euclidean distance) and moves in the opposite direction. If there is more than one predator at the same distance one of the predators is picked randomly and the prey moves away from it regardless if it is approaching the other predator(s). The prey moves only nine out of each ten turns. Any two units can occupy the same cell simultaneously at any time.

In each turn predators are given a reward that has both an individual and a global component. The individual component is based on their distance to the prey (d). This reward is equal to 1.0, if the predator has captured the prey or $(1.0 - d/d_{max})/10.0$ otherwise. The constant max_d represents the maximum distance between any two positions in the grid world. The global component of the reward is calculated by multiplying the partner's reward by a given number g , with $g < 1$. In these experiments we used $g = 0.25$. After a successful catch the predator that caught the prey is randomly relocated in the arena.

This version of the pursuit problem does not involve either explicit cooperation or competition between the predators, although cooperative behaviour has been detected in some experiments as will be explained below.

5.1 Agents and scenarios

Agents in these experiments use either *Q-Learning* (QL) or *Evolutionary Algorithms* (EAs) to learn the required skill. Heuristic agents perform a fixed (hand-coded) policy that always tries to reduce the distance to the prey as much as possible. These agents do not request advice, but they always reply to advice requests from other agents.

The scenarios used in these experiments are the following:

1. Individual: Four arenas, each with two predators and one prey. In each arena all predators use the same learning algorithm and they do not exchange any information with their peers.
2. Social Heterogeneous: Four arenas, each with two predators and one prey. In each arena all predators use the same learning algorithm and advice may be requested by a predator to any of its seven peers in the same or other arenas. Two arenas have EA-agents, the other two have QL-agents.
3. Social Heuristic: Similar to the Social scenario but with an extra arena where two Heuristic agents are performing the same task (hunting one prey) and may also be chosen as advisors.

4. Social Homogeneous: Four arenas, each with two predators and one prey. In all arenas all predators use the same learning algorithm (QL or EAs) and advice may be requested by a predator to any of its seven peers in the same or other arenas.

For each of the above scenarios 11 trials were made (x4, or x8, agents of each type) with different random seeds. Each trial ran for 9000 epochs and each epoch has 150 turns. For each trial there is a corresponding test which runs for 1000 epochs without learning or exchange of advice. Each agent in the beginning of a test loads the parameters that the corresponding agent saved during training when it achieved its best absolute score. Partnerships were kept unchanged by this procedure.

Two experiments were made differing only in the dimension of the grid. In the first the grid was 10x10, and in the second 20x20 positions wide.

5.2 Experiment 1

The objective of the first experiment was to examine the behaviour of *advice-exchange* in a problem where QL-agents, when working individually, are clearly superior to Heuristic and EA-agents. In this experiment the dimension of the arena is 10x10 and *advice-exchange* is not allowed until mid trial (epoch 4500).

Each point in the graphics (and value in the tables) is based on 44 (11 trials x 4 agents) or 88 (11 x 8) different trials of an agent using a particular algorithm in a particular scenario. For the training data the average quality per epoch of the agent's actions for all experiments was calculated and these values were averaged for every 200 epochs, for better understandability of the graphics.

Each graphic shows the results of an agent that uses a given learning algorithm (QL or EA) when not using *advice-exchange* (labelled: *Individual*), in a social heterogeneous environment with and without heuristic experts (*Social Heuristic* and *Social Heterogeneous*, respectively) and in a social homogeneous environment composed solely of other agents using the same learning algorithm (*Social Homogeneous*). The averaged results of the heuristic experts are presented in all figures to facilitate the comparisons between different graphics.

The agent that produced the best average score in test for each algorithm-scenario pair was chosen as representative and its average score during the 1000 epochs of testing is shown on the column labelled "Best test avg" of tables 2 and 3. The average score in test per agent-type and scenario is also presented in the last column of the same tables, and it is labelled "Avg in test".

One of the most interesting, and important, facts in this experiment is that QL-agents in individual scenarios learn to perform better than Heuristic and EA-agents. This performance is achieved by developing joint strategies. Notice that there is no explicit attempt to develop cooperative strategies and the agents have no information concerning

their partners in their description of the state. Cooperation emerges spontaneously when one of the agents takes advantage of the other's strategy. These observations confirm the statements in (Sen et al., 1994; Sen and Sekaran, 1998) where the authors defend that it is possible for QL-agents to learn coordinated strategies without exchanging information or knowing that there is another agent acting in the same environment.

Another interesting fact, and also of great importance to the following discussion, is that it is common to observe one of the QL-agents profit more than the other from these joint strategies. In certain cases this phenomenon resembles more a form of parasitism than actual cooperation. This also holds if agents do not share the reward and was observed in experiments documented in (Nunes and Oliveira, 2002a). In these cases the joint strategy would often be destroyed because one of the members was not performing well enough and thus would seek advice (quite often from its own partner). When getting advice its strategy would change and the "parasite" would see its score drop to average/low levels in a few epochs simply because the dynamics of its environment had changed due to the change in its partner's behaviour. Since agents have no information regarding their partners they could only attribute the drop in their performance to changes in the environment and would have to re-evaluate its strategy based on the new dynamics. This usually caused the reward values to drop for quite a few epochs until the agent adapted itself to the new strategy of its partner. In some cases this new strategy was not so easily exploitable and the "ex-parasite" would never achieve the same level of performance and would fail in the final tests because it was trying to use a joint strategy with a peer that did not cooperate.

One of the reasons for the introduction of a global component in the reward function was precisely to maintain these successful groups. The sharing of rewards was relatively successful in maintaining most of the groups together even if one agent would get only a small percentage of its partner's reward. The break-up of joint strategies can still be observed in the current experiments, although in a much smaller scale than in previous ones. It was also observed, in some cases, that when both agents maintain their strategies reasonably stable for long enough the performance of the "parasited" agent can increase slowly without destroying the cooperative strategy.

As seen in table 2, QL-agents perform better in this experiment than either Heuristic or EA-agents. Curiously, although individual performances are better in terms of its average, the best tests are from QL-agents trained in social environments with *advice-exchange*. The difference is slight but this observation is consistent with previous tests.

EA-agents perform considerably better in *Social Heterogeneous* and *Heuristic* environments than in the *Individual* trials, proving that *advice-exchange* does improve the scores of low-performance agents to levels that are

Table 2: Best and average test results in experiment 1 for each scenario-algorithm pair, ordered by best test averages.

Scenario	Alg.	Best test avg	Avg in test
Social Het.	QL	0.20740	0.18471
Social Hom.	QL	0.20605	0.18803
Social Heu.	QL	0.20447	0.17101
Individual	QL	0.20413	0.19434
Social Het.	EA	0.20358	0.17627
Social Heu.	EA	0.19468	0.17118
Social Heu.	Heu.	0.17617	0.17345
Individual	EA	0.16148	0.14368
Social Hom.	EA	0.15850	0.14164

close to those of the advisors. Nevertheless, interferences caused by trusting heuristic experts too much can be noticed in the difference between the results for EA-Agents in *Social Heterogeneous* and *Heuristic* environments.

The exchange of advice in *Social Heterogeneous* scenarios produces the highest score in the best results column for EA-agents, although in *Homogeneous* scenarios it seems not to be very profitable, reaching slightly lower scores than in the *Individual* scenarios. This slight drop can be attributed to disturbances caused in a few trials when an agent is switching strategies in middle of the training.

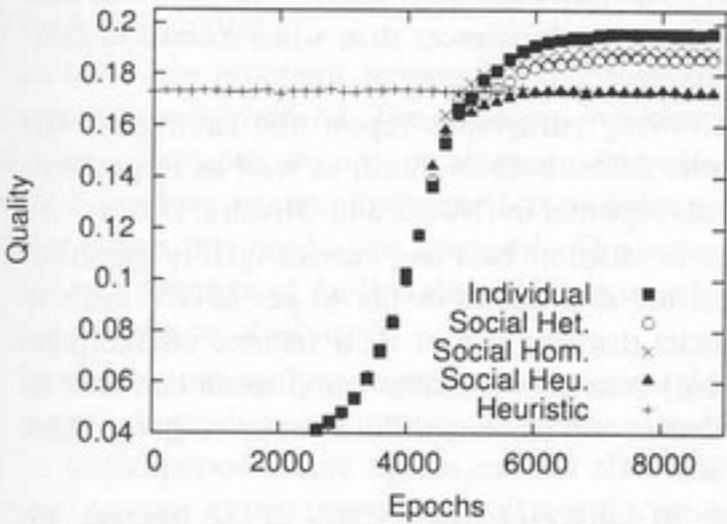


Figure 1: Training results for QL agents in experiment 1.

In figure 1 we can see the average evolution of the results of QL-agents in different scenarios during training. It is noticeable that at epoch 4500, when *advice-exchange* starts, social QL-agents suffer a small destabilization and in *Heuristic* scenarios start to take advice from heuristic experts, which leads them off their track and into a local maxima from which they do not seem able to escape.

In what concerns the performance of EA-agents in training, depicted in figure 2, we can notice that it is easier for an EA-agent to follow the performance of Heuristic agents (whose advice is more stable) than that of QL-agents.

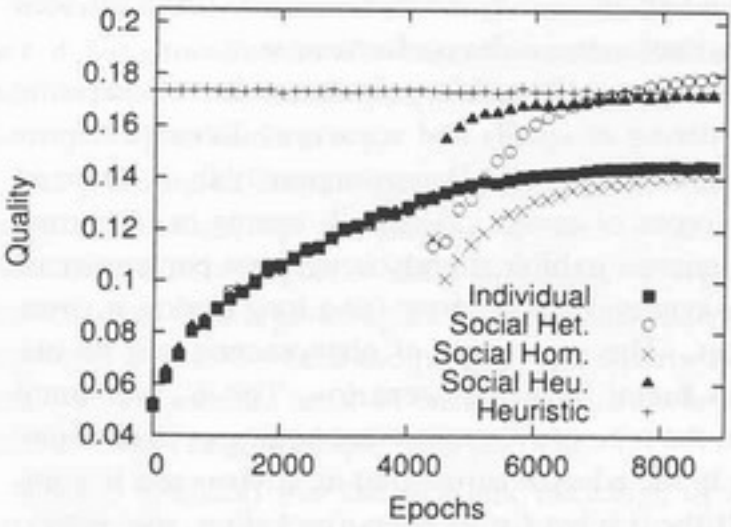


Figure 2: Training results for EA agents in experiment 1.

The number of advice-exchanged during training is relatively small. The highest value reported for both experiments is an average of 7 requests per epoch (150 turns) measured over the full trial (9000 epochs). It is important to notice however that advice is not allowed during at least half of these epochs and that the traffic generated is “bursty”, concentrating in a few epochs, usually between epochs 4500 and 5000.

5.3 Experiment 2

The objective of this experiment is to examine the behaviour of *advice-exchange* in a problem where EA-agents (when working individually) are, in average, superior to QL-agents. This experiment is, in all respects, similar to experiment 1, except in the dimension of the arenas (in this case 20x20).

Table 3: Best and average test results in experiment 2 for each scenario-algorithm pair, ordered by best test averages.

Scenario	Alg.	Best test avg	Avg in test
Social Heu.	QL	0.12291	0.12176
Social Heu.	Heu.	0.12227	0.12165
Social Heu.	EA	0.12201	0.11940
Social Het.	QL	0.11507	0.10721
Social Het.	EA	0.11477	0.10822
Individual	EA	0.11462	0.10661
Social Hom.	EA	0.11290	0.10693
Individual	QL	0.11079	0.08840
Social Hom.	QL	0.10820	0.08990

Contrary to the previous experiment in this QL-agents have serious difficulties in learning an appropriate policy individually. In this experiment heuristic agents behave considerably better then their peers.

We can see in the best results column that agents in *Social Heterogeneous* and *Heuristic* environments have better results than their counterparts in *Individual* scenarios,

and that when advised by heuristic experts, both QL and EA-agents achieve similar performances.

The average results are very similar to the best in terms of the ordering of agents and scenarios, although the ordering of *Individual* and *Homogeneous* trials is reversed for both types of agents. Also, QL-agents in *Heterogeneous* scenarios exhibit slightly better best performances than EA-agents (their teachers for a long period in these scenarios). The same type of phenomenon can be observed in *Social Heuristic* scenarios. This is, in a small measure, the type of effect we were hoping to find, which consists in the advisee surpassing its advisor and becoming itself the teacher for its peers (including, sometimes, its previous advisor).

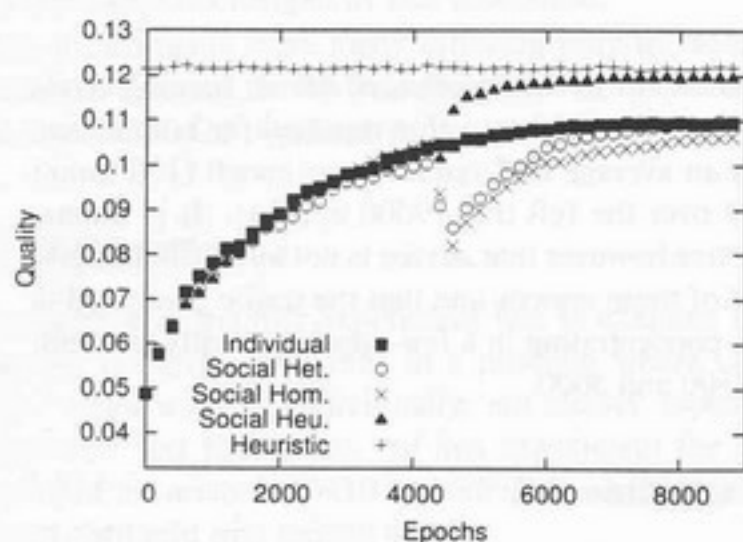


Figure 3: Training results for EA agents in experiment 2.

In figure 4 we can clearly see the effects of *advice-exchange* in the average training behaviour of QL-agents. The most interesting fact is that, contrary to the previous experiment, exchanging advice amongst agents of the same type improves considerably the performance of QL-agents, which means that by exchanging information they are able to learn better, in average, than when working individually. However, this effect is hardly noticeable in the test results. This difference between train and test results can be attributed to a sharper definition of the best action for each state when using *advice-exchange* during training. This effect (the improvement in the scores in *Homogeneous trials*) is not noticeable in the learning curves of EA-agents (figure 3).

In summary, QL-agents benefit from *advice-exchange*, being able, not only to copy their teachers, but often to have slightly higher performances than the agents from which they received advice. This effect in QL-agents can be attributed to a more stable strategy or to a fine tuning of the solution proposed by their EA peers. EA-agents do not seem to benefit so clearly, in average, from *advice-exchange* as their QL-peers, although the best results are similar to those of the advisors. This indicates some instability in the training of social EA-agents. By observing the learning curves and the test logs we can conclude that some of this instability can be attributed to the change from Heuristic to QL advisors.

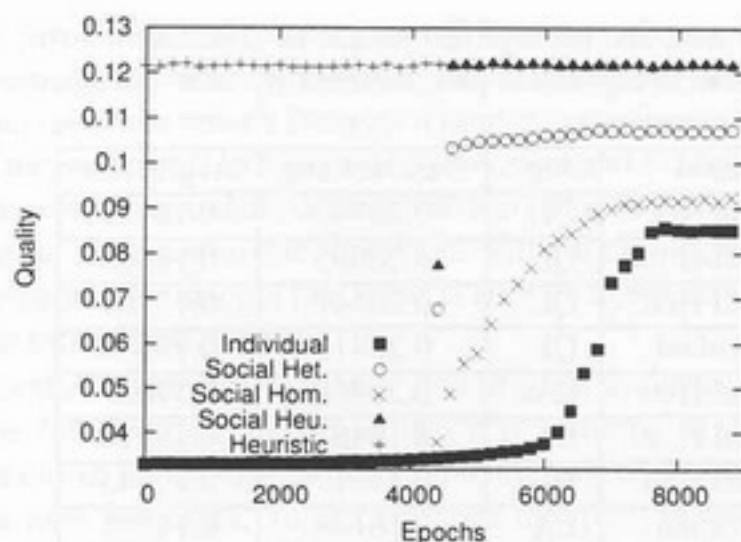


Figure 4: Training results for QL agents in experiment 2.

6 Discussion

One of the main reasons for the selection of these two algorithms (QL and EA) to test *advice-exchange* in the pursuit domain was that they have quite different internal structures and exploration strategies. EA usually climbs faster to a reasonable quality plateau but seems unable to fine-tune the solution. QL-agents often take very long to find any type of solution to the problem, in some cases failing to reach any solution within the experiment time-frame, but after having solved the problem a few times they quickly learn to perform at an expert level. The intended effect of exchanging advice would be that different agents could pull each other out of local maxima, thus achieving better performances than when trained in *Individual* scenarios.

The following paragraphs report and summarize the observations made, both in these, as well as in previous experiments reported in (Nunes and Oliveira, 2002a).

The use of absolute best and current quality measures in the decision of whether or not to get advice leads to worse results than the use of their relative counterparts (cq_i and bq_i) because a spurious good result can lead an agent that does not have a solid strategy to give others "bad advice".

The use of *advice-exchange* early in the training, although it improves the initial learning rate of QL-agents, may lead them to local maxima, i.e. non-optimal strategies that are similar to the ones obtained by their advisors.

In problems where EA-agents reach a reasonable score and QL-agents are unable to solve the problem (either individually or by exchanging advice with other QL-agents), advice from EA-agents is beneficial and leads to considerably better average results for social QL-agents. In the reverse situation (problems where QL-agents perform better), most EA-agents benefit from the advice of their QL-peers near the end of the training, but others are unable to profit from the teachings and in some cases performance drops because EA-agents fail to switch from their previous solution to the one they are being taught and use an in-between strategy which is frequently, very

poor. Another fact that can cause this drop is that sometimes EA-agents are being taught a strategy that requires two cooperating partners, and since the other partner may not be requesting advice, or may be advised by the wrong advisor, the “teamwork” that originated the good score of the advisor is never achieved by the EA-agents.

As mentioned previously QL-agents seem to develop joint strategies, and in some cases one of the partners has, consistently, higher scores than the other. Nevertheless, the actions of the low-performance agent are necessary for the common goal of catching a prey and thus to maintain the score of the high-performance partner. It was observed that when the lower quality agent is advised by its own partner (or any other agent) and changes its policy the average quality of both agents drops, sometimes irreversibly. When low-performance agents try to get advice from others that are engaged joint strategies it is fundamental that both agents in the advisee-group take advice from the two distinct partners in the advisor-group.

The frequent exchange of advisor during training may cause severe instability in both types of agents. Being unable to concentrate on learning one strategy, and getting conflicting advice from several advisors frequently leads agents to long periods of low scores. Also, the sudden withdrawal of advice can hurt advisees (especially QL-agents). It was observed that when a QL-agent did not have a consolidated policy, as soon as it reached a score in which advice was no longer requested, its performance dropped. This happens mainly in experiment 2, which requires a long sequence of similar actions with low reward to achieve a capture.

To solve the problems mentioned above several new concepts were introduced. The measures in which agents base their decisions are not instantaneous but calculated using a window, or an infinite horizon technique, based on the values that need to be accessed. This contributes to the improvement of the decision of when to get advice and in the choice of advisors.

The problem of early convergence to local maxima due to exchange of advice was mitigated by the introduction of an initial period where agents are not allowed to exchange advice. At this time the period is still a pre-set parameter but it can easily be defined as a condition based on the derivative of the performance of an agent, similarly to many criteria for stopping training. When this derivative approaches zero for a certain number of epochs we can conclude that the agent will not be able to improve its performance any further alone, and proceed in the process of getting advice. It is important to mention that this solution, although it leads to better final scores, does so at the expense of a slower initial convergence in QL-agents.

Due to the specific nature of the learning algorithms we have noticed that QL-agents are much easier to advise than their EA peers. We have introduced *advice-replay* to cope with this disadvantage of EA-agents. Improvements in the time EA-agents take to incorporate advice and improve their scores were clearly noticeable. We are,

nevertheless, fully aware that the more efficient the advice intake procedure is, the easier it is to have all agents rapidly going to the same area of the search-space with a consequent loss of diversity.

The problem of learning joint strategies from another group was solved by exchanging *advisor preferences*. This consists in one agent sending a message to its partner advising it to trust a given agent. This solution is reasonable for a problem where the partners are clearly identified, but refinements must be made for cases when it is unclear which are the cooperating partners.

Another problem was the frequent exchange of advisors. This was dealt with by giving preference to the last advisor. This simple strategy has clearly diminished the instabilities in training during *advice-exchange*.

The most important aspects in exchanging information are in the selection of when, and from where, to get this information. The decision of “when” is based on the relative quality statistics, mentioned above and in the concepts of “*self-confidence*” and “*confused about a given state*”. The first has a twofold effect, it prevents spurious bad results from leading an agent to seek advice (possibly abandoning a good strategy) and provides an advisee-agent the extra-time necessary to solidify its strategy after a sequence of advice. The second concept is a first step to define agents’ learning stages. At some points an agent does not need continuous advice (because it has a relatively good score) but is sometimes unclear as to which is the best response for a given state. It is important, at this stage to ask for help only in some situations and be critical about the advice that is given. This may be considered an intermediate learning stage in which the agent makes the transition from continuous advice to autonomous exploration. The “*confused about state*” condition makes this transition easier and less turbulent.

The other important aspect that needs to be solved when dealing with exchange of information is: “where” to get it from. It was soon evident that in certain problems the agent with the best score was not the best advisor for all or part of its peers. This can happen due to a number of reasons, most of which were mentioned above: spurious good results, advisor is using a joint strategy, dynamics of the advisor’s environment are not similar enough to the advisee, etc. To solve the problems that we have faced in the choice of advisors we used the concept of *trust*. Trust is a measure of how useful the advice of other agents was in the past to a given agent. This has also contributed to stabilize the *advice-exchange* process by helping agents to choose better advisors.

Table 4 presents a comparison of the average duration of a trial in several different scenarios. It is important to notice that in *Heuristic* scenarios there are 10 agents in 5 arenas, while other scenarios have only 8 agents in 4 arenas. As for all other characteristics all scenarios are in equal stand. The most time-consuming event in the trials is undoubtedly *advice-replay*, without which the difference between individual and social scenarios in terms of

computation time would be considerably lower.

Table 4: Average time per trial (in seconds) measured in 11 trials on the 20x20 scenario.

Scenario	Avg time (sec)
Individual	263
Social Heu.	970
Social Het.	1696

As mentioned above, there are still some problems that need to be addressed, namely:

- Improve procedures for coordinated learning of joint strategies.
- Consider options to adaptively balance the choice between individual exploration and reliance on advice.
- Automate the choice of values for the majority of the parameters that in these experiments are pre-set.
- Create the means to modify the learning behaviour according to the phase of the training (initial, high climb, stabilization or fine-tuning).
- Study how to combine information from different sources.

One of the main expectations that drives this research is to create teams of agents that through communication may be able to learn more and faster than the same number of agents could when working individually. This was observed so far only in a minor scale and mostly in the harder problems with which we have experimented. It is important to assert if this behaviour does occur in larger scale, and, if so, under which circumstances.

7 Conclusions and Future Research

The results obtained show that *advice-exchange* can be beneficial for the performance of learning agents even when the exchanged information is very simple. In both experiments, the type of agents that has lower performances achieves better scores when being advised. The use of *advice-exchange* in situations where the action of an agent depends on the strategy of others requires added capabilities on the part of agents to select and combine information from several sources, or to make better use of meta-learning information, such as the results of other agents or their effectiveness as advisors. The performance of most social agents can achieve levels similar or sometimes slightly higher than the best of the peers they have contact with, but clear improvements over the performance of the best of the members of a group were not observed in these experiments.

The process of human learning has several stages of growing autonomy: imitation; seeking advice for a complete solution; seeking advice for a particular step of a solution, or for how to relate a new problem with previous ones and, finally, a relatively autonomous stage where people simply request opinions to some trusted peers to increase the confidence in the solutions they have learned, or to adapt themselves to new situations. The introduction of these stages of evolution in the process of *advice-exchange* can be an interesting step towards the development of a technique that can autonomously shift the way in which advice is being used.

The application of similar methods to other type of learning agents, as well as other problems, is also an important step in the validation of this approach. The undergoing research in a traffic-control problem, as an extension of the work presented in (Nunes and Oliveira, 2002b), is of great importance to the validation of this technique in a more complex environment.

A development of the way *trust* is used by agents, associating it with agent-situation pairs, may allow the advisee to differentiate who is the expert on the particular situation it is facing and improve the effectiveness of *advice-exchange*. Another interesting issue is that humans usually offer unsolicited advice for some situations. Either great new discoveries or actions that may be harmful for the advisee seem to be of paramount importance in the use of advice. Rendering unsolicited advice at critical points, also seems like a promising approach to improve the skills of a group of learning agents. The same applies to the combination of advice from several sources.

Our major aim is to, through a set of experiments, derive some principles and laws under which learning in the Multiagent Systems framework proves to be more effective than, and inherently different from, just having agents learning as individuals.

References

- M. Benda, V. Jagannathan, and R. Dodhiawalla. On optimal cooperation of knowledge resources. Technical Report BCS G-2012-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA, 1985.
- J. Clouse and P. Utgoff. A teaching method for reinforcement learning. In *Proc. of the Ninth Int. Conf. on Machine Learning*, pages 92–101, 1992.
- J. A. Clouse. *On integrating apprentice learning and reinforcement learning*. PhD thesis, University of Massachusetts, Department of Computer Science, 1997.
- J. A. Clouse and P. E. Utgoff. Two kinds of training information for evaluation function learning. In *Proc. of AAAI'91*, 1991.
- M. Glickman and K. Sycara. Evolution of goal-directed behavior using limited information in a complex en-

- vironment. In *Proc. of the Genetic and Evolutionary Computation Conference, (GECCO-99)*, 1999.
- T. Haynes, R. Wainwright, S. Sen, and D. Schoenfeld. Strongly typed genetic programming in evolving cooperation strategies. In *Proc. of the Sixth Int. Conf. on Genetic Algorithms*, pages 271–278, 1995.
- J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 1996.
- J. R. Koza. *Genetic programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge MA, 1992.
- L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.
- L. Nunes and E. Oliveira. Advice-exchange in heterogeneous groups of learning agents. Technical Report 1 12/02, FEUP/LIACC, 2002a. URL <http://home.iscte.pt/~lmmn/www>.
- L. Nunes and E. Oliveira. On learning by exchanging advice. In *Proc. of the First Symposium on Adaptive Agents and Multi-Agent Systems (AISB'02)*, 2002b.
- L. Nunes and E. Oliveira. Advice exchange between evolutionary algorithms and reinforcement learning agents: Experimental results in the pursuit domain. Technical Report 2 03/03, FEUP/LIACC, 2003. URL <http://home.iscte.pt/~lmmn/www>.
- B. Price and C. Boutilier. Implicit imitation in multiagent reinforcement learning. In *Proc. of the Sixteenth Int. Conf. on Machine Learning*, pages 325–334, 1999.
- B. Price and C. Boutilier. Imitation and reinforcement learning in agents with heterogeneous actions. *Seveteenth International Conference on Machine Learning ICML2000*, 2000.
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, 1:318–362, 1986.
- R. Salustowicz. *A Genetic Algorithm for the Topological Optimization of Neural Networks*. PhD thesis, Tech. Univ. Berlin, 1995.
- S. Sen and P. P. Kar. Sharing a concept. In *Working Notes of the AAAI-02 Spring Symposium on Collaborative Learning Agents*, 2002.
- S. Sen and M. Sekaran. Individual learning of coordination knowledge. *Journal of Experimental, Theoretical Artificial Intelligence*, 10:333–356, 1998.
- S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proc. of the National Conf. on AI*, pages 426–431, 1994.
- M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proc. of the Tenth Int. Conf. on Machine Learning*, pages 330–337, 1993.
- C. J. C H. Watkins and P. D. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3):279–292, 1992.
- S. D. Whitehead. A complexity analysis of cooperative mechanisms in reinforcement learning. *Proc. of the 9th National Conf. on AI (AAAI-91)*, pages 607–613, 1991.
- X. Yao. Evolving artificial neural networks. In *Proc. of IEEE*, volume 87, pages 1423–1447, 1999.

Evolving Strategies for Agents in the Iterated Prisoner's Dilemma in Noisy Environments

Colm O'Riordan,
Dept. of Information Technology,
National University of Ireland,
Galway, Ireland.
Email: colmor@gemina.nuigalway.ie

Abstract

In this paper, we discuss the evolution of agents in a multi-agent system. Within this system, agents interact with each other; these interactions are modelled in an abstract manner using ideas from game theory. This paper focuses on the iterated prisoner's dilemma. We discuss properties we believe to be of importance with respect to fitness of strategies in traditional environments and also in environments where noise is present. Specifically, we discuss notions of *forgiveness* (where strategies attempt to forgive strategies that defect in the game) and *memory* (where strategies are reactive to immediate events but maintain a memory of past interactions).

1 Introduction

The motivations for further research in the iterated prisoner's dilemma, given that it has been well-studied in a variety of domains (Axelrod and D'Ambrosio (1994)) are two-fold:

- the assumption of cooperative behaviour in many domains (for example, multi-agent systems) is not always valid, and
- the assumption of a noise-free environment is also not valid in certain domains. An *intended* cooperative gesture by one participant may be interpreted as a non-cooperative gesture by the receiver due to a number of reasons (for example, ambiguity in the message; conflicting goals and differences in ontologies maintained by the participant). A transmitted gesture may be lost or damaged in transmission which may result in the receiver mis-interpreting the message. These potential problems may result in a cooperative gesture being recognised by the receiver as a defective gesture and vice versa.

The latter problem has not been widely studied, with only a handful of researchers considering noise and errors in games such as the iterated prisoner's dilemma (Bendor et al. (1991), Kahn and Murnighan (1993), Miller (1996), Mueller (1988)).

Subsequent sections outline some recent results obtained in experiments dealing with the iterated prisoner's dilemma in noisy environments. The approach taken in this research has been to design strategies based on heuristics, to validate these strategies in fixed

environments and to search the range of properties of these strategies using evolutionary computational search techniques.

Section 2 presents related research in the areas of the prisoner's dilemma and noisy environments. Section 3 motivates the need to develop new strategies for noisy environments. Section 4 introduces two classes of strategies that for environments. Experimental results regarding these strategies are presented in Section 5. The paper concludes with some observations regarding the evolved strategies and their properties.

2 Related Work

2.1 Prisoner's Dilemma

In the prisoner's dilemma game, two players are both faced with a decision—to either cooperate(C) or defect(D). The game is often expressed in the canonical form in terms of pay-offs:

	Player 1	
	C	D
Player 2	C (λ_1, λ_1)	(λ_2, λ_3)
	D (λ_3, λ_2)	(λ_4, λ_4)

where the pairs of values represent the pay-offs (rewards) for players Player 1 and Player 2 respectively. ($\lambda_3 < \lambda_1 < \lambda_4 < \lambda_2$. The constraint $2\lambda_1 > \lambda_2 + \lambda_3$ also holds.)

In the iterated version, two players will play numerous games (the exact number not known to either player). Each player adopts a strategy to determine whether to cooperate or defect at each of the moves in the iterated game.

A computer tournament (Axelrod (1984)) was organised to pit strategies against each other in a round-robin manner in an attempt to identify successful strategies and their properties. The winning strategy was *tit-for-tat*; this strategy involved cooperating on the first move and then mirroring one's opponent's move on all subsequent moves.

The initial results and analysis suggested that the following properties are necessary for success—niceness (cooperate first), retaliatory (be willing to punish strategies that defect), forgiving (be willing to cooperate following a cooperation) and clear (strategy should be sufficiently simple to render it possible for an opponent to ascertain the strategy).

In a second tournament (Axelrod (1984)), of the top 16 strategies, 15 were found to be nice. These results indicate that cooperative strategies are useful if there is a high chance that the strategies will meet again.

2.2 Forgiveness in the Prisoner's Dilemma

In earlier work (Riordan (2000)), we explored the notion of forgiveness—the ability or readiness to play cooperatively in the game once mutual defection has been detected¹. The type of forgiveness we report is of a reactive nature, where cooperative gestures were played immediately following the detection of a spiral of defections.

In our experiments (O'Riordan (2001)), using an evolutionary computation technique, we encoded, in each chromosome, several aspects of a strategy's behaviour. These include: behaviour on the first move, behaviour following a defection by opponent, behaviour following a cooperation by opponent, number of mutual defections to allow before entering to *forgiving* behaviour, number of successive cooperative gestures to make in order to *forgive*, whether a strategy will use calculations based on a longer memory to choose the next move and finally a threshold (number of previous cooperations less the number of previous defections) under which strategies will not cooperate.

¹Note: this differs from Axelrod's notion of contrition which is played by strategies to recover from the effect of their own defective behaviour

In these experiments, the evolved strategies played in an environment comprising several well known fixed strategies which were not subjected to evolutionary pressures.

On all trials the first genes corresponding to the first three properties converged extremely quickly and confirm findings by many others—cooperate on first move, cooperate following a cooperation and defect following a defection.

The fifth gene does not converge to zero in any of the experiments, indicating a *forgiving* behaviour is a useful property for any strategy to maintain. The level of *forgiveness* and the length of the spiral have varied in the trials. These experiments allowed us to explore a range of properties of *forgiving* strategies.

Further experiments involved co-evolving a set of strategies where the initial set of strategies were created randomly and evolved in the environment of creatures playing the iterated prisoner's dilemma.

In these experiments, we see that *forgiveness* quickly becomes the norm; strategies choose to forgive strategies with whom they are locked in a mutual spiral of defection. Furthermore, we see that in not all runs is cooperation on the first move selected as the norm. We believe the presence of forgiveness in the population reduces the evolutionary advantage conferred on those that initially cooperate; the danger of mutual defection is not as costly as the strategies are likely to forgive.

2.3 Noisy Environments

The majority of work in the iterated prisoner's dilemma has focused on the games in a noise-free environment, i.e., there is no danger of a signal being misinterpreted by the opponent or of the signal being damaged in transit.

There are different means that can be chosen to introduce noise to the simulation:

- mis-implementation (when the player makes a mistake implementing its choice)
- mis-perception (when one player misperceives the other player's signal or choice)

Fundunberg and Maskin (1990), argue that "if mistakes are possible, evolution may tend to weed out strategies that impose drastic penalties for deviations".

Kahn and Murnighan (1993) find that in experiments dealing with the iterated prisoner's dilemma in noisy environments, cooperation is more likely when players are sure of each other's payoffs. Miller's experiments

in genetic algorithms applied to the prisoner's dilemma results in the conclusion that cooperation is at its greatest when there is no noise in the system and that this cooperation decreases as the noise increases (Miller (1996)).

Hoffman (2000) reports that results are sensitive to the extent to which players make mistakes (mis-implementation or mis-perception). In particular, cooperation is vulnerable to noise as it is supported by conditional strategies. For example, in a game between two *tit-for-tats*, a single error would trigger a series of alternating defections.

We argue, and demonstrate with empirical results, that in both clean and noisy environments, the willingness to break spirals of mutual defection by incorporating a level of forgiveness is a desirable trait in achieving higher fitness (a function of pay-offs scored in the game) among agents. Furthermore we suggest that as environments become noisy a longer memory is beneficial; i.e., one should base the reaction not on just the previous few moves but on a longer memory of the game.

3 Noise in the IPD

To further motivate the need to study strategies for the prisoner's dilemma in noisy environments, it is worth considering the effect of noise on well-known, well-researched existing strategies.

If we consider two *tit-for-tat* strategies interacting in a noise free environment, we obtain an ongoing mutual cooperation.

Introducing noise ($=p$) we get three possible states—mutual cooperation, mutual defection and repeated cooperation-defection pairs. This is illustrated in Figure 1.

This gives us the following transition matrix:

$$\begin{pmatrix} (1-p)^2 & 2(1-p)p & p^2 \\ (1-p)p & (1-p)^2 + p^2 & (1-p)p \\ p^2 & 2(1-p)p & (1-p)^2 \end{pmatrix}$$

Given P , we can calculate P^n . If α is a row of P , then α^n will converge. This vector contains values indicating the probability that the system will be in that state following n iterations.

$$\begin{pmatrix} (1-p)^2 & 2(1-p)p & p^2 \\ (1-p)p & (1-p)^2 + p^2 & (1-p)p \\ p^2 & 2(1-p)p & (1-p)^2 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}$$

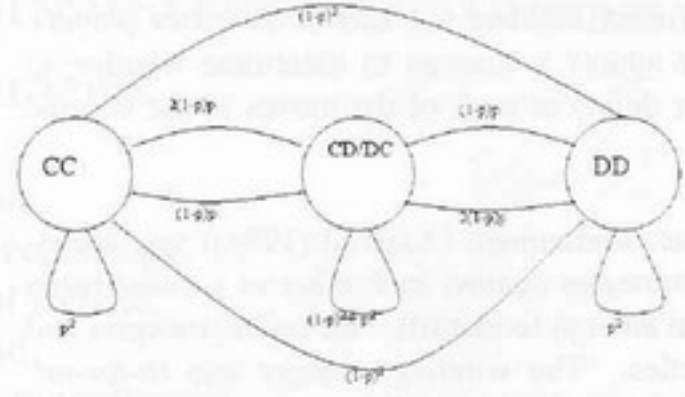


Figure 1: State transition diagrams for two *tit-for-tats* in a noisy environment

We also have $\alpha_1 + \alpha_2 + \alpha_3 = 1$

Solving, we find that for any noise ($p > 0$), we get $\alpha = (\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$. So given a society of strategies playing the prisoner's dilemma with a *tit-for-tat* strategy, we see that the society will (given an infinitely long game) arrive at a reward equal to

$$(\frac{\lambda_1}{4} + \frac{\lambda_1 + \lambda_2}{2} + \frac{\lambda_4}{4})$$

We can see that the presence of noise is detrimental to the overall fitness of a society of *tit-for-tat* like strategies.

We can do similar analysis for other simple strategies and derive the amount of time spent in each state, given an infinite game.

Empirical analysis of societies of homogeneous agents (i.e., all playing the same strategy) provides us with more motivation for studying forms of forgiveness in noisy environments. In Figure 2, we plot the fitness of societies of strategies. We include societies of well-known strategies: *tit-for-tat*, *all-c* (where strategies cooperate all the time) and *spiteful* (where strategies cooperate initially, cooperate for as long as opponent cooperates, but once opponent defects, spiteful defects continuously).

We see that *all-c* is not affected to a large degree by the introduction of noise; the fitness of the society remains high. This is as expected as *all-c* does not retaliate and punish defections by the opposing strategy. However, we see that the fitness in the society of *tit-for-tats* falls dramatically with the introduction of any noise. The society of *spiteful* strategies is seriously affected by the introduction of even a small amount of noise. As noise increases, the overall fitness increases as strategies have a higher chance of obtaining scores other than the score for mutual defection.

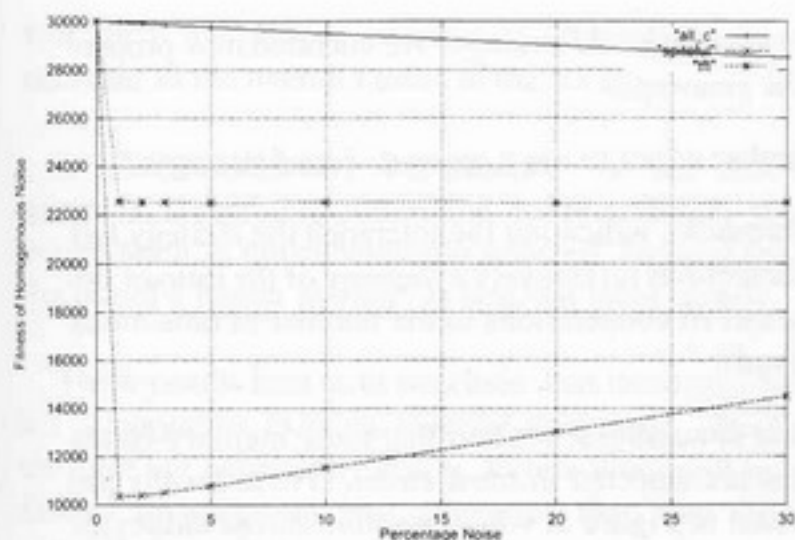


Figure 2: Fitness of societies in noisy environments

4 Forgiveness as a means to deal with Noise

The basic premise behind *forgiving* strategies in this paper is to recover from the effects of noise (spirals of defection) and regain mutual cooperation if possible. Given a spiral of defection (both strategies defecting continuously, or less seriously two strategies alternating between receiving rewards λ_2, λ_3), *forgiving* strategies attempt to play a cooperative gesture (a number of cooperations) to break the spiral.

We can define a family of such strategies differing according to the values of a number of parameters—the length of a spiral until action is taken, the length of the cooperative gesture and the amount of times a strategy forgives an opponent.

These strategies are reactive in that they react immediately to the recent history of moves; an alternative means to view forgiveness is to consider ‘memory-based’ strategies (i.e. those that use a longer history of events in determining their next move). Moves played will be determined by the game history (or summary of this history), and not solely on the immediate past. The motivation behind this approach is that costly spirals of defection can be avoided if strategies do not react immediately to defections.

In Figure 3, to illustrate the potential advantages of these approaches, we plot the fitness of societies comprising *forgiving* strategies and *memory-based* strategies respectively; we see that both strategies are relatively robust to the introduction of noise.

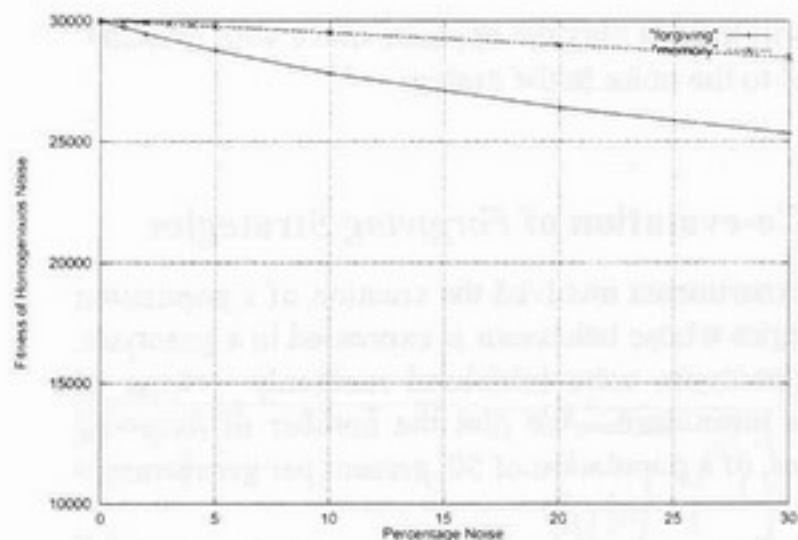


Figure 3: Fitness of societies in noisy environments

5 Evolving Strategies in a Noisy Environment

In the previous sections, we have discussed two classes of strategies (*forgiving* and *memory-based*) and have shown their effectiveness in noisy environments. These represent two classes of strategies that could be used to combat the potentially detrimental effect of introducing noise into an environment.

It is difficult to say, in advance, which type of strategy is likely to be selected for, if any, in an evolutionary setting. Furthermore, it is not clear which values for the various parameters would be chosen (e.g., length of spiral, thresholds in the memory-based strategies).

Given the huge range of possible strategies and emergent behaviours based on interactions, evolutionary computation provides a powerful mechanism to search this space of potential strategies.

Three sets of experiments were carried out:

- co-evolution of *forgiving* strategies
- co-evolution of *memory-based* strategies
- co-evolution of both *forgiving* and *memory-based* strategies.

5.1 Experimental Setup and Results

In all three sets of experiments, we co-evolve populations of strategies playing the iterated prisoner’s dilemma in noisy environments where a genetic algorithm is used to evolve successful strategies. In each genotype, properties of strategies are represented. To calculate fitness, each strategy represented is played against all other represented strategies. Noise is implemented via a mis-implementation, i.e., a strategy when playing a

move, will instead play the opposite move with probability equal to the noise in the system.

5.2 Co-evolution of *Forgiving* Strategies

These experiments involved the creation of a population of strategies whose behaviour is expressed in a genotype. These genotypes were initialised randomly. Noise of 1% was introduced. We plot the number of *forgiving* strategies, of a population of 50, present per generation.

The genotype used in these experiments encodes a number of properties of the strategies. These are:

- the first move,
- the move following a cooperation,
- the move following a defection,
- whether or not to forgive,
- the length of time to wait during a mutual spiral before evoking cooperative moves to play as a cooperative gesture (*forgiving*),
- how often to forgive.

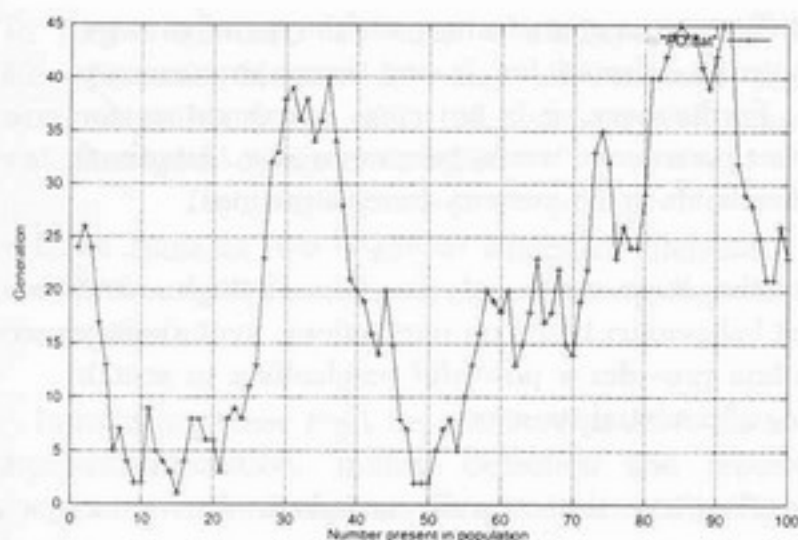


Figure 4: *Forgiving* strategies in an evolutionary setting in noisy environments

The results, over simulations, with 1% noise are plotted in Figure 1. We plot whether *forgiveness* is selected for or not. We find *forgiveness* is not being selected consistently and, in some cases, where it does become a dominant property in the population, it is not stable. For higher levels of noise, this becomes even more pronounced—*forgiving* (re-establishing cooperation after a defection) is not selected for consistently.

5.3 Co-evolution of *Memory-based* Strategies

Similarly, in these simulations, we report on strategies in noisy environments that may ignore some defections by

using a *memory-based* strategy. We encoded new properties in the genotype:

- whether or not to use a *memory-based* strategy,
- a threshold, indicating the tolerance the strategy has to defections (effectively a measure of the ratio of the number of cooperations to the number of defections played).

In these simulations, we find that these *memory-based* properties are selected in most cases. These results are summarised in Figure 5, where *memory-based* strategies are the majority strategy in the society.

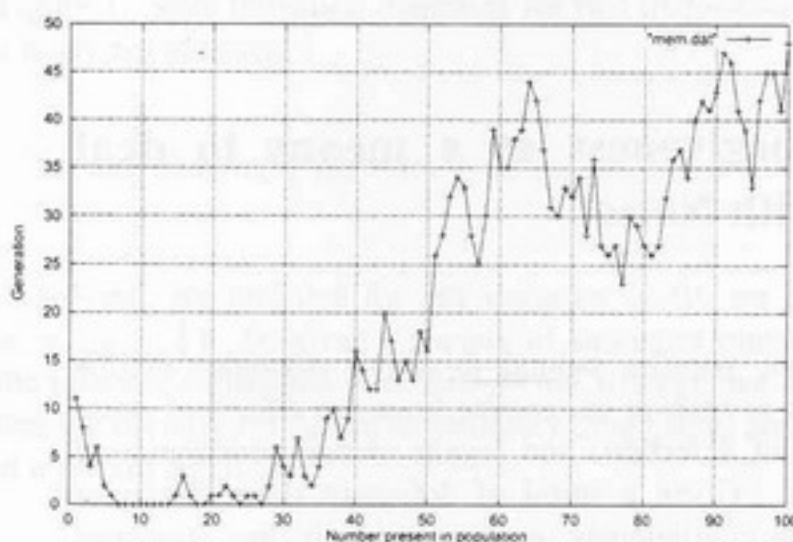


Figure 5: *Memory-based* strategies in an evolutionary setting in noisy environments

5.4 Co-evolution of both classes of Strategies

In the third set of experiments, we encode properties to represent both classes of strategies in order to ascertain which of the two classes will be selected in an evolutionary setting. In these experiments we evolve a population of 100 strategies playing the iterated prisoner's dilemma. In Figures 6 and 7, we plot the number of *forgiving* and *memory-based* strategies present. These show, again, that *forgiving* strategies are not stable whereas *memory-based* strategies show higher levels of stability.

6 Conclusion

The iterated prisoner's dilemma is an oft-studied game in many domains. In multi-agent systems, it serves as a useful abstraction of many forms of agent interactions. This paper examined some properties of strategies playing the game in a noisy environment.

The introduction of a low level of noise into the system has an immediate effect on many strategies and we saw

that their willingness to react to defections causes a decrease in the overall fitness of the society.

In the evolutionary experiments we see that forgiveness, based on attempting to re-establish mutual cooperation, is not consistently selected. We also see that the use of a longer memory is selected more readily.

These results lead us to conclude, that these approaches are superior to existing reactive strategies. However, they are not necessarily stable. Future work will involve further investigation into properties that may promote stable cooperation in noisy environments.

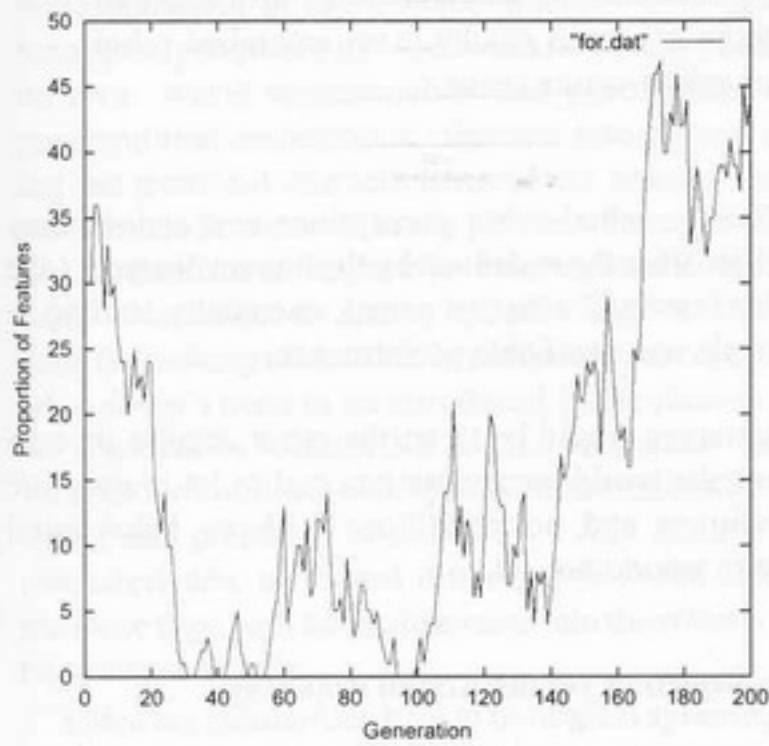


Figure 6: Number of *forgiving* strategies present

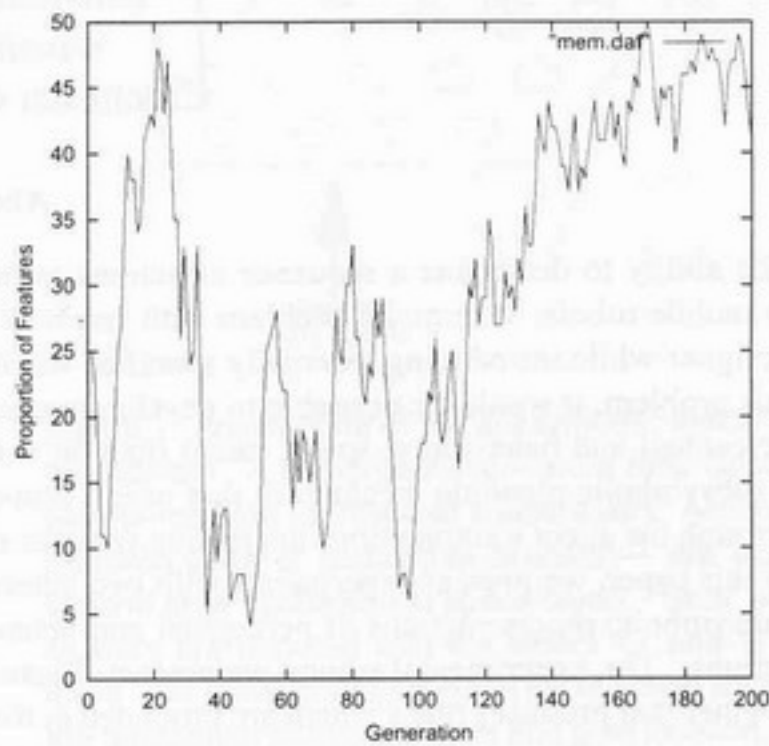


Figure 7: Number of *memory-based* strategies present

References

R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

R. Axelrod and L. D'Ambrosio. An annotated bibliography on the evolution of cooperation. http://www.ipps.lsa.umich.edu/ipps/papers/coop/Evol_of_Coop_Bibliography.txt, October 1994.

J. Bendor, R.M. Kramer, and S. Stout. When in doubt cooperation in a noisy prisoner's dilemma. *Journal of Conflict Resolution*, 35(4):691–719, 1991.

D. Fundunberg and E. Maskin. Evolution and cooperation in noisy repeated games. *The American Economic Review*, 80(2):274–179, 1990.

R. Hoffman. Twenty years on: The evolution of cooperation revisited. *Journal of Artificial Societies and Simulation*, 3(2), 2000.

L.M. Kahn and J.K. Murnighan. Conjecture, uncertainty, and cooperation in prisoner's dilemma games: Some experimental evidence. *Journal of Economic Behaviour & Organisation*, 22:91–117, 1993.

J.H. Miller. The coevolution of automata in the repeated prisoner's dilemma. *Journal of Economic Behaviour & Organisation*, 29:87–112, 1996.

U. Mueller. Optimal retaliation for optimal cooperation. *Journal of Conflict Resolution*, 31(4):692–724, 1988.

C. O'Riordan. Forgiveness in the iterated prisoner's dilemma and cooperation in multi-agent systems. *UK-MAS*, (4), December 2001.

C. O' Riordan. A *forgiving* strategy for the iterated prisoner's dilemma. *Journal of Artificial Societies and Social Simulation*, 3(4), 2000.

Experiments in Subsymbolic Action Planning with Mobile Robots

John Pisokas and Ulrich Nehmzow

Dept. of Computer Science

University of Essex

Wivenhoe Park

Colchester CO4 3SQ, UK

Abstract

The ability to determine a sequence of actions in order to reach a particular goal is of utmost importance to mobile robots. One major problem with *symbolic* planning approaches regards assumptions made by the designer while introducing externally specified world models, preconditions and postconditions. To bypass this problem, it would be desirable to develop mechanisms for action planning that are based on the *agent's* perceptual and behavioural space, rather than on externally supplied symbolic representations. We present a subsymbolic planning mechanism that uses a non-symbolic representation of sensor-action space, learned through the agent's autonomous interaction with the environment.

In this paper, we present experiments with two autonomous mobile robots, which use autonomously learned subsymbolic representations of perceptual and behavioural space to determine goal-achieving sequences of actions. The experimental results we present illustrate that such an approach results in an embodied robot planner that produces plans which are grounded in the robot's perception-action space.

1 Introduction

Determining (Sequences of) Actions

Besides the ability to stay operational (e.g. obstacle avoidance *etc.*) and to navigate, action selection is the third major competence required for autonomously operating mobile robots.

One possibility for achieving action selection is to determine a world model manually, to specify permitted operators (actions) applicable to it, and to use standard artificial intelligence search techniques to find paths through the action space that will lead from an initial state to the goal state. STRIPS (Fikes and Nilsson, 1971; Nilsson, 1984) is an early example of this approach.

More recently hybrid architectures were developed which employ symbolic planning techniques. They attempt to tackle problems related with symbol grounding (Harnad, 1990) by incorporating a reactive layer (Arkin, 1998). Well known examples are SOMASS (Malcolm, 2000) and ATLANTIS (Gat, 1992).

However, there are substantial weaknesses to approaches involving symbolic representations:

- Because of the external identification of operators, only part of the behavioural space available to the robot may be exploited.
- Similarly, only part of the perceptual space available to the robot may be used, due to the difference of perception and experience of the world between robot and human designer.

- Finally, actual robot perceptions and actions may differ from those defined by the human designer (due to sensor and actuator noise), eventually leading to brittle and unreliable performance.

An alternative would be to let the robot acquire its own subsymbolic world representation, and to let it associate preconditions and postconditions with its behavioural repertoire autonomously.

Action Selection versus Action Planning

The bulk of research on subsymbolic planning to date has focused on simulated agents, rather than situated, embodied mobile robots. In Baldassarre (2001a,b), to name a recent example, simulated robots were able to select *one* optimal action (with respect to a fixed performance criterion), given a perceptual state. In contrast to such 'action selection' mechanisms, we present a mechanism for 'action planning' that allows the association of *multiple* actions with each perceptual state, and the selection of actions with respect to *variable* performance criteria.

Reaction-Diffusion Dynamics

To achieve this, we used the concept of reaction-diffusion dynamics, introduced by Turing (1952). The use of reaction-diffusion dynamics to accomplish a search through a *symbolic* search space was proposed by Steels (1988), and has been used frequently in Artificial Intelligence for searching in artificial worlds. Applications

such as Tower-of-Hanoi problem or abstract path planning are presented, for instance, in Fleuret and Brunet (2000) and Fomin et al. (1996). In contrast to this work, we use reaction-diffusion dynamics for searching through a *subsymbolic* search space.

Regarding applications in robotics, Zeller et al. (1996) for instance present a solution to a path planning problem for a real robot arm. However, heuristics were introduced *a priori* (in the form of a gradient descent search) adding *tacit knowledge*¹ component to the planner, making the planner task specific. A similar approach is taken by (Fomin et al., 1996) in a simulated path planning task. In contrast to this approach, we avoid heuristics or other task specific knowledge altogether in the planner.

Contribution

In this paper we present experiments with a subsymbolic action planning mechanism that is based on *learned* representations of the robot's world. The agent autonomously explores its behavioural repertoire and learns its own world representation and precondition-action-postcondition associations. Specific information regarding the technical characteristics of the sensors and actuators is not introduced in the planner. Instead, we leave the mechanism to acquire its own representation of the robot's sensor-motor space, and later exploit this knowledge for making plans. Our hypothesis is that such knowledge doesn't need to be introduced to the planner and as we illustrate in sections 3.2 and 3.3 taking this approach we avoid introducing task specific assumptions by the designer and problems of scalability. Any specific sensor characteristics, or indeed difference between sensors of the same type, will be incorporated into the robot's knowledge automatically.

There are similarities here to biological systems, in that the signals transmitted through the nervous system do not convey information about the type of receptor, but merely the signal produced by the receptors (Kandel et al., 1995).

2 The Action Planning Mechanism

The architecture of the subsymbolic planning mechanism is depicted in figure 1. There are two layers: a perceptual space layer and a behavioural space layer. In the perceptual space layer a self-organising feature map (Kohonen, 1984) clusters sensory perception (section 2.1). In the behavioural space layer the clusters of the first layer are linked by actions available to the robot (section 2.2). To determine 'paths' between start and goal perceptions, reaction-diffusion dynamics are used (section 2.3).

¹*Tacit knowledge* is the kind of knowledge that the system cannot reason about. A form of tacit knowledge is the procedural code in a program.

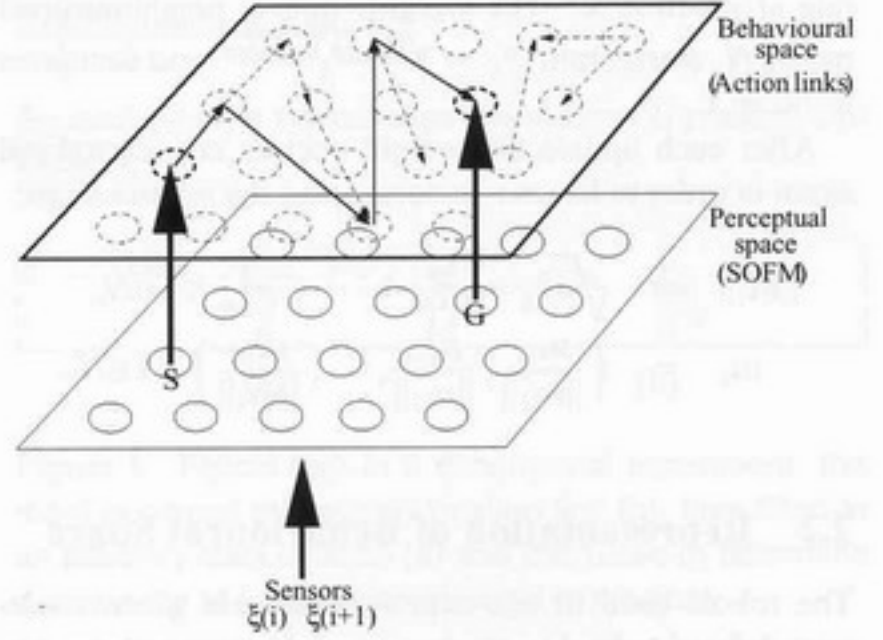


Figure 1: Architecture of the subsymbolic action planning mechanism. A self-organising feature map clusters sensory perception (perceptual space layer). Action links — between pairs of perceptual clusters — are stored in a second layer (behavioural space layer). 'Goal' and 'start' clusters are denoted with the letters 'G' and 'S' respectively. The result of planning is a list of action links that link the perceptual clusters of start and goal location.

2.1 Representation of Perceptual Space

To establish an internal representation of the robot's perceptual space through learning, we used a self-organising feature map (SOFM). There are two reasons for this choice: i) clustering is unsupervised, and therefore does not require clustering teaching signals to be supplied, and ii) clustering is topological, thus counteracting the effects of perceptual noise.

Initially, during the robot's exploration and learning phase, the robot is left to explore its environment executing random motor actions from its behavioural repertoire and collecting perception-action-perception triples. The acquired sensory perceptions are fed into the SOFM and organised in increasingly clear defined clusters. When this phase is completed, raw sensory perception is 'encoded' in a topological fashion by the SOFM.

The weight vectors of the SOFM are initialised as pseudo-random vectors of unit length. As similarity measure we use the Euclidean distance between the input vector $x(t_k)$ and each of the weight vectors $m_i(t_k)$.

$$\|x(t_k) - m_i(t_k)\| \forall i \in Units$$

where $m_i = \{\mu_{i,1}, \mu_{i,2}, \dots, \mu_{i,n}\}$ is the weight vector of unit i , $x = \{\xi_1, \xi_2, \dots, \xi_n\}$ is the input vector, and n is the number of input signals. The weight update rule is:

$$\begin{aligned} m_i(t_{k+1}) &= m_i(t_k) + a(t_k)[x(t_k) - m_i(t_k)] \text{ for } i \in N_c \\ m_i(t_{k+1}) &= m_i(t_k) \text{ otherwise.} \end{aligned}$$

The learning rate a starts with value 1 and progressively decreases to 0, when training stops. $a(t_k)$ is the learning

rate at iteration k . The weights update neighbourhood radius N_c starts from $N_c = \frac{\sqrt{\text{Units number}}}{2}$ and decreases to $N_c = 1$.

After each update the weight vectors are normalised again in order to be unit vectors, using the relationships:

$$\begin{aligned} \|m_i\| &= \sqrt{\mu_{i,1}^2 + \mu_{i,2}^2 + \dots + \mu_{i,n}^2} \quad \forall i \in N_c \\ m_i &= \left\{ \frac{\mu_{i,1}}{\|m_i\|}, \frac{\mu_{i,2}}{\|m_i\|}, \dots, \frac{\mu_{i,n}}{\|m_i\|} \right\} \quad \forall i \in N_c. \end{aligned}$$

2.2 Representation of Behavioural Space

The robots used in our experiments were given a certain, defined behavioural repertoire of motor actions, e.g. 'move forwards', 'turn left' etc.

After the representation of perceptual space (see section 2.1) is completed, the robot uses the perception-action-perception triples acquired while exploring its environment — by executing corresponding actions — to train the behavioural space layer. Obviously, each action performed took the robot from an initial perceptual state to a resulting perceptual state.

Motor actions are therefore linked to perceptions encoded on the SOFM, and are consequently entered into the SOFM as directional action links between initial and final perceptual state labelled with the corresponding action (see figure 1).

In this manner, the robot develops a representation of perception-action-perception triples throughout the exploratory phase (which is terminated after a preset amount of exploratory actions has been performed).

2.3 The Action Selection Mechanism

Once the 'map' described in sections 2.1 and 2.2 is established, it is relatively straightforward to search the map for complete paths between the current (initial) sensory perception and an externally specified goal perception.

One major difficulty here is that either or both of the current physical location and the goal location may never have been visited before, meaning that no entry for them is found in the map. The way we tried to overcome this problem is by exploiting the topological nature of the map: we search for mapped similar (neighbouring) perceptual clusters, and use those for action planning.

The search mechanism used is one of diffusion. Placing an imaginary 'scent' at the start location and spreading it along all existing links pointing out of the start location, the 'scent' will eventually arrive at the goal location in perceptual space, indicating which neighbouring node to the goal (or the goal node itself) network node has a complete path from start to goal.

Once the robot has executed the first motor action of the plan, the planning algorithm as described above is invoked again. This is to avoid errors due to incorrect execution of a motor action (noise, etc).

3 Experiments

3.1 Experimental Method

Experiments were conducted with autonomous mobile robots. The robots were capable of moving in 2 dimensional metric space and had a repertoire of four distinct actions (e.g. 'move forwards 50 cm', 'move backwards 50 cm', 'turn left 40°', and 'turn right 40°'). During the experiments the robot's task was to stop at a specified position and orientation in its environment.

The experiments were divided into two distinct phases. During the initial *exploration phase*, the robot explored its environment performing predefined number of randomly selected actions out of its behavioural repertoire, and acquired (i) a subsymbolic representation of perceptual space, using the self-organising feature map (SOFM), and (ii) a representation of perception-action-perception associations. The self-organising feature map that we used was a two dimensional toroid with different size for each experiment, as described in the particular sections. The precondition-action-postcondition associations were stored as a list of labelled links (action links) between pairs of perceptual clusters of the SOFM.

In a subsequent *application phase*, the robot was initially placed at a 'goal' location (to obtain the sensory perception of the goal), then lifted to an arbitrary 'start' location and left to determine a sequence of motor actions that would take it to the goal. The robot achieved this through a mechanism of reaction-diffusion dynamics, as described in section 2.3. After finding a path to the goal the first of the actions of the plan was performed. Subsequently the planning process was repeated, until the current sensor perception activated the same perceptual cluster as the goal perception.

3.2 Experiments with Essex' Marvin Robot

Experimental Setup

In a first series of experiments we used an Essex' Marvin robot, which has eight ultrasonic range finders which were used as sensor inputs to the mechanism (figure 2).

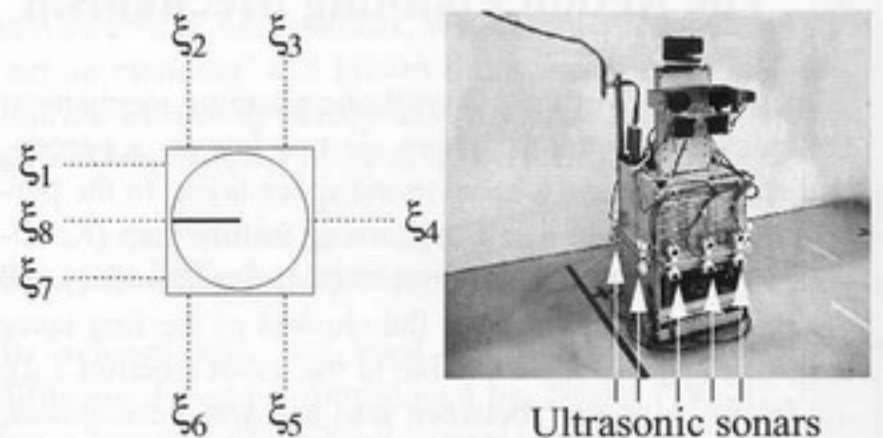


Figure 2: Top view and side view of the Essex' Marvin robot. The robot has eight ultrasonic sensors, as indicated.

The robot was placed in the experimental arena depicted in figure 3 and has a repertoire of four actions ('move forwards 50 cm', 'move backwards 50 cm', 'turn left 40°', and 'turn right 40°').

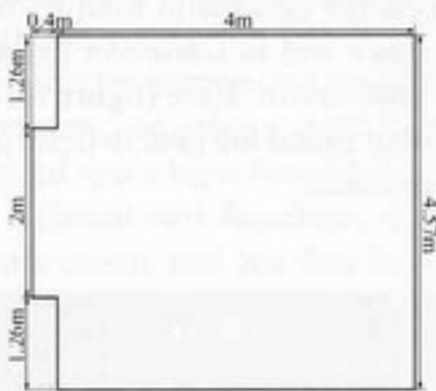


Figure 3: The experimental arena used during the experiments with the Essex' Marvin robot.

During the exploration phase, the robot explored its environment performing 294 randomly selected actions out of its behavioural repertoire. The self-organising feature map that we used was a two dimensional toroid with 7x7 units. During training each training example was presented to the SOFM 5000 times. The learning rate α starts from value 1 and decreases to 0 in steps of $\frac{1}{\text{Training Cycles}} = 0.0002$ after each training iteration. Therefore, after completing 5000 training cycles becomes 0 and the training stops. The weights update neighbourhood radius N_c starts from $N_c = 3$ and decreases to $N_c = 1$. The precondition-action-postcondition associations were stored *afterwards* as a list of action links (links labelled with the action taken) between pairs of perceptual clusters of the SOFM.

In the subsequent application phase, the robot was initially placed at a 'goal' location (to obtain the sensory perception of the goal), then lifted to an arbitrary 'start' location and left to determine a sequence of motor actions that would take it to the goal. The robot achieved this through a mechanism of reaction-diffusion dynamics, by which a marker propagated from the goal perceptual cluster *at even pace* in all directions until it reached the robot's current perceptual cluster. The sequence of actions that leads from the network node that is maximum activated when the robot is at the 'start' position to the node that corresponds to the 'goal' position will indicate the shortest path to the goal in terms of amount of actions — however, this does not mean that it will also indicate the shortest physical path to the goal. Nevertheless, as far as we are interested in *any* existing path to the goal, this shortcoming is not serious. As we will see in section 3.4 by introducing cost related to actions we can influence the planning to favour particular plans.

After finding a path to the goal the first of the actions of the plan was performed. Afterwards the planning process was repeated, until the current sensor perception activated the same perceptual cluster as the goal perception.

Experimental Results

An example of a typical experimental run is presented in figure 4.

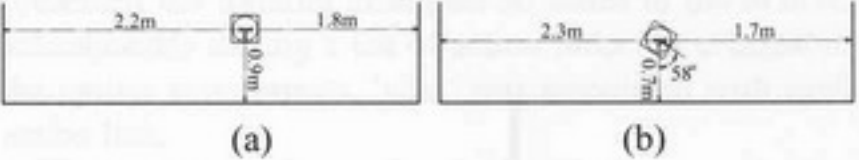


Figure 4: Typical task in 2 dimensional experiment: the robot is placed at the goal location first (b), then lifted to an arbitrary start location (a) and instructed to determine a sequence of action that will take it to the goal.

Start and goal location shown in figure 4 differ both by angle (0° and 58° resp.) and distance to the nearest wall (90 cm and 70 cm resp.). Figure 5 shows the sequence of actions performed by the robot in order to move from start to goal.

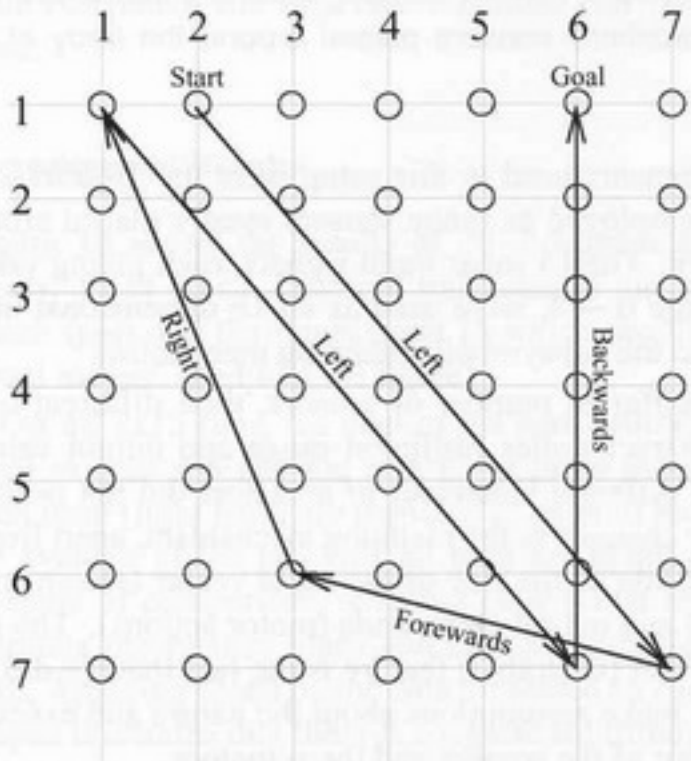


Figure 5: Sequence of actions (shown in the perceptual space of the SOFM) performed to solve the task depicted in figure 4. Note that the circles represent units of the SOFM and that the SOFM we use is toroid.

Looking at figure 5, it might appear that the action sequence forwards-right-left-backwards partially cancels the effect of individual motor actions, returning the robot to the original position. On closer inspection one finds that this is not true. In fact, this simple experimental run clearly illustrates that the robot learned its own behavioural idiosyncracies, in this case the asymmetry between supposedly symmetric actions such as 'left' and 'right'.

This realisation illustrates that the subsymbolic planning mechanism avoiding use of symbolic representations produces grounded plans (Harnad, 1990).

3.3 Experiments with a Magellan Pro

In a second set of experiments we transferred the subsymbolic action planning mechanism to a Magellan Pro robot (see figure 6), which has a larger and different sensor apparatus than the Marvin robot.



Figure 6: The Magellan Pro robot has sixteen ultrasonic sonar sensors, sixteen bumper switches, and sixteen infrared proximity sensors placed around the body of the robot.

The sensors used in this setup were the 16 ultrasonic sonars employed as range sensors evenly placed around the robot. The 16 sonar input signals, each giving values in a range 0 – 4, were used as an 16 dimensional input vector to the subsymbolic planning mechanism.

The different number of sensors, their different technical characteristics (different range and output values) and the different behaviour of actuators did not necessitate any changes to the planning mechanism, apart from a re-definition of the size of the input vector (sensor input signals) and output commands (motor actions). The reason for this (desirable) feature is the fact that we did not have to make assumptions about the nature and expected behaviour of the sensors and the actuators.

Experimental Setup

For the experiments presented here the experimental arena depicted in figure 7 was employed. It is a square area with side 2.80m and 4 cylindrical obstacles placed in order to give a variety of sensor perception through the field. During the training phase the robot selects random actions from its repertoire and performs them. Doing so it collects 400 training samples for training the subsymbolic planning mechanism. The SOFM we used was 9x9 units and toroid. During training each training example was presented to the SOFM 50 times. The learning rate α starts again from value 1 and decreases to 0 in steps of $\frac{1}{\text{Training Cycles}} = 0.02$ after each training iteration. The weights update neighbourhood radius N_c starts from $N_c = 4$ and decreases to $N_c = 1$.

Experimental Results

After training has been completed the application phase follows. In figures 7 and 8 one can see two different experimental runs of the application phase. The first one (figure 7) depicts the successful finding of the goal both in perceptual space and in Cartesian space. The second figure depicts another run. Here (figure 8), due to aliasing problem, the robot found the goal in perceptual space, but not in Cartesian space.



Figure 7: The positions and orientations of the Magellan Pro robot in the arena during one of the experiments. The environment is a square arena with side length 2.80m and 4 cylindrical obstacles placed inside it. On the left the robot is placed on the start position. In the middle the robot is placed on the goal position. On the right the robot is on the goal it finally found.



Figure 8: The positions and orientations of the Magellan Pro robot in the arena during another experiment. On the left the robot is placed on the start position. In the middle the robot is placed on the goal position. On the right the robot is on the goal it finally found. Specified and found goals are perceptually identical, which is the reason for the outcome of this experiment.

This second set of experiments confirms that the subsymbolic planning mechanism is able to learn the behavioural idiosyncrasies of the particular robot. Of course we should take into account that the specific hardware characteristics of a robot define and limit its abilities and suitability for a given task. This ability of the subsymbolic action planning mechanism to adapt to the idiosyncrasies of the particular robot makes it robust in real world applications, where sensors and motors do not have identical, predictable characteristics.

Furthermore, the experimental results presented in the two last sections vividly illustrate that the subsymbolic planning mechanism is embodied and the plans it creates are grounded (Harnad, 1990; Sun, 2000).

3.4 Introducing ‘Cost’ Functions on the Planning Mechanism

Being able to determine *any* sequence of actions to take the robot to the goal is a useful property. However, in practical applications it is desirable to favour certain solutions over others.

We achieved this by introducing the notion of ‘cost’ to the planning process, assigning a ‘cost’ to each action link of the behavioural space layer (see figure 1). Specifically, we used two different cost functions, one that favoured translational movement, and one that favoured rotational movement of the robot.

The robot aims to execute the path with the minimum cost every time.

Experimental Setup

The Magellan robot was used for this experiments, it is again capable of performing four actions: ‘move forwards 50 cm’, ‘move backwards 50 cm’, ‘turn left 40°’, and ‘turn right 40°’.

The robot is placed in front of a wall and the task is to obtain a specified distance and orientation. This environment was selected because it minimises the aliasing problem.

Initially, the robot performs random actions, from the four actions in its repertoire, moving in the arena and collecting 400 precondition-action-postcondition training data triples. This data was used for training the action planning mechanism. Drawings of the field we used for the experiments can be seen in figure 9.

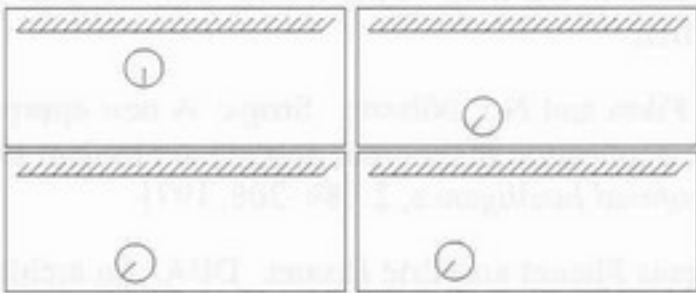


Figure 9: The positions and orientations of the robot in the field during the experiments with the two cost functions. On the top left and top right the robot is placed on the start and goal positions respectively. The bottom left shows the goal found using cost favouring translation, the bottom right depicts the goal found using cost favouring turning. In both cases the goal was reached successfully, albeit using different plans (see figure 10).

The sensors used in this setup are 16 ultrasonic sonars employed as range sensors evenly placed around the robot. The task for the robot was again to move to a specified position and orientation in the environment, using a repertoire of the four actions. A two dimensional toroid self-organising feature map of size 13x13 units was used for perception clustering. By using smaller SOFM the aliasing problem increases because there is not enough

resolution to the perception clustering. The learning rate a starts again from value 1 and decreases to 0 in steps of $\frac{1}{\text{Training Cycles}} = 0.02$ after each training iteration. The weights update neighbourhood radius N_c starts from $N_c = 6$ and decreases to $N_c = 1$. During training we presented the training examples 50 times to the SOFM, subsequently storing a list of action links. In contrast to the earlier experiments, ‘cost’ was associated with each action link.

The reaction-diffusion dynamics now incorporated the cost, in that the marker travelled more slowly along links with higher cost. During diffusion dynamics search the cost of an *action* link is translated to propagation time, and therefore the ‘scent’ arrives first at the goal along the ‘cheapest’ path. This shortest path is then a plan towards the goal and the first *action* of this plan is executed.

Once a link between goal and current perceptual cluster was found, the first action of the sequence was executed, and the entire planning process repeated until current and goal perceptual cluster were identical.

Costs are assigned to action links after training has been completed and thus cost functions can vary as desired.

Experimental Results

Figure 10 shows the results of 30 independent experiments, 15 of which favoured translational moves of the robot² (part *a* of the figure), and 15 which favoured rotational moves³ (part *b* of the figure).

For all 2x15 runs, the cost of the path followed (dark bars) in each run and the cost if the other measure had been used (light bars) are depicted in pairs of bars. Also the average of the costs of all runs are displayed as a measure of comparison, it shows clearly that the action planning mechanism plans and executes a minimal cost path. Applying t-tests on the data presented on each of the graphs illustrates that there is a significant difference between the path costs using the optimisation function and an alternative function ($p=0.0021$ and $p<0.0001$ respectively).

The experiments demonstrated that the robot, starting from the same location and having the same goal, is able to pursue the task with different plans each time. This is of great advantage because in real world where we should not assume a specific perceptual postcondition following the execution of an *action*. Planning now is more flexible and tolerant to environmental, sensor, and motor noise⁴. Therefore we have a robust system tolerant to noisy data during training and application, and designed to plan in the real world.

²Here the cost was ‘1’ for translational moves and ‘20’ for rotational ones.

³Here the same cost values were used, but with the allocation reversed.

⁴With the term *noise* here we mean all the unpredictability involved with operation in real world.

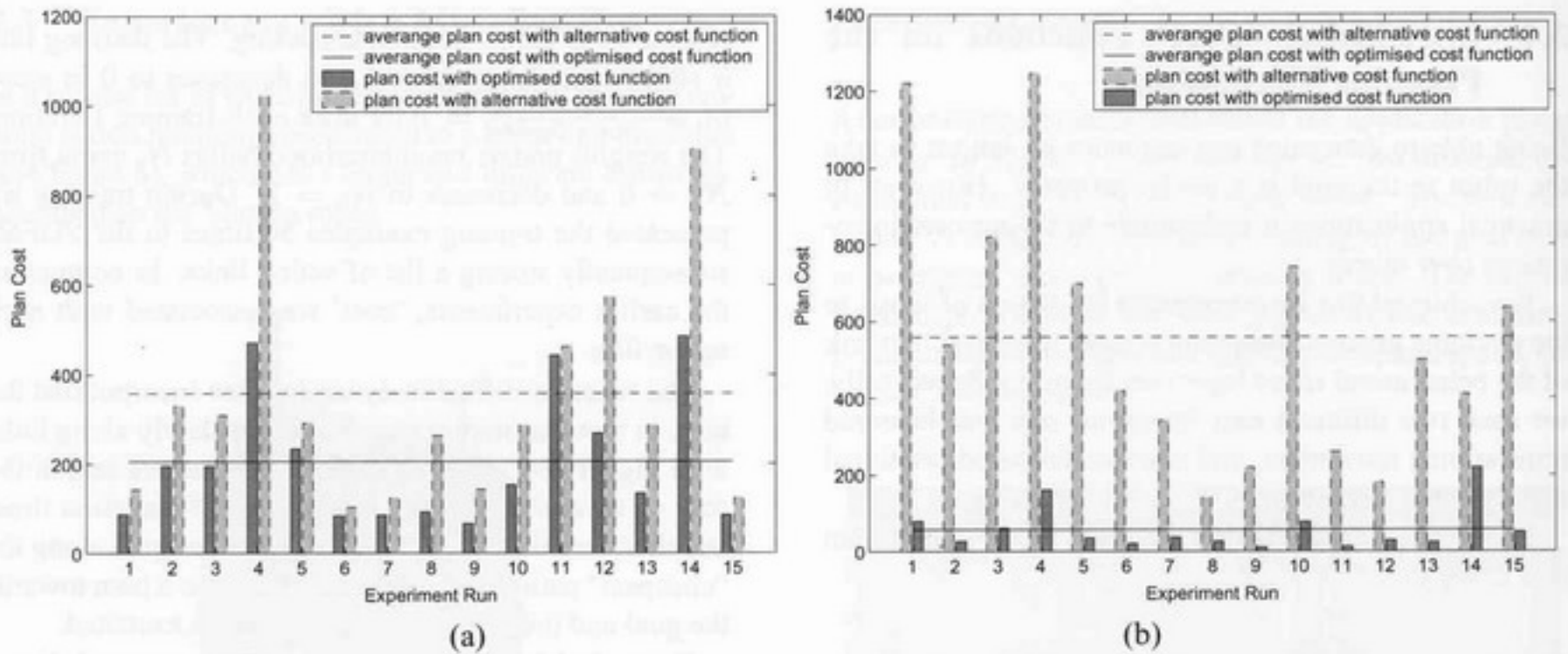


Figure 10: Comparison of optimised (dark bars) and alternative (light bars) path cost functions of 2 x 15 experimental runs. Figure (a) shows the optimisation favouring translational moves, figure (b) that for optimisation favouring rotational moves. Average costs for the optimised (solid line) and the alternative cost function (dashed line) are also displayed.

4 Discussion

The subsymbolic planning mechanism presented in section 2 was able to determine task-achieving sequences of actions, without using symbolic representations, but instead applying previously acquired knowledge in new ways, essentially ‘assembling’ novel plans from smaller components which had previously been learned (section 2.3).

Transferring the mechanism to another robot demonstrated that the subsymbolic planning mechanism does not depend on specific assumptions about the technical characteristics of the robot (section 3.3). Furthermore, subsymbolic planning proved to be capable to employ cost functions and plan optimum paths (section 3.4) with respect to different performance criteria (cost functions).

These experimental results illustrate that such a subsymbolic planning approach lead to an embodied robot planner that produces grounded plans.

One weakness of the proposed mechanism is that only plan components that have explicitly been encountered previously can become part of a plan. In other words, the mechanism as proposed here is unable to generalise regarding the behavioural space.

Ongoing work at Essex addresses this weakness, aiming to enable the robot to make plans towards and from areas of the perceptual space that the robot has not explored before.

Acknowledgements

John Pisokas is sponsored by IKY (State Scholarship Foundation of Greece), whose support is gratefully acknowledged.

References

- R. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.
- G. Baldassarre. A planning modular neural-network robot for asynchronous multi-goal navigation tasks. In *Proceedings of the 2001 Fourth European Workshop on Advanced Mobile Robots - EUROBOT-2001*, pages 223–230, 2001a.
- G. Baldassarre. *Planning with neural networks and reinforcement learning*. PhD thesis, University of Essex, 2001b.
- R.E. Fikes and N.J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.
- Francois Fleuret and Eric Brunet. DEA: An architecture for goal planning and classification. *Neural Computation*, 12(9):1987–2008, 2000.
- T. Fomin, T. Rozgonyi, C. Szepesvári, and A. Lörincz. Self-organizing multi-resolution grid for motion planning and control. *International Journal of Neural Sciences*, 7:757–776, 1996.
- E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings AAAI-92*, volume 4, pages 809–815, 1992.
- S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.
- Eric R. Kandel, James H. Schwartz, and Thomas M. Jessell, editors. *Essentials of neural science and behavior*. Appleton and Lange, Stanford, 1995.

- T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, 1984.
- C. Malcolm. Behavioural modules in robotic assembly. Lecture Notes, 2000. Division of Informatics, University of Edinburgh.
- Nils J. Nilsson. Shakey the robot. Technical Report 323, SRI International, 1984.
- Luc Steels. Steps towards common sense. In *Proceedings ECAI 1988*, pages 49–54, 1988.
- R. Sun. Symbol grounding: A new look at an old idea. *Philosophical Psychology*, 13:149–172, 2000.
- A. Turing. The chemical basis for morphogenesis. *Phil. Trans. Roy. Soc.*, 37:129–152, 1952.
- M. Zeller, R. Sharma, and K. Schulten. Topology representing network for sensor-based robot motion planning. In *Proceedings of the 1996 World Congress on Neural Networks*, pages 100–103. INNS Press, 1996.

Contributed Short Papers

Introduction

The paper presents a model of adaptive behaviour based on thresholds with feedback. The model is designed to simulate the behaviour of a system that can adapt to changes in its environment. The model is based on the following assumptions:

- The system has a set of parameters that can be adjusted.
- The system has a set of thresholds that determine when a parameter should be adjusted.
- The system has a feedback mechanism that allows it to learn from its past performance.

The model is implemented in a simple programming language. The results of the simulation are presented in the form of a graph. The graph shows the system's performance over time. The performance is measured in terms of the system's ability to adapt to changes in its environment. The results show that the system is able to adapt to changes in its environment and maintain a high level of performance.

Model Description

The model is a simple programming language. It is designed to simulate the behaviour of a system that can adapt to changes in its environment. The model is based on the following assumptions:

- The system has a set of parameters that can be adjusted.
- The system has a set of thresholds that determine when a parameter should be adjusted.
- The system has a feedback mechanism that allows it to learn from its past performance.

The model is implemented in a simple programming language. The results of the simulation are presented in the form of a graph. The graph shows the system's performance over time. The performance is measured in terms of the system's ability to adapt to changes in its environment. The results show that the system is able to adapt to changes in its environment and maintain a high level of performance.

Adaptive Behaviour based on Evolving Thresholds with Feedback

Tom De Wolf*, Tom Holvoet*

[Tom.DeWolf, Tom.Holvoet]@cs.kuleuven.ac.be

*DistriNet, Dept of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee (Leuven), Belgium

Abstract

In this paper, we propose a model for dynamic task selection based on an evolving threshold mechanism. The threshold evolves based on statistical reinforcement with experience the agent gets from positive and negative feedback. This gives a model where the task selection is tightly coupled with the real success of the tasks and with the perceptions of the agent in the form of stimuli. An example and experiment in the context of ant simulation shows that the agent adapts itself in a flexible way, and at the same time uses experience to achieve a simple kind of specialisation.

1 Introduction

In this paper we consider behaviour-based situated agents (Wooldridge and Jennings, 1995), (Wooldridge, 2002), (Ferber, 1999). An agent is an autonomous entity situated in an environment. An agent perceives the environment to a limited extent without maintaining an internal model of the environment. An agent is reactive to changes in this environment by adapting its behaviour to the current needs.

The behaviour of an agent is determined by the tasks the agent executes. A task is composed of actions the agent executes in a certain order, such as moving one step ahead or picking up an object. To make the behaviour of an agent adaptive we need *dynamic task selection*. Dynamic task selection allows agents to adapt their behaviour at run-time by selecting a new task. A task can not be too coarse-grained because of flexibility. For example when simulating ants, 'forage' is too coarse-grained where 'follow pheromone trail' and 'return food to nest' can provide the required flexibility.

In this paper we propose a model for dynamic task selection combining work from (Bonabeau et al., 1999) and (Maes and Brooks, 1990). Bonabeau et al. (1999) describe a evolving stimulus-threshold mechanism. Instead of just increasing and decreasing the threshold as in (Bonabeau et al., 1999), our model uses a measure for relevance of a task based on positive and negative feedback from (Maes and Brooks, 1990). Due to feedback the threshold evolves. Therefore our model is stronger related to the success of the behaviour of an agent. This model allows the agent to adapt itself in a flexible way based on its perceptions, and at the same time uses experience to achieve a simple kind of specialisation. The evolution of the threshold is related to sub-symbolic learning.

First we describe stimuli as the information that agents use for the dynamic task selection. Then the selection mechanism itself is explained. After that some experi-

ments are described and we end with a conclusion.

2 Stimuli

An agent selects a task based on information from the environment and based on experience. We call both forms of information *stimuli*. A stimulus has a certain type and an intensity to indicate how strong the stimulus is. Note that for each type of stimulus there can be multiple instances (e.g. multiple food items).

The first form of information are *external stimuli*. There are three kinds of external stimuli (Schelfhout and Holvoet, 2002): (1) environmental stimuli that are not put in the environment by agents (e.g. a food source), (2) direct stimulus by interaction with another agent (e.g. touching each other), (3) stigmergic (Grassé, 1959) stimuli, i.e. communication from other agents through the environment. An example of the latter in the context of ants are the pheromones that are laid down by one individual. The other ants may respond to these at a later time. In general we can see stigmergic stimuli as marks that agents leave behind for other agents.

The second form of information are *internal stimuli* that serve as an internal feedback mechanism, generated by the agent itself. While executing a task, the agent determines the type of stimulus (positive or negative feedback) and the intensity of these internal stimuli. The type and intensity of the feedback can vary at each step in a task. For example, when an agent is following a trail of pheromones a smaller intensity is set for the positive feedback when approaching the trail than when the agent is already on the trail. A positive feedback reflects a partially or fully successfully executed task and it is useful to choose this task again. A negative feedback can be generated to indicate that the task was not appropriate or failed to execute. These stimuli are taken into account the next time the agent must select a task to execute.

We use the notion of *relevant stimuli for a task* to describe which stimuli influence which tasks. The more relevant stimuli present at a certain moment in time for a task and the higher their intensity, the higher the probability that the task will get active.

3 The Selection Mechanism

An agent uses the selection mechanism each time the agent has finished executing one primitive action of a task. This way the agent can adapt after every primitive action. As input the mechanism gets the current stimuli¹ and as output the mechanism gives the task that is most appropriate at this moment. If this is the same task as the previous one the agent remembers how much of the task was executed already and continues to complete the task with an extra step, the next primitive action. If it is another task, this task is executed from the beginning.

We use a response threshold model to decide which task is most appropriate for the agent. As shown in (Bonabeau et al., 1999; Schelfhout and Holvoet, 2002), threshold models are very adaptive and give good results. On the one hand, a *threshold value* is associated with every task. On the other hand, a *stimulus value* is computed for each task, based on the current relevant stimuli. Thresholds refer to likelihood of reacting to relevant stimuli for that task. Low-threshold tasks are performed at a lower level of the stimulus value than high-threshold tasks. The specialisation in a certain task stems from the evolution of thresholds according to received feedback stimuli from previous experience.

Defining the mechanism requires an answer to the following questions: (a) How do we compute the thresholds, and how do the thresholds evolve? (b) How do we compute the stimulus value? (c) How do we choose the most appropriate task based on those values? We elaborate on each question in the following paragraphs.

3.1 Evolution of Thresholds

The evolution of thresholds generates differentiation in task performance in a multi agent system composed of initially identical entities. Task allocation in this case is emergent and more robust with respect to perturbations of the system than when the thresholds are fixed. In Bonabeau et al. (1999) a threshold is increased when the task is not active and decreased when active. The authors do not consider feedback. This way a threshold can get saturated and a task that is active but fails still gets a positive effect in the form of a decreasing threshold.

Our approach introduces the concept of a *statistical reinforcement* to compute the threshold values. The method is based on positive and negative feedback stimuli in relation to the fact if a task is active or not. Maes and Brooks (1990) propose a method to calculate the *relevance* of a

task at a certain moment. A relevant task is defined as 'a task that is positively correlated to positive feedback and not positively correlated to negative feedback'.

Table 1: Statistics for one task

	Active	Not Active
Positive Feedback	j	k
No Positive Feedback	l	m
Negative Feedback	jn	kn
No Negative Feedback	ln	mn

Table 1 shows the statistics that are maintained for each task. For example, j is the number of times positive feedback occurred when the task was active. All the values are initialised at a certain value X (for example X=10). Each time the statistics are updated all numbers are multiplied with $X/(X+1)$. This multiplication ensures that the impact of past experiences on the statistics is less than the impact of more recent experiences and the form of the factor has a smooth evolution as result. For example, if a positive feedback occurred when the task was active j is increased with 1 after the multiplication, so $j' = (j_{old} * X/(X+1)) + 1$. A value like k that is not increased gets the value $k_{old} * X/(X+1)$. X acts as a parameter to control how adaptive the behaviour of the agent is. The agent adapts more quickly for a small value of X, and slower when it is large. Also, this multiplication avoids saturation of the thresholds.

The relevance of the task at a certain moment in time is computed as follows:

$$relevance = (corr(P, A) - corr(N, A)) \quad (1)$$

Here, N stands for negative feedback and $corr(P, A)$ is the correlation between occurrence of positive feedback and the fact that the task is active. This correlation is computed as shown in equation (2).

$$corr(P, A) = \frac{j * m - l * k}{\sqrt{(m + l) * (m + k) * (j + k) * (j + l)}} \quad (2)$$

This is a statistical measure (the Pearson product-moment correlation coefficient) for the degree to which the status of the task (active or not active) is correlated with the positive feedback and ranges from -1 to 1. Similarly for $corr(N, A)$. The relevance in (1) then ranges from -2 to 2 meaning that the task is very relevant at that moment when it is close to 2 and not relevant at all when close to -2.

In our mechanism the relevance is reflected in the threshold. The threshold is low when the relevance is high. The threshold is expressed in units of stimulus intensity to allow a comparison with the stimulus value. Because we express the stimulus value (see below) or the intensity of the relevant stimuli as a percentage, we also want the thresholds to range from 0% to 100%. The

¹ perceptions, internal feedback, ... that are present at this moment

computations are shown in equation (3). Now we have a threshold that evolves based on the experience the agent gets, i.e. the statistics in Table 1 are updated based on feedback stimuli.

$$threshold = |(relevance + 2) * 25 - 100| \quad (3)$$

3.2 Computation of Stimulus Values

The other value we use in our approach is the *stimulus value*. This value is a measure for the amount of relevant stimuli for a task at a certain moment in time. Multiple relevant stimulus intensities must be combined into one value. We assume that all relevant stimulus types are equally important for a task. The strength of the stimulus value must reflect the presence of all relevant stimulus types and the intensity of each of them. An approach to do this is to use a percentage value. To calculate this percentage, we use a heuristic that only considers the strongest instance of a stimulus type and that heuristic ignores the number of instances (other approaches are possible):

We can compare the stimulus value with a large bottle that has to be filled. A full bottle then corresponds to a stimulus value of 100%. As said before, each task has a number of relevant stimulus types. Now let us divide the bottle in as much equal parts as there are relevant stimulus types. For each relevant stimulus type we search for the instance with the strongest intensity in the currently perceived stimuli. This intensity also ranges from 0% to 100%. We then take the part of the bottle that corresponds with this stimulus and fill it for as much percent as indicated by the intensity of the strongest stimulus of that type. This way the bottle gets filled until a certain level and the value that corresponds to that level is the stimulus value for that task.

3.3 Making an optimal decision

A final issue that needs explanation is how the threshold is compared with the stimulus value, yielding, for each task, a probability of becoming active. For each task a probability is computed as in equation (4), where s is the stimulus value and th is the threshold.

$$probability = \frac{s^2}{s^2 + th^2} \quad (4)$$

This probability ranges from 0 to 1. When the stimulus value exceeds the threshold value the probability will be high and close to 1. When the stimulus value does not exceed the threshold, the formula makes sure that when there are stimuli for the task, this task still has a probability to become active but it is much smaller and close to 0. When the stimulus value equals the threshold the probability is 1/2. Therefore, individuals with a lower threshold are likely to respond to a lower stimulus value. A task is chosen to execute with respect to these probabilities. We can choose the task with the highest probability or we can

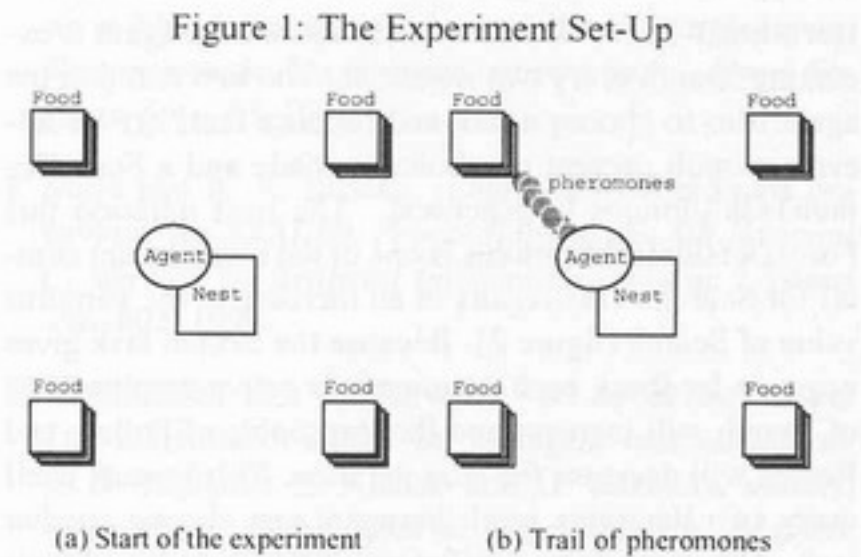
choose one of the tasks probabilistically where equation (4) gives the probability of being chosen. The latter is done in Bonabeau et al. (1999) and adds some randomness to the mechanism. In the experiments that follow, we take the task with the highest value for equation (4). This way we can analyse the behaviour of an agent in more detail because we then know what should happen.

4 Experimental results

In this section we first describe the experiment set-up and then the results are given and discussed.

4.1 Experiment Set-Up

The goal of this experiment is to show that the proposed dynamic task selection approach gives very adaptive agents. We have made an ant-simulation and in this experiment we consider a single agent or ant in terms of the stimulus value, the evolving threshold and the resulting probability. The ant starts from the nest and at an equal distance there are four piles of food, 10 food items each. The initial situation is shown in Figure 1(a). We used 4 piles to make sure that the agent will find one of them in a relatively short time. The results that are described later limit themselves to getting one of those piles completely back to the nest. This way we can see the agent finding the food, returning it to the nest while laying a pheromone trail and following the trail back to the food to get another food item. Figure 1(b) shows a trail from the nest to the food pile.



The role of the ant in this experiment is to 'Forage' and the way we have split this role up into flexible tasks is shown in Table 2. Each task does one action each execution cycle. For example, for Search each cycle the agent moves one step in a random direction. Table 2 also shows the relevant stimuli for each task that are used to calculate the stimulus value. In the description column the task is roughly described and you can see that it depends on the

Task	Relevant Stimuli	Description
Search	PositiveFeedbackStimulus, ForceDefaultTaskStimulus	The agent walks around randomly, searching for food. Always returns negative feedback because it is better to do other things when possible.
Return	PositiveFeedbackStimulus, FoodStimulus	When there is food, the agent approaches the food (=positive feedback 50%) and then picks it up and returns to the nest while laying a trail (=positive feedback 100%). A negative feedback is given at the end, when the nest has been reached to indicate that it is now better to do other tasks.
Follow	PositiveFeedbackStimulus, FoodPheromoneStimulus	When there is FoodPheromone it approaches the strongest pheromone and follows the trail of that pheromone towards the food (= positive feedback 100%). A negative feedback is given at the end of the trail to indicate that the trail has been followed completely and thus other tasks may be more appropriate.

Table 2: The tasks for the Forage role and their relevant stimuli

stage of the task what feedback is given to the agent for adjusting its thresholds.

The results shown here are also valid in the case of multiple agents because interactions are considered as stimuli and the experiment shows that the task selection mechanism reacts very well on stimuli, and thus also on interactions. In the case of a conflict, e.g. picking up the same object, one of the two agents will be first and the other agent sees that a stimulus is not there anymore.

4.2 Results

Figures 2,3 and 4 show the evolution of the stimulus value, the threshold and the resulting probability in that order. On the horizontal axis you can find the iteration number. In the following detailed description of the experiment we refer to these numbers.

Iteration 1-272. The first 272 iterations the agent is executing Search every two iterations. The first iteration the agent tries to choose a task and because there are no relevant stimuli present no choice is made and a ForceDefaultTaskStimulus is generated. The next iteration this ForceDefaultTaskStimulus is one of the two relevant stimuli for Search. This results in an increase in the stimulus value of Search (figure 2). Because the Search task gives negative feedback each iteration it is active the threshold of Search will increase and the thresholds of Follow and Return will decrease the next iteration. This repeats itself every two iterations until the agent can choose another task. The result is an oscillation of the stimulus amount of Search (figure 2), the thresholds (figure 3) and the probability of Search (figure 4).

Iteration 273-285. At iteration 273 the food pile is in the limited vision of the agent. This means that there are 10 instances of FoodStimulus and these are relevant for Return. The next iteration Return is executed by moving one step towards the food and this gives a positive feedback of 50% intensity. These two iterations (273-274) result in an increasing stimReturn line in Figure 2

until 62.5% and the positive feedback also has an influence on the thresholds (figure 3): thSearch and thFollow increase because they are less relevant and thReturn decreases. This is also reflected in a higher probability for Return (figure 4).

The agent continues to approach the food and reaches it at iteration 285. At that iteration it picks up one Food item and increases the intensity of the returned positive feedback to 100%. This results in a increase of the stimulus value for Return at iteration 285 in Figure 2, a little increase in probability in Figure 4 and the threshold is not influenced by the intensity of the feedback and gets the normal increase as a result of the presence of the positive feedback.

Iteration 286-298. After picking up the food the next step in Return is to return to the nest and lay a trail of pheromones while returning. The agent starts with laying pheromones at iteration 286 as shown in the increase of the stimulus value for Follow in figure 2 (FoodPheromoneStimulus is relevant for this task). Because the threshold of Follow has increased due to the positive feedbacks while it was not active this stimulus value is not strong enough to exceed its threshold and the probability increases but stays close to 0. This way the agent continues to lay its trail to the nest.

At iteration 297 the agent loses sight of the food and so the stimulus value of Return decreases. At iteration 298 the agent reaches the nest, puts down the food in the nest and generates a negative feedback to indicate that Return is completed and probably not useful anymore. The stimulus value for Return decreases, the threshold for Return increases and the thresholds for Follow and Search decrease. This gives a higher probability to Follow (figure 4) and the agent starts following the trail that is present.

Iteration 299-311. At iteration 299 Follow generates positive feedback for itself because the ant is following the pheromones successfully. This gives an increase in the stimulus value (figure 2), an extra decrease in the thresh-

old (figure 3) and an extra increase in the probability (figure 4) for Follow. This also gives an increase in the thresholds of the other two tasks. At iteration 300 the ants can see the food pile again so there is an increase in stimulus value and in probability for Return as shown in figures 2 and 4.

Until iteration 310 the agent continues to follow the trail. In the mean time, figure 2 shows that the stimulus value for Follow decreases gradually anyway and this is a result of the evaporation of the pheromones. The pheromones disappear gradually so old information disappears, eventually old paths to food will disappear. At iteration 310 the end of the trail is reached and results in no more positive feedback (decrease in stimulus value) and in negative feedback for Follow (increasing threshold and decreasing probability). Because of this the thresholds of the other two tasks decrease and because there are relevant stimuli for Return (FoodStimulus) the present probability increases even more. This results in the execution of Return starting from iteration 311.

Iteration 312-End. From this point on the whole story starts again and repeats itself 10 times, one for each food item, until iteration 550. At that point the last food item has been returned to the nest and because the nest is reached a negative feedback is generated for Return. Then the only stimuli present are FoodPheromoneStimuli from the remaining trail and thus Follow is executed. At the end of the trail (iteration 558) a negative feedback is generated so the stimulus amounts decrease, the threshold increases and the probability decreases. Again the only stimuli are FoodPheromoneStimuli and Follow is executed again. The agent repeats this until the trail has disappeared completely and this results in an oscillation of the stimulus value for Follow, the thresholds and the probability for Follow. At iteration 556 the trail is completely gone and no stimuli are left. A ForceDefaultTaskStimulus is generated and again the agent starts executing Search every two iterations.

5 Conclusion

In this paper, we propose a model for dynamic task selection based on a threshold mechanism. Statistical reinforcement makes the threshold evolve. This approach is based on (Bonabeau et al., 1999) but the threshold evolution is based on positive and negative feedback, as described in (Maes and Brooks, 1990). The added value with respect to (Bonabeau et al., 1999) is that the threshold is more tightly coupled with the real success of the current task. The added value with respect to (Maes and Brooks, 1990) is that we compare the relevance/threshold with a stimulus value based on the agents perceptions.

Some issues are left open in this paper. A first issue to consider is how to specify what tasks are for a certain problem. In this paper some guidelines were given but

we have no intention to state that this is the best granularity. A second issue is related to the calculation of the stimulus value. Future work must be done on how to incorporate the number of instances of a stimulus type into the value.

Nevertheless, as experiments show, it works very well. De Wolf et al. (2002); De Wolf and Jaco (2001-2002) show that we can even use this approach in an hierarchical setting where we first choose the wanted higher level task (Forage, Nurse, ...) and then choose within that higher level task the wanted lower level task (Search, Follow, ...). The advantages of this model are that the agent can adapt itself in a flexible way, and at the same time use experience to achieve a simple kind of specialisation.

References

- E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.
- T. De Wolf and L. Jaco. Multi-agent systems: a world for ants. Master's thesis, Department of Computer Science, KULeuven, 2001-2002.
- T. De Wolf, L. Jaco, T. Holvoet, and E. Steegmans. A nested layered threshold model for dynamic task allocation. In M. Dorigo et al., editor, *Proceedings ANTS 2002, LNCS 2463*, pages 290-291, 2002.
- J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison Wesley, 1999.
- P.P. Grassé. La reconstruction du nid et les coordinations inter-individuelles chez *bellicositermes natalensis* et *cibutermes* sp, la théorie de la stigmergie: Essai d'interprétation des termites constructeurs. *Insect Sociaux*, 6:41-83, 1959.
- P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *AAAI-90, Proceedings of the 8th National Conference on Artificial Intelligence*, volume 2, pages 796-802, 1990.
- K. Schelfhout and T. Holvoet. To do or not to do: The individual's model for emergent task allocation. In D. Kudenko E. Alonso and D. Kazakov, editors, *Proceedings of the AISB '02 Symposium on Adaptive Agents and Multi-Agent Systems*, pages 111-115. The Society for the Study of Artificial Intelligence and Simulation of Behaviour, 2002.
- M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2002.
- M. Wooldridge and N.R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), June 1995.

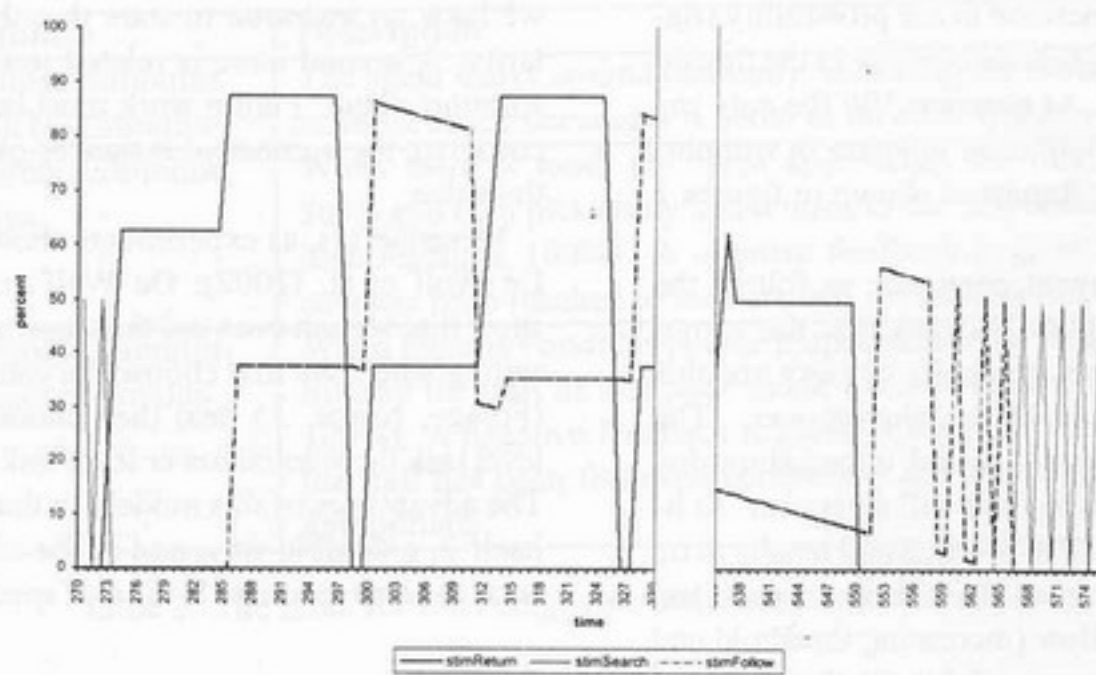


Figure 2: Stimulus Amount over time

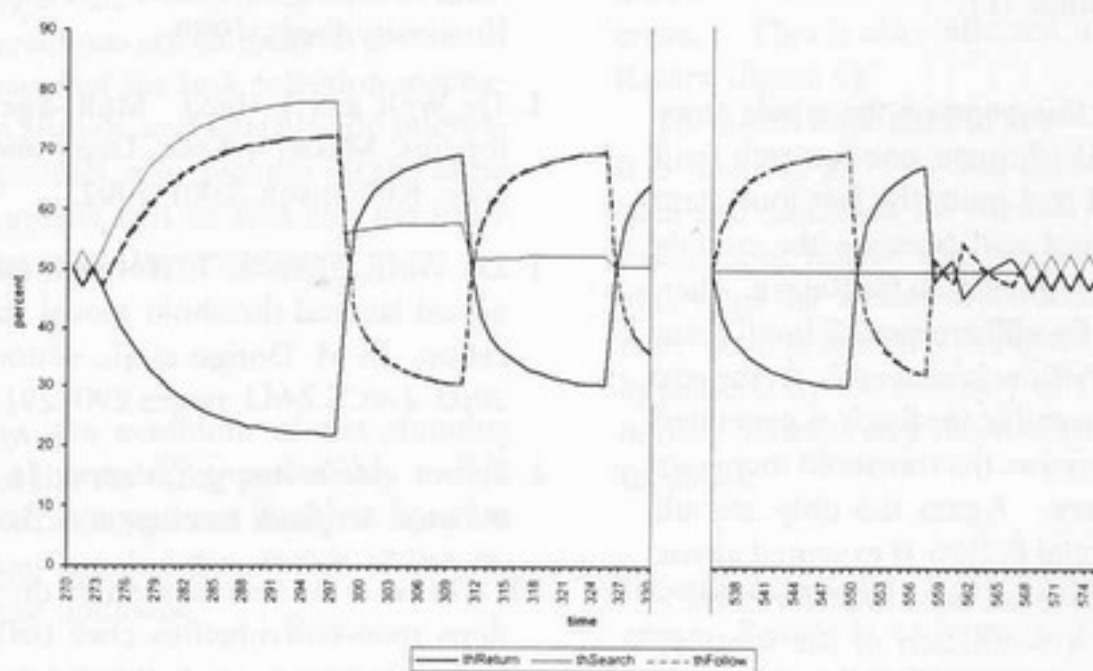


Figure 3: Threshold over time

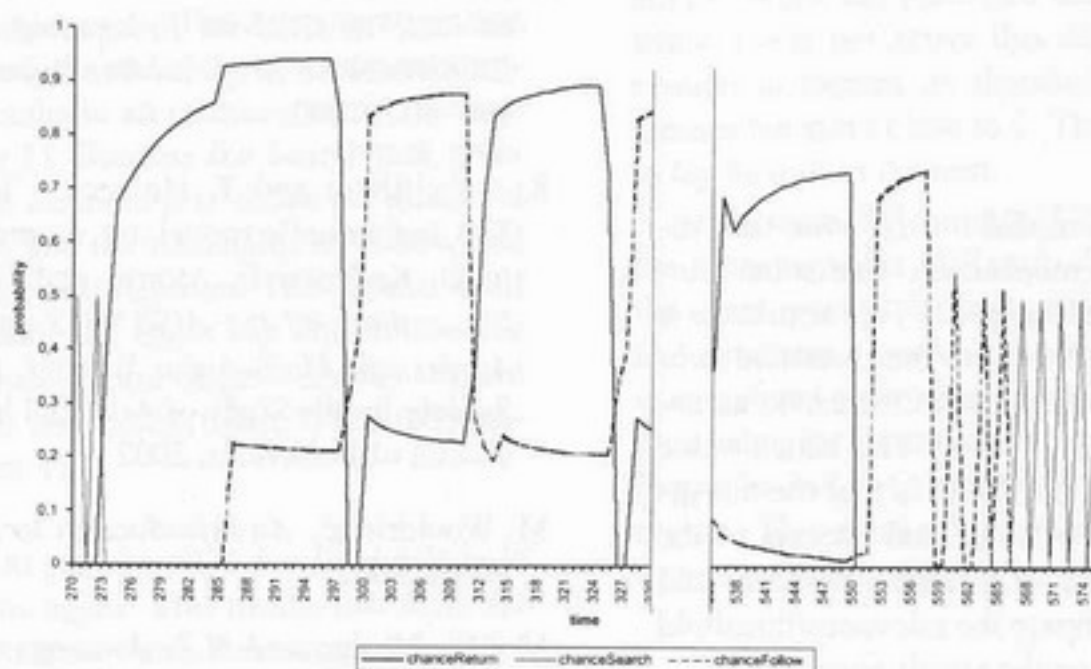


Figure 4: Probability over time

Coordinated Exploration in Stochastic Common Interest Games

Katja Verbeeck

*Vrije Universiteit Brussel (COMO)
Pleinlaan 2 1050 Brussel, Belgium
kaverbee@vub.ac.be

Ann Nowé

†Vrije Universiteit Brussel (COMO)
Pleinlaan 2 1050 Brussel, Belgium
asnowe@info.vub.ac.be

Karl Tuyls*

†Vrije Universiteit Brussel (COMO)
Pleinlaan 2 1050 Brussel, Belgium
ktuyls@vub.ac.be

Abstract

Recently we proposed a new exploration technique for individual reinforcement learners, which helps them to coordinate on the Pareto Optimal Nash equilibrium of a game. This technique in which agents may exclude one or more of the actions from their action space, can be seen as a discrete version of the traditional ϵ -greedy exploration technique. In this paper we refine this exploration technique further, with a standard technique from general search problems, i.e. random restarts. Due to this refinement, we are able to prove convergence to the Pareto Optimal Nash equilibrium in general stochastic common interest games. Moreover communication becomes unnecessary. Experiments show this technique on 2 challenging test problems and examine its use in larger joint action spaces.

1 Introduction

Common Interest Games are a special case of multi-agent systems (MAS), in the sense that the individual utility of the agents coincides with the joint utility of the group and that the system has only one state. As this reduces some of the difficulties individual reinforcement learning agents experience in a multi-agent environment, such as not knowing the others payoff and having delayed rewards, convergence to the optimal Nash equilibrium is still not guaranteed in the general stochastic case, see Claus and Boutilier (1998); M.Lauer and M.Riedmiller (2000); S.Kapetanakis and Kudenko (2002).

The main problem is that in games and MAS in general, agents influence each other. Values or rewards are sensed for combinations of actions of different agents, and therefore agents have to learn in a product or joint action space. One way to handle this is to model explicitly the presence of other agents. However as long as the agents are not fully informed this challenges some important principles of multi-agent systems, like distributed control, communication costs etc. Since most of the time agents are not fully informed; they only have limited information about each other and the overall system. They have to base their decisions on partial information regarding the state of the system and/or on some private local knowledge. This may lead the system to local optimality instead of global optimality or even to oscillations.

Recently we formulated a new exploration technique for uninformed agents to overcome some of the problems with learning in the joint action space. The general idea is that agents explore by excluding some of their actions now and then, to give the team of agents the opportunity to look for better solutions in a reduced joint action

space. Some variants of the exploration technique mentioned, were reported in Nowé et al. (2001); K.Verbeeck et al. (2002).

In this work, we focus on coordinated exploration that allows independent, uninformed agents to reach an optimal solution in repeated common interest games. In common interest games individual rationality agrees with group rationality and a Pareto optimal solution exists¹. In K.Verbeeck et al. (2002) convergence for the exploration technique was only shown for special common interest games, i.e. diagonal games. Here we extend the exploration technique with an idea from traditional random search techniques, i.e. random multi-starts. Due to this refinement we are able to prove convergence for general, stochastic common interest games. Moreover communication, which was already limited in K.Verbeeck et al. (2002), is now no longer needed.

This paper is organized as follows. In the next section the exploration technique of K.Verbeeck et al. (2002) is explained for stochastic common interest games. The refinement is given, together with its convergence result. In the third section some benchmark coordination problems, i.e. the climbing and penalty game are solved (the stochastic versions). Furthermore an analysis and comparison is given on the use of the different exclusion techniques in larger joint action spaces. Finally the last section discusses related and future work.

2 Exploring by Excluding

Our new exploration technique, of which variants can be found in Nowé et al. (2001); K.Verbeeck et al. (2002), is used by the agents in periods of play. At the beginning of

* Author is a Ph.d. student, funded by a doctoral grant of the institute for advancement of scientific technological research in Flanders(IWT)

¹ An outcome of a game is said to be *Pareto optimal* if there exists no other outcome, for which all players simultaneously do better.

learning, agents behave selfish or naive; i.e. they ignore the other agents in the environment and use reinforcement learning to maximize their individual payoff. The first period of play ends² when the agents have found a Nash equilibrium. This will happen because it is assured by results from learning automata theory, see Narendra and Thathachar (1989). Which Nash equilibrium the agents find is not known in advance, it depends on the initial conditions and the basin of attraction of the different equilibria.

At the end of a period agents decide whether or not to shrink their action space, by excluding the last action which they converged to. We see this exclusion-decision as some form of exploration decision. After exclusions took place, individual values are initialized and selfish or naive learning restarts in possible smaller joint action space. A new Nash equilibrium is converged to and again exploration may take place. The more Nash equilibria are found at the end of the simulation, the better the agents can select the best one.

As the learning automata result, mentioned above, also holds for stochastic games, the techniques described here will also extend to stochastic games. In a stochastic game the payoffs for the agents are probability functions of success or failure.

Compared to classical exploration techniques, such as ϵ - greedy techniques, this exploration technique can be seen as a discrete version of it. Instead of gradually giving an action less chance of being chosen, here actions are completely excluded. Afterward, they may become again available for selection.

A first exclusion technique for common interest games, which was reported in K.Verbeeck et al. (2002) is given in the next subsection. Some refinements are added in the second subsection, which allows us to say more about the convergence results of the technique.

2.1 One Exclusion and Communication

The general solution given in K.Verbeeck et al. (2002) for the exploration decision the agents have to make after every period of play, is given in Figure 1. The agents decide independently whether they have the need to explore any further. Their decision is based on how good the last action they converged to is, opposed to the best result the agent ever received during play. The parameter ϵ is the probability that an agent will exclude the current played action. This parameter depends on how good the current played Nash equilibrium is compared to the best outcome the agent ever received. It also depends on the experience the agent already has. When this action was already played often, more belief is attached to the learned value.

After decisions are made, the agents communicate it to each other. When more than two agents³ are willing to

```
## EXPLORATION PHASE
if (time_to_synchronize) {
  communicate_to_all ( explore_decision);
  decisions_all := receive_from_all (explore_decision);
  if more-than-two (decisions_all) {
    if (my_explore_decision = explore)
      my_actionSet :=
        my_original_actionSet - last_action;
    uniform_initialization_of_prob_not_excl_actions;
  }
}

## DECISION PHASE
{
  visits := nr-of-visits (last_action);
  alpha := (1 - (1 : (1 + visits)));
  epsilon := (* alpha ( average(last_action) :
    best_payoff_so_far));
  my_explore_decision := generate_decision (epsilon);
}
```

Figure 1: Pseudo code of the exploration and decision phase for the one-exclusion technique with communication.

explore, the agents who wanted to exclude the last action played, actually do it and independent learning can restart in a new subspace of the joint action space.

Important to note is that in this algorithm, an agent never excludes more than one action. The reason for this is to make sure that no Nash equilibria is cut off forever⁴. Therefore initialization takes places after every new exploration phase.

Although experimental results prove the usefulness of the method, see K.Verbeeck et al. (2002), no convergence results are available.

2.2 Many Exclusions and Multi-starts

As it would shrink the action space considerable more when multiple exclusions are allowed, the new idea is to use multi-starts to assure that every Nash equilibrium can be reached. Therefore the new exploration phase we propose is found in Figure 2.

After every period of selfish play, all the agents will exclude the action belonging to the Nash equilibrium they converged to, without any communication. In only a few periods of play the joint action space will have been shrunk considerably. Off course a lot of Nash equilibria may be cut out of the joint actions space. When the agents have only 2 actions left in their action space, they repair their original action space and put themselves at a random place in the joint action space. Exclusions can

²For the time being, period lengths are well chosen constants.

³Because of the definition of a Nash equilibrium, minimal two agents are needed to leave a Nash equilibrium.

⁴In case of 2-player games, Nash equilibria are not co-located on the same hyperplane and by excluding the actions of a Nash equilibrium no other equilibria disappear. This type of games was called diagonal games in K.Verbeeck et al. (2002)

```

## EXPLORATION PHASE
if (time_to_synchronize) {
  if (last_payoff(last_action) more_than
    best_payoff_so_far(last_action)) {
    best_payoff_so_far(last_action) :=
      last_payoff(last_action);
  }
  if (more_than_2_actions(my_actionSet) {
    my_actionSet :=
      my_actionSet - last_action;
  }
  else
    my_actionSet :=
      my_original_actionSet;
  random_initialize_prob_of_not_excl_actions;
}

```

Figure 2: Pseudo code of the exploration phase for the many-exclusion technique with restarts.

start all over again, and a different path through the joint action space is followed. For every action converged too, the maximal payoff ever received is sampled and remembered. After convergence took place, the agents still have time until the end of the period, to learn the reward associated with the Nash equilibrium they converged to. In our settings, they have to learn the probability on success, i.e. on average how many times a reward of 1 is received. By taking many restarts, different paths through the joint action space will be followed and a lot of information on the actions belonging to Nash equilibria will be gathered and the best one can be selected.

2.3 Convergence

With this technique we will be assured that if every Nash equilibria in the joint action space is visited infinitely often, in the same sense as every state-action pair is visited infinitely often in Q-learning, see Watkins and Dayan (1992), eventually the agents will convergence to the Pareto Optimal Nash equilibrium.

Theorem 1 *Under the assumption that the period lengths of selfish play are sufficiently long to give the agents the opportunity to converge to a Nash equilibrium and to let them approximate the payoff information of that Nash equilibrium accurately, and under the assumption that runtime is sufficiently long for the agents to visit enough Nash equilibria, agents using the many-exclusions, multi-start exploration technique as described in section 2.2 will converge to the Pareto optimal Nash equilibrium in common interest games.*

Proof: After playing period t agent j updates its value-information *best_payoff_so_far*, (noted q for short) for

the action $a_t^{(j)}$ it converged to as follows:

$$q_{t+1}^{(j)}(a) = \begin{cases} q_{t+1}^{(j)}(a) & a \neq a_t^{(j)} \\ \max\{q_t^{(j)}(a), \text{average}(u_t)\} & \text{otherwise} \end{cases}$$

Where u_t is the Nash equilibrium to which the agents converged to in period t , and from which action $a_t^{(j)}$ is a part of. *average*(u_t) is the sampled average payoff for equilibrium u_t . After sufficient periods are played and enough equilibria are found, the $q_t^{(j)}(a)$ values reflect the average payoff of the best Nash equilibria, to which action a belongs. The agents will then be able to pick the best performing Nash equilibria as long as the average values approximate the real payoff values of these equilibria close enough. This will be the case when the period lengths are sufficiently large. q.e.d.

The key solution is that the updates happen only when the agents know they are playing a Nash joint action, after a period of play is finished. Instead of updating at the level of every play, updates happen at the level of periods of play. So what actually happens is a kind of optimistic Q-learning but only on the actions which belong to Nash joint actions. In M.Lauer and M.Riedmiller (2000) optimistic Q-learning happens for every action selection or joint action. This works fine on deterministic reward or payoffs, however fails for stochastic payoffs. Here the period lengths must be set so that the average values can be accurately estimated.

3 Experiments

In this section we demonstrate the use of the exclusion-exploration technique. We first show how it behaves on two challenging games for the study of learning coordination in multi-agent systems. Then we analyze and compare the many-exclusion with multi-starts technique from section 2.2 in games of different joint action space-size with the one-exclusion technique from section 2.1. We also tested the technique of using only random restarts after every period of play. Using nor exclusions, nor random restarts, would result in independent agents using ordinary reinforcement learning of which is known that they will converge to a Nash equilibrium, however not necessarily the optimal one, Claus and Boutilier (1998); Jafari et al. (2001). Therefore we did not include independent reinforcement learners in our analysis.

3.1 Testbed coordination problems

The climbing game and penalty game, see Figure 3 and Figure 4 were introduced by Claus and Boutilier (1998) and are two difficult problems from the viewpoint of coordination. In the first game the punishment for miscoordination in the neighborhood of the the optimal joint action (a_1, a_1) is extremely high, and therefore convergence to it is very difficult for the agents. In the penalty

game 2 different optimal joint actions co-exist, however when the agents each choose to play the other optimal action, this mis-coordination is punished by a penalty term k . In both games the agents may be tempted to play the safe, non-optimal actions, which is (a_3, a_3) for the climbing game and (a_2, a_2) for the penalty game

	a_1	a_2	a_3
a_1	(11, 11)	(-30, -30)	(0, 0)
a_2	(-30, -30)	(7, 7)	(6, 6)
a_3	(0, 0)	(0, 0)	(5, 5)

Figure 3: The climbing game from Boutilier (1999)

	a_1	a_2	a_3
a_1	(10, 10)	(0, 0)	(k , k)
a_2	(0, 0)	(2, 2)	(0, 0)
a_3	(k , k)	(0, 0)	(10, 10)

Figure 4: The penalty game from Boutilier (1999)

In Figure 5 you find the results for the stochastic versions of the climbing and penalty game. In the stochastic version, the rewards are normalized between 0 and 1. These reward are not used deterministically, but they give a probability on having success, i.e. reward 1 or not, i.e. reward 0⁵. But even if these games are played stochastically, both the one-exclusion technique with communication and the many exclusion technique with multi-starts work fine on these benchmark problems. Even using just multi-starts without exclusions gives good results. In general all 2-players games are easily handled by exclusion techniques, because exclusions do not cut away other Nash equilibria, see K.Verbeeck et al. (2002), except when multiple optimal joint actions exist. However the restarts assure us that all equilibria are visited.

For the climbing game, the high negative reward is avoided due to excluding the second action a_2 , or even by initializing the action probabilities so that the first action a_1 gets a high probability. The latter is what the restart may do. For the penalty game, there is no mis-coordination between different optimal joint actions, as for all 3 techniques used the agents will play the first best action they see. This was also suggested in M.Lauer and M.Riedmiller (2000).

The differences between the techniques will become more clear in larger joint action spaces, however we can already see that using both exclusions and restarts is beneficial for finding more different Nash equilibria of the game.

⁵For now the implementation works with binary output learning automata, which is called the P-model. Generalization to continuous-values outputs is possible.

3.2 Results in larger joint action spaces

Most work on learning in games, is limited on 2-player games. We also want to evaluate the exclusion techniques on more agent games, which may have more than 2 or 3 actions. Although convergence is proved, the practical implementation may need some attention. Therefore we developed a generator for matrix games, in which the pay-offs for the different joint actions are uncorrelated random numbers between 0 and 1. The following Figures give results for the 3 techniques averaged over 100 iterations of play. For every iteration a new game of the same size was generated so that results are not dependent on the degree of difficulty of the first game generated.

In figure 6 average results are given for the many-exclusion-restart technique. Using both exclusions as multi-starts, assures that a good proportion of the Nash equilibria are visited. Almost 100% in small joint action spaces an still about 94% in the larger ones. It may seem strange that although the optimal Nash equilibria is visited, the players sometimes do not recognize it. This is a consequence of the fact that rewards are stochastic and when payoffs are very close to each other⁶, the sampling technique used here by the agents is not able to separate them. So actually when convergence is not reached, agents still find extremely good solutions which are almost inseparable from the optimal one. We tested this by redefining convergence in the last experiment of Figure 6. We agreed the agents had converged when the difference in payoff of the joint action they recognized as the best and the true optimal one was no more than 0.0001. This increases performance considerable as can be seen in Figure 6. In general by letting the agents do more sampling during period of play by enlarging the period length will reduce this effect. Of course when the joint action space becomes larger, the percentage of Nash equilibria visited decreases and the optimal joint action may not be visited yet. This may be solved by letting the agents play more periods.

We reran the same experiments with the same settings for the technique where only restarts and no exclusions are done after every period of play, to see if exclusions are really necessary. These results can be found in Figure 7. Both percentages of convergence and percentages of Nash equilibria visited are worse than those of Figure 6, which confirms the idea that exclusions are useful for guiding the coordinated search in the joint action space.

We also compared the many-exclusion- multi-start results with those of the one-exclusion-communication technique. The results of the latter can be found in Figure 8. It can be seen that convergence results are worse, mainly because less Nash equilibria are found.

⁶Very close means here, they only differ in the third or fourth decimal.

game	method	% of convergence	% of Nash equilibria found
climbing game	excl + restarts	100	98
climbing game	excl + commun.	100	93.5
climbing game	only restarts	97	90
penalty game($k = 0$)	excl + restarts	100	77.23
penalty game($k = 0$)	excl + commun.	100	67
penalty game($k = 0$)	only restarts	100	67
penalty game($k = -100$)	excl + restarts	100	98.35
penalty game($k = -100$)	excl + commun.	100	67
penalty game($k = -100$)	only restarts	100	79

Figure 5: Average results for the penalty and climbing game over 100 iterations.

#agents	#actions	#joint actions	% of convergence	% of nash equilibria found
2	3	9	97	99.67
2	5	25	95	100
3	3	27	96	98.38
3	5	125	87	93.67
5	3	243	78	94.51
5	5	3125	36	94.39
5	5	3125	68	94.39

Figure 6: Average results for the exclusion-restart technique over 100 iterations in different sized joint action spaces. Convergence is redefined in the last run. It suffices that the difference between the payoff of the optimal joint action and that of the one converged to is not greater than 0.0001.

4 Related Work

The issue of learning in games as a start for learning in MAS in general, has recently attained a lot of interest.

In common-interest or cooperative games, which was the subject of this paper, the idea of careful exploration is also used by others. In Claus and Boutilier (1998) independent Q-learners try to compute an equilibrium point by gradually reducing the exploration frequency. However convergence is not necessarily to the optimal equilibrium point. In S.Kapetanakis and Kudenko (2002), the Q-value of an action in the Boltzmann exploration strategy, is changed by an heuristic value, taking into account how frequently an action produces its maximum corresponding reward. This heuristic, which is called FMQ performs well in deterministic and partial stochastic games, however in fully stochastic games the heuristic may perform poorly. In M.Lauer and M.Riedmiller (2000) a convergence proof for independent optimistic Q-learners in coordination games is given, however the algorithm is not suited for stochastic games. In this paper, we viewed agents excluding actions from their action space as a discrete form of exploration. Our technique was able to solve challenging testbed coordination problems, even for the stochastic versions of it. Even more, convergence to the Pareto optimal joint action is proved. As is usual not done, we also experimented our technique in larger joint action spaces. Practical issues such as sampling effects may be-

come a problem there, so we need to further experiment on this.

Moreover the general idea of exploring by excluding is also promising in the sense that it is not only limited for use in common interest games. In Nowé et al. (2001); Verbeeck et al. (2002) independent learners are proved to be able to converge to equilibria in conflicting interest games, thanks to the idea of careful exploration with exclusions. Only now and then limited information is communicated. In the future we will further extend the idea of exploring by excluding to general-sum games.

In Hu and Wellman (1998), the work of Littman (1994) was extended for general sum games. However, both approaches are designed for learning in the joint action space directly and are therefore not suited for situations where the communication cost is not neglectable.

References

- C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 478 – 485, Stockholm, Sweden, 1999.
- C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In

#agents	#actions	# joint actions	% of convergence	% of nash equilibria found
2	3	9	93	93
2	5	25	91	97
3	3	27	95	94.59
3	5	125	73	88
5	3	243	75	88.53
5	5	3125	23	94.8

Figure 7: Average results for the restart technique without exclusions over 100 iterations in different sized joint action spaces.

#agents	#actions	#joint actions	% of convergence	% of nash equilibria found
2	3	9	83	85
2	5	25	89	98
3	3	27	86	94.33
3	5	125	78	88
5	3	243	73	86.7
5	5	3125	33	98

Figure 8: Average results for the one exclusion technique with communication over 100 iterations in different sized joint action spaces.

Proceedings of the fifteenth National Conference on Artificial Intelligence, pages 746 – 752, 1998.

J. Hu and M.P. Wellman. Multi agent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, 1998.

C. Jafari, A. Greenwald, D. Gondek, and G. Ercal. On no-regret learning, fictitious play, and nash equilibrium. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 223 – 226, 2001.

K. Verbeeck, A. Nowé, T. Lenaerts, and J. Parent. Learning to reach the pareto optimal nash equilibrium as a team. In *Proceedings of the 15th Australian Joint Conference on Artificial Intelligence.*, pages 407 – 418. Springer-Verlag LNAI2557, 2002.

M.L. Littmann. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 157 – 163, 1994.

M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the seventeenth International Conference on Machine Learning*, 2000.

K. Narendra and M. Thathachar. *Learning Automata: An Introduction*. Prentice-Hall International, Inc, 1989.

A. Nowé, J. Parent, and K. Verbeeck. Social agents playing a periodical policy. In *Proceedings of the 12th European Conference on Machine Learning*, pages 382 –393, Freiburg, Germany, 2001. Springer-Verlag LNAI2168.

S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *AAAI 2002*, 2002.

K. Verbeeck, A. Nowé, and J. Parent. Homo equalis reinforcement learning agents for load balancing. In *Proceedings of the 1st Workshop on Radical Agent Concepts*, page to appear. Springer-Verlag to appear, 2002.

C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279 – 292, 1992.

Emergence of organisations, emergence of functions

Davy Capera, Jean-Pierre Georgé, Marie-Pierre Gleizes, Pierre Glize

IRIT - Université Paul Sabatier,

118, Route de Narbonne, 31062 TOULOUSE Cedex, France

0033 - 5.61.55.82.95

{capera, george, gleizes, glize}@irit.fr

Abstract

In this paper, we present an approach for the design of complex adaptive systems, based on adaptive multi-agent systems and emergence. For this, we first introduce the Amas technology (Adaptive Multi-Agent Systems). This technology gives local agent design criteria so as to enable the emergence of an organization within the system and thus, of the global function of the system. Then, we describe an application in e-education by explaining its technical working and some experiments, which are thereafter discussed. Finally, we explain the phenomenon of emergence *via* some definitions extracted from literature and our own computer science oriented definition, and we discuss its use in our application. The originality of our approach lies in the very generic manner our re-organization rules work and that they are completely independent from the function the system has to compute.

1. Introduction

Software is produced for applications which are getting more and more complex, distributed, with a strong internal dynamic (example: a data mining system on the Internet) or dynamic in its interactions with the environment in which it evolves (example: a network management system). The classic computer techniques are not suitable because of the ever-growing design task for the developer. That is the reason for the existence of so many software development techniques, which nevertheless only manage to slow down - without stopping - the increase of human energy needed for this development.

Making machines more autonomous is a way to simplify the task of the designer. Formal theories can help us to represent and reason about time, space and dynamics of an evolving world. But we can also consider that such specifications are useless or even impossible in some situations:

- The system's environment is dynamic, making it ineffective to enumerate exhaustively all the situations the system may encounter (Hewitt, 1991).
- The system is open and therefore dynamic because it is constituted of a shifting number of components (Costa, 1996).
- The task the system has to achieve is so complex that we cannot guarantee a perfect design.
- The way by which the system may achieve the task it has been assigned is difficult or even impossible to apprehend globally by the designer.

The ability for artificial systems to confront really unexpected situations (like the ones resulting from a lack of spatial or temporal knowledge) implies a research mainline that leads to a different system design method than the traditional global top-down approach based on the modelling of the world (Weiß, 1996).

This paper is about the use of emergence of *organisation* as a design basis. An organisation between the agents enables them to produce a collective function, that is why, by way of the emergence of organisations, we think that the important point is the global function that emerges. The originality of the Amas technology (Adaptive Multi-Agent Systems), described in this article, is that it gives the basis for the design of the agents by

giving criteria that are local to the agents and that guide their behaviour so as to make their organisation emerge. Thus, the global system adapts without any adaptation of the agents' skills themselves.

2. The adaptive multi-agent system (Amas) technology

By specifying *a priori* a model for a system that will have to deal with unexpected events, you constrain (maybe inopportunately) the space of possibilities. Since Bertalanffy, many authors have studied "systems of different order that can be apprehended only by studying their parts taken separately" (Bertalanffy, 1968).

Each part P_i of a system S achieves a partial function f_{p_i} of the global function f_s . f_s is the result of the combination of the partial functions f_{p_i} , noted by the composition operator " $\circ$ ". The combination being determined by the current organisation of the parts, we can deduce $f_s = f_{p_1} \circ f_{p_2} \circ \dots \circ f_{p_n}$. As generally $f_{p_1} \circ f_{p_2} \neq f_{p_2} \circ f_{p_1}$, by transforming the organisation, you change the combination of the partial functions and therefore you modify the global function f_s . So this is a way to adapt the system to the environment.

For a multi-agent system, implementing this adaptation implies that the designer only has to take care of the agent by giving it the means to decide autonomously to change its links with the other agents, without no global control of the system (Horling, 2001).

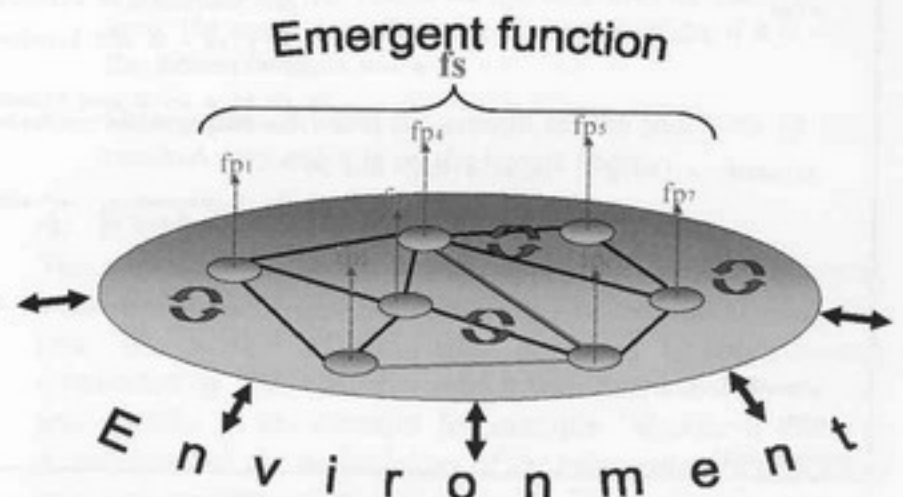


Fig. 1 : Interacting agents constitute the system.

In itself, the emergent organisation is an observable organisation which has not been given first by the designer of the system. But what we are most interested in is the emergence of the system's function that is produced by the organisation between the agents at a given time (cf. figure 1). Each agent computes a partial function f_{pi} . The combination of all the partial functions produces the global emergent function f_s . Depending on the interactions between themselves and with the environment, the agents change their interactions i.e. their links. This is what we call self-organisation.

To achieve this, we provide a local criterion for the agents. This criterion allows an agent to detect and to remove the non cooperative situations from its own point of view, in using its own knowledge and beliefs (belief is the representation of the others). This criterion will guide the behaviour of the agents and thus their interactions, so as to make the global function emerge. According to this theory (Camps, 1998):

- the system is evolving in an environment with which it interacts,

- the system is constituted by parts : the agents,
- each agent has its own function to compute,
- the global function of the system is not implemented in the agents and there is no global control,
- the conception is a bottom-up conception : you first define the agents.

The interactions between the agents will enable them to obtain the appropriate behaviour from the global system. We start from the principle that, to have a good behaviour, the agents that constitute a system have to be "at the right place, at the right time" in the organisation. To achieve this, each agent is programmed to be in a cooperative situation with the other agents of the system (Gleizes, 1999). Only in this case, an agent always receives relevant information for it to compute its function, and it always transmits relevant information to others. The designer provides the agents with local criteria to discern between cooperative and non-cooperative situations. The detection and then elimination of non-cooperative situations

```

c ⊆ P                                /* c is a subset of percepts */
a ∈ A                                /* a is an action */
SR ⊆ SkillRules                       /* SR is the subset of applicable behaviors from the skill rules */
BR ⊆ BeliefRules                      /* BR is the subset of applicable behaviors from the belief rules on other agents */
NCSR ⊆ NonCooperativeSituationRules /* NCSR is the set of applicable behaviors in the current non cooperative situation */
R = SR ∪ BR ∪ NCSR ⊆ Behavior Rules /* R is the total set of applicable behaviors */
(c,a) / c ∈ P and a ∈ A              /* (c,a) is a behavior */
(c1,a1) ≺ (c2,a2)                    /* express a priority relation of the behavior (c1,a1) on (c2,a2) */

/* The action function allows the agent to decide what action is relevant at the present time according to the perceptions p */
Function action (p:P) : A
var fire : f(SR)                      /* fire is the set of possible behaviors obtained from the f function */
var fireCooperativeAction : f'(BR)    /* fireCooperativeAction is the set of possible cooperative behaviors obtained from the f' function */
var fireNCS : f*(NCSR)                /* fireNCS is the set of behaviors deleting the current non cooperative situations obtained from the f* function */
var a : A

begin
  fire := {(c,a) / (c,a) ∈ SR, p ∈ c}
  fireCooperativeAction := {(c',a) / (c,a) ∈ fire and (c',a) ∈ BR and c' = c ∪ c* et p ∈ c ∪ c* }
  if fireCooperativeAction ≠ ∅ then /* If fireCooperativeAction is not empty, then the agent is in cooperative situation */
    for each (c,a) ∈ fireCooperativeAction do
      if (∃(c', a') ∈ fireCooperativeAction / (c', a') ≺ (c,a)) then return a
    endif
  endfor
  else /* Non cooperative situations are detected */
    /* If fire = ∅ and fireCooperativeAction = ∅, then the agent cannot use its percepts it is a non cooperative */
    /* If fire ≠ ∅ and fireCooperativeAction = ∅, then the agent cannot do 'a' because it beliefs 'a' non cooperative for other agents */
    fireNCS := {(c,a) / (c,a) ∈ NCSR and p ∈ c}
    /* By consequence, the agent must fire a behavior belonging to NCSR in order to suppress the non cooperative state */
    for each (c,a) ∈ fireNCS do
      if (∃(c', a') ∈ fireNCS / (c', a') ≺ (c,a)) then return a
    endif
  endfor
endif
return null

```

Fig. 2. The algorithm describing the Cooperative Behaviour of an agent.

between agents constitute the engine of self-organization.

You can see in figure 2 an algorithm which details the basic behaviour of a cooperative agent according to the Amas technology.

3. Application in e-education

The goal of the Forsic software is to enhance the skills and know-how of trainers in the information searching domain (Link-Pezet, 2000). It concerns the search for relevant information sources using a textual description of these sources. The trainers must modify regularly their teaching's content and their teaching resources (Gleizes, 2000). To adapt in real time to this dynamic of the domain (technical as well as terminological), we designed the Forsic kernel to manage dynamically the description of the trainers' skills, of the pedagogical resources and of the students' training needs during functioning and without asking too much involvement from the human user.

This kernel should be empty at the beginning and should constitute itself while processing the different searches.

The developed system must enable the set of the words used in the textual descriptions to organise themselves in the most optimal manner, so as to provide relevant answers to the requests concerning this knowledge. So, our achievement is about the modelling of a semantic using an emergent organisation, called semantic organisation. For this, we designed an adaptive multi-agent system. In input, the system receives knowledge it will learn, or requests to which it will answer. In output, the system orders and presents the entities that are the most relevant for the request.

To summarise, the objective of this multi-agent system is to store and organise the knowledge that we give to it and to answer to requests concerning this knowledge.

3.1 The semantic organisation in Forsic

In order to conceive the multi-agent system that produced the desired results we used the Amas technology. There are 2 types of agents in the system (Carré, 1999):

- Agents of type Trainers represent the users who own information. So, their number is variable and their role is to aggregate the descriptions of the knowledge of these users, which they transmit to the agent Interface when they are active (for example: trainers).
- The agents of type Term wrap all the terms of the descriptions learned by the system.

According to the basic parts of an agent, the beliefs are the pondered links that link the agent to other terms by a father-son relationship. There is no value zero on the links because the

value zero means that an agent has no beliefs on the other. When it is created, the agent is at least in relation with the agents appearing in the same description as itself. If the agent already exists in the system, it keeps the links it already had and adds the links to the agents present in the new description. The interactions between the agents consist of message forwarding, following existing links.

Each agent of the type Trainer or Term possesses an activation function with two thresholds: a wake-up threshold and an activating threshold. So the agent can have three states:

- The passive state occurs when neither of its sons is active.
- The awaked state corresponds to an undetermined state: he's got active sons but the sum of the weights of their beliefs is insufficient.
- The active state means that the sum of the weights of the beliefs associated to its active sons is greater than the activating threshold.

3.2 The self-organisation rules

The AMAS technology imposes to study exhaustively and locally to each agent, every case of non-cooperative situation which could possibly arise. But this is done in a generic manner as we will see it here.

A direct activation of an agent Term occurs when the term that this agent represents is present in the textual description sent to the organisation. In order to propagate the activation of an agent, the agents that receive the corresponding message have to be able to become active too.

After the activation stage, only the awaked (and not active) agents correspond to the case of non-cooperation, because an awaked agent received some signals but it could not process its function. These agents become the generators of non-cooperative situations toward their sons so as to warn them that they may have made an error concerning their state. If an agent receives a majority of non-cooperative messages, it will propagate them to its sons. After the detection of the non-cooperative situation, each agent tries autonomously to eliminate it. This is done by adjusting the weights of the fathers or sons so as to make the agent become, either completely passive or completely active, and no more in contradiction with its acquaintances. This adjusting can lead to some reorganisation:

- Rising when the weight on the link with its father is greater than the absolute value of the activation threshold: the agent goes up in the local hierarchical organisation.
- Descent when the weight on the link with its father reached zero: the agent goes down in the organisation, if it is not on the lowest level.
- Disappearance when the weight on the link with its father reached zero and it is on the lowest level.

4. Experimental results

The application presented in this paper is constituted by thirteen trainers. These people describe their competencies using free text. The amount of knowledge described is approximately constituted by 300 sentences written with a terminology more or less specific to the domain; for example "Realise a teaching program about the methodology of the information retrieval and use, for students of the first cycle. This program must be relevant with the student's subject". When the terms are very

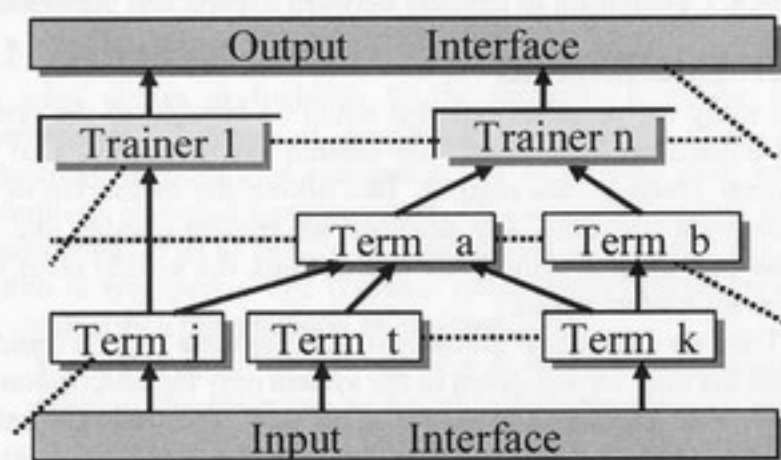


Fig. 3. The semantic organisation

specific, the corresponding competency elements are very specific too and concern a restricted number of trainers that are consequently indexed by these terms. On the contrary, a very generic and little decisive term leads to a very important number of less generic sub-sectors.

4.1 Self-organization

The result of the self-organisation process is the easiest to observe in the system. All new term is considered as specific because it was never previously encountered. So, it is directly linked to the trainer whose competence is described. But, if this term is common, its tendency will be to go down the organisation. Inversely, a term very often employed at a given time could later become specific and go up in the organisation.

Organization Levels Terms	Level 1		Level 2		Level 3		Level 4	
	C1	C2	C1	C2	C1	C2	C1	C2
Documentary	150	60	0	180	0	30	0	0
Internet	50	50	0	160	0	0	0	0
IST	50	50	0	70	0	30	0	0
How	40	0	0	20	0	90	0	80
Intranet	50	30	0	0	0	0	0	0
Science	130	140	0	90	0	70	0	20
Methodology	140	50	0	650	0	30	0	0
No	40	0	0	0	0	35	0	5

Specificity

↑

Intranet
Internet
IST
Science
Methodology
Documentary
How
No

Genericity

Table 1. The semantic organisation after the description of four trainers (columns C1), and then thirteen trainers (columns C2).

In the semantic organisation, we have selected a representative specimen of eight terms employed in the textual descriptions of the trainer’s competencies. For each of them, we analysed their position in the organisation that emerges from the learning. These results are written in the table below: each line corresponds to a term and the values of each column correspond to the cumulated influence at a given level of the organisation. Each agent which represents a term (the French term IST means technologic and scientific information) has parent links with others agents (normally these agents are more specific). The strength of these links is the influence (an integer). A cumulated influence at level 1 shows that the term is specific because it is directly associated to one or more trainers.

Table 1 gives the place of 8 agents Term in the semantic organisation after the learning of the descriptions of four trainers (among 13). These terms are all directly linked to the trainers: it is evident that the system has at this moment a non-hierarchical organisation (all weight is in column C1 of Level 1 only). The amount of descriptions is not sufficient to differentiate the terms that are more relevant than the others.

After that, we integrate the descriptions of the nine others trainers. This corresponds to the creation of 531 agents Term. The position of the 8 agents Term we are looking at in the system has considerably evolved and a differentiation of their role has been realised (as seen by comparing C1 and C2). We could observe that the number of levels is comparatively reduced (four at the maximum), but this offers a very important combinatorial space. This is due to the fact that the average connectivity of the agents is around eight.

With the results of table 1, the terms could be ranged on a genericity-specificity scale in comparing the influence of terms two by two. We obtain the gradation (see on the figure 3) which

is relatively relevant for the studied domain. The words “No” and “How” belong to the common language, they went down in the levels of the organisation. On the opposite, the words “Internet” and “Intranet” are relevant for the information retrieval, but only some trainers have the right competency and are indexed by them.

4.2 Functional adequacy

To know if the system is functionally adequate in its environment, we need to decide if the self-organisation process between its agents leads itself to interpret correctly the requests and to give right answers. Only an external observer of the system could judge the relevance of the answers.

To study this evaluation, we used a set of requests given both to the system and to three experts of the domain. The graphic in figure 4 compares the answers in terms of distance between the answers of the system and the three experts’ answers. For example, the first request is the sentence: “I’m looking for a formation in documentary methodology for the first cycle in Economy or Law” (request number one).

A null distance means that the experts’ answers and the answers of the system are the same. A distance of five means that the system has given answers which are not agreed by the expert. It is the case in the request number nine which consists of the simple sentence: “documentary searching”. This is a request considered as too vague by the system and so it does not manage to activate the particular representations it has of the trainers. The human experts are not blocked by this not very significant request. Nevertheless, the system gives an average answer rate that is not so far from the experts’ one (about 7 among 10), without taking into account that there are also some disagreements between the experts. We could conclude that the system is successful in allowing the emergence of a function which seems satisfying. And this without the coding in the system of a linguistic and contextual knowledge of the domain.

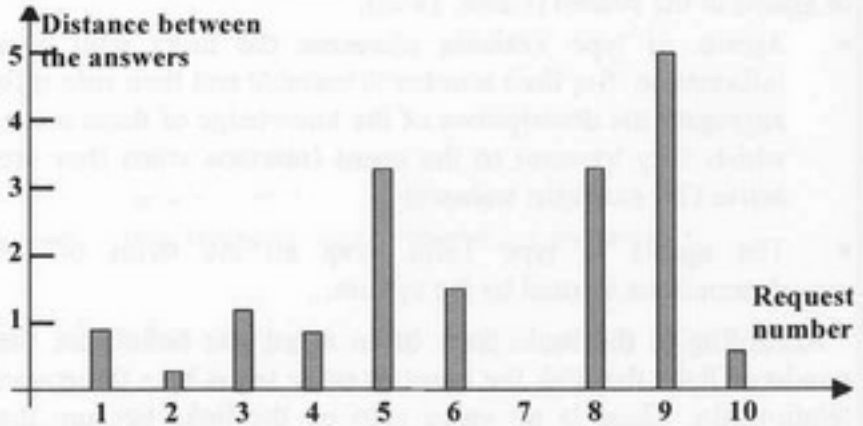


Fig.4. Comparison of answers between experts and the system.

4.3 Equilibrium

To study the perturbations that could be brought to the system by enhancing its content, we centred on the analysis of the answer given to one request. This allows the measuring of the functional adequacy. For example, for request number one and after providing it with all the descriptions, the system gives four relevant trainer names.

Then we interrogate the system with the same request (number one) but after having given to the system only the descriptions of these four trainers. The system gives only three relevant trainer names. This result clearly indicates that the following learning

allow it to modify the representation of the domain it has in order to allow this fourth trainer's name to emerge.

As it arranges the answers in order of relevance, it shows that this fourth name is the worst answer. We note also that this trainer is given only by one of the three experts who participate to the system evaluation. We can say that, even if the descriptions of the nine trainers which are in the same competency domain have modified the behaviour, they don't affect the relevance of the previous answers.

A second experiment on the equilibrium was done by integrating in the semantic organisation an important number of new domains. This corresponds to 100 content providers and to 3000 new terms. It is the description of a catalogue for the electronic commerce with one thousand of various products such as: computers, paintings, medicine or mechanical pieces. The previous domain, the documentary searching represents now only a small part in the semantic organisation. However, there is a clear partition because the competencies of the new domains are so far from the old one (the term employed are new).

After we have given all the descriptions, i.e. after the learning, we see that there is no effect on the existing system, because the previous requests give the same answer as before the learning of this new information. The equilibrium of the sub-part related to the documentary domain can be understood because the non-cooperative situations generated by the new learning interfere only a little with the initial system.

4.4 Analysis

The results obtained on several cases show that the system is able to provide average relevant answers. Nevertheless, in some cases the system is not able to achieve associations that people do automatically. In the example: "*documentary searching*" the system is not able to answer because there is no semantic of the word, which has to be made, in general, by a learning on a long period. At the initial step the system is empty, and the self-organisation process makes these associations. However, the learning is very fast, and each description enriched the representation previously acquired.

The evident interests of this approach are multiple:

- It is an open system. It permanently accepts every new terminology and new Trainers.
- Consequently, the environment can also be dynamic.
- The self-organisation process is totally independent of the domain.

5. Discussion on emergence in artificial systems

5.1 Definition

So what allows us to judge of the emergent character of a phenomenon? Instead of pretending to give an exact and exhaustive definition of emergence, we will rather give, using definitions extracted from literature, the particularities that seem fundamental to us to be able to comprehend in a precise way the notion of emergence. The common inter-related proprieties that let us identify a phenomenon as emergent are:

- The observation of an ostensive phenomenon (it imposes itself to the observer) at the global or macroscopic level. Emergence is defined in terms of the irreducibility of properties associated with a higher level theory to

properties associated with components in a lower level theory" (Kim, 1999).

- The radical novelty of the phenomena (it is not visible at the microscopic level and it is impossible to predict it). "On one hand, emergence presupposes appearance of novelty – property, structure, form or function –, and on the other hand, it implies that it is impossible to describe, explain or predict these new phenomena in physical terms from the basic conditions defined on the lower levels" (Van de Vijver, 1997).
- The coherence and correlation of the phenomena (it has an own identity but it is strongly linked to the parts that produce it). "Emergence refers to the arising of novel and coherent structures, patterns, and properties during the process of self-organisation in complex systems" (Goldstein, 1999).

During the last fifteen years, a whole research field slowly appeared around the concept of emergence so as to exploit its so particular characteristics in computational systems (Holland, 1997). "Emergent Computation" (Forrest, 1990) (Gilbert, 1995) (Santa Fee institute, Internet), is the term used in general to refer to the research main line on which our work stands.

Our works in this domain during the last decade lead us to give a "technical" definition of emergence, and therefore with a strong computer science coloration. It is based on three points:

1. **The subject.** The goal of a computational system is to realise an adequate function, judged by a relevant user. It is this function, which may evolve during time, that has to emerge.
2. **The condition.** This function is emergent if the coding of the system does not depend in any way of the knowledge of this function. This coding has to contain the mechanisms allowing the adaptation of the system during its coupling with the environment, so as to tend anytime towards the adequate function.
3. **The method.** To change the function, in the Amas technology, the system only has to change the organisation of its components. The mechanisms which allow the changes are specified by self-organisation rules providing autonomous guidance of the components behaviour without any knowledge on the collective function.

So when we design an agent for a MAS, the code of the agent doesn't contain any knowledge of the collective function we want the MAS to compute.

5.2 Emergence of the semantic organisation

The semantic organisation, meaning the linking of the terms known by the system, is emergent. The emergent phenomenon consists of the fact that the outputs of the system are really relevant answers to requests. The system gives a sense to these outputs by self-organising.

In the studied application, the observer is the user of the system and he has access only to the results i.e. to the answers of the system. He is placed at the global or macro level and is able only to judge the relevance of the obtained answers, as it is shown by the results in section 4. Consequently, we observe an ostensible phenomenon at the macro level.

The answers which are the function of the system, are emergent because they are not coded in the agents term. In fact, an agent term contains only one word and it is not an answer in itself. This confers to the phenomenon the nature of radical novelty. Moreover, an agent executes its function at its local level without knowing the consequences at the global level, and this does not allow the prediction of the global behaviour of the system.

Nevertheless, the term agent's objective is to aggregate itself to other agents. This is the way to create an organisation or a coherent structure with a kind of equilibrium (showed in the section 4). These measures show the self-keeping of the function when the system modifies itself after new interactions with the user. Not only does the adding of contradictory descriptions with the existing ones imply the modification of the answers given at the macro level, but also, a bad answer at the macro level forces the agents of the system to reorganise themselves.

The properties expressed above such as the ostensible character, the novelty, the equilibrium, the dynamic, the interdependency between the macro level and the micro level confer to the semantic organisation and to its function (the relevant answers) the emergent characterisation.

The emergence described above shows well that it is the organisation that provides the function of the system. It is necessary, by consequence, for designing a multi-agent system, to construct the most relevant organisation between the agents. For multi-agent systems plunged in evolving environments, the organisation must evolve to allow the system to adapt itself. Frequently in multi-agent applications, the organisation cannot be completely specified at the design stage because of the complexity, the distribution and the dynamic of the domain. The Amas theory allows the organisation to emerge.

In comparison to most works on the design of multi-agent systems, for which the organisation is described at the beginning of the design (Ferber, 2000), the originality of the Amas theory consists of the fact that we do not have to give an initial organisation. In fact, the organisation self-constructs and self-maintains itself during the functioning of the system and can only be observed.

The Amas theory allows the manipulation of the organisation as an emergent entity, because it creates it, and also like a dynamic entity because it causes its evolution depending on the interactions between the system and its environment.

6. Conclusion

In the field of the manipulation of organization, we chose the path of emergence. Therefore, in this paper, we presented, on the experimental level, the results and the functioning of an adaptive multi-agent system, and on the theoretical level, the concept of emergence and the Amas technology referring to it. Throughout the paper, we explained why we think that this original approach for the manipulation of dynamic multi-agent organisations is especially relevant for many applications (see <http://www.irit.fr/SMAC/>).

Between others, this approach allows us to make abstraction of constraints like theoretical models, complex protocols, planning tools, and nevertheless be appropriate for the design of open and heterogeneous systems, needing complex interactions, coordination and learning.

Our goal in the near future is to spread this way of designing MAS. For this, we are actually working on a design methodology called Adelfe (laboratory for the design of

emergent functionality software) for the engineering of open and evolutionary systems. Adelfe is a national project that aims to lead a programmer step-by-step through the designing of an adaptive system using our MAS technology.

References

- Bertalanffy von, L.: General System Theory - George Braziller Editions, New York (1968).
- Camps Valérie, Gleizes Marie-Pierre, Glize Pierre: Une théorie des phénomènes globaux fondée sur des interactions locales - Actes des Sixième journées francophones IAD&SMA Pont-à-Mousson - Editions Hermès, Novembre (1998).
- Carré Jérôme, Machonin André, Glize Pierre: Un système multi-agent auto-organisateur pour l'apprentissage d'un profil utilisateur - Actes des septième journées francophones IAD&SMA, Editions Hermès, (1999).
- Costa, A. C. R., Demazeau Yves. 1996; Toward a Formal Model of Multi-Agent Systems with Dynamic Organizations. In Proceedings of the Second International Conference on Multi-Agent Systems - AAAI Press/The MIT Press (1996).
- Ferber, J., Gutknecht, O., Jonker, C., Müller, J., Treur, J.: Organization models and behavioural requirements specification for multi-agent systems. ICMAS2000 Proceedings, Fourth International Conference on Multi-agent Systems. IEEE Computer Society (2000).
- Forrest, Stephanie: Emergent computation: self-organizing, collective, and cooperative phenomena in natural and artificial computing networks. Proceedings of the ninth annual CLNS conference (1990).
- Gilbert, N., Conte, R.: Artificial Societies, The computer simulation of social life. UCL Press (1995).
- Gleizes Marie-Pierre, Link-Pezet Jo, Glize Pierre: An Adaptive Multi-Agent Tool for Electronic Commerce - Second International Workshop on Knowledge Media Networking - Gettysburg, USA - (2000)
- Gleizes Marie-Pierre, Camps Valérie, Glize Pierre: A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. Fourth european congress on systemic (1999).
- Goldstein, Jeffrey: Emergence as a Construct: History and issues. Emergence Volume 1, Issue 1, pp 49-71 (1999)
- Hewitt, Carl: Open Information Systems Semantics for Distributed Artificial Intelligence. In Artificial Intelligence, Special Volume Foundations of Artificial Intelligence. Elsevier Science Publishers B. V. Vol. 47 (1991).
- Holland, John H.: Emergence - from Order to Chaos. Addison-Wesley, (nov. 1997)
- Horling, B., Benyo, B., Lesser, V.: Using self-diagnosis to adapt organizational structures. AGENTS'01 Proceedings. ACM (2001).
- Kim, Jaegwon: Making Sense of Emergence - Philosophical Studies, N°95 (1999)
- Link-Pezet Jo, Gleizes Marie-Pierre, Glize Pierre: FORSIC: A Self-Organizing Training System. IJIS/SAS Orlando, (2000)
- Santa Fee Institute : www.santafe.edu.
- Van de Vijver, G.: Emergence et explication. Intellectica : Emergence and explanation (1997)
- Weiß, G.: Distributed artificial intelligence meets machine learning. Learning in multi-agent environment. ECAI'96 Workshop LDAIS, ICMAS'96 Workshop LIOME. Springer (1996)

Real-time Simulation for Flood Forecast: an Adaptive Multi-Agent System STAFF

Jean-Pierre Georgé, Marie-Pierre Gleizes, Pierre Glize, Christine Régis

Institut de Recherche en Informatique de Toulouse
Université Paul Sabatier
118, route de Narbonne
31062 Toulouse Cedex, France
{george,gleizes,glize,regis}@irit.fr

Abstract

Flood forecast implies to face to a complex and dynamic environment. The STAFF real-time simulator uses an adaptive model for flood forecast, which is composed of two levels of self-organizing multi-agent systems. The goal of each upper level agent is to compute the water level variation during a unitary period (typically an hour) and uses for that a weighted sum of agents in the lower level. The model's adaptive nature is obtained by adjustment of these weights, decided from the co-operation between the agents. It makes the model generic and improves its performances. The first part of the paper introduces the complexity of the flood forecast simulation and shows the need for an adaptive system. We then present the benefits of the AMAS theory in such complex and ill-defined problems. In the third part of the paper, we describe the system architecture derived from the application of the AMAS theory. Then we analyze many use cases (noisy and missing data, totally upstream stations, real-time learning,...) in which usual hydrological models are inadequate. Finally we study the properties of this learning approach in regard with classical criteria (space search, convergence speed) and related works. STAFF is operational since 2001 in the Midi-Pyrenees region of France. It will be installed in all the country during this year.

1. Introduction

Periodically and particularly these last years in several french regions, exceptional precipitations have brought about ruining unforeseeable floods which caused real human and material disasters, sometimes locally, sometimes on most extensive superficies. Some, as in the Somme region, during most than a month, have generated the necessity to evacuate and put up precariously thousands of people. Those recent historical events show again with evidence the great challenge of flood forecast studies (Bras, 2001). The complexity of the flood forecast simulation is issued of the incapacity to exactly forecast the quantity and the location of rains. As long as this domain will remain approximated, and because this environment is almost unpredictable, the design of STAFF uses a totally different approach from the existing models. Principles of autonomy, cooperation and self-organization have been outlined to overcome several environment perturbations using the AMAS theory (Adaptive Multi-Agent System). We show how the theory has been applied setting out in detail the architecture and the working principle of an adaptive multi-agent system. The main characteristic of this model is to adapt itself automatically to its environment. This adaptation allows to use this model independently of its implantation location to have forecasts everywhere and also to take into account the evolutions of the environment and the specific character of each event in order to keep its forecast capacities at the average and long time. To be really adaptive, the system does not make use of any information peculiar to a basin (i.e. which do not come from sensors). There are two types of data coming from sensors:

- measures of level given as water level variation,
- measures of rain i.e. quantity of precipitation.

From these measures, STAFF which is installed on a station has to make a forecast of water level variation over a couple of hours

(for example a six hour forecast at Toulouse). An algorithm for cooperative self-organization is in fact automatically built when the system is designed according to the AMAS theory. In the flood forecast domain, although the general problem of a six hours forecast remains always the same, its representation is evolving over time (entry data evolution, sensors out of order,...) which implies a real time adjustment of the algorithm.

2. The AMAS theory

2.1 The question of functional adequacy

A system is functionally adequate when it has this "right" behavior -judged by an external observer who knows the environment. On the opposite, when the system is functionally inadequate, at least one unexpected response occurs for the environment viewpoint. If the environment cannot accept this behavior from the system, it will later have a behavior that will disturb the system. Thus, system and environment are all the time in an interacting process where the resulting activity of one influences the other.

These interactions can be classified into three categories: cooperative activity, antinomic activity, indifferent activity. From evidence, a functionally adequate system cannot have antinomic interactions with its environment. A system is self-adaptive when it tries all the time to be functionally adequate, even if the environment is dynamic. An adaptation process can only occur if the system, which has to adapt itself, is plunged into an environment. The system behavior modifies the state of the environment that in turn reacts in exerting a pressure on the system. Thus, the adaptation is processed by coupling. In our work, we consider the system as a multi-agent software and the environment as both a human and a geographical region perceived by the way of disseminated sensors.

2.2 Adaptive multi-agent systems

We define an adaptive multi-agent system as a multi-agent system which is able to change its behavior to adjust itself in its dynamical environment, either to realize the task it is intended to complete or to improve its functionality or performance. This system is characterized by the following points:

- The system is plunged in a dynamical environment.
- The system realizes a function.
- The system is composed of interacting autonomous agents.
- Each agent of the system realizes a partial function. The global function is the result of the composition of the partial functions of the agents and of their interactions.
- The organization of the system determines the result of the system. The composition of partial functions is determined by relations which link agents i.e. the organization.

Thus, without changing the agents, modifying the organization between them provides a new global function: it is a mean to self-adapt. The remaining question is: when and how does an agent decide to modify its relations with others agents?

2.3 The AMAS theory for cooperative self-organization

The AMAS theory (Gleizes, Georgé, Glize, 2000) (Gleizes, Camps, Glize, 1999) allows to guarantee the convergence to a desired global function. This is obtained by the following theorem: *For any functionally adequate system in a given environment there is a system having a co-operative internal medium which realizes an equivalent function.*

The consequence of this theorem is that the agents composing a system must try to achieve their individual goals and must be co-operative with each others to allow the global system to be functionally adequate. That is to say the system cooperates better with its environment and improves its functional adequacy without requiring the knowledge about the global function it must realize. Designing a cooperative multi-agent system with the AMAS theory consists in defining for each agent, all possible non cooperative situations from their local point of view and the actions associated to these states in order to eliminate them. When the system is running in a dynamic environment, an internal process is observed in which the relations between agents are modified. Thus, the reorganization of the partial functions achieved by each agent leads to a modification of the global function of the system, and non co-operative states due to unexpected events are progressively eliminated (see also <http://www.irit.fr/SMAC>).

In the agent's viewpoint, a non cooperative situation is like an exception in a classical program. The designer has not only to describe what an agent has to do to achieve its local goal but also which local situations this agent must avoid and if these situations are detected how to remove them. Three kinds of non cooperative situations can be detected by an agent:

- when a signal perceived from its environment is not understood and not read without ambiguity;
- when the information perceived does not induce the agent to an activity process;
- when concluding results lead to act in a useless way in the environment (such as conflict and concurrency).

3. The multi-agent systems of STAFF

STAFF is an adaptive model of flood forecast (Sontheimer, Cornuau, Vidal and Glize, 2001) because it is based on a set of multi-agent systems themselves using the AMAS theory, as showed in the figure 1. The aim of STAFF is to be able to compute the flood forecast at any point of the basin without any programmed information (physical and hydrological). It has only all the data gauges on the basin and the real current river level at the desired point in which STAFF must give a flood forecast value in real time.

3.1 The STAFF architecture

The upper level is a MAS having the goal of computing the water level variation during many hours. Each agent of this MAS is an hourly agent having the goal to give the level variation during an hour (for example between $t+3$ and $t+4$).

Each hourly agent is a MAS of the lower level. An agent of this MAS is associated to a sensor measure and his goal is to find the real influence of this entry during the time period of the hourly agent.

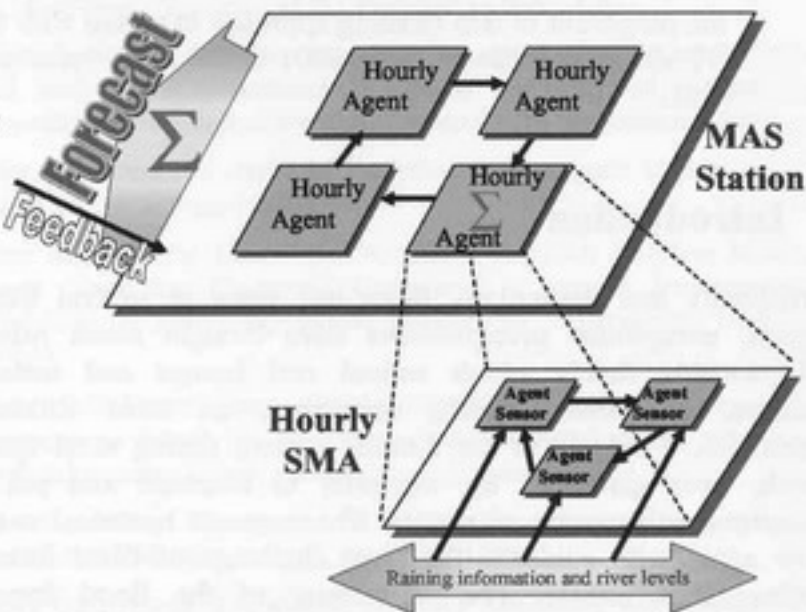


Figure 1: The STAFF architecture composed of two levels MASs

In practice, STAFF allows to approach in real time a forecast curve by a succession of straight line segments displayed by discontinuous lines, like a linear function. The experiment will allow to validate that the self-organizing cooperative algorithm which is used in our adaptive system leads effectively to the two following results:

- the selection of pertinent parameters to produce the nearest straight line to the curve at current time (it consists in searching for the linear function).
- to adjust these parameters in real-time to find a new straight line approaching the new curve form.

In order to indicate the self-organization steps, we will use the following notations:

- S_i is the value of a sensor agent associated to the sensor i (which could correspond indifferently to a rain gauge or a river gauge without any additional information on its location).
- ω_i is the weight applied to the entry value i . This is an integer varying between 0 and 2000. A value less than zero means that this sensor is currently irrelevant to explain the output (the hourly forecast).

- $\Delta\omega$ is the minimal value of the increment applied to the weight ω_i . This value is always predefined to 1.
- The hourly forecast F_k (from time $t+k-1$ to time $t+k$) is the weighted sum of all the entries greater than zero. $F_k = \sum (\omega_i * S_i)$, $\forall i/\omega_i > 0$
- Forecast is the result given by the MAS station. $\text{Forecast} = \sum F_k$
- Feedback is the real variation of the river level between time $t-1$ and t .

3.2 Self-organization in an hourly MAS

To approximate the measure curve by a linear function, the specific influence of each sensor information has to be determined. Depending on the flood dynamic character, this influence is totally unknown at the design phase of the system. This is the reason why we encapsulate each sensor measure by an agent which is in charge of determining this influence as a weight.

Each entry (several hundred at all) is a measure coming from sensors of the Garonne's hydrological basin. On a station, all the sensors intervene as a pondered sum of their weights in order to compute the water level variation during an unitary period of one hour. Thanks to this systematic adjustment, the modeling is adaptive and the relation non linear in spite of using a simple balanced sum.

The aim of a sensor agent is to determine in what part the measure it is associated to intervenes in a hourly forecast. The typical non cooperative situation for a sensor agent consists in bad evaluating its influence. This case appears every time the forecast agent, it is working for, has to readjust its own forecast. Non cooperative situations depend on the correlation and influence notions. Correlation indicates if a measure has to be used to compute the forecast and if so influence determines in what part.

There are three types of non cooperative situations to which a sensor agent may be faced:

1. The entry value of an agent is not correlated with the feedback (the station evolution). This is a non cooperative situation of uselessness, because the agent cannot at this moment explain the output. The weight value must be diminished: $\omega_i = \omega_i - \Delta\omega$.
2. The entry value of an agent is correlated with the feedback, but ω_i is currently negative (this entry was mainly in the past not correlated). This agent could be useful and the weight value must be increased: $\omega_i = \omega_i + \Delta\omega$.
3. The entry is correlated but the forecast given previously for the current time t (for which we know now the feedback value) is erroneous. The wrong influence of the agent must be modified in order to decrease the forecast error. Because at the same time many agents can act in the same way we are in the concurrency situation. $\omega_i = \omega_i + \text{Sign}(\text{Feedback} - \text{Forecast}) * \Delta\omega$.

3.3 Self-organization in the MAS station

In order to take into account the very last sensors information, the forecast delay for a forecast agent has to be the shortest as possible. The inferior limit for the forecast delay is equal to the range period of the sensors which is of one hour on the basin of the river. Thus we have built a multi-agent system made of several forecast agents, each having a forecast range of one hour. The number of the forecast agents in the multi-agent system of a station depends on the number of hours in the forecast delay. Each forecast agent computes its forecast for its own period of one hour. The sum of the hourly agents gives the multi-agent system forecast associated to the station.

The presence of several hourly agents in the same station multi-agent system may lead to conflicts situations between their respective results. STAFF is coded to take into consideration only the most critic of them leading to the following treatment :

1. The first forecast agent (named F_1) compares its previous forecast with the last station measure (Feedback) as indicated in figure 2. A difference is a conflict with the real value which implies and adjustment inside the corresponding hourly MAS (see 3.2).
2. The others forecast agents compare their previous forecast with the last one of the previous agent (for example F_{k+1} with F_k). The difference is also a conflict which implies an adjustment inside the corresponding hourly MAS. Nevertheless an agent forecast (here F_k) presents an error in comparison to the corresponding measure. Experiments have shown that the error induced by the adjustment delay is most important than the one induced by the adjustment compared to a slightly erroneous forecast.

4. Experimental results and analysis

Since 2001, STAFF is working under the software platform called Sophie, and used by the experts' DIREN (Direction Régionale de l'Environnement) which are responsible of flood management in Midi-Pyrénées region of France. Sophie is used by all the organisms responsible of flood announce in metropolitan France. This explains why the results presented here concern only sites of the Midi-Pyrénées region and are obtained from real graphical results presented by the Sophie platform.

4.1 Flood forecast for an intermediary station

The X-axis represents time over 10 days of measures (240 hours). The Y-axis is the evolution of a river level (in meters). The dark-blue curve is the real evolution of the river, whereas the red curve is the forecast of this river given by STAFF. The light blue and light green curves are the results of classical hydrological models (see in Fig. 2).

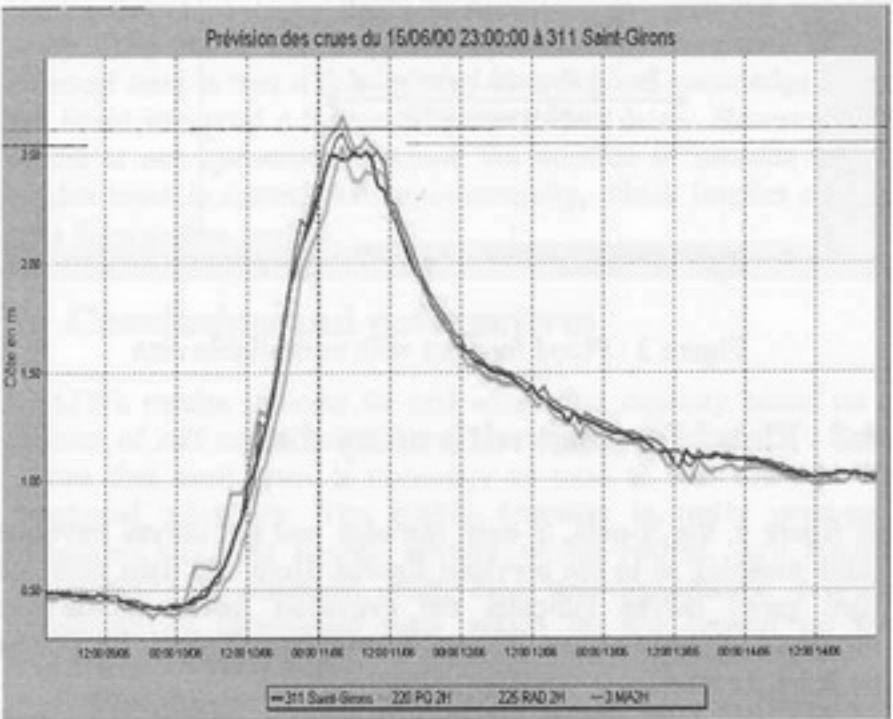


Figure 2 : Typical result for an intermediary station

From evidence we can say that the predictive STAFF model is very closed to the real evolution of the river. Generally speaking, on a

dozen locations where classical models exist in Midi-Pyrénées, the STAFF software gives at least better results than the previous models.

4.2 Flood forecast with unavailable data

The caption associated to the figure 3 is identical as the previous one, there is only one classical model (the light green curve). But this model is unavailable during approximately 36 hours because of the main data used for computing its mathematical function is missing due to the breakdown of associated sensors. This crash is a very awkward situation because the river level increases more one than meter and the expert can not have critical information to his disposal. During the same period, the Staff model is always available because it computes a linear function where unavailable data is forced to zero. From evidence this leads to an erroneous prediction, increased by the absence of the most significant entry data.

But, the STAFF adaptation process is always running and this allows to adjust in real-time the influence of inputs. Thus unavailable data are compensated by other entries which are not exactly so relevant, nevertheless able to predict correctly the starting flood. We must quote the few number of examples (less than thirty) allowing a relevant and important self-organizing process. When the missing inputs reappear later (the light green curve appears again), a second adaptation process allows to find the previous organization between inputs.

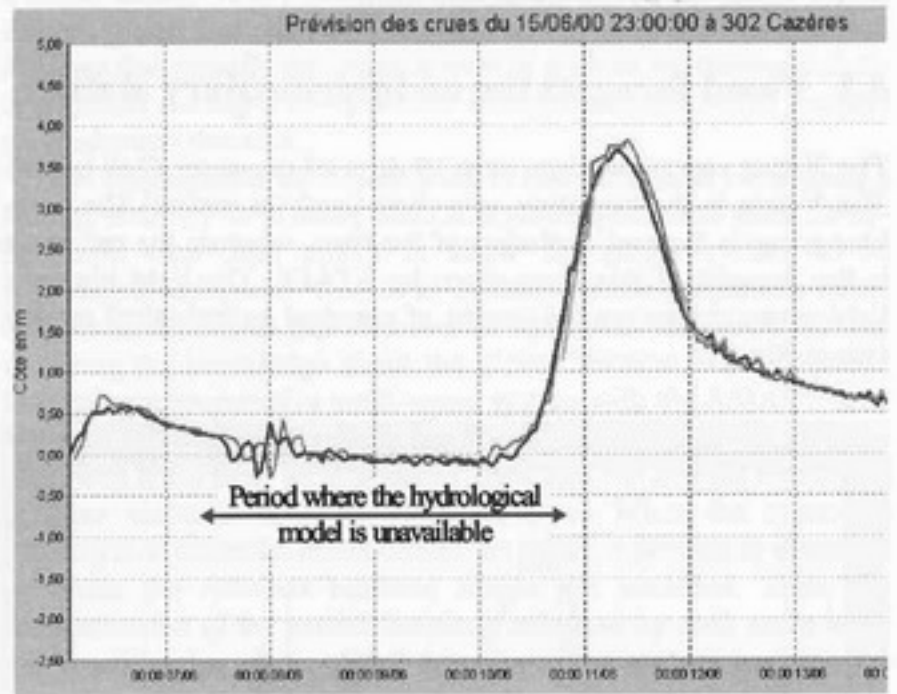


Figure 3 : Flood forecast with unavailable data

4.3 Flood forecast with noisy data

In figure 4, the X-axis, Y-axis, the blue and red curves have the same meaning as in the previous figures. Here, the light blue and light green curves indicates the evolution levels of the two important upstream rivers. We can observe two distinct phenomena on these stations :

The light blue station has three important periods of missing data (when the flood starts, at the middle and on the maximum). The light red station is very noisy due to non natural evolution. In fact, its location on the river is near a dam constructed to produce electricity.

The STAFF model gives not very precise results, reflecting the noisy and absent data observed on the upstream stations. Nevertheless STAFF gives some idea about the river fluctuation useful for the expert decision. During the past years some hydrological models were tested and given up on this station because they give no relevant result.

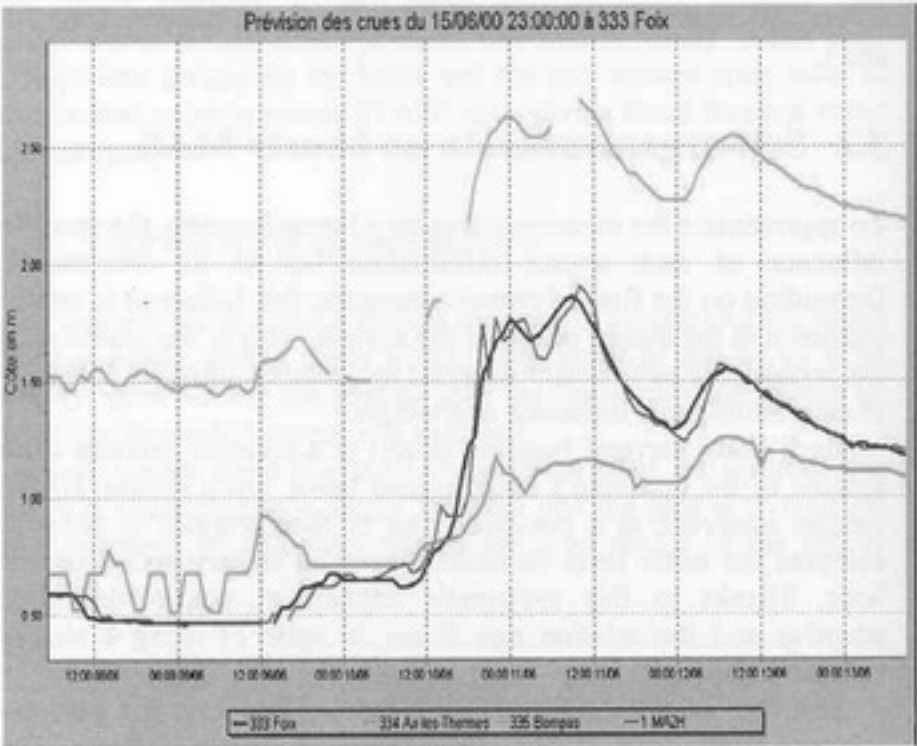


Figure 4 : Flood forecast with noisy data

4.4 Flood forecast for an upstream station

In a hydrological network there are many upstream stations, mainly located in the middle mountain Pyrénées for our application. In these twenty stations no hydrological models cannot be of use because the essential information about the river evolution on the upstream are missing by principle. We tried to apply the STAFF software in these cases, and surprisingly it works sufficiently well.

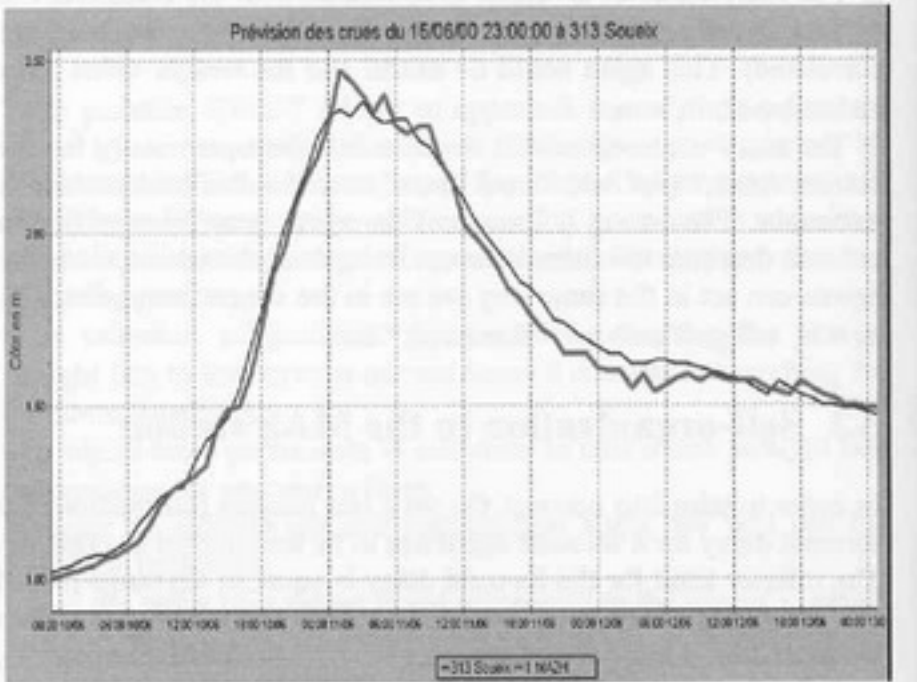


Figure 5: Flood forecast for an upstream station

To succeed in giving such a precise forecast on a station for which the traditional models do not work, STAFF principally uses

the measures obtained from sensors situated on neighbouring slopes. In distributing its entries this way, STAFF does no more model the flow propagation, rather the precipitations moving from a basin slope to another.

5. Properties of the system

The previous part was mainly developed to show without ambiguity the real efficiency of the STAFF software, in order to say that it is 'functionally adequate' according to the terminology of the AMAS theory. In fact, the theory claims that when components of a system are in co-operative situation, the system has a suitable behavior in the environment. It is only by reducing conflicts between forecast agents that STAFF reduces highly the forecast error. In order to know the efficiency of a learning theory we must have an idea about the space search and the amount of examples required to obtain a good function.

5.1 A huge space search

We have experimentally observed that the part each data entry takes in a forecast is bounded by the upper limit 2000. Usually STAFF takes into account the data coming from the ten past hours, and the network of sensors (raining information and river level) is about 80. In this usual configuration, the global space search is equal to $2000^{80 \times 10}$, corresponding approximately to 10^{2640} which is hugely bigger than the space dimension acceptable for classical algorithm.

In order to obtain a suitable model of the flood variation in a given location, we have used information from historical flood stored in files containing 10 days periods of data (240 hours). A learning phase generally consists in 5 to 10 flood periods, in which the system has typically 100 examples to learn (a rising water level has not a 10 days duration). Thus, the 10^{2500} space search is well explored in 500 to 1000 examples! This is not an isolated result because learning was done for fifty stations (fifty separate functions), but very surprising in regard to the results obtained from classical learning methods. At this stage of the AMAS theory we can not give a formal demonstration, but only a technical explanation composed of two parts:

5.2 A highly distributed adaptation

The AMAS theory imposes to search all the uncooperative situations a given agent could be faced to during its lifetime; considered during its perception, reasoning and action phases. Usual learning methods are focused only on some of those situations; for example, genetic approaches are mainly interested in competition between individuals for obtaining a behavior closed to the evaluation function. During the presentation part of the STAFF architecture we have seen three uncooperative cases: unproductiveness, conflict and concurrency. Conflict and concurrency are often studied, but we can reasonably think that unproductiveness (here when input data are missing) is always considered as an exception treatment by learning algorithms, and not another useful case for the system self-adaptation. The non-isolated cases showed in sections 4.2 and 4.3 are practical examples of this usefulness. Thus, taking into account all the encountered uncooperative situations for learning is another good way to improve the convergence speed.

5.3 The convergence speed

When an AMAS agent detects any uncooperative situation from its local point of view, it tries to suppress it locally by adjusting its behavior or even modifying its own function. This is not generally enough, and thus leads the concerned agent to call for help others agents. These called agents try to adapt their own behavior and even to create secondary helps, and so on... Thus, many agents could be requested to overcome only one initial unexpected event; a solution which could appear to be time consuming. On the opposite it is very flexible because the set of involved agents depends on the difficulty to solve the problem, and they intervene appropriately. When we have systems composed of agents acting in a highly intricate manner, this is certainly better than a rigid adjustment algorithm. Moreover, we see experimentally that the STAFF self-organizing process is no more than five seconds for a typical set of one thousand agents inside the system.

6. Related works

Classical physic-hydrological forecast models (Loumagne and Vidal, 1995) consist in generic formula the parameters of which are determined from measures on ground and from historical account of flood. Some of them are linear models which compute the forecast at $t+\Delta t$ for a station according to the flow growing between t and $t-\Delta t$ at the same station, others follow the principle of the "rain-flow" model which is function of a flood coefficient and the action delay of rain. Non linear models are also used to simulate the attenuation of the flood wave between upstream and downstream stations.

The building and adjustment of these formula are time expensive (several months) and do not always give good results at every point of the basin (upstream stations, new station) nor for every kind of flood (slow, with constant evolution phase, ...) or when the sensors break down.

Neural networks have been used to model flood forecast. An original approach (Vergnes, 1995) is based on a neural network to compute an hourly forecast which is then taken as an entry of the same network as many times as necessary to reach the forecast delay. The main advantage of this method compared to the classical ones is that it does not use hydrological knowledge about the basin excepted a notion of propagation delay. However this model is not operational because the number of neurons in the hidden layer is determined experimentally, which implies several tests for a station model.

7. Conclusion and perspectives

STAFF's results indicate its real adaptation capacity based on a process of self-organization based on two agents levels. We have shown that each level is necessary to tend at best towards the functional adequacy. The global function is really emergent compared with the agents activity which realize very simple treatments and compared with their self-organization process conducted by adjustment rules which do not depend on the searched global function.

A flood forecast system must be able to take into account time and space information determined by the topography of the basin. This is one of the reasons why we must spend many time to adjust classical physic-hydrological models. In the STAFF approach this information emerges from the learning process based on the AMAS theory. This can really be observed by the weight

associated to each entry of each hourly agent. And this is really an emergent behavior because the self-organizing process is only based on uncooperative analysis (uselessness, conflict, and concurrency).

Keith Sawyer (2000) says "In many computational models of emergence, the end state of the system is of primary interest. In collaborative emergence, the most interesting aspects of the simulation will be the incremental processes of emergence and downward causation". STAFF and the others applications using the AMAS theory are example of collaborative emergence, in which the goal is not to obtain a given end state but a never ending adaptation process because the systems are plunged in a dynamic environment. We think that it is an important class of MAS simulating hard problems and having specific characteristics (dynamic environment, non termination, huge space search).

References

Bras L.Rafael, Ivanov Valeri, Vivoni Enrique. The Value of Distributed Modeling to Flood Forecasting. NASA Land Surface Hydrology and GAPP Meeting, Potomac, MD, April 2001

Gleizes Marie-Pierre,Georgé Jean-Pierre, Glize Pierre. A theory of complex adaptive systems based on co-operative self-organization. Demonstration

in electronic commerce. Workshop on Self-Organizing Multi-Agent Systems, Milton Keynes, UK, July 2000, see also <http://www.irit.fr/SMAC>

Gleizes Marie-Pierre, Camps Valérie, Glize Pierre. A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. Fourth European Congress on Systemic, Valence, 1999 , see also <http://www.irit.fr/SMAC>

Sontheimer Thomas, Cornuau Pascale, Vidal Jean-Jacques, Glize Pierre. Application d'un système adaptatif pour la prévision des crues dans le bassin de la Garonne. Conférence SIRNAT'01, Systèmes d'Informations et Risques Naturels

Loumagne Cécile, Vidal Jean-Jacques, Feliu Claudine, Torterotot Jean-Philippe., Roche P.A.: Procédure de décision multimodèles pour une prévision de crues en temps réel. Revue des Sciences de l'Eau, 8 (4) (1995)

Vergnes Jean-Christophe. Etude de modèles de prévision à la station de Nant. Exploitation de l'utilisation des réseaux neuronaux en prévision de crues. Report DIREN (1995)

Sawyer R.Keith. Simulating Emergence and Downward Causation in Small Groups. Lecture Notes in Artificial Intelligence, Vol. 1979, Springer Verlag Berlin Heidelberg,(2000) 49-67

SHORTENING OF EVOLUTIONARY AGENT LEARNING USING SEQUENTIAL PATTERN METHODS

Jerzy J. Kórczak¹, Piotr Lipinski^{1, 2}

¹Université Louis Pasteur, LSIIT, CNRS, Strasbourg, France

²University of Wrocław, Institute of Computer Science, Wrocław, Poland

Email: {jjk, lipinski}@dpt-info.u-strasbg.fr

Abstract: This paper presents an optimization of intelligent agent systems that base their knowledge on evolutionarily-generated artificial experts built on a set of defined rules. Problems of delays in real-time financial data analysis systems are discussed and bottlenecks of real-time intelligent agents, especially the need to update or regenerate these agents, are analyzed. The proposed optimization concerns improvement of the agent regeneration genetic process, which reduces computing time without harming results. The idea consists of discovering recurrent patterns in a sequence of generated chromosomes and using these patterns to create new agents, thereby accelerating the process of rules selection. Results of a few experiments on real-life data from the Paris Stock Exchange are described. Moreover, some future directions are pointed out at the end of the article.

1. Introduction

Real-time data analysis remains a grand challenge. Methods aspiring to be applied in real time should not only be efficient, but also quick to compute. Real-time intelligent agents should learn as quickly as possible, otherwise the efficiency of the methods may lose significance because the results will become delayed.

This paper refers to a class of intelligent agents that base their knowledge on artificial experts built on a set of defined rules. These experts are generated on the basis of available past data and applied to new data until they are efficient. Since properties of analyzed data change with time, the efficiency of experts falls, after which they should be regenerated or updated. The process of expert generation is time-consuming, prompting many ideas for its improvement. Certainly, updating an expert takes less time than regenerating one, but efficiency of an updated expert is often lower than that of a newly-generated one. Dependency of expert updating and regeneration procedures, along with their efficiency and computing time, is an interesting area of study.

In this paper, an intelligent agent from a real-time financial data analysis system is considered as an example of the agent class mentioned above. The regeneration procedure is based on the additional knowledge extracted from a list of previously generated experts. Extraction of the knowledge is performed using discovering sequential patterns algorithm [1].

This paper is structured in the following manner. The next section defines the problem, describes the data structure and mentions several constraints. The financial knowledge base containing trading rules is described in Section 3. In Section 4 the rule selection process is briefly presented with a proposed improvement. Section 5 validates the approach by discussing a few experiments. Finally, Section 6 concludes the paper and points out some directions for further research.

2. Problem definition

Assume that an event occurs on the financial market at time t_0 . The information about the event reaches the trader at time t_1 due to technical reasons such as a transmission delay. As soon as the trader receives the information, he or she starts analyze it, resulting in making a decision and consequently taking an action at time t_2 . This decision is transmitted and reaches the market at time t_3 .

Although the data analysis methods and the decision support system may work well, the trader's decisions may be inappropriate if they reach the market too late. The trader makes a decision on the basis of the market state at time t_0 , but the decision is executed on the market at time t_3 . The situation on the market changes every second, so the market state at time t_3 may be completely different from that at time t_0 if the delay $t_3 - t_0$ is too great.

The delays $t_1 - t_0$ and $t_3 - t_2$ are often independent of the trader, as they mainly depend on technical causes such as data transmission speed or constraints of available financial services. However, the delay $t_2 - t_1$ depends only on the trader - to be precise, on the trader's data analysis and decision-making procedures. Thus, in order to speed up the trading process and increase trading efficiency, one should focus on minimizing the delay $t_2 - t_1$.

Methods of minimizing the delay will be presented later. Now, the problem will be revisited taking into consideration the nature of financial data.

Generally, there is an enormous amount of data available on the stock market. However, not all of them are necessary for the purpose of analysis. It is usually sufficient to use aggregated data of a given time period, meaning that the trader will not receive all data from the market as a continuous data stream, but at the end of a given period. For instance - every hour, he or she may receive a brief report containing aggregated data summarizing the situation on the market in the last period.

Let T be a given time period. At time t_0 , data referring to the interval $(t_0 - T, t_0]$ become available on the market. These data are being pre-processed and aggregated. The aggregated data are transmitted to the trader, who receives them at time t_1 . Hence, at time t_1 , the trader possesses summarized information about the market state in the period $(t_0 - T, t_0]$ and begins to analyze it. In consequence, the trader makes a decision which will be sent to the market at the time t_2 and will reach it at the time t_3 . As mentioned before, the delay $t_3 - t_0$ is of great importance for trading quality and efficiency.

In practice, the aggregated financial time series consists of opening, maximum, minimum and closing prices, the transaction volume, and the market index value aggregated within the period of one minute. The process of aggregation proceeds as follows: The stock price of the first transaction in the given period constitutes the opening price. The maximum and minimum prices are computed as the highest and lowest prices in the period. The price of the last transaction makes up the closing price. The transaction volume represents the number of traded stocks, usually in thousands of items. Finally, the market index is equal to the last market index value in the period. It may happen that there was no transaction during the considered period, in which case the prices are equal to the closing price from the previous period and the transaction volume is set for 0.

Besides the aggregated data that describe the market state, taking into consideration realized transaction and current rates, other information may be useful such as book order data, described in the following sections, or non-numerical data such as press dispatches.

The gathered data are processed using a financial knowledge base that stores a number of trading rules based on technical analysis.

3. Financial knowledge base

Technical analysis assumes that the future market behavior may be predicted on the basis of previous observations of stock quotations and a few other financial indicators such as the value of the market index [6], [16], [17]. Rules based on technical analysis use a large amount of past data, aggregated over a given period. Several rules are simple, requiring only a few recent quotations, and are thus easy to formulate and less time-consuming to compute. However, most rules work on longer periods and require more data, so computing them is more complex and more time-consuming.

In practice, one of the most important properties is the time of rule execution. Some of the above-mentioned rules require more time to be calculated and executed while the application of others is less time-consuming. The rule execution time affects also the rule efficiency, because the market situation may change during the calculation. Thus, in practice, sometimes simpler rules turn out to be more efficient than complicated rules, even if the latter seem to have a better theoretical background.

4. Overview of the rule selection process

The rule selection process constitutes the soul of the system. It is responsible for the selection of a proper set of rules taken from all the rules in the knowledge base, in order to eliminate those rules inefficient in the current market state and choose those suitable in the current situation. The selected set of rules is referred to as an expert. The selection must

be done every time the market state changes. In practice, this is performed every minute unless the market state is unusually stable.

The rule selection process is based on a genetic algorithm [12]. The detailed description of genetic algorithms is not an object of this paper and may be found in papers such as [2], [3], [4], [7], [10], [14]. Only a brief overview of the algorithm is presented here, emphasizing time consumption aspects.

The genetic algorithm deals with a population of chromosomes. Each chromosome is a binary string representing a set of rules, an expert. The presence of a given rule in the expert chromosome is indicated by 1 and the absence by 0. For instance, 00110110 represents a set composed of the rules numbered 2, 3, 5 and 6. The algorithm begins with a randomly generated population that evolves through genetic operators. It searches for a suitable set of rules maximizing an objective function. The function is founded mainly on the profit generated by the set of rules and some risk indicators. Details of the objective function construction were presented in [12].

The goal of the modification is to improve the throughput of the expert discovery process by shortening the time of the computing process. The idea is to reuse the previously-generated experts - more precisely, it is to discover patterns that represent the most frequently used trading rules on the basis of codes of recent experts. A trading expert pattern is an abstraction from a sequence of the best experts that keeps rules that recur in specific market situations. Using this approach, the newly created experts inherit "the building blocks" from the pattern that is already a subset of a successful family of proven trading rules.

The expert patterns are dynamic and generative. They are dynamic because they are recursively computed from the previously generated experts. Therefore, a large part of the new expert chromosome is already defined by the most recent pattern. They are also generative patterns because they indicate how to create new experts. Note that only the part of the chromosome that is not covered by the pattern is varied during the process of expert discovery. Of course, the process will be quicker if the number of instantiated positions in the chromosome is higher.

5. Experiments

Experiments have been carried out online over the space of a few days on the Paris Stock Exchange. Trading experts were generated every minute based on quotations and bid-and-ask offers deferred by one minute. Typically, to create an expert, the genetic algorithm uses 250 trading rules executed during the preceding four hours of trading. The population of 50 experts is evolved during 400 generations. These parameters resulted from experiments carried out on various financial series. The overall time of expert discovery for one security using the typical genetic algorithm was approximately 20 seconds on the Pentium IV 2 GHz processor. The goal was to shorten the time to compute an expert in order to maximize the expert generation throughput.

Table 1. An extraction of the last 15-minute pattern of trading rules

Time	f ₁	f ₂	f ₃	f ₄	f ₅	f ₆	f ₇	f ₈	f ₉	f ₁₀	f ₁₁	f ₁₂	f ₁₃	f ₁₄	f ₁₅	f ₁₆	f ₁₇	f ₁₈	f ₁₉	f ₂₀
15:45	1	0	1	1	1	0	0	1	0	0	0	1	0	1	0	1	1	0	1	1
15:46	1	0	1	1	1	0	0	1	0	0	0	1	0	1	0	1	1	0	1	1
15:47	1	0	1	1	0	1	1	1	1	0	0	0	0	0	0	1	1	0	1	1
15:48	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	0	1
15:49	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	0	1
15:50	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	0	1
15:51	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	0	1
15:52	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	0	1
15:53	1	0	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	0	1
15:54	1	0	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1	0	1	1
15:55	1	0	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1	0	1	1
15:56	1	0	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1	0	1	1
15:57	1	0	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1	0	1	1
15:58	1	0	1	1	0	0	0	0	1	0	0	0	1	1	1	1	1	0	1	1
15:59	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1
Freq. (N)	15	0	9	15	3	1	1	3	12	6	1	8	5	13	5	15	15	0	8	15

In the experiments, only 10% to 15% of rules have been either constantly used, or not used at all, during the last 360 minutes. Therefore, to find a trading pattern, shorter intervals have been analyzed. Table 1 shows an extraction of the last 15-minute pattern of trading rules. The 15-minute pattern covers approximately 40% of the rules in expert chromosomes. The frequency of the usage of these rules over a period of 15 minutes is presented in Table 2. The 10-

minute pattern and the 5-minute pattern cover 60% and 85%, respectively. Based on the observations, the two last intervals were considered to be too short to produce an expert pattern. In addition, in these last two cases the search space is too restricted, inhibiting the exploration by genetic operators. The 15-minute pattern seems to be a compromise among the quality of trading, the computing time, and the space exploration facility. In this case, according to numerous trials, the time of expert generation has been shortened by approximately 50%.

In the experiment, the 15-minute pattern was computed each minute and then used to generate a new trading expert. The search space was reduced. Nevertheless, a large number of variant experts were evolved using only the part of the chromosome not covered by the pattern. To avoid premature convergence and local solutions, after every 10 minutes for a period of 5 minutes the experts were generated according to the classical algorithm. The process of pattern discovery was then run again. This solution makes the expert design extensible, reusable and adaptable to specific market situations.

Table 2. The frequency of the usage of 250 trading rules over a period of 15 minutes

N	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Number of rules, occurring N times	65	10	6	10	4	6	12	10	9	13	6	3	7	10	6	71

The results of these experiments are encouraging. Many tests show that the pattern captures a subset of an already successful group of rules applied to a recurring trading problem that arises within a certain context. On average, decreasing the time of expert generation has considerably improved the overall throughput of the system. Two aspects of this improvement should be underlined. First, the system is now able to generate a trading expert in approximately 12 seconds after the latest market observation. Consequently, the interval for quote aggregation may be reduced from one minute to less than 15 seconds. Second, using the one-minute deferred quotations, a trading agent may obtain advice for more securities traded on the market. Of course, the results might be more impressive if one used more powerful computers.

Figure 1. Dependency of efficiency and computing time on size of discovered sequential patterns.

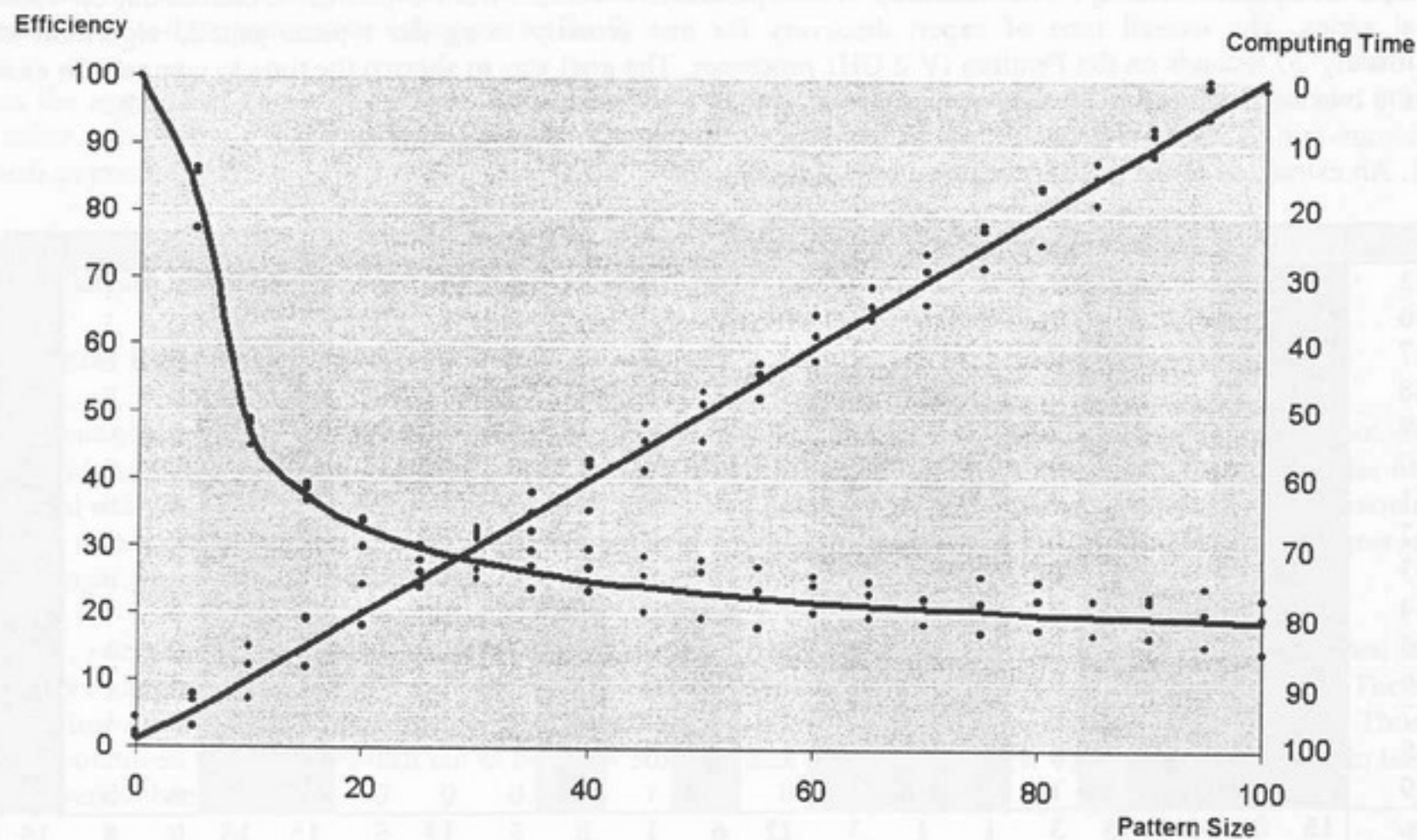


Figure 1 illustrates the dependency of efficiency and computing time on the size of the building blocks of discovered sequential patterns. In the case where the patterns are not taken into account, i.e. the size is equal to 0, the system achieves the maximum efficiency because the expert is searched through the whole space without any constraints. However, in this case, the required computing time is high because the calculation is complex. If a small pattern is introduced, the system efficiency becomes slightly lower but still sufficient. The computing time becomes a bit shorter. On the other hand, when the pattern covers a large part of chromosomes, the efficiency is rather low because the

process of expert discovery is highly constrained and thus the result cannot be optimal. However, since the size of search space becomes small, the process is completed in short time. Certainly, the exact shape of curves presented depends on the particular case.

6. Conclusion and future directions

This paper addresses the problem of delay in decision support systems for stock trading. Even if the methods of data analysis used are suitable and efficient, decisions made on the basis of their results may turn out to be inappropriate because they reach the market too late.

In this paper, an improvement of the system based on a genetic algorithm has been proposed. The idea, consisting of searching for recurrent patterns in trading experts and using them to create new experts, permits to accelerate the process of selecting trading rules. However, more experiments are needed to further investigate the efficiency of the approach on longer test periods and various time series.

At the moment, patterns are looked for based on a set of genetic experts that come from the last 15 minutes of trading. It is tempting to shorten this period. This would, however, cause the pattern to become larger, but this may provoke premature convergence of the genetic algorithm, and hence the mutation operators would have to be modified in order to avoid focusing on a small part of the search space.

On the other hand, one can analyze patterns occurring over longer periods in order to promote or eliminate rules that are permanently chosen or rejected. In this case, it would appear to be necessary to develop a strategy which would allow a rule to come back to the previous status after a certain time. Otherwise, a rule might be eliminated forever, even if it would turn out to be efficient in the future.

Reduction of the computing time enables the further development of the trading system. Currently, the data aggregation period is equal to one minute, which implies that the system generates advice every minute. Since the decision making process has been accelerated, it is tempting to lessen the aggregation period so that the system will react faster to the market state, for instance, every 30 seconds.

Finally, further research would be needed to estimate precisely the reduction rate which could be obtained due to the modification. Further experiments are necessary to make a deeper analysis, but the preliminary results are extremely promising.

References

- [1] Agrawal, R., A., Srikant, R., "Mining Sequential Patterns", Proc. of the International Conference on Data Engineering, ICDE, Taipei, Taiwan, 1995.
- [2] Allen, F., Karjalainen, R., "Using Genetic Algorithms to Find Technical Trading Rules", Journal of Financial Economics, 51, 1999, pp.245-279.
- [3] Back, T., "Evolutionary Algorithms in Theory and Practice", Oxford University Press, New York, 1995.
- [4] Bauer, R., J., "Genetic Algorithms and Investment Strategies", Wiley, 1994.
- [5] Caldwell, R., "Nonlinear Financial Forecasting", Proc. of the First INFFC, Finance and Technology, 1997.
- [6] Colby, W., Meyers, T., "The Encyclopedia of Technical Market Indicators", Down Jones-Irwin, 1990.
- [7] Davis, L., Handbook of Genetic Algorithms, Van Nostrand, 1991.
- [8] Dempster, M. A. H., Jones, C. M., "A Real-Time Adaptive Trading System using Genetic Programming", Quantitative Finance, Vol. 1, 2001, pp. 397-413.
- [9] Dunis, C., et al., "Optimising Intraday Trading Models with Genetic Algorithms"
- [10] Goldberg, D. E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley, 1989.
- [11] Korczak, J., Lipinski, P., Roger, P., "Evolution Strategy in Portfolio Optimization", [in] Artificial Evolution, ed. P. Collet, Lecture Notes in Computer Science, vol. 2310, Springer, 2002, pp. 156-167.
- [12] Korczak, J., Roger, P., "Stock timing using genetic algorithms", [in] Applied Stochastic Models in Business and Industry, 2002, pp. 121-134.
- [13] Markowitz, H., "Portfolio Selection", Journal of Finance, 7, 1952, pp. 77-91.
- [14] Michalewicz, Z., "Genetic Algorithms + Data Structures = Evolution Programs", Springer Verlag, New York, 1994.
- [15] Moody, J., "Performance Function and Reinforcement Learning for Trading Systems and Portfolios", Journal of Forecasting, 1997.
- [16] Murphy, J., "Technical Analysis of the Financial Markets", NUIF, 1998.
- [17] Neely, C., Weller, P., Dittmar, R., "Technical Analysis in the Foreign Exchange market Profitable? A Genetic Programming Approach", Journal of Financial Quantitative Analysis, 32, 1997, pp.405-426.
- [18] Sharpe, W., "Capital Asset Prices : A Theory of Market Equilibrium under Conditions of Risk", Journal of Finance, 19, 1964, pp. 425-442.
- [19] Weigend, A., S., Gershenfeld, N., A., "Time Series Prediction: Forecasting the Future and Understanding the Past", Addison-Wesley, 1993.

Autonomous Itinerary Planning for Mobile Agents

Christian Erfurth*

*Friedrich-Schiller University Jena
07740 Jena, Germany
cen@inf.uni-jena.de

Wilhelm Rossak†

†Friedrich-Schiller University Jena
07740 Jena, Germany
rossak@inf.uni-jena.de

Abstract

In the field of intelligent distributed computing in dynamic computer networks, we focus on mobile agents and mobile agent systems, especially on the itinerary problem and the adaptivity of mobile agents. This paper outlines our current research activities which are based on the Tracy mobile agent system, developed at FSU during the last few years. For our purposes we expand this system by adding intelligent and adaptable system components, as well as a basic information infrastructure.

1 Introduction

Mobile agents are autonomous and proactive software entities which act on behalf of an owner, communicate and cooperate with each other, and have the ability to migrate through a heterogeneous network of computers [White (1996); Bradshaw (1996)]. We at FSU look at (mobile) agents mostly from the viewpoint of distributed computing systems, i. e. as a new paradigm for open peer-to-peer computing.

In a typical scenario, a mobile agent visits more than one host in a network of so-called agencies (dedicated service platforms) to fulfill its task. Thus, it needs a "travel plan" when it starts its journey, a usually fixed plan that is provided by its programmer/owner. However, due to the size and possibly dynamical behavior of modern (ad-hoc) networks that change much faster than the agent-owner's perception of the net, mobile agents must by now develop the ability to construct and pro-actively adapt their own travel plans, which we call an *itinerary*.

We at FSU, in the TRACY project [Tracy Project], focus as one hot-spot on the itinerary problem of mobile agents by improving the basic technical efficiency of their migration and by increasing their autonomy and adaptivity in the planning and travel phase. This paper outlines our current research and first results in the area of itinerary planning.

2 A Sample Scenario

Before we start to outline our framework we want to introduce the scenario for a typical application. It should help to describe our idea and our goals.

A user (the owner) hands over a task to an agent. Such a task should not contain information on *HOW* to fulfill its demands. Hence, the agent has to organize the journey through the network by itself. Therefore, the agent searches for suitable services on a map provided by the

local agency. This map contains information on services within the local net domain (set of neighboring agencies) and some network characteristics. The search result is a set of agencies within the network that should be visited. Now the agent may trigger a Route Planner to use the available map's information on connection topology and qualities to identify a possible (fast) trip through the network. The result is a first local travel plan – the itinerary. Before the agent begins the trip, it might use a Migration Planner to optimize the trip from an efficiency perspective. Now the agent "executes the itinerary" and starts the migration. During the trip the agent visits service points at remote agencies and communicates and cooperates with other agents.

At any point in time, but at least when migrating to further away agencies in another net domain, the agent may fine-tune and re-adapt its itinerary. This is achieved by taking advantage of the new maps now available locally. This "re-planning process" can be repeated as many times as it seems necessary for the agent to fulfill its task. Finally, after finishing its trip, the agent hands over the results to its principal. This might include a description of the visited agencies, a kind of travel report. During the migration, there is no communication necessary between agent and owner.

As indicated, we want to provide an infrastructure that enables agents to be more autonomous. A user should concentrate on *WHAT* the agent has to do and not on *HOW*. We also want an itinerary, and thus the traveling agent, to be highly adaptive to changes in the environment and to be scalable for execution in very large networks. To achieve these goals, a general framework and the related infrastructure have been designed and tested, using the scenario we just described as a reference model and the TRACY mobile agent system as implementation technology.

3 Framework Overview

As indicated in the last section, there are a set of modules which support agents to fulfill their tasks. The framework consists of three modules: The *Map Module*, the *Route Planner Module*, and the *Migration Planner Module* (see fig. 1).

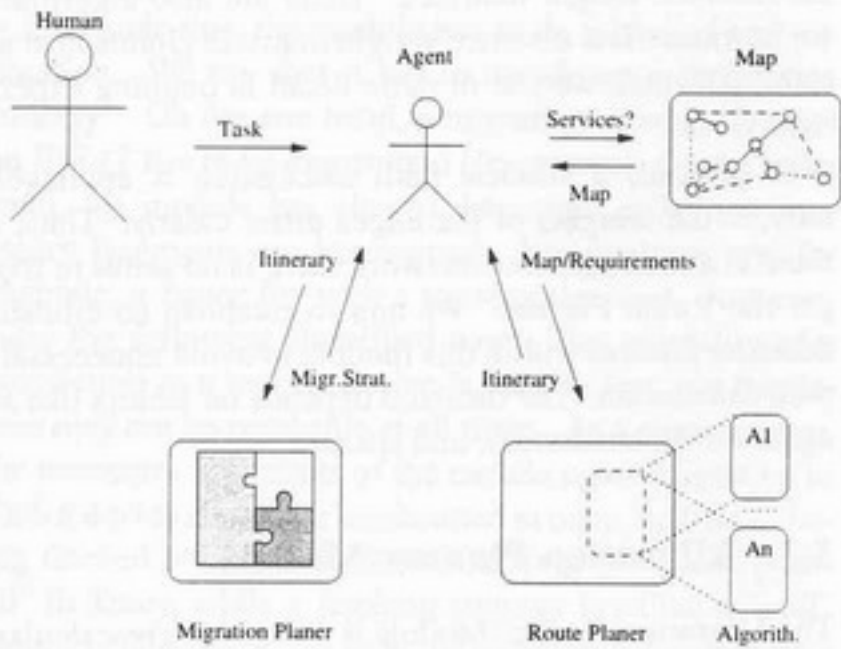


Figure 1: Major Infrastructure Components

3.1 Map Module

The Map Module [Schreiber (2002)] is responsible for collecting data to create a local information base which we call *map*. Every agency manages its local map. Because of the given quantity, it is not possible to store complete information of the network of all agencies. Hence, the map is split into a local area map (current, exact data) and a map of remote areas (summarized, potentially inconsistent data). This technique is like a fish-eye view known from photography - the further away from the center the more blurred.

Remark: A local area of an agency corresponds with the (sub-)network the agency belongs to. So the number of agencies within a local area is in direct relation to the number of nodes within the network. We use the Tracy Domain Concept [Braun et al. (2001a)] to network the agencies within the local area (a *domain*) and to network domains (via a Domain Master). For each domain, there exists a *Domain Manager* which manages and represents a domain (see fig. 2).

The Map Module contains net monitoring sensors which collect map data. There are two kinds of data: Agency properties like a service list, supported protocols, etc., and line characteristics like bandwidth, latency, etc. Furthermore, the map is split into a local and a remote map, each structured in a node information part (a list) and a line information part (a matrix) (see fig. 3). The update frequency for local area data is higher than that for remote areas (represented by the Domain Masters).

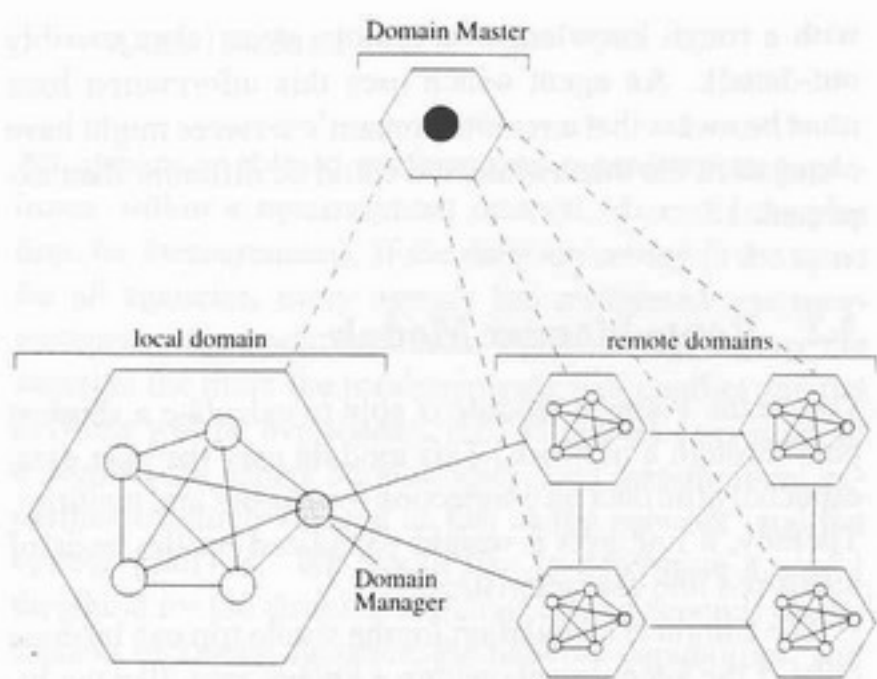


Figure 2: Tracy Domain Concept

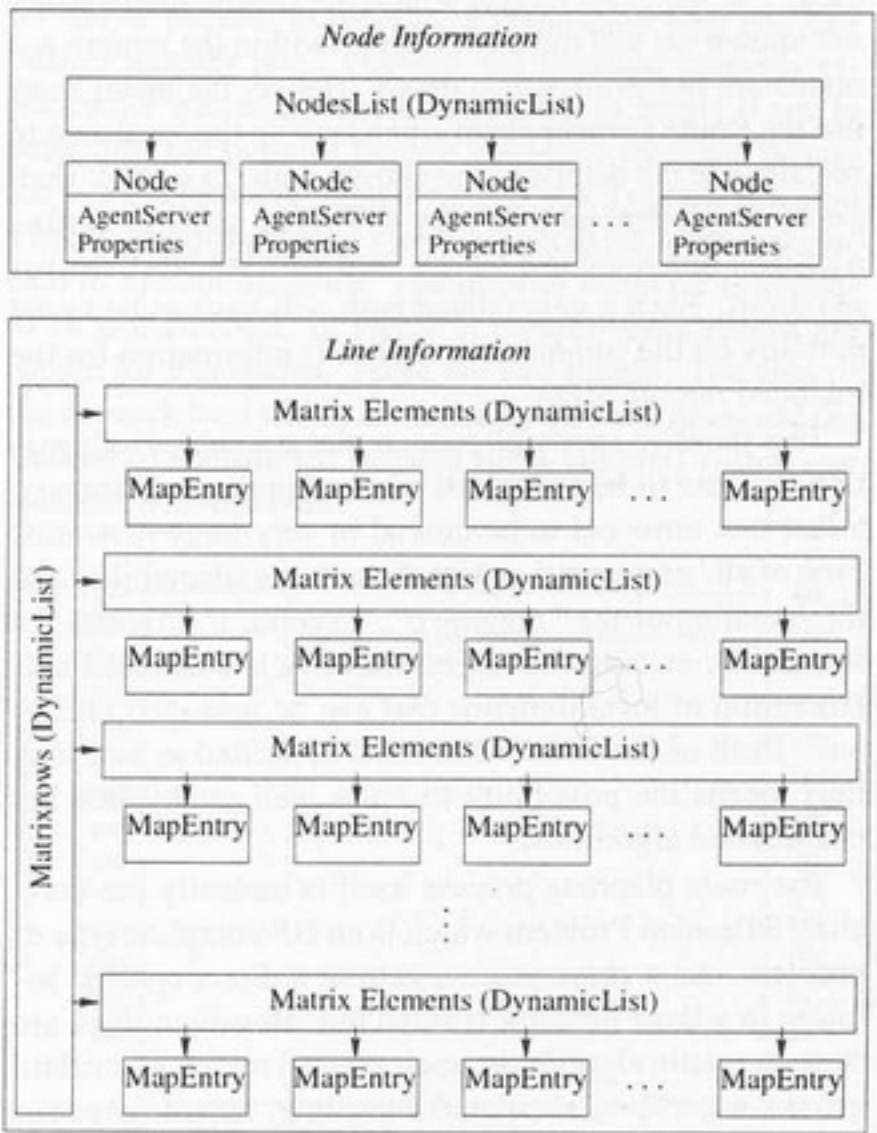


Figure 3: Map Data Structure

At regular time intervals the sensors do star-shaped measurements within a local area to collect line data. The measurement results and the agency properties then are exchanged between agencies to complete/update the map data. A Domain Manager runs also measurements with other domains (remote Domain Manager). Thereby, summarized domain maps are exchanged (information reduction – fish-eye technique).

As a consequence, each agency has a map with nearly complete and up-to-date information on the local area and

with a rough knowledge on remote areas (also possibly out-dated). An agent which uses this information base must be aware that a remote domain's services might have changed in the meanwhile and could be different than expected.

3.2 Route Planner Module

The Route Planner Module is able to calculate a shortest path through a network. This module uses the map data, especially the data on connection topologies and qualities. Thereby, a line gets a weight calculated on the basis of measured line characteristics.

The complete calculation for the whole trip can be done only, if the agent travels within a known area, like the local domain and between known remote domains. If the agent visits other remote domains, information is incomplete, i. e. the exact targets within the remote domains are not known yet and the line qualities within the remote domains are not available in detail. Hence, the agent must use the Route Planner again when leaving the local area to re-calculate the details of the chosen path. In spite of that, generating a general path may still make sense to outline a generic first route that can be used as basis for all further iterations. Such a generalized path will have to be based partially on the summarized (blurred) information for the included remote areas.

This iterative approach reduces the amount of information that has to be processed when planning an itinerary, a fact that turns out to be crucial in very large networks. First of all, an iterative approach supports adaptability and the handling of the "unknown". Second, it provides for scalability, as even the largest network is conceived as a federation of local domains that can be managed one by one. Each of the domains in itself is limited in size and, thus, opens the possibility to reuse well established but not scalable algorithms.

The route planning process itself is basically the Traveling Salesman Problem which is an NP complete type of problem. As a consequence, getting a direct optimal solution in a large network is ruled out. However, there are some heuristic algorithms (such as local search algorithm, genetic algorithm, simulated annealing, neural networks etc.) that have been applied extensively for solving such problems [Jonhson and McGeoch (1997)] in a more limited environment. The comparative performance of the algorithms depends on the problem and the given detailed circumstances.

To be able to test different algorithms, the Route Planning Module has an interchangeable calculation component (see fig. 1). Our current results indicate that classical local optimization algorithms (2-Opt, 3-Opt, Lin-Kernighan are quite simple and very fast algorithms which deliver nearly optimal results) seem to be the best fit for our application (Jonhson and McGeoch (2002) compares various algorithms). In this type of algorithms, in order to improve a feasible tour, it is modified by delet-

ing some edges, thus breaking the tour into paths and then reconnection those paths in another possible way [Lin (1965)]. Basically the algorithms in this family differ in the number of edges which will be exchanged: 2-Opt uses 2 edges, 3-Opt uses 3 edges, and Lin-Kernighan uses a variable number of edges.

These algorithms are designed to handle symmetrical distance/weight matrices. There are also algorithms for asymmetrical distance/weight matrices [Johnson et al. (2002)], which we test in more detail in ongoing experiments.

In general, a shortest path calculation is applicable only, if the weights of the edges differ clearly. Thus, if there is a homogeneous network there is no sense to trigger the Route Planner. We aim to establish an efficient decision process within this module to avoid unnecessary path calculation. The decision depends on factors like an agent's size, bandwidth, and latency.

3.3 Migration Planner Module

The Migration Planner Module is designed to re-calculate the itinerary to achieve an efficient journey. This module is more Tracy-specific and based on the Tracy Migration Component (supports fine-grained migration, see Braun et al. (2001b)). To resume execution of a migrated agent on a remote agency, the agent's code with all referenced code parts (transitive hull) is needed as well as the agents execution state and its current data. The state and the data have to be transmitted in any case. But a code part (in Java normally a class file) is needed only, if the execution needs code contained in it. Our goal is to transmit only those code parts which are needed at the remote agency for execution (see fig. 4). Thus, the network load caused by the agent is reduced and transmission times are shortened. Under some circumstances, it is even possible to split a single class into smaller classes or reduce a class using byte-code analyzing techniques [Fensch (2001)].

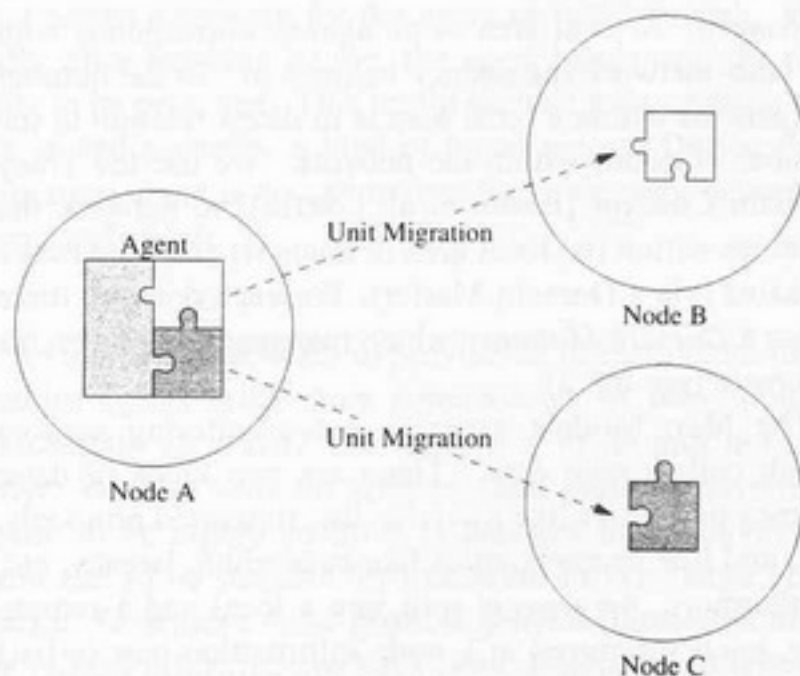


Figure 4: Migration of selected agent parts

Of course, while executing the agent, necessary code *fragments* may also be downloaded from elsewhere dynamically. The task of the Migration Module is to decide what is more efficient: Transmit all code parts not available at the target agency along with the agent state and data, transmit only (possibly) necessary code parts/fragments, or even no code parts at all (dynamical download).

To decide this, the module has to do a kind of cost estimation. We say that it has to decide on a "migration strategy". On the one hand, a migration strategy focuses on *WHAT* has to be transmitted (fragments). On the other hand, the module has also to determine *HOW* the necessary fragments are transmitted. Net qualities are, for example, a factor for such a strategy decision. Just consider the following simplified case: The reliability of a connection to a mobile device is usually low and the device may not be reachable at all times. As a consequence, the necessary fragments of the mobile agent migrating to such a device should be transmitted at once, instead of being fetched on demand. Such a strategy is called "push-all" in Tracy, while a fetching strategy is called a "pull" strategy. In general, the exact relation between all factors is quite complex and has to be determined using mathematical models and experimental validation. For further details, we refer to Braun et al. (2001b).

In our framework, the Migration Module is an optimization module that complements the Route Planner and the Map Module, but is not a mandatory part of the itinerary planning algorithm. As its execution is fairly expensive in terms of used resources, and it might even happen that in the end the Route Planner and the Migration Planner contradict each other (!), the Migration Module is invoked only under specific circumstances, i. e. in the case that the network topology is fairly stable and the agent itself has a long life with fixed or very similar service requirements. In this case the optimization module runs once and the agent may use the proposed strategy repeatedly, if it fits its needs.

4 Trial Runs

This section describes first tests with the system's modules. First results are indicated within the subsections.

4.1 Creation of Maps

We did some measurements regarding the performance and behavior of the Map Module. Our interest concentrates on network load which is caused by the module while collecting data and on the type of dynamic behavior in the network that we are able to map with this module.

The normal way the module's sensors work at an agency can be described as follows:

```
FOR every known agency s DO
  do measurement with s;
```

```
    WAIT duration;
LOOP.
```

An agency is able to perform one measurement experiment within a measurement interval (duration plus time for measurement). If the duration setting is the same for all agencies, every agency has performed one measurement experiment within an interval. The shorter the duration the more the measurements will conflict and the network will be overloaded, too. If duration is set to 0 seconds the sensor continuously does measurement experiments which will run as fast as the network (and the system itself) is. We would like to determine a useful threshold for the duration which of course depends on the number of known agencies, the network capabilities, and the kind of measurement experiment.

For our tests, we chose the bandwidth measurement experiment (causes most network load) which transmits 64 KByte packets to measure transmission time and calculates transfer rate. Thereby, the duration between experiments was decreased from 32 sec to nearly 0 sec to determine the point of highest map-able dynamic behavior. We did measurements with 2, 3 and 5 agencies within a homogeneous network environment to get a first impression of system behavior. The general behavior is similar in all experiments. In figure 5, measurement results are shown for 5 agencies. There are two graphs: The sum of the network load which is caused by all 5 agencies and the amount of transmitted packets (data volume) within one measurement interval.

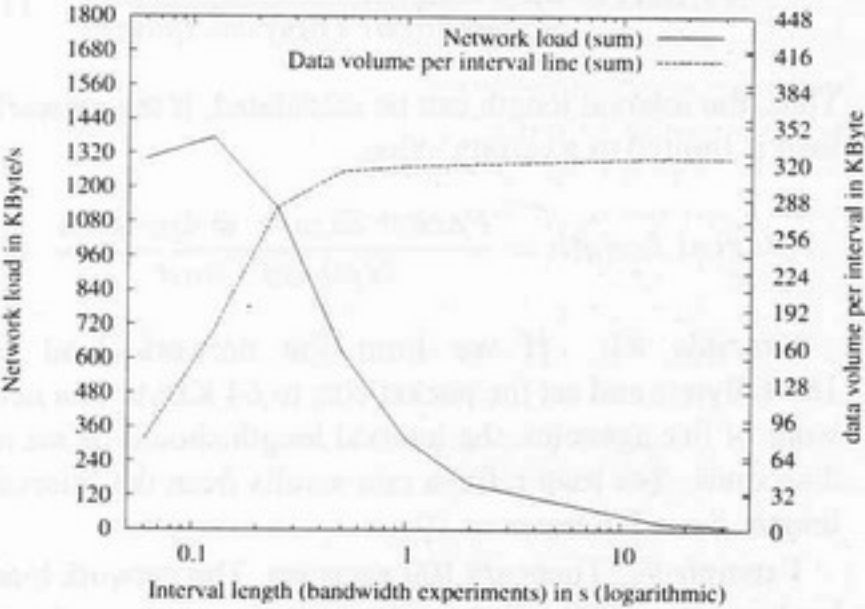


Figure 5: Network load caused by bandwidth experiments

Normally, all agencies should finish one experiment within one interval. Thus, the data volume should be about 320 KByte (5 agencies, each transmit one packet of size 64 KBytes) as long as the defined duration is nearly equal to the complete measurement interval. The graph "Data volume per interval" in figure 5 shows a distinct behavior: At about 0.5 sec the data volume per interval decreases and finally crashes. The experiment's time becomes significant greater than the defined duration. The

network load increases until a certain saturation at about 0.1 sec (graph "Network load" in figure 5) is achieved and decreases with shorter duration values (experiments influence each other). As expected, the transfer rate crashes, if the defined duration is too short and experiments influence each other too much. Looking at figure 6, the measured transfer rate crashes at a duration of about 1 sec.

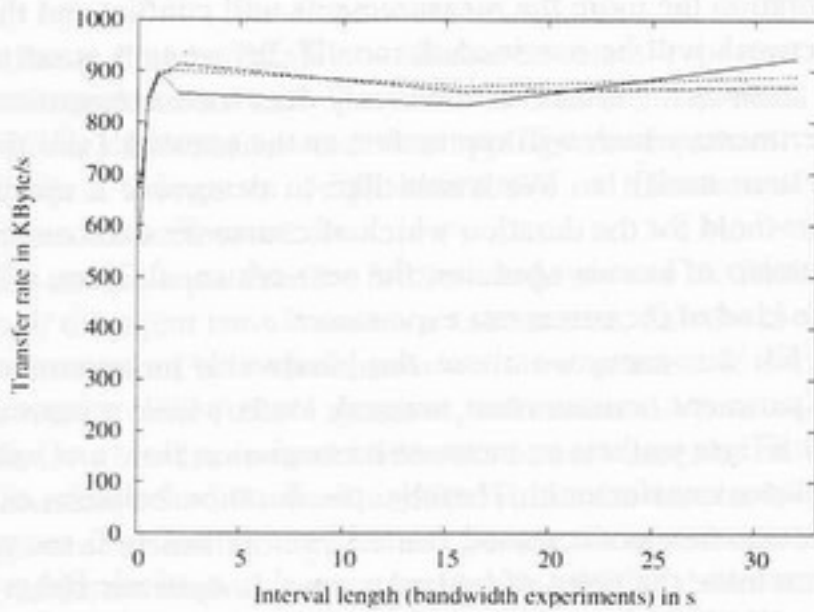


Figure 6: Measured transfer rate at an agency

The theoretical network load, i. e. the whole network traffic caused by the bandwidth experiments, can be calculated by:

$$Netload = \frac{Packet\ Size * \#Agencies}{Experiment\ Interval\ Length} \quad (1)$$

Thus, the interval length can be calculated, if the network load is limited to a certain value:

$$Interval\ Length = \frac{Packet\ Size * \#Agencies}{Netload\ Limit} \quad (2)$$

Example #1: If we limit the network load to 160 KByte/s and set the packet size to 64 KByte in a network of five agencies, the interval length should be set to 2 seconds. The map refresh rate results from the interval length: $2s * 5\ Server = 10s$.

Example #2: There are 100 agencies. The network load limit is set to 1000 KByte/s (distributed over the net) and the packet size is 64 KByte. Then, the interval length can be set to 6.4 seconds. The refresh rate on the map is about 10 minutes for this case.

The refresh rate we can achieve limits the type of dynamic behavior we are able to manage. If changes happen faster than we are able to refresh map data, they might go unnoticed. As our two examples indicate, we look at an upper limit of about ten minutes (100 agencies are a pretty well loaded local area) and get down to seconds with a smaller number of agencies. This is not hard real-time behavior, but opens a wide number of possible applications for this type of autonomous route planning.

4.2 Planning a Shortest Path

We also ran first tests with the Route Planner module. In networks with real traffic we faced some difficulties due to heavy oscillations of transfer rates. In spite of that, our first results indicate that it makes no sense to do extensive optimizations with the route planner within a homogeneous network environment. In fact, this is not a surprising result. Our future tests will focus on determining how heterogeneous a network has to be so that formal route planning is efficient and cost effective.

4.3 Optimize Network Load

The implementation of the Migration Planner is not finished yet. Thus, we made only a few tests which indicate that an optimization is at least useful if it can be used again and again for a certain mobile agent or for a set of agents (run once, use repeatedly) and the environment's dynamic behavior is not too high. Furthermore, an optimization is only feasible for known parts of an itinerary. Thereby, the level of knowledge on these parts effects the quality of an optimization. Nevertheless, it should be possible to provide always a "next-hop-optimization" especially for highly dynamic itineraries.

Future tests will be designed to show in which cases an optimization run is efficient and what quality of knowledge is essential.

5 Summary and Outlook

In this paper we provided a look at our framework to improve the pro-activity of mobile agents. Thereby, we have identified three modules: The Map Module, the Route Planner, and the Migration Planner. With the help of these modules, an agent is able to adapt to a changing network and service environment autonomously. Thereby, the locally available maps in all domain nodes are the distributed information base. Changes are first detected by and then reflected in those maps. Hence, we are able to map an environment which changes not too fast and inherits a certain stability, as network sensors must be able to keep track of changes. Currently, we run measurement experiments to determine the threshold for the dynamic network behavior we can master. Especially the Migration Planner Module is, therefore, also still under steady construction. First results indicate that we are able to handle changes in a network that stabilizes at least for a few minutes. In our future work we will characterize those types of environments further. Furthermore, we will finish soon our implementation and optimization efforts (Route and Migration Planner), and characterize the application of our framework in more detail by determining bounds and rules for necessary decisions.

References

- Jeffrey Bradshaw, editor. *Software Agents*. The MIT Press, Menlo Park, CA, 1996.
- Peter Braun, Jan Eismann, and Wilhelm Rossak. A Multi-Agent Approach To Manage a Network of Mobile Agent Servers. Technical Report 12/01, Friedrich-Schiller-Universität Jena, Institut für Informatik, 2001a.
- Peter Braun, Christian Erfurth, and Wilhelm Rossak. Performance Evaluation of Various Migration Strategies for Mobile Agents. In *Fachtagung Kommunikation in verteilten Systemen (KiVS 2001), Hamburg (Germany), February 2001*, 2001b.
- Christian Fensch. Class Splitting as a Method to Reduce Network Traffic in a Mobile Agent System. Master's thesis, Friedrich-Schiller-Universität Jena, Institut für Informatik, 2001.
- Gregory Gutin and Abraham P. Punnen, editors. *The Traveling Salesman Problem and its Variations*. Kluwer Academic Publishers, May 2002.
- David S. Johnson, Gregory Gutin, Lyle A. McGeoch, Anders Yeo, Weixiong Zhang, and Alexei Zverovitch. Experimental Analysis of Heuristics for the ATSP. In Gutin and Punnen (2002), pages 445–488.
- David S. Johnson and Lyle A. McGeoch. The Traveling Salesman Problem: A Case Study in Local Optimization. In E.H.L.Aarts and J.K.Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley and Sons, Ltd., 1997.
- David S. Johnson and Lyle A. McGeoch. Experimental Analysis of Heuristics for the STSP. In Gutin and Punnen (2002), pages 369–444.
- S. Lin. Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal*, 44:2245–2269, 1965.
- Steffen Schreiber. Beschreibung und Analyse von dynamischen Netzen für Agentensysteme. Master's thesis, Friedrich-Schiller-Universität Jena, Institut für Informatik, April 2002.
- Tracy Project. Tracy – The Mobile Agent System, 2003. URL <http://tracy.informatik.uni-jena.de>.
- James E. White. Mobile agents. In Bradshaw (1996), pages 437–472. URL <http://www.genmagic.com/agents/Whitepaper/whitepaper.html>.

Adaptive Firms and Emergent Organizational Forms

Zahia Guessoum

LIP6, OASIS team
4 place Jussieu, 75252 cedex 5
(331) 44 27 87 43
Zahia.Guessoum@lip6.fr

Lilia Rejeb

LERI, MODECO team
²Rue des Crayères, REIMS Cedex2
(331) 44 27 87 43
redjeb@poleia.lip6.fr

Rodolphe Durand

EM Lyon
23, avenue Guy de Collongue BP174,
69132 Ecully Cedex
durand@em-lyon.com

Abstract

In this paper, we present a multi-agent system which models a set of firms in competition with each other within a shared market. The firms are represented by agents; each firm is represented by a reactive or adaptive agent. We show the advantages of adaptive agents to represent firms. Moreover, we underline the limits of the economic models which represent only the firms without considering the organizational forms. We then propose to integrate the organizational forms. We introduce a model of organizational forms and define the various interactions between the firms and the organizational forms.

1 Introduction

Several models of firm dynamics and economic evolution have been developed recently based on empirical studies of industries. Works by Baum and Rao (1999), Lewin and al. (1999) and others investigate on historical data to explain how institutional, ecological, and strategic factors influence firm performances and survival.

Complementing the empirical analysis of historical data, other works intend to simulate the economic evolution -see for example (Mizuta and Yamagat, 2001), (Lomi and Larsen, 1996), (Levinthal, 1997), (Bruederer and Singh, 1996), etc. They often study a set of firms in competition. They model the internal structure of firms, their decision process and the various interactions and relations which exist between them. However, no work, as far as we know, studies economic models which represent explicitly firms and organizational forms and relationships between them.

Multi-agent systems provide a good paradigm to model these populations of evolutionary firms which are characterized by a collection of entities, which interact in a dynamic environment. This paper aims to present a more complete economic model and a multi-agent simulation of this model. We thus propose an economic model which integrates firms and organizational forms, and represents the various relations between them: firm-firm, firm-form and form-firm. So, the model defines the relation between 1) the micro level which is represented by the firms, and 2) the macro level which is represented by the organizational forms. It is then used to show:

- how do variations of the firm's parameters affect the organizational forms?
- how do variations of organizational forms influence the firms?
- how do firms interpret these influences?

It is therefore necessary to first model the firms and then the organizational forms.

This paper is organized in four sections. In the second and third sections, the model of the firms and their representation by a multi-agent system are presented. The fourth section defines and models organizational forms. It also describes and models the interactions between the firms and the forms.

2 Firms

The proposed model defines a set of firms in competition. They interact indirectly via the market. It is the result of an extensive research which was conducted on a representative sample of manufacturing firms. The database is collected by the French National Bank (*Banque de France*). Each firm aims to improve its performances. It observes the performances of the others in order to deduce their strategies. It chooses a suitable strategy according to its local context and the global one (its perception of other firms in the same market). In this model, a firm is mainly characterized by:

- X is a set of resources which are represented by an array of variables. They represent a firm's bundle of resources and competencies.

- Y is a set of performances which are represented by an array of variables. They depend on X.
- K is the amount of the available capital.
- B is the R&D budget. It corresponds to the flow of investment a firm disposes for improving its stock of resources and competencies (X variables).

A firm is also characterized by a set of strategies. A strategy defines the priorities of the resources.

The budget is thus allocated to the resources according to their priorities and a method of assignment. For instance, a strategy may affect the maximum of the budget to the resource with most priority over the others or only part of the budget (one unit of the resource).

3 Firms as Agents

In our multi-agent model, each firm is represented by an agent. This agent has two main behaviors: perception and deliberation.

3.1 The Firm Perception Behavior

The perception manages the interaction between the firm and its environment (the market and the other firms). It allows the firm to observe the evolution of the market and then to build a representation of the competition.

The observable data are the ones shown in the market such as the performances of the various firms. An elementary competition model can be therefore defined by the following data: the best, the average and the worst performances of the population of firms.

3.2 The Firm Deliberation Behavior

The deliberation behavior follows the following steps:

- Determination of relative performances.
- Calculation of the new capital and budget.
- The choice and application of a strategy i.e. modification of X.
- Calculation of the new performances.

After the update of the competition representation which is provided by the perception behavior, the deliberation process calculates the performance indicator V. The latter corresponds to a relative performance evaluation of a firm in relation to both its past and the market. It takes into account several factors: 1) the firm's profitability in comparison with the average value in the market; 2) the evolution of the firm's profitability from t-1 to t; 3) an index that gives a premium for the best market performer at time t; and 4) the evolution of the firm's market performance from t-1 to t.

V is used to update the internal parameters such as the capital and the budget of R&D.

$$K_{t+1} = h(k_t, V_{t+1})$$

$$B_{t+1} = g(V_{t+1}, K_{t+1}, r)$$

where r is the unspent budget of R&D resulting from the precedent period.

A decision process chooses then a strategy to update the resources (see following section). After their variation by the strategy, the resources X are transformed according to the Lisrel model to obtain performances Y. The Lisrel model corresponds to a statistical test on 50 French industries of the relationships between the resource variables and firm performances. For more detail of the Lisrel model, see (Durand and Guessoum, 1999).

Another characteristic of the firm activity is the exit rule. If a firm's capital decreases too much due to successive poor results, the firm may stop its activity. In our experiments, we set the following exit rule:

If ($K_t < 1/3 K_0$) **then** exit.

Where K_0 is the initial capital of the firm.

3.3 The Firm Decision Process

The selection of a strategy is the result of the decision process. Several mechanisms can be used to represent this decision process: inference engine, case based-reasoning, classifiers, etc. We distinguish two kinds of mechanisms: reactive mechanisms and adaptive mechanisms.

Reactive mechanisms are based on a fixed set of rules which are provided by the designer before run time.

Example of rule:

if ($V_t - V_{t-1} < 0$) **then** strategy_1

Adaptive mechanisms deal with the dynamic variations of the firm environment (market and other firms). Each firm builds a symbolic internal representation of its environment. This representation and the internal state are then used to determine the strategy. To build dynamically the decision rules, we used a classifier system. A classifier is defined by :

[context] [strategy] reward

The context integrates internal data (K, X, V,...) and data of the competition (Y_min, Y_average, Y_max).

Example:

```
Classifier_1:
[K = 500, V = 0.4
X = {2,1,5,3,4,8,6,2,1,1,3}
Y_Min {33,12, 45.11, 70.90}
Y_Max {36.2, 52.2, 76.1}
Y_average {34.1, 49.6, 75.7}]
[Strategy= strategy_1]
s= 0.5
```

The initial classifier list is empty. The basic execution cycle of our classifier system is (see Holland , 2000).

- Perception of the market and definition of the current context C.
- C is compared with the context parts of all classifiers of the current classifier list. The matching classifiers form a match set. If no rule matches C, create a new classifier with the context C. The choice of this classifier strategy is random.
- Select the best classifier Cl out of the matching set.
- Execute Cl's strategy.
- Reinforce Cl.

The matching function of two classifiers (with contexts Ci and Cj) may be defined as follows:

$$(w_0|K_i - K_j| + \sum_{k=1}^n \frac{w_k}{k} |X_{ik} - X_{jk}| + w_2|Y_{\min_i} - Y_{\min_j}| + w_3|Y_{\max_i} - Y_{\max_j}| + w_4|Y_{\text{average}_i} - Y_{\text{average}_j}|) < s$$

with $\sum_{i=1}^s w_i = 1$, $n = Nb \text{ resources}$, $s = 0.1$ is a threshold.

The reward is incremented (decremented) if the application of the strategy improves (decreases) the firm performances.

3.4 Experiments

To implement the proposed model, we used the platform DIMA (Z. Guessoum and J.-P. Briot, 1999). The latter provides a library of JAVA classes which have been reused easily to build the various agents.

For the validation of the implemented multi-agent system, we carried out several experiments with reactive and/or adaptive agents. We considered a population of 600 firms of which the half is adaptive and the other reactive. The two subpopulations have the same parameters at the beginning. Their difference resides only at the level of their decision-making processes.

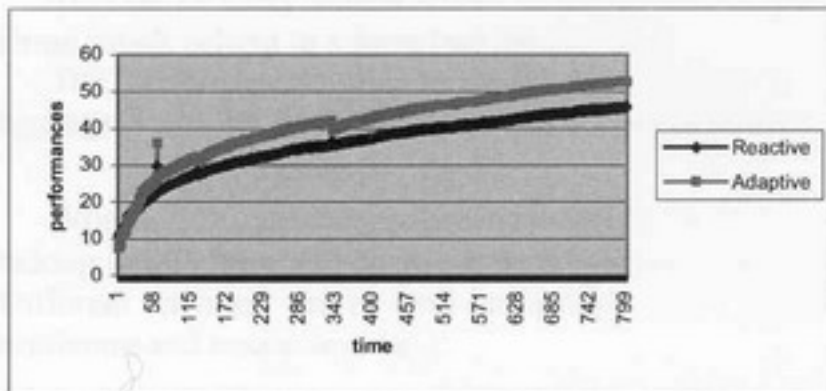


Figure 1. Comparison of reactive and adaptive

Figure 1 shows that adaptive firms are more powerful long (middle)-term than reactive firms. This is explained by the fact that they define their strategies according to more various criteria.

All the adaptive firms have a first difficult phase however. The initial classifier list is empty, strategies are chosen randomly. Their difficulties can be therefore explained by this random choice of strategies.

3.5 Discussion

During our experiments, we noticed that the death rate of the affected firms increases in an environment where the competition is strong. A firm leaves the market if it presents behaviors of overexploitation and surexploration. A firm can survive if 1) it consolidates its niches (resources and constraints), 2) it changes towards a new line of business, and 3) it is able to make competition in a turbulent environment and establish a balance between the capacities of exploration (research of new strategies) and of exploitation (reproduction of existing strategies) -see for more detail (Baum and Rao, 1999).

So, firms have to deal with the other firms but they have also to deal with the organizational forms. Thus, our initial model (see Sections 2) can be complemented in order to take into account the relation of the firms with the organizational forms.

4 Organizational Forms

Baum and Rao (1999) define organizational forms in term of their fundamental characteristics: goals, relation of authority, technology and strategy. A form gathers a set of firms which have these same fundamental characteristics and differ by secondary characteristics (a number and size). Hannan and Freeman (1989) consider that competition occurs between organizational forms rather than between firms.

An organizational form is an abstract entity that gathers a set of similar firms. Similar firms have similar behaviors and similar structures.

A good illustrative example of organizational forms is in the book market. The district bookstores, the FNAC or Virgin channels, and the completely online book delivery services as AMAZON are three examples of organizational forms. At the beginning, there were only some district bookstores and with the arrival of the FNAC, for example, this kind of stores (little stores, specific themes books) which didn't have the same structures nor strategies as the FNAC (big stores, different kinds of products: books, media products...) is disappearing. The last type of organizations is also different, they offer the same services but on the net.

4.1 Form Model

The proposed firm model (see Section 2) is based on the concept of strategy. Each strategy gives the variations of the firm resources. The variations define the differences and the similarities (in term of performances) of firms. Firms may indeed be gathered according to these variations of resources that are the results of the applied strategies. A form is thus defined by:

- the resource variations,

- and the performances of the associated firms.
- The resource variations are described as a granulation of the global variation interval $[0, 1]$. 0 corresponds to no resource variation and 1 corresponds to the allocation of the whole budget to the resource.

Different techniques can be therefore used to determine a granulation. We distinguish two main categories: crisp granulation and fuzzy granulation. Zadeh (2001) underlined that the fuzzy granulation is inspired by the human capability to operate on, and reason with, perception based-information. It provides thus a best representation of the various variation classes and a very good decision process.

Each variation of a resource can be thus described by the following symbolic values: small, medium or big. In our experiments, we use the same fuzzy granulation for the various resources (each element of X). This granulation is defined and updated by an expert. He/she can, for example, choose smaller fuzzy values (for example four instead of three values, see Figure 2).

Each form is then associated to a vector of fuzzy values. For example, if we consider three resources, the fuzzy values (*small*, *medium* and *big*) may correspond to a form that gives priority to the third resource.

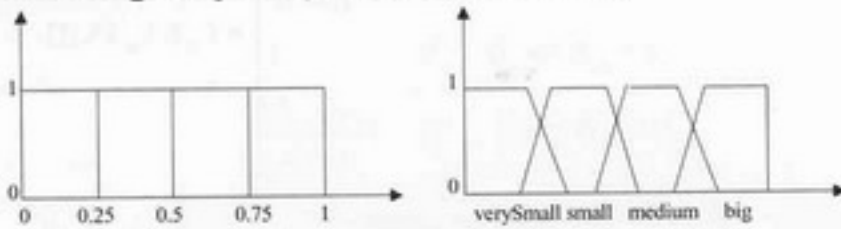


Figure 2: Example of granulation

Let's consider a set of K fuzzy values:

$FZ = (fv_1, fv_2, \dots, fv_K)$ and N resources. Each form fo_j may be therefore associated to a set of N fuzzy values:

$$(fv_{j1}, fv_{j2}, \dots, fv_{jn}) \text{ where } fv_{jp} \in FZ$$

This set of fuzzy values allows to determine the set of firms which belong to a form (see Section 4.3).

The performances of a form are then defined as an aggregation of the performances of the associated firms.

$$fo_j_y = \text{aggreg}(Y1, Y2, Ym)$$

Where $Y_{i, i=1..m}$ are the performances of the M firms that belong to this form and *aggreg* is an aggregation operator. Different operators can be used (arithmetic mean, median, minimum and maximum, etc.).

4.2 Integrations of Firms and organizational Forms

The relation between the two levels (micro and macro) is expressed in our model by the fact that the organization of firms produces indications (variations of resources) which express their internal strategy. On the other hand, the firm decision process relies on a representation of the population of the organizational forms and their evolution.

We thus introduce a new component which plays the role of a glue between the firms and the organizational forms. The architecture of our multi-agent system has therefore three components (see Figure 3):

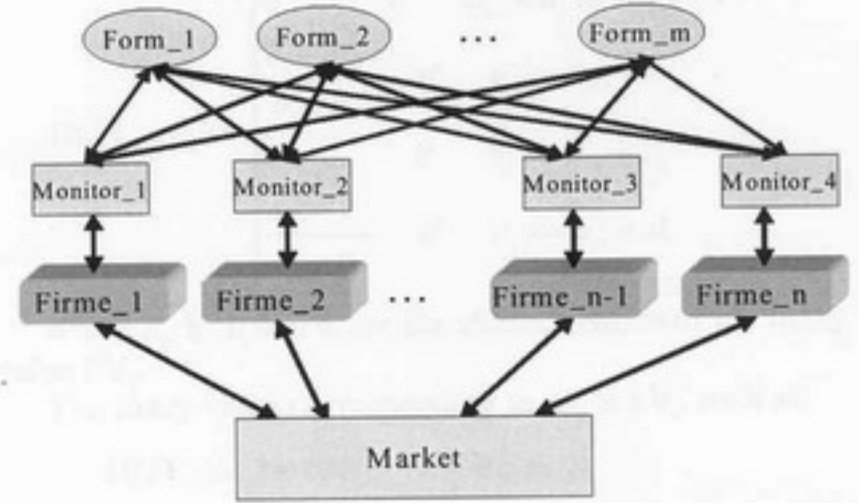


Figure 3. Firms and organizational forms.

- *Micro component (Firms)*: an organization of firms which have the capacity to perceive their environment (market, organizational forms) and act continuously according to their goals and the environment evolution.

- *Macro component (Organizational forms)*: a population of entities which represent the organizational forms. This macro level provides the system (mainly the population of firms) with a capacity of adaptation and a capacity of self-organization.

- *Micro-Macro component (Monitors)*: is represented by a set of agents (called monitors). The latter realize the coupling between the firms and the organizational forms. They observe the firms continuously, establish a set of variation and provide this information to the organizational forms on one hand. On the other hand, they perceive the population of organizational forms. The perceived data is then interpreted by the firms and exploited by their decision process. The monitors also control the organization of the firms and the population of organizational forms by adding new firms and new organizational forms, and removing organizational forms when needed.

The Micro-Macro interactions are described in more detail in the next section.

4.3 Interactions of Firms and Organizational forms

The interactions between firms and organizational forms are realized by an organization of *monitors*. The organization of *monitors* maintains the resource variations and the organizational forms with their fuzzy values and performances. Any modification of the firms is perceived by the *monitors*, this modification is then used to determine the associated organizational forms which adapt consequently. For instance, the evolution of the firm performances is used to update the performances of the

associated organizational form. On the other hand, any modification of an organizational form is perceived by the *monitors* and then propagated to the firm which may adapt consequently.

Moreover, *monitors* control the organizational forms. So, they have two main roles: Interaction Manager and Controller.

Interaction Manager. It manages the interaction between a firm and the associated organizational form. It observes the resource variations of the firm and determines the closest form by using the fuzzy values. It computes the respective grades of membership of the firm f_i to the various forms fo_j :

$$\Pi(f_i, fo_j) = \sum_{k=1}^N \Pi(FV_k / \Delta_{i,k}) / N$$

where:

FV_k is the fuzzy value representing the resource k for the form fo_j

$\Delta_{i,k}$ is the variation of the resource k for the firm f_i

$$\Pi(FV_k / \Delta_{i,k}) = \begin{cases} 0 & \text{if } \Delta_{i,k} < a_j \text{ or } \Delta_{i,k} \geq d_j \\ \frac{b_j - \Delta_{i,k}}{b_j - a_j} & \text{if } a_j \leq \Delta_{i,k} < b_j \\ 1 & \text{if } b_j \leq \Delta_{i,k} < c_j \\ \frac{\Delta_{i,k} - c_j}{d_j - c_j} & \text{if } c_j \leq \Delta_{i,k} < d_j \end{cases}$$

a_j, b_j, c_j et d_j are the characteristics of the fuzzy value FV_k (see Figure 4).

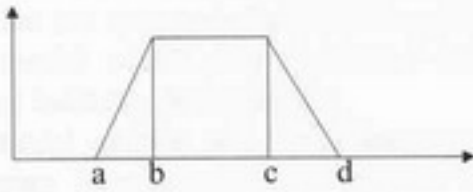


Figure 4: Characteristic of a Fuzzy Value

The firm f_i belongs then to the form fo_j such as:

$$\Pi(f_i, fo_j) = \max_{j=1;m} \Pi(f_i, fo_j)$$

and

$$\Pi(f_i, fo_j) > thGm$$

where $thGm$ is a threshold provided by the expert and m is the number of organizational forms.

Some firms can be therefore attached to no form. These firms are called orphans.

Controller. It controls the organizational forms. It adds new forms and removes existing ones when needed. It allows to create new forms when the performances of orphan firms are high. The best firm f_i is therefore used to create a new form. The fuzzy values of this new form are determined as follows:

For each resource k of the firm f_i and its variation $\Delta_{i,k}$, and for each fuzzy value $FV_j \in FZ$, it computes the grade of membership $\Delta_{i,k}$ to FV_j as follows:

$$\Pi(FV_j / \Delta_{i,k}) = \begin{cases} 0 & \text{if } \Delta_{i,k} < a_j \text{ or } \Delta_{i,k} \geq d_j \\ \frac{b_j - \Delta_{i,k}}{b_j - a_j} & \text{if } a_j \leq \Delta_{i,k} < b_j \\ 1 & \text{if } b_j \leq \Delta_{i,k} < c_j \\ \frac{\Delta_{i,k} - c_j}{d_j - c_j} & \text{if } c_j \leq \Delta_{i,k} < d_j \end{cases}$$

where a_j, b_j, c_j and d_j are the characteristics of the fuzzy value FV_j .

The fuzzy value corresponding to $\Delta_{i,k}$ is FV_j , such as:

$$\Pi(FV_j / \Delta_{i,k}) = \max_{j=1,K} (\Pi(FV_j / \Delta_{i,k}))$$

The set of $FV_{j=1,K}$ is used to define a new form. The *controller* computes the grades of membership of the other orphan firms to this new form and updates consequently the set of orphan firms and the new form.

New organizational forms can appear and existing ones can disappear. Organizational forms are constrained by their performances. So, when the performances of an organizational form decrease too much the *Controller* removes it.

4.4 Experiments

To deal with organizational forms, we extend the model of adaptive firms proposed in Section 3. We enriched the classifier context by data on the organizational forms.

Example:

```
Classifier_f:
[K = 500, V= 0.4
X = {2,1,5,3,4,8,6,2,1,1,3}
Y={35.25, 52.33, 75.12}
Y_Min (33.12, 45.11, 70.90)
Y_Max (36.2, 52.2, 76.1)
Y_average (34.1, 49.6, 75.7) }
Form = Form_2, BestForm = Form_4
WorstForm = Form_6]
[Strategy= strategy_1]
S= 0.7
```

For the validation of the adaptation of firms, we realized experiments with two kinds of adaptive firms (see Sections 3.3 and 4.3) named respectively Adaptive and AdapToForms. We considered a population of 900 firms: 300 reactive, 300 adaptive and 300 AdapToForms. Figure 5 shows that adaptive firms are more powerful when they deal with the whole context (Local context, competition and organizational forms). The difference between the two kinds of adaptive firms is significant (see series Adaptive and AdapToForms in Figure 5).

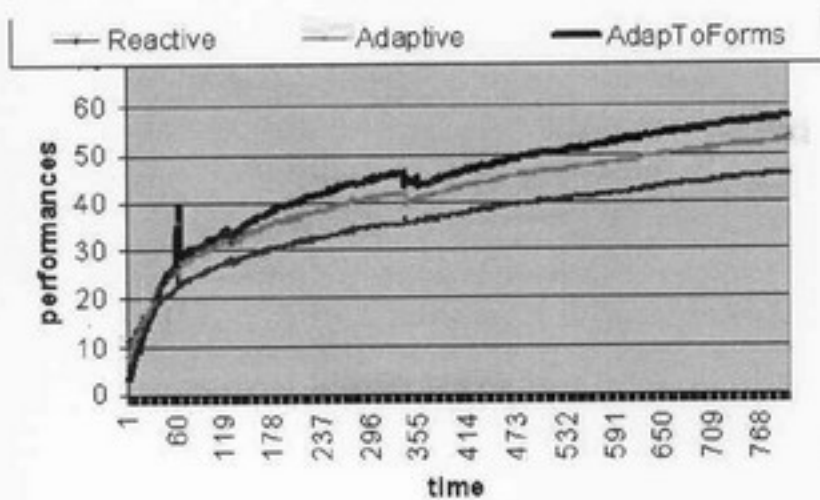


Figure 5: Comparison of reactive and adaptive firms

5 Conclusion

In this paper, an example of application of multi-agent systems to the simulation of economic models was presented. The proposed model of firms is very simple, but it reflects significant characteristics of the firms.

The paper illustrated first the interest of adaptive agents by comparing the performances of two populations of firms: reactive and adaptive. Then, it showed the limits of the economic models which represent a set of firms in indirect interaction via the market without taking into account the organizational forms. The economic model was then enriched by introducing organizational forms.

Experiments with models integrating firms and organizational forms have been presented and discussed. They show that the organizational forms are useful to have a complete model which provides novel insights on firm dynamics and industry demography.

The obtained results are interesting and promising. However, more experiments are needed to validate the proposed models and architecture.

References

- A. Joel A.C. Baum and Hayagreeva Rao. Evolutionary Dynamics of Organizational Populations and Communities. *Handbook of Organizational Change and Development*, Oxford University Press, 1999.
- Erhard Bruederer and Jitendra Singh. Organizational Evolution, Learning, and Selection: A genetic Algorithm-Based model. *Academy of management journal*, 39(5):1322-1349, 1996.
- Kathleen M. Carley. Adaptive Organizations and Emergent Forms. *IEEE* 769:2-3 1998.
- Giovanni Dosi and Richard R. Nelson. An Introduction to Evolutionary Theories in Economics. *Journal of Evolutionary Economics*, 4:153-172, 1994.
- Rodolphe Durand and Zahia Guessoum. Competence systemics and survival, simulation and empirical analysis. *Competence 2000 Conference*, Helsinki, 10-14, 2000.
- Zahia Guessoum and Jean-Pierre Briot. From active objects to autonomous agents, *IEEE Concurrency*, 7(3):68-76, 1999.
- John Holland, Lashon B. Booker, Marco Colombetti, Marco Dorigo, David E. Goldberg, Stephanie Forrest, Rick Riolo, Robert E. Smith, Pier Luca Lanzi, Wolfgang Soltzmann and Stewart W. Wilson. What Is a Learning Classifier System? *LCS'99, LNAI 1813*:3-32, 2000.
- Daniel A. Levinthal. Adaptation on Rugged Landscapes. *Management Science*, 43(7):934-950, 1997.
- Arie Y. Lewin and Henk W. Volberda. Prolegomena on Coevolution: A framework for Research on Strategy and New Organizational Forms. *Organization Science*, 10(5):519-534, 1999.
- Arie Y. Lewin, Chris P. Long and Timothy N. Carroll. The Coevolution of New Organizational Forms. *Organization Science*, 10(5): 535-550, 1999.
- Alessandro Lomi and Erik R. Larsen. Interacting Locally and Evolving Globally: A computational Approach to the Dynamics of Organizational Populations, *Academy of Management Journal*, 39(4), 1287-1321, 1996.
- Hideyuki Mizuta and Yoshiaki Yamagata. Agent-based Simulation for Economic and Environmental studies. *JSAI 2001 Workshops*, Springer-Verlag Berlin, *LNAI 2243*:142-152, 2001.
- Richard R. Nelson and Sidney G. Winter. An evolutionary theory of Economic Change. *Harvard University Press*, 1982.
- Edith T. Penrose. The theory of the growth of the firm. *Oxford Basil BlackWell*, 1959.
- Margaret A. Peteraf. The cornerstones of competitive advantage: a resource-based view. *Strategic Management Journal*, 14:179-191, 1993.
- Lotfi A. Zadeh. A new direction in AI: Toward a computational theory of perceptions. *AI Magazine* 22(1):73-84, 2001.