Sven A. Brueckner
Giovanna Di Marzo Serugendo
David Hales
Franco Zambonelli (Eds.)

LNAI 3910

Engineering Self-Organising Systems

Third International Workshop, ESOA 2005
Utrecht, The Netherlands, July 2005
Revised Selected Papers

Springer

Lecture Notes in Artificial Intelligence 3910

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Sven A. Brueckner
Giovanna Di Marzo Serugendo David Hales
Franco Zambonelli (Eds.)

Engineering Self-Organising Systems

Third International Workshop, ESOA 2005
Utrecht, The Netherlands, July 25, 2005
Revised Selected Papers

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Sven A. Brueckner
Altarum Institute
3520 Green Court, Suite 300, Ann Arbor, MI 48105-1579, USA
E-mail: sven.brueckner@altarum.org

Giovanna Di Marzo Serugendo
Birkback (University of London), Computer Science and Information Systems
Malet Street, London WC1E 7HX, UK
E-mail: dimarzo@dcs.bbk.ac.uk

David Hales
University of Bologna, Department of Computer Science
Mura Anteo Zamboni 7, 40127 Bologna, Italy
E-mail: hales@cs.unibo.it

Franco Zambonelli
Università di Modena e Reggio Emilia, Dipartimento di Scienze e Metodi dell'Ingegneria
Via Allegri 13, 42100 Reggio Emilia, Italy
E-mail: franco.zambonelli@unimore.it

Library of Congress Control Number: 2006923000

CR Subject Classification (1998): I.2.11, C.2.4, C.2, D.2.12, D.1.3, H.3, H.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-33342-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-33342-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11734697 06/3142 5 4 3 2 1 0

Preface

The idea that self-organisation and emergence can be harnessed for the purpose of solving tricky engineering problems is becoming increasingly accepted. Researchers working in many diverse fields (such as networks, distributed systems, operating systems and agent systems) are beginning to apply this new approach. This book contains recent work from a broad range of areas with the common theme of utilising self-organisation productively.

As distributed information infrastructures continue to spread (such as the Internet, wireless and mobile systems), new challenges have arisen demanding robust and scalable solutions. In these new challenging environments the designers and engineers of global applications and services can seldom rely on centralised control or management, high reliability of devices, or secure environments. At the other end of the scale, ad-hoc sensor networks and ubiquitous computing devices are making it possible to embed millions of smart computing agents into the local environment. Here too systems need to adapt to constant failures and replacement of agents and changes in the environment, without human intervention or centralised management.

Self-organising applications (SOAs) are able to dynamically change their functionality and structure without direct user intervention to meet changes in requirements and their environment. The overall functionality delivered by SOAs typically changes progressively, mainly in a non-linear fashion, until it reaches (emerges to) a state where it satisfies the current system requirements and therefore it is termed self-organising or emergent behaviour. Self-organising behaviour is often the result of the execution of a number of individual application components that locally interact with each other aiming to achieve their local goals, for example, systems that are based on agents or distributed objects. The main characteristic of such systems is their ability to achieve complex collective tasks with relatively simple individual behaviours, without central or hierarchical control.

However, in artificial systems, environmental pressures and local interactions and control may lead to unpredicted or undesirable behaviour. A major open issue is therefore how to engineer desirable emergent behaviour in SOAs and how to avoid undesirable ones given the requirements and the application environment. To address this issue, approaches originating from diverse areas such as non-linear optimisation, knowledge-based programming and constraint problem solving are currently been explored. Furthermore, SOA engineers often take inspiration from the real world, for example from biology, chemistry, sociology and the physical world. Typical examples of SOAs are systems that reproduce socially based insect behaviour, such as ants-based systems, artificial life, or robots. Although the results achieved so far are promising, further work is required until the problem is sufficiently addressed.

More specific fundamental questions that need an answer are: How do we structure the application components and their interactions, so that the self-organisation process results in the desired functionality? How do we validate that the application performs to the requirements within the range of scenarios expected during deployment? What means of influencing the dynamics of the application do we have available and how effective are they? On the one hand, multi-agent simulations and analytic modelling can be used to study emergent behaviour in real systems. On the other hand, results from complexity theory can be applied in engineering of both multi-agent systems and self-organising systems.

To address these issues the ESOA series of workshops was established. The aim is to open a dialog among practitioners from diverse fields, including: agent-based systems, software engineering, information systems, distributed systems, complex systems, optimisation theory and non-linear systems, neural networks, and evolutionary computation. Although backgrounds are diverse, the focus is always clear – to harness self-organising principles to solve difficult engineering problems.

This book includes revised and extended papers presented at the Third ESOA workshop held during the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) conference held in Utrecht, The Netherlands in July 2005. The workshop received 25 submissions, out of which 12 papers were selected for a long presentation and 6 papers for short presentation.

The first workshop (ESOA 2003) followed a theme of applying nature-inspired models to fields as diverse as network security, manufacturing control, and electronic markets. The second workshop (ESOA 2004) included papers on self-assembly of software, robots task allocations, design methods, and stigmergy-based applications. Both workshops were held during the AAMAS conferences in 2003 and 2004 respectively and post-proceedings are published by Springer, (volumes LNAI 2977 and 3464).

ESOA 2005 included a number of papers related to methodologies and engineering practices. This shows that research in the field of self-organising applications is maturing from novel techniques that work in specific contexts to more general engineering proposals. This book is structured into three parts reflecting the workshop session themes.

Part I presents novel self-organising mechanisms. Jelasity et al. present a self-organising mechanism for maintaining and controlling topology in overlay networks based on gossiping. Georgé et al. describe "emergent programming" through self-organisation of a program's instructions. Picard et al. show how cooperation among agents serves as a self-organisation mechanism in the framework of a distributed timetabling problem. Nowostaswski et al. present the concept of "evolvable virtual machines" architecture for independent programs to evolve into higher levels of hierarchical complexity; Hales presents a P2P re-wiring protocol that allows peers with different skills to spontaneously self-organise into cooperative groups. Dimuro et al. present a self-regulation algorithm for multi-agent systems based on a sociological model of social exchanges.

Armetta et al. discuss a protocol for sharing critical resources based on a two-level self-organised coordination schema.

In Part II methodologies, models and tools for self-organising applications are presented. Brueckner et al. present an agent-based graph colouring model favouring distributed coordination among agents with limited resources in a real-world environment. Marrow et al. describe applications using self-organisation based upon the DIET multi-agent platform. Saenchai et al. present a multi-agent-based algorithm solving the dynamic distributed constraint satisfaction problem. De Wolf et al. present an approach combining simulation and numerical analysis for engineering self-organising systems with some guaranteed macroscopic behaviour. Gardelli et al. discuss self-organising security mechanisms based on the human immune system, and their verification through simulation. Renz et al. discuss the need of using mesoscopic modeling to provide descriptions of emergent behaviour.

Part III presents specific applications of self-organising mechanisms. Ando et al. apply the stigmergy paradigm to automated road traffic management. Fabregas et al. discuss a model inspired from bee behaviour and apply this model to an example of cultural heritage. Van Parunak et al. discuss a sift and sort algorithm for information processing inspired by ants sorting and foraging. Tatara et al. present an agent-based adaptive control approach where local control objectives can be changed in order to obtain global control objectives. Hadeli et al. discuss measures of reactivity of agents in a multi-agent and control approach based on stigmergy.

Finally, we wish to thank all members of the Programme Committee for returning their reviews on time (all papers submitted to the workshop were reviewed by two to three members of the Programme Committee) and for offering useful suggestions on improving the workshop event. Also we thank all those who attended the workshop and contributed to the lively discussions and question and answer sessions.

January 2006

Sven Brueckner
Giovanna Di Marzo Serugendo
David Hales
Franco Zambonelli
Organising Committee
ESOA 2005

Organization

Programme Committee

Yaneer Bar-Yam, New England Complex Systems Institute, USA
Sergio Camorlinga, University of Manitoba, Canada
Vincent Cicirello, Drexel University, USA
Marco Dorigo, IRIDIA, Université Libre de Bruxelles, Belgium
Noria Foukia, University of Southern California, USA
Maria Gini, University of Minnesota, USA
Marie-Pierre Gleizes, IRIT Toulouse, France
Salima Hassas, University of Lyon, France
Manfred Hauswirth, Swiss Federal Institute of Technology, Switzerland
Mark Jelasity, University of Bologna, Italy
Margaret Jefferies, The University of Waikato, New Zealand
Manolis Koubarakis, Technical University of Crete, Greece
Mark Klein, MIT Sloan School of Management, USA
Ghita Kouadri Mostefaoui, University of Fribourg, Switzerland
Soraya Kouadri Mostefaoui, University of Fribourg, Switzerland
Marco Mamei, University of Modena and Reggio Emilia, Italy
Paul Marrow, BT, UK
Philippe Massonet, CETIC, Belgium
Alberto Montresor, University of Bologna, Italy
Andrea Omicini, University of Bologna, Italy
Daniel Polani, University of Hertfordshire, UK
Martin Purvis, University of Otago, New Zealand
Mikhail Smirnov, Fraunhofer Fokus, Berlin, Germany
Paul Valckenaers, Katholieke Universiteit Leuven, Belgium

Table of Contents

Part I: Self-organising Mechanisms

Part II: Methodologies, Models and Tools

T-Man: Gossip-Based Overlay Topology Management*

Márk Jelasity** and Ozalp Babaoglu

University of Bologna,
Dipartimento di Scienze dell'Informazione,
Mura Anteo Zamboni 7, 40126 Bologna, Italy
{jelasity, babaoglu}@cs.unibo.it

Abstract. Overlay topology plays an important role in P2P systems. Topology serves as a basis for achieving functions such as routing, searching and information dissemination, and it has a major impact on their efficiency, cost and robustness. Furthermore, the solution to problems such as sorting and clustering of nodes can also be interpreted as a topology. In this paper we propose a generic protocol, T-Man, for constructing and maintaining a large class of topologies. In the proposed framework, a topology is defined with the help of a *ranking function*. The nodes participating in the protocol can use this ranking function to order any set of other nodes according to preference for choosing them as a neighbor. This simple abstraction makes it possible to control the self-organization process of topologies in a straightforward, intuitive and flexible manner. At the same time, the T-Man protocol involves only local communication to increase the quality of the current set of neighbors of each node. We show that this bottom-up approach results in fast convergence and high robustness in dynamic environments. The protocol can be applied as a standalone solution as well as a component for recovery or bootstrapping of other protocols.

1 Introduction

In large, dynamic, fully distributed systems, such as peer-to-peer (P2P) networks, nodes (peers) must be organized in a connected network to be able to communicate with each other and to implement functions and services. The neighbors of the nodes—the "who is connected to whom", or "who knows whom" relation—define the *overlay topology* of the distributed system in question. This topology can dynamically change in time, and in every time point, it defines the possible interactions between the nodes.

Although it would be desirable, it is typically very difficult to ensure that all nodes are aware of every other participating node in the system. The reason is

* This work was partially supported by the Future and Emerging Technologies unit of the European Commission through Project BISON (IST-2001-38923) and DELIS (IST-2002-001907).

** Also with MTA RGAI, SZTE, Szeged, Hungary.

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 1–15, 2006.

that the set of participating nodes changes quickly, and (due to the large number of nodes) it is not feasible to maintain a complete list of the nodes. This means that all nodes are aware of only a limited subset of other nodes, so efficient and robust algorithms are necessary to create, maintain and optimize the topology.

Overlay topology forms the basis for, or has a major impact on many functions. It is well known that functions such as searching, routing, information dissemination, data aggregation, etc, need special topologies for good performance and high efficiency. Furthermore, solutions to other problems including sorting and clustering can be readily expressed as topologies. For example, in the case of sorting, we are looking for a linear structure that represents some total ordering relation. For all these functions, numerous topologies have been suggested and even more protocols to construct and repair them have been proposed.

Motivated by these observations, we consider topology management as a general purpose function that is desirable in distributed systems. In this paper we specifically target very large scale and highly dynamic systems. Key requirements of topology management in such environments include robustness, scalability, flexibility and simplicity. Besides, it is a great advantage if a topology manager is flexible enough to allow for changing the managed topology at run time *on demand*, without having to develop a new protocol for each possible topology from scratch. Since topology is a very general abstraction, that can be used to express solutions to problems and to enhance and support other functions, such functionality would allow us to increase the efficiency of deploying fully distributed application dramatically. We would need only one running topology component and the application area of the system could be changed at run time whenever necessary. With a protocol that supports quickly changing topologies, it even becomes possible to automatically *evolve* topologies through, for example, an evolutionary process.

In this paper we propose a generic protocol, T-Man, with the aim of fulfilling the requirements outlined above. The desired topology is described using a single ranking function that all nodes can apply to order any subset of potential neighbors according to preference for actually being selected as a neighbor. Using only local gossip messages, T-Man gradually evolves the current topology towards the desired target structure with the help of the ranking function. We show experimentally that the protocol is scalable and fast, with convergence times that grow only as the logarithm of the network size. These properties allow T-Man to be practical even when several different topologies have to be created on demand, and also in dynamic systems where the set of nodes or their properties change rapidly. Additionally, the general formulation of the ranking function allows us to deal with a wide range of different topologies.

Although this work is concerned mainly with exploring the basic properties of T-Man by examining simple topologies like ring, mesh and binary tree, it is possible to illustrate its practicality with more realistic applications. We briefly outline three such applications: sorting, clustering and a distributed hash table (DHT).

Related work includes gossip-based protocols, that have gained notable popularity in various contexts [1, 2, 14]. In this paper we suggest a novel application

of the gossip communication model to solve the topology management problem. Issues related to topology management itself have also received considerable attention. Examples from the vast literature include DHTs [7, 11, 13], unstructured overlays [9,3], and superpeer topologies [16]. As for topology construction, Massoulié and Kermarrec [6] propose a protocol to evolve a topology that reflects proximity, Voulgaris and van Steen [15] propose a method to jump-start Pastry. Unlike these specific solutions, T-Man is a generic framework and can be used to construct and maintain a large class of different topologies quickly in a simple and scalable manner.

2 The Problem

We assume that we are given a (perhaps random) overlay network, and we are interested in constructing some desirable topology by connecting all nodes in the network to the right neighbors. The topology can be defined in many different ways and it will typically depend on some properties of the nodes like geographical location, semantic description of stored content, storage capacity, etc. We need a formal framework that is simple yet powerful enough to be able to capture most of the interesting structures. Our proposal is the *ranking function* that defines the target topology through allowing all nodes to sort any subset of nodes (potential neighbors) according to preference to be selected as their neighbor.

For a more formal definition, let us first define some basic concepts. We consider a set of nodes connected through a routed network. Each node has an address that is necessary and sufficient for sending it a message. Nodes maintain addresses of other nodes through *partial views* (*views* for short), which are sets of c *node descriptors*. In addition to an address, a node descriptor contains a *profile*, which contains those properties of the nodes that are relevant for defining the topology, such as ID, geographical location, etc. The addresses contained in views at nodes define the links of the *overlay network topology*, or simply the *topology*. Note that parameter c defines the node degree of the overlay network and is uniform for all nodes.

We can now define the *topology construction problem*. The input of the problem is a set of N nodes, the view size c and a *ranking function* R that can order a list of nodes according to preference from a given node. The ranking function R takes as parameters a base node x and a set of nodes $\{y_1, \ldots, y_m\}$ and outputs a set of orderings of these m nodes. The task is to construct the views of the nodes such that the view of node x, denoted view_x, contains exactly the first c elements of a "good" ranking of the entire node set, that is, $R(x, \{\text{all nodes except } x\})$ contains a ranking that starts with the elements of view_x. We will call this topology the *target topology*.

In the presence of churn (ie, when nodes constantly join and leave the overlay network) we talk about maintenance of the target topology instead of construction. Instead of a formal definition, we define the problem as staying "as close as possible" to the target topology. The actual figures of merit to characterize maintenance can be largely application dependent in this case.

One (but not the only) way of obtaining ranking functions is through a distance function that defines a metric space over the set of nodes. The ranking function can simply order the given set according to increasing distance from the base node. Let us define some example distance-based topologies of different characteristics. From now on, to simplify our language and notation, we use the nodes and their profiles interchangeably.

Line and ring. The profile of a node is a real number. The distance function for the line is $d(a,b) = |a-b|$. In the case of a ring, profiles are from an interval $[0,N]$ and distance is defined by $d(a,b) = \min(N-|a-b|, |a-b|)$ Ranking is defined through this distance function as described above.

Mesh, tube and torus. The 1-dimensional topology defined above can be easily generalized to arbitrary dimensions to get for example a mesh or a torus. The profiles are two-dimensional real vectors. The distance for the mesh is the Manhattan distance. It is given by calculating the 1-dimensional distance described above along the two coordinates and returning the sum of these distances. Applying the periodic boundary condition (as for the ring) results in a tube for one coordinate and a three dimensional torus for both coordinates.

Binary tree. A low diameter topology can be constructed from a binary tree: the profiles are binary strings of length m, excluding the all zero string. Distance is defined as the shortest path length between the two nodes in the following undirected rooted binary tree. The string $0 \ldots 01$ is the root. Any string $0a_2 \ldots a_m$ has two children $a_2 \ldots a_m 0$ and $a_2 \ldots a_m 1$. Strings starting with 1 are leafs. This topology is of interest because (unlike the previous ones) it has a very short (logarithmic) diameter of $2m$.

There are very important ranking functions that cannot be defined by a global distance function, therefore the ranking function is a more general concept than distance. The ranking functions that define sorting or proximity topologies belong to this category. Examples will be given in Section 6.1.

3 The Proposed Solution

The topology construction problem becomes interesting when c is small and the number of nodes is very large. Randomized, gossip-based approaches in similar settings, but for other problem domains like information dissemination or data aggregation, have proven to be successful [2, 4]. Our solution to topology construction is also based on a gossip communication scheme.

3.1 The Protocol

Each node executes the same protocol shown in Figure 1. The protocol consists of two threads: an active thread initiating communication with other nodes, and a passive thread waiting for incoming messages.

Each nodes maintains a view. The view is a set of node descriptors. A call to MERGE(view_1,view_2) returns the union of view_1 and view_2.

```
do at a random time once in each
consecutive interval of T time units
  p ← selectPeer()
  myDescriptor ← (myAddress,myProfile)
  buffer ← merge(view,{myDescriptor})
  buffer ← merge(buffer,rnd.view)
  send buffer to p
  receive buffer_p from p
  buffer ← merge(buffer_p,view)
  view ← selectView(buffer)
```

(a) active thread

```
do forever
  receive buffer_q from q
  myDescriptor ← (myAddress,myprofile)
  buffer ← merge(view,{myDescriptor})
  buffer ← merge(buffer,rnd.view)
  send buffer to q
  buffer ← merge(buffer_q,view)
  view ← selectView(buffer)
```

(b) passive thread

Fig. 1. The T-MAN protocol

The two key methods are SELECTPEER and SELECTVIEW. Method SELECTPEER uses the current view to return an address. First, it applies the ranking function to order the elements in the view. Next, it returns the first descriptor (according to this ordering) that belongs to a live node. Method SELECTVIEW(BUFFER) also applies the ranking function to order the elements in the buffer. Subsequently, it returns the *first c elements* of the buffer according to ranking order.

The underlying idea is that in this manner nodes improve their views using the views of their current neighbors, so that their new neighbors will be "closer" according to the target topology. Since all nodes do the same concurrently, neighbors in the subsequent topologies will be gradually closer and closer. This also means that the views of the neighbors will keep serving as a useful source of additional, even better links for the next iteration.

Last but not least, we need to explain the origin and role of the buffer RND.VIEW. This buffer contains a random sample of the nodes from the entire network. It is provided by a *peer sampling service* [3]. The peer sampling service described in [3] is implemented in a very similar fashion: nodes periodically exchange their random views and update their local views thereby creating a new random sample. These random views define an approximately random overlay network. The buffer RND.VIEW is the current set of neighbors in this random overlay network. The peer sampling service is extremely robust to failure and maintains a connected network with a very high probability.

The role of the random buffer is most important in large diameter topologies. In this case, if a node has a low quality neighbor set and if most of the rest of the nodes have a high quality neighbor set (forming a large diameter topology, e.g., a ring), then this node needs to perform many exchanges until it can reach the optimal set of neighbors, because the speed of "finding its neighborhood" is related to the diameter of the topology. The random buffer adds long range links that help speeding up convergence.

Although the protocol is not synchronous, it is often convenient to refer to *cycles* of the protocol. We define a cycle to be a time interval of $T/2$ time units where T is the parameter of the protocol in Figure 1. Note that during a cycle, each node is updated once on the average.

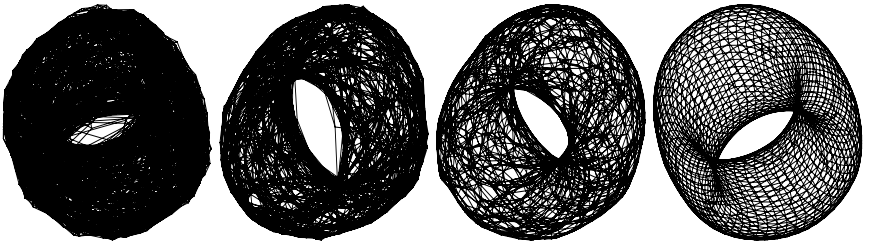


Fig. 2. Illustrative example of constructing a torus over $50 \times 50 = 2500$ nodes, starting from a uniform random topology with $c = 20$. For clarity, only the nearest 4 neighbors (out of 20) of each node are displayed.

Figure 2 illustrates the results of the protocol when used to construct a small torus (visualizations were done using [5]). For this example, it is clear that 15 cycles are sufficient for convergence, and the target topology is already evident even after very few cycles. As we will see, T-Man proves to be extremely scalable and the time complexity of the protocol remains in this order of magnitude even for a million nodes.

3.2 Optimizations

We can increase the performance of the protocol by applying two well known techniques described in [1]. First, we set a connection limit of 1, that is, in each interval of T time units (i.e., two cycles), we allow each node to receive at most one connection. Since each node also initiates one connection in this interval, this means that during two cycles, each node communicates at most twice. Second, we also apply *hunting*, that is, instead of trying only one peer, each node actively keeps looking for nodes (from the current view) that have not exceeded their connection limit in the given interval. Our preliminary experiments indicate that these techniques noticeably improve the convergence speed of T-Man.

As another possibility for optimization, note that a node does not need to send the entire buffer containing the union of the fresh descriptor, the old view and the random buffer. In fact, the contacted node will use at most the c closest items from this buffer so it is sufficient to first sort the buffer applying the ranking function *of the peer*, and sending the first c items. Since all nodes use the same ranking function, they can easily emulate the ordering from the point of view of any other node.

4 Simulation Experiments

All the simulation results presented in this paper were produced using PeerSim, an open-source simulator developed at the University of Bologna [10].

We examine the three distance-based ranking functions that define the ring, torus and binary tree topologies, as defined in Section 2. The motivation of this

choice is that the ring is a large diameter topology and it is relevant for the sorting application (Section 6.1), the binary tree is of a logarithmic diameter and the torus is relevant in proximity problems being based on a 2-dimensional grid. The network sizes (N) examined are 2^{14}, 2^{17} and 2^{20}. We initialize the profiles of the nodes in a regular manner, that is, in the case of the ring topology, we assign the numbers $1, 2, \ldots, N$ to the nodes, and likewise for the torus $((1,1),(1,2),\ldots,(\sqrt{N},\sqrt{N}))$ and the binary tree (all binary strings of length $\log_2 N$).

This regularity is not critical for the success of the protocol. On the contrary, one of the important applications is sorting an arbitrary set of numbers, as we argue in Section 6.1. However, this controlled setting allows us to monitor the dynamics of the protocol in a more informed manner as the distance function becomes equivalent to the hop count in the *target topology* defined by the links that connect nodes at distance 1 (the target links). During the experiments we focus on the dynamics of the number of target links that are found. As a measure of performance, the *convergence factor* is defined as the factor by which the number of target links found increases from one cycle to the next. Note that a constant convergence factor means exponential increase.

The NEWSCAST protocol was used as the implementation of the peer sampling service [3], which works very similarly to T-MAN maintaining a dynamic random overlay and using it to provide random peers. The NEWSCAST protocol is extremely scalable and robust, and its communication cost is similar to that of T-MAN. The cache size of NEWSCAST was 30 and its cycle length was identical to that of T-MAN. In this section, we focus on convergence starting from a random network, that is, the views are initialized at random and the nodes start to run the protocol at the same time. In Section 5 we examine the effect of churn, that is, with nodes continuously joining and leaving the network.

The results are shown in Figure 3. The results clearly indicate a logarithmic relationship between network size and convergence speed. This is illustrated especially well by the plots comparing the convergence factor for different network sizes as a function of time. We can see a constant shift of convergence time when the network size is increased by the same multiplicative factor (2^3). Quite interestingly, initial convergence does not depend on the view size c, nor does it depend on the characteristics of the target topology.

When the topology has already converged, the few nodes that are still incorrectly linked can be though of as "climbing" on the converged structure during the consecutive cycles of T-MAN. This means that in this end phase convergence time does depend on the target topology. Accordingly, in the binary tree topology, we observe rapid convergence. In fact, logarithmic convergence, because the evolved structure allows for efficient routing, being low diameter. Similar arguments hold for the torus, only the convergence time there is not logarithmic but grows with the square root of the network size in the worst case. In both cases, we can observe fast convergence even for the smallest view size.

The case of the ring is different, because the target topology has a large diameter that grows linearly with the network size, so the remaining few misplaced

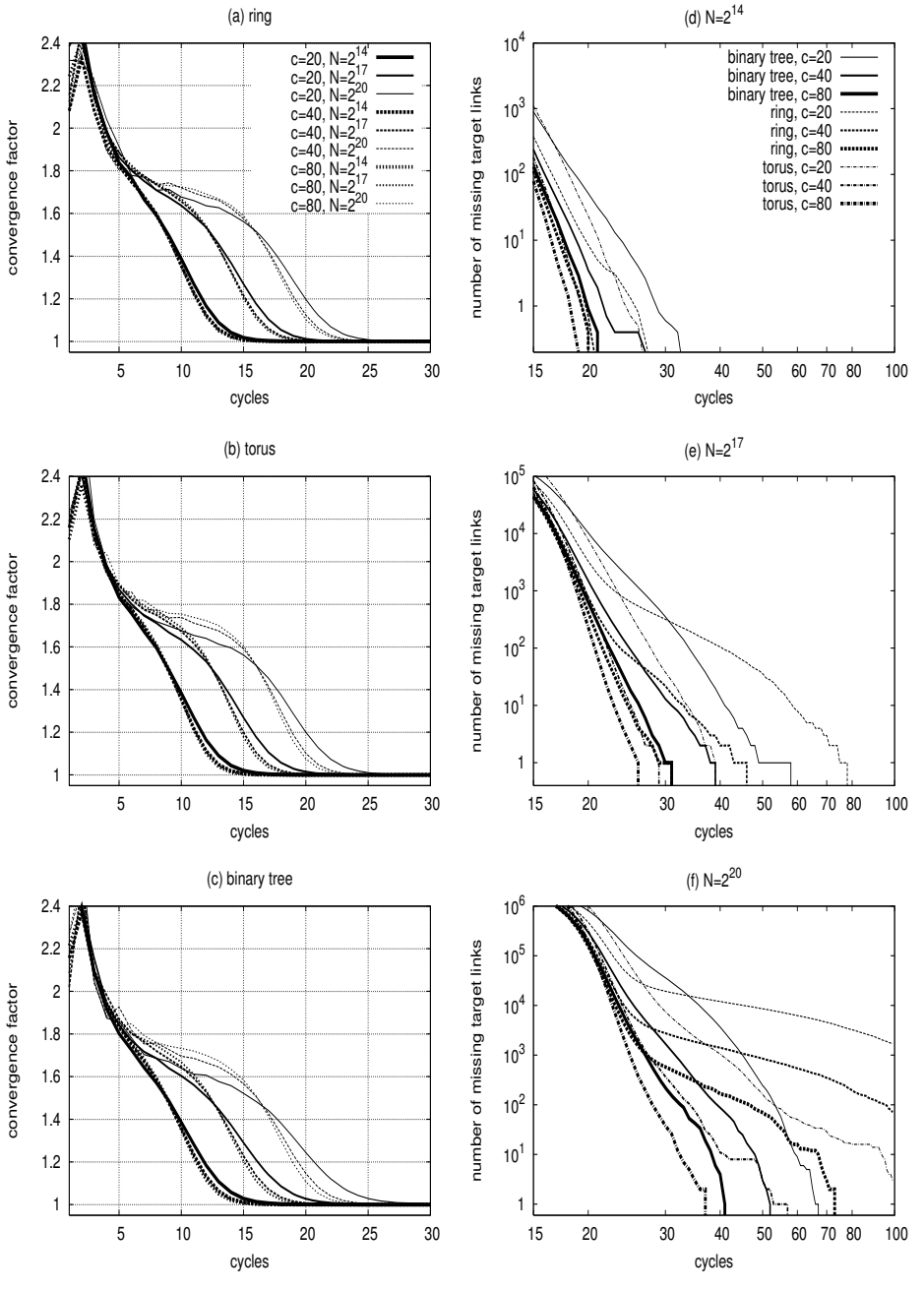


Fig. 3. Comparison of convergence speed in the initial phase and in the final phase for network sizes $N = 2^{14}, 2^{17}, 2^{20}$ and $c = 20, 40, 80$ for the ring, torus and binary tree topologies. The results displayed are averages of 10 runs for $N = 2^{14}$ and $N = 2^{17}$, and show a single run for the case $N = 2^{20}$.

nodes reach their destination slower. Still, for $c = 80$ we have *perfect* convergence after cycle 72 even for $N = 2^{20}$, and only a small percentage of target links are missing with $c = 20$ and $c = 40$. For the smaller network sizes we always observe full convergence in less than 80 cycles, independently of the characteristics of the target topology.

5 Self-healing

In this section we consider scenarios with churn, that is, with nodes constantly leaving and joining the network. We introduce a simple extension of the protocol to increase its adaptivity and subsequently we experimentally evaluate the proposed solution.

5.1 Age-Based View Update

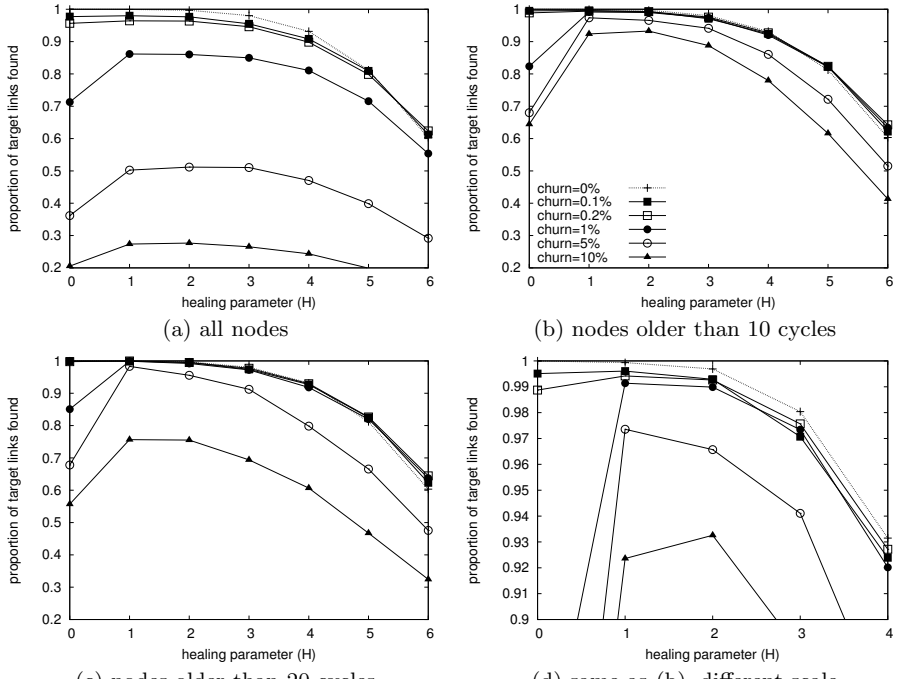
We extend the protocol given in Figure 1 by a simple technique to handle dynamic environments. The key idea is that we remove a few old descriptors from the view in each cycle. As a result, we expect to decrease the number of "dead links", that is, descriptors describing nodes that are no longer in the network. By decreasing the number of dead links, we expect to increase the quality of the views of the live nodes.

To implement this idea, the node descriptors stored in the view must also contain an age field. This field is initialized to be zero (when the node adds its own descriptor to the buffer to send) and increased for all view entries every time the node communicates. Before merging the view to the buffer to be sent, each node removes the H oldest descriptors from the view. Finally, the MERGE operation has to be modified to prefer the freshest item when removing duplicate items.

5.2 Experimental Results

To test the efficiency of this solution, we performed experiments with different scenarios involving churn. In all experiments, network size was 10^4, and $c = 20$. The cache size of NEWSCAST (the applied peer sampling service) was 30 and its cycle length was identical to that of T-MAN. Churn was modeled by removing a given percentage of randomly selected nodes from the network in each cycle and replacing them with new nodes that were initialized with random links. The ranking function defined a 1-dimensional ring. However, due to churn, node profiles were initialized by a random 62 bit integer, not regularly as in Section 4. For this reason, to define a connected ring, we applied the direction dependent version of the ranking function as described in Section 6.1.

The results of the experiments, illustrating various settings for the healing parameter H and churn rates, are shown in Figure 4. First of all, note that the churn rates can be considered very high. If we set the cycle length parameter $T/2 = 10s$, then, based on the Gnutella trace described in [12], the churn rate is less than 0.2% per cycle. In this light, 5% or especially 10% churn is extremely high.



(a) all nodes (b) nodes older than 10 cycles

(c) nodes older than 20 cycles (d) same as (b), different scale

Fig. 4. Experimental results in the presence of churn. $N = 10,000$, $c = 20$.

Each point in the plots was generated by running the respective scenario until 300 cycles, to make sure that the proportion of correctly identified target links converges, and taking this converged value. The first observation is that over all the nodes, high churn rates decrease the overall quality of the views in the network quite significantly. However, for such high churn rates, the network always contains many newcomers. If we consider the convergence times presented in Section 4, we can note that for a newcomer at least 10 cycles are necessary to optimize its view, so we cannot reasonably expect better performance according to this measurement. However, if we restrict ourselves to nodes that are old enough, we get a very different picture. For $H = 1$ and $H = 2$, we observe a very good quality network even for churn rates of 10% which is especially notable because the expected lifetime of a node in this network is only 10 cycles. In fact, the number of nodes older than 10 cycles is around 3500, one third of the network.

We can also observe that too aggressive "healing" in fact damages the network even when there is no churn. The setting $H = 6$ is consistently worse than any other setting. However, the positive effect of self-healing can be observed when comparing the case of $H = 1$ with $H = 0$ (no healing). This consistently results in a significant performance improvement. In general, $H = 1$ appears to be the best choice, except in the most extreme churn where $H = 2$ is slightly better.

As a final note, it is interesting to observe that performance is in fact no so bad even without the application of the healing method ($H = 0$). This is due to the fact, that in our scenarios the overall number of dead links is guaranteed not to decrease below 50%. To see this, consider the case when the proportion of dead links is p in the network and we remove n nodes, replacing them by n new nodes, that have links to random *live* nodes. Due to the removal, the number of dead links on average decreases by ncp while it increases by the number of links that pointed to these nodes: on average $nc(1-p)$. if we assume that all nodes in the network have the same in-degree (it is true for our ranking function here). This dynamics always converges to $p = 0.5$. This fact emphasizes the importance of the bootstrapping method, especially in the presence of extreme churn.

6 Application Examples

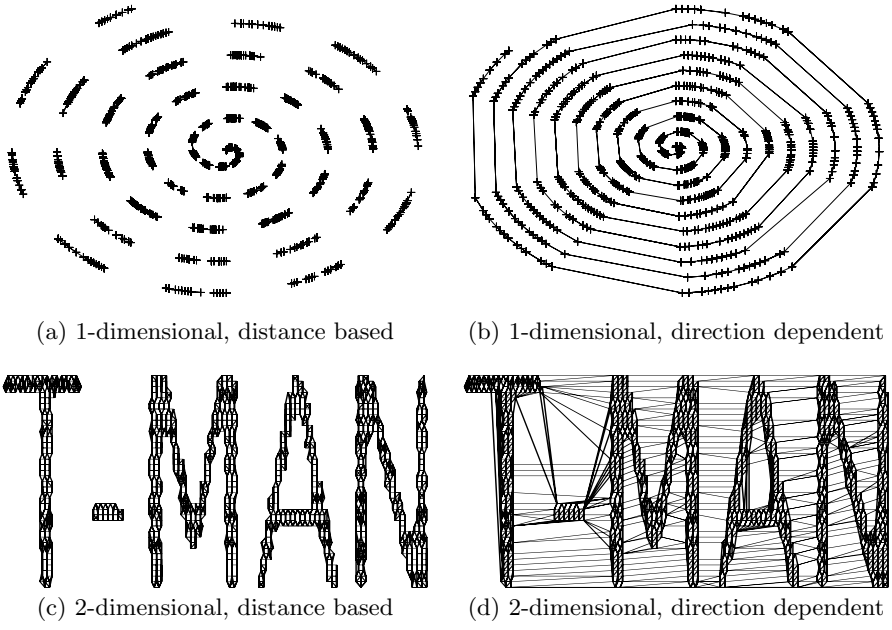
The primary goal of this section is to underline the generality of the approach by outlining the main ideas in using T-Man to solve some potentially important applications.

6.1 Clustering and Sorting

So far we have considered rather artificial settings to understand the behavior of T-Man better. In particular, the profiles of the nodes were initialized in a regular manner. In practice this will hardly happen. Typically, the properties of the nodes follow some specific distribution. This distribution can also be "patchy", there can be dense clusters separated by unrepresented regions in the profile space. Very informally, when applying T-Man in such a setting using a simple distance-based ranking function, the resulting topology will be clustered and most likely disconnected, because nodes in the same cluster will eventually be linked to each other only. An illustrative example is given in Figure 5 for the 1- and 2-dimensional cases. In many applications, like clustering based on semantic, geographic or any other definition of proximity, this property can be exploited to find the desired clusters.

In the case of the sorting problem, where we would like to connect each node to those nodes that directly preceed and follow them according to some total ordering relation, we need to prevent clustering. This can be achieved by the following direction dependent ranking. First, separate the set of nodes to be ranked into two groups: one that is to the left, and another that is to the right of the base node. Order these two sets according to the underlying desired ordering. Merge the ordered sets so that a node that had index i in any of the sets is assigned index $2i$ or $2i + 1$ in the final ranking, choosing randomly between these two possibilities. Applying the 1-dimensional ranking function makes it possible to practically reduce the sorting problem to the one dimensional topology construction problem that we have studied extensively in Section 4. In Section 5 we used exactly this sorting method as a ranking function.

Direction dependent ranking can be easily extended to other problems, for example, creating a *connected* topology in two dimensions that reflects geo-



(a) 1-dimensional, distance based

(b) 1-dimensional, direction dependent

(c) 2-dimensional, distance based

(d) 2-dimensional, direction dependent

Fig. 5. Illustrative example of converged topologies obtained with distance-based and direction dependent ranking, with $N = 1000$, $c = 20$. The line is displayed as spiral for convenience. Only the closest 2 and 4 links are shown from each node for the 1- and 2-dimensional example, respectively.

graphical proximity. In this case, a node divides the space into four quarters, and classifies each node to be ranked into four categories accordingly. The node then sorts the nodes in each class according to an underlying distance function, and produces the ranking similarly to the two dimensional case: if a node has index i in any of the four quarters, then it will be assigned an index randomly from between $4i$ and $4i + 3$.

The effect of direction dependent ranking is illustrated by two small examples in Figure 5. In the case of both the distance based and direction dependent ranking the nodes are mapped to points forming the plotted structures: equal length intervals in 1-dimension and letter-shaped clusters in 2-dimensions. The profile of the nodes is defined as their 1- or 2-dimensional coordinates, respectively. Observe the clustering effect with the distance based ranking and, with direction dependent ranking, the perfect sorting in 1-dimension and the connected topology in 2-dimensions.

6.2 A DHT

As an illustration, we very briefly present a simplistic way of evolving a distributed hashtable (DHT) topology with T-Man. The ranking function for the target topology is defined by a XOR-based distance. The distance we use is not

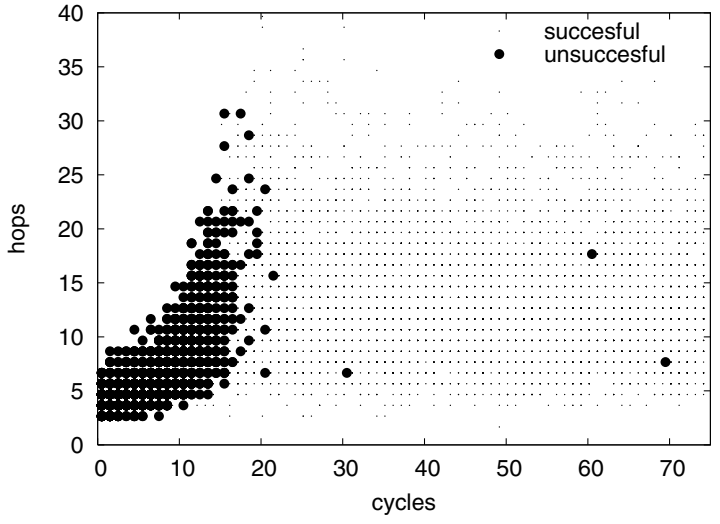


Fig. 6. Hop count and success of routing as a function of time (cycles). Each point represents the result of the routing algorithm from a random node to another random node. In each cycle 100 experiments are shown, with a small random translation. Node ID-s are random 62 bit integers, network size is $2^{20} (> 10^6)$, size of routing table is 60: 30 from the Pastry-inspired topology, 30 from the ring.

that of [7]. Instead, we define the XOR distance over a set of binary numbers as the number of bit positions in which they have a different digit. This ranking function is responsible for evolving long range links. The idea is that in this topology each node should know about nodes that differ from it in a few digits only, resulting in a link set pointing to ID-s with a varying length of common prefix (including long prefixes); a well known way of achieving efficient routing.

As a backup, we also evolve a sorted ring topology using another instance of T-Man as described above, to maximize the probability that routing is successful. The routing table is composed of the neighbors in these two topologies, and the next hop for a target is selected based on numeric difference between the ID of the target and the table entries (now interpreting ID-s as numbers). We require strictly decreasing difference to avoid loops. The links from the ring topology are used only if no suitable links are available in the XOR-based topology. If the distance cannot be decreased but the target is not found, the routing attempt is failed.

Figure 6 illustrates the convergence of the routing performance while the topology is being evolved, starting from random routing tables. We can observe that the number of missed targets quickly becomes insignificant (from cycle 23 only 3 cases out of the 5300 shown in the figure), and the hop count of both the successful and unsuccessful routes remains low. Note that in our example, assuming a perfect topology, the worst case hop count would be 20.

Finally, note that this approach is mainly for illustration. The protocol presented in [8] for building the Chord [13] DHT represents a more realistic example.

7 Conclusions and Future Work

We have presented a protocol for topology management, T-Man, that is simple, general and fast. Simplicity makes it easier to implement, debug and understand. Generality allows it to be applied as an off-the-shelf component for prototyping or even as a production solution that could be implemented even before the final desired topology is known. In fact, the ranking function can be generated dynamically by users, or by some monitoring service, and the corresponding topology can be created on the fly. Finally, speed makes it possible to construct a topology quickly from scratch (recovery from massive failures or bootstrapping other protocols on demand) or where topology maintenance is in fact equivalent to the continuous re-creation of the topology (for example, due to massive churn).

Our current work is towards the application of T-Man for jump-starting existing DHT implementations and providing them robustness in the presence of massive failures and extreme churn [8]. We are also continuing our study of T-Man at an abstract level to better understand its behavior and characterize its scope and performance. In particular, it would be important to characterize the class of topologies that are "easy" or "difficult" for T-Man. Note that any *arbitrary* topologies can be expressed by at least one appropriate ranking function; in fact in general by many ranking functions: any function that ranks the neighbors in the target topology highest is suitable. This means that the open questions are: which of the possible ranking functions is optimal for a given problem, and how does convergence and the speed of convergence depend on the different topologies. Although the protocol does certainly not work with the same efficiency for all problems, we observed very similar performance for rather different and important topologies, so the empirical results are promising.

References

1. Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Vancouver, British Columbia, Canada, August 1987. ACM Press.
2. Patrick Th. Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulié. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5):60–67, May 2004.
3. Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In Hans-Arno Jacobsen, editor, *Middleware 2004*, volume 3231 of *Lecture Notes in Computer Science*, pages 79–98. Springer-Verlag, 2004.
4. Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, August 2005.
5. Yehuda Koren. Embedder. `http://www.research.att.com/~yehuda/index_programs.html`

6. Laurent Massoulié, Anne-Marie Kermarrec, and Ayalvadi J. Ganesh. Network awareness and failure resilience in self-organising overlays networks. In *Proceedings of the 22nd Symposium on Reliable Distributed Systems (SRDS 2003)*, pages 47–55, Florence, Italy, 2003.
7. Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, 2001.
8. Alberto Montresor, Márk Jelasity, and Ozalp Babaoglu. Chord on demand. In *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P 2005)*, pages 87–94, Konstanz, Germany, August 2005. IEEE Computer Society.
9. Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Building low-diameter peer-to-peer networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 21(6):995–1002, August 2003.
10. PeerSim. http://peersim.sourceforge.net/.
11. Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Rachid Guerraoui, editor, *Middleware 2001*, volume 2218 of *Lecture Notes in Computer Science*, pages 329–350. Springer-Verlag, 2001.
12. Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. Measuring and analyzing the characteristics of Napster and Gnutella hosts. *Multimedia Systems Journal*, 9(2):170–184, August 2003.
13. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 149–160, San Diego, CA, 2001. ACM, ACM Press.
14. Robbert van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2):164–206, May 2003.
15. Spyros Voulgaris and Maarten van Steen. An epidemic protocol for managing routing tables in very large peer-to-peer networks. In *Proceedings of the 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, (DSOM 2003)*, number 2867 in Lecture Notes in Computer Science. Springer, 2003.
16. Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering (ICDE 2003)*, Los Alamitos, CA, March 2003. IEEE Computer Society Press.

Basic Approach to Emergent Programming: Feasibility Study for Engineering Adaptive Systems Using Self-organizing Instruction-Agents

Jean-Pierre Georgé, Marie-Pierre Gleizes, and Pierre Glize

IRIT, Université Paul Sabatier,
118 route de Narbonne, 31400 Toulouse, France
{george, gleizes, glize}@irit.fr

Abstract. We propose to investigate the concept of an Emergent Programming Environment enabling the development of complex adaptive systems. This is done as a means to tackle the problems of the growth in complexity of programming, increasing dynamisms in artificial systems and environments, and the lack of knowledge about difficult problems and their solutions. For this we use as a foundation the concept of *emergence* and a multi-agent system technology based on cooperative self-organizing mechanisms.

The general objective is then to develop a complete programming language in which each instruction is an autonomous agent trying to be in a cooperative state with the other agents of the system, as well as with the environment of the system. By endowing these *instruction-agents* with self-organizing mechanisms, we obtain a system able to continuously adapt to the task required by the programmer (i.e. to program and reprogram itself depending on the needs). The work presented here aims at showing the feasibility of such a concept by specifying, and experimenting with, a core of *instruction-agents* needed for a sub-set of mathematical calculus.

1 Introduction

In the last few years, the use of computers has spectacularly grown and classical software development methods run into numerous difficulties. Operating systems are a good example of extremely complex software which are never exempt of problems. The classical approach, by decomposition into modules and total control, cannot guaranty the functionality of the software given the complexity of interaction between the increasing and variable number of modules, and the shear size of possibilities. Adding to this, the now massive and inevitable use of network resources and distribution only increases the difficulties of design, stability and maintenance.

1.1 Neo-computation Problems

This state is of interest to an increasing number of industrials, including IBM who wrote in a much relayed manifesto : "*Even if we could somehow come up*

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 16–30, 2006.

with enough skilled people, the complexity is growing beyond human ability to manage it. Pinpointing root causes of failures becomes more difficult, while finding ways of increasing system efficiency generates problems with more variables than any human can hope to solve. Without new approaches, things will only get worse" [14].

These kind of applications are what we call *neo-computation problems*, namely: autonomic computing, pervasive computing, ubiquitous computing [18], emergent computation, ambient intelligence, amorphous computing... This set of problems have in common the inability to define the global function to achieve, and by consequence to specify at the design phase, a derived evaluation function for the learning process. They are characterized by :

- a great number of interacting components (intelligent objects, agents, software);
- a variable number of these components during runtime (open system);
- the impossibility to impose a global control;
- an evolving and unpredictable environment;
- a global task to achieve.

1.2 Problem Solving by Emergence

Given the previous characteristics, the challenge is to find new approaches to conceive these new systems by taking into account the increasing complexity and the fact that we want reliable and robust systems. For this, because of the similarities, it seems opportune to look at natural systems - biological, physical or sociological - from an artificial system builder's point of view so as to understand the mechanisms and processes which enable their functioning.

In Biology for example, a lot of natural systems composed of autonomous individuals exhibit aptitudes to carry out complex tasks without any global control. Moreover, they can adapt to their surroundings either for survival needs or to improve the functioning of the collective. This is the case for example in social insects colonies [4] such as termites and ants [3]. The study of swarm behaviours by migratory birds or fish shoals also shows that the collective task is the result of the interactions between autonomous individuals. Non supervised phenomena resulting from the activity of a huge number of individuals can also be observed in human activities such as the synchronization of clapping in a crowd or traffic jams. But the most surprising is still the appearance of human consciousness out of the chemical and electrical jumble of our brain.

There is a common factor among all theses systems : the emergent dimension of the observed behaviour. Thus it is quite legitimate to study emergence so as to understand its functioning or at least to be able to adequately reproduce it for the design of artificial systems. This would enable the development of more complex, robust and adaptive systems, needed to tackle the difficulties inherent to *neo-computation* problems. In this way, interesting and useful emergent phenomena will be used in artificial systems when needed. Contrariwise, they will still appear sooner or later the more complex the systems are getting but will be unexpected

and unwanted. To prevent this, one orientation would be, in our opinion, that the scientific community studies and develops new theories based upon emergence. The prerequisites of such a theory could be resumed in four points :

- to start from the Systems Theory field;
- to focus on the parts of the system and their functioning;
- to depend neither from the systems finality, nor its environment (there can still be constraints or some form of feedback but there should be no imposed behaviour for the system);
- to be independent from the material support into which a given system will be incarnated (biological, technological, ...) : it has to be generic;

It is noteworthy that some research is already being done for quite some years now to bring emergence into artificial systems, but it is still very localized. For example, the *Santa Fe Institute* has acquired an international renown for its works on complexity, adaptive complex systems and thus emergence. These are also the preoccupations of *Exystence*, the European excellence network on complex systems, or the recently begun *ONCE-CS*, the Open network of Centres of Excellence in Complex Systems.

1.3 Going to the Lowest Level: The Instructions

If we suppose that we can manage to use the emergent phenomena to build artificial systems, this will be by specifying the behaviour of the parts of the systems so that it will enable their interactions to produce the expected global emergent behaviour of the system. A relevant question would be to ask about what parts we are focusing on and on which level. As with classical software engineering, any decomposition could be interesting, depending on the nature of the system being build.

We propose here to focus on the lowest possible level for any artificial system : the instruction level. We will explain our theoretical and experimental exploration of the concept of *Emergent Programming*. This concept is explained in the next section (section 2). Its use relies on emergence and self-organization (section 5) on one hand, and on a multi-agent approach called *AMAS* (Adaptive Multi-Agent System)[11] (section 3) on the other hand. A sub-problem which we called the *elementary example* has been thoroughly explored and is presented in section 4 where we then show how the learned lessons can lead us forward in our exploration of *Emergent Programming* and more generally of problem solving using emergence.

2 Emergent Programming

2.1 The Concept

In its most abstract view, *Emergent Programming* is the automatic assembling of instructions of a programming language using mechanisms which are not explicitly informed of the program to be created. We may consider that for a programmer to produce a program comes down to finding which instructions to

assemble and in which precise order. This is in fact the exploration of the search space representing the whole set of possible programs until the right program is found. However, if this exploration is easy when the programmer has a precise knowledge about the program he wants and how to obtain it, it grows more and more difficult with the increase of complexity of the program, or when the knowledge about the task to be executed by the program becomes imprecise or incomplete. Then are we not able to conceive an artificial system exploring efficiently the search space of the possible programs instead of having the programmer do it ? Only very few works exists on this topic. One noteworthy try has been done by Koza using Genetic Algorithms and a LISP language [16], but the main hindrance of GA is the need for a specific evaluation function for each problem, which can be very difficult to find. At the opposite, we aim at an as generic as possible approach.

To approach the problem of *Emergent Programming* concretely, we chose to rely on an adaptive multi-agent system using self-organizing mechanisms based on cooperation as it is described in the *AMAS* theory. This theory can be considered as a guide to endow the agents with the capacity to continuously self-organize so as to always tend toward cooperative interactions between them and with the environment. It then claims that a cooperative state for the whole system implies the functional adequacy of the system, i.e. that it exhibits a behaviour which satisfies the constraints of the parts of the system as well as from the environment (e.g. a user).

2.2 The Instruction-Agents

In this context, we define an agent as an instruction of a programming language. Depending on the type of the instruction he is representing, the agent possesses specific competences which he will use to interact with other *instruction-agents*. A complete program is then represented by a given organization of the *instruction-agents* in which each agent is linked with partners from which he receives data and partners to which he sends data. The counterpart of the execution of a classical program is here simply the activity of the multi-agent system during the exchange of data between the agents.

2.3 The Reorganization Process

We can now appreciate all the power of the concept : a given organization codes for a given program, and thus, changing the organization changes the final program. It comes down to having the agents self-organize depending on the requirements from the environment so as to continuously tend toward the adequate program (the adequate global function). In principle, we obtain a system able to explore the search space of the possible programs in place of the programmer. Everything depends on the efficiency of the exploration to reach an organization producing the right function. An important part of our work on *Emergent Programming* has been the exploration of the self-organization mechanisms which enable the agents to progress toward the adequate function, depending on the

constraints of the environment but without knowing the organization to reach or how to do it (since this is unknown for the problems we are interested in).

2.4 A Neo-programming Environment

The system will not be able to grow *ex nihilo* all by itself, all the more if we want to obtain higher level programs (programs with more complex behaviours). As the programmer with his classical programming environment, the *neo-programmer* will have to affect the development of the system through a *neo-programming environment*, at least at the beginning. It is a matter of supplying the tools to shape the environment of the system so as to have this environment constrain the system toward the adequate function. But in a pure systems theory's view, the *neo-programmer* is simply part of the environment of the system.

But the *neo-programming environment* will certainly have to be more than a simple envelope for the developing system. We will probably need to integrate some tools for the observation of the evolution of the system, means to influence this evolution, the type and proportions of *instruction-agents*, to affect some aspects of the structure. Moreover, a complex program is generally viewed as a modular construct and the *neo-programmer* may want to influence this modular structure, either by manipulating some sorts of "bricks", each being an *emergent programming* system, or by letting these "bricks" self-organize in the same manner as their own components.

At the end, we will obtain a system able not only to "find" *how* to realize the adequate function, but also to continuously adapt to the environment in which it is plunged, to react to the strongly dynamic and unpredictable nature of real world environments, and all this by presenting a high grade of robustness. Indeed, because of its nature, the system would be able to change its internal structure any time and by consequence its performed function, or even grow by adding instructions to respond to some partial destruction or to gain some new competences.

The research we did on *Emergent Programming* was to explore the feasibility of the concept. For this, we restrained the programming language to the instructions needed for a subset of mathematical calculus, of which the *elementary example* (section 4) is a representative. We specified such a core of agents and put it through experimentation. For this an environment has been implemented : *EPE* (*Emergent Programming Environment*) [9]. These experimentations enabled us to explore different self-organization mechanisms for the *instruction-agents* so as to find those who lead to the emergence of the adequate function. Part of these mechanisms are described here.

3 Using Cooperative Agents as the Engine for Self-organization

3.1 Adapt the System by Its Parts

We consider that each part P_i of a system S achieves a partial function f_{pi} of the global function f_s (Figure 1). f_s is the result of the combination of the partial

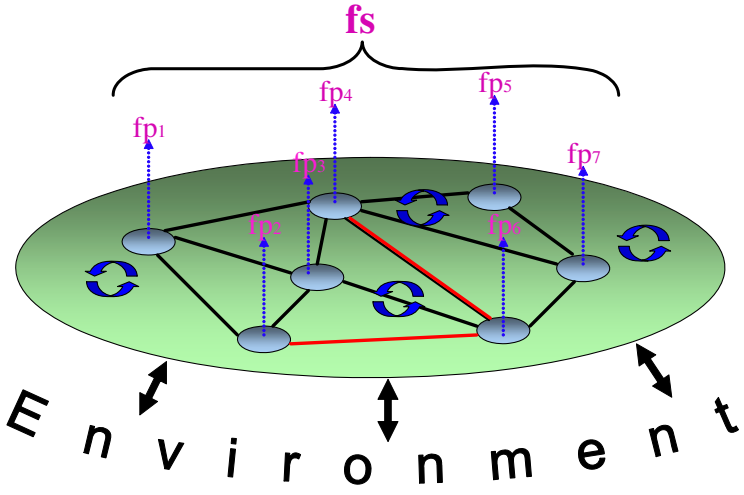


Fig. 1. Adaptation: changing the function of the system by changing the organization

functions f_{pi}, noted by the operator "o". The combination being determined by the current organization of the parts, we can deduce $fs = f_{p1} \, o \, f_{p2} \, o \, ... \, o \, f_{pn}$. As generally $f_{p1} \, o \, f_{p2} \neq f_{p2} \, o \, f_{p1}$, by transforming the organization, the combination of the partial functions is changed and therefore the global function f_s changes. This is a powerful way to adapt the system to the environment. A pertinent technique to build this kind of systems is to use adaptive multi-agent systems. As in Wooldridge's definition of *multi-agent systems* [19], we will be referring to systems constituted by several autonomous agents, plunged in a common environment and trying to solve a common task.

3.2 The Theorem of Functional Adequacy

Cooperation was extensively studied in computer science by Axelrod [1] and Huberman [15] for instance. "*Everybody will agree that co-operation is in general advantageous for the group of co-operators as a whole, even though it may curb some individual's freedom*" [12]. Relevant biological inspired approaches using cooperation are for instance *Ants Algorithms* [7] which give efficient results in many domains. In order to show the theoretical improvement coming from co-operation, we have developed the *AMAS* (Adaptive Multi-Agent System)[11] theory which is based upon the following theorem. This theorem describes the relation between cooperation in a system and the resulting functional adequacy[1] of the system.

[1] "Functional" refers to the "function" the system is producing, in a broad meaning, i.e. what the system is doing, what an observer would qualify as the behaviour of a system. And "adequate" simply means that the system is doing the "right" thing, judged by an observer or the environment. So "functional adequacy" can be seen as "having the appropriate behaviour for the task".

Theorem. *For any functionally adequate system, there exists at least one cooperative internal medium system that fulfils an equivalent function in the same environment.*

Definition. *A cooperative internal medium system is a system where no* Non-Cooperative Situations *exist.*

Definition. *An agent is in a Non-Cooperative Situation (*NCS*) when : (1) a perceived signal is not understood or is ambiguous; (2) perceived information does not produce any activity of the agent; (3) the conclusions are not useful to others.*

3.3 Consequence

This theorem means that we only have to use (and hence understand) a subset of particular systems (those with cooperative internal mediums) in order to obtain a functionally adequate system in a given environment. We concentrate on a particular class of such systems, those with the following properties [11]:

- The system is cooperative and functionally adequate with respect to its environment. Its parts do not 'know' the global function the system has to achieve via adaptation.
- The system does not have an explicitly defined goal, rather it acts using its perceptions of the environment as a feedback in order to adapt the global function to be adequate. The mechanism of adaptation is for each agent to try and maintain cooperation using their skills, representations of themselves, other agents and environment.
- Each part only evaluates whether the changes taking place are cooperative from its point of view - it does not know if these changes are dependent on its own past actions.

This way of engineering systems has been successfully applied on numerous applications with very different characteristics for the last ten years (autonomous mechanisms synthesis[6], flood forecast[10], electronic commerce and profiling,...). On each, the local cooperation criterion proved to be relevant to tackle the problems without having to resort to an explicit knowledge of the goal an how to reach it.

3.4 The Engine for Self-organization

The designer provides the agents with local criterion to discern between cooperative and non-cooperative situations. The detection and then elimination of NCS between agents constitute the engine of self-organization. Depending on the real-time interactions the multi-agent system has with its environment, the organization between its agents emerges and constitutes an answer to the aforementioned difficulties of *neo-computation problems* (indeed, there is no global control of the system). In itself, the emergent organization is an observable organization that has not been given first by the designer of the system. Each agent computes a partial function f_{pi}, but the combination of all the partial functions

produces the global emergent function fs. 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-organization.

By principle, the emerging purpose of a system is not recognizable by the system itself, its only criterion must be of strictly local nature (relative to the activity of the parts which make it up). By respecting this, the *AMAS* theory aims at being a theory of emergence.

4 The Elementary Example

We tried to find an *emergent programming* system as simple as possible (i.e. with the smallest number of agents with the simplest functioning), but still needing reorganizations so as to produce the desired function. The advantages of such a case study are that it is more practical for observation, that it leads to less development complexity and that it presents a smaller search space.

4.1 Description

The specification of each agent depends on the task he has to accomplish, of his "*inputs*" and "*outputs*". The agents communicate by messages but to accomplish the actual calculation, we can consider that the agents are expecting values as inputs to be able to provide computed values as outputs. Schematically, we can consider exchanges between agents as an electronic cabling between outputs and inputs of agents.

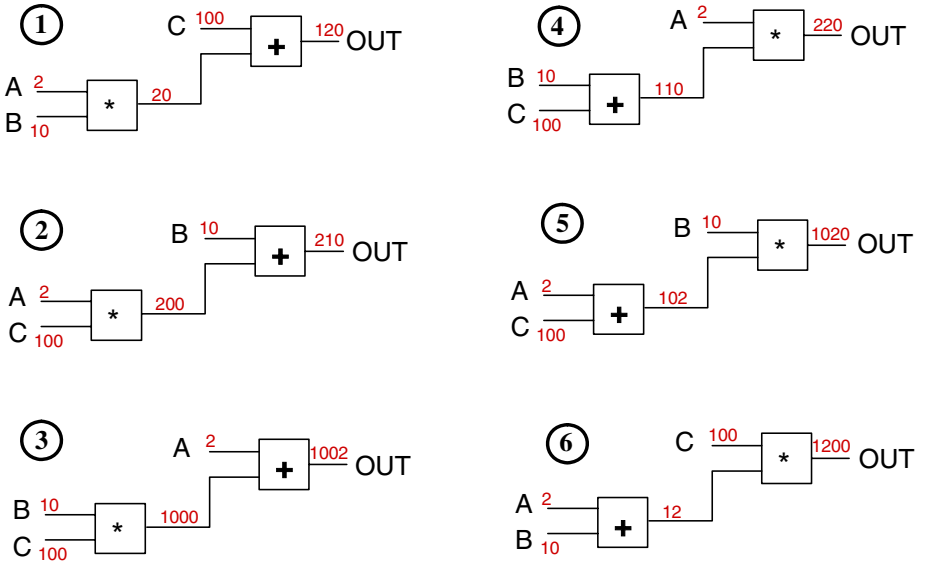


Fig. 2. The 6 different possible types of functional organizations for the elementary example

The elementary example we choose is constituted of 6 agents : 3 "*constant*" agents, an "*addition*" agent, a "*multiplication*" agent and an "*output*" agent. A "*constant*" agent is able to provide the value which has been fixed at his creation. The 3 the system contains have been given sufficiently different values so as to prevent calculation ambiguity : *AgentConstantA* (value = 2), *AgentConstantB*(value = 10) and *AgentConstantC* (value = 100). Combined with *AgentAddition* and *AgentMultiplication*, the values produced by the system are results from organizations like $(A+B)*C$ or any other possible combination. *AgentOut* simply transmits the value he receives to the environment. But he is also in charge of retrieving the feedback from the environment and forward it into the system.

The size of the complete search space is 6^5, that is 7776 theoretically possible organizations, counting all the incomplete ones (i.e. where not every agent has all his partners). There are 120 complete organizations and among those, 24 are functional (they can actually calculate a value) if we count all the possible permutations on the inputs which do not change the calculated value. In the end, we have 6 types of different organization (cf. Figure 2) producing these 6 values : 120, 210, 220, 1002, 1020 and 1200. The aim is to start without any partnerships between agents and to request that the system produces the highest value for example.

4.2 Reorganization Mechanisms

In accordance with the *AMAS* theory, the agent's self-organizing capacity is induced by their capacity to detect NCS (Non-Cooperative Situations), react so as to resorb them and continuously act as cooperatively as possible. This last point implies in fact that the agent also has to try to resorb NCS of other agents if he is aware of them: to ignore a call for help from another agent is definitely not cooperative. We will illustrate this with the description of two NCS and how they are resorbed.

Detection
NCSNeedIn : The agent is missing a partner on one of his inputs. Since to be cooperative in the system he has to be useful, and to be useful he has to be able to compute his function, he has to find partners able to send values toward his input.

Most NCS lead the agent to communicate so as to find a suitable (new) partner. These calls, because the agents have to take them into account, also take the shape of NCS.

NCSNeedInMessage : The agent receives a message informing him that another agent is in a *NCSNeedIn* situation.

Resorption
NCSNeedIn : This is one of the easiest NCS to resorb because the agent only has to find any agent for his missing input. And the agents are potentially always able to provide as many values on their outputs for as many partners as needed. The agent has simply to be able to contact some agent providing values of the right

type (there could be agents handling values of different types in a system), i.e. corresponding to his own type. So he generates a *NCSNeedInMessage* describing his situation (his needs) and send it to his acquaintances (because they are the only agents he knows).

NCSNeedInMessage : The agent is informed of the needs of the sender of the NCS and his cooperative attitude dictates him to act. First, he has to judge if he is relevant for the needs of the sender, and if it is the case, he has to propose himself as a potential partner. Second, even if he is not himself relevant, one of its acquaintances may be. He will do what the *AMAS* theory calls a resorption by restricted propagation : he tries to counter this NCS by propagating the initial message to some acquaintances he thinks may be the most relevant.

For each NCS the agent is able to detect (there are 10 NCS in total for these agents), a specific resorption mechanism has been defined. It is a precise description of the decision making of the agent depending on his state and on what it perceives. For other NCS, the mechanisms become quite complicated, and require a long description. For an exhaustive presentation, please refer to [9].

These NCS and their symmetric for a missing partner on an output enable the system to produce an organization where each agent has all his needed partners. To obtain the functional adequacy for the system means that the final organization is able to produce the expected result. The main question is how to introduce mechanisms in the resorption of the NCS to enable the agents as a whole to reach this organization. For this, they need some kind of "*direction*" (but on local criterion) to get progressively closer to the solution, a local information to judge this proximity. The information used here is simply a "smaller/bigger" feedback type that the environment sends to the system and that will be dispatched between the agents by propagation and by taking other the goal (smaller or bigger). The agent then tries to satisfy its new goal and staying at the same time the most cooperative possible with the other agents. This will bring the system as a whole to produce a smaller or bigger value.

Of course, the agents will get into conflict with other agents when trying to reach these goals and the self-organizing mechanisms take that into account. Each agent also manipulates a knowledge about the prejudice he inflicts or may inflict following changes he induces in the organization. For this, the agents communicate to each other, when necessary, their current goal and state. When choosing a new partner an agent takes into account the impact of its decision upon the concerned agents, i.e which agents will be hindered from reaching their goal, which agents will be in a worse state than before and in what proportion. By minimizing these prejudices (which is a form of cooperation), the whole organization progresses.

It is important to note that the information which is given as a feedback is not in any way an explicit description about the goal and *how* to reach it. Indeed, this information does not exist : given a handful of values and mathematical operators, there is no explicit method to reach a specific value even for a human. They can only try and guess, and this is also what the agents do. That is why we believe the resolution we implemented to be in the frame of emergence.

4.3 Results and Discussion

Results. The elementary example has been implemented in Java as a multi-threaded agent platform able to run any type of instruction-agents. It also simulates the environment for an organization of instruction-agents and provides tools for the observation and analysis of the reorganization process.

First of all, the internal constraints of the system are solved very quickly : in only a few reorganization moves (among the 7776 possible organizations), all the agents find their partners and a functional organization is reached. Then, because the system is asked to produce the highest value for example (configuration 6, Figure 2), other NCS are produced and the system starts reorganizing toward its goal. In accordance with the AMAS theory, the system is considered to provide an adequate behaviour when no more NCS are detected (the environment is satisfied by the produced results).

On a few hundred simulations, the functional adequacy is reached in a very satisfactory number of organization changes. Since the search space if of 7776 possible organizations, a blind exploration would need an average of 3.888 checked organizations to reach a specific one. Since a functional organization possesses 4 identical instances for a given value (by input permutations), we would need 972 tries to get the right value. Experimentation shows that, whatever the initial organization (without any links or one of the 6 functionals), the system needs to explore less than a hundred organizations among the 7776 to reach one of the 4 producing the highest value. We consider that this self-organization strategy allows a relevant exploration of the search space. A noteworthy result is also that whatever organization receives the feedback for a better value, the next organization will indeed produce a better value (if it exists).

Emergent Programming : A Universal Tool. If we define all the agents needed to represent a complete programming language (with agents representing variables, allocation, control structures, ...) and if this language is extensive enough, we obtain maximal expressiveness : every program we can produce with current programming languages can be coded as an organization of *instruction-agents*. In its absolute concept, *Emergent programming* could then solve any problem, given that the problem can be solved by a computer system. Of course, this seems quite unrealistic, at least for the moment.

Problem Solving Using Emergence. But if we possess some higher-level knowledges about a problem, or if the problem can be structured at a higher level than the instruction level, then it is more efficient and easier to conceive the system at a higher level. This is the case for example when we can identify entities of bigger granularity which therefore have richer competences and behaviours, maybe adapted specifically for the problem.

Consequently, we will certainly be able to apply the self-organizing mechanisms developed for Emergent Programming to other ways to tackle a problem. Indeed, *instruction-agents* are very particular by the fact that they represent the most generic type of entities and that there is a huge gap between their functions and the function of a whole program. The exploration of the search

space, for entities possessing more information or more competences for a given problem can only be easier. In the worst case, we can always try to use Emergent Programming as a way to specify the behaviour of higher-level entities (recursive use of emergence).

Let us consider for instance the problem of ambient intelligence : in a room, a huge number of electronic equipments controlled each by an autonomous microchip have as a goal the satisfaction of the users moving around it from day to day. The goal itself, user satisfaction, is really imprecise and incomplete, and the way to reach it even more. We claim that this problem is an ideal candidate for a problem solving by emergence approach: let us endow the entities with means to find by themselves the global behaviour of the system so as to satisfy the users. The challenge is to define the "right" self-organizing behaviours for the different equipments for them to be able to modify the way they interact to take into account the constraints of every one of them plus the external stimuli from the users (order, judgement, behaviour, ...). And we are convinced that this can only be done if the self-organization mechanisms tightly fit the frame of emergence.

5 Emergence and Self-organization

If we study specialized literature on emergence or self-organization, we can see that these are tightly linked. Yet, at the same time, we can see a lot of works focusing exclusively on the second without any mention, or only a brief, about the first. One explanation could be that the notion of emergence is quite abstract, even philosophical, making it difficult to fully grasp and therefore delicate to manipulate. At the opposite, self-organization is more concrete by its description in terms of mechanisms and thus, more easily used. But by concentrating solely on the mechanisms, are we not taking the risk to leave the frame of emergence? We give here a description of self-organization integrating emergence.

5.1 What Is Self-organization ?

The self-organization field has from the very beginning tried to explore the internal mechanisms of systems producing emergent phenomena[2]. They tried to find the general functioning rules explaining the growth and evolution of the observed systemic structures, to find the shapes the systems could take, and finally to produce methods to predict the future organizations appearing out of changes happening at the component level of the systems. And these prospective results had to be applicable on any other system exhibiting the same characteristics (search for generic mechanisms). There are abundant definitions and descriptions of characteristics of emergence and self-organization in literature. We can sum it up as this:

Definition. *Self-organization is the set of processes within a system, stemming from mechanisms based on local rules which lead the system to produce structures or specific behaviours which are not dictated by the outside of the system* [8][13][17].

5.2 Understanding Self-organization

In most definitions about emergence and self-organization, there is the notion under some form or another of strictly local rules and resulting behaviours. There is also the strong constraint for these behaviors not to be imposed, dictated, explicitly informed or constrained by the environment of the system. The local character of a rule gives a strict and clear framework. But concerning the influence of the environment on the system, being it directly or through some internal rules, the exact characterization of this influence can be particularly difficult and vague.

Let us take the example of Bénard convection cells which is a classical example of self-organization. The phenomenon produced by self-organization is here the shape of the movement of water molecules which creates these particular observable flux structures (when looking top down at water just before it starts boiling, we can see hexagonal cells covering the bottom). The local rules are here the movement and collisions of the molecules. The fact that the molecules move more easily when they move in the same direction (because of less collisions) creates circulation fluxes. But the surfaces of the container as well as the influx of heat which forces the molecules to move are indeed part of the environment of the system and influence the behaviour of the system. We then have to decide of the impact and nature of this influence on the behaviour of the molecules and system. We can argue that it is indeed this influx of heat which compels the molecules to move and that the surfaces of the container also strongly constrain these movements. But this is not enough to explain *how* the molecules have to move, only that they have to, and in a given border. It is indeed the local collision rules which lead to the emergence of the hexagonal cells. The frame of self-organization seems here relatively clear after analysis but could have been argued against at the beginning.

In fact, in many cases the environment dictates very strong templates for the system to follow. Even if these templates are followed at a local level by autonomous entities, the more strong and precise they are, the less we think we can pretend to be in a self-organization frame. When wanting to use self-organization as the internal mechanism of an artificial system, we have to keep a critical attention on the influence the environment has on the system.

5.3 Using Emergence in Artificial Systems

Our work in this domain during the last decade lead us to give a "technical" definition of emergence in the context of multi-agent systems, and therefore with a strong computer science colouration. It is based on three points: what we want to be emergent, at what condition it is emergent and how we can use it [5].

1. **Subject.** The goal of a computational system is to realize an adequate function, judged by a relevant user. It is this function (which may evolve during time) that has to emerge.
2. **Condition.** This function is emergent if the coding of the system does not depend on the knowledge of this function. This coding has to contain the

mechanisms facilitating the adaptation of the system during its coupling with the environment, so as to tend toward an adequate function.

3. **Method.** To change the function the system only has to change the organization of its components. The mechanisms which allow the changes are specified by self-organization rules providing autonomous guidance to the components' behaviour without any explicit knowledge of the collective function nor how to reach it.

6 Conclusion

We aimed at studying the feasibility of the concept of *Emergent Programming* by using self-organizing *instruction-agents*. We presented in this paper the concept and how we studied it. For this, we first described the frame of self-organization and emergence as we think can be applied in artificial systems. Then we described a generic approach for adaptive systems based upon a multi-agent system where the agents are endowed with self-organizing mechanisms based upon cooperation and emergence.

An elementary example has been used as a case study. Its implementation, and experimentation with, lead to the definition of the self-organizing mechanisms of the *instruction-agents* so as to enable them to make the system reach a given goal.

This study has been an interesting work to explore self-organization in MAS when confronted to difficult problems that we are persuaded need an Emergent solution. We claim that this approach would be really relevant for *neo-computation* problems such as ambient intelligence, if not directly with *instruction-agents*, by using the same kind of cooperative self-organization mechanisms.

References

1. R. Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 1984.
2. P. Ball. *The Self-Made Tapestry.* Oxford Press, 1999.
3. E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence - from natural to artificial systems.* Oxford University Press, 1999.
4. S. Camazine, J.-L. Deneubourg, N. Franks, J. Sneyd, and E. Theraulaz, G.and Bonabeau. *Self-organization in biological systems.* Princeton University Press, 2002.
5. D. Capera, J. Georgé, M.-P. Gleizes, and P. Glize. Emergence of organisations, emergence of functions. In *AISB'03 symposium on Adaptive Agents and Multi-Agent Systems*, April 2003.
6. D. Capera, M.-P. Gleizes, and P. Glize. Mechanism type synthesis based on self-assembling agents. *Journal on Applied Artificial Intelligence*, 18(8-9), 2004.
7. M. Dorigo and G. Di Caro. *The Ant Colony Optimization Meta-Heuristic.* McGraw-Hill, 1999.
8. J. Georgé, B. Edmonds, and P. Glize. *Self-organizing adaptive multi-agent systems work*, chapter 16, pages 321–340. Kluwer Publishing, 2004.

9. J.-P. Georgé. *Résolution de problèmes par émergence - Étude d'un Environnement de Programmation Émergente*. PhD thesis, Université Paul Sabatier, Toulouse, France, 2004. http://www.irit.fr/SMAC/EPE.html.
10. J.-P. Georgé, M.-P. Gleizes, P. Glize, and C. Régis. Real-time simulation for flood forecast: an adaptive multi-agent system staff. In D. Kazakov, D. Kudenko, and E. Alonso, editors, *Proceedings of the AISB'03 symposium on Adaptive Agents and Multi-Agent Systems(AAMAS'03)*, pages 109–114, University of Wales, Aberystwyth, April 7-11 2003. SSAISB.
11. P.-P. Gleizes, V. Camps, and P. Glize. A theory of emergent computation based on cooperative self-oganization for adaptive artificial systems. In *Fourth European Congress of Systems Science*, Valencia, Spain, 1999.
12. F. Heylighen. Evolution, selfishness and cooperation; selfish memes and the evolution of cooperation. *Journal of Ideas*, 2(4):70–84, 1992.
13. F. Heylighen. *Encyclopedia of Life Support Systems*, chapter The Science of Self-organization and Adaptivity. EOLSS Publishers Co. Ltd, 2001.
14. P. Horn. Autonomic computing - ibm's perspective on the state of information technology. http://www.ibm.com/research/autonomic, 2001.
15. B. Huberman. *The performance of cooperative processes*. MIT Press / North-Holland, 1991.
16. J. R. Koza. Evolution and co-evolution of computer programs to control independently-acting agents. In *From animals to animats : proceedings of the first international conference on Simulation of Adaptative Behavior (SAB)*. MIT Press, 1991.
17. I. Prigogine and G. Nicolis. *Self Organization in Non-Equilibrium Systems*. J. Wiley and Sons, New York, 1977.
18. M. Weiser and J. S. Brown. Designing calm technology. *PowerGrid Journal*, 1(1), 1996.
19. M. Wooldridge. *An introduction to multi-agent systems*. John Wiley & Sons, 2002.

ETTO: Emergent Timetabling by Cooperative Self-organization

Gauthier Picard, Carole Bernon, and Marie-Pierre Gleizes

IRIT, Université Paul Sabatier,
F-31062 Toulouse Cedex, France
{picard, bernon, gleizes}@irit.fr
http://www.irit.fr/SMAC

Abstract. Cooperation is a means for multi-agent systems to function more efficiently and more adaptively. Cooperation can be viewed as a local criterion for agents to self-organize and then to perform a more adequate collective function. This paper mainly aims at showing that with only local rules based on cooperative attitude and without any global knowledge, a solution is provided by the system and local changes lead to global reorganization. This paper shows an application of cooperative behaviors to a dynamic distributed timetabling problem, ETTO, in which the constraint satisfaction is distributed among cooperative agents. This application has been prototyped and shows positive results on adaptation, robustness and efficiency of this approach.

1 Introduction

As a consequence of the growing complexity in the number of stakeholders or in the dynamics of software environments, artificial systems are more and more difficult to suitably design. The global function of those systems is often fuzzily specified but parts of the system are easily identifiable and local theories to model are well known. Such systems, qualified as *emergent* are studied by biologists and physicians for some years now. The two main properties of these systems are: the *irreductibility* of macro-theories to micro-theories [1] and the *self-organizing mechanisms* which are the origin of adaptivity and appearance of new emergent properties [8].

Kohonen networks or ant algorithms are two relevant examples of artificial transcriptions of self-organizing mechanisms [9, 3]. When considering less specific tasks, deciding when to reorganize in order to adapt to the environmental pressure, for reaching a global goal, needs equipping parts of the system with cognitive capabilities. Therefore parts become autonomous agents. As a response to this need of decision-making, the AMAS (*Adaptive Multi-Agent Systems*) approach proposes *cooperative attitude* as the local criterion used by agents to reorganize. Here, coooperation is not limited to ressource or task sharing, but is a behavioral guideline. Cooperation is viewed in a proscriptive way: agents have to locally change their way to interact when they are in *non cooperative situations* (or NCS). In AMAS, an agent is cooperative if it verifies the following meta-rules [5]:

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 31–45, 2006.

c_{per}: perceived signals are understood without ambiguity,
c_{dec}: received information is useful for the agent's reasoning,
c_{act}: reasoning leads to useful actions toward other agents.

If an agent detects it is in a NCS ($\neg c_{per} \vee \neg c_{dec} \vee \neg c_{act}$), it has to act to come back to a cooperative state and therefore to change the organization. The functional adequacy theorem [7] ensures that the function of the system is adequate – the system produces a function which is cooperative[1] for its environment – if every agent has such a cooperative behavior. Therefore, designing adaptive systems is equivalent to providing agents with a cooperative attitude and then ensuring the functional adequacy of the system.

The objective of this paper is to show that with only local rules based on cooperative attitude and without any global knowledge, a solution is provided by the system and local changes lead to global reorganization and then to a more adapted global function. In the next sections, this approach is illustrated by defining a cooperative behavior for agents having to dynamically solve an academic timetabling problem. Teachers and students groups have to find partners, time slots and rooms to give or to take some courses. Each actor has some constraints concerning its availabilities or required equipment. Moreover, a teacher can add or remove constraints at any time during the solving process via an adapted user interface. Such an application clearly needs adaptation and robustness. The system must be able to adapt to environmental disturbances (constraints modifications) and not to compute new solutions at each constraint changing. The correct organization has to emerge from actors interactions. This problem has been called ETTO, for *Emergent TimeTabling Organization*. To solve this problem, two kinds of agents have been identified and are presented in section 2. These agents respect several cooperation rules that are expounded in section 3. By now, the goal is not to be the most efficient, but to provide mechanisms to tackle changes with a minimal impact, as experiments show with results on adaptation and robustness of the AMAS approach in section 4. Section 5 discusses the approach and compares it to existing ones before concluding in section 6.

2 ETTO Agents

Two different classes of agents have been identified to tackle the ETTO problem: *Representative Agents* (RA) and *Booking Agents* (BA). The exploration of the solution space, a n-dimensional grid of cells, is delegated to Booking Agents. Each cell c_i of the grid is constrained (time slots, number of places, ...). All the constraints of a cell are regrouped in a set C_{c_i}. Cooperation between agents must lead to a correct organization by efficiently exploring the grid. This representation is shown in figure 1.

2.1 Representative Agents

RAs are the interface between human actors (teachers or student groups) and the timetabling system. They own constraints (called *intrinsic constraints*) about

[1] Not antinomic and not useless.

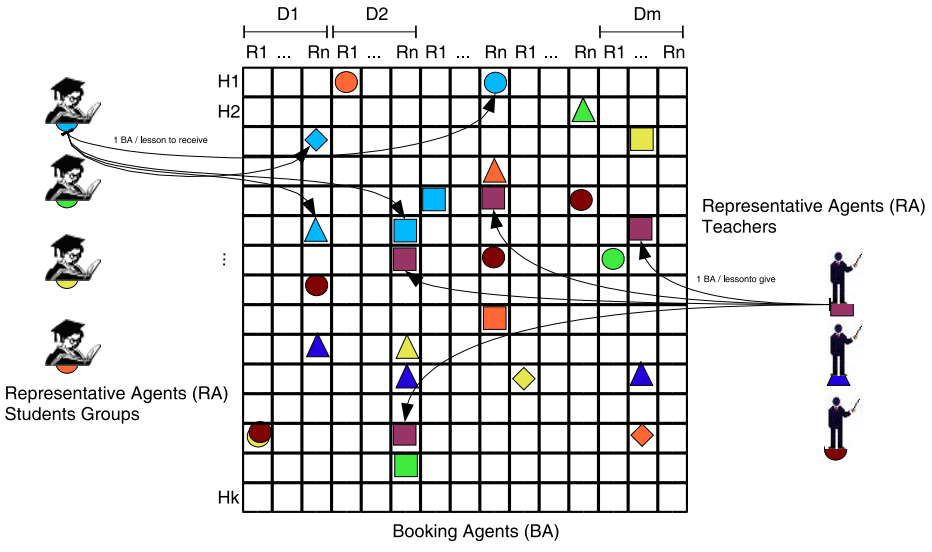


Fig. 1. Agents and environment in ETTO – each actor owns a Representative Agent and several Booking Agents to search for partners and reservations in the n-dimensional grid to meet personal constraints

availability, equipment requirements (projectors), or any other kind of personal constraint. To concurrently explore the possibilities of partnership and room reservation, those agents delegate exploration to Booking Agents. A RA (called *proxy*) creates as many BAs (called *delegates*) as it has courses to give (for teachers) or to take (for student groups), and randomly positions them on the schedule grid at the beginning of the solving. BAs from a same RA are called *brothers*. The task of a RA is simple: to warn its delegate BAs when its user adds or removes constraints and to inform all its delegate BAs when one of its delegate BAs produces new constraints (called *induced constraints*), to ensure coherency.

2.2 Booking Agents

BAs are the real self-organizing agents in ETTO. They have to reserve time slots and rooms and to find partners (student groups for teachers and vice versa) in accordance with constraints owned by their proxy.

In a cooperative situation, a BA, which is in a grid cell (i.e. a time slot in a room for a given day) books it and partners with another BA. But this nominal situation is not ensured at the beginning since BAs are randomly positioned. BAs then need to reorganize, i.e. to change their partnership and reservations, to find an adequate timetable. Such situations are NCS (see section 1). Therefore, BAs must be able to respond to NCS by respecting cooperation rules (see section 3).

Actions a BA can perform are simple: to partner (or unpartner) with another BA, to book (or unbook) a cell (by marking it with a virtual *post-it*

with its address), to move to another cell[2], and to send messages to other agents *it knows*. A BA only knows its proxy and the BAs it encounters at runtime.

The life-cycle of a BA is a classical "perceive-decide-act" process as proposed by Capera et al [5]:

1. During the *perception* phase, the BA checks its messages (coming from other BAs or its proxy) and updates data about the cell (BAs in the cell, post-its, properties of the cell) in which it is positioned,
2. During the *decision* phase, the BA must choose the next action to perform to be as cooperative as possible, in accordance with the cooperation rules,
3. During the *action* phase, the BA performs the chosen action.

To perform its tasks, a BA ba_i has the following *local* properties, capabilities and knowledge:

- its current position in the grid ($cell(ba_i)$), which is the only cell the BA ba_i can see since it does not have a global knowledge of the whole grid,
- its current partner ($partnership(ba_i, ba_j)$ with $i \neq j$),
- its current reservation of the cell c_j ($reservation(ba_i, c_j)$ and $rCell(ba_i)$),
- its proxy ($proxy(ba_i)$),
- its search time ($time(ba_i)$) for a reservation, since it has no more reservation,
- the time slot of a cell ($slot(c_j)$),
- a limited memory of known BAs to send messages to (the set $knows(ba_i)$ or the predicate $knows(ba_i, ba_j)$), which is empty at the beginning of the solving and will be updated during the grid exploration,
- a set of intrinsic constraints (CI_{ba_i}) which are attached to the BA at its creation by its proxy RA,
- a set of constraints induced by its brothers (CB_{ba_i}) which are attached to and updated by its proxy RA when one of its brother reserves a cell to avoid ubiquity situations (two BAs of the same RA book the same time slot, for example),
- a set of constraints induced by its partner (CP_{ba_i}) which are attached to each partnership and updated when the partner changes its constraints to take into account the partner's preferences,
- a set of constraints induced by its reservation (CR_{ba_i}) to avoid partnering with a BA which is not available at certain time slots,
- the set of constraints from a first set which are non compatible with constraints of a second set ($nonCompatible(C_i, C_j) \subseteq C_i$) to process potential partners or cells to reserve,
- a function to weight constraints ($w(c_i) > 0$). The higher the weight is, the more difficult the constraint can be relaxed. A constraint c_i cannot be relaxed if $w(c_i) = +\infty$.

[2] BAs do not know the whole grid, so moving to another cell implies defining the cells it knows from it.

A macro, NC, is defined to simplify notations:

Definition 1. *The set of non compatible constraints between two BAs is* $NC_{ba_i,ba_j} = nonCompatible(CI_{ba_i} \cup CB_{ba_i} \cup CR_{ba_i}, CI_{ba_j} \cup CB_{ba_j} \cup CR_{ba_j})$.

To determine the non compatible constraints between two BAs, the constraints coming from partners (CP) are not taken into account. In the same manner, for determining if a cell is compatible with a BA's constraints, constraints from the current reservation (CR) are not included:

Definition 2. *The set of non compatible constraints between a BA and a cell is* $NC_{ba_i,c_j} = nonCompatible(CI_{ba_i} \cup CB_{ba_i} \cup CP_{ba_i}, C_{c_j})$.

By using NC, two constraint owners (BA or cell) can know if they are compatible:

Definition 3. $compatible(x, y) \equiv (NC_{x,y} = \emptyset)$.

Before starting the solving, there is no absolute way to decide what are the most difficult sub-problems to solve. Moreover, the difficulty degree could evolve due to the dynamic evolution of the problem description. Therefore, during the solving, each agent must be able to evaluate the difficulty it has to find a partner or a reservation. A BA ba_i can calculate the cost of a reservation of a cell c_j ($rCost(ba_i, c_j)$) and the cost of a partnership with another BA ba_j ($pCost(ba_i, ba_j)$) as following:

- $rCost(ba_i, c_j) = (\sum_{c \in NC_{ba_i,c_j}} w(c))/time(ba_i)$,
- $pCost(ba_i, ba_j) = \sum_{c \in NC_{ba_i,ba_j}} w(c)$.

Dividing by the $time(ba_i)$ of search prioritizes the BA which is searching a cell for a long time. In fact, informally, helping agents having difficulties to find a position within the organization is cooperative.

2.3 Basic Behavior

BAs have two orthogonal goals: *find a partner* and *find a reservation*. The main resolution algorithm is distributed among BAs and relies on the cooperation between agents. Solving is the result of dynamic interactions between distributed entities (BAs). As BAs have to reach two main individual goals, the nominal behavior they follow can be expressed in terms of the achievement of these goals, as shown in the algorithm 1.

During the perception phase, the BA checks its mailbox, in which other BAs can put messages about partnership requests or reservations. If these messages inform that its goals are reached (partnership and reservation), it moves to its reserved cell only if this reservation is not too constrained. If the agent has relaxed some or if it lacks partner or reservation, it will explore the grid to find a (better) solution and analyze encountered BAs in memory and known cells, i.e. it verifies whether encountered BAs or cells better fit its constraints. Exploring the grid implies the capability for the agent to choose a next cell to explore. In the next experimentation, this is randomly done.

Algorithm 1. Basic behavior for a BookingAgent

```
while alive do
  processMessages()
  if partner AND reservation then              //reservation is optimal
    if rCost(ba_i,rCell(ba_i)) == 0 then
      moveTo(reservedCell)
    else          //analyze cell to find either partner or reservation
      processCurrentCell()
    endif
  else
    moveTo(nextCell);                    //choose another cell to explore
    addBAsToMemory();                 //memorize BAs which are in the cell
    processEncounteredBAs(); //verify whether they fit with constraints
    if NOT (reservation OR partner) then            //goals not reached
      processCurrentCell()                      //analyze the current cell
    endif
  endif
done
```

2.4 Constraint Management and Related Works

Actions may lead to add new induced constraints. For example, a BA which books a cell corresponding to a given hour at a given day warns its brothers, via its proxy, that this time slot is forbidden to avoid ubiquity situations. Conversely, if a BA unbooks a cell, it must inform its brothers. Therefore, a BA must process two kinds of constraints: intrinsic ones, which come from the actor its proxy RA represents, and induced ones, that come from its brother BAs. Of course, some problems may not have any solution without constraint relaxation. As a consequence, BAs must be able to affect priorities and weights to constraints as in fuzzy CSP or weighted CSP [2]. But, contrary to classical dynamic CSP [6], memory of previous states is sprayed within all the BAs which could be distributed within several servers. Finally, contrary to all these approaches, BAs only reason on a limited number of known BAs to find a good solution as in distributed CSP [16] or more accurately in distributed constraint optimization problems (DCOP) [12] that aim at optimizing (minimizing) the sum of relaxed constraints. Since BAs are agents, they do not have any global knowledge. Therefore, constraint optimization is shared by BAs, and the solution emerges from their local peer-to-peer interactions.

Nevertheless, our approach remains different from above-mentioned ones, because the main objective is not to provide an algorithm that is sound, complete, and terminates, but to define local and robust mechanisms, able to implement a global solving. Similarly to applications of ant algorithms on scheduling problems [15], BAs alter their environment (the grid) with markers to indicate the cells they book and to constrain the other agents. The main difference with the usage of pheromone is the way the markers disappear. In the ant approach, markers evaporate with time. In our algorithm, markers are removed consequently to ne-

gotiation between BAs in booking conflict (see section 3.4). Finally, this work is close to local search based CSP approaches such as [11], but by using the agent paradigm to encapsulate constraints.

3 Cooperative Self-organization Rules

The solving algorithm we propose is distributed among BAs and resides in cooperative self-organization. As said in the previous section, a nominal behavior of an agent is not sufficient to lead the collective to an adequate organization. BAs have to respect some cooperation rules to reach a correct global state (a pareto-optimal solution). Designing cooperative agents is then equivalent to implementing the cooperation meta-rules (see section 1). Five different situations for reorganization are identified. The two firsts do not respect the c_{dec} meta-rule of AMAS. The three next ones do not respect c_{act}. In this example, there is no c_{per} violation because all BA agents are identical and can understand each other.

The idea is to design these rules as *exceptions* in classical object programming, at the agent level and not at the instruction level. This concept really fits with the proscriptive approach proposed by [5]. As for exceptions, designers have to specify the condition of the exception throwing and the action to perform in the exception case. The following cooperation rules are then presented as condition-action pairs. Conditions are not exclusive. Nevertheless, a policy must be defined in the case of multiple NCS: from c_{dec} to c_{act}, for example.

3.1 Partnership Incompetence ($\neg c_{dec}$)

One of the goals a BA has to reach is to find a partner. If a BA ba_i encounters, in a cell, another BA ba_j it cannot partner with; ba_i is, using the AMAS terminology, incompetent [5]. For example, a BA representing a teacher's course meets another BA representing another teacher's course. As the only entity able to detect this partnership incompetence is the agent itself, this latter is the only one which changes the state of the organization by changing its position to encounter other more relevant BAs. Moreover, to enable a more efficient exploration of partnership possibilities, ba_i will memorize the location and the BAs known by ba_j for exchanging them during further encounters.

This cooperative self-organization rule can be summed up in the following table:

Name: **Partnership Incompetence** (for agent ba_i)
Condition: $\exists j(j \neq i \wedge knows(ba_i, ba_j) \wedge (\neg compatible(ba_i, ba_j) \vee (pCost(ba_i, ba_j) \geq pCost(ba_i, partnership(ba_i)))))$
Action: `memorize(ba`$_i$`,knows(ba`$_j$`));move`

The $pCost$ comparison allows ba_i to decide if the potential new partner ba_j is less constraining than the current one.

3.2 Reservation Incompetence ($\neg c_{dec}$)

In the same manner than partnership incompetence, BAs must be able to change organization when their reservations are not relevant. This reservation incompetence NCS occurs when a BA ba_i occupies a cell which constraints do not fit its own constraints. For example, a BA representing a teacher's course is in a cell representing a room with not enough seats for this course. Then ba_i must move to explore the reservation possibility space.

Name: **Reservation Incompetence** (for agent ba_i)

Condition: $\neg compatible(ba_i, cell(ba_i)) \vee (rCost(ba_i, cell(ba_i)) \geq rCost(ba_i, reservation(ba_i)))$

Action: `memorize(`ba_i`,cell(`ba_i`));move`

To improve the exploration of the grid, a BA memorizes the cells in which this NCS occurs to share it during negotiation or to avoid it when moving, like in a tabu search for example.

3.3 Partnership Conflict ($\neg c_{act}$)

Situations during which a BA wants to partner with another partnered BA may append. The agent must react to this partnership conflict by partnering or by moving. In this case, the cooperation is directly embedded within the resolution action: the partnership will be performed with the agent that has more difficulties to find partners (by comparing the $pCost$).

Name: **Partnership Conflict** (for Agent ba_i)

Condition: $\exists j \exists k (j \neq i \wedge i \neq k \wedge knows(ba_i, ba_j) \wedge compatible(ba_i, ba_j) \wedge partnership(ba_j) = ba_k)$

Action:

```
if (pCost(ba_i,ba_j) < pCost(ba_i,partnership(ba_i)))
   then partner(ba_i,ba_j)
   else move
```

When it partners, a BA must inform its previous partner and its proxy. Since this algorithm is distributed, the `partner(`ba_i`,`ba_j`)` action must be atomic (in the sense of critical section access), but is composed of the following instructions:

```
unpartner(ba_i,partnership(ba_i));
setPartner(ba_i,ba_j);
inform(partnership(ba_i));
inform(proxy(ba_i))
```

3.4 Reservation Conflict ($\neg c_{act}$)

As for partnership, reservation may lead to conflict: a BA wants to reserve an already booked cell. A reservation conflict can be specified as follows:

Name: **Reservation Conflict** (for Agent ba_i)

Condition: $\exists j(j \neq i \wedge (reservation(ba_j, cell(ba_i)) \vee \exists k(reservation(ba_j, c_k) \wedge slot(cell(ba_i)) = slot(c_k) \wedge proxy(ba_i) = proxy(ba_j))) \wedge compatible(ba_i, cell(ba_i))$

Action:

```
if (rCost(ba_i,cell(ba_i)) < rCost(ba_i,reservation(ba_i)))
   then book(ba_i,cell(ba_i))
   else move
```

When it books a cell, a BA must warn its previous partner and its proxy to inform not to book in the same time slot. Similarly to the `partner` action, the atomic `book(ba`$_i$`,cell(ba`$_i$`))` action is composed:

```
unbook(ba_i,reservation(ba_i));
setBook(ba_i,cell(ba_i));
inform(partnership(ba_i));
inform(proxy(ba_i))
```

3.5 Reservation Uselessness ($\neg c_{act}$)

In the case a BA is in the same cell than one of its *brothers*, reservation is useless. Therefore it can leave the cell without analyzing it, and its occupants, to find another more relevant one.

Name: **Reservation Uselessness** (for agent ba_i)

Condition: $cell(ba_i) = cell(partnership(ba_i))$

Action: `processEncounteredBAs();move`

Processing encountered BAs corresponds to verifying, in a limited memory list of BAs the agent has already encountered or another agent has shared during a negotiation, whether the agents can find a relevant partner with a minimum partnership cost.

4 Prototyping and Experiments

To validate the algorithm we propose, an ETTO prototype were developed and several tests carried out to underline the influence of cardinality, the benefit for

dynamic problem resolution and the robustness. Experiments are based on a French benchmark for the timetabling problem[3]. This requirements set is decomposed into four variants from simple problem solving without constraint relaxation to system openness by adding or removing agents and constraints at run time. For each of them, we proposed a solution – not unique in many cases.

4.1 Influence of Cardinality

Cooperation is a collective glue to enhance collective interactions. Therefore, it really becomes a relevant self-organization criterion in systems with a high cardinality. Figure 2 shows the evolution of solving time as a consequence of the growing number of BAs in the system. For these experiments, we keep the same exploration space size by increasing the number of cells in the grid proportionally to the number of agents. Only availability constraints are owned by teachers: one time slot per day is forbidden. Once the maximum reached (average 8 BAs), the number of cycles (during which every agent acts one time) decreases as the number of BAs increases. The time that varies the less is the real time. Therefore, it is the more relevant indicator of the solving time evolution. Beyond 32 BAs, it has a logarithmic curve. More BAs the system has, more efficient the solving is – if a solution exists.

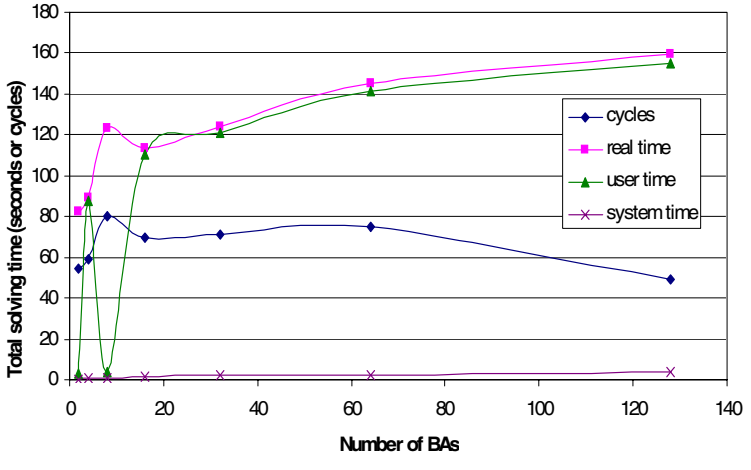


Fig. 2. Variation of solving time in terms of the number of BAs

4.2 Constraint Relaxation

In these experiments, agents must relax constraints to find a solution. Figure 3 shows the efficiency of ETTO solving, for a variant with 36 BAs requiring constraint relaxation. Reservations are set later than partnerships. ETTO found a solution with a constraint cost of 10 in 265 cycles. This cost represents the sum of all the constraints BAs had to relax, i.e. the sum of the weights of

[3] `http://www-poleia.lip6.fr/~guessoum/asa/BenchEmploi.pdf`

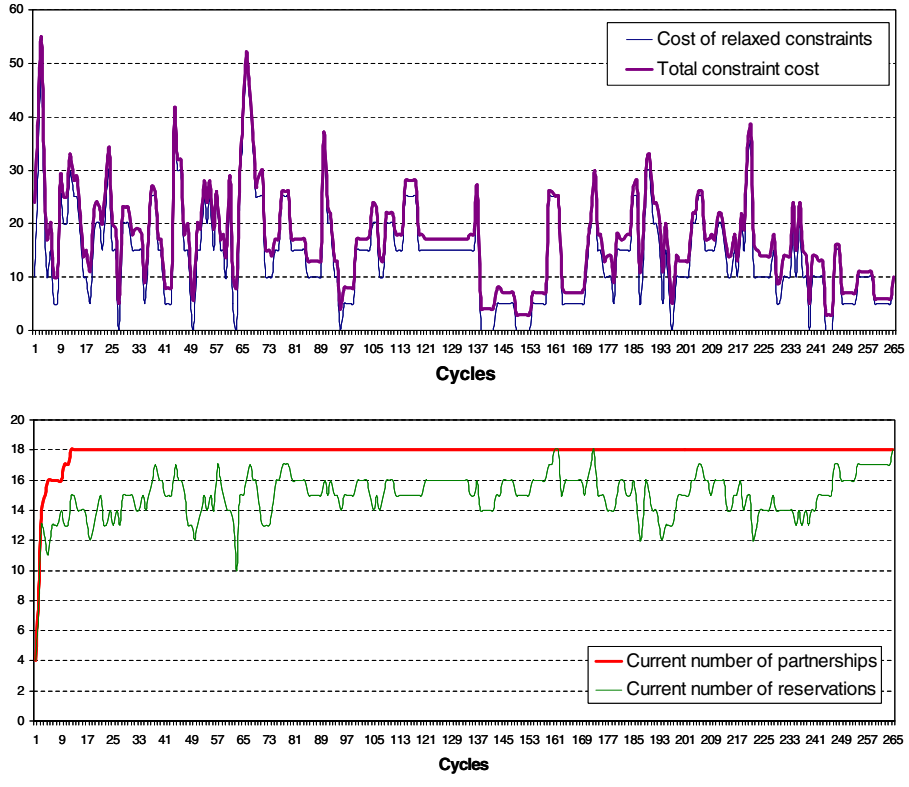


Fig. 3. Global constraint (*at top*) and partnership (*at bottom*) variations at run time for a solution requiring constraint relaxation

relaxed constraints. Nevertheless, the current prototype does not manage the cooperative slot sharing during negotiation and therefore, when a BA moves, it randomly chooses the next cell.

4.3 Dynamic Resolution

The two first experiments show the benefit of using cooperation to obtain an efficient timetable solving. A third serie of experiments tests the benefit in terms of robustness and dynamics. In these experiments, constraints become dynamic. Room or actors' availabilities may change at run time. Moreover, some agents can appear or disappear. By taking into account the chosen modeling, adding constraints is not different from adding agents that carry constraints. Figure 4 shows results on an experiment with initially 36 BAs. At cycle 364 – at the stabilization of the system – 8 BAs were removed, increasing the cost of relaxing constraints since each agent cannot find a relevant partner. 20 cycles later, 8 new agents with adequate constraints are plunged into the grid. The system only runs 7 cycles (from 384 to 391) to find an adequate organization with a null constraint cost.

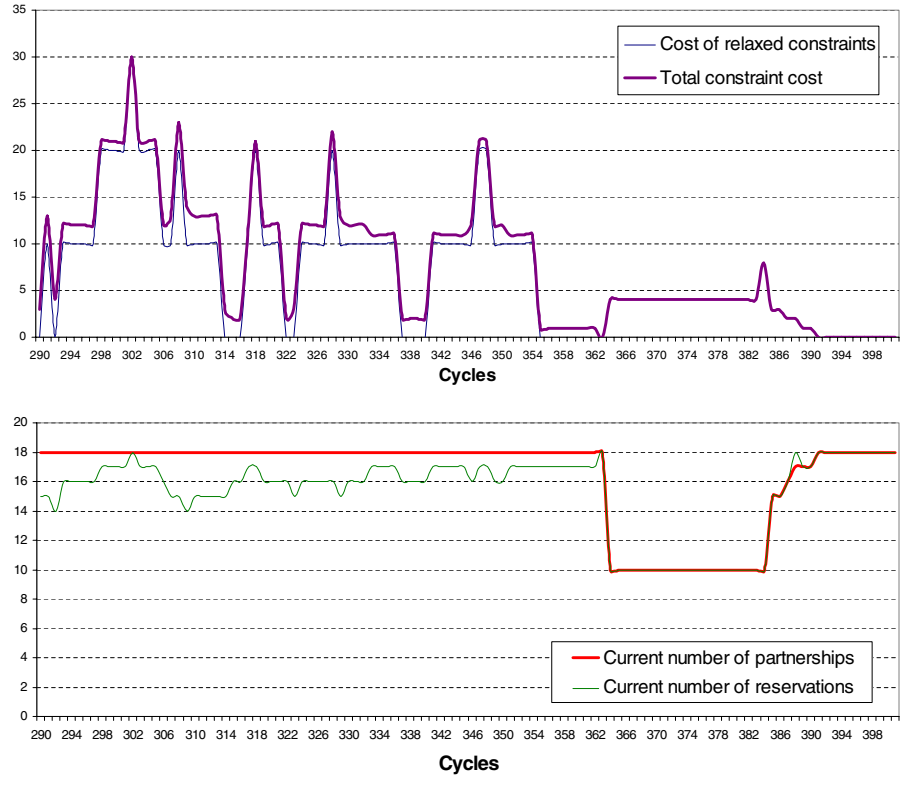


Fig. 4. Global constraint (*at top*) and partnership (*at bottom*) variations at run time with a removing of 8 agents after the system stabilization

5 Discussion

University timetabling problems in the real world are dynamic problems. Restarting from scratch each time a constraint is modified (added, removed) would not be efficient and some works are interested in this problem. Usually, the main objective is to have the smallest impact possible on the current solution as in [13] in which this is done by introducing a new search algorithm that limits the number of additional perturbations. In [4], explanations are used as well to handle dynamic problems, especially, new operators are given to repropagate once a constraint removed and its past effects undone. In ETTO, as soon as a constraint is added or removed for an agent, this latter questions its reservations and its possible partnership; if it judges that they are inconsistent with its new state, it tries to find new ones by roaming the grid and applying its usual behaviour. If a new agent is added, it immediately begins searching for a partnership and if it is removed, then all its reservations and constraints are deleted from the system and its possible partner warned.

The main feature of ETTO is that modifications are then done *without stopping the search* for a solution while this latter is in progress, unexpected events are processed while actors are changing their constraints. Furthermore, this ability to insert agents has enabled us to show that adding supernumerary agents helps finding a solution and gives better results. This can be explained by the fact that the added agents can disrupt others which are satisfied with a solution that could be optimized. The main difference between our approach and the DCOP one [12], which considers the distributed optimization of the constraint cost, is the fact that the set of agents (BAs and RAs) are not totally ordered. RAs send constraints only to their child BAs and BAs send update data to only their father RA.

But ETTO has also weaknesses. For example, processing over-constrained problems is not fully efficient because even if agents have found a solution, they *continue to explore the grid* to find more relevant solutions. As agents have a *limited view* of the environment, they cannot take into account the global constraint cost to stop the exploration. To use ETTO, we consider it exists an oracle (a human manager) who will halt the solving process when the organization fits his requirements – a constraint minimum level, for example. The search for a cell in the grid is not efficient either because it is *randomly* made by an agent. For the time being, we were not interested by efficiency, we just wanted to show that our approach by self-organization is *feasible* and can produce positive results as it has been shown. The main positive result is that nothing in the behavior of agents makes assumption on how the solution timetable is obtained. This solution solely *emerges* from their local interactions. Nevertheless, a future step would be to enhance this search by *adding a limited memory* to agents concerning, for instance, the cells they visited in the past. We will also try other self-organizing mechanisms like the T-Man approach, which has been presented in the ESOA'05 Workshop [10].

Two problems are generally considered in most of the CSP approaches: a search problem in which a timetable that satisfies all the (hard/soft) constraints is first found and, then, an optimization problem which consists in minimizing an objective function that takes soft constraints into account. As the solving in ETTO is distributed among the agents, a global view of the solution does not exist and giving a global objective function is not possible. Therefore, optimization problems cannot be tackled in a global way. But, cooperation seems a relevant criterion anyway. In fact, we had applied this approach to more simple CSP (N queens, $N^2/2$ knights, etc) for which cooperation quickly provides solutions if they exist. In problems with no solution, systems quickly reach stable states with minimal constraint violation [14].

6 Conclusion

We think that the inherent distributed aspect of the timetabling problem explains a processing by a MAS. We proposed then a solution based on adaptive multi-agent systems in which cooperation is a local criterion for agents to change their interactions with others and to make the global, a priori unknown, func-

tion emerge from these interactions. This kind of programming can be efficient enough to solve problems that are complex, not well or incompletely specified and for which the designer does not possess a predefined algorithm. This is shown by the preliminary results obtained by ETTO as well as other previous works done in the solving problem domain.

We chose a rather simple example to apply ETTO, one perspective is to apply it to a more realistic timetabling problem or to a benchmark such as the one given by the *Metaheuritics Network*[4]. This will allow us to compare our results with other approaches such as genetic algorithms or simulated annealing.

References

1. S. Ali, R. Zimmer, and C. Elstob. The question concerning emergence : Implication for Artificiality. In Dubois, D.M., editor, *1st CASYS'97 Conference*, 1997.
2. S. Bistarelli, H. Fargier, U. Mantanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based constraints CSPs and valued CSPs: frameworks, properties, and comparison. *Constraints: an International Journal*, 4(3):199–240, 1999.
3. E. Bonabeau, G. Theraulaz, J.-L. Deneubourg, S. Aron, and S. Camazine. Self-Organization in Social Insects. *Trends in Ecology and Evolution*, 12:188 –193, 1997.
4. H. Cambazard, F. Demazeau, N. Jussien, and P. David. Interactively Solving School Timetabling Problems using Extensions of Constraint Programming. In *Proc. of the 5th International Conference of the Practice and Theory of Automated Timetabling (PATAT), Pittsburg, USA*, 2004.
5. D. Capera, G. JP., M.-P. Gleizes, and P. Glize. The AMAS theory for complex problem solving based on self-organizing cooperative agents. In *1st International TAPOCS Workshop at IEEE 12th WETICE*, pages 383 –388. IEEE, 2003.
6. Dechter, Meiri, and Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
7. J.-P. Georgé, B. Edmonds, and P. Glize. Making Self-Organising Adaptive Multiagent Systems Work. In *Methodologies and Software Engineering for Agent Systems (Chapter 16)*, pages 321–340. Kluwer, 2004.
8. J. Goldstein. Emergence as a Construct : History and Issues. *Journal of Complexity Issues in Organizations and Management*, 1(1), 1999.
9. T. Kohonen. *Self-Organising Maps*. Springer-Verlag, 2001.
10. J. M. and O. Babaoglu. T-Man: Gossip-based Overlay Topology Management. In *Engineering Self-Organizing Applications – Third International Workshop (ESOA) at the Fourth International Joint Conference on Autonomous Agents and Multi-Agents Systems (AAMAS'05), July 2005, Utrecht, Netherlands*, 2005.
11. S. Minton, M. Johnston, P. A., and P. Laird. Minimizing Conflicts: a Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence*, 58:160–205, 1992.
12. P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. An Asynchronous Complete Method for Distributed Constraint Optimization. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, pages 161–168, 2003.

[4] `http://www.metaheuristics.org/`

13. T. Müller and H. Rudova. Minimal Perturbation Problem in Course Timetabling. In *Proc. of the 5th International Conference of the Practice and Theory of Automated Timetabling (PATAT), Pittsburg, USA*, 2004.
14. G. Picard and P. Glize. Model and Experiments of Local Decision Based on Cooperative Self-Organization. In *2nd International Indian Conference on Artificial Intelligence (IICAI'05), 20-22 December 2005, Pune, India*, 2005.
15. K. Socha, J. Knowles, and M. Sampels. A MAX-MIN Ant System for the University Timetabling Problem. In *Proceedings of 3rd International Workshop on Ant Algorithms, ANTS'02*, volume 2463 of *LNCS*, pages 1 –13. Springer-Verlag, 2002.
16. M. Yokoo, E. Durfee, Y. Ishida, and K. Kubawara. The Distributed Constraint Satisfaction Problem : Formalization and Algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10:673–685, 1998.

Self-adaptation and Dynamic Environment Experiments with Evolvable Virtual Machines

Mariusz Nowostawski[1], Lucien Epiney[2], and Martin Purvis[1]

[1] Department Information Science,
University of Otago, Dunedin, New Zealand
{mnowostawski, mpurvis}@infoscience.otago.ac.nz
[2] Swiss Federal Institute of Technology,
EPFL, Lausanne
lucien.epiney@epfl.ch

Abstract. Increasing complexity of software applications forces researchers to look for automated ways of programming and adapting these systems. Self-adapting, self-organising software system is one of the possible ways to tackle and manage higher complexity. A set of small independent problem solvers, working together in a dynamic environment, solving multiple tasks, and dynamically adapting to changing requirements is one way of achieving true self-adaptation in software systems. Our work presents a dynamic multi-task environment and experiments with a self-adapting software system. The Evolvable Virtual Machine (EVM) architecture is a model for building complex hierarchically organised software systems. The intrinsic properties of EVM allow the independent programs to evolve into higher levels of complexity, in a way analogous to multi-level, or hierarchical evolutionary processes. The EVM is designed to evolve structures of self-maintaining, self-adapting ensembles, that are open-ended and hierarchically organised. This article discusses the EVM architecture together with different statistical exploration methods that can be used with it. Based on experimental results, certain behaviours that exhibit self-adaptation in the EVM system are discussed.

1 Introduction

Existing evolutionary computation techniques, such as genetic programming (GP) [3], linear genetic algorithms (GAs) [13], and others [2], have proved to be successful in a broad range of optimisation problems and applications. These methods, however, are operating on a predefined, fixed fitness landscape and therefore are very difficult or even impossible to be used in multi-task dynamical environments. In this article we propose a new model of evolutionary computation that can be used in highly dynamic environments. Moreover, our model can be used with traditional linear GA evolutionary learning, with random search, and with many other stochastic search methods. Our framework consists of a set of independent computing cells that compete for limited resources. These computing cells are able to dynamically change their functionality and functional dependency to meet changes in their environment. They form a web of interacting computational agents that exhibit self-organisation and self-adaptability without direct user interaction.

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 46–60, 2006.

2 Computation and Biological Inspirations

Current research in EC emphasises information-centric methods that are inspired by Darwinian theory of random mutations and natural selection. This is visible in well-established computational optimisation methods, such as genetic algorithms (GA), genetic programming (GP), and their variations, such as assorted artificial life systems. Despite some successes, the typical simple single-layer evolutionary systems based on random mutation and selection have been shown to be insufficient (in principle) to produce an open-ended evolutionary process with potential multiple levels of genetic material translation [1, 17].

The Evolvable Virtual Machine architecture (EVM) is a novel model for building complex hierarchically organised software systems. In this article we describe how the original abstract EVM model [9] has been extended by the elements of symbiogenesis, that allow independent computing elements to engage in symbiotic relationships.

From the biological perspective, the abstract EVM model is primarily based on Darwin's principle of natural selection, which is widely used in current computational models of evolutionary systems for optimisation or simulation purposes, and in evolutionary computation (EC) in general. Some authors regard natural selection as axiomatic, but this assumption is not necessary. Natural selection is simply a consequence of the properties of population dynamics subjected to specified external constraints [1].

Our work proposes an evolutionary model inspired by the theory of hypercycles [1], autopoiesis [7], and symbiogenesis [8]. A system which connects autocatalytic or self-replicative units through a cyclic linkage is called a *hypercycle*. Compared with a simple autocatalyst or a self-replicative unit (which we can consider here to be a "flat" structure) a hypercycle is self-reproductive to a higher degree. This is because each of the intermediaries can itself be an autocatalytic cycle.

The EVM system consists of an interconnected network of processing units (cells or agents) that can only interact with their neighbours. These processing units are autonomous, they do not have pre-assigned functions, can specialise in different tasks and can utilise the processing structures of their neighbours. Initially, each single processing unit potentially benefits from Universal Turing Machine (UTM) equivalent computing capabilities. In time, some of the cells can specialise in tasks requiring different (lower) computing power, i.e. specialisation of the virtual machines occurs. On the other hand, each processing unit can make use of other processing units, via a symbiosis-like relationship, thus creating a web of interconnected machines.

2.1 Symbiogenesis and Specialisation

Proponents of symbiogenesis argue that symbiosis is a primary source of biological variation, and that acquisition and accumulation of random mutations alone are not sufficient to develop high levels of complexity [5, 6]. K. Mereschkowsky [8] and I. Wallin [14] were the first to propose that independent organisms merge (spontaneously) to form composites (new cell organelles, new organs, species, etc). For example, important organelles, such as plastid or mitochondria, are thought to have evolved from an endosymbiosis between a Gram-negative bacterium and a pre-eukaryotic cell.

Another (less speculative) phenomenon that occurs at all levels of biological organisation from molecules to populations, is specialisation. It is the process of setting apart a particular sub-system (reducing its complexity) for the purpose of better performance and/or efficiency of a particular function. Our working hypothesis is that specialisation, together with symbiosis, is necessary to reach higher complexity levels.

Some recent work in incremental reinforcement learning methods also advocate the retention of learnt structures (or learnt information) [10]. The sub-structures developed or acquired during the course of the program self-improvement process are retained in the program data-structures. It is therefore surprising that this general procedure has not been exhibited by any of the (standard) evolutionary programming models, such as GP or GAs [13]. Although these evolutionary programming models are inspired by biological evolution, they do not share some significant aspects that are recognised in current evolutionary biology, neither can they be used (directly) in an incremental self-improvement fashion[1].

2.2 Abstract Self-organising Application Architecture

Simplified, the model presented here can be depicted schematically as in Figure 1, with inputs and outputs connected to the external environment. The environment consists of a set of current tasks to be solved, with an appropriate resource pool associated with each task. The environment also keeps track (via the tasks) of the current resource utilisation for each given task. The user can dynamically plug-in new tasks, and remove existing ones, at will. In Figure 1 the resource marked as `R1`, is utilised by 5 computational cells (cells with same gray hue on the grid). Similarly, the other two resources `R2` and `R3` are utilised by cells indicated by other matched shading types.

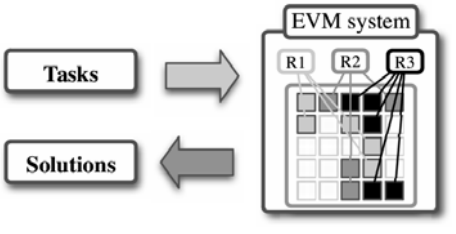


Fig. 1. EVM system from an end user point of view

3 EVM Assembly

The programming language used for search in EC plays an important role. Some languages are particularly suited for some, but not for all, problems. An appealing aspect

[1] GA and GP maintain some developed substructures during the course of evolution towards a particular solution, however, as soon as the equilibrium is reached, or the optimal solution found, all the intermediate substructures are quickly "forgotten". When trying to learn a new task, the search process must start from scratch again, and the common practice in the application of GA/GP is to restart the search from a random population. In contrast, self-improvement in multitask environment never restarts the search from a random population for new tasks. The idea is to maintain and reuse previously evolved structures.

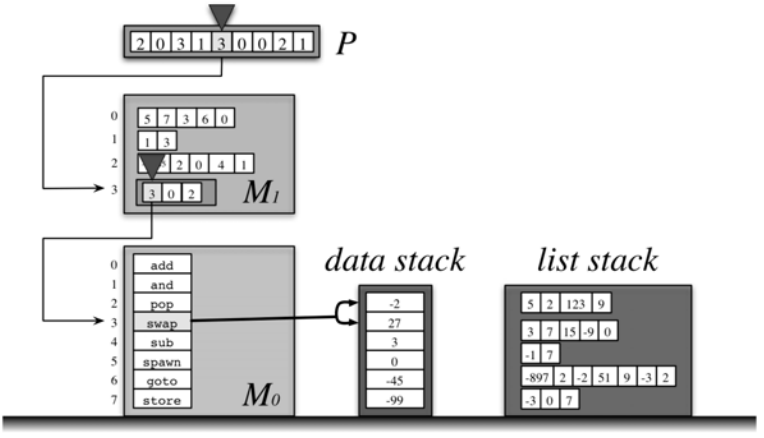


Fig. 2. Schematic representation of two-level machine execution. The execution frame is executing the 5^{th} instruction of program P, that calls program 3 of machine M_1. The first instruction of the 3^{rd} program of M_1 is the primitive instruction `swap` that will swap the two topmost elements of the data stack.

of a multi-level search process is that, in principle, it is possible to specify a new base level and a new programming language that is specialised for a given task at that level. We want the EVM to exploit this property.

In our work we deal with programs capable of universal computation (e.g. with loops, recursion, etc.). In other words, the virtual machine running our programs must be Universal Turing-machine equivalent.

None of the existing languages used in EC provides mechanisms to manipulate machine levels – a property needed for our EVM implementation. There are other features we would like the base machine language to possess that none of the existing languages have. For example, it is desirable that a language be capable of redefining itself. Thus the primitive instruction set must allow the evolutionary process to restructure and redefine that set. We would also like to have a programming language that is highly expressive, that is, we want solution programs to typically encountered tasks to be expressible in compact form. Moreover, we believe that there are efficiency and expressibility advantages for a language with solution spaces that are highly recursive.

Some existing languages possess some of these desired properties, but no single one possesses all of them. For this reason we have designed our own specialised programming language, called the EVM assembly. The principal objective of this programming language is to facilitate searches for languages specialised for a given set of problems.

4 EVM Implementation

Our current implementation of the EVM architecture is based on a stack-machine, such as Forth, or the Java Virtual Machine (JVM). In fact, with small differences, it is comparable to an integer-based subset of the JVM. The implementation is written entirely in Java, and developers can obtain it from CVS [2].

[2] `http://www.sf.net/projects/cirrus`

The basic data unit for processing in our current implementation is a 32-bit signed integer. The basic input/output and argument-passing capabilities are provided by the operand stack, called here *the data stack*, or for short *the stack*. The data stack is a normal integer stack, just as in a JVM for example. All the operands for all the instructions are passed via the stack. The only exception is the instruction `push`, which takes its operand from the *program* itself. Unlike the JVM, our virtual machine does not provide any operations for creating and manipulating arrays. Instead, EVM facilitates operations on lists. There is a special stack, called *the list stack* for storing integer-based lists.

Execution frames are managed in a similar way to the JVM, via a special execution frames stack. There is a lower-level machine handle attached to each of the execution frames. This is a list of lists, where each individual list represents an implementation of a single instruction for the given machine. In other words, the machine is a list of lists of instructions, each of which implements a given machine instruction. Of course, if the given instruction is not one of the Base Machine units (primitive instructions for that machine), the sequence must be executed on another lower-level machine. The Base Machine implements all the primitive instructions that are not reified further into more primitive units.

Potentially, EVM programs can run indefinitely and therefore each thread of execution has an instruction time limit to constrain the time of each program in a multi-EVM environment. Each execution thread (a single program) has a maximum number of primitive instructions that it can execute. Once the limit is reached, the program unconditionally halts.

The EVM offers unrestricted reflection and reification mechanisms. The computing model is relatively fixed at the lowest-level, but it does provide the user with multiple computing architectures to choose from. The model allows the programs to reify the virtual machine on the lowest level. For example, programs are free to modify, add, and remove instructions from or to the lowest level virtual machine. Also, programs can construct higher-level machines and execute themselves on these newly created levels. In addition, a running program can switch the context of the machine, to execute some commands on the lower-level, or on the higher-level machine. All together it provides near limitless flexibility and capabilities for reifying EVMexecution.

4.1 Extensions

One possible way of extending current EVM implementation is by adopting bias-optimal search primitives [4], or the incremental search methods [11]. To narrow the search, one can combine several methods, for example it is possible to construct a generator of problem solver generators, and employ multiple meta-learning strategies. A more detailed description of the abstract EVM architecture is given elswhere [9].

5 Specialisation of an Individual Machine

Given an initial set of instructions I, the specialisation mechanism will aim to find programs from I^* (the set of all possible sequences of instructions) that solve tasks defined by the environment. We have tried, independently, three different search methods for our EVM model: 1) random search; 2) GA with variable lengths of chromosomes; and 3) stochastic search based on a probability distribution of individual instructions.

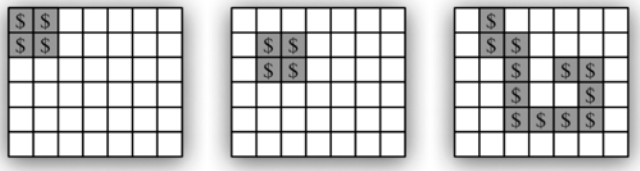


Fig. 3. Grid configuration for experiments 1, 2, and 3 respectively (the actual size of the grid is 100x100 instead of 7x7). Dark cells contain rewards.

For preliminary testing of these different search methods we used programs that control an agent moving on a two-dimensional discrete grid. The cells on the grid may contain rewards. Grids for experiments 1, 2, and 3 are depicted in Figure 3. In experiments 1 and 2 the rewards are persistent on the grid. In experiment 3, the agent can obtain each reward only once from a given cell (i.e. volatile rewards). This constraint has been added to force the agent to follow a path. The initial position of the agent is always the top left corner. Execution time is limited to 100 time steps for the first two experiments. This means, the ideal program for experiment 1 would be 100 instructions long, and will collect 100 reward units. For experiment 2, the perfect program would be again 100 instructions long, and would collect 99 reward units. Note, that there is a trade-off between the total number of rewards and the length of the program. For a program of length 6 that utilizes loop, the total number of rewards for experiment 2 is 66. For third experiment the time limit is set to 12 time steps and the total number of possible reward units collected is 12. The base instruction set has been extended with 4 special instructions: `down`, `up`, `right`, and `left` to move the agent on the grid.

5.1 Specialisation with the Use of Random Search

Random search is the simplest mechanism to specialise a machine. For each single cell it generates random programs. If one of the programs is successful, it will stay; and if it is not successful, then a new program will be randomly generated and tested[3]. On one hand, random search cannot take advantage of regularities in the fitness landscape. But on the other hand, it has no parameters, is fast, and needs very little memory.

For arithmetical problems, the landscape basically consists of one big peak with a steep slope (rewards are either all or nothing). In such circumstances, random search is appropriate and performs quite well when compared with other methods. Moreover, by implementing the simplest possible search mechanism for every cell, it is possible to focus on macroscopic behavioural patterns, i.e. how cells interact to compute a complex solution.

[3] In our implementation this is implemented by the following algorithm. First step: create a machine with a randomly selected program. On request return that program, and if that program received a reward, *freeze* it, i.e. always return the same program to the evaluator. If the program solves a task, it will be rewarded and its cumulative rewards will be higher than zero. If the program does not solves the task, subtract a fixed amount from the cumulative rewards. Return the same *frozen* program for each request, until the frozen program cumulative rewards are zero, and the program is starving. Then, go back to the first step, and create different random program.

In the general case, though, most problems (like the maze experiments) display regularities in the fitness landscape. For that reason, we seek more complex search mechanisms that can take advantage of these regularities.

5.2 Specialisation with the Use of Genetic Algorihtms

For our implementation of the genetic algorithm (GA) module, chromosomes are represented as lists of integers. There is an extra mutation operator that inserts or deletes individual instructions into the program, and for all the experimental runs the probabilities of addition and removal were set to the same rate. The initial size of the program, as well as all the other mutation and crossover probabilities, varied, so it was possible to hand-tune the parameters for a given problem to achieve satisfactory GA performance. For experiment 1 we used a population size of 1000 and up to 2000 generations, with the probability of crossover set at 0.8 and the mutation rate set at 0.01 (the add/remove probability was set at 0.05). The GA-based search did not have any trouble finding suboptimal solutions that utilize loops, but it was unable to find a global optimum.

Trade-off between exploration and exploitation. The main drawback of the GA-based search was that it does not dynamically adapt to different exploration strategies for different environments. Exploration in GAs is basically performed by the mutation operation. For example, in experiment 2, often a single run of GA-based search was unable to find any rewarded sequence for 3000 generations with 1000 individuals. Thus, the success of the GA run depended purely on the initial randomly generated population. If it was initialized without any good *building blocks*, the population was unable to discover any useful subsequence purely by mutations alone[4].

Converging to suboptimal solutions. The GA-based search always prefers shorter solutions to the longer ones, because shorter ones are statistically more stable. The heavy use of loops prevents GA-based search from finding global optima (apart from experiment 3, where GA-based search in half of the runs was able to find the global solution).

Stable prefix and bloat of introns at the tail. GA-based search generated solutions that are characterized by a relatively stable prefix, and a very long chaotic tail of introns. Introns are simply instructions that are either not executed at all, or, when executed do not have any negative or positive side-effects for the program. Again, exploration of the program space tends to concentrate on the end of the program structure. The closer to the beginning, the more stable the instruction become. This is very similar to the "freezing" mechanism discussed in the next section in regard to stochastic search.

5.3 Specialisation with the Use of Stochastic Search

For the stochastic search we assumed that the number of instructions per program, and the number of programs per machine were limited. The basic idea is that by limiting the size of the machine we can assign a probability distribution to each instruction of the machine M. For each instruction of M, there is a probability distribution over its possible values (Figure 4).

[4] The probability of such an event is around 0.0000000164: there are 78 instructions in total, and the minimal length of a rewarded program is 2.

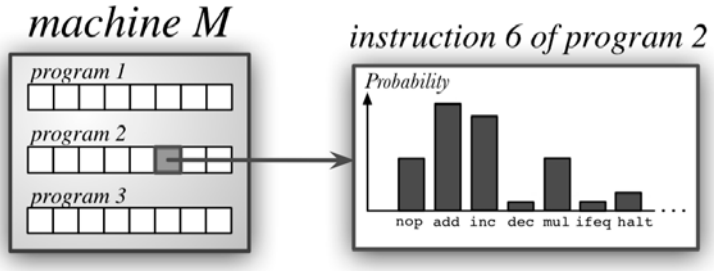


Fig. 4. Every instruction of a machine M contains a probability distribution over its possible values

To evaluate a machine, specific values are randomly picked representing the instructions of M according to their probability distributions. The result is an instance m of M, which will be used in an attempt to solve a problem. Programs of m will then be executed until a solution is found. Depending on their success, probabilities of programs of m will be increased or decreased. Every time a reward r is gained with a program p, the probabilities of p's instructions are increased. On the other hand, if p was not successful (implying no reward), the probabilities of its instructions were decreased slightly.

In addition to finding the solution to a particular problem, this mechanism is aimed at *remembering* solutions. The intent is to store these solutions in the machine's list of programs. For instance, a machine M can specialise in solving arithmetical operations. The environment provides several arithmetical problems. M will more or less randomly explore the space of possible programs until it starts finding the first solutions. Then by solving the same tasks again and again, probabilities will increase, and solutions will progressively be stored in M's programs.

Correlation between instructions. The primary drawback of the stochastic approach (and to a certain extent GAs, as well) is that it does not take into account correlations between instructions. The program's measure of quality highly depends on all its instructions taken together. Changing one of them may disrupt the entire program and penalize the other instructions, even if these are likely to be beneficial. Storing conditional probabilities might be a potential solution to that problem. Ideally, we could store successful subprograms' *patterns* of any length in a probability tree. It would then be equivalent to a Markov-chain based stochastic search. Experiments are currently being performed to evaluate this approach.

Freezing mechanism. Since instructions are randomly selected according to their probability distributions, there is always a probability of not selecting an important instruction i. To be sure i is in the program, the search mechanism tends to increase its probability in consecutive placeholders as well. Therefore, the search wound up confined to local optima such as

`right down right down down right down` (3 reward units)

for experiments 1 and 2. This is especially disappointing, because during the evolutionary process, the search mechanism does find some good solutions (up to 66 reward units in the 2nd experiment) but fails to remember them. The search process can be

enhanced by introducing a *freezing mechanism* that will progressively "freeze" values in a program's instructions according to the following quasi-algorithm:

1. Assign $n \leftarrow 1$ (start from the first instruction)
2. If the probability of the n-th instruction has been almost maximal (within predefined threshold) for the number of iterations (another predefined constant), set the probability to 1 and don't modify it anymore (*freeze* it). Set $n \leftarrow n + 1$
3. Reset all the probabilities of the tail of the program, i.e. for all the instructions $n+1$ and above.
4. $n \leftarrow n + 1$ and return to step 2 until the entire program has been frozen.

The freezing mechanism has proved to be highly effective in many different problems we have tried, and it has always outperformed the stochastic search without freezing. In experiments 1 and 2, indeed, the search converges to solutions containing a loop, either with 2 or 4 agent movements instructions inside it (the more movement instructions, the higher the total reward). For instance, the second experiment produces these solutions:

`down right right left const_2 goto` (50 reward units)
`down right right down left up const_2 goto` (66 reward units)

The first program moves the agent: down, right, right, then left. After that sequence, a value of 2 is placed on top of the stack (instruction `const_2`, and instruction `goto` moves back to the third instruction in the program (`right`). The sequence of agent movements: `right`, `left` is executed indefinitely in the loop.

Long programs. Even with the help of the freezing mechanism, it becomes very difficult to build up longer programs, especially when a shorter (yet less rewarded) one is possible. The third experiment demonstrates this. Instead of finding the better 12-instructions-long program, all of our search methods are more likely to attempt to find a pattern to insert in a loop, e.g.: `right down right down down const_0 goto` (7 reward units). Since this program is shorter, it is more likely to be picked. It gets smaller rewards than the best solution, but it gets them more often, and for this reason it dominates. Again, this issue may be corrected by a probability tree.

6 Experiments with a Web of Interacting Agents

Suppose that several machines lie on an n-dimensional grid (all executed asynchronously). In addition to primitive instructions (like `add`, `swap`,...), a machine can also access its neighbours' programs. There is no specific constraint on the grid's topology. It can be n-dimensional, and the neighbourhood can be as big as desired. In the extreme case, we could imagine a grid or web in which every machine can access every other machine. So far we have run experiments with two-dimensional grids with four neighbours, but other experiments with different topologies are planned to investigate further the impact of neighbourhood relation and locality. A program now can look like the following: `add dup` *p*rogram2ofLeftNeighbour `mul` *p*rogram1ofRightNeighbour.

Moreover, if a program gets a reward, it will share it with any neighbour's program used to compute the solution. Both of them will benefit from their relationship. In other

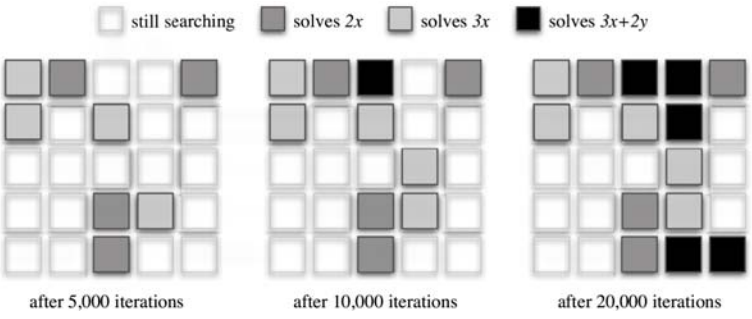


Fig. 5. Typical run exhibiting self assembly. After 5,000 iterations, several machines can solve the two simple tasks ($2x$ and $3x$). After 10,000 iterations, one machine ($m_{1,3}$) uses its neighbours to solve the hard task ($3x + 2y$), and all of the machines share the rewards (symbiosis). Shortly afterwards, some of its neighbours take advantage of it: they simply solve the task by calling $m_{1,3}$ (parasitism). Finally, after 20,000 iterations, we can observe another cluster of solutions at the bottom of the grid.

words, symbiotic relationships may appear between programs. This ability to access neighbours' programs has thus opened the door to complex hierarchical organization. As a consequence, machines are now able to collaborate to solve complex problems.

Let us consider the following example. On the grid are some machines (agents) specialised in list manipulation, and some specialised in recursion problems. The task is now to solve a problem involving both list manipulation and recursion. Cells surrounded by machines specialising for these two problems have the opportunity to find solutions to the task by accessing programs of their neighbours. Some machines will specialise in solving these simpler tasks and might be used by their neighbours to solve a more difficult task. A simple case, with two simple arithmetic tasks ($2x$ and $3x$) and one more complicated ($3x + 2y$), is depicted in Figure 5.

This computational model is well-suited for incremental problem solving. If solutions (or some part thereof) to some tasks can be reused to solve more complex ones, the EVM will take advantage of it. Indeed in such conditions, useful neighbours are more likely to appear.

6.1 Environment

From a machine learning point of view, the environment consists of a set of tasks to be solved (*multitask learning*). For the search process to be efficient, the model should fulfill the following requirements:

1. All tasks must be solved (eventually).
2. Solving difficult tasks should lead to greater rewards than easy ones.
3. Computational resources (i.e. agents not currently solving any task) should focus on unsolved tasks.
4. Solutions must not be forgotten, as long as they are useful.
5. Knowledge diffusion across the web of interacting machines should be facilitated.
6. Dynamic environments should be supported: tasks can be added and/or removed at any time, dynamically.

From an artificial life point of view, one can view the environment as having to manage food *resources*, that are to be dispatched to the agents trying to consume them. Every resource represents a task to be solved. Every resource has two attributes: *quantity* and *quality*. Values for these attributes specify how much food (reward) will be given to the cell that consumes from the given resource.

The parameter $QUANTITY$ (capitalized to highlight its static nature) represents the abundance of resources in the environment. This value is set as an *a priori* conjecture by the modeller and is the same for each of the resources. It allows us to tune the amount of agents that will be able to survive.

The resource's quality has to reflect how difficult a task is. It facilitates a mechanism to give more rewards for hard tasks (requirement 2). There are several ways of measuring the difficulty of a task. Some are *a priori* (using expert knowledge), but it is more interesting to adjust it dynamically based on the observed difficulty. For example, the resource's quality may be set based on the observed average time it takes to solve it, or on how many agents can solve it, etc. We decided to set the resource's quality to the current minimal number of cells required to solve the task. It will reflect dynamically the task's complexity without depending on randomness and without the use of extra parameter that would need to be tuned for the search process.

When a cell consumes a resource, it gets the following amount of food:

$$food = \frac{QUANTITY\ quality}{consumers},$$

where $consumers$ is the number of agents eating the resource. Moreover, a cell has to share its food with all the neighbours it used to solve the task. Every cell used will get the same share of food [5].

At every iteration, a cell needs to eat a certain amount of food: F_{NEEDED}. If it eats more, it can makes provisions by storing it. On the other hand, if it eats less it will die from starvation once its provisions are empty.

$$provision_t = provision_{t-1} + food - F_{NEEDED}$$

6.2 Parameters and Their Impact

The two main parameters: the resource's quantity and the food needed for a cell to survive can be represented as one parameter $DENSITY$.

$$DENSITY = \frac{QUANTITY}{F_{NEEDED}\ SIZE},$$

where $SIZE$ is the total number of cells. This simplifies the model, because only the respective ratio is really important. $DENSITY$ controls the utilisation of the cells on the web. Figure 6 depicts two different settings for that parameter.

Equilibrum/stability. Another parameter, $PROVISION_{MAX}$, has been added. It sets a maximal bound for provisions stored by a cell. Its value drastically affects the dynamism of the web. If $PROVISION_{MAX}$ is high, most of the cells are stable and

[5] For these early experiments, we have chosen a very simple reward mechanism. More complicated models will be investigated in our future work.

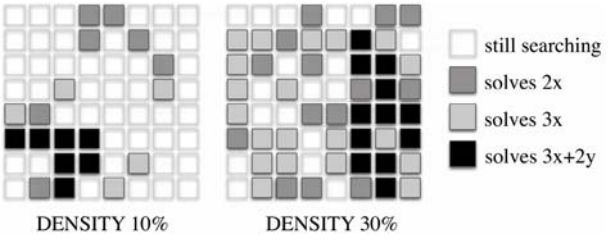


Fig. 6. Two different settings for the $DENSITY$ parameter (left: 10%; right: 30%)

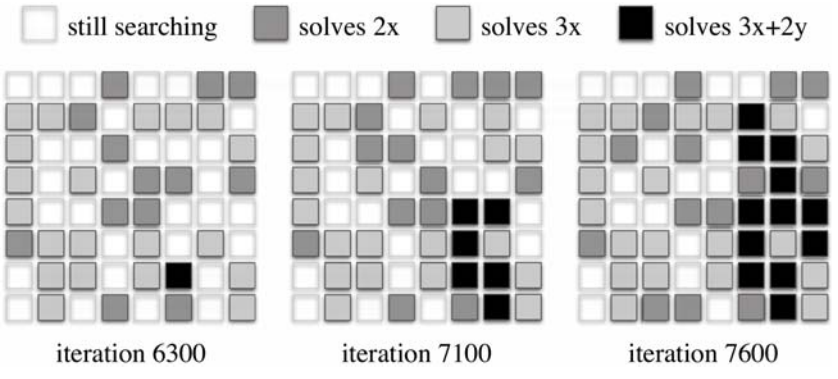


Fig. 7. Knowledge diffusion

only a few appear and disappear (scenario A). If $PROVISION_{MAX}$ is low, we observe much more dynamic structural patterns on the web, with cyclic episodes similar to a kind of *catastrophe* scenario [12]. Good solutions spontaneously appear in the web, and after a while there are too many cells competing for the same resource. As a consequence, the quantity of the resource they are consuming decrease below the F_{NEEDED} threshold. Since they don't have enough provisions, they will soon almost all disappear. New cells can then start a new cycle (scenario B).

There seems to be no smooth transition between these two dramatically different scenarios. Scenario A represents a stable and fixed solid state, similar to Wolfram's class 1 of cellular automata (CA) classification [15]. Scenario B represents a cyclic state, and is similar to Wolfram's CA class 2. Wolfram's Classes 3 and 4 can be achieved by tuning the F_{NEEDED} parameter.

Knowledge diffusion. There is another interesting behaviour of interacting machines that can be observed. When a cell C_s solves a difficult task for the first time, the solution is almost immediately parasited by its neighbours. That phenomenon (see Figure 7) enables to diffuse a solution around the successful cell C_s, thus rendering this solution accessible to an increasing number of agents. Since some agents may need this solution to compute a more difficult problem, knowledge diffusion is highly desirable. Competition between parasites is very intense. They usually appear, survive a

couple of iterations, disappear, after a while appear again, and so on. The dynamism exhibited looks like C_s is trying to reach something in its neighbourhood. For instance, if the diffusion manages to reach the neighbourhood of a cell C_1 that needs it, it will be used and thus the whole chain of parasites from C_s to C_1 will receive a lot of rewards and survive.

Once some other cells in the web solve the same task as C_s by their own (without parasiting), it becomes more and more difficult for the parasites of C_s to survive (as they always have to share their food with the cells they use). As a consequence, knowledge diffusion will progressively decrease.

7 Film Analysis vs. Quantitative Studies

The study so far was mainly based on running simulations and preparing (off-line) the 2D movies (animation sequences) of the dynamics of the evolving web of cells. Once the movie has been generated, it has been observed and analysed by the researchers. Most of the observed phenomena would be very difficult to be discovered by any other means. Of course, observations performed by the movie technique are very subjective, and do not represent statistically meaningful results. This is, however, the first step. Once certain phenomena are identified, then it is possible to prepare experiments and collect enough statistical data to confirm the initial observations. For the initial investigations, where some of the phenomena are not really expected or even completely unknown, the movie technique proved to be very successful.

It is our believe that utilisation of video sequences in this study is a valuable and essential element. It is one of the ways, if not the only way, to capture complex spatio-temporal aspects of certain phenomena, that cannot be predicted in advance. This technique has been used with great success in the field of cellular automata e.g. [16]. For 1D cellular automata the spatio-temporal aspect can be captured by a 2D image of the evolution of the single line of cells. However, for 2D almost all the studies must be done with video sequences (on-line or off-line).

8 Summary

An architecture of dynamic hierarchically organised virtual machines as a self-organising computing model has been presented. It builds on the Turing-machine-based traditional model of computation. The model provides some of the necessary facilities for open-ended evolutionary processes in self-organising software systems. The EVM system exhibits self-adaptation and self-maintenance. The EVM components are autonomous, they are executed asynchronously, they have no assigned specific function and can interact with the local neighbours. The EVM system exhibits emerging properties and hierarchical organisation via selection mechanisms, and symbiotic relationships between components.

The EVM architecture is particularly effective when applied to problems with an inherently well-organised structure. The results obtained so far suggest (although much more extensive experimentation is needed) that it can outperform random search, GA and Markov-chain based search techniques, for problems that exhibit well organised

structural patterns, and when the problem search can be split into subspaces that can be explored independently/incrementally by the EVM web of agents. In general, it appears to perform as well as a bias-optimal search techniques (and as well as standard GA). During our experiments, for specific problems, with well-defined incremental subproblems, it outperformed both GAs and stochastic search.

More experimental data needs to be collected, and more formal comparisons with existing program search techniques is planned for the future. We also plan to investigate different topological environments, environments with resource locality, and investigating the influence of introducing mobility of cells, to boost diffusion.

Nevertheless, we believe that this computational mechanism can be successfully applied to a broad range of tasks, and through its inherent hierarchical organisation, can prove to be well suited for managing highly complex computational systems. Combined with existing evolutionary search techniques, like GAs, it offers unique ability of collapsing abstraction levels and managing dynamically interdependencies between computing agents.

Bibliography

[1] Manfred Eigen and Peter Schuster. *The Hypercycle: A Principle of Natural Self-Organization*. Springer-Verlag, 1979.

[2] David B. Fogel, editor. *Evolutionary Computation – The Fossil Record*. IEEE Press, New York, USA, 1998.

[3] John R. Koza, Forrest H. Bennett, David Andre, and Martin A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann Publishers, 1999.

[4] Leonid A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[5] Lynn Margulis. *Origin of Eukaryotic Cells*. University Press, New Haven, 1970.

[6] Lynn Margulis. *Symbiosis in Cell Evolution*. Freeman & Co., San Francisco, 1981.

[7] Humberto R. Maturana and Francisco J. Varela. Autopoiesis: The organization of the living. In Robert S. Cohen and Marx W. Wartofsky, editors, *Autopoiesis and Cognition: The Realization of the Living*, volume 42 of *Boston Studies in the Philosophy of Science*. D. Reidel Publishing Company, Dordrech, Holland, 1980.

[8] Konstantin Sergeivich Mereschkowsky. Über Natur und Ursprung der Chromatophoren im Pflanzenreiche. *Biol. Zentralbl.*, 25:593–604, 1905.

[9] Mariusz Nowostawski, Martin Purvis, and Stephen Cranefield. An architecture for self-organising evolvable virtual machines. In Sven Brueckner, Giovanna Di Marzo Serugendo, Anthony Karageorgos, and Radhika Nagpal, editors, *Engineering Self Organising Sytems: Methodologies and Applications*, number 3464 in Lecture Notes in Artificial Intelligence. Springer Verlag, 2004.

[10] Juergen Schmidhuber. A general method for incremental self-improvement and multiagent learning. In X. Yao, editor, *Evolutionary Computation: Theory and Applications*, chapter 3, pages 81–123. Scientific Publishers Co., Singapore, 1999.

[11] Juergen Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54:211–254, 2004.

[12] René Thom. *Structural stability and morphogenesis*. Benjamin Addison Wesley, New York, 1975.

[13] Michael D. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. A Bradford Book, MIT Press, Cambridge, Massachusetts/London, England, 1999.

[14] Ivan Wallin. *Symbionticism and the Origin of Species*. Williams & Wilkins, Baltimore, 1927.
[15] Stephen Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.
[16] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, Inc., first edition, May 2002.
[17] Sewall Wright. Evolution in mendelian populations. *Genetics*, 16(3):97–159, March 1931.

Choose Your Tribe! - Evolution at the Next Level in a Peer-to-Peer Network*

David Hales

Department of Computer Science,
University of Bologna, Italy
dave@davidhales.com

Abstract. Many peer-to-peer (P2P) applications benefit from node specialisation. For example, the use of supernodes, the semantic clustering of media files or the distribution of different computing tasks among nodes. We describe simulation experiments with a simple selfish re-wiring protocol (SLAC) that can spontaneously self-organise networks into internally specialized groups (or "tribes"). Peers within the tribes altruistically pool their specialisms, sharing tasks and working altruistically as a team – or "tribe". This approach is scalable, robust and self-organising. These results have implications and applications in many disciplines and areas beyond P2P systems.

1 Introduction

Open Peer-to-Peer (P2P) networks (in the form of applications on-top of the internet) have become very popular for file sharing applications (e.g. Kazaa[1], Gnutella[2], Bittorrent[3]). However, can such technology be applied to other computing tasks? For example consider a system in which some nodes have lots of free storage, some high bandwidth and others non-firewalled connections to the network. Those nodes could cooperate to provide a data back-up service – something that no individual node could provide. Obviously, in such a situation, if there is demand for a back-up service we would wish the nodes to, somehow, get together and provide the service – but how?

One solution (and currently, it would seem, the only viable one for deployable applications) is to code the process of specialisation, coordination and cooperation into the protocol directly for each different kind required. So for example, where semantic clustering of media files is required for file sharing, protocols exist that implement it[4] [11]. Where systems require supernodes [14], again, these are implemented directly. There are two problems with this approach; firstly, for every kind of specialisation required a programmer must envisage this *a priori*, design a

* This work partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS).

[1] The Gnutella home page: http://www.gnutella.com

[2] The Kazaa home page: http://www.kazaa.com

[3] The BitTorrent home page: http://www.bittorrent.com. See [3] for a description of the way BitTorrent works.

[4] For example see the MLdonkey system: http://mldonkey.org

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 61 – 74, 2006.

protocol then implement and test it. Secondly, since this process is complex enough on its own, it is generally assumed that nodes will follow the protocol – it is rare to find protocols robust to node failure, noise or malicious behaviour, such as free riding, although this is, *to a certain extent*, true within the BitTorrent system [3].

Additionally, it is also rare that nodes can spontaneously change their specialism if they come to recognise that they might be able to do better following a different role. The specialism of the node tends to be hard-coded or relies on user level switches. This kind of approach limits the ability of the system to automatically adapt to changing task scenarios – however see [14] in which supernodes are dynamically allocated to improve performance.

Ideally, we would like a more general approach that could be applied to a range of different task domains with minimal tuning. We would like the approach to offer dynamic specialisation and re-specialisation if nodes come to recognise they could do better playing another role and have the ability to do so or if the task domain changes requiring different kinds of skills to be combined. In addition, we want the system to be able to deal with freeriders and errant or malicious nodes but also to support altruistic cooperation between specialists when this is required for job completion. Finally, we require this to be as scalable, self-organising and robust as possible.

In this paper we do not claim to have addressed all these issues to the level of deployment, what we propose is, we claim, the beginnings of an approach that may allow us to address these issues. In the simulated scenarios so far implemented, our results are very encouraging and we plan to continue this line of work.

In the following sections, we state our assumptions concerning behaviour in open peer-to-peer systems then we introduce the SLAC algorithm in general terms. We follow by formulating a minimal task domain scenario called the SkillWorld, to which we wish to subject a simulated P2P network running SLAC. We then describe how we apply SLAC within SkillWorld and present some experiments and results. We interpret the results and describe a "typical history" in the SkillWorld.

At the end of the paper we summarise what we have observed and what it means. We claim that the results indicate a process that has possibly profound implications and applications beyond just P2P systems.

2 Behavioural Assumptions in Open Networks

How do nodes behave in open P2P networks? Of course, the simple answer is, assuming nodes are autonomous: *anyway they like to behave!*

Given this fact, how then do we proceed to devise protocols that will lead to desired system-level functions? Obviously, we have to begin by making assumptions about the *likely behaviour* of other nodes in the network. Such assumptions should be as realistic as possible but also simple enough to be practically computable and transferable between a number of domains. Assumptions made here are essentially the axioms of a kind of mini *social theory* which then informs the design of peer software.

Many approaches (often unconsciously) inherit assumptions from previous social sciences (e.g. economics, socio-biology, sociology). For example, if we assume nodes will behave *"rationally*" in the context of *classical game theory*, then, we compute "Nash equilibrium" - as some researchers do – we inherit our assumptions from

game-theory which is a body of knowledge assuming perfect rationality and perfect information. The basic approach is to assume that all individuals have perfect knowledge of the game being played and all possible outcomes along with infinite computational time and common knowledge that all individuals are the same in these respects. Given these assumptions it is sometimes possible to analytically derive the "Nash Equilibria" of the game being played. The idea is that given the previous classical assumptions any system will find and stay in a Nash Equilbiria. However, it is unclear that such assumptions hold in dynamic open P2P networks and the derivation of such equilbria within dynamic topologies and changing populations is currently beyond state-of-the-art analytical techniques.

In the context of socio-biological models [13, 18], which are based on the evolution of behaviours of interacting animals over time, the assumption is that behaviours (or strategies) reproduce in proportion to their average fitness (utility or score) such that fitter behaviours become more numerous over time. Additionally such models assume that mutation in the form of random changes in behaviour also take place. This *evolutionary game theory* approach allows for an ecology of behaviours to evolve over time. In addition, there is no requirement that agents have perfect rationality or perfect information – just enough, such that better performing strategies tend to increase in the population. For biological systems, this occurs via Darwinian evolution where utility equates to fitness. However, P2P networks don't evolve in a Darwinian fashion. Nodes don't reproduce and it is unclear what "fitness" means in this context.

We have shown in recent work results from evolutionary models *can* be applied in networks if we allow nodes the ability "copy and re-wire" within the network to improve their own situation [7, 8]. This latter innovation demonstrates it is possible to import work originally modelled in a conventional evolutionary framework into a dynamic network model. Nevertheless, in the absence of any deductive proof of the equivalence of evolution and the re-wire rules it is necessary to implement and test previous mechanisms to determine if the properties of interest can be carried over into networks.

Summary of assumptions concerning open P2P networks:

1. Nodes are in the network for what they can get out of it
2. Nodes modify their behaviours to improve their individual benefit
3. Nodes have limited knowledge about other peers and the network in general

The first assumption would appear to be plausible within open P2P networks. In the currently popular file-sharing networks the majority of uses download and run peer client software (and hence join the network) in order to get something (e.g. to download a movie or a music file). It certainly is true that some people would join for other reasons. For example, a user may join to feel "part of an online community" [17] or to distribute only their own content - not downloading. Some could aim to damage the functionality of the network by distributing malicious content. However, we argue that neither of these motivations informs the majority of the nodes. In any case, most functions would be *enhanced* by purely altruistic behaviour (such as distributing content without downloading) and we *conjecture* that there are at least as many pure altruistic as pure malicious nodes in working networks.

The second assumption is more problematic – *who says nodes within a given P2P network change behaviours to improve their benefit?* Our argument here is rather speculative - if not conjectural. We start from the assumption of autonomy and argue that the function of peer client software is *ultimately* under the control of the user. For example, users may change operating system or client software settings (e.g. limiting upload speeds), download new versions of a peer client (e.g. incorporating ways to improve download success and rates) or simply hack their own code if they have the required skills. Of course, a hacked client can be distributed to others if it appears to have desirable properties and will tend to be adopted if it delivers those properties to others. We therefore claim that currently, this kind of process is occurring at the user level – via the adoption of various clients and the control of various node-level settings. The problem hidden in this assumption is that the space of available behaviours that each user can choose from varies over time and is also dependent on the knowledge of the user, the kind of network connection, form of operating system and many other related factors. However, we note that similar assumptions have provided some insight into human socio-cultural phenomena at least as complex as the socio-cultural phenomena of P2P systems [2].

The third assumption would appear to be a necessary one in any large and highly dynamic system – it is not practical or possible to collate accurate global statistics in most such systems.

3 The SLAC Algorithm

In previous work we showed how a simple "copy and re-wire" rule (or protocol or algorithm) could produce high-levels of cooperation within simulated P2P networks performing collective tasks. We named this algorithm "SLAC" because it uses *Selfish Link and behaviour Adaptation to produce Cooperation*. We showed that nodes in a

```
DO periodically forever
  select a random node j from the network
  compare utility of this node (i) with node j
  IF utility of j is higher (Uj >= Ui)
    drop all current links (clear view of i)
    copy links of node j (copy view from j to i)
    add link to j (add to view i a link to j)
    copy behavioural strategy of j
    with a low probability (mutation rate 1)
      drop all current links (clear view of i)
      add a link to a randomly chosen node
    with a low probability (mutation rate 2)
      change the behavioural strategy randomly
   END IF
END DO
```

Fig. 1. The generic SLAC algorithm. Each node executes this algorithm.

network could emerge cooperation within the single-round Prisoner's Dilemma (PD) game, under, what we argue, are plausible assumptions about the kinds of behaviour we find in P2P systems. We also demonstrated that the same results could carry over into a more realistic file-sharing P2P task domain [6, 15].

The basic algorithm assumes that peer nodes have the freedom to change behaviour (i.e. the way they handle and dispatch requests to and from other nodes) and drop and make links to nodes they know about. In addition, it is assumed nodes have the ability to discover other nodes randomly from the network, compare their performance against other nodes (via a comparison of utilities) and copy the links and (some of) the behaviours of other nodes. As discussed above, we assume that nodes will tend to use their abilities to selfishly increase their own utility in a greedy and adaptive way (i.e. if changing some behaviour or link increases utility then nodes will tend to select it).

In a SLAC network each node stores a neighbour list or "view" which holds the links to each neighbour node. Over time, nodes engage in some activity and generate some measure of utility *U* (this might be number of files downloaded or jobs processed etc, depending on the domain).

Periodically, each node (*i*) compares its performance against another node (*j*), randomly selected from the population. If $Ui < Uj$ then node *i* drops all current links and copies all node *j* links and adds a link to *j* itself. Also, periodically, and with low probability, each node adapts its behaviour and links in some randomized way using a kind of "mutation" operation. Mutation of the links involves removing all existing links and replacing them with a single link to a node randomly drawn from the network. Mutation of the behaviour involves some form of randomized change - the specifics being dictated by the application domain (see later). Obviously for SLAC to work we need to be able to choose a suitable node-level measure which we can use to represent the utility of the node. Moreover, we assume that nodes can compare these utilities – this can cause a number of potential problems and we discuss these in the conclusion section of this paper.

Previous "tag" models, from which SLAC was developed [6-9] have indicated that for good scalability properties the rate of mutation applied to the links needs to be higher, than that applied to the behaviour, by about one order of magnitude. In the context of the algorithm show in figure 1 this means that "mutation rate 1" >> "mutation rate 2".

When applied in a suitably large population, over time, the algorithm follows a kind of evolutionary process in which nodes with high utility tend to replace nodes with low utility with nodes periodically changing behaviour and moving in the network. However, as will be seen, this does not lead to the dominance of selfish behaviour, as might be intuitively expected, because a form of incentive mechanism emerges via a kind of ostracism in the network. The process can also be viewed as a kind of "cultural group selection" process (see later discussion).

4 The SkillWorld Scenario

In order to determine if the SLAC approach can support specialisation within tribes we construct a abstract and minimal simulated task domain that requires nodes to

perform specialized tasks cooperatively in order to satisfy their individual needs. We call the task domain SkillWorld and it is an adaptation of a sociologically inspired scenario originally given in [9, 10].

The SkillWorld consists of a population of N nodes. Each node may have zero or more links (up to a maximum of 20) to other nodes. Links are undirected such that the entire population can be considered as an undirected graph G with each vertex being a node and each edge being a link. Each vertex (or node) is composed of three state variables – a "skill type" $s \in \{1,2,3,4,5\}$, an "altruism flag" $a \in \{0,1\}$ and a satisfaction score or "utility" $u \in R$ (where R is a positive real number).

Periodically, with uniform probability, a node i is selected from the population N. A "job" J is then generated marked with a randomly chosen skill sJ. The skill is selected, again randomly with uniform probability, from the domain $\{1,2,3,4,5\}$. Job J is then passed to node i. If node i posses the correct matching skill (i.e. if $s_i = sJ$) then node i may process the job itself without any help from other nodes. For successfully processing a job J the receiving node gains one unit of credit: $u \leftarrow u + 1$.

This process of generating and passing jobs to nodes represents user-level requests for services – such as, for example, searching for a particular file, performing some processing task or storing some data. In the SkillWorld we don't represent the actual jobs to be done, rather, we represent the skill required to perform the job. In our minimal scenario, each job only requires one skill to be completed.

But what if node i receives a job for which it does not have the correct skill (i.e. if $s_i \neq sJ$) ? In this case i passes the job request to each neighbour in turn until all have been visited or one of them, j, agrees to process the job J. A neighbour j will only agree to process J if its skill matches ($s_j = sJ$) and the altruism flag is set ($a_j = 1$). If j does agree to process the job then this *costs j* a quarter unit of utility ($uj \leftarrow uj - 0.25$) yet *increases* the utility of i by one unit ($ui \leftarrow ui + 1$).

What this means is that node i looks for an altruistic neighbour with the correct skill to process job J. If i finds such a neighbour (j) it increases its utility *as if* it had completed the job itself whereas *j decreases its utility*. This reflects the notion that j is altruistically processing J for the benefit of i and that users are happy when jobs submitted to their nodes are completed but are not happy when jobs from other nodes use their node resources with no immediate benefit to themselves.

5 SLAC in the SkillWorld

We apply the SLAC algorithm within SkillWorld by making the node skill types and the altruism flags into evolvable state variables such that they are copied from more successful nodes (based on utility) and mutated occasionally with low probability.

Although SLAC has previously been demonstrated as successful in promoting cooperation in both a Prisoner's Dilemma playing scenario [6] and a simple file-sharing scenario [6] it has not yet been applied within a scenario requiring intra-group (or tribe) specialisation *in addition* to altruism. We are therefore asking a lot from a simple algorithm: to self-organise the population into altruistic yet internally specialised tribes that pass and process jobs using their various skills.

The SkillWorld is the simplest scenario we could think of that captures a process of specialisation for this initial investigation. We have a small number of skills (five in

these simulations) and we only pass jobs to immediate neighbours. Each node and job is related to a single skill only (rather than a subset of skills which would seem more realistic). Also we assume nodes can change skills at will (randomly via mutation). This latter assumption might not hold if skills relate to physical or unchangeable characteristics of nodes like storage or bandwidth for example. However, at this stage we leave more realistic scenarios with multi-hop passing and more complex skill set arrangements to future work.

In order to measure the success of SLAC we take a simple measure - the proportion of submitted jobs that are completed. We can infer that a network in which the majority of jobs submitted are completed is sustaining internally cooperative and specialised tribes since the only way to complete most jobs is for nodes to pass them to altruistic neighbours with required skills.

5.1 Some Experiments and Results

Initially we ran a set of simulation experiments in which we initialised all nodes in the population with uniformly randomly selected skills, altruism flags and links. We experimented with a number of network sizes determining for each how many cycles before the high performance was achieved.

In order to measure the success of SLAC we take a simple measure - the percentage of submitted jobs that are completed (PCJ). We can infer that a network in which the majority of jobs submitted are completed is sustaining internally cooperative and specialised groupings (or tribes) since the only way to complete most jobs is for nodes to pass them to altruistic neighbours with the required skills.

We categorized "high performance" as a PCJ > 90% and ran simulations until this value was obtained - recording the number of cycles required. Hence, if SLAC was working well in the SkillWorld we would hope that within a small number of cycles the PCJ would become high.

We used a mutation rate of 0.001 on skill type *s* and altruism flag *a* (shown in figure 1 as "mutation rate 2"). Mutation on the links (shown in figure 1 as "mutation rate 1") was an order of magnitude higher (0.01). We carry over this assumption – that the mutation rate on the links should be higher than that on the "strategies" – from previous experimental work comparing several different scenarios and models [8]. We fixed the maximum number of links between nodes to 20. Links are undirected and therefore symmetric. If an operation results a node requiring a new link and it already has the maximum then a random link is discarded by the node and the new link accepted. Using this method nodes never refuse new links but may often lose old ones. This adds to the noisy and dynamic nature of the scenario.

Figure 2 shows results from 30 individual simulation runs. Each point is a different run showing the first cycle at which the PCJ > 90%. As can be seen, high performance is attained within a few tens of cycles even for networks of size $N = 10^5$. Notice that there appears to be a very slight upward trend in cycles as N increases, however, this is negligible – the results therefore indicate close to *zero scaling cost*. This highly desirable property was also evidenced in a previous application of SLAC to a simulated fire-sharing scenario [6]. Figure 3 shows results under the same conditions except that all nodes are initialised to be selfish ($a = 0$). This gives a kind of "worst case scenario" as far as altruism evolving. It is important to show that the

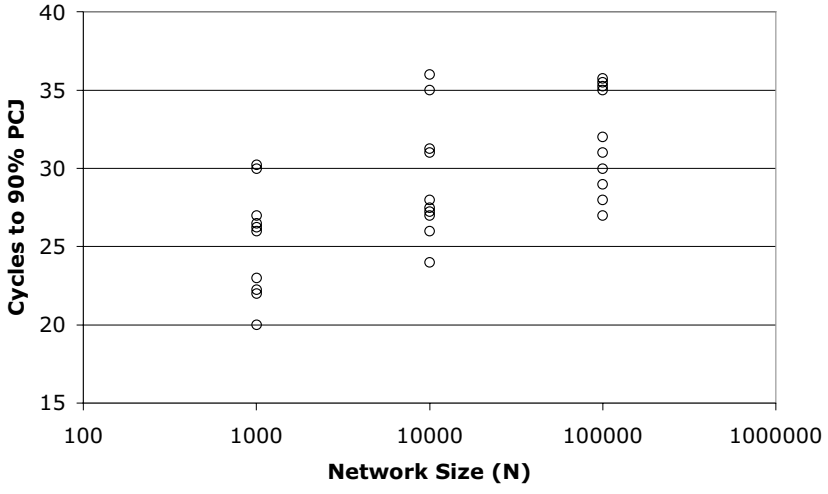


Fig. 2. Number of cycles to high performance for different network sizes. When PCJ > 90% this means that over 90% of all jobs submitted to nodes are completed. Note: overlapping circles have identical values.

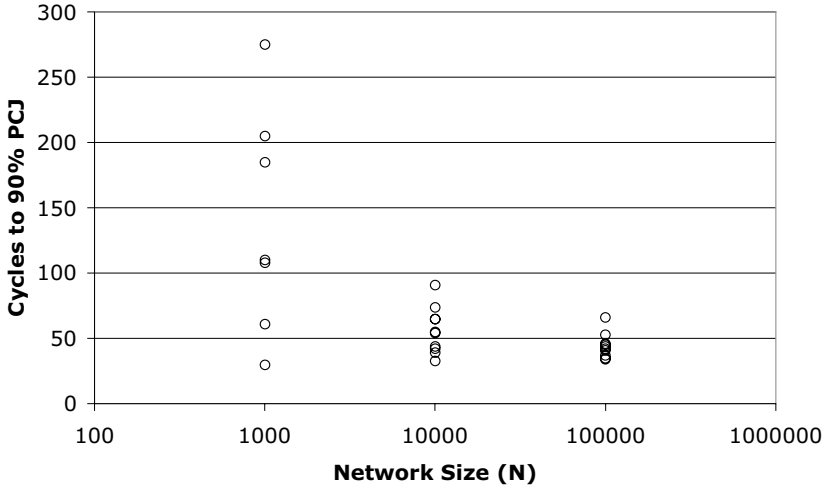


Fig. 3. Number of cycles to high performance for different network sizes when all nodes are initialised selfish ($a = 0$). This can be compared to the random initialisation results in figure 2. Note that there is a reverse scaling cost here. The results for N = 1000 are worse than shown since three outliers at about 1000 cycles are not shown here.

system can escape from this, since this demonstrates that even if a complete failure of node altruism should occur (either through chance or malicious attacks) then the system can recover relatively quickly. We notice here the reverse scaling properties

that we originally noticed and analysed in a previous "tag" model [9]. Essentially, with bigger populations, there is more likelihood of the chance formation of a small altruistic tribe. This then goes on to "seed" the population with altruism[5].

Interestingly, it was found that for populations where $N < 1000$ high performance was *not* produced even when runs were extended to several thousand cycles. Intuitively this is consistent with the "group selection" hypothesis concerning how SLAC operates. With small populations, there are not enough nodes to form enough competing groups (or tribes) so evolution cannot operate at the group level.

5.2 History in the SkillWorld – Tribal Dynamics

One way to convey the dynamics of a typical SkillWorld simulation run is to describe a typical "history" in narrative form – this method is sometimes used in computational sociology, particularly in work with artificial societies [1, 5] carried over from more traditional sociological methods of explanation. In the rest of this section we give such a "typical history". Although we will make general points we will also refer to a specific single simulation, run given in Figure 4, to illustrate our analysis.

Initially, the SkillWorld is a random graph, all nodes are connected via a few hops and clustering is low. Skills and altruism are randomly scattered. Very quickly, the graph breaks into a population of many disconnected components because nodes quickly re-wire themselves to better performing nodes. The better performing nodes are initially the non-altruists who exploit their groups (or tribes) selfishly. However, this is a non-sustainable strategy since this exploitation causes nodes to leave their exploited tribes and join tribes in which there is less exploitation – nodes in tribes with less exploiters in them do better (higher utility) because they are cooperating as a team. The tribes dominated by non-altruists quickly "wither away" as nodes leave. When no nodes are left then the tribe no-longer exist – in this way *tribes die*, even though *nodes do not die*. This emergent property of the birth and death of tribes lays the ground for evolution to operate that the group (tribe) level.

Figure 4 indicates the above process occurring in the first 10 cycles or so. Notice that the number of selfish nodes peaks, and the proportion of completed jobs (PCJ) bottoms out, at about cycle 10. The number of components (i.e. tribes) increases in the early phase peaking just before cycle 20 (representing a peak of 60 components).

Altruistic tribes function well and grow as more nodes join, new tribes are occasionally formed as nodes randomly, through mutation, split from a tribe. As altruistic tribes grow larger they eventually become "infected" or "invaded" by a non-altruist node – either by mutation of an existing member node or the entering of a new node to the tribe.

When this happens the tribe is quickly destroy via dispersion since a non-altruist will exploit the tribe selfishly and this will lead to many more nodes quickly copying that node until the tribe "dies" because all nodes leave it – because a tribe dominated by selfish nodes gives lower utility to *all nodes* within it than one dominated by altruists.

---

[5] See [9] for a more detailed explanation of this reverse-scaling cost including the beginnings of an analytical treatment.

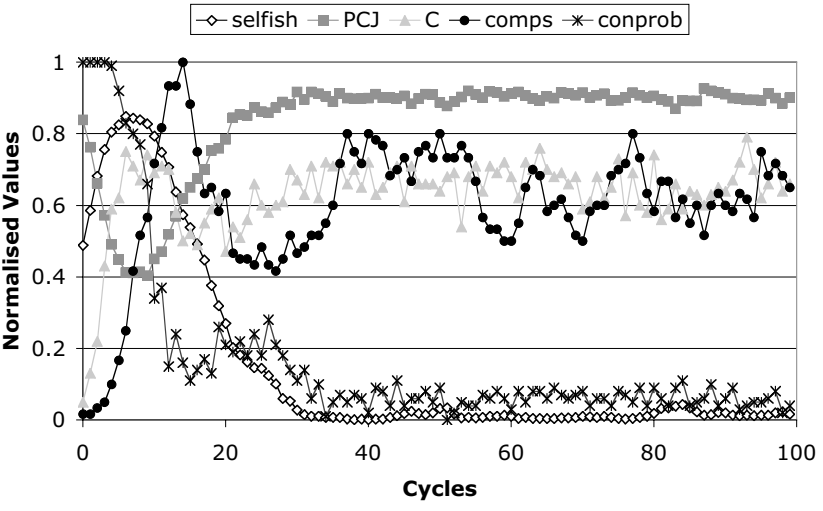


Fig. 4. The time series of a typical single run in SkillWorld (N=1000). Shown are the number of selfish nodes as a proportion of the entire population (selfish), the proportion of submitted jobs that get completed (PCJ), the clustering coefficient (C), the number of components in the population (comps, which is normalised by dividing by 60) and the average probability that a route exists between any two nodes (conprob).

Figure 4 shows, from about cycle 20 onward, the above process occurring. A decrease in the number of components (comps) and an increase in completed jobs (PCJ) are correlated with a decrease in the number of selfish nodes (selfish). This is because altruistic tribes grow in size – reducing the total number of components (comps) and reducing selfish nodes (selfish). By about cycle 30 selfishness is very low and completed jobs (PCJ) reaches a high level. Notice that the dynamic nature of the formation and dissolution of the tribes is reflected in the variation of the number of components over time (comps) after PCJ goes high.

History in the SkillWorld is the history of the formation, growth and destruction of tribes. From the simple rules of the SLAC algorithm an *evolutionary process emerges at the tribal or group level*. Essentially one can think of this evolution as the competition between tribes to retain nodes to continue to exist. This process is in constant flux due to mutation and movement, no equilibrium state is attained and no tribe lasts forever. As long as new altruistic tribes are created at least as rapidly as they are destroyed then altruism can survive.

Figure 5 shows a small detail of snapshots of the population over time (space does not permit full size snapshots). As can be seen, tribes quickly emerge and grow, producing various structures and sizes with internally specialised nodes.

5.3 Tribal Structures

Within the SkillWorld, tribes with different structures and skill mixes will support different levels of utility – a highly connected tribe with an even mix of skills would

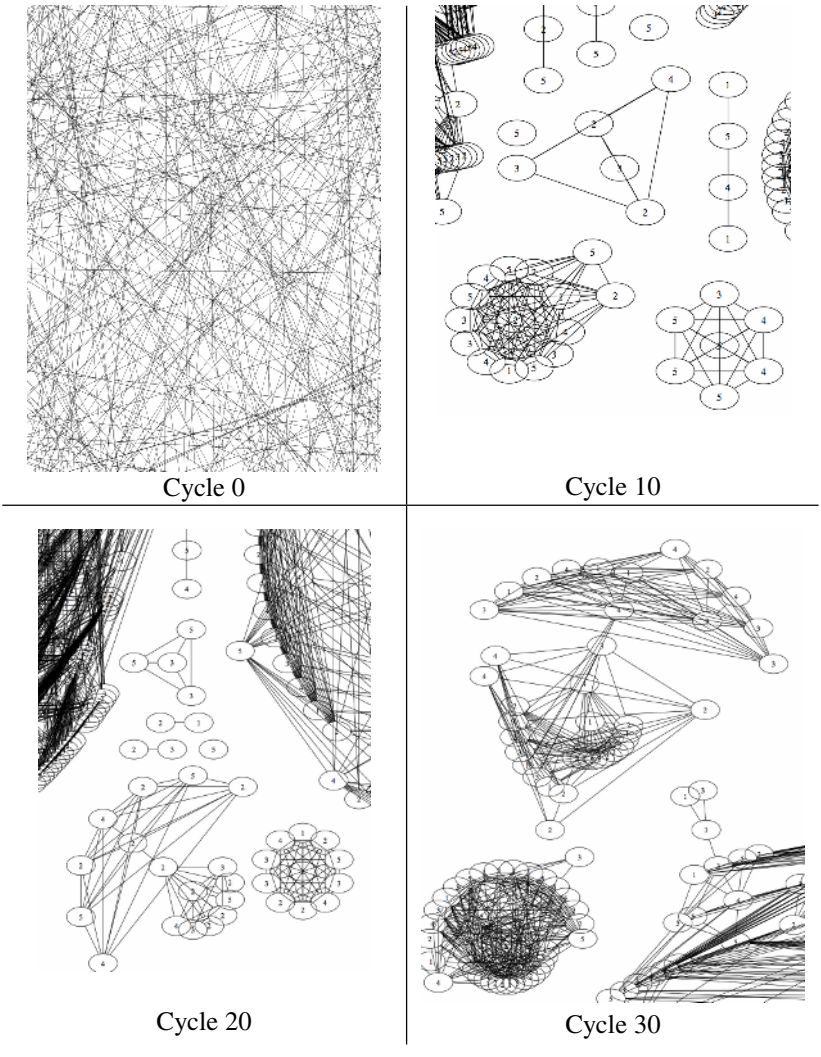


Fig. 5. Details showing just a small part of the entire population for the same typical run as show in figure 4. From an initially random graph, disconnected components (we call tribes) emerge with internal specialisation and rich structure. The numbers in the nodes represent the node skill type.[6]

produce better results than a tribe missing some skill. Hence, selection at the tribe level (group selection) will tend to operate to structure the tribes into more optimal structures of skill types. We would therefore expect to see tribes composed of nodes possessing each skill type linked together such that a node receiving a job can either process it directly or will be directly linked to a node with the appropriate skill willing

[6] Full sized pictures can be found at http://www.davidhales.com/esoa05pics/

to do the job. In the SkillWorld then, we have tribe level selection not only operating to control selfishness but also to tune the internal (organisational) structure of the tribe.

We find this particularly exciting since we believe that by increasing the complexity of the task domain and giving nodes a little more freedom to hop more than one link within their tribe it should be possible to evolve tribes with *complex organisational structures* tuned to performing in the given task domain. Moreover, since the tribes are constantly evolving they should be able to *change their structure dynamically to address a change in the task domain*[7].

6 Conclusion

We have demonstrated that the SLAC algorithm can be applied in a scenario (the SkillWorld) requiring node specialisation in addition to the suppression of selfish behaviour. When the algorithm is executed the network quickly divides into competing "tribes" (disconnected components). An evolutionary process then emerges at the group level selecting efficient internally specialised tribes – which deliver high levels of service with respect to user submitted jobs at the nodes.

We adapted the SkillWorld scenario from a previous model developed for the purposes of social scientific theorising [10]. The previous "tag-based" model relied on mean-field mixing (with no population structure) and followed a conventional evolutionary process.

Our belief that the SLAC algorithm works via a kind of group selection occurring at the level of the "tribe" gave us the *a priori* expectation that it would select tribes that could perform well in the SkillWorld. In this sense, *dare we claim the beginning of a "proto-theory"* allowing us to make some modest qualitative predictions?

More generally, we claim that this paper demonstrates concretely within a dynamic network the emergence of what has been termed a "meta-state transition" (MST) within evolution [12]. It has been argued that the emergence of life itself and major steps in biological evolution (e.g. multi-cellular organisms) and social evolution (e.g. large complex societies) occur over such MST's. In this context we advance our results as possibly of great theoretical insight.

It is important to understand that *the concept of the "tribe" is actually a theoretical construct* we use to help to explain and understand the *emergent phenomena* produced by the SLAC algorithm over time. The tribes are not "programmed" into the nodes *a priori* but rather emerge from the interplay of task domain, interaction and the SLAC algorithm. We use the concept of "tribes" because we believe it to be valuable in beginning to understand, control and theorise about what is occurring in SLAC networks. However, since the tribes are emergent we do not *begin* with a "theory of tribes" rather we observe, experiment and induce knowledge about them. As discussed below, this does not preclude, but, in fact, should support, the formation of an analytical theory – we hope.

Since the nodes do not die or model genetic operators, the tribe level selection process can be viewed as a kind of artificial *cultural group selection process*. What is

[7] Further experiments not detailed here, demonstrate that even when all skills in the population are initialised to the same single type – the network quickly adapts into an even skill spread due to mutation on the skills and selection at the tribe level.

quite extraordinary is that *such a simple node level algorithm (SLAC) based on a few plausible assumptions about preferential attachment can lead to such complex and useful group level evolutionary dynamics*.

A key issue however, is that, although SLAC is simple to implement the dynamics are complex and currently it is not know how analytical tools can be applied to truly understand, predict and prove the properties of SLAC. So far the only "proofs" we have are in the form of "existence proofs" demonstrated by empirical analysis of simulation runs. Such "proofs" are not watertight and can always be questioned given anomalous results from future simulation studies (rather like experiments in the natural sciences). We have some confidence in the general results from SLAC-like algorithms however (such as those based purely on "tags") since there have been a number of replications of those results from multiple independent implementations using different languages, machines and programmers [4]. However, none of this offers predictive insight into the process as a good analytical model would. What we currently have is a kind of "toolbox" of algorithmic heuristics that appear to be reasonably robust over some minimal task domains and scenarios.

Another open issue is the concept of utility as used in the simulations given here. We assume that some simple measure can be readily calculated and that such measures can be compared between nodes. Both such assumptions may not hold in many task domains. Additionally, there may be incentives for nodes to lie about their utility – we leave an exploration of these issues to future work.

However, currently, the only way to apply these methods to new domains is to simulate and experiment – copying and adapting heuristics that worked previously in similar domains. Perhaps this is not so far away from the edit / compile / debug cycle of good old-fashioned software engineering (GOFSE). This could bode well for future progress.

Acknowledgements

Thanks go to Mark Jelasity, Simon Patarin, Ozalp Babaoglu, and Alberto Montresor from Bologna and Bruce Edmonds and Scott Moss from Manchester for discussions, ideas and encouragement.

References

[1] Axelrod, R.: A model of the emergence of new political actors. In *G. N. Gilbert and R. Conte (eds.), Artificial Societies*. UCL Press, London, 1995.

[2] Binmore, K.: *Just Playing, Game Theory and the Social Contract*, Volume II, Cambridge, Mass., and London: The MIT Press, 1998.

[3] Cohen, B. Incentives Build Robustness in BitTorrent (2003). Presented at the 1st Workshop on the Economics of Peer-2-Peer Systems, June 5-6, Berkley, CA, (available at: http://www.sims.berkeley.edu/ research/conferences/p2pecon/), 2003.

[4] Edmonds, B. and Hales, D.: Replication, Replication and Replication - Some Hard Lessons from Model Alignment. Journal of Artificial Societies and Social Simulation 6(4), 2003.

[5] Epstein J.M. and Axtell R.: *Growing Artificial Societies - Social Science from the Bottom Up*, Cambridge MA, MIT Press, 1996.

[6] Hales, D.: From selfish nodes to cooperative networks - emergent link based incentives in peer-to-peer networks. In *Proc. of the 4th IEEE International Conference on Peer-to-Peer Computing (P2P2004)*, IEEE Computer Soc. Press, (available at: http://www.davidhales.com), 2004.

[7] Hales, D.: Self-Organising, Open and Cooperative P2P Societies – From Tags to Networks. *Proceedings of the 2nd Workshop on Engineering Self-Organising Applications (ESOA 2004), LNCS 3464*, pp.123-137, 2005.

[8] Hales, D.: Change Your Tags Fast! – a necessary condition for cooperation? *Proceedings of the Workshop on Multi-Agents and Multi-Agent-Based Simulation (MABS 2004), LNAI 3415*, Springer, 2005.

[9] Hales, D.: Cooperation without Space or Memory: Tags, Groups and the Prisoner's Dilemma. In *Moss, S., Davidsson, P. (Eds.) Multi-Agent-Based Simulation. Lecture Notes in Artificial Intelligence, 1979*, pp.157-166. Berlin: Springer-Verlag, (available at http://www.davidhales.com), 2000.

[10] Hales, D.: Evolving Specialisation, Altruism and Group-Level Optimisation Using Tags. In *Sichman, J. S., Bousquet, F. Davidsson, P. (Eds.) Multi-Agent-Based Simulation II. Lecture Notes in Artificial Intelligence 2581*, pp.26-35 Springer, (available at http://www.davidhales.com), 2002.

[11] Handurukande, S. A.-M. Kermarrec, F. Le Fessant and L. Massoulié (2004) Exploiting Semantic Clustering in the eDonkey P2P Network. Presented at the 11th ACM SIGOPS European Workshop (SIGOPS), Leuven (Belgium), Sep 2004.

[12] Heylighen F.: Evolution, Selfishness and Cooperation, *Journal of Ideas*, Vol 2, # 4, pp 70-76, 1992.

[13] Maynard-Smith, John: *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[14] Montresor, A.: A Robust Protocol for Building Superpeer Overlay Topologies. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing*, pages 202-209, Zurich, Switzerland, August 2004. IEEE Press, 2004.

[15] Qixiang Sun & Garcia-Molina: SLIC: A Selfish Link-based Incentive Mechanism for Unstructured Peer-to-Peer Networks. In *Proceedings of the 24th IEEE international Conference on Distributed Systems (March 2004)*. IEEE computer Society, 2004.

[16] Roberts, G. & Sherratt, T. N. Similarity does not breed cooperation *Nature* 418, 449-500, 2002.

[17] Strahilevitz, Lior,: Charismatic Code, Social Norms, and the Emergence of Cooperation on the File-Swapping Networks. *Virginia Law Review*, Vol. 89, (available at: http://ssrn.com/abstract=329700), 2003.

[18] Trivers, R.: The evolution of reciprocal altruism. *Q. Rev. Biol*. 46, 35-57, 1971.

Exchange Values and Self-regulation of Exchanges in Multi-agent Systems: The Provisory, Centralized Model

Graçaliz Pereira Dimuro[1] and Antônio Carlos da Rocha Costa[1,2]

[1] Programa de Pós-Graduação em Informática,
Universidade Católica de Pelotas, 96010-000 Pelotas, Brazil
{liz, rocha}@ucpel.tche.br
[2] PPPGC, UFRGS, 91501-970 Porto Alegre, Brazil

Abstract. This paper introduces systems of exchange values as tools for the self-regulation of multi-agent systems. Systems of exchange values are defined on the basis of the model of social exchanges proposed by J. Piaget. A model of social control is proposed, where exchange values are used for supporting the regulation of the performance of social exchanges. Social control is structured around two coordinated functions: the evaluation of the current balance of exchange values and the determination of the target equilibrium point for such balance, and the maintenance of the balance of exchange values around the current target equilibrium point. The paper focuses on the second function of social control, introducing a (for the moment, centralized) equilibrium supervisor that solves the problem of keeping the system in a state of equilibrium by making use of a Qualitative Markov Decision Process that uses intervals for the representation of exchange values.

1 Introduction

Social control [1, 2] is a powerful notion for explaining the self-regulation of a society. On the other hand, following J. Piaget [3], agent interactions can be conceived as *social exchanges* that also include an evaluation process, through which so-called *qualitative exchange values* are associated with each stage of exchange.

The dynamics of such social exchanges constitutes, through the exchange of values, a *qualitative economy*, and the regulation of such economy of qualitative values, through various kinds of social mechanisms and processes (like rules, habits etc.), constitutes the main purpose of social control [3].

In this paper, we build on our previous work on social exchanges in multi-agent systems [4, 5, 6, 7], to consider in a preliminary fashion the way social control can be construed as a self-regulated system of exchange values.

In Sect. 2, we summarize the sociological bases of the work. In Sect. 3, we present our proposal for the self-regulation of exchanges, based on *equilibrium supervisors*, using Interval Mathematics to represent exchange values and Qualitative Interval Markov Decision Processes (QI-MDP) to regulate the equilibrium. Section 4 is the Conclusion.

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 75–89, 2006.

2 Sociological Bases of the Work

2.1 Piaget's Theory of Exchange Values

We summarize here J. Piaget's approach to social exchanges [3].

Interactions between individuals are understood as exchanges of *services* among them, involving not only the realization of services by some individuals on behalf of others, but also the evaluation of such services, from various points of view, by every individual involved in them.

The evaluation of a service by an individual (either the server of the service or its client) is done on the basis of a *scale* of so-called *exchange values*, that are of a qualitative nature, since such values express subjective evaluations.

Exchange values give rise to a *qualitative economy of social exchanges*, where individuals acquire credits for services they have performed, and debits to others for services the others have performed to them. The balances of exchange values allow individuals to observe the state of equilibrium of the social exchanges (even between just two individuals) and to react according to such state (e.g., trying to enforce equilibrium, to overcome high debts, to keep their status as privileged beneficiaries of the exchanges etc.).

A *social exchange* between two agents, α and β, is performed involving two types of stages. In stages of type $\mathrm{I}_{\alpha\beta}$, the agent α realizes a service for the agent β. The exchange values involved in this type of exchange stage are the following:

- $r_{\mathrm{I}_{\alpha\beta}}$ is the value of the *investment* done by α for the realization of a service for β. An investment value is always *negative*.
- $s_{\mathrm{I}_{\beta\alpha}}$ is the value of β's *satisfaction* due to the receiving of the service done by α.
- $t_{\mathrm{I}_{\beta\alpha}}$ is the value of β's *debt*, the debt it acquired to α for its satisfaction with the service done by α.
- $v_{\mathrm{I}_{\alpha\beta}}$ is the value of the *credit* that α acquires from β for having realized the service for β.

In stages of the type $\mathrm{II}_{\alpha\beta}$, the agent α asks the payment for the service previously done for the agent β, and the values related with this exchange stage have similar meaning: $v_{\mathrm{II}_{\alpha\beta}}$ is the credit charged by α on β; $t_{\mathrm{II}_{\beta\alpha}}$ is the debit acknowledge by β; $r_{\mathrm{II}_{\beta\alpha}}$ is the cost of the return service performed by β to α; $s_{\mathrm{II}_{\alpha\beta}}$ is the satisfaction got back by α. The order in which the exchange stages may occur is not necessarily $\mathrm{I}_{\alpha\beta} - \mathrm{II}_{\alpha\beta}$.

Piaget's modelling of social exchanges aim at the formalization of the rules that determine the equilibrium of social exchanges:

$$\begin{array}{ll} \textbf{Rule } \mathrm{I}_{\alpha\beta}: & (r_{\mathrm{I}_{\alpha\beta}} = s_{\mathrm{I}_{\beta\alpha}}) \wedge (s_{\mathrm{I}_{\beta\alpha}} = t_{\mathrm{I}_{\beta\alpha}}) \wedge (t_{\mathrm{I}_{\beta\alpha}} = v_{\mathrm{I}_{\alpha\beta}}) \\ \textbf{Rule } \mathrm{II}_{\alpha\beta}: & (v_{\mathrm{II}_{\alpha\beta}} = t_{\mathrm{II}_{\beta\alpha}}) \wedge (t_{\mathrm{II}_{\beta\alpha}} = r_{\mathrm{II}_{\beta\alpha}}) \wedge (r_{\mathrm{II}_{\beta\alpha}} = s_{\mathrm{II}_{\alpha\beta}}) \\ \textbf{Rule } \mathrm{I}_{\alpha\beta}\mathrm{II}_{\beta\alpha}: & v_{\mathrm{I}_{\alpha\beta}} = v_{\mathrm{II}_{\alpha\beta}} \end{array}$$

Rule $\mathrm{I}_{\alpha\beta}$ states the conditions for the internal equilibrium of stage $\mathrm{I}_{\alpha\beta}$, implying that $r_{\mathrm{I}_{\alpha\beta}} = v_{\mathrm{I}_{\alpha\beta}}$. **Rule** $\mathrm{II}_{\alpha\beta}$ states the conditions for the internal equilibrium of stage $\mathrm{II}_{\alpha\beta}$, implying that $v_{\mathrm{II}_{\alpha\beta}} = s_{\mathrm{II}_{\alpha\beta}}$. **Rule** $\mathrm{I}_{\alpha\beta}\mathrm{II}_{\alpha\beta}$ states the conditions for the external equilibrium between the two stages, $\mathrm{I}_{\alpha\beta}$ and $\mathrm{II}_{\alpha\beta}$, implying that $r_{\mathrm{I}_{\alpha\beta}} = s_{\mathrm{II}_{\alpha\beta}}$.

2.2 Social Organization, Social Equilibrium and Social Control

The *organization* of a society is a structure $O = (A, F, Ro, E, BV, Ru)$, where: A is the set of *agents*; F is the set of *services* (functions) that agents, and sets of agents, may provide for each other; Ro is the set of *social roles* that agents may be assigned to; E is the set of *social exchanges* that agents may perform; BV is the set of *balances of exchange values* that supports the various ways agents may evaluate social exchanges; Ru is the set of *social rules* that may be used to regulate the agents' behaviors. We let undetermined the details of the elements of such sets, in order to have an *abstract* notion of social organization, that may be instantiated in various ways, in various applications.

A few more *complementary elements* should be added, so that the dynamics of the organization can be explained:

- the way the set E of all possible social exchanges are related to the subset of agents that are capable of realizing them together, given by $Cap : \wp(A) \rightarrow E$;
- the way each social function is *implemented* by a set of agents in the form of a social exchange, given by $I : F \times \wp(A) \rightarrow E$;
- the way each agent evaluates the performance of an exchange, according to the piagetian theory of exchange values explained above, given by the function $Ev : E \times A \rightarrow BV$;
- the way each social rule determines the *permitted*, *obligatory* and *forbidden* behaviors of agents in a social exchange, according to the balance of exchange values assigned to the social exchange, regarding the performance of a social function, given by $Ru : F \times A \times E \rightarrow \wp(IBeh \times \{\mathbf{p}, \mathbf{o}, \mathbf{f}\})$.

In any dynamical system where a notion of *equilibrium* can be defined, two related concepts immediately apply, namely, the concepts of *deviation* and *compensation*. Deviation is any action that may happen in a system and lead it to *disequilibrium* (away from equilibrium). Compensation is any action that may happen in a system, when it is in disequilibrium, and lead it back to equilibrium.

In social organizations, balances of exchange values can be taken as the bases for the definition of *social equilibrium* [3]: in general, a social organization is in equilibrium if the balance of exchange values is such that there is an equanimous distribution of exchange values among the agents[1]. Often, the organization is said to be in equilibrium if the balance of exchange values of every agent, with respect to each other agent with which it has interacted, is zero.

Regulation is the process of determining which compensation should be performed, at a given moment, to compensate a deviation, when the system is in disequilibrium. In Homan's terms [2], which were adopted in [1], Piaget's process of regulation is a *social control* process.

Social rules specify mechanisms of social control by stipulating, for each state of disequilibrium, the kind of action that should be performed in order to re-establish the equilibrium of the system.

[1] Where equity is defined in a way that depends on the particular organization that is being considered.

3 Self-regulation of Social Exchanges Based on Equilibrium Supervisors

3.1 Using Interval Mathematics to Represent Exchange Values

Interval Mathematics [8] is a mathematical theory introduced in the 1960's that aims at the automatic and rigorous controlling of the errors that arise in numerical computations. Any real number $x \in \mathbb{R}$ is represented by a real interval $X = [x_1, x_2]$, with $x_1, x_2 \in \mathbb{R}$ and $x_1 \leq x \leq x_2$. The set of intervals is denoted by $\mathbb{IR}$. x_1 and x_2 denote, respectively, the left and right endpoints of X.

In this paper, intervals are used to capture the qualitative nature of Piaget's concept of scale of exchange values [3]. Consider the set $\mathbb{IR}_L = \{[x_1, x_2] \mid -L \leq x_1 \leq x_2 \leq L, x_1, x_2 \in \mathbb{R}\}$ of real intervals bounded by $L \in \mathbb{R}$ $(L > 0)$ and $s \in \mathbb{R}$, called the reference value, such that $-L < s < L$. Let $\mathcal{IR}_L^s = (\mathbb{IR}_L, +, \Theta_s, \widetilde{\cdots}_s, \approx_s)$ be an *s-centered scale of interval exchange values*, where:

(i) $+ : \mathbb{IR}_L \times \mathbb{IR}_L \to \mathbb{IR}_L$, $X + Y = [\max\{x_1 + y_1, -L\}, \min\{x_2 + y_2, L\}]$ is the *L-bounded addition* operation.

(ii) An *s-reference interval* is any $X \in \mathbb{IR}_L$ such that $mid(X) = s$, where $mid(X) = \frac{x_1+x_2}{2}$ is the mid point of X. The set of s-reference intervals is denoted by Θ_s.

(iii) An *s-compensation interval* of $X \in \mathbb{IR}_L$ is any interval $X' \in \mathbb{IR}_L$ such that $X + X' \in \Theta_s$. The set of s-compensation intervals of X is denoted by $\widetilde{X}_s$.

(iv) $X \approx_s Y \Leftrightarrow \exists Y' \in \widetilde{Y}_s : X + Y' \in \Theta_s$ defines the *qualitative equivalence relation*.

For a given $s \in \mathbb{R}$, an interval $\mu\widetilde{X}_s \in \widetilde{X}_s$ is said to be the *least* s-compensation interval of X if whenever there exists another s-compensation interval $S \in \widetilde{X}_s$ for X it holds that $d(\mu\widetilde{X}) \leq d(S)$, where $d(X)$ is the diameter of X, defined by $d(X) = x_2 - x_1$. For all $X \in \mathbb{IR}_L$, it follows that:

Proposition 1. *For a given $s \in \mathbb{R}$, it holds that (i) $\widetilde{X} = [-mid(X) + s - k, -mid(X) + s + k]$, with $k \geq 0 \in \mathbb{R}$; (ii) $\mu\widetilde{X}_s = [-mid(X) + s, -mid(X) + s]$.*

To avoid the problems of representing the real number s in the floating point system and of the rounding errors that arise in any numerical computation [8], it is a good practice to consider a given tolerance $\epsilon \in \mathbb{R}$ $(\epsilon \geq 0)$ such that the reference value for the scale is given by the interval $s_\epsilon = [s-\epsilon, s+\epsilon]$, with $s-\epsilon$ and $s+\epsilon$ being machine numbers. An s_ϵ-reference interval is any $X \in \mathbb{IR}_L$ such that $mid(X) \in s_\epsilon$. The set of s_ϵ-reference intervals is denoted by Θ_{s_ϵ}. The least s_ϵ-compensation interval is then given by $\mu\widetilde{X}_{s_\epsilon} = [-mid(X)+s-\epsilon, -mid(X)+s+\epsilon]$.

3.2 Self-regulation of Social Exchanges

In general, the *exchange values-based mechanism of social control* may be put to operate in two ways. On the one hand, social rules may be enforced by *authorities* that have the capacity to force the agents of the society to follow such rules. On

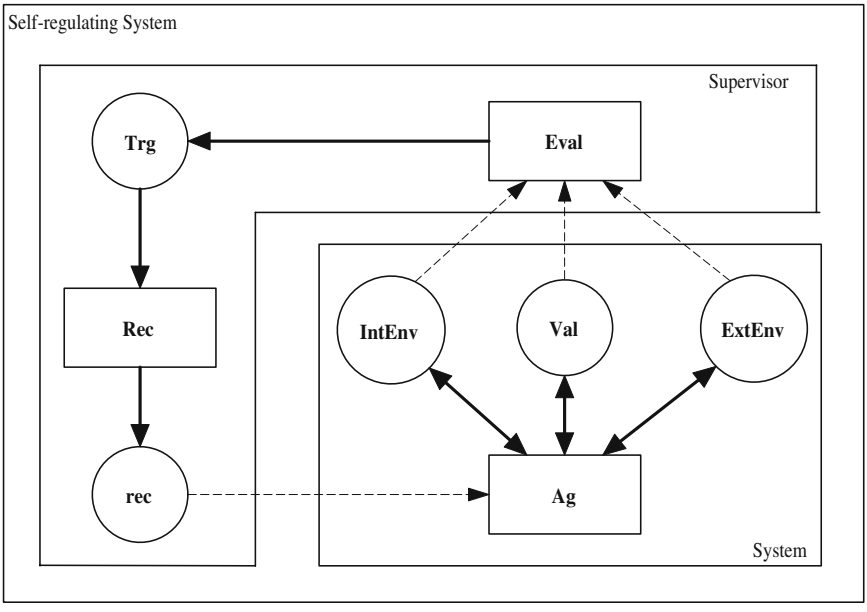


Fig. 1. The mechanism of self-regulation

the other hand, social rules may be *internalized* by the agents, so that agents follow such rules because they were incorporated into the agents' behaviors.

In the following, as a preparatory step to the future modelling of decentralized social control mechanisms based on rules internalized in agents, we introduce a centralized model version of social control, extending the notion of supervisor of social equilibrium, first proposed in [5].

We consider that a *supervisor of social equilibrium* is a component of the society (possibly an agent) that is able to determine, at each time, the best target equilibrium point for the system and to recommend to the agents the set of actions that should be performed in order to lead the system towards the equilibrium, and to maintain the system equilibrated until another equilibrium point is required.

Figure 1 pictures our idea of self-regulation based on supervisors of social equilibrium. Ag is the set of agents that constitute the system. $ExtEnv$ is the *external environment* of the system. $IntEnv$ is its *internal environment* (all system objects, excluding the agents). Val is the *set of values* the agents exchange while their are interacting.

$Eval$ is the *evaluator*, the module of the supervisor that is responsible for evaluating the current state of the system, both in what concerns its internal condition of equilibrium and in what concerns the demands coming from the external and the internal environments. Trg is the *target equilibrium point* that the evaluator determines as the point of equilibrium in which the system should be at the current moment. Rec is the *recommender*, the module of the supervisor

that analyzes the current balance of exchange values and, given the current target equilibrium point, makes recommendations of exchanges to the agents. *rec* is the set of recommendations that may be given by the recommender to the agents.

The logic of the regulation process is embedded in the equilibrium supervisor in the form of a *recommendation policy*, that determines, at each time, for each state, an appropriate action aiming to reach the desirable equilibrium state.

3.3 The Modelling of Social Exchanges

Let T be a set of discrete instants of time. A *qualitative interval exchange-value system* for modelling the exchanges from an agent α to an agent β is a structure $\mathbf{IR} = \langle \mathcal{IR}_L; (r_{\mathrm{I}}, s_{\mathrm{I}}, t_{\mathrm{I}}, v_{\mathrm{I}}), (r_{\mathrm{II}}, s_{\mathrm{II}}, t_{\mathrm{II}}, v_{\mathrm{II}}) \rangle$, where

$$r_{\mathrm{I}_{\alpha\beta}}, s_{\mathrm{I}_{\beta\alpha}}, t_{\mathrm{I}_{\beta\alpha}}, v_{\mathrm{I}_{\alpha\beta}}, r_{\mathrm{II}_{\beta\alpha}}, s_{\mathrm{II}_{\alpha\beta}}, t_{\mathrm{II}_{\beta\alpha}}, v_{\mathrm{II}_{\alpha\beta}} : T \to \mathbb{IR}_L \cup \{\bot\} \tag{1}$$

are the *exchange-value functions* that evaluate, at each time instant $t \in T$, the investment, satisfaction, debt and credit values, among each two different interacting agents α and β, in each stage I and II, respectively. The symbol $\bot$ denotes an undefined exchange value. In the following, for $k = r, s, t, v$, we use the notation $k_{\mathrm{I}_{\alpha\beta}}(t) = k^t_{\mathrm{I}_{\alpha\beta}}$, $k_{\mathrm{II}_{\alpha\beta}}(t) = k^t_{\mathrm{II}_{\alpha\beta}}$, $k_{\mathrm{I}_{\beta\alpha}}(t) = k^t_{\mathrm{I}_{\beta\alpha}}$ and $k^t_{\mathrm{II}_{\beta\alpha}}$. At a given time instant t, the following constraints must be satisfied:

$$r^t_{\mathrm{I}_{\alpha\beta}} = \bot \Rightarrow s^t_{\mathrm{I}_{\beta\alpha}} = t^t_{\mathrm{I}_{\beta\alpha}} = v^t_{\mathrm{I}_{\alpha\beta}} = \bot;\ v^t_{\mathrm{II}_{\alpha\beta}} = \bot \Rightarrow t^t_{\mathrm{II}_{\beta\alpha}} = r^t_{\mathrm{II}_{\beta\alpha}} = s^t_{\mathrm{II}_{\alpha\beta}} = \bot;$$
$$r^t_{\mathrm{I}_{\alpha\beta}} \neq \bot \Rightarrow v^t_{\mathrm{II}_{\alpha\beta}} = \bot, \tag{2}$$

where *(i)* $r^t_{\mathrm{I}_{\alpha\beta}} = \bot$ denotes that the agent α did not realize a service for the agent β at time t, and, therefore, all the other corresponding exchange values in the stage I resulted undefined; *(ii)* $v^t_{\mathrm{II}_{\alpha\beta}} = \bot$ denotes that the agent α, at time t, did not charge the credit for a service previously done for the agent β, and, therefore, all the other corresponding exchange values in the stage II resulted undefined. The implication (2) means that it is not possible for an agent α to realize a service for an agent β and, at the same time t, to charge him a credit. From (2) it follows that is also required that $v^t_{\mathrm{II}_{\alpha\beta}} \neq \bot \Rightarrow r^t_{\mathrm{I}_{\alpha\beta}} = \bot$.

A *configuration of exchange values* for any pair of agents α and β at a time instant t is specified by one of the tuples of exchange values $(r^t_{\mathrm{I}_{\alpha\beta}}, s^t_{\mathrm{I}_{\beta\alpha}}, t^t_{\mathrm{I}_{\beta\alpha}}, v^t_{\mathrm{I}_{\alpha\beta}})$, $(r^t_{\mathrm{I}_{\beta\alpha}}, s^t_{\mathrm{I}_{\alpha\beta}}, t^t_{\mathrm{I}_{\alpha\beta}}, v^t_{\mathrm{I}_{\beta\alpha}})$, $(v^t_{\mathrm{II}_{\alpha\beta}}, t^t_{\mathrm{II}_{\beta\alpha}}, r^t_{\mathrm{II}_{\beta\alpha}}, s^t_{\mathrm{II}_{\alpha\beta}})$, $(v^t_{\mathrm{II}_{\beta\alpha}}, t^t_{\mathrm{II}_{\alpha\beta}}, r^t_{\mathrm{II}_{\alpha\beta}}, s^t_{\mathrm{II}_{\beta\alpha}})$.

The *exchange balance* of stages of type I of a social exchange process between any pair of agents α and β that has occurred during T is a tuple

$$b^T_{\mathrm{I}_{\{\alpha,\beta\}}} = \left(r^T_{\mathrm{I}_{\alpha\beta}}, r^T_{\mathrm{I}_{\beta\alpha}}, s^T_{\mathrm{I}_{\alpha\beta}}, s^T_{\mathrm{I}_{\beta\alpha}} \mid t^T_{\mathrm{I}_{\alpha\beta}}, t^T_{\mathrm{I}_{\beta\alpha}}, v^T_{\mathrm{I}_{\alpha\beta}}, v^T_{\mathrm{I}_{\beta\alpha}} \right), \tag{3}$$

where, for $k = r, s, t, v$, $k^T_{\mathrm{I}_{\alpha\beta}} = \sum_{t \in T} k^t_{\mathrm{I}_{\alpha\beta}}$ and $k^T_{\mathrm{I}_{\beta\alpha}} = \sum_{t \in T} k^t_{\mathrm{I}_{\beta\alpha}}$, for all $k^t_{\mathrm{I}_{\alpha\beta}} \neq \bot$ and $k^t_{\mathrm{I}_{\beta\alpha}} \neq \bot$. The values to the left of the symbol "|" are called *material* values; the ones to the right are *virtual* values [3]. The exchange balance of stages of type II, $b^T_{\mathrm{II}_{\alpha\beta}}$, is defined analogously. The *general exchange balance* is given by

$$b^T_{\{\alpha,\beta\}} = b^T_{\mathrm{I}_{\alpha\beta}} + b^T_{\mathrm{II}_{\alpha\beta}}. \tag{4}$$

The *material results* $\mathbf{m}_{\alpha\beta}$ and $\mathbf{m}_{\beta\alpha}$, according to the points of view of α and β, respectively, of a social exchange process between such agents occurring during T are given by the sum of the material values involved in the process:

$$\mathbf{m}^T_{\alpha\beta} = r^T_{\mathrm{I}_{\alpha\beta}} + s^T_{\mathrm{II}_{\alpha\beta}} + r^T_{\mathrm{II}_{\alpha\beta}} + s^T_{\mathrm{I}_{\alpha\beta}}, \;\; \mathbf{m}^T_{\beta\alpha} = r^T_{\mathrm{I}_{\beta\alpha}} + s^T_{\mathrm{II}_{\beta\alpha}} + r^T_{\mathrm{II}_{\beta\alpha}} + s^T_{\mathrm{I}_{\beta\alpha}}. \quad (5)$$

Analogously, the *virtual results* $\mathbf{v}_{\alpha\beta}$ and $\mathbf{v}_{\beta\alpha}$ are given by:

$$\mathbf{v}^T_{\alpha\beta} = t^T_{\mathrm{I}_{\alpha\beta}} + v^T_{\mathrm{II}_{\alpha\beta}} + t^T_{\mathrm{II}_{\alpha\beta}} + v^T_{\mathrm{I}_{\alpha\beta}}, \;\; \mathbf{v}^T_{\beta\alpha} = t^T_{\mathrm{I}_{\beta\alpha}} + v^T_{\mathrm{II}_{\beta\alpha}} + t^T_{\mathrm{II}_{\beta\alpha}} + v^T_{\mathrm{I}_{\beta\alpha}}. \quad (6)$$

The *general results* takes into account all kinds of values is obtained by:

$$\mathbf{g}^T_{\alpha\beta} = \mathbf{m}^T_{\alpha\beta} + \mathbf{v}^T_{\alpha\beta}, \;\; \mathbf{g}^T_{\beta\alpha} = \mathbf{m}^T_{\beta\alpha} + \mathbf{v}^T_{\beta\alpha}. \quad (7)$$

A social exchange process between a pair of agents α and β is said to be in *equilibrium* around $s \in \mathbb{R}$ if, given a tolerance $\epsilon \in \mathbb{R}$ ($\epsilon \geq 0$), it holds that $\mathbf{g}^T_{\alpha\beta} \in \Theta_{s_\epsilon}$ and $\mathbf{g}_{\beta\alpha} \in \Theta_{s_\epsilon}$. The *material equilibrium* around $s \in \mathbb{R}$, with tolerance $\epsilon \in \mathbb{R}$ ($\epsilon \geq 0$), is achieved when $\mathbf{m}^T_{\alpha\beta} \in \Theta_{s_\epsilon}$ and $\mathbf{m}_{\beta\alpha} \in \Theta_{s_\epsilon}$. If a social exchange process between a pair of agents α and β is in equilibrium around 0, then the system is said to be in *equilibrium* in the sense of Piaget. This special case of equilibrium problem was subject of previous works [4, 5].

To extend these concepts to multi-agents systems composed by m agents, a matrix-like notation is introduced. An $m \times m$ interval $\star$-*matrix* $[x_{ij}]^\star$ is defined as the interval $m \times m$ matrix $[x_{ij}]$ where $x_{ij} = \star$ whenever $i = j$, that is:

$$[x_{ij}]^\star = \begin{pmatrix} \star & x_{12} & \cdots & x_{1m} \\ x_{21} & \star & \cdots & x_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ x_{m1} & x_{m2} & \cdots & \star \end{pmatrix}$$

A 2×2 $\star$-matrix is denoted schematically as an ordered pair

$$(x_{12}, x_{21})^\star. \quad (8)$$

For $X = [x_{ij}]^\star$ and $Y = [y_{ij}]^\star$, we define: *(i)* the *addition*: $X + Y = [x_{ij} + y_{ij}]^\star$; (ii) the *qualitative equivalence relation*: $X \approx_{s_\epsilon} Y \Leftrightarrow (\forall i \neq j)\ x_{ij} \approx_{s_\epsilon} y_{ij}$, for a given tolerance ϵ. Additionally, *(iii)* any $\star$-matrix $[n_{ij}]^\star$ such that $n_{ij} \in \Theta_{s_\epsilon}$ is called an s_ϵ-reference $\star$-matrix. The set of such $\star$-matrices is denoted by N_{s_ϵ}.

In a multi-agent system of m agents, the exchange values determined by the functions in (1) are given by the eight $m \times m \times \#T$ interval $\star$-matrices $\mathcal{K}_{\mathrm{I}} = \left[k^t_{\mathrm{I}_{\alpha\beta}}\right]^\star$ and $\mathcal{K}_{\mathrm{II}} = \left[k^t_{\mathrm{II}_{\alpha\beta}}\right]^\star$, called *investment* ($\mathcal{K} = \mathcal{R}$), *satisfaction* ($\mathcal{K} = \mathcal{S}$), *debt* ($\mathcal{K} = \mathcal{T}$) and *credit* ($\mathcal{K} = \mathcal{V}$) *matrices* for the exchange stages I and II, respectively, that occurred between each two agents α and β in T.

For $t' \in T$, $\mathcal{K}^{t'}_{\mathrm{I}} = \left[k^{t'}_{\mathrm{I}_{\alpha\beta}}\right]^\star$ and $\mathcal{K}^{t'}_{\mathrm{II}} = \left[k^{t'}_{\mathrm{II}_{\alpha\beta}}\right]^\star$ are $m \times m$ $\star$-matrices. For a time sequence $T = t_1, \ldots, t_n$, the matrices of *global* investment, satisfaction, debt and credit are given by $\mathcal{K}^T_{\mathrm{I}} = \sum_{t=t_1}^{t_n} \mathcal{K}^t_{\mathrm{I}}$ and $\mathcal{K}^T_{\mathrm{II}} = \sum_{t=t_1}^{t_n} \mathcal{K}^t_{\mathrm{II}}$, for $\mathcal{K} = \mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{V}$.

The exchange balance of stages of type I, given in (3), is represented by a tuple of $\star$-matrices: $B_{\mathrm{I}}^T = \left(\mathcal{R}_{\mathrm{I}}^T, \mathcal{S}_{\mathrm{I}}^T \mid \mathcal{T}_{\mathrm{I}}^T, \mathcal{V}_{\mathrm{I}}^T\right)$. Analogously, the exchange balance of stages of type II is represented as a tuple B_{II}^T . The general exchange balance given in (4) can be represented by the tuple $B^T = \left(\mathcal{R}^T, \mathcal{S}^T \mid \mathcal{T}^T, \mathcal{V}^T\right)$, where

$$\mathcal{K}^T = \mathcal{K}_{\mathrm{I}}^T + \mathcal{K}_{\mathrm{II}}^T, \text{ for } \mathcal{K} = \mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{V}. \tag{9}$$

The material, virtual and general results of social exchange processes, given in (5), (6), and (7), are evaluated, respectively, by $\mathbf{M}^T = \mathcal{R}^T + \mathcal{S}^T$, $\mathbf{V}^T = \mathcal{T}^T + \mathcal{V}^T$ and $\mathbf{G}^T = \mathbf{M}^T + \mathbf{V}^T$, where $\mathcal{R}^T, \mathcal{S}^T, \mathcal{T}^T, \mathcal{V}^T$ are given in (9). A multi-agent system is said to be in equilibrium around $s \in \mathbb{R}$, with tolerance ϵ, if $\mathbf{G}^T \in N_{s_\epsilon}$. It is in material equilibrium around $s \in \mathbb{R}$ when $\mathbf{M}^T \in N_{s_\epsilon}$.

3.4 Solving the Equilibration Problem Using a QI–MDP

The *equilibrium supervisor*, at each instant of time, analyzes the conditions and constraints imposed by the external and the internal environments, determines the target equilibrium point, evaluates the material results of exchange processes between each pair of agents and makes suggestions of exchanges for them, in order to keep the material results of exchanges in equilibrium. The equilibrium supervisor also takes into account the virtual results of the exchanges in order to decide which type of exchange stage he shall suggest for the two agents to realize. To achieve that purpose, the equilibrium supervisor models the exchanges between each pair of agents as simultaneous Markov Decision Processes (MDP) [9].

Consider a bound $L \in \mathbb{R}$ $(L > 0)$ for the set of real intervals $\mathbb{IR}_L$ and the set $S = \{-L, -L + \frac{L}{n}, -L + 2\frac{L}{n}, \ldots, L - 2\frac{L}{n}, L - \frac{L}{n}, L\} \subseteq \mathbb{N}$ of possible reference values induced on $\mathbb{IR}_L$ by a given $n > 0 \in \mathbb{N}$. Given a target equilibrium point $\omega \in \mathbb{R}$, a reference value $s \in S$ is chosen such that $\omega \in [s - \epsilon, s + \epsilon]$, for a given tolerance $\epsilon \in \mathbb{R}$ $(0 < \epsilon < \frac{L}{n})$ and machine numbers $s - \epsilon$ and $s + \epsilon$.

Let $\hat{E}_s = \{E_s^{-n-\frac{sn}{L}}, \ldots, E_s^{-1}, E_s^0, E_s^1, \ldots, E_s^{n-\frac{sn}{L}}\}$ be the set of $2n + 1$ equivalence classes of intervals $X \in \mathbb{IR}_L$, defined, for $i = -n - \frac{sn}{L}, \ldots, n - \frac{sn}{L}$, as:

$$E_s^i = \begin{cases} \{X \mid s + i\frac{L}{n} \leq mid(X) < s + (i+1)\frac{L}{n}\} & \text{if } -n - \frac{sn}{L} \leq i < -1 \\ \{X \mid s - \frac{L}{n} \leq mid(X) < s - \epsilon\} & \text{if } i = -1 \\ \{X \mid s - \epsilon \leq mid(X) \leq s + \epsilon\} & \text{if } i = 0 \\ \{X \mid s + \epsilon < mid(X) \leq s + \frac{L}{n}\} & \text{if } i = 1 \\ \{X \mid s + (i-1)\frac{L}{n} < mid(X) \leq s + i\frac{L}{n}\} & \text{if } 1 < i \leq n - \frac{sn}{L}. \end{cases} \tag{10}$$

The classes $E_s^i \in \hat{E}_s$ are the supervisor representations of classes of unfavorable $(i < 0)$, equilibrated $(i = 0)$ and favorable $(i > 0)$ material results of exchange processes, related to the reference interval s that approximates the target equilibrium point w. Whenever it is understood from the context, we shall denote by E^- (or E^+) any class $E_s^{i<0}$ (or $E_s^{i>0}$). The *accuracy* of the equilibrium supervisor is given by $\kappa_n = \frac{L}{n}$. The range of the midpoints of the intervals that belong to a class E_s^i is called the *representative* of the class E_s^i. Whenever it is clear from the context, we identify a class E_s^i with its representative.

The states of the QI–MDP model are $\star$-matrices $E = \left[E^i_{\alpha\beta}\right]^\star$, where each entry $E^i_{\alpha\beta} \in \hat{E}_s$ is the class representing the material results of the social exchange process between α and β, from the point of view of the agent α. For the analysis of the equilibrium, we shall consider each pair of co-related classes of material results $\left(E^i_{\alpha\beta}, E^j_{\beta\alpha}\right)$. The $\star$-matrix $\left[E^0_{\alpha\beta}\right]^\star$ is the *terminal* state, representing that the system is in equilibrium around s.

The *actions* are state transitions $\left[E^i_{\alpha\beta}\right]^\star \overset{\left[A^i_{\alpha\beta}\right]^\star}{\mapsto} \left[E^{i'}_{\alpha\beta}\right]^\star$, with $E^i_{\alpha\beta}, E^{i'}_{\alpha\beta} \in \hat{E}_s$ and $\left[A^i_{\alpha\beta}\right]^\star$ being an interval $\star$-matrix operator such that $mid\left(E^i_{\alpha\beta} + A_{\alpha\beta}\right) \in E^{i'}_{\alpha\beta}$, where each *interval action* $A^i_{\alpha\beta}$ should be of the following types:

- a *compensation interval*, denoted by C^i_s, of a class representative E^i_s;
- a *go-forward-k-step interval*, which is an interval, denoted by F^i_k, that transforms a class E^i_s into $E^{(i+k)\neq 0}_s$, with $i \neq L$;
- a *go-backward-k-step interval*, which is an interval, denoted by B^i_{-k}, that transforms a class E^i_s into $E^{(i-k)\neq 0}_s$, with $i \neq -L$.

The set $\mathcal{C}$ of compensation intervals is shown in Table 1. The set $\mathcal{F}$ of go-forward intervals and their respective effects are partially presented in Table 2. The set of go-backward intervals, denoted by $\mathcal{B}$, can be specified analogously.

For example, in the case of just two agents α and β, for the classes of material results (using the notation in (8)), with $-n - \frac{sn}{L} \leq i < -1$ and $1 < j \leq n - \frac{sn}{L}$:

$$\left(E^i_{\alpha\beta}, E^j_{\beta\alpha}\right)^\star \equiv \left(\left[s + i\frac{L}{n}, s + (i+1)\frac{L}{n}\right], \left[s + (j-1)\frac{L}{n}, s + j\frac{L}{n}\right]\right)^\star,$$

it follows that the *compensation–compensation* action $\left(C^i, C^j\right)^\star$ and the *go-backward*$_{-3}$–*go-forward*$_{+2}$ $\left(B^i_{-3}, F^j_{+2}\right)^\star$ action are specified by, respectively,

$$C^i = \left[-\frac{2i+1}{2}\frac{L}{n} - \epsilon, -\frac{2i+1}{2}\frac{L}{n} + \epsilon\right], \tag{11}$$

$$C^j = \left[\frac{(1-2j)}{2}\frac{L}{n} - \epsilon, \frac{(1-2j)}{2}\frac{L}{n} + \epsilon\right], \tag{12}$$

Table 1. Specification of compensation intervals

State	Compensation Interval $C^i \in \mathcal{C}$
$E^{i,-n\leq i<-1}_s$	$[-\left(\frac{2i+1}{2}\frac{L}{n}\right) - \epsilon, -\left(\frac{2i+1}{2}\frac{L}{n}\right) + \epsilon]$
E^{-1}_s	$[\frac{1}{2}\left(\frac{L}{n} + \epsilon\right) - \epsilon, \frac{1}{2}\left(\frac{L}{n} + \epsilon\right) + \epsilon]$
E^0_s	$[0, 0]$
E^1_s	$[-\frac{1}{2}\left(\frac{L}{n} + \epsilon\right) - \epsilon, -\frac{1}{2}\left(\frac{L}{n} + \epsilon\right) + \epsilon]$
$E^{i,1<i\leq n}_s$	$[\frac{(1-2i)}{2}\frac{L}{n} - \epsilon, \frac{(1-2i)}{2}\frac{L}{n} + \epsilon]$

Table 2. Specification of some go-forward intervals and their respective effects

State	Go-forward interval $F^i_{+k} \in \mathcal{F}$	Effect
$E_s^{i,-n-\frac{sn}{L} \leq i < -2}$	$\left[k\frac{L}{n} - \epsilon, k\frac{L}{n} + \epsilon\right]_{0<k\leq -i-2}$	$E_s^i \mapsto E_s^{i+k, i<i+k\leq -2}$
E_s^{-2}	$\left[\frac{L}{n} - \epsilon, \frac{L}{n} + \epsilon\right]$	$E_s^{-2} \mapsto E_s^{-1}$
$E^{i,-n-\frac{sn}{L} \leq i < -1}$	$\left[k\frac{L}{n} - \epsilon, k\frac{L}{n} + \epsilon\right]_{1-i\leq k\leq n-\frac{sn}{L}-i-1}$	$E_s^i \mapsto E_s^{i+k, 1<i+k\leq n-\frac{sn}{L}}$
E_s^{-1}	$\left[k\frac{L}{n} - \epsilon, k\frac{L}{n} + 2\epsilon\right]_{2\leq k\leq n-\frac{sn}{L}}$	$E_s^{-1} \mapsto E_s^{k-1, 1<k-1\leq n-\frac{sn}{L}}$
E_s^0	$\left[k\frac{L}{n}, (k+1)\frac{L}{n}\right]_{0<k\leq n-\frac{sn}{L}-1}$	$E_s^0 \mapsto E_s^{k+1, 1<k+1\leq n-\frac{sn}{L}}$
$E_s^{i, 1<i\leq n-\frac{sn}{L}}$	$\left[k\frac{L}{n} - \epsilon, k\frac{L}{n} + \epsilon\right]_{0<k\leq n-\frac{sn}{L}-i}$	$E_s^i \mapsto E_s^{i+k, i<i+k\leq n-\frac{sn}{L}}$

$$B^i_{-3} = \left[-3\frac{L}{n} - \epsilon, -3\frac{L}{n} + \epsilon\right], \tag{13}$$

$$F^j_{+2} = \left[2\frac{L}{n} - \epsilon, 2\frac{L}{n} + \epsilon\right], \tag{14}$$

resulting in the state transitions, with $-n - \frac{sn}{L} \leq i < -1$ and $1 < j \leq n - \frac{sn}{L}$:

$$\left(E^i_{\alpha\beta}, E^j_{\beta\alpha}\right)^\star \overset{(11,12)^*}{\mapsto} \left(E^0_{\alpha\beta}, E^0_{\beta\alpha}\right)^\star, \left(E^i_{\alpha\beta}, E^j_{\beta\alpha}\right)^\star \overset{(13,14)^*}{\mapsto} \left(E^{(i-3)}_{\alpha\beta}, E^{(j+2)}_{\beta\alpha}\right)^\star.$$

Given a target equilibrium point $w \in \mathbb{R}$ (which specify the reference value s), the equilibrium supervisor has to find, for each state $[E^i_s]^\star$ representing the actual material result, the action that shall achieve the terminal state $[E^0_s]^\star$ (representing that the system is in equilibrium around s) or, at least, another state from where the terminal state can be achieved, with the least number of steps.[2] Such action generates an *optimal exchange recommendation*, consisting of a partially defined exchange stage that the agents are suggested to perform. This partial definition shall be completed by the analysis of the virtual results, which allows the specification of which particulary types of exchange stages (I or II) should be considered. However, since the agents are autonomous, they may not follow the recommendations exactly. This means that there is a probability that the system achieves another state different from the one expected (or suggested) by the supervisor. Even if the agents follow a recommendation exactly, we will show that the effect may not be the expected by the supervisor.

For a given tolerance $0 \leq \epsilon < \kappa_n = \frac{L}{n}$, where κ_n is the equilibrium supervisor accuracy, we define:

Definition 1. *A* Qualitative Interval Markov Decision Process (QI–MDP)*, for keeping the social exchanges in a multi-agent system in equilibrium around a reference interval s, is a tuple $\langle \mathbf{E}_s, \mathbf{A}, \mathbf{H}, \mathbf{R}\rangle^{L,n}_{\epsilon}$, where:*

[2] We observe that the choice of such actions are also regulated by the rules of the social exchanges, and, therefore, there are some state transitions that are not allowed.

(i) *The set of the states of the model is the set of $m \times m$ star-matrices*

$$\mathbf{E}_s = \{\left[E^i_{\alpha\beta}\right]^\star \mid E^i_{\alpha\beta} \in \hat{E}_s\}$$

of classes of material results as specified in (10).

(ii) *The set of the actions of the model is the set of $m \times m$ star-matrices*

$$\mathbf{A} = \{\left[A^i_{\alpha\beta}\right]^\star \mid A^i_{\alpha\beta} \in \mathcal{C} \cup \mathcal{F} \cup \mathcal{B}\},$$

of compensation $C^i \in \mathcal{C}$, go-forward $F^i_{+k} \in \mathcal{F}$ and go-backward $B^i_{-k} \in \mathcal{B}$ intervals.

(iii) $\mathbf{H} : \mathbf{E}_s \times \mathbf{A} \to \Pi(\mathbf{E}_s)$ *is the state-transition function, that gives for each state and each action, a probability distribution over the set of states;*

(iv) $\mathbf{R} : (\mathbf{E}_s \times \mathbf{A}) \to \mathbb{R}$ *is the reward function, giving the expected immediate reward gained by choosing an action* $\left[A^i_{\alpha\beta}\right]^\star$ *when the current state of the model is* $\left[E^i_{\alpha\beta}\right]^\star$.

A sample reward function $\mathbf{R} : (\mathbf{E} \times \mathbf{A}) \to \mathbb{R}$ that conforms to the idea of supporting a recommendation function that is able to direct pairs of agents into social equilibrium is partially sketched in Table 3.[3] This particular function illustrates various requirements that should be satisfied by all reward functions of the model. Observe, for instance, that if the current state is of the type $(E^-_s, E^+_s)^\star$, then the best action to be chosen is the *compensation-compensation* action $(C, C)^\star$, which results in a state transition $(E^-_s, E^+_s)^\star \mapsto (E^0_s, E^0_s)^\star$. Any other choice will make the agents either take a long way to the equilibrium or get away from it. On the other hand, if the current state is of type $(E^-_s, E^-_s)^\star$, then a *compensation-compensation* action $(C, C)^\star$ would generate a recommendation of agent exchanges of *satisfaction-satisfaction* type, which is impossible according to the model of social interactions [3], since it is impossible for an agent to get a satisfaction value from no service at all. The function R states that $(C, C)^\star$ is a very bad action to be chosen in such situation.

Table 3. Partial schema of the reward function R for the case of two interacting agents

R	(C, C)	$(0_\epsilon, C)$	$(C, 0_\epsilon)$	(B_{-1}, F_{+1})	(B_{-3}, F_{+3})	(F_{+1}, B_{-1})	(C, B_{-1})	(B_{-3}, C)
(E^-, E^+)	30	20	-30	-5	-10	3	20	20
(E^+, E^+)	30	20	20	0	0	0	18	20
(E^-, E^-)	-30	-30	-30	30	0	30	28	26

Any optimal policy $\pi^* : \mathbf{E} \to \mathbf{A}$ should satisfy the set of requirements expressed by the schema partially sketched in Table 4 (for any pair of agents α and β). Notice that it is a non deterministic policy.

[3] The notation given in (8) was used in the tables 3, 4 and 5, omitting the symbol $\star$ of the $\star$-matrices, for simplicity.

Table 4. Partial schema of the optimal policy π^* for the case of two interacting agents

State	Action	State	Action
$(E^i, E^j)^{1<j\leq n}_{-n\leq i<-1}$	(C^i, C^j)	$(E^i, E^0)_{-n\leq i<-1}$	$(F^i_{+(-i+1)}, B^0_{-1})$
$(E^i, E^1)_{-n\leq i<-1}$	(C^i, B^1_{-1})	(E^{-1}, E^0)	(F^{-1}_{+1}, B^0_{-1})
(E^{-1}, E^1)	(F^{-1}_{+1}, B^1_{-1})	$(E^i, E^{-1})_{-n\leq i<-1}$	$(F^i_{+(-i+1)}, B^{-1}_{-1})$
$(E^j, E^i)^{1<j\leq n}_{-n\leq i<-1}$	(C^j, C^i)	$(E^{-1}, E^j)_{-n\leq j<-1}$	(F^{-1}_{+2}, B^j_{-1})
$(E^1, E^i)_{-n\leq i<-1}$	(B^1_{-1}, C^i)	(E^{-1}, E^{-1})	$(F^{-1}_{+2}, B^{-1}_{-1})$
$(E^j, E^{-1})_{1<j\leq n}$	(C^j, F^{-1}_{+1})	$(E^i, E^j)_{-n\leq i,j<-1}$	$(B^j_{-1}, F^i_{+(-i+1)})$ or
(E^1, E^{-1})	(B^1_{-1}, F^{-1}_{+1})		$(F^i_{+(-i+1)}, B^j_{-1})$

Table 5. Partial schema of the optimal value recommendation ρ_{π^*}

State	Optimal policy	Recommendation	Label
$(E^i, E^j)^{1<j\leq n}_{-n\leq i<-1}$	$(C^i > 0, C^j < 0)$	$((r_{\beta\alpha}, C^j), (s_{\alpha\beta}, C^i))$	R_1
$(E^i, E^j)_{1<i,j\leq n}$	$(C^i < 0, C^j < 0)$	$((r_{\alpha\beta}, C^i), (s_{\beta\alpha}, C^j))$ or $((r_{\beta\alpha}, C^j), (s_{\alpha\beta}, C^i))$	R_2 R_3
$(E^0, E^j)_{1<j\leq n}$	$(0_\epsilon, C^j < 0)$	$((r_{\beta\alpha}, C^j), (s_{\alpha\beta}, 0_\epsilon))$	R_4
$(E^0, E^i)_{-n\leq i<-1}$	$(B^0_{-1} < 0, F^i_{+(-i+1)} > 0)$	$((r_{\alpha\beta}, B^0_{-1}), (s_{\beta\alpha}, F^i_{+(-i+1)}))$	R_5
$(E^{-1}, E^j)_{1<j\leq n}$	$(F^{-1}_{+1} > 0, C^j < 0)$	$((r_{\beta\alpha}, C^j), (s_{\alpha\beta}, F^{-1}_{+1}))$	R_6
$(E^1, E^i)_{-n\leq i<-1}$	$(B^1_{-1} < 0, C^i > 0)$	$((r_{\alpha\beta}, B^1_{-1}), (s_{\beta\alpha}, C^i))$	R_7
(E^{-1}, E^1)	$(F^{-1}_{+1} > 0, B^1_{-1} < 0)$	$((r_{\beta\alpha}, B^1_{-1}), (s_{\alpha\beta}, F^{-1}_{+1}))$	R_8
$(E^1, E^{-1})_{1<j\leq n}$	$(B^1_{-1} < 0, F^{-1}_{+1} > 0)$	$((r_{\alpha\beta}, B^1_{-1}), (s_{\beta\alpha}, F^{-1}_{+1}))$	R_9
$(E^i, E^1)_{-n\leq i<-1}$	$(C^i > 0, B^1_{-1} < 0)$	$((r_{\beta\alpha}, B^1_{-1}), (s_{\alpha\beta}, C^i))$	R_{10}
(E^{-1}, E^0)	$(F^{-1}_{+1} > 0, B^0_{-1} < 0)$	$((r_{\beta\alpha}, B^0_{-1}), (s_{\alpha\beta}, F^{-1}_{+1}))$	R_{11}
(E^0, E^{-1})	$(B^0_{-1} < 0, F^{-1}_{+1} > 0)$	$((r_{\alpha\beta}, B^0_{-1}), (s_{\beta\alpha}, F^{-1}_{+1}))$	R_{12}
$(E^i, E^j)_{-n\leq i,j<-1}$	$(F^i_{+(-i+1)} > 0, B^j_{-1} < 0)$ or $(B^j_{-1} < 0, F^i_{+(-i+1)} > 0)$	$((r_{\beta\alpha}, B^j_{-1}), (s_{\alpha\beta}, F^i_{+(-i+1)})$ or $((r_{\alpha\beta}, B^j_{-1}), (s_{\beta\alpha}, F^i_{+(-i+1)})$	R_{13} R_{14}

The *optimal value recommendation* associated to an optimal policy π^* is a $\star$-matrix operator ρ_{π^*} that gives, for each state $\left[E^i_{\alpha\beta}\right]^\star$ and optimal action $\pi^*\left[E^i_{\alpha\beta}\right]^\star = \left[A^i_{\alpha\beta}\right]^\star$, partial definitions of recommended exchange stages, consisting of $\star$-matrices whose elements in symmetric positions are either $\left(r_{\alpha\beta}, A^i_{\alpha\beta}\right)$ and $\left(s_{\beta\alpha}, A^j_{\beta\alpha}\right)$, or $\left(s_{\alpha\beta}, A^i_{\alpha\beta}\right)$ and $\left(r_{\beta\alpha}, A^j_{\beta\alpha}\right)$, where $(r_{\lambda\delta}, W)$ means the realization, by the agent λ, of a service with investment value $W < 0$, and $(s_{\delta\lambda}, W')$ means δ's satisfaction with interval value W', for receiving the service. The optimal value recommendation ρ_{π^*}, corresponding to the the optimal policy shown in Table 4, is partially sketched in Table 5.

Finally, the equilibrium supervisor has to decide which types of exchange stages (I or II) should be recommended. This is done by the analysis of the virtual results from the points of view of each pair of agents α and β (see (6)):

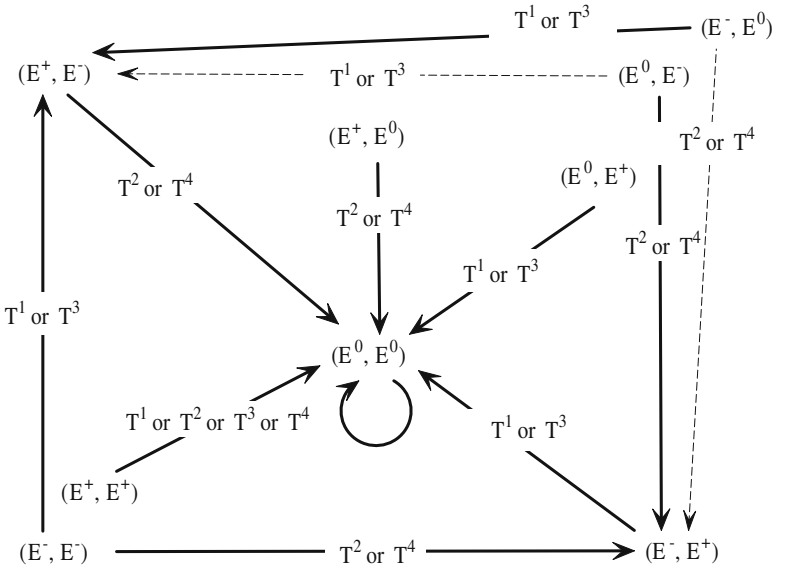


Fig. 2. Effects of stage and optimal value recommendations (simplified diagram)

- If $\mathbf{v}_{\alpha\beta} > 0$, then α is able to charge β the credit for services previously done. In this case, an exchange stage T^1 of type $\mathrm{II}_{\alpha\beta}$ should be recommended .
- If $\mathbf{v}_{\beta\alpha} > 0$, then it is the case that the agent β can charge α the credit for services previously done, indicating that an exchange stage T^2 of type $\mathrm{II}_{\beta\alpha}$ should then be recommended.
- If $\mathbf{v}_{\alpha\beta} \leq 0$, then the agent α does not have any credit to charge α. Therefore, the service done by the agent β must be spontaneous. In this case, an exchange stage T^3 of type $\mathrm{I}_{\beta\alpha}$ should then be recommended.
- If $\mathbf{v}_{\beta\alpha} \leq 0$, then the agent β does not have any credit to charge β, resulting that an exchange stage T^4 of type $\mathrm{I}_{\alpha\beta}$ should then be chosen.

Stage recommendations and their combined effects with the optimal value recommendations are sketched in the simplified state transition diagram shown in Fig. 2. The dot lines represent alternative paths that were not considered as optimal recommendations since they may seem unfair according to social rules.

3.5 Analysis of the Stability of the Model

The analysis of the stability of the model is concerned with the reachability of the terminal state in terms of number of steps that are necessary to achieve the equilibrium whenever it is perturbed. For that, we consider only the case in which the *agents always follow the recommendations given by the equilibrium supervisor*. We show that, even in this favorable case, the decision process is a non-trivial one, due the qualitative nature of exchange values and to the restrictions imposed by the definition of exchange, that always requires a service to be

done in any exchange stage. However, we show that under some conditions, it is always possible to have the system equilibrated in *at most four steps*.

Let $\mathrm{M}^\tau = (\mathbf{m}^\tau_{\alpha\beta}, \mathbf{m}^\tau_{\beta\alpha})^\star$ be the material results of a social exchange process performed by the agents α and β, at step τ. For a given tolerance ϵ, equilibrium supervisor accuracy $\kappa_n \frac{L}{n}$, and a reference value s, the following results hold:

Proposition 2. *(i) If* $\mathbf{m}^0_{\alpha\beta} \in E^{-1}_s$ *and* $\mathbf{m}^0_{\beta\alpha} \in E^1_s$*, then the system achieves the equilibrium in one step if and only if* $1 < \frac{\kappa_n}{\epsilon} \leq 3$*. (ii) If* $\mathbf{m}^0_{\alpha\beta} \in E^i_s$*, with* $1 < i \leq n$*, then it is possible to get* $\mathbf{m}^\tau_{\alpha\beta} \in E^0_\alpha$ *in at most* $\tau = 2$ *steps if and only if* $1 < \frac{\kappa_n}{\epsilon} \leq 3$*. (iii) If* $\mathbf{m}^0_{\beta\alpha} \in E^i_s$*, with* $-n \leq i < -1$*, then it is possible to get* $\mathbf{m}^\tau_{\beta\alpha} \in E^0_\beta$ *in at most* $\tau = 2$ *steps if and only if* $1 < \frac{\kappa_n}{\epsilon} \leq 3$*. (iv) If* $\mathbf{m}^0_{\alpha\beta} \in E^i_s$*, with* $1 < i \leq n$ *and* $s + \frac{2i+1}{2}\frac{L}{n} - \epsilon \leq mid\left(\mathbf{m}^0_{\alpha\beta}\right) \leq s + \frac{2i+1}{2}\frac{L}{n} + \epsilon$*, then* $\mathbf{m}^1_{\alpha\beta} \in E^0_\alpha$.

It follows that an individual transition from E^i, with $1 < i \leq n$ or $-n \leq i < -1$, to the equilibrium can be done in at most two steps ($E^i \mapsto E^1$(or E^{-1}) $\mapsto E^0$). However, in any interaction between two agents, combined transitions departing from a state $(E^i_s, E^j_s)^\star$ or $(E^j_s, E^i_s)^\star$, with $1 < i \leq n$ and $-n \leq j < -1$, may result in a state different from $(E^1_s, E^{-1}_s)^\star$, $(E^{-1}_s, E^1_s)^\star$ or $(E^0_s, E^0_s)^\star$. The worst case is when the system is in the state $(E^i_s, E^j_s)^\star$, with $-n \leq i, j < -1$, since two simultaneous positive compensation actions are not allowed. In this case, the optimal recommendation (Table 5) leads the agents to equilibrium in at most four steps, by one of the following transitions:

$$
\begin{aligned}
(E^i, E^j)^\star_{-n \leq i,j < -1} &\overset{R13}{\mapsto} (E^1, E^j)^\star_{-n \leq j < -1} \overset{R7}{\mapsto} (E^0, E^{-1})^\star \\
&\overset{R12}{\mapsto} (E^{-1}, E^1)^\star \overset{R8}{\mapsto} (E^0, E^0)^\star \\
(E^i, E^j)^\star_{-n \leq i,j < -1} &\overset{R14}{\mapsto} (E^j, E^1)^\star_{-n \leq j < -1} \overset{R10}{\mapsto} (E^{-1}, E^0)^\star \\
&\overset{R11}{\mapsto} (E^1, E^{-1})^\star \overset{R9}{\mapsto} (E^0, E^0)^\star.
\end{aligned}
$$

4 Conclusion

We proposed a model, based on Interval Mathematics and Markov Decision Process, for the self-regulation of exchanges in multi-agent systems, using exchange values and social rules to structure the self-regulating mechanism.

We note that the work presents just a mechanism for the establishment of self-regulation. Motivations for adopting it are not contemplated in the mechanism itself: agents should find in other grounds the motivation for such adoption.

Immediate future work will be concerned with the case of an equilibrium supervisor that is not able to fully determine the material balance of social exchange processes with complete reliability (i.e., it is not allowed to know all the exchange values of the two agents). In this case, a partially observable Markov decision process (POMDP) shall be considered (see, p.ex., [10]), since the equilibrium supervisor shall be able to make external observations (also probabilistic) to help him to decide about the recommendations.

Further future work will deal with the internalization, in each agent of the system, of the model decision process introduced in the paper, so that the mechanism of exchange values-based social control that it supports can be performed in a decentralized way. It is expected that such decentralized social control mechanism can be modelled as a qualitative version of the Multiagent MDP [11].

Acknowledgments. This work has been partially supported by CNPq and FAPERGS. The authors thank the referees for their valuable comments.

References

1. Castelfranchi, C.: Engineering social order. In Omicini, A., Tolksdorf, R., Zambonelli, F., eds.: Engineering Societies in the Agents World. Springer, Berlin (2000) 1–18
2. Homans, G.C.: The Human Group. Harcourt, Brace & World, New York (1950)
3. Piaget, J.: Sociological Studies. Routlege, London (1995)
4. Dimuro, G.P., Costa, A.R.C.: Interval-based markov decision processes for regulating interactions between two agents in multi-agent systems. In Dongarra, J., Madsen, K., Wasniewski, J., eds.: Post-Proceedings of the Workshop on State-of-the-Art in Scientific Computing, PARA'04, Lyngby, Lecture Notes in Computer Science, Berlin, Springer (2004) (to appear).
5. Dimuro, G.P., Costa, A.R.C., Palazzo, L.A.M.: Systems of exchange values as tools for multi-agent organizations. Journal of the Brazilian Computer Society **11** (2005) 31–50 (Sichman, J., Dignum, V., Castelfranchi, C. (eds.), Special Issue on Agents' Organizations).
6. Rodrigues, M.R., Costa, A.C.R., Bordini, R.: A system of exchange values to support social interactions in artificial societes. In: Proceeding of the 2nd International Conference on Autonomous Agents and Multiagents Systems, AAMAS'03, Melbourne (2003) 81–88
7. Rodrigues, M.R., Costa, A.C.R.: Using qualitative exchange values to improve the modelling of social interactions. In Hales, D., Edmonds, B., Norling, E., Rouchier, J., eds.: Proceedings of the 4th Workshop on Agent Based Simulations, MABS'03, Melbourne, Lecture Notes in Artificial Intelligence 2927, Berlin, Springer (2003) 57–72
8. Moore, R.E.: Methods and Applications of Interval Analysis. SIAM, Philadelphia (1979)
9. White, D.J.: Markov Decision Processes. Wiley, New York (2002)
10. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observabe stochastic domains. Artificial Intelligence **101** (1998) 99–134
11. Boutilier, C.: Sequential optimality and coordination in multiagent systems. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI'99, Stockholm (1999) 478–485

A New Protocol to Share Critical Resources by Self-organized Coordination

Frederic Armetta, Salima Hassas, and Simone Pimont

LIRIS, Nautibus, 8 Bd Niels Bohr,
Université Claude Bernard-Lyon 1,
43 Bd du 11 Novembre F-69622, Villeurbanne

Abstract. In this paper we propose a new approach to share critical resources through self-organized coordination. Our approach addresses two levels : first, it allows the expression of agents needs in terms of resources occupancy (getting/leaving a resource) with respect to agents objectives; second, it allows to find a global resources exchange scheme between agents, fitting the agent needs thanks to a negotiation process. Negotiation between agents is held in a decentralized way and allows to produce complex contracts. The underlying protocol is applied to schedule manufacturing tasks on a set of critical machines.

Keywords: critical resources sharing, self-organization, emergent behavior, swarm intelligence, manufacturing scheduling.

1 Introduction

Sharing critical resources is a large scale problem. It is well represented in heterogeneous domains like the management of critical communication nets, supply chains scheduling, processes scheduling, etc. Providing a solution has to take into account dynamic aspects of the problem expression in its domain: how to adapt the current solution face to disturbances ? In this paper we propose a new protocol to share critical resources through self-organized coordination. Our approach addresses two levels: first, it allows the expression of agents needs in terms of resources occupancy (getting/leaving a resource) with respect to agents objectives; second, it allows to find a global resources exchange scheme between agents, fitting the agent needs thanks to a negotiation process. Negotiation between agents is held in a decentralized way and allows to produce complex contracts. In section 2, we describe our general problematic. In section 3, we present existing approaches and motivate our choice. We identify drawbacks and lacks for the selected approach, in the context of very constrained situations. To enhance the selected approach capability for the constrained sharing of critical resources problem we propose a new model in section 4. We illustrate our model by some simulations in section 5 and present first results, before concluding the paper with a discussion and the presentation of our future works in 6.

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 90–103, 2006.

2 Presentation of the General Problem: The Sharing of Critical Resources in a Constrained Environment

The problem of critical resources sharing is considered in a repairing context: every time a disturbance occurs one has to adapt the current solution. We choose to manage the resources use in advance. Let's us consider a set of agents that have to be alloted on a set of critical resources. Assignment of agents on resources must fit both the problem constraints and the agents objectives. For example, in a scheduling problematic agents should respect a chronological order (constraints) and a due date for their allocation (objective).

The described problem seems to be very difficult . The solutions search space size is very important. Each agent could acquire several resources and each resource could be allocated to several agents. In this paper, we consider different approaches to address this problem. We then observe lacks of multi-agents based approaches in adequately tackeling the resource sharing problem.

3 Addressing This Problem: A Contextual Approach

3.1 Dimensions Through Which the Problem Could Be Addressed

To address this problem one has to answer to three main questions:

- how to organize the covering of the set of potential solutions ?
- how to reduce the complexity ?
- how to organize the solving mechanism ?

Moving mode through the search space. The way to cover the search space could either be achieved through a predefined mode, where one can compute the set of possibilities in a pre-stated order, or through a non pre-established order when the number of possibilities could not be estimated or characterized a priori.

Guides to cover the search space. An efficient covering of search spaces is a well known problem in optimization. Two main approaches allow to reduce the complexity of the exploration process.

First, we could reduce the size of the search space by using domain specific knowledge. This process is called branch cutting, in a branch and bound exploration mode and some heuristics could efficiently reduce the search space exploration. One can use knowledge based approaches for specific application domains. Using some mathematical analysis can prove the lack of relevance for sub-parts of the space of search.

Second, we can take advantage of a process that manifests a naturally good behaviour. With approaches known as evolutionary , such like ant colonies optimization, simulated annealing, neuronal nets, immune-based systems, genetic algorithms, particle swarm optimization, etc.

Centralized versus decentralized. The set of possible guides to cover the search space hardly depends on the selected organization structure:

- centralized: there is only one active process. It controls all the data, it makes all the decisions in a centralized way.
- decentralized: several processes act separately. They have to cooperate in a decentralized way.

3.2 Contribution for the Coordination

An adequate approach. Our study domain is concerned by complex systems approaches. For such systems, composed of a set of interrelated entities, each entity's evolution impacts on the other entities evolution. Dynamics of such systems is non linear due to retroactives interactions making the system global behaviour hard to understand and control. The scheduling repairing problem exhibits some characteristics of complex systems. Indeed, for this problem, repercussions of each change occurring in the scheduling repairing process, are propagated through the whole system (see figure 7). The question is thus, how to control the system face to disturbances? The Multi-agents paradigm provides an interesting tool to represent this problem. Each agent embodies a system entity, and interact with the set of other agents in order to emphasize good global characteristics for the system. This challenge is not easy but appears almost natural: rather than finding efficient ways to reduce the time consuming for the problem solving, we attempt to understand the characteristics of the problem behaviour and then to lead the system evolution. In our approach, the current solution evolves through a non pre-established covering in a decentralized manner.

Description of the proposed approach. Our approach is characterized by the following:

- Self-organization: Process through which the organization of a system spontaneously increases, without external control. This organization evolves on the opposite way of the system's entropy increasing. Energy inside the system is continuously dissipated through the system's components to maintain some organizational structures.
- Stigmergy: Indirect mode of communication, mediated by the environment. This principle has been first introduced by Grassé [5], while studying the behavior of social insects. For our problem, we use stigmergy as a mean for agents to elaborate complex contracts in a fully decentralized manner.
- Eco-resolution: Paradigm first introduced by J. Ferber in 1989 [4]. It bases the resolution of a problem on a mechanism of a state space search, by reactive agents guided by a satisfaction fitness function. We use the eco-resolution as the mean to achieve self-organization. Each agent tries to get a set of resources fitting its needs.

Known protocols for multi-agents negotiations. In the following, the described approaches are generally concerned by our application domain (scheduling). Nevertheless, the principle exhibited by these ones are not limited to a

specific domain and fit the general problem of constrained critical resources sharing. For the scheduling problem, we distinguish two kind of approaches:

- to direct the products, covering the production system, at each their manufacturing step, without any prediction of the system capacity in the course of time.
- to anticipate the covering for products. To address this problem, we consider more complex contracts and additional constraints.

The second one is much difficult to realize but in [2], we show that this approach allows making decision in a less blind manner, in order to respect constraints such as the respect of due dates, congestions reduction, etc.

In [3], objects are dispatched over critical resources on which they consume a part of time. This approach which improves the system performances, reproduces the labor division in social insects. The paper authors explain how it can be useful to represent the problem in the course of time to make decisions less blindly. However, this is not yet yet achieved in the proposed approach. What are the difficulties that prevent the anticipation of allocations ? In [2], blinded decision can decrease the solving process performances. Each time an agent succeeds in changing its assigned resource, it disturbs the system equilibrium. Decentralized approaches are susceptible to chaotic behavior and sub-optimality. In [8] we show how *the agent activity can slow down the general solving process* when it is not controlled for resource sharing problems. This is the first subject of interest for our problematic.

[11] defines the contract-net protocol as a process of decentralized negotiation, for tasks allocations on critical resources, based on the sending of a call for proposals/solutions. It is useful for a consumer agent to request supplier agents. However, is this protocol appropriated when it is necessary to establish contracts between several agents at the same time in a complex context ? How to represent the complex characteristics of the problem with a two-by-two exchanges based protocol ? This will be the second subject of interest for our problematic. Generally, one use the Contract-Net protocol to assign product to machines in the course of time. Doing so, it is possible to use heuristics to improve the system performance. In [12] and [13] the authors consider some cooperative cues between products competing for an assignment to increase the system performance.

To tackle much complex problems considering more constraints (for instance, to make a schedule by anticipation), it is necessary to adapt the Contract-Net protocol. in [10], a generic negotiation model is presented. Each agent negotiates in a decentralized way so as to obtain a set of critical resources. Rules define the set of solutions to explore. We notice that for this negotiation model, conflicts between two negotiations are not considered (the negotiation with the highest priority wins and blocks the other ones). This negotiation model corresponds to a search space exploration with backtracking when a rule fails, and using a rule that determines the next rule to apply. Nevertheless, the proposed approach could not address the issue of: *how to elaborate complex contracts in a global negotiating context ?*

Some studies are based on parallelising negotiations between agents. In this case, the issue is: how could agents increase their capacity to negotiate with several other ones at the same time ? Then, one has to detect the failure for some agents to participate in a negotiation process, to prevent agents to be blocked, etc ([1]). In [7], agents attempt to make flexible contracts, in order to make contracts evolve during the negotiation process. These approaches are applied to e-commerce.

In all these works on Contract-Net based negotiations, enhancement has been done on the protocol itself rather than on the approach to address complexity. The question is thus: Does the Contract-Net architecture satisfy the problematic requirements ?

The process described in [6] is based on a self-organizing behaviour. Local interactions between agents allow the obtention of a global problem solving. However, the perceived improving must be significant when an agent changes its resource using. Indeed it blindly impacts on the overall system,because of the complex characteristics of the problem.

Difficulties and propositions. We exhibited in 3.2 two subjects of interest for our problematic. 1) The agent activity can slow down the general resolving and 2) the contracts elaboration should be done in a global negotiating context. How to tackle them efficiently ?

First, it is not easy to make the solution evolve in a decentralized way for constrained problems. The global solution improvement in decentralized systems are based on local decisions making. For instance, an agent can change its plans to enhance its profit. This kind of local decision propagates perturbations through the whole system which has to reach stability again. during a processes of resources acquiring/realizing by a set f agents, at the end of a moving sequence, we can evaluate the profit or the loss of the agent activity in the system. As the search space covering is not pre-established, if the solution quality decreases there is no way to go back. The more better states are local the more it is hard for the system to efficiently get around a state. For the sharing resource problems, bad moves are sanctioned by a high decreasing in the solution quality. Good solution states are insulated and very complex to meet. Then, we propose to elaborate complex contracts in a dedicated environment in order to validate contracts in a global context. Then, we can evaluate the impact of a contract in advance and prevent mistakes. We propose to centralise the contracts validation. Nevertheless, we propose to fully decentralize the contracts elaboration process.

Second, it's difficult to elaborate complex contracts among agents. There is a very important set of possible contracts. It is necessary to consider relations between contracts under construction. Note that none of the studied approaches fully express relations between contracts to elaborate. To do so, we need a specific architecture to represent these relations. We propose to elaborate contracts using a stigmergic negotiation. Doing so, each agent perceives it's local negotiating environment (the agreements of partners agents, etc.) and expresses it's needs to other agents. This process prevents the latency times during the negotiation, and allows to easily represent relations between contracts and to make them evolve, etc.

4 The Coordination Model Proposed

In this section, we present a generic model of coordination. Relying on it, we attempt to enhance the relevance of moves in the search space. In [9] Parunak's team shows that moderating agent activity results in increasing the system performances. Indeed, the agent activity can prevent the stabilization of the multi-agent system on a good quality solution. Instead of reducing this activity, we propose a model to check the relevance of each agent activity. Our model allows to foresee both positive and negative aspects of each agent action. In this paper, we focus on how to elaborate an efficient collaboration between agents. The described model will be applied in real-time problems for further works.

In 3.2, we described two drawbacks in the proposed approach for the sharing of critical resources. First, we show that the system has not to decide blindly during its evolution. Thus, each agent move will be validated in a global context before being effective. Second, we underline the difficulties to elaborate collaboration between agents. We choose to use reactive behaviors inspired from insect societies allowing local and global collaborations between agents. The global collaboration pattern will have to emerge instead of to be a priori stated. In 4.1, we describe the agent model. In 4.2, we present the way to elaborate collaborations, in 4.3 we explain how they are validated or not by the system.

4.1 The Agent Model

For our problem, an amount of critical resources is assigned to each agent, with respect to the agents objectives. To do so, an agent can exchange some of its resources with other agents. The described model acts at two levels: a collaboration level where plans are elaborated and an application level where the elaborated plans are applied.

On the application level, the real time situation of agents is represented: what are their associated part of critical resources ? what are their current objectives ? This level represents the state of the system and it is a part of the problem search space.

On the collaborative level is represented, the capacity for agents to exchange parts of resources, to collaborate in order to get a global satisfaction. The aimed resolution process is represented at this level.

4.2 The Collaboration Principle

In this section, we study the way by which collaborations are formed. A collaboration consists in an exchange of critical resources between agents. We focus on local collaborations, that can lead to global cooperations described in 4.3. The global cooperation constitutes a global complex contract. Local potential collaborations provide raw material for the global collaboration.

Intentions communication. Each agent has to communicate its intentions to the collaborative level. We define two kinds of intentions:

– Intention to get a resource: an agent can associate to this intention its preference based on its associated satisfaction.
– Intention to leave a resource: an agent indicates that it could leave its actual resource to improve the system (non selfish behavior).

An agent expresses only one intention for each of the concerned resources. However, it can express several intentions of getting, for each of the resources it needs. These marked intentions are used for the elaboration of local potential collaborations.

Local potential collaborations elaboration. Local collaborations are created by a process from the collaborative level while using the marked intentions. This process regularly associates complementary intentions but do not evaluate their relevance. For instance, in Figure 1, intentions to get and to leave resource $R1$ from $A1$ and $A2$ are associated in a simple local collaboration.

The local collaboration pattern can be more complex. For instance, if a resource can be shared by several agents, a collaboration can be formed by N agents leaving the resource and M agents getting it. That is the case for the scheduling problem.

The set of local collaborations so considered constitutes the raw material for the global collaboration. Agents are committed in contradictory potential collaborations. They could not respect all their potential commitments at the same time. In Figure 2, $A1$ can't give its leaving resource to agents from $Collab1$ and $Collab2$ at the same time. In the same way, it doesn't intend to get more resources that it needs to be fully satisfied. Our approach easily accommodates if all constraints are not met during the resolution process. It allows to find a

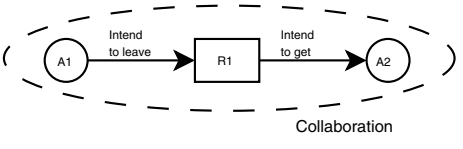


Fig. 1. Collaboration

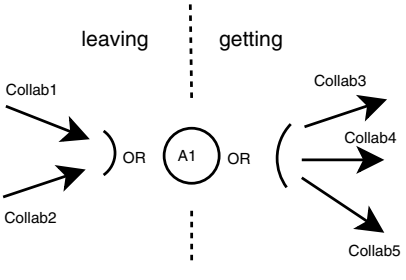


Fig. 2. Collaboration network

solution on a larger range, using a rich raw material representing a very big space of search. Other approaches attempt to elaborate fully coherent collaborations, which is very difficult to achieve. Since there is a wide space to explore, we represent the solution in a collaborations network as follows.

4.3 Dynamic Collaborations Net

Once the collaboration net is elaborated, a process inspired from the swarm intelligence carries out the final plan to apply to the application level. The two processes (to elaborate the net and to apply a resolution process on it) could be realized in a parallel way thanks to the decentralized aspect of the negotiation process.

Collaborations general principle. We attempt to make a global collaboration, that has to maximize the amount of respected constraints, emerge. The objective of the collaborative level is to enhance the coherence of the agent resource exchanges. We allow the temporarily constraints violation. We represent them as disturbances that could be propagated on the application level. The collaborative level must gradually provide plans that reduce the amount of agents that doesn't meet their objectives. However, the most the emerging process is powerful, the less disturbances are propagated on the application level. By using the swarm intelligence, we attempt to cope the complexity of the problem. The stigmergic negotiating process has to give good characteristics for the coordination of local and global behaviours thanks to the stigmergic communication.

In Figure 3 is represented the result of collaborations evolving in the dynamic net. Some collaborations are validated by the system, some are not.

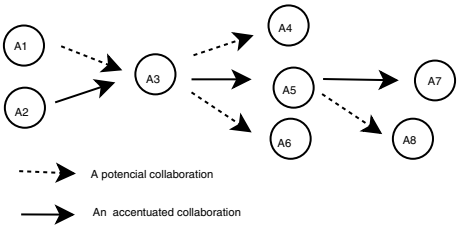


Fig. 3. Collaboration network

Global sense based on local activity of agents. Since we attempt the emerging process to fit the system objectives, the resolution process can't be local. The resources are shared on the overall system. Local resolutions have to influence global evolutions. Note that the opposite is true because global evolution is based on local ones. Using a decentralized system, the emerging organization results from local interactions.

An emerging collaboration comes out from the collaboration net. For exemple, in Figure 3 a part of the result of the emerging organization is represented. The path formed by continuous arrows constitutes a persistent pattern. It can influence the global evolution. Of course, it can disappear in case it doesn't make sense in a more global context.

5 Simulation Results

In this section, we do not simulate the global resolution for the generic problem described. We only attempt to underline that simple local simple behaviours can show up efficiently the global collaboration hidden in a pre-established network of collaborations. We consider that each agent attempts to get *nbGet* resources from its getting collaborations and to leave *nbLeave* ones from its leaving collaborations. We only report resolvable problems.

5.1 Implemented Behaviours

Representation and perception of the collaborations. We assign a pheromone to each collaboration. Pheromone value evolves according to a confidence value, attributed by agents to the collaboration.

Using its local vision, each agent evaluates the state of a collaboration:

- natural collaboration: agents sort their collaboration lists thanks to their perceived qualities (resulting from their pheromone values). There are two sorted collaboration lists: one for the getting collaboration (getting resource collaborations), one for the leaving collaboration (leaving resource collaborations). A collaboration appears natural for an agent if it takes place in the beginning of its associated list, fitting the necessity to get a resource or to leave a resource for an agent ($nbGet \geq position$ or $nbLeave \geq position$).
- conform collaboration: a collaboration is conform if it is natural for all its associated agents (for the simple topology used, local collaborations are always between two agents).

Agent activity. An agent acts when it is unsatisfied. If an agent doesn't get enough conform getting collaboration, it acts in a getting way. If it doesn't get enough conform leaving collaboration, it acts in a leaving way.

Reinforcement for a collaboration type. Reinforcement can be done for the getting collaboration and/or for the leaving collaboration. An agent tries to reinforce the pheromones associated to the collaboration of the selected collaboration type. For each collaboration, a reinforcement is transmitted if $Random(0, hopeToBeTransmit) > threshold$. Lets us consider the definition of $hopeToBeTransmit$ for a collaboration:

- If the collaboration is natural: $hopeToBeTransmit = NaturalBase + Random(0, NaturalHazard)$.
- If the collaboration is not natural: $hopeToBeTransmit = NonNaturalBase + gapConsideration * gap$

$threshold$, $hopeToBeTransmit$, $NaturalHazard$, $NonNaturalBase$, $gapConsideration$ are some parameters. $gapConsideration$ corresponds to the percentage (by using the value of pheromones) between the current collaboration and the weaker natural collaboration for the type.

Evaporation process. Pheromones values undergo a regular evaporating process.

5.2 Results

Context. For these first results, we consider the capacity for the system to reach a solution. We measure the number of activity periods before the global satisfaction of agents. For each activity period, agents are called in a random order so as to realize their reactive behaviour:

- perceiving process: perception of the associated collaboration pheromone values. Computing of their states (natural or not). The agent asks adjacent agents to define the conformity of collaborations. It computes its satisfaction from the obtained results.
- acting of the environment: if an agent is unsatisfied, it acts for each collaboration type that must be reinforced.

Coordination capacity on a small perimeter. For this problem, the solution is unique (Figure 4). Each collaboration except plotted collaborations must be confirmed by the system. The system reach the solution after 3.7 periods of activity for agents (average value for 1000 executions, standard deviation: 2.5).

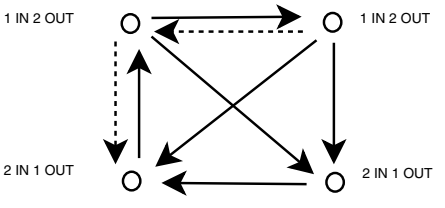


Fig. 4. A first problem

Impact of the size of the coordination. For the problem 5, we attempt to measure the resolving time evolving in front of the exponential increasing of the space of search. As a matter of fact, the size of the search space is equal to $2^{amountofcollaboration}$. Each collaboration can be validated by the system or not. In spite of the simple form used for the scalable problem (a circle), the global coordination problem is not easy for a system that doesn't knows the circle notion. Then, we must consider that the behaviour could react in the same manner for other coordination forms.

The system must invalidate by half the set of potential collaborations. There is only two solutions for odd problems. We add two solutions for even problems: agents can accommodate locally two by two. Even if the cycle activity measure increases exponentially with the problem size, the coordination can be achieved in reasonable time for the studied problems (chart 6). We don't think that a centralized approach could easily solve the problem without adding specific knowledge about the circle notion. It is difficult to cope with the great size of the search space efficiently.

We applied a genetic optimization on the parameters presented in 5.1. This allowed to divide by more than three the presented resolution time for the scalable problem. It made the resolution time quasi proportional to the problem size. We

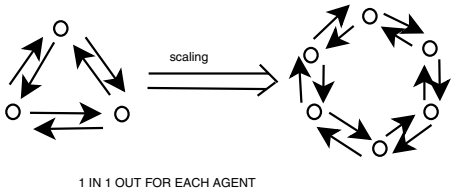


Fig. 5. A scalable problem

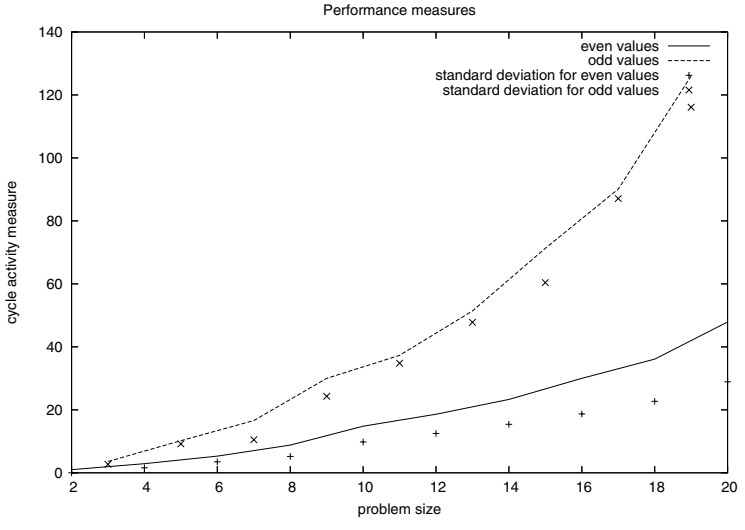


Fig. 6. Impact of the size on the coordination

suspect these results to be problem specific (specific to the set of predefined problems on witch we applied the genetic optimization). In ongoing works, we will study the impact of the net topologie with respect to the resolution complexity, in order to generalize these results. Although the genetic optimisation can increase or reveal some beneficial behaviours, we still need to find the appropriated local agent behaviours first. We consider the genetic optimisation just as a simple tool.

First conclusion. The described experiments constitute the first tests on the system behaviour. We have measured the performance of the system under two specific problems that may not represent all the possible ones. Nevertheless, the first results seem promising. The indicated convergence characteristics seem to be interesting but we are going further to validate it.

6 Industrial Application

6.1 The Considered Problem: Industrial Scheduling

Industrial scheduling is a hard problem. First, many of the scheduling parameters such as processing times, materials getting times, resources availability, etc, are

subject to uncertainty. Second, it is hard to explicitly define the objective for the production system. It often results from opposite objectives, that may evolve over time.

These two aspects of the problem complicate the use of classical computing approaches like Operational Research tools. To represent the objectives at a bottom level (inside the agents) can be much useful.

6.2 Adequateness to the Sharing Critical Resources Protocol

Resources consuming. For the manufacturing problem, machines represent resources, products represent resource consumers. Each machine provides an availability to be shared between products (temporal planning representation). Products cover the production system, they have to find a place on machines for each one of their tasks in a chronological order. Each task is represented by an agent searching for a suitable allocation.

Objective expression. In order to fit the problem definition, we distinguish two parts in the agent task objectives:

- get a robust place face to the uncertainty of parameters, according to the due date: the precise definition of this objective is outside the scope of this paper. This consists in getting a place allowing to move locally without to propagate a disturbance.
- get a place corresponding to specific preference: for instance, to choose a specialized machine.

Then, each time an agent formulates an intention to the collaborative level, it provides associated satisfaction based on its objectives. We note that the preference of agent for part of resources could easily be expressed in the implemented reactive behaviour. We could consider the preference of the agent to calculate $hopeToBeTransmit$ for the pheromone reinforcement process.

A real need to coordinate the moves in the search space. Product tasks regularly change their place to increase their satisfaction face to disturbances. Moving in a blindly way results in a perturbation applied on the system as an unknown factor (Figure 7). Placed tasks have to shift forward or backward on machines plannings in order to accommodate the arriving tasks. Even the departure planning of a moving task creates disturbances. Indeed, The resulting free space must be filled in order to maximize the use of the associated machine. These shifts may produce broken constraints. The task forced to move may not respect the chronological order of tasks for the associated product anymore. In this case, the constraint propagates by the product tasks to the system, and some of those tasks have to find a new place. In Figure 8, we show how an effective collaboration can be applied without to propagate any disturbance. This could be done by the validation of contracts in a global context, and using an adequate process to consider links between contracts through the resolution.

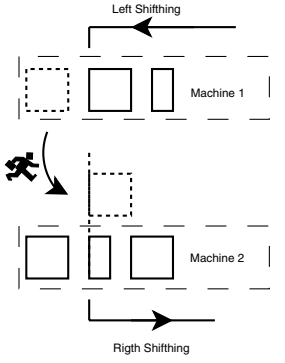


Fig. 7. Moving impact of blindly activity

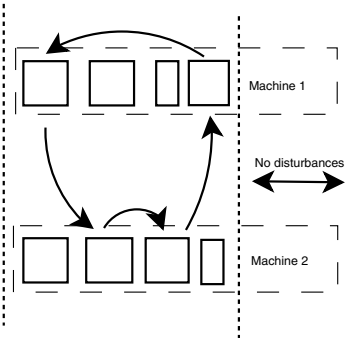


Fig. 8. Moving impact of collaborative activity

7 Conclusion

In this paper, we have described the critical resources sharing problem as a very complex one. We have proposed an approach based on decentralized reactive behaviours and have discussed their fitness to this kind of problems with respect to other classical optimization approaches. In order to reduce the resolution time we have underline the need for an efficient coordination of the agent activity. We have pointed some limitations and drawbacks of some existing coordination approaches, and motivated the proposal of a new generic protocol, based on stigmergy. We have presented some simulations to illustrate our approach and given some first promising results. Before concluding, we have presented an instance of our problematic through an industrial application domain, which is scheduling under disturbances. For this domain, there are relatively few reports about the production of an anticipated planning using reactive behaviours. We think that this limitation is due to the lack of capacity to consider complex characteristics between contracts during the resolution process with known protocols.

References

1. Samir Aknine, Suzanne Pinson, and Melvin F. Shakun. An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent Systems*, 8(1):5–45, 2004.
2. Frédéric Armetta, Salima Hassas, Simone Pimont, and Emmanuel Gonon. Managing dynamic flow in production chains through self-organization. In *2nd International Workshop on Engineering Self-Organising Applications (ESOA'04) to be held at the AAMAS'04*, New York, USA, July 2004.
3. Vincent Cicirello and Stephen Smith. Wasp-like agents for distributed factory coordination. Technical Report CMU-RI-TR-01-39, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 2001.
4. Jacques Ferber. *Objet et Agents : une étude des structures de représentation et de communication en Intelligence Artificielle.* Thèse de doctorat, Université Paris VI (Jussieu), 1989. In French.
5. P.-P. Grassé. La reconstruction du nid et les coordinations inter-individuelles chez *Bellicositermes natalensis* et *Cubitermes sp.* La théorie de la Stigmergie: Essai d'interprétation du comportement des Termites Constructeurs. *Insectes sociaux*, 6:41–80, 1959. In French.
6. Hadeli Karuna, Paul Valckenaers, Constantin Bala Zamfirescu, Hendrik Van Brussel, Bart Saint Germain, Tom Holvoet, and Elke Steegmans. Self-organising in multi-agent coordination and control using stigmergy. In Giovanna Di Marzo Serugendo, Anthony Karageorgos, Omer F. Rana, and Franco Zambonelli, editors, *Engineering Self-Organising Applications, First International Workshop, ESOA 2003. Melbourne, Victoria, July 15th, 2003. Workshop Notes*, pages 53–61, 2003.
7. Nguyen, T. D, and Jennings. Managing commitments in multiple concurrent negotiations. *Journal Electronic Commerce Research and Applications(Issue 4)*, 2005.
8. H. Van Dyke Parunak, Sven A. Brueckner, Robert Matthews, and John Sauter. How to calm hyperactive agents. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 1092–1093. ACM Press, 2003.
9. H. Van Dyke Parunak, Sven A. Brueckner, Robert Matthews, and John Sauter. How to calm hyperactive agents. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 1092–1093. ACM Press, 2003.
10. Alain Taquet Philippe Mathieu. Une forme de négociation pour les systèmes multi-agents, 2000. In French.
11. R.G. Smith and R. Davis. Frameworks for co-operation in distributed problem solving, 1981.
12. John M. Usher and Yi-Chi Wang. Negotiation between intelligent agents for manufacturing control, 2000.
13. Yi-Chi Wang and John M.Usher. An agent-based approach for flexible routing in dynamic job shop scheduling. In *11th Industrial Enginering Research*, 2002.

Information-Driven Phase Changes in Multi-agent Coordination

Sven A. Brueckner and H.V.D. Parunak

Altarum Institute, 3250 Green Court,
Suite 300, Ann Arbor, MI 48105
{sven.brueckner, van.parunak}@altarum.org

Abstract. Large systems of agents deployed in a real-world environment face threats to their problem solving performance that are independent of the complexity of the problem or the characteristics of their specific solution mechanism. One such threat is the degrading of the quality of agent coordination mechanisms when faced with delays in the flow of critical information among the agents introduced by communication latencies. In this paper we demonstrate in a simple model of locally interacting agents that the emerging system-level performance may degrade very suddenly as the rate of individual decision making increases against the availability of up-to-date information. We present results from extensive simulation experiments that lead us to select a locally accessible metric to adapt the agent's individual decision rate to values that are below this phase change. Given the generic nature of the coordination mechanism that is analyzed and the information-theoretic metric, the adaptation mechanism may increase the deployability of large-scale agent systems in real-world applications.

1 Introduction

In many real-world applications (e.g., manufacturing control, intelligent sensor networks, fine-grained robotics) multi-agent systems comprise many entities with sometimes severely restricted resources (i.e., processing, memory, bandwidth). These entities often exchange information to coordinate their individual activities and to achieve meaningful system-level behavior.

The interactions of the agents transmit information that influences the agents' decision processes. Thus the performance of the individual agent as well as the whole agent system directly depends on the timely and accurate flow of information. But, in systems of resource-limited agents this requirement may not be met since communication requires time and may be noisy.

A simple graph-coloring model enables us to explore systematically the impact of the speed of the information flow on the quality of the decision processes in a large-scale agent system. The complex emerging dynamics of the decision processes include a robust phase change in the system performance driven by the latency in individual communications. Knowing the location of this phase change is very important in the tradeoff between the speed in which a solution is achieved by the system and the quality of the solution.

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 104 – 119, 2006.

In deployed systems the communication latency is determined by many factors and it may vary over time as well as over the space inhabited by the agents. Furthermore, we seldomly are in a position to control the speed of the information flow and thus our solution-time vs. solution quality tradeoff has to be determined by the rate of the individual agent's decision process.

Based on the analysis of the phase structure of the system's performance and its reflection in metrics that are accessible to the individual agent we present a generic local control mechanism that permits the agents to determine their currently optimal decision rate to achieve a good solution in the shortest time possible. Our local control mechanism induces meta-level coordination in that it observes the performance of the system through local indicators and guides the individual agent activity for the greater good. These are the basic characteristics of the cognitive functions of introspection and learning, realized in our case through emergent sub-symbolic reasoning.

The remainder of this paper is structured as follows. In section two we present our agent-based graph-coloring model. In section three we discuss our experimental approach and a software infrastructure that we developed to explore large regions of a model's parameter space automatically. In section four we explore the complex dynamics and the emergent phase structure of the graph-coloring model. In section five we introduce a generic local control mechanism that realizes system-level introspection and learning and we demonstrate the improvement of the performance achieved in the graph-coloring example. We conclude in section six.

2 Distributed Graph Coloring

The graph-coloring problem is a fundamental challenge problem to which many other coordination tasks may be reduced. In its general form it seeks to assign one color out of a globally fixed set of size G to each node in an undirected graph so that the number of edges that connect nodes of the same color is minimized.

We directed our attention to graph-coloring in a multi-agent coordination context in the DARPA ANTS program, where the team from Kestrel Institute explored the dynamics of a distributed sensing and tracking system controlled in real-time by agents that are severely restricted in their processing and communication capabilities [8]. In the application, an agent corresponds to a small robotic radar sensor that can light one of three segments of the sky at any time. It takes at least three nearby sensors to focus on the same region of the airspace above to track an object. The graph-coloring problem in this case maps nodes to regions in space in which at least three sensors overlap and edges connect regions that share the same sensor. The number of colors relates to slots in a repetitive schedule of the sensors that need to be coordinated to track objects discovered by at least one sensor.

Our research in this program was concerned with the general dynamics of distributed resource allocation. Phrased in this context, each node in the graph represents a task, each color represents a resource, and an edge between two nodes (tasks) indicates that a single resource cannot service them simultaneously. Thus, graph-coloring models (spatio-)temporal coordination tasks that are very common in multi-agent applications in general.

In this paper we do not present a new approach to solving the graph-coloring problem. Rather we use the distributed solution algorithm proposed by the Kestrel team to demonstrate the impact of delayed information flow on the solution quality and speed. Soft, real-time distributed graph coloring [5] assigns an agent to each node in the graph that needs to be colored. Thus there are N agents (one for each node in the graph) in the multi-agent system and, according to the undirected edges among the nodes, each agent has a number of direct neighbors to whom it communicates changes of its color.

In our experiments we typically considered random graphs in which each node has a fixed number of neighbors (K). We implemented multiple ways (indexed by the GC parameter) of sequentially constructing such graphs, each of which resulted in graphs with specific characteristics. For instance, in one graph construction mechanism, we randomly distributed the nodes on a unit square and assigned each node those K nearest neighbors that did not yet have their complete set of neighbors assigned. This mechanism typically produces graphs that may be embedded in low-dimensional spaces.

Another mechanism selects randomly among those nodes that have the least number of neighbors assigned already and connects the chosen one to another of these most incomplete nodes. This mechanism tends to yield graphs with a very short characteristic path length. All experiments reported in this paper were conducted with graphs constructed by this mechanism, but the observations and conclusions also hold for other GC parameters.

In the chosen graph-coloring algorithm, any agent cyclically decides whether to reconsider its color choice or not. The so-called activation decision is taken nondeterministically with a fixed probability, which we call the agent's Activation Level (AL). The randomized activation of the nodes in the graph is intended to (statistically) prevent simultaneous color changes among neighboring nodes.

Figure 1 sketches the basic decision process that a node executes if it is activated. At first, taking into account the currently known colors of the node's neighbors, it computes the local Degree of Conflict (DoC) for each available color. The local DoC is the node's main performance metric. For any (assumed or real) color of the node, it is the number of neighbors that share this color divided by the overall number of neighbors of this node. The agents attempt to solve the coloring problem across the graph by individually minimizing this local metric.

After determining the expected consequences of each available option, the agent selects its color in two steps. First, it may reduce the option set depending on the permitted change in the local DoC. The "Movement Direction" (MD) parameter determines, whether only colors that improve (reduce) the local DoC (MD="Only-Up"), colors that do not increase the conflict (MD=``LateralAndUp"), or all colors (MD= ``Any") may be selected.

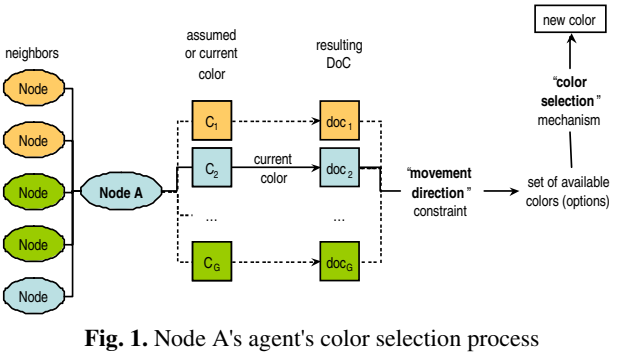


Fig. 1. Node A's agent's color selection process

The agent keeps its current color if no color meets the respective constraint.

In a final step the agent selects its new color from the remaining option set. We experimented with three different mechanisms determined by the model's ``Color Selection'' (CS) parameter. Either the agent selects the next color randomly from the reduced option set (CS=``Random''), or it chooses probabilistically with the individual probabilities inversely proportional to the expected local DoC (CS=``Roulette''), or the agent chooses randomly only among the colors that promise the largest improvement in the local DoC (CS=``Best'').

Table 1. Graph-Coloring Model Parameter

Model Parameter	Description
N	Number of Nodes in Graph
K	Number of Neighbors per Node
G	Number of Colors Available
GC	Mechanism to Construct Random Graph
AL	Probability to Activate Color Decision Process
RP	Probability to Randomly Change Color
CL	Time of Message Transfer to Nearest Neighbor
MD	Constraint on Permitted Change of Local DoC
CS	Mechanism to Select Color from Option Set

If the color that the agent selects is different from the current color, the agent communicates the change to all its direct neighbors in the graph. In our model we delay the arrival of the change messages at the neighboring nodes by a globally fixed time specified in the "Communication Latency" (CL) parameter.

Our model also includes a simple noise process. Any node has a fixed probability to "fail". In each cycle, independent of whether the color decision process is activated or not, the node may randomly select any color with a probability specified in the "Reset Probability" (RP) parameter.

Table 1 lists all available model parameters that may be varied in the exploration of the emergent system dynamics. In the following section three we present a software infrastructure that supports a systematic exploration of this parameter space.

3 Systematic Parameter Sweeps

Even though the individual decision processes and agent interactions may be very simple as they are for instance in the distributed graph-coloring model, formally analyzing the complex emergent dynamics of a large system of these agents quickly becomes intractable. But it is exactly the dynamics in the large that primarily interest us.

Constructing a software simulation of the respective model and then experimentally exploring selected regions of the model's parameter space provides a solution to this dilemma that is chosen these days by many researchers. But there are many pitfalls to be avoided when experimenting with complex nondeterministic models with large parameter spaces. For instance, the introduction of artifacts by the system's random-number generator has been discussed extensively. Another important issue is

finding interesting regions in vast parameter spaces while only having limited time and computational resources available. We propose an approach to this issue in another paper [2].

Another non-trivial problem in following the experimental approach is plain bookkeeping. Mapping out the phase structure of the distributed graph-coloring model requires the setup, execution and analysis of hundreds of thousands of experiments – a task that cannot be achieved manually.

We developed a parameter sweep infrastructure that enables us to explore the dynamics of a given simulation model efficiently. The infrastructure automatically sets up experiments either through XML configuration files in the file system or through JAVA objects directly handed over to the simulation thread. The use of XML configuration files permits us to run individual experiments manually at any time and it also allows us to farm out experiments on to any available computer in our network that has access to the fileserver.

After a simulation experiment has been completed success-fully, it stores its results in an XML report file in the file system. In any given parameter sweep we may be interested in a number of metrics such as traces of individual agent states or statistical measures aggregated over agent populations and many simulation cycles. But in general we do not require all possible metrics to be recorded at all time. Therefore we specify with the configuration of a parameter sweep, which of the measurements should be reported.

Reporting our metrics in XML format enables the parameter sweep infrastructure to aggregate the results of many experiments efficiently into one report whose structure reflects the dimensions of the region of the parameter space that had been explored. Additional filters applied after the completion of the sweep translate the XML file into any format required by our chosen analysis software. We currently use Mathematica, Microsoft Excel, and specifically tailored JAVA programs to analyze and graphically display our experiments such as those reported in this paper.

The integration of the parameter sweep infrastructure with a new simulation model is very simple. We configure a parameter sweep using a meta-level XML setup file, which specifies which parameters are to be "swept" and how they are communicated to the underlying simulation software. We also separate the collection of our report data from the simulation itself similar to the Swarm package's Observer Swarm concept [7].

We have used our parameter sweep infrastructure in several research projects to explore the dynamics of agent systems. Currently, it supports experiments with an agent-based supply-network simulation implemented in the Swarm package, a JAVA implementation of a distributed formation flying mechanism for robotic planes, and the emergent dynamics of a swarming path planning algorithm.

4 Information-Driven Phase Changes

The dynamics of the distributed graph-coloring system may be explored in many dimensions and they are reflected in various local and global metrics. In the following we map the general layout of the dynamics along the major dimensions and then focus on the sudden transition from good to bad performance.

4.1 Parameter Space

Table 1 lists the parameters of our agent model of distributed graph coloring. These parameters may be grouped into problem parameters, solution parameters, and environmental parameters.

Problem parameters specify the particular graph-coloring task. These include the number of nodes (N), their arrangement into an undirected random graph (K, GC), and the number of colors available (G). Varying these parameters while keeping the ones of the other groups fixed explores the algorithm's performance under various problems.

Solution parameters configure the operation of the distributed graph-coloring algorithm. This group includes the permitted change of the local DoC (MD), the method of selecting a new color (CS), and the average decision rate of a node (AL). Varying these parameters explores the performance of different classes of the solution approach for the same problems. The parameters of the solution algorithm are the ones that need to be adapted in the deployment of the agent system to solve a certain range of graph-coloring problems in a particular environment.

Environmental parameters define the conditions under which a particular solution mechanism attempts to solve a given problem. We model the delay in communication among the nodes (CL) and the rate of node failure (RP). Exploring these dimensions helps us understand the impact of a particular deployment scenario on the solution of a given coordination problem.

4.2 Metrics

The agents in the distributed graph-coloring problem interact locally to solve a global problem without being explicitly aware of their joint task. Therefore we consider the global solution to be emergent.

The emerging dynamics of the solution mechanism applied to a specific problem in a particular environment may be observed in a variety of metrics. These metrics may be based on processes inside the individual agent, observations across the agent population at a given point in time, or aggregations over time.

The primary performance metric is the global Degree of Conflict, which measures the quality of the solution to the graph-coloring problem at a particular point in time. Similar to the local DoC metric, it is defined as the portion of edges in the overall graph that connect nodes of the same color. It is the goal of the agent population to minimize this metric.

To measure the quality of improvement provided by any solution mechanism, we compare the observed performance with the expected performance of a random search mechanism. In our case we assume for the random baseline that every agent selects its color uniformly random. In this case the expected global and local DoC equals 1/G. Therefore, we usually plot the DoC observed in experiment multiplied by G to show the change in the performance compared to random agent behavior. For configurations in which the product of observed DoC and G is smaller than one, our algorithm outperforms random behavior.

Any agent makes its color decision based on local information provided by its neighbors. It is the general approach of the distributed graph-coloring mechanism in

our model to optimize the global DoC by individual local improvements. In analyzing such individual hill-climbing mechanisms it is typically useful to consider the "steepness" of the hill, which determines the guidance that an agent receives from the information that is used by its decision process.

A metric called Option Set Entropy (OSE) estimates the guidance in the currently available local information as it is used by the agent's decision process. The OSE is the normalized Shannon (or Information) Entropy [14] applied to the probability of an agent's selecting a particular color in a decision cycle at a specific point in time. This probability is determined by the currently known colors of the node's neighbors, the constraints on the change of the local DoC (MD parameter), and the chosen color selection mechanism (CS parameter).

We compute the OSE for an individual agent based on the selection probabilities computed by the agent in a decision cycle. All colors that are excluded by the MD constraint are assigned a probability of zero. If the agent uses the random color selection mechanism, all remaining colors share the same probabilities. If CS equals Roulette, the probabilities of the MD-reduced option set are those used in the roulette wheel selection. Finally, if the agent only chooses among the best options, all but the best colors have a probability of zero and the best ones have equal (non-zero) probabilities.

We compute the OSE as $-\sum_{i=1}^{G} p_i \ln(p_i) \Big/ \ln(G)$ where p_i is the probability of the agent to select the i-th of the G colors in this decision cycle. The metric reaches its maximum value of one if the agents select randomly among the G colors.

OSE is a local metric that an individual agent may compute without any additional information. This is an important characteristic to qualify a metric as a candidate for individual agent learning.

A third important metric on the operation of the distributed graph-coloring mechanism is the amount of false information that is used by an agent in a color decision cycle. The False Information Percentage (FIP) measures the proportion of neighbors for whom the node assumes the wrong color when it makes its decision.

Table 2. Additional Graph-Coloring Model Metrics

Metric	Description
Agent Activation Count (AAC)	Number of Agents Activated in a Simulation Cycle
Changed Color Portion (CCP)	Percentage of Nodes that have Changed their Color in the Last Cycle
Color Change Entropy (CCE)	Information Entropy in Distribution of Recent Color Change Events in the Graph – Computed over whole Graph and Multiple Cycles
Graph Characteristics (GC)	Watts' Graph Metrics - Characteristic Path Length and Clustering Coefficient (REFERENCE)
Random Option Size (ROS)	Number of Colors among which an Agent chooses Uniformly Random
Time to Solution (TTS)	Estimate of the Number of Cycles it takes for the Network Dynamics to Approximate a Fixed Location in State Space

False information is a direct result of the communication latency introduced by the real-world constraints of the deployment of the agent system. At the time an agent chooses its color, its neighbors may have already changed their colors but the message sent by those neighbors has not yet reached this node. Thus, the agent uses stale color information.

The FIP metric is non-local in time in the sense that the agent can only determine the percentage of false information after the fact. But it is local to the agent and thus, by keeping track of the assumed color values that went into a decision process and subsequently arriving (time-stamped) messages that falsify the assumptions, the agent may compute the FIP of a decision cycle CL time-steps later. In our simulation environment we are of course able to take the global view and measure the FIP of an agent directly, but we never let the agent access this information.

Table 2 lists additional metrics that we found useful to measure during our exploration of the model's parameter space, but which we will not report in this paper.

4.3 Mapping the Problem Space

Any undirected graph has a unique chromatic number, which is the minimum number of colors required to color the nodes without any conflict. In terms of our model, it is the smallest G for which there exists a color assignment of the graph that results in a global DoC of zero.

Our exploration of the emergent dynamics of distributed graph coloring was part of a larger project [12] in which we researched the general dynamics of resource allocation. Considering the graph coloring problem as an instance of resource allocation, we focus on the transition from an underloaded state, in which we have more resources (colors) available than required, to an overloaded state, in which there are not enough colors to resolve all conflicts.

No algorithm is known to determine the chromatic number of an arbitrary graph analytically. As a consequence we are not able to determine the location of the transition from an underloaded to an overloaded system just on the basis of our problem parameters. Furthermore, previous experiments with a generalized Minority Game model [13] showed that the characteristics of the solution mechanism strongly influence the emergent dynamics near the problem transition, which may lead to complex patterns of sub-optimal performance.

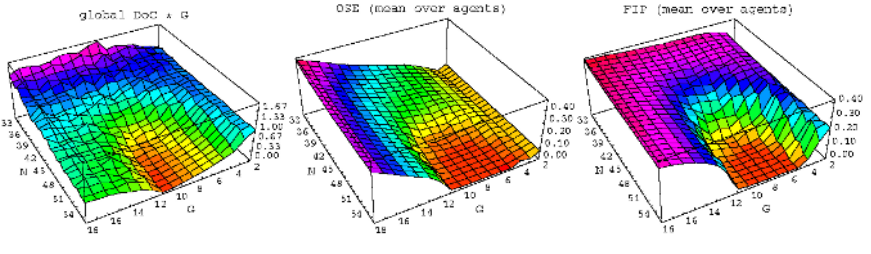


Fig. 2. Parameter Sweep over N and G

Figure 2 shows our three metrics (DoC, OSE, FIP) for varying configurations of the problem parameters N and G. Each coordinate in a plot corresponds to the statistical mean of the respective metric over 32 simulation experiments of 2500 cycles with varying random seed. The data used in computing the metrics is gathered beginning at cycle 2000, to avoid initial transients. We vary N from 32 to 56 nodes and G from 2 to 18 colors. All other model parameters are fixed to K=30, GC=MinimumNeighbors, AL=33%, RP=0%, CL=1, MD=Any, CS=Best.

Considering the DoC*G plot, we find that there is a region near N>44, G=10 in which the agents clearly outperform the random baseline. We also find that the system performs worse than random for similar values of G but smaller values of N. Finally, we see the system performance approaching the random baseline as we increase the number of colors available. Figure 3 sketches the location of these three performance regions and Table 3 categorizes the observed values in our metrics by region.

Table 3. Observed metrics by region

Metric	Region		
	asymptotic	**better**	**worse**
DoC * G	medium	low	high
OSE	high	low	medium
FIP	high	low	high

How may we explain the structure of the observed system performance in comparison with the random baseline? The answer for the asymptotic region can be found right away: Increasing G beyond the transition from the overloaded to the underloaded region in problem space makes the graph-coloring problem so easy that even a random assignment already provides a good solution. Thus, it is not the case that the performance of the agent system fails, but rather, the performance of the random baseline improves with increasing colors available.

Considering the OSE metric in the asymptotic region we notice a slight decrease of the guidance (increasing entropy) for the average agent as we move to very large values of G. This observation supports our hypothesis, since in the asymptotic region many colors may provide the perfect solution and thus the local hill-climbing algorithm of the agents has many equally perfect options available even on "top of the hill".

We also observe higher values of FIP in this region. This is due to the fact that even at the optimal local no-conflict state, there are multiple colors that are as good as the current one and thus the CS=Best mechanism may select randomly among them. As long as the agent changes the color of its node there will be messages sent to the node's neighbors, whose delay result in false information used by the neighbors. The use of false information does not have a negative impact on the system's DoC metric, since most choices are good ones anyway. However, the continuous change of color uses processing and communication resources without improving the solution.

In the region where the observed performance performs better than random, the agents also quickly find a stable color assignment (low FIP). Especially for larger values of N we see that the algorithm stabilizes even though a conflict-free color assignment is not found (DoC*G>0). The hill-climbing algorithm of the individ-

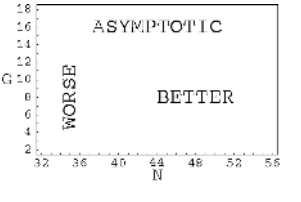


Fig. 3. System Performance Compared to Random Baseline

ual agent (MD=Any, CS=Best) drives the overall system into a good local optimum from which it cannot escape.

The plot of the FIP metric supports this hypothesis. The individual agent quickly finds a color assignment, which it cannot improve. Therefore only few color changes are communicated, which in turn reduces the portion of false information (FIP) in the agents' decisions.

In contrast, in the region in which the system performs worse than random, the FIP metric remains high but the OSE metric is low. This indicates that the system does not settle into a local optimum (high DoC) because the agents take well-guided decisions (low OSE) based on outdated information (high FIP) – the system is thrashing.

We hypothesize that the descent into thrashing behavior is caused by the increased connectivity of the overall graph. In all configurations the number of neighbors per node (K) is fixed to 30. Thus, in a system with only 31 nodes, each node must be the neighbor of all other nodes and thus it has to coordinate its color directly with every node in the graph. Once we increase the number of nodes and therefore decrease the K/N ratio, a node has to coordinate only with a subset of the graph and the problem becomes easier.

Assume a state in which the nodes have a high probability to change their color. With increasing connectivity of the graph, the probability that a node's direct neighborhood includes one that just changed its color increases. Thus, high rates of color change produce increasing FIP values for decreasing N. Using outdated information to solve a conflict that was already solved may actually reinstate the conflict. This is particularly true if the number of available colors is small. Therefore we expect thrashing behavior of the distributed graph-coloring mechanism in the low-N low-G region of the problem space.

Figure 4 shows the three different system-level behaviors that we observe in the distributed graph-coloring model.

4.4 Phase Changes

The goal of the agent population to color their graph with the lowest degree of conflict is fulfilled in the asymptotic region and in the better-than-random region. Only in the region of low N and low G we find that thrashing induced by the delay of critical information leads to less-than-optimal performance.

The communication latency parameter is the main driver of the performance decrease in our model. In a deployed system communication latency is an environmental parameter that usually may not be influenced by the agents or the agent programmer. To improve the performance of the agent system for low-N low-G problems, we have to find a solution parameter that counteracts the delay in communication.

Thrashing occurs because the agents make color decisions before critical information has arrived. In our model it is therefore the rate at which decisions are taken – determined by the agents' activation level parameter (AL) – that needs to be adapted to the particular problem and environmental parameters. In section five we introduce a local adaptation mechanism that provides this capability, but first we analyze the transition into the trashing region in more detail.

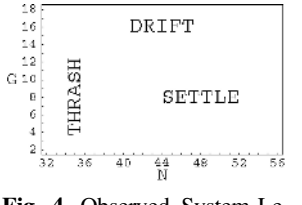
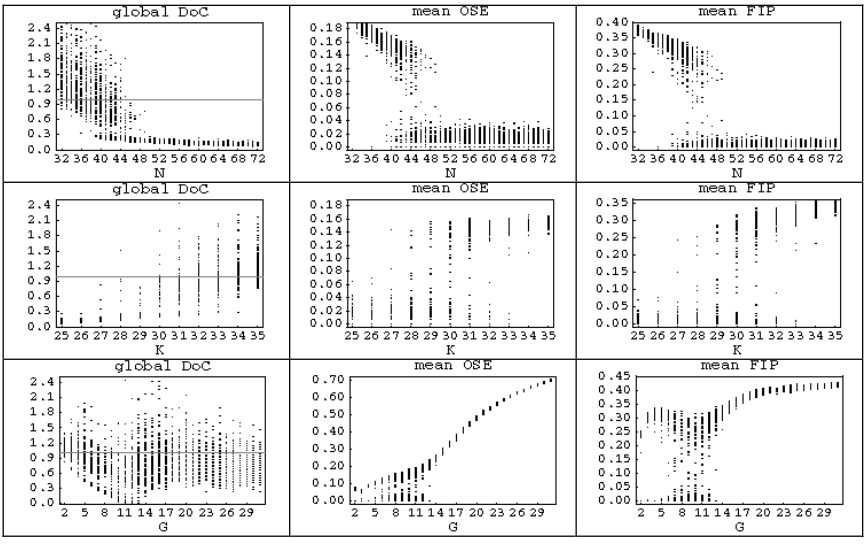


Fig. 4. Observed System-Level Behavior

Table 4. Plotting individual experiments reveals a phase shift into thrashing



The plots in Table 4 show a series of parameter sweep experiments that vary the three problem parameters N, K, and G independently while keeping the remaining other parameters fixed to N=44, K=30, G=4, AL=33%, RP=0%, CL=1, GC=MinimumNeighbors, MD=Any, CS=Best. In all diagrams we plot the respective metric (DoC, OSE, FIP) for each of the 100 individual experiments of 10,000 cycles per configuration, beginning with cycle 9,500.

Plotting results from individual experiments rather then the mean over all experiments at one configuration as we did in Figure 2 shows that the dynamics of our model do not change gradually as we move into the thrashing region. Rather, thrashing is a qualitatively different type of system behavior that either occurs, or not.

Consider, for instance, a move from high to low values of N. Initially, we are in the region of better-than-random behavior, characterized by low values of DoC, OSE, and FIP. But suddenly, at N=49, we encounter systems that do not fit this profile, since the observed values in our metrics are significantly higher. Especially in OSE and FIP we find an extremely large jump.

As we further decrease the values of N, both types of dynamics co-exist for a number of configurations. This overlap region is robust even as we continue to run our experiments for extremely long periods of time. Therefore we conclude that the selection of the respective attractor in system behavior (thrashing or benign) must be driven by the fine-structure of the graph that must be colored. At this point we have no data to explain, which graph-theoretic metric would best describe the boundary of the basins of attraction.

Finally, for even lower values of N, we find that all experiments result in robust thrashing behavior.

Our description of these changes as a “phase change” relates to the notion of a “phase transition” in physics. Many physical systems can exist in distinct phases and exhibit discontinuities in passing between them. Common examples are melting and

evaporation, or ferromagnetic transitions at the Curie temperature. Our results are particularly reminiscent of physical first-order phase transitions such as the melting and evaporation of water, in which multiple phases can coexist. Physical phase transitions have been a fruitful source of inspiration in computer science, particularly in studying issues of computational complexity [4, 6, 10]. In keeping with usage in the physics community, we believe it is important to reserve the term "phase transition" for a point of nonanalyticity in the behavior of a system, and describe our observations as "phase changes" or "phase shifts." That is, a "transition" is characterized on the basis of the mathematical behavior of a model of the system, while a "change" is an empirical observation based on experimentation. It will often be the case that the two correspond, but careful use of vocabulary supports the important distinction between empirical and theoretical results.

5 Introspection and Learning

Two of the system-level behaviors exhibited by the distributed graph coloring model (better than random, asymptotic to random) correspond to good problem-solving performance, either because the agent population is "intelligent" enough to solve a complex problem, or because the problem is so simple that even a random solution is likely to be good already. In the third region in parameter space, thrashing prevents the agents from finding and maintaining a good solution. To perform well in this region, agents must possess introspection (the ability to detect that they are thrashing) and be able to learn from their experiences.

The distributed graph-coloring model represents a class of agent systems in which the agents are only able to interact locally based on potentially incomplete or outdated knowledge. Such systems occur for instance in real-world applications that deploy large numbers of agents in a physical environment with limited resources available to the individual agent (e.g., swarming robotics, sensor networks). As a consequence of the real-world constraints, any mechanism that is supposed to detect and counteract the emergence of thrashing behavior must operate at the individual agent level in regards to its input data and its influence on the system dynamics. Just as thrashing is an emergent phenomenon, thrashing prevention must be an emergent functionality as well to be implemented in a completely distributed and decentralized fashion.

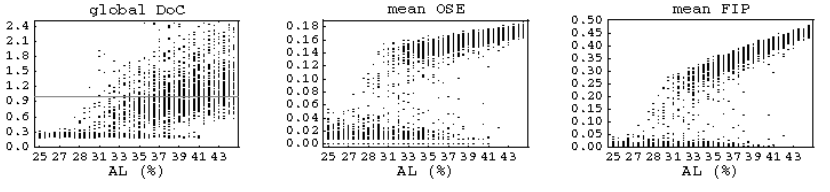


Fig. 5. Sweep over AL

In our discussion of the regions of performance we identified the delay of critical information as the main driver for the emergence of thrashing behavior and we suggested that the rate of decision making (parameter AL) may be the most suitable solution parameter that needs to be adapted to prevent thrashing. Figure 5 plots another

sweep around the general configuration chosen for the plots in Table 4. In this sweep we only vary the AL parameter, but we again find that the systems eventually fall into the thrashing attractor for large enough AL. In this set of experiments neither the complexity of the problem nor the characteristics of the environment change. It is only the rate at which the agents take their color decisions that determines whether the system is thrashing or not. Thus, we conclude that for any configuration there exists a critical maximum AL value, above which thrashing will occur.

Any system with a globally fixed AL parameter faces a dilemma. If AL is too high, some possible scenarios (problem and environmental parameters) will lead to thrashing, while if it is too low, response will be slower than would otherwise be possible. Real-world applications are often driven toward this dilemma. Economic pressures on a resource allocation system tend to keep the number of resources low, corresponding to low G, where random choice is ineffective, while time pressures will tend to force decision rates up, corresponding to high AL.

We address this dilemma by expanding the individual agent behavior with a local "AL-Learning" mechanism. Our proposed mechanism causes the agent to skip probabilistically some decision cycles that it normally would have executed given its preset AL parameter. Thus, we are able to slow an agent's decision rate down until the thrashing stops. With this learning mechanism in place we may preset the AL parameter to high values that permit short solution times, relying on the agents to modulate their own activity to avoid thrashing.

The change in the behavior of an individual agent is driven by an additional parameter, which we call the agent's No-Action Pheromone (NAP). It determines the likelihood that an agent's color decision is skipped and therefore an agent with a NAP of zero operates at the decision rate specified by the AL parameter, while NAP values larger than zero reduce the decision rate.

Initially, any agent's NAP is zero, but over time the agent collects evidence that may increase the parameter. It is the rate and strength of the locally observed evidence that drive the increase in NAP.

With the accumulation of knowledge (evidence) by the agent, we face a truth-maintenance problem. Over time the agent may observe states that may be interpreted as evidence for thrashing, but that are just part of the normal operation. Also, for instance through changes in other agents' behavior, old evidence may no longer justify a decreased decision rate. Therefore, following the truth-maintenance approach of natural agent systems (e.g., insect colonies), an agent immediately starts to forget any knowledge that it receives unless it is continuously reinforced.

We find that the observed percentage of false information is a superior source for the extraction of evidence for thrashing behavior at the individual agent level. As the plots in Table 4 show, we generally observe low FIP values when the global performance (degree of conflict) is high and high FIP values if the DoC is high. As mentioned before, the individual agent cannot observe the portion of false information in its current color decision directly, but by keeping track of incoming and used color information, it may determine the FIP with a delay equal to the communication latency of the system. Our experiments show that using this delayed FIP is sufficient to remove the system from the thrashing region.

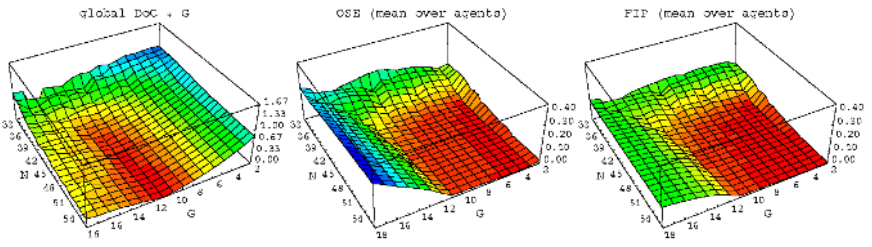


Fig. 6. Parameter Sweep over N and G with AL-Learning

We treat the current NAP value as a concentration of a digital pheromone as we used them in several previous applications ([11], [3], [1]). Consequentially, we deposit additional pheromone into NAP in proportion to the strength of the current evidence, which is the locally observed delayed FIP. At the same time we continually "evaporate" the pheromone, reducing its concentration through multiplication with a fixed No-Action Probability Evaporation (NAPE) factor between zero and one.

Based on our formal analysis of the dynamics of digital pheromones [1], we know that a repeated evaporation and deposit of the strongest evidence (FIP=1) drives the NAP parameter to approximate a fixed point of 1/(1-NAPE). Treating this fixed point as the upper limit, the agent skips a color decision cycle with a probability of NAP*(1-NAPE).

Figure 6 repeats the parameter sweep experiment conducted for Figure 2 with the emergent introspection and learning mechanism at a NAPE value of 0.9. For easier comparison we scaled the z-axes to the same interval as in plots in Figure 2. We see that the thrashing region is completely gone (no performance worse than random) and almost completely replaced by good performance in the other two regions.

A companion paper [9] analyzes this approach to calming hyperactive agents in more detail, and shows how it can be generalized to more concrete resource application problems and to reasoning about approaching deadlines.

6 Conclusion

In this paper we present a model of distributed coordination among agents with limited resources in a real-world environment. The distributed graph-coloring mechanism, proposed in [5], deploys agents in an environment that delays the transfer of information in the local interactions of neighboring agents in an undirected graph. The agents exchange information to assign colors from a finite set to their respective node in the graph in a way that globally minimizes the number of neighbors with the same color. The graph-coloring problem is an abstraction of many important agent coordination problems.

The high degree of complexity of the emergent system-level dynamics suggests an experimental approach to the analysis of the model, especially for large systems as they occur in real-world applications. To support systematic experimentation and gathering of data from individual experiments we developed a generic software

infrastructure that executes for a specified subset of the model's parameter space multiple replica of the model with different random seeds. The infrastructure enables us to explore the dynamics of an agent simulation model efficiently.

The parameters of the distributed graph-coloring model fall into three different classes. Problem parameters configure the specific graph-coloring problem, solution parameters specify the features of the agent's color decisions, and environmental parameters characterize the deployment scenario in which the agents solve the particular problem.

Analyzing the dynamics of distributed graph coloring in a restrictive environment we find that there are three distinct regions of system behavior. On the one hand, the system may show good performance either because the agent population is sufficiently intelligent to settle on a low-conflict solution, or because the problem is so easy that almost any randomly selected configuration results in a low degree of conflict. On the other hand, the agents may be prevented from finding and maintaining a good solution because critical information is delayed and the system falls into thrashing behavior.

Detailed analysis of the transition into the thrashing region reveals that thrashing is a qualitatively different phase in behavior space. Changing any of our model parameters to transition into this phase takes us through an overlap region in which systems may or may not fall into the thrashing attractor. This observation leads us to conclude that there must be an implicit parameter, such as a characteristic of the randomly created graph, that determines the attractor selection in the sensitive overlap region.

Thrashing significantly reduces the problem-solving performance of the system. We find that for any configuration there is a maximum AL value, above which the system falls into the thrashing attractor. The individual agent may manipulate the solution parameter AL and thus we propose an adaptive mechanism based on the local metric FIP (false information percentage) that dynamically adjusts the effective decision rate of the agent to prevent thrashing. Our experiments verify the performance improvements.

The addition of the emergent introspection and learning mechanism drastically improves the performance of distributed systems of agents coordinating in a limiting environment. Rather than being forced into very conservative global decision rates to guarantee that thrashing cannot occur in any possible scenario, we may start the agents with a higher decision rate that leads to faster solutions and slow them down only as thrashing occurs

Acknowledgements

This work is supported in part by the DARPA ANTS program, contract F30602-99-C-0202 to Altarum, under DARPA PM Vijay Raghavan. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government. We gratefully acknowledge the cooperation of Stephen Fitzpatrick at Kestrel Institute for consultations on their graph coloring model for sensor allocation.

References

[1] S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Dr.rer.nat. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.

[2] S. Brueckner and H. V. Parunak. Resource-Aware Exploration of Emergent Dynamics of Simulated Systems. In *Proceedings of AAMAS'2003*, Melbourne, Australia, 2003.

[3] S. A. Brueckner and H. V. D. Parunak. Swarming Agents for Distributed Pattern Detection and Classification. In *Proceedings of Workshop on Ubiquitous Computing, AAMAS 2002*, Bologna, Italy, 2002.

[4] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proceedings of IJCAI-91*, pages 331-337, Morgan Kaufmann, 1991.

[5] S. Fitzpatrick and L. Meertens. Soft, Real-Time, Distributed Graph Coloring using Decentralized, Synchronous, Stochastic, Iterative-Repair, Anytime Algorithms: A Framework. Technical Report KES.U.01.5., Kestrel Institute, 2001.

[6] T. Hogg, B. A. Huberman, and C. Williams. Phase Transitions and the Search Problem. *Artificial Intelligence*, 81:1-15, 1996.

[7] C. Langton, R. Burkhart, and G. Ropella. The Swarm Simulation System. 1997. http://www.swarm.org.

[8] L. Meertens and S. Fitzpatrick. Peer-to-Peer Coordination of Autonomous Sensors in High-Latency Networks using Distributed Scheduling and Data Fusion. Technical Report KES.U.01.09, Kestrel Institute, 2001.

[9] H. V. D. Parunak, S. Brueckner, R. Matthews, and J. Sauter. How to Calm Hyperactive Agents. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS 2003)*, Melbourne, Australia, 2003.

[10] H. V. D. Parunak, S. Brueckner, J. Sauter, and R. Savit. Effort Profiles in Multi-Agent Resource Allocation. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS02)*, pages 248-255, 2002.

[11] H. V. D. Parunak, S. A. Brueckner, J. Sauter, and J. Posdamer. Mechanisms and Military Applications for Synthetic Pheromones. In *Proceedings of Workshop on Autonomy Oriented Computation*, 2001.

[12] H. V. D. Parunak, R. Savit, S. A. Brueckner, and J. Sauter. A Technical Overview of the AORIST Project. ERIM, Ann Arbor, MI, 2001. URL http://www.erim.org/cec/projects/aorist/AORIST_Snapshot_0104.pdf.

[13] R. Savit, S. A. Brueckner, H. V. D. Parunak, and J. Sauter. Phase Structure of Resource Allocation Games. *Physics Letters A.*, 2002.

[14] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. Urbana, IL, University of Illinois, 1949.

Self-organising Applications Using Lightweight Agents

Paul Marrow[1] and Manolis Koubarakis[2]

[1] PICT Research Centre, BT plc, Orion 1 PP 12, Adastral Park, Ipswich IP5 3RE, UK
paul.marrow@bt.com
[2] Intelligent Systems Laboratory, Dept. of Electronic and Computer Engineering,
Technical University of Crete, University Campus - Kounoupidiana,
73100 Chania, Crete, Greece
manolis@intelligence.tuc.gr

Abstract. Self-organisation in nature is responsible for many complex and persistent phenomena. This suggests that self-organisation may be useful in the creation of complex applications. Multiagent systems use multiple agents to execute complex activities, and thus may be a basis for self-organising applications. In this paper we describe applications using self-organisation based upon the DIET multi-agent platform that supports lightweight agents. Multi-agent systems can be created that support decentralisation, scalability and adaptability. We show that these application properties are useful for information sharing in mobile communities via self-organising among middle agents, and via peer-to-peer interaction between agents.

1 Introduction

The complexity of the living world [e.g. 5] and of physical systems [23] has attracted much interest as a source of inspiration for applications. The persistence and ubiquity of the natural world suggests that self-organisation may be a means of constructing complex applications [2, 10]. Self-organising systems [10] function without central control, and through local interactions. Multi-agent systems (such as Farm [12], JADE [15], JAF [28] and GAIA [33]) provide a means of programming systems distributed across many autonomous entities. This suggests that they may be a useful basis for implementing self-organising systems, providing they have appropriate properties, including adaptability, scalability and decentralisation.

Self-organising systems need to be able to control and adapt their properties in a robust manner. Where such systems consist of many agents the client-server model has been widely used, as elsewhere in computing. Middle agents or brokers [7] can be introduced as intermediaries between different classes of agents [7, 17, 31]. Both these configurations raise problems of the reliability of key elements: of the server or the middle-agents. Alternatively agents can use a decentralized peer-to-peer configuration, e.g. [2, 22]. These systems suggest that decentralisation is a useful property to support robust self-organising applications.

Another property that may be useful for self-organised systems is that of scalability. Scalability has been referred to as the capability to adapt to a required performance level or number of users by adding resources to a system [25]. Scalable changes in software resources may imply not more than linear change in performance

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 120 – 129, 2006.

as system requirements vary. Self-organisation may make this easier, by giving some capability to anticipate changes in the environment, and adaptability may increase the capability to respond to these changes. Since self-organising applications may not be able to predict completely the level of demands upon them, some capability for scalability will be useful.

Decentralised control combined with scalability suggests effective performance. An additional useful property of self-organised applications is to be able to adapt to a changing environment, especially requirements of users. Adaptation in multi-agent systems may take place by a variety of different means [8, 21]. Because self-organising applications depend on local interaction between agents, adaptation will arise in part from interaction between agents [e.g. 8, 20]. A self-organising system that has some capability for adaptation when combined with decentralisation and scalability should be able to support robust applications.

This paper presents some examples of applications that show self-organisation. In the next section we outline the system used, based on the DIET Agents platform [9, 11]. In section three two applications that use self-organised agent systems are described: the self-organising communities application and the P2P-DIET peer-to-peer service. Finally we consider how these applications show results consistent with robust self-organised systems.

2 System Outline

2.1 The DIET Agents Platform

The applications described in this paper have been implemented using the DIET Agents platform [9, 11]. This software platform draws inspiration from complex interactions in natural ecosystems [30]. Many organisms with little individual intelligence can respond to the environment through group interaction (e.g. [5]). Consequently the platform is designed around agents with limited individual capabilities. Intelligent behaviour can emerge from the interaction between agents.

The DIET Agents platform is designed as a three-layer architecture:

- Application Layer
- Application Reusable Component (ARC) Layer
- Core Layer.

The lowest layer, the *core layer*, contains the DIET kernel, enabling DIET environments and providing the basic capabilities for agent creation. The kernel also provides for connections between agents, which allows messages to be passed between them. The core layer also contains basic support for debugging and visualisation.

The *application reusable component layer* contains software components that are reusable between multiple applications, but are not essential for inclusion in the core layer. Examples of the components included are ones supporting remote communication and event scheduling.

The *application layer* contains code specific to particular applications, as well as debugging and visualisation code that may be application specific. Establishing agents in this way enables the multi-agent system so created to be linked to a visualisation system so that users can visualise agent behaviour [18].

The architecture described here simplifies the development of applications, as core classes required for multi-agent systems are provided in a manner that can be easily extended without limiting too much the direction of potential applications. The DIET core platform has been released as Open Source under the GPL Open Source licence. More information is given on the DIET Open Source web site [9].

2.2 Kernel Properties

The DIET kernel defines a hierarchy of elements that provide a basis for creation of multi-agent systems:

- Worlds
- Environments
- Agents
- Connections
- Messages.

Agents reside in environments. Environments implement a "physics" for DIET agents. Worlds are placeholders for environments - a world manages functionality shared between environments. A world is also the access point to the kernel for debugging and visualisation components.

Agents are at the centre of the DIET element hierarchy. Agents are defined in terms of addresses. This consists of an *environment address*, determined by the environment in which it resides, and an *agent identity*, defined when the agent is created. The agent identity has two components, a *name tag* specific to that agent, and a *family tag* common to other agents with the same functionality. The family tag allows for groups of agents with common function to be identified by an application. The name tag allows individual agents to be differentiated.

Agents have limited initial behaviour (which can be extended as required by application developers). The environment provides functionality for agents to *create* other agents, *migrate* to different environments, *communicate* with other agents, and to *self-destruct* if they are no longer required (but they cannot destroy other agents).

The kernel implementation is "resource constrained" and "fail-fast". The kernel actions are *resource constrained* because there are explicit limits on the resources that can be used. The kernel actions are *fail-fast* because when an action cannot be executed instantaneously, it fails immediately. The fail-fast, resource constrained implementation of the kernel actions protects the system against overload. When the system becomes overloaded, it offers basic protection by rejecting actions and throwing exceptions, which provide feedback to agents. Additional agent capabilities can be created as part of applications.

2.3 Application Properties

2.3.1 Decentralisation

DIET agents can be distributed across multiple environments and multiple worlds if needed. No agent need be the controlling agent, or server, so there is no obligation for a client-server architecture. Consequently the DIET platform is appropriate for peer-to-peer applications [4] which can be used to implement decentralised control. The flexibility of peer-to-peer configurations means that decentralisation does not prevent some sort of localised centralisation, where this is useful [20].

2.3.2 Scalability

Low resource use by agents in a multi-agent system can facilitate scalability. In the DIET platform, low resource demand by individual agents, combined with thread-sharing between agents, means that a very large number of agents can be maintained simultaneously. The specific amount of resources required by each agent, and the resource overhead of the platform are application-dependent, but the initial system design facilitates low resource requirements. DIET agents use only one thread each (at most). If insufficient threads are available, agents can temporarily give up their thread, only to reclaim it later. Scalable use of system resources was demonstrated across a 32-processor Beowulf cluster supporting 100000 agents [19].

2.3.3 Adaptability

DIET agents can adapt by collaboration among a large number of agents. A demonstration of this is in the use of DIET agents as individuals in an evolutionary algorithm, evolving agent preferences [11, 20]. The application is a collaboration tool. Each user is represented by a user agent, deployed in a DIET environment. The user agent carries information about its user's interests, indicating the preferred area of collaboration. Multiple users participate via this application, either on the same or different machines. Environments are linked in a network, intended to approximate a fully connected decentralised peer network [20].

User agents deploy scout agents to interact via the peer network. The scout agents travel to other environments to interact with other scout agents. The preference of scout agents for environments is selected via an evolutionary algorithm using scout agents as individuals. Over multiple generations populations of scout agents will adapt their behaviour to facilitate interaction between distributed users with common interests [20]. This algorithm represents a means of mediating collaboration between distributed users, and supporting adaptability of agents.

3 Self-organised Applications

DIET agents can draw upon self-organisation, forming persistent communities that arise from the interactions between the agents rather than from central control. Two examples are presented here: self-organising communities [29] and P2P-DIET [16].

3.1 Self-organising Communities

Information sharing among multiple users can be organised in several ways. Centralised solutions [3, 17] can provide effective applications, but they do depend on a single agent not getting overloaded. Decentralised solutions [1, 26] have an advantage, and can self-organise to respond to changes in user behaviour.

Self-organising communities [29] supports community formation among users with common interests. User agents represent users. User agents each register with one or more middle agent. User agents (requestors) send queries about their user’s interests to middle agents. Given these queries, each middle agent then searches among the

information it holds from other user agents registered with it. If it can respond to a query based on this information then the search is completed. Otherwise, the middle agent communicates with other middle agents in order to try and obtain the information. Once this has been done, the middle agent relays results of the search to the requestor. Having done this, the middle agent examines whether the search was successful, in that there was a user agent (provider) matching the query. If so, the requestor and provider both gain positive awards. If not, the requestor gets a negative award.

Following award assignment, the middle agent checks whether both the requestor and provider are within its group. If not, the middle agent of the requestor transfers the provider to its group, so that both user agents are in the same group. User agents with lower awards move towards user agents with higher awards. The consequence of the award scheme is that the processing of queries stimulates the formation of communities, made up of users with common interests. The querying behaviour of user agents builds up a profile of their interests, which is updated by successive queries. This is a highly scalable process that operates efficiently with many users [29, 6].

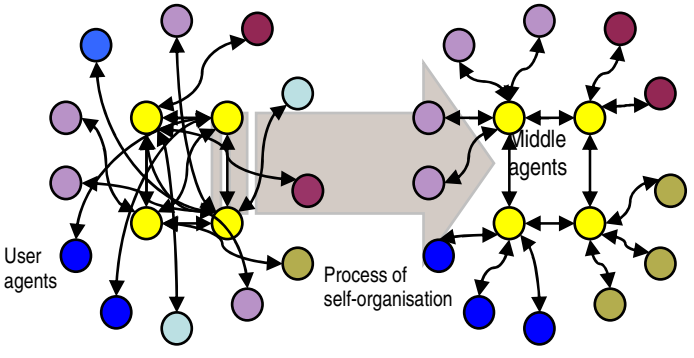


Fig. 1. Self organisation of agent communities - different colours represent different user interests

Experiments were carried out with simulated data on the formation of user communities, and time and efficiency of searches based on the user communities formed around middle agents [29]. Multiple experiments always showed self-organisation in terms of the formation of user communities of user agents around middle agents. Measurement of self-organisation activity was carried out through analysis of the success rate of queries. Investigation of success rates of query results and time taken to search data showed an increase in success rate once communities of users had started to form, culminating in a success rate close to the optimal rate once user communities had reached a stable state. The query time for individual queries also declined as a consequence of the formation of user communities around middle agents. The number of queries per user required for communities to reach a stable state increased with the number of users, but in a near linear manner, so that performance scaled effectively [29].

3.2 Implementing Peer-to-Peer Systems with P2P-DIET

The DIET platform has also been used to implement the extensible P2P service P2P-DIET [14, 16, 24]. P2P-DIET is a super-peer system and has two kinds of nodes: super-peers and clients (see Fig. 2). Each node is implemented in a DIET environment. Super-peers are equal and have the same responsibilities. Each super-peer serves a fraction of the clients and keeps indices on the resources of those clients to be able to answer queries efficiently. Clients can run on user computers where resources are also stored. Clients interact directly with one another to access resources.

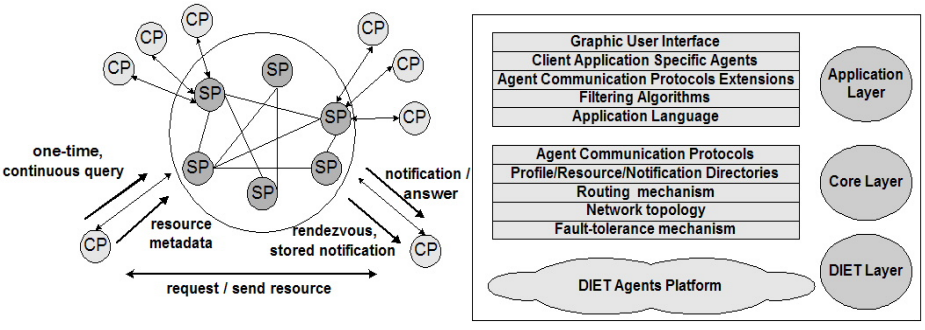


Fig. 2. P2P DIET

P2P-DIET supports the typical one-time query scenario of P2P networks. Answers are returned to the access point of the client originating the query and are then passed to the client for further processing. P2P-DIET also supports long-standing (continuous) query scenarios. Clients may subscribe to the system with a continuous query expressing their long-standing information needs. Whenever a resource is published at an access point, P2P-DIET makes sure that clients with profiles matching the metadata of this resource are notified.

A client can be connected to P2P-DIET through a single super-peer node, which is the access point of the client. Clients are allowed to migrate to a different access point and can use dynamic IP addresses. Clients can connect, disconnect or even leave the system silently at any time. When a client is off-line, notifications matching its continuous queries are stored by the access point of the client and are delivered to it the next time that it connects to the network. P2P-DIET provides message authentication and message encryption using public key cryptography. Public/private keys are also used to securely identify peers since it is not possible to identify a peer from its IP address because peers may use dynamic IP addresses.

The P2P-DIET implementation makes use of the capabilities of lightweight mobile agents offered by the DIET kernel to implement various local management tasks and P2P protocols. For example, in each super-peer environment, a *data management* agent keeps indices on resource meta-data and continuous queries to achieve scalable query processing and filtering by each super-peer, while a *router agent* achieves correct flow of network messages by using shortest paths and minimum-weight

spanning trees. For each ad-hoc query posed by a user, a *query answering* agent starts from the user's client environment and, using information from router agents, migrates to all super-peers to find all resources that match the user query. Similarly, for each continuous query subscribed by a user, a s*ubscriber agent* starts from the user's client environment and migrates to appropriate super-peer environments to subscribe the query.

P2P-DIET shows that the DIET Agents platform with its self-organization capabilities can be used to develop large scale P2P applications. The P2P-DIET application takes advantage of self-organisation at two levels: the super-peer network and the DIET Agents platform that is used to realize P2P-DIET. At the first level, P2P-DIET super-peers self-organize into a network which deals efficiently with pull and push information requests while adapting to super-peer joins, leaves or failures. At the second level, lightweight DIET Agents self-organize to implement P2P-DIET functionalities. The operation to be performed by an agent is not determined by any kind of higher level of control. On the contrary, its current status and the state of its environment determine agent decisions. Newly created agents are independent to travel around the network and collect query answers.

P2P-DIET is an extensible system for the development of applications supporting the two scenarios discussed above (see the layered view of the system on the right-hand side of Figure 2). In the current demo of the system publications and subscriptions are expressed using a well-understood attribute-value model called *AWPS* in [16]. *AWPS* is based on named attributes with value text interpreted under the Boolean and VSM or LSI models. The query language of *AWPS* allows Boolean combinations of comparisons *A op v*, where *A* is an attribute, *v* is a text value and *op* is one of the operators "equals", "contains" or "similar" ("equals" and "contains" are Boolean operators and "similar" is interpreted using the VSM or LSI model of Information Retrieval. E.g., upon receiving a new publication, a P2P-DIET node can filter 3 millions of long-standing queries in just under 200 milliseconds.

4 Discussion

We have considered two applications using self-organisation that build upon the properties of the DIET Agents platform. The self-organising communities application demonstrates the capability to organise user agents into communities of interest that facilitate information exchange. This community formation varies with increasing numbers of users and hence increasing numbers of agents, but produces a mean time of response to query that scales almost linearly with number of users. This shows not only a versatility for different sizes of problem with different numbers of users, but also, since community formation appears to always occur, self-organisation that appears not to be disrupted by different starting conditions. Self-organisation between agents leads to conditions that are more productive for individual agents than those in which they started. Furthermore, it ensures reliability of the application as queries from user agents are solved.

P2P-DIET shows how self-organisation using the DIET Agents platform can be used to develop large scale P2P applications. Although distributed P2P applications can be built without using agents, the DIET agents platform allowed us to develop such applications quickly and effectively. However, more experience with such

applications is necessary in order to come up with a general methodology for the development of self-organising software systems (presumably, this methodology will be based on traditional software engineering principles as well as new inspirations from self-organization). The use of self-organization and the sophisticated indexing schemes of [27] helped us to achieve high performance and scalability in P2P-DIET. We have not yet carried out detailed experimentations using P2P-DIET beyond these reported in [27], so it is not clear at this stage how much of the scalability of P2P-DIET is due to indexing and how much to the self-organizing protocols utilized.

Both the examples discussed here address problems of information retrieval among a diverse community of users. Multi-agent based applications bring the capability to partition information across multiple agents, and thus to break up the problem into smaller parts. This also allows for decentralisation of the control of the application, giving greater robustness. An element of self-organisation makes control easier in a decentralised network, as elements associate without additional external stimulus.

Both applications implement peer-to-peer networks in the form of super peer networks. These have been shown to have advantages over pure peer-to-peer networks, in terms of efficiency of search [32]. The applications discussed here show some of the usefulness of super peer networks. Further work needs to be done to investigate the performance of applications based on these networks in relation to other peer-to-peer information retrieval applications, and the consequences of adjusting properties of the networks used on application performance.

These examples have shown ways how lightweight multi-agent systems can be used to leverage self-organisation among agents to produce information management applications. The applications here draw upon relatively simple forms of self-organisation and of information representation. Next steps for the development of such applications include considering extensions to the network architecture linking agents, and investigating different types of queries or responses from such agent-based collaboration tools. Further development of applications in this area must follow further the potential for peer-to-peer computing [2, 4] and autonomic computing [13] to support the development of robust applications.

Acknowledgements

We thank the participants of the DIET project, and the European Commission for providing funding (project number IST-1999-10088).

References

[1] Athanassiou, E Chirichenescu, D Gleizes, MP Glize, P Lakoumentas, N Sclenker, H Leger, A Moreno, JI 1999 Abrose: A cooperative multi-agent based framework for marketplace. IATA, Stockholm, 1999.

[2] Babaoglu, O Meling, H Montresor, A 2002 Anthill: A framework for the development of ant-based peer-to-peer systems. Proc. IEEE Intl. Conf. Distributed Computer Systems, 2002, pages 15-22.

[3] Bayardo, RJ Jr, Bohrer, W Brice, A Cichocki, J Fowler, A Helal, V Kashyap, T Ksiezyk, G Martin, M Nodine, M Rashid, M Rusinkiewicz, R Shea, C Unnikrishnan, A Unruh, A Woelk, D 1997 Infosleuth: agent-based semantic integration of information in open and dynamic environments. Proc. ACM SIGMOD-97, 1997.

[4] Bonsma, E Hoile, C 2002 A distributed implementation of the SWAN peer-to-peer look-up system using mobile agents. In: 1st Workshop on Agents and Peer-to-Peer Computing, Bologna, 2002.
[5] Camazine, S, Deneuborg, J-L, Franks, NR, Sneyd, J Theraulaz, G Bonabeau, E 2001 *Self-Organization in Biological Systems.* Princeton University Press.
[6] Cid-Sueiro, J Wang, F 2002 A scalability analysis of self-organising agent communities. Learning02 Workshop, Madrid, 2002.
[7] Decker, K Sycara, K Williamson, M 1997 Middle-agents for the internet. Proc. IJCAI-97, pp. 578-583.
[8] De Wilde, P Chli, M Correira, L Ribeiro, R Mariano, P Abramov, V Goossenaerts, J 2003 Adapting populations of agents. In: *Adaptive Agents and Multi-Agent Systems*, Alonso, E., Kudenko, D. & Kazakov, D. (eds.) pp. 110-124, LNAI 2636, Springer.
[9] DIET Open Source web site: http://diet-agents.sourceforge.net/Index.html
[10] Di Marzo Serugendo, G Foukia, N Hassas, S Karageorgos, A Mostefaoui, SK Rana, OF Ulieru, M Valckenaers, P Van Aart, C 2004 Self-organisation: paradigms and applications. In: *Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering*, Di Marzo Serugendo, G Karageorgos, A Rana, OF Zambonelli, F (eds.), pp. 1-19. LNA 2977, Springer.
[11] Hoile, C Wang, F Bonsma, E Marrow, P 2002 Core Specification and Experiments in DIET: A Decentralised Ecosystem-inspired Mobile Agent System. Proc. 1st Int. Conf. Autonomous Agents and Multi-Agent Systems (AAMAS2002), 2002, pp. 623-630.
[12] Horling, B Mailer, R Lesser, V 2004 Farm: a scalable environment for multi-agent development and evaluation. In: *Advances in Software Engineering for Multi-Agent Systems*, Lucena, C Garcia, A Romanovsky, A Castro, J Alencar, P (eds.) pp. 220-237. Springer.
[13] Horn P 2001 Autonomic computing: IBM's perspective on the state of information technology. IBM Autonomic computing manifesto, IBM 2001.
[14] Idreos, S Koubarakis, M Tryfonopoulos, C 2004 P2P-DIET: An Extensible P2P Service that Unifies Ad-hoc and Continuous Querying in Super-Peer Networks. ACM SIGMOD Conference 2004, Demo Paper, pp. 933-934.
[15] JADE web site: http://jade.tilab.com/
[16] Koubarakis, M Tryfonopoulos, C Idreos, S Drougas, Y 2003 Selective Information Dissemination in P2P Networks: Problems and Solutions. *ACM SIGMOD Record*, Special issue on Peer-to-Peer Data Management, Aberer, K (ed.), 32(3), September 2003.
[17] Kuokka, D Harada, L 1995 Supporting Information Retrieval via Matchmaking. In: AAAI Spring Symposium on Information Gathering, 1995.
[18] Lengen, RH van, Bähr, T Hagen, H Marrow, P Bonsma, E Hoile, C 2004 Component based visualisation of DIET applications, *Proc. Dagstuhl Workshop on Scientific Visualisation 2003*, G.-P. Bonneau, T. Ertl and G.M. Nielson (eds.), Springer, 2004.
[19] Marrow, P 2001 Scalability in multi-agent systems: the DIET project. Workshop on Infrastructure for Agents, Multi-Agent Systems and Scalable Multi-Agent Systems at Autonomous Agents 2001.
[20] Marrow, P Hoile, C Wang, F Bonsma, E 2003 Evolving preferences among emergent groups of agents. In: *Adaptive Agents and Multi-Agent Systems*, Alonso, E., Kudenko, D. & Kazakov, D. (eds.) LNAI 2636, Springer, 2003, pp. 159-173.
[21] Menezes, R Tolksdorf, R 2004 Adaptiveness in Linda-Based Coordination Models. In: *Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering*, Di Marzo Serugendo, G Karageorgos, A Rana, OF Zambonelli, F (eds.), pp. 212-213. LNAI 2977, Springer.

[22] Montresor, A Meling, H Babaoglu, O 2002 Messor: load-balancing through a swarm of autonomous agents. Proc. 1st Workshop on Agents and Peer-to-Peer Systems, Bologna, 2002.
[23] Nicholis, G Prigogine, I 1989 *Exploring Complexity: an introduction.* W.H. Freeman.
[24] P2P-DIET web site: http://www.intelligence.tuc.gr/p2pdiet
[25] Steen, M van, van der Zijden, S Sips, H 1998 Software engineering for scalable distributed applications. In Proceedings 22nd International Computer Software and Applications Conference (CompSac), 1998.
[26] Takada, Y Mohri, T Fujii, H 1998 Multi-agent system for virtually integrating distributed databases. Fujitsu Sci. Tech. J 34(2), 245-255.
[27] Tryfonopoulos, C Koubarakis, M Drougas, Y 2004 Filtering Algorithms for Information Retrieval Models with Named Attributes and Proximity Operators, Proceedings of the 27th Annual ACM SIGIR Conference. July 25-July 29, 2004, Sheffield.
[28] Vincent, R Horling, B Lesser, V 2001 An agent infrastructure to build and evaluate multi-agent systems: the Java Agent Framework and Multi-Agent System Simulator. In: *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, Wagner & Rana (eds.). LNAI 1887, Springer.
[29] Wang, F 2002 Self-organising communities formed by middle agents. Proc. 1st Int. Conf. Autonomous Agents and Multi-Agent Systems (AAMAS2002), 2002, pp. 1333-1339.
[30] Waring, RH 1988 Ecosystems: fluxes of matter and energy. In: Cherret, JM (ed.), *Ecological Concepts*, pp. 17-42, Blackwell Scientific.
[31] Wong, HC Sycara, K 2000 A taxonomy of middle-agents for the Internet. Proc. 4th Intl. Conf. Multi-Agent Systems.
[32] Yang, B Garcia-Molina H 2003 Designing a super-peer network. IEEE International Conference on Data Engineering, 2003.
[33] Zambonelli, F Jennnings, NR Wooldridge, M 2003 Developing multiagent systems: the GAIA methodology. ACM Trans. on Software Engineering and Methodology 12, 317-370.

Solving Dynamic Distributed Constraint Satisfaction Problems with a Modified Weak-Commitment Search Algorithm

Koragod Saenchai[1], Luigi Benedicenti[2], and Raman Paranjape[2]

[1] Khon Kaen University, Khon Kaen, 40002, Thailand
korsan@elec.kku.ac.th
http://www.elec.kku.ac.th
[2] University of Regina, Regina, SK, S4S 0A2, Canada
{luigi.benedicenti, raman.paranjape}@uregina.ca
http://www.uregina.ca

Abstract. Constraint Programming research is currently aimed at solving problems in a dynamically changing environment. This paper addresses the problem of solving a Dynamic Distributed Constraint Satisfaction Problem (Dynamic DCSP). The solution proposed is an algorithm implemented in a multi- agent system. A Dynamic DCSP is a problem in which variables, values and constraints are distributed among various agents. Agents can be freely added to or removed from the system. Most advanced applications cannot be represented by DCSPs, but they can be modeled by Dynamic DCSPs. The algorithm described in this paper is an extension of the Asynchronous Weak Commitment Search algorithm (AWCS) originally proposed by Yukoo [10]. The extended algorithm is designed to cope with the dynamically changing parameters of a Dynamic DCSP. The proposed algorithm differs from other Dynamic DCSP algorithms because it allows an unlimited number of changes to any of the variables, values, or constraints. This paper describes an agent system implementing the modified AWCS algorithm and verifies its effectiveness by applying it to a dynamic N-Queens problem. The results prove the applicability of the modified algorithm to Dynamic DCSP.

1 Introduction

This paper presents the dynamic extension of the Asynchronous Weak-Commitment Search algorithm (AWCS) originally proposed by Yukoo [10]. The AWCS algorithm is used to solve Distributed Constraint Satisfaction Problems (DCSPs) using a multi-agent system. A multi-agent system is a system in which a cluster of agents work or collaborate together to reach their goals. Constraint Satisfaction Problems are defined by three main components: variables, values, and constraints. To solve a DCSP, each variable must take a value that satisfies all of the constraints. In DCSPs, distributed systems are introduced to overcome the limitations of solving Constraint Satisfaction Problems using a single centralized system. The literature suggests that multi-agent systems can be used to solve DCSPs. Each agent becomes responsible for one or more distinct variables, depending on the algorithm and the specific problem.

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 130 – 137, 2006.

The original version of the Asynchronous Weak-Commitment search algorithm is limited to a static environment; the numbers of variables, values and constraints are fixed for each run. On the other hand, most real world applications, such as hospital scheduling [2],[5], airport scheduling, job-shop scheduling [8], or resource allocation[4, 6] are subject to dynamic changes due to new requirements, or when resources are being added or removed. It is difficult and impractical to have to change a system or to rerun a program every time there is a requirement change. This paper introduces an extended AWCS algorithm to solve a Dynamic DCSP. The algorithm allows one to add or remove variables, values and constraints without having to stop and restart the program or to remodel the problem. The approach proposed is innovative in that it allows an unlimited number of changes to the problem, without the need to restart the agent system.

1.1 Original Weak-Commitment Search Algorithm

The Asynchronous Weak-Commitment Search algorithm is an efficient algorithm used to solve Distributed Constraint Satisfaction Problems using Distributed Multi-Agent Systems [10]. It is a hybrid algorithm that merges the iterative improvement (Hill climbing) and min-conflict algorithms [7, 3]. In the words of the author "agents can revise a bad decision without an exhaustive search by dynamically changing the priority order of agents" [10]. The fundamental concept of the AWCS algorithm is to use the min-conflict heuristic for new value selection, and to maintain a weak commitment to a partial solution by dynamically changing the priority order of agents. A partial solution is recorded as a "nogood" solution whenever there is a search failure. Each nogood is recorded in a list to prevent the algorithm from falling into a previous situation.

2 Proof of Concept

This paper uses a dynamic version of the N-Queen Problem as a proof of concept of the modified AWCS algorithm, since the N-Queen problem is a classical constraint satisfaction problem. The goal of this problem is to place N queens in an N by N chessboard, in a way that they will not be able to eliminate each other. A set of x1, x2, …, xn represent the queens in the chessboard. The set of constraints between xi and xj is $(xi \neq xj) \cap (|i-j| \neq |xi - xj|)$. Thus, one queen can eliminate another queen if they are in the same row, or same column, or same diagonal. The system design includes three main agents: the Loader agent, the Name Service agent, and the Queen agent as shown in Fig.1. The detailed description of each agent follows.

The first agent, the loader agent, is responsible for loading all other agents. During the testing phase, the loader agent also automatically loads new Queen agents into the system to test that the environment can change dynamically. Whenever a solution is found for a certain configuration, a new Queen will be added to the system. This action changes both the number of Queens in the system and the size of the chessboard to match. Therefore, all dynamic changes are effected through the loader agent. Agents can be added at any time during the execution, but the testing script only adds them at the end of a run to test the validity of each solution before moving on to the next.

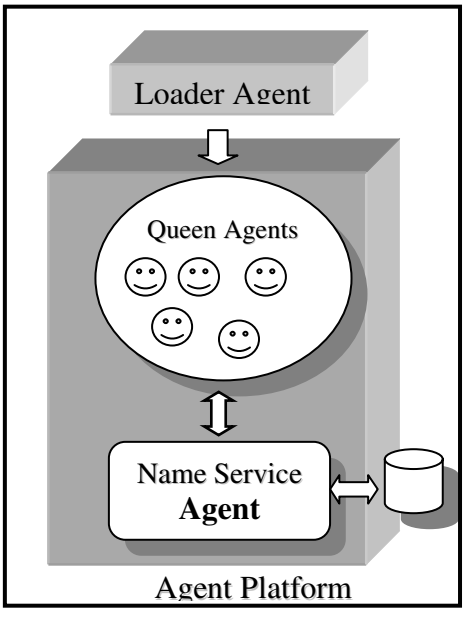


Fig. 1. Dynamic DCSPs System Design

The Name Service agent is responsible for name registration of all other agents and for putting the agent names in a list. This list is provided to every agent registering with the Name Service Agent. Notifications are sent to all agents registered with the system for any agent additions or deletions. The Name Service agent is responsible for keeping track of all the Queens that are added to or removed from the system.

Queen agents represent players in the chessboard. They can communicate by sending messages. This will help them reach a solution. Queen agents are autonomous and can move asynchronously. This means that they can make a decision based on their internal algorithm and the information they receive from other agents. Queen agents will keep listening for messages, even after a solution is found. This allows them to react to changes in the environment. Queen agents can be added to or removed from the system at any time. The number of Queens in the system defines the size of the chessboard; the size is increased if a new Queen is added to the system, and the size is reduced if a Queen is removed from the system. The remaining Queens in the chessboard adapt to the changing environment, and search for new solutions dynamically.

3 Modified AWCS Algorithm

In the modified AWCS algorithm, each Queen agent is assigned to one variable representing its row that is associated with the Queen's column position in the chessboard. Rows are determined and assigned automatically to each Queen, and are not changeable. The dynamic extension part begins as follows (Fig. 2). When the system starts, each agent registers itself with the Name Service agent. The Name Service agent adds the agent's name to a list. To ensure the dynamic character of the system, each agent is made aware of other agents being added to or removed from the system. Thus, the

number of queens and the list of all the agents representing them are made accessible to every agent. The size of the chessboard is automatically defined by the number of Queens in the system. This design does not allow a chessboard with more than one

```
When a Queen agent first created do:
   Send Register message to a Name Service agent
     Request an agent list from a Name Service agent
       Initiate an initial value and priority
          add those parameters into an Agent_View
            current value (vi) = initial value
             current priority (pi) = initial priority = 0;
end do;

When receive an agent_list from a Name Service agent
   Size of chessboard = number of agents in the list
     Send OK? (xi, value, priority) to other agents in the
list;
       end do;
```

Fig. 2. Dynamic Extension part of AWCS algorithm

```
When  receive OK? (xj, vj, pj) do:
  If agent_view already contains (xj, vj, pj)
    have already acted on that, do nothing
     else add (xj, vj, pj) to an agent_view
  if agent_list does not contain Xj
    size of chessboard = size +1;
      check agent_view; end if
        end if; end do;

When  receive nogood (xj, nogood) do:
  add nogood to nogood_list
    check_agent_view; end do;

Method Check agent_view
  When current value is not consistent with agent_view do:
    If no value is consistent with higher priority in an
agent_view
    then Backtrack;
      else do a min-conflict with lower priority agents
        current value (vi) = minimum conflict number; end if
send OK? (xj, vj, pj) to all other agents; end do;

Method Backtrack
  When an empty solution set happen, broadcast no solution
do:
   add nogood into nogood_sent
     send nogood to all other agents
       Increase pritoity -> current priotity = 1 + pmax
         Do min-conflict with lower priority agents
           Current value = min-conflict value
             Send OK? ( xi, vi, pi) to other agents; end do;
```

Fig. 3. Excerpt of the AWCS algorithm from [10]

Queen per row. After receiving the first message from the Name Service Agent, each Queen agent will send "OK?" to all other agents, and wait.

Fig. 2 and 3 constitute the modified AWCS algorithm. Queen agents may receive three different types of message: "OK?," "nogood," and "agent_list." When a Queen agent receives "OK?" it will add the three parameters coming with the message (xj, vj, pj) into its agent_view. If the "OK?" comes from a Queen xj that is not in the agent's agent_list, it will add that Queen into the agent_list and expand its value domain (or the size of the chessboard). Then it will call the check agent_view method. If the current value of the agent's variable is not consistent with the received agent_view (i.e., the constraints are not satisfied), the Queen agent will find a new value that is consistent with higher priority agents and that has the minimum conflict with lower priority agents. If there is no value left in the domain that is consistent with higher priority agents, the agent will perform a backtracking step. In the backtracking procedure, a "nogood" is sent to other agents and recorded in a nogood_list. After that the agent will increase its priority and perform a min-conflict procedure with lower priority agents. The min-conflict procedure adjusts the agent value to be in minimum conflict with the agents of lower priority. Finally, the agent will update its agent_view and send "OK?" with the new value and priority to all other agents.

4 Working Example

This part illustrates the execution of the modified AWCS algorithm for an N-Queens problem. The example starts with the 4-queens problem shown in Fig. 4(a). Each agent starts by sending its initial value and priority to all other agents. The priority of each agent is shown in parentheses. Each agent will then check its consistency with all higher priority agents. All agents have the same priority number 0, so the priority

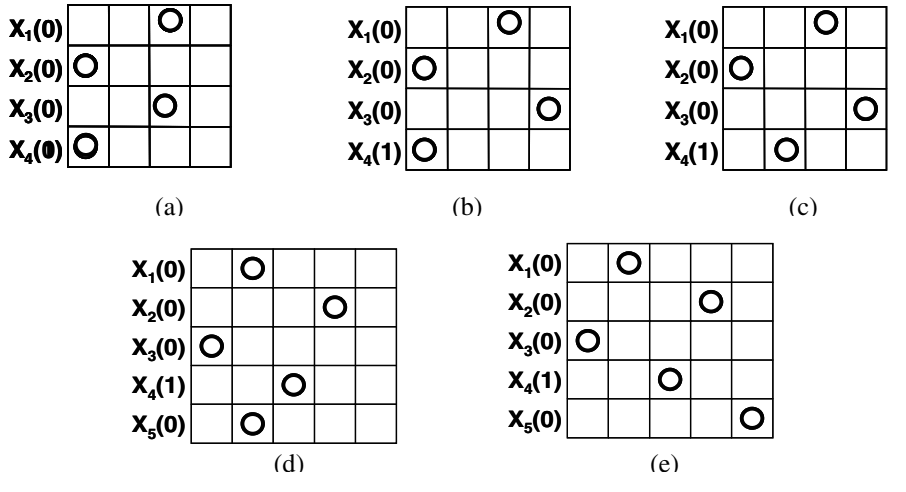


Fig. 4. Example of Algorithm Execution

will be decided by the alphabetical order of the agent names. As a result, Agents X3 and X4 are not consistent with an agent_view. The minimum conflict value for X3 is number 4, as shown in Fig. 4(b). At the same time, X4 has no consistent value, so it sends "nogood" to all higher priority agents and performs a Backtracking step. Since there is more than one minimum conflict value to choose from, the agent will choose the value according to its alphabetical (or numerical, in this case) order. In this case the current value is also a minimum conflict number, but the agent will ignore the current value and choose the next number. Therefore, X4 will choose value number 2, and a solution will be found.

After a solution is found, all agents remain waiting for messages. Fig. 4(d) shows what happens when a new Queen is added to the chessboard. The chessboard will now become a 5 by 5 chessboard. The new Queen will initialize its initial value and priority and will send those parameters to all agents. Therefore, all other agents will be made consistent with the late updated agent_view. Only X5 will find itself not consistent with agent_view, so it will move to column number 5 and a solution will be found. Then, all agents will wait for new messages.

5 Results

A set of tests was performed to verify the effectiveness of the algorithm. The algorithm was applied to a series of N-Queen problems as follows. Initially, the algorithm was tested on static problems with 4, 5, and 6 queens respectively. Since solutions to all these problems already exist in the literature, it was simple to prove that the algorithm is able to find a solution to the static N-Queens problem. An analysis of the steps followed by the algorithm showed that the solution trace mimics the steps of the standard Asynchronous Weak-Commitment Search algorithm.

Subsequently, the algorithm was tested with a dynamic problem as follows. The test started with 4 Queens, and whenever a solution was found, a new queen was added to the system automatically. This configuration was chosen to confirm that the static solution was found for every number of queens. The results show that the system can adapt to new changes by increasing the size of the chessboard to match the new number of Queens and find valid solutions. The dynamic algorithm was run for a large number of times (greater than 100) with an empty chessboard at the beginning.

Since every agent runs in a separate thread, and thread scheduling is not regulated by the platform but rather by the operating system, every time the platform was run there was a simple randomization in the start time of agents. Moreover, message exchange is not serialized in any way in the agent platform chosen for the implementation of the agent system. The order of arrival of messages to the Name Service agent was thus not guaranteed. This ensured that a sufficient amount of randomization occurred between runs.

The analysis of the execution traces logged by the agent platform revealed that the "nogood" part of the algorithm was not needed for fewer than 8 queens, but for 8 or more queens, the "nogood" algorithm successfully prevented backtracking loops. The size of the "nogood" list is not examined in this paper, and will be the subject of a detailed analysis in the future.

6 Conclusions and Future Work

The main contribution in this paper is an extension of the Asynchronous Weak-Commitment Search algorithm that deals with dynamic Distributed Constraint Satisfaction Problems. The extension makes it possible to vary any number of parameters of the problem at any given time and still maintain consistency in order to find a solution.

The algorithm is still currently being studied. Its performance is the same as that of the AWCS algorithm, but its dynamic capacity allows it to consider new variables, values, and constraints. A formal analysis of the usefulness of this feature has not been performed yet, but it is safe to assume that for incremental problems such as dynamic timetables (in which the changes occur as increments to an existing timetable) there is a definite advantage in starting with a known good configuration.

In the future, the authors intend to pursue further the dynamic part of the algorithm and to further improve the algorithm possibly with stochastic modeling coming from the additional knowledge that the algorithm has of the dynamic changes. This will be particularly useful in situations where there is high variability. Further analyses on optimal deployment of agent platforms will also follow.

References

1. Bessiere C. Arc-consistency in dynamic constraint satisfaction problem. In Proceeding of the Ninth National Conference on Artificial Intelligence, pp. 221-226, 1991.
2. Hannebauer M., and S. Müller. Distributed constraint optimization for medical appointment scheduling. Proceedings of the fifth international conference on Autonomous agents, p. 139-140, 2001
3. Minton S, M.D. Johnson, A.B. Phillips, and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. Artificial Intelligence 161-205, 1992.
4. Modi P.J., H. Jung, M. Tambe, W. M. Shen, and S. Kullkarni. Dynamic Distributed Resource Allocation: A Distributed Constraint Satisfaction Approach, Revised Papers from the 8th International Workshop on Intelligent Agents VIII, p.264-276, August 01-03, 2001
5. Paulussen T. O., A. Zöller, A. Heinzl, A. Pokahr, L. Braubach, and W. Lamersdorf. Dynamic patient scheduling in hospitals. In Coordination and Agent Technology in Value Networks, 2004
6. Rybski P., S.A. Stoeter, M. Gini, D. F. Hougen, N. Papanikolopoulos. Dynamic Scheduling of Fixed Bandwidth Communications channels for controlling multiple robots. In Proceedings of the fifth international conference on Autonomous agents, p. 153-154, 2001
7. Selman B, H. Levesque, and D. Mitchell. A new method for solving hard satisfaction problems. In Procedding of the Tenth National Conference on Artificail Intelligence, pp. 440-446, 1992.
8. Tumer K., and J. Lawson. Collectives for multiple resource job scheduling across heterogeneous severs. In Proceedings of the second international joint conference on Autonomous agents and multiagent systems, p. 1142-1143, 2003.

9. Yokoo M. Asynchronous waek-commitment search for solving distributed constraint satisfaction problems. In Proceeding of the first International Conference on Principles and Practice of Constrant Programming, pp. 88-102. Springer-Verlag. Lecture Notes in Computer Science 976, 1995.
10. Yokoo M. Distributed Constraint Satisfaction: Foundation of cooperation in multi-agent systems. Springer, 2001.
11. Yokoo M. Weak-Commitment search for solving constraint satisfaction problems. In Proceeding of the Twelth Natinal Conference on Artificial Intelligence, pp. 313-318, 1994.

Development of Self-organising Emergent Applications with Simulation-Based Numerical Analysis

Tom De Wolf[1], Tom Holvoet[1], and Giovanni Samaey[2]

[1] AgentWise@DistriNet Research Group
[2] Scientific Computing Research Group,
Department of Computer Science, KULeuven,
Celestijnenlaan 200A, 3001 Leuven, Belgium
{Tom.DeWolf, Giovanni.Samaey, Tom.Holvoet}@cs.kuleuven.be

Abstract. The goal of engineering self-organising emergent systems is to acquire a macroscopic system behaviour solely from autonomous local activity and interaction. Due to the non-deterministic nature of such systems, it is hard to guarantee that the required macroscopic behaviour is achieved and maintained. Before even considering a self-organising emergent system in an industrial context, e.g. for Automated Guided Vehicle (AGV) transportation systems, such guarantees are needed. An empirical analysis approach is proposed that combines realistic agent-based simulations with existing scientific numerical algorithms for analysing the macroscopic behaviour. The numerical algorithm itself obtains the analysis results on the fly by steering and accelerating the simulation process according to the algorithm's goal. The approach is feasible, compared to formal proofs, and leads to more reliable and valuable results, compared to mere observation of simulation results. Also, the approach allows to systematically analyse the macroscopic behaviour to acquire macroscopic guarantees and feedback that can be used by an engineering process to iteratively shape a self-organising emergent solution.

1 Introduction

In an industrial research project EMC^2 [1], we examine the possibilities of a *self-organising emergent* solution for an Automated Guided Vehicle (AGV) warehouse transportation system. In this case study a group of AGVs has to transport incoming loads from pick up locations to specific destinations in the warehouse. Experience has shown that the current centralised system has problems with scalability because it cannot handle many AGVs efficiently. Also, the system is not flexible, i.e. the current solution cannot handle frequent changes in the transportation problem and it needs to be customised and optimised each time the system is deployed. There is a need for a decentralised system that adapts itself to each different situation. In [2], both 'emergence' and 'self-organisation' are defined and a combination of both is a promising approach for a complex and dynamic system such as the AGV system. Other examples of systems where self-organised emergent solutions are advantageous are telecommunication networks, flexible manufacturing networks, and dynamic traffic networks.

The goal of engineering self-organising emergent systems is to acquire a system with a coherent macroscopic behaviour which meets the requirements and results solely from

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 138–152, 2006.

autonomous local activity and interaction. *Macroscopic* is defined as being observed as an overall or global pattern, structure, or behaviour of the system as a whole. Despite multiple initiatives, how to systematically build such a system remains an open issue. Today, such systems are mostly built in an ad-hoc manner. However, experience in the industrial project EMC^2 [1] revealed that before even considering a self-organising emergent system in an industrial context, guarantees are needed that the required coherent macroscopic behaviour is achieved and maintained.

This paper describes an approach that allows to systematically analyse the macroscopic behaviour to guarantee that the requirements are achieved. Realistic agent-based simulations are combined with existing scientific numerical analysis algorithms for dynamical systems. This combination leads to more reliable and valuable results, compared to mere observation of simulation results, because the analysis algorithm itself obtains the results on the fly by steering and accelerating the simulation process according to the algorithm's goal. In order to also achieve a more systematic approach for building self-organising emergent systems, it is proposed to integrate the analysis approach into the engineering process such that a constant feedback loop between scientific analysis and engineering shapes a self-organising emergent solution. In the end, a systematic simulation-based engineering process can be achieved.

The paper is structured as follows. In section 2 the AGV case study that is used throughout the paper is described and motivated. Then, section 3 describes the analysis approach and how to systematically apply it. Section 4 discusses how to systematically engineer a self-organising emergent system by exploiting scientifically founded analysis results. Finally, a conclusion and some directions for future work are given.

2 The Case Study: Automated Guided Vehicles

In an industrial research project EMC^2 [1], our group develops self-organising emergent solutions for AGV warehouse transportation systems. Experience revealed an industrial need for guarantees about the macroscopic behaviour. Therefore, the AGV case is used throughout the paper. The automated industrial transport system, that we consider, uses multiple transport vehicles. Such a vehicle is called an AGV and is guided by a computer system (on the AGV itself or elsewhere). The vehicles get their energy from a battery and they move packets (i.e. loads, materials, goods and/or products) in a warehouse. Each individual AGV is capable of only a limited number of local activities such as move, pick packet, and drop packet. The goal of the system is to transport the incoming packets to their destination in an optimal manner.

A simulator[1] is developed for such an AGV system that allows to execute the simulations needed in the analysis approach described later in this paper. The screen-shot in figure 1 illustrates the problem setting described above: the locations where packets must be picked up are located at the left of the figure, the destinations are located at the right, and the network in-between consists of stations and connecting segments on which the AGVs move. Segments are unidirectional. A bidirectional segment is constructed using two overlapping unidirectional segments. Packets are depicted as rectangular boxes and some of the AGVs shown hold a packet, others do not.

[1] http://www.cs.kuleuven.be/~distrinet/taskforces/agentwise/agvsimulator/

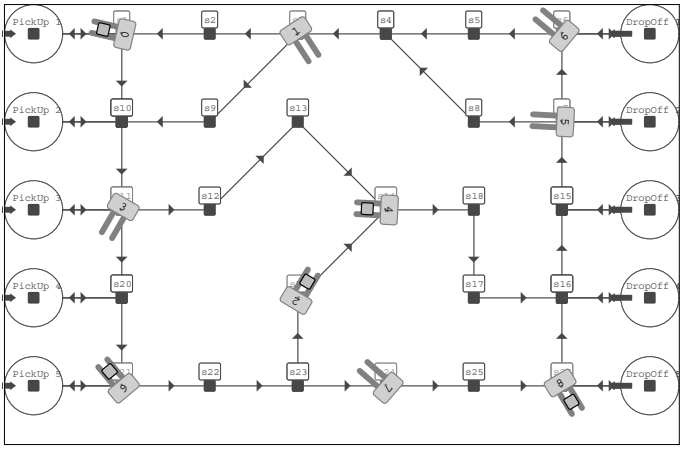


Fig. 1. Screen-shot of AGV Simulator

The AGV problem is a dynamic problem with non-deterministic features (e.g. packets can arrive at any moment, AGVs can fail, obstacles can appear). The project with our industrial partner [1] has shown that a solution using a central server to control all AGVs cannot handle the frequent changes efficiently. The central server has to constantly monitor the warehouse and each AGV to detect and react to changes by steering each AGV. Because AGVs are moving constantly, reacting on changes has to occur instantaneously. The central server becomes a bottleneck in the presence of frequent changes. As a consequence, the current central solution is not scalable and can only handle a limited number of AGVs. Also, the system is not flexible when it has to be deployed, i.e. the system needs to be customised and optimised each time it is deployed in another warehouse. Therefore, larger and dynamic AGV systems require a self-organising emergent solution in which the AGVs adapt to the changing situations themselves by only using locally obtained information, local interactions, and local activity. However, before even considering a self-organising emergent solution for the AGV system, guarantees are needed that the required coherent macroscopic behaviour is achieved and maintained. Thus, an analysis approach integrated into the engineering process, that offers guarantees, is required.

3 The Analysis Approach

The main question here is how to know if a certain self-organising emergent system exhibits the required macroscopic behaviour. The analysis approach described in this section focusses on how to analyse the macroscopic behaviour of such a system and should result in guarantees about that macroscopic behaviour. Firm guarantees could be obtained if the system is modelled formally and the required macroscopic behaviour is proofed analytically. However, constructing a formal model and correctness proof of a complex interacting computing system is infeasible. Wegner [3] proves this based on the fact that computing systems using interaction are more powerful problem

solving engines than mere algorithms. Interaction models are even so powerful that they can be denoted to be incomplete, in the mathematical sense. Completeness ensures that all possible behaviour is modelled. Wegner shows that one cannot model all possible behaviour of an interaction model and thus formally proving correctness of interactive models (e.g. self-organising emergent systems) is not merely difficult but impossible.

The alternative is to use an empirical and scientifically founded method to analyse the macroscopic behaviour, which is also advocated in [4, 5]. Empirical analysis requires focussing on relevant properties and ignoring irrelevant ones. Macroscopic properties are typically quantified with measurable variables which we define as *macroscopic variables*. Using macroscopic variables, i.e. an incomplete representation of the system behaviour, is not problematic because physicists achieve their pragmatic goals of prediction and control also by dealing entirely with incomplete observable representations.

Section 3.1 focusses on what is important to analyse in self-organising emergent systems, i.e. which macroscopic variables. Before any analysis is done, the system needs to be modelled. Section 3.2 discusses the distinction between aggregate-based models and individual-based simulation models and why the latter is to be preferred. Section 3.3 describes the analysis approach that combines realistic individual-based simulation models with an existing arsenal of numerical analysis algorithms to analyse the behaviour of self-organising emergent systems. And finally, section 3.4 discusses interesting macroscopic properties in the AGV case.

3.1 Analysis of Self-organising Emergent Systems: Trends

Before the analysis approach is outlined, we first need to define the results that are expected from the analysis of self-organising emergent solutions. What kind of macroscopic properties and variables are we interested in?

Self-organising emergent systems promise to be scalable, robust, stable, efficient, and to exhibit low-latency [6], but also behave non-deterministically. Even if the required macroscopic behaviour is achieved, the exact evolution is not predictable [6]. However, self-organising emergent systems exhibit *trends* that are predictable. A trend is defined to be the evolution of the macroscopic behaviour when the average is taken over a number of system runs. Due to the dynamics of self-organising emergent systems, robustness is preferred to an optimal macroscopic behaviour. Optimality is only achieved when the conditions in which the system operates remain rather static. But a static situation will never be reached in the presence of frequent changes. Preferring robustness above optimality implies that most temporal deviations from the required behaviour are allowed as long as the required behaviour is maintained in a trend, i.e. in the evolution of the average behaviour. Temporal deviations are often necessary to explore the space of possibilities, and to counteract the frequent system changes.

Therefore, the main results expected from an analysis of the macroscopic behaviour of self-organising emergent systems are statements about the macroscopic behaviour that ensure the required trend in the evolution of that behaviour. We define such statements as *macroscopic guarantees*. The analysis approach described in section 3.3

focusses on the analysis of trends in order to obtain macroscopic guarantees. Other issues of the dynamics of the system (i.e. microscopic behaviour, frequency of deviations and how large they are, etc) are also important but are outside the scope of this paper.

3.2 Modelling: Individual-Based Versus Aggregate-Based

Before any analysis is done, a model of the analysis subject (i.e. the macroscopic behaviour) is needed. Generally, there are two kinds of models. First, there are so called *aggregate-based models* which are constructed using macroscopic variables as building blocks and defining the relationships between those variables. One implicitly assumes that the evolution of the macroscopic variables is the result of the behaviours of individuals in the system. Traditionally such models are *equation-based models* where the evolution of an equation represents the evolution of the macroscopic behaviour. Second, there are *individual-based models* which consist of a set of agents that explicitly encapsulate the behaviours of the various individuals that make up the system. The macroscopic behaviour is then observed by running a simulation of the model.

In the context of self-organising emergent systems, individual-based models have a number of advantages with respect to equation-based models (see [7]). The most relevant advantages are:

- Individual-based models are easier to construct. They are a natural 1-on-1 mapping between the system and the model. Often such a model just equals the system. Individual-based models are appropriate for domains characterised by a high degree of localisation and distribution which is the case with self-organising emergent systems. Equation-based models are often infeasible because such mathematical formal models are impossible to construct for complex interacting systems as discussed at the beginning of section 3 and in [3]. Even if they are possible, the model would be too complex for reasonable manipulation and comprehension.
- Individual-based models make it easy to adjust the model in order to play "what-if" games without translating into or constructing a new equation-based model.
- Equation-based models may yield less realistic results compared to an individual-based model. This is mainly due to the simplification that is often required to construct an equation while every detail in the individual behaviours can have a significant impact on the macroscopic result. For example, Wilson [8] offers a detailed study that compares individual-based models and equation-based models for a predator-prey system and finds that the equation-based models can result in qualitatively different behaviours compared to the real behaviour, especially due to the stochastic behaviour in the individual-based simulation.

On the other hand, equation-based models are very popular mainly due to the availability of a whole arsenal of numerical tools and techniques for analysing system dynamics. In contrast, individual-based models lack such scientific techniques. Mere observations of individual-based simulations only results in reliable guarantees if a lot of simulations are executed for a long time. In the context of self-organising emergent systems, simulations are expensive and thus the amount of simulation time needs to be minimised.

Parunak [7] argues that one should choose for either individual-based modelling or for equation-based modelling depending on the problem at hand. However, the analysis approach described in section 3.3 combines the advantages of both approaches, i.e. realistic individual-based simulation models and aggregate-based numerical analysis. The need for equations is eliminated and the simulation process is accelerated, i.e. according to the the goal of the analysis algorithm the amount of expensive simulation time is limited to only essentially necessary simulations.

3.3 "Equation-Free" Macroscopic Analysis

This paper uses a *"equation-free" macroscopic analysis* approach [9, 10], that combines numerical analysis algorithms and realistic individual-based simulation models. Traditionally, numerical analysis is applied to equation-based models. The macroscopic behaviour is modelled by an equation (a macroscopic equation) and numerical algorithms are used to obtain quantitative statements about the macroscopic properties. However, as discussed earlier, in complex and dynamical systems, deriving a macroscopic equation is often not possible, unless the system is very simple. The "equation-free" approach resolves this issue by replacing the equation-based model by a realistic individual-based simulation model. This approach also analyses simulation measurements, but, in contrast to mere observation of simulation results, the numerical analysis algorithms acquire the results themselves by steering the simulation process towards the algorithm's goal. For example, a numerical algorithm could have as its goal to find a steady state behaviour if present, i.e. the measured behaviour converges to a single value. Another goal could be to extrapolate the behaviour as far as possible in time. The advantage is that the results are calculated on the fly and only those simulations are executed that are actually needed to obtain a specific result. The latter reduces the computational effort drastically compared to mere simulations where one typically simulates for a huge number of time steps starting from time step 0. The results are of equal or even better scientific value as the equation-based analysis because the same scientific numerical algorithms are used and possible discrepancies between the model and the real system dynamics (e.g. [8]) are avoided by using a realistic individual-based model . Note that numerical algorithms assume a rather smooth behaviour (i.e. a rather continuous evolution over time). As described in section 3.1, for self-organising emergent systems, the main focus is on analysing trends which are expected to evolve gradually.

The approach. The "equation-free" macroscopic analysis approach was proposed in [9, 10]. The observation is that most numerical techniques have no explicit need for the macroscopic equation; all they need is a routine that *evaluates* the equation for a given value of the macroscopic variables. Once the equation evaluations are replaced with a suitable *simulation*, all the numerical analysis algorithms can be readily applied.

To achieve this, the following procedure (illustrated in figure 2), called a *"macroscopic time-stepper"* [9], is performed. First the initial values (x_i) for all the macroscopic variables under study are supplied to the analysis algorithm. Then, given those values, the *initialisation operator* (`init`) initialises a number of simulations accordingly, i.e. such that a measurement of the macroscopic variables after initialisation equals the given initial values. Because there are multiple ways to initialise a simulation for the same

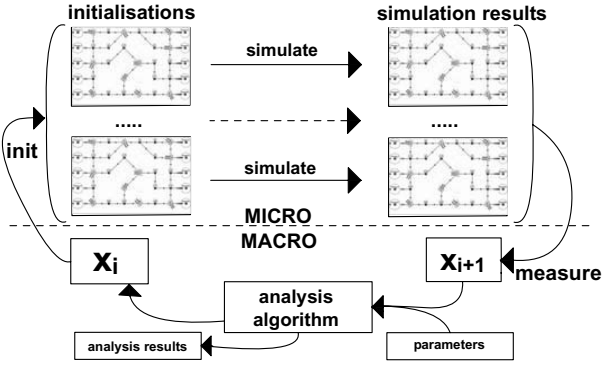


Fig. 2. Equation-Free Accelerated Simulation guided by the analysis algorithm

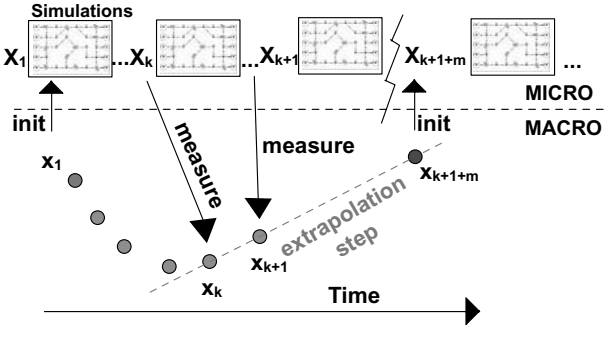


Fig. 3. Equation-Free Accelerated Simulation over Time

value of a macroscopic variable, these "degrees of freedom" need to be initialised randomly in multiple initialisations. Then the *simulations* (`simulate`) are executed for a predetermined duration using the individual-based simulation code. At the end, one *measures* (`measure`) the averages over all simulations of the values of the macroscopic variables (x_{i+1}), which are given to the analysis algorithm as a result. As such, the equation and its evaluation are replaced by the simulation code. Then the analysis algorithm processes the new results to obtain the next initial values. The algorithm itself decides on the configuration of the simulations such as the initial conditions and the duration. The `init-simulate-measure` cycle is repeated until the analysis algorithm reaches its goal.

There exists a whole arsenal of numerical algorithms to analyse system dynamics. One example is called the *projective integration algorithm* where the goal is to make a considerable acceleration of the simulations over time by minimising the number of needed simulation steps through extrapolation. Figure 3 illustrates the basic idea. First an initial value x_1 for the measured macroscopic variable is chosen by the analysis algorithm. Using the initial value x_1, a (set of) simulation(s) is initialised with micros-

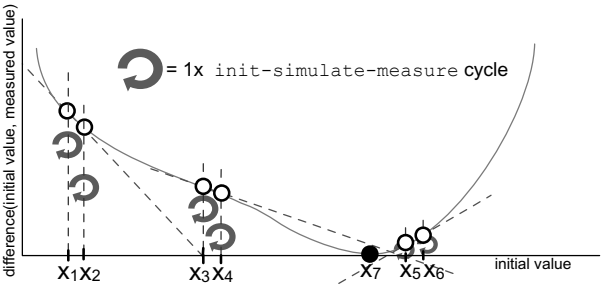


Fig. 4. Equation-Free Newton Analysis

copic value(s) X_1 (i.e. `init` operator on figure 3) and executed for a certain duration. At some points in time, one measures the new value for the macroscopic variable (i.e. `measure` operator on figure 3). The measure operator is repeated for a number of times such that enough successive values x_k are available for the projective integration algorithm to make the extrapolation step that skips m time steps of simulation. As such, a new value x_{k+1+m} is estimated with extrapolation, using a number of measured values (x_k and x_{k+1} in figure 3). Starting from the new value x_{k+1+m} the process is repeated by initialising new simulations with microscopic value(s) X_{k+1+m}. As such an acceleration over time is achieved.

Simulations can also be accelerated in other ways. Suppose the goal is to obtain the steady state behaviour, i.e. we look for values of the macroscopic variables that remain constant as time evolves. Instead of computing the time evolution from time step 0 until the system stays at the steady state long enough, one uses numerical procedures that determine steady states in a more direct and efficient way, e.g. Newton's algorithm [11]. Denote the macroscopic time-stepper starting from an initial macroscopic value x_i, performing a simulation for a fixed duration, and measuring the new macroscopic value by $\Phi(x_i)$. Denote the measured value of the macroscopic variable after the simulation as $x_{i+1} = \Phi(x_i)$. The steady state x^* is then computed by solving the equation

$$\Phi(x^*) - x^* = 0 \tag{1}$$

numerically by Newton's algorithm. This iterative method is illustrated in figure 4. The x-axis contains the initial values x_i and the y-axis the differences $|x_i - \Phi(x_i)|$ between the initial and the measured values for one cycle of the macroscopic time-stepper. First one chooses two initial values x_1 and x_2 close to each other. For each of them a `init-simulate-measure` cycle is done to get the measured values. Through these measured values an estimation for the derivative of the plot in figure 4 is used to extrapolate to a new estimation x_3 for the steady state. Then the cycle is repeated (x_3 - x_6) until the conditions and thresholds of the Newton algorithm decide to have reached a steady state x_7.

In contrast to mere observation of simulation results, only a limited set of rather short simulations are necessary to generate consecutive approximations for the steady state and a scientific algorithm objectively decides on the accuracy of the result. Note that the due to the equation-free approach equation (1) itself is not required. Newton's

algorithm only needs the evaluation of the equation at certain points in time and this is replaced by a suitable simulation measurement. Following the same equation-free principle, e.g. in the presence of parameters, more general tasks, such as parameter optimisation or control can be performed as well [9]. As such, a more focussed and accelerated simulation-based analysis approach, guided by the analysis algorithm's goal, is used to obtain more reliable and valuable results than mere observation of simulation results. As such the proposed method constitutes a bridge between classical numerical analysis and microscopic (e.g. agent-based) simulation.

Road Map. There are a number of steps needed for the equation-free approach to work. An overview of is given with ant foraging based on pheromones as an example:

1. *Identification of macroscopic properties*. The goal of the analysis approach is to systematically acquire results that give macroscopic guarantees. A systematic approach often implies that one divides the problem into manageable subproblems. The macroscopic behaviour of a self-organising emergent system typically consists of a number of *macroscopic properties* that have to be maintained. For the ant foraging example, macroscopic properties such as the amount of food gathered, the directed movement of the ants, and the shape of the pheromone path are important. In a first step, each of the macroscopic properties are considered separately in the analysis approach and one has to identify which properties are important for the considered case study (examples for the AGV case are given in section 3.4).
2. *Identification of macroscopic variables*. In the context of self-organising emergent systems, the challenge is to find variables that are measures for the macroscopic properties under study. In other words, a *quantification of the macroscopic properties* in terms of measurable variables is needed. For example, using entropy to measure the concentration of ants on pheromone paths or to measure how focussed ants choose a direction is a possibility [12].
3. *Related macroscopic variables*. Are there other variables for macroscopic properties that influence the property under study? If so, then these variables have to be incorporated into the analysis process. Otherwise, the evolution of the system is not correctly and completely represented and analysed. An underlying assumption of the equation-free analysis approach is that a set of measurable variables are found that offer an adequate description of the macroscopic system dynamics. For example, only entropy of the concentration of ants is not enough. This variable omits the evolution of the pheromones which also influence the ant-movement. Therefore, variables are needed that capture the macroscopic evolution of the pheromones.
4. *Microscopic variables*. For each macroscopic variable, the corresponding variables of the simulation at the level of the individual entities in the system, that influence the macroscopic variable need to be identified. With ant foraging, this will include variables such as the exact positions of the ants, the strengths and positions of pheromones, and if an ant holds food or not.
5. *Measurement operator*. Define a measurement operator that measures the macroscopic variables from the microscopic variables in the simulation.
6. *Initialisation operator*. An operator needs to be defined that allows to initialise the microscopic variables of the simulation or multiple simulations to reflect the given

values for the macroscopic variables. When there are degrees of freedom in the initialisation, these are initialised randomly and multiple simulations are considered to average this randomness. For example, re-initialising the ant system requires positioning each ant but given the concentration of ants, the exact positions allow some degrees of freedom.

7. *Micro-macro scale separation*. For the equation-free approach to work and to be efficient, it should be checked that the microscopic variables evolve on a much faster timescale than the macroscopic variables that determine the macroscopic evolution. Thus, changes in the state of the individual entities in the system (e.g. movement of ants) need to be fast compared to the evolution of the overall system behaviour (e.g. changes in pheromone path shape or ant concentration). If this is not the case, then any error introduced by the extrapolation and/or initialisation procedure could significantly influence the results and hence create errors.
8. Define the different *steady scenarios* to analyse. A macroscopic guarantee that holds in all possible conditions in which a system can be executing is difficult, if not impossible to give. A *steady scenario* is defined as a setting for the system in which certain assumptions are made about the operational conditions (i.e. initial conditions, possible changes, and the frequency of change). This step of the road map identifies the system parameters that need to be modified in order to cover the range of possible operational conditions for the system. For example, one can consider steady scenarios where the system has a high utilisation load, a low utilisation load, or a scenario where there is a frequent oscillation between high and low utilisation loads. In the ant foraging example, parameters such as the number of ants involved, the evaporation rate of the pheromones, and the amount of food present can be modified. As a consequence, macroscopic guarantees are always given with respect to a specific steady scenario and for a specific macroscopic property (see step 1 of road map). A complete analysis result of the macroscopic behaviour then consists of multiple macroscopic guarantees.
9. *Analysis algorithm*. In the end, an analysis algorithm is to be chosen. Depending on the kind of data and the goals, e.g. finding the steady state behaviour, optimising the value of a system parameter (e.g. which evaporation rate is optimal for a certain scenario), controlling the operational modus, one selects a suitable numerical analysis algorithm (e.g. projective integration, Newton's method).

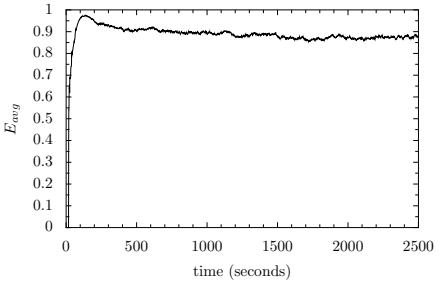
With self-organising emergent systems, an open issue is understanding how the macroscopic behaviour is accomplished by the individual entities. There is a gap between the result seen at the macroscopic level and the rules at the microscopic level that cause that result. Some of the steps in the road map are far from trivial, because some knowledge on how to bridge the micro-macro gap is required. Especially defining the measurement and initialisation operators is challenging. This requires to know how to represent a macroscopic property as a variable, how to measure it from the microscopic level and how to initialise a microscopic simulation to reflect a macroscopic variable. However, a successful application of the "equation-free" analysis road map can result in new insights in how the macroscopic level is related to the microscopic level. And, as shown in section 3.4, the approach offers the ability to check if a certain set of macroscopic variables completely captures the evolution of the macroscopic property.

3.4 Macroscopic Variables in the AGV Case

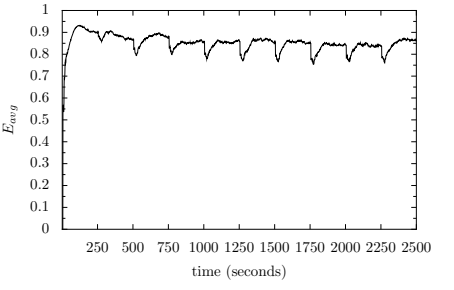
In each application domain, the important macroscopic properties will be different. The requirements that have to be achieved by the macroscopic behaviour determine the important macroscopic properties and the desired guarantees about them. In the AGV case a number of issues are important, some examples:

– *Distribution of AGVs* over the factory floor is a macroscopic property from which one could require that on average the AGVs are equally distributed over the factory floor, i.e. a maximum coverage. In a recent paper [13] we used a spatial entropy measure as a macroscopic variable that reflects the distribution of AGVs.

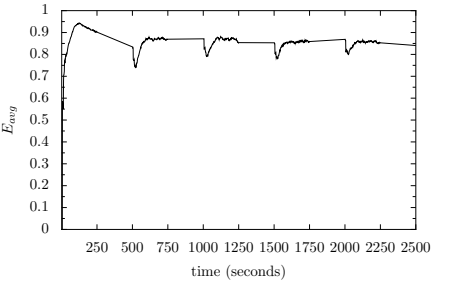
The results validated the described analysis approach. For example, in figure 5(a) the evolution of the average entropy during normal begin-to-end simulations is shown, i.e at each time step the average over 100 runs is taken. In figures 5(b) and 5(c) the same evolution is shown but obtained in two different ways. A first simulation process was executed in which simulations are done over 250 time steps, then the set of measured macroscopic variables (including entropy) is used to initialise a new set of simulations and again simulations of 250 time steps are done, and so on (figure 5(b)). The results show that a simulation based on re-initialisation with the chosen set of macroscopic variables reflects the correct average evolution compared to figure 5(a). Thus, the re-initialisation based simulation process allows



(a) A normal begin-to-end simulation



(b) A re-initialisation based simulation



(c) An extrapolated simulation

Fig. 5. Different simulation processes for the Distribution of AGVs reflected by the average spatial entropy (E_{avg}) evolution (high entropy is equal distribution, low entropy is unequal distribution)

to check if a chosen set of macroscopic variables is enough to reflect the evolution of the system, i.e. there are no other macroscopic variables needed to have a representative analysis (see step 3 of Road Map).

A second simulation process was executed in which the projective integration algorithm was used averaged over 100 simulation runs with a simulation duration and extrapolation step of 250 time steps. The results in figure 5(c) show that one can analyse the evolution of the average entropy with half the simulation time needed than normal begin-to-end simulation. As shown on figure 5, the distribution evolved to a steady state behaviour, i.e. a stable and almost equal distribution of the AGVs over the factory floor is achieved. More details can be found in [13]. Applying Newton's algorithm (see section 3.3) allows to confirm that this is a steady state behaviour scientifically (i.e. more accurate than mere observation) with only a few simulations.

– *Throughput* is the important characteristic in the AGV case. Throughput is defined as the number of packets transported in a certain time span (e.g. one hour). The challenge here is to identify macroscopic variables that allow to represent and measure the throughput evolution. Because throughput is a characteristic expressed over time and because the analysis approach expects the ability to measure the macroscopic variables at each time step, time-independent macroscopic variables are needed from which one can calculate the throughput as a post-analysis step.

  A possible set of macroscopic variables is the following:

  - The number of packets in transport (NBP_T), i.e. the number of AGVs currently holding a packet.
  - The number of packets in the queues at the pick-up locations (NBP_Q). Each pick-up location has a queue in which packets arrive. NBP_Q is the sum of all queue lengths.

  Assume that packets arrive in the pick-up location queues at a fixed arrival rate $AR_{\Delta t}$, i.e. the number of packets arriving in a time span of Δt. Then the throughput can be calculated as follows. Define the queue growth rate as the increase or decrease of total queue length in a time span of Δt, i.e. $QR_{\Delta t} = NBP_{Q,t} - NBP_{Q,t-1}$ with $NBP_{Q,t}$ the total queue length at time t. If the number of packets in transport is rather constant (i.e. $NBP_T = cte$) then throughput is defined as $T_{\Delta t} = AR_{\Delta t} - QR_{\Delta t}$. Another possible approach to calculate the throughput is to count the number of packets that are delivered at drop-off locations in a time span Δt. However, the number of drop-offs is a time dependent variable and re-initialising based on this variable at one moment in time is not possible and useful. The calculation given above only uses time independent variables, i.e. real state of the system.

It is clear that the biggest challenge in applying the analysis approach is to find suitable macroscopic variables that reflect the evolution of macroscopic properties for which one needs certain guarantees. Based on macroscopic variables a re-initialisation of a simulation at one moment in time is required. So, one seeks to get time independent variables. The intention of this paper is to give an idea of how the approach can be applied. Further analysis results that actually give guarantees are for a future publication.

4 Engineering Based on Analysis Results

The biggest problem with engineering self-organising emergent systems is the lack of a systematic approach to build a solution that meets the requirements. Despite some efforts (e.g. [14, 15]) to find a systematic engineering approach, finding an approach that starts from the macroscopic requirements and systematically constructs a system by deriving the behaviours of the individual agents in the system from the macroscopic requirements seems infeasible. Therefore, a combination of creatively building a solution and analysing it based on scientifically founded experimental methods is promising. Traditional engineering design methods tend to be based on a bottom-up approach in which known components are assembled into subsystems from which the system is constructed and then tested for the required properties. The design is modified in an iterative manner until the system meets the requirements. As discussed in [4, 5], a formal design method often can not do the job and the authors argue that such a method does not exist. One needs an experimental scientifically founded method.

We propose an integration of the systematic analysis approach, which we described earlier, into the engineering process. First of all, as a creative activity, one builds a first prototype of the system based on experience and combining existing mechanisms (e.g. [16]) and guidelines (e.g. [15]) to achieve a self-organising emergent system. Then one systematically analyses the system with respect to the wanted macroscopic requirements using the analysis approach described above. Feedback from that analysis is then used in a next engineering cycle to adjust and tune the solution in order to systematically evolve towards a final solution that meets all the requirements. Of course, the feedback obtained through the analysis of self-organising emergent systems can also result in more experience and guidelines to use in future engineering processes.

Using the above analysis approach extensively in an engineering process gives a number of possibilities. Some examples are:

- Bifurcation Analysis based on Parameters: As explained in section 3.3, the analysis approach allows numerical analysis algorithms to directly steer the simulation process in order to obtain simulation results on the fly and as efficiently as possible. One such analysis algorithm is a so called bifurcation analysis. Based on a certain parameter and for a given macroscopic variable the algorithm analyses what the influence of that parameter is on the macroscopic behaviour, i.e. on the macroscopic variable. For example, in the AGV case the number of AGVs used could be an important parameter. The results can then indicate how many AGVs are ideal for a given system to meet its requirements as best as possible and what happens when the number of AGVs goes beyond the ideal range of values. Also, in the throughput example (see section 3.4) the arrival rate $AR_{\Delta t}$ is an interesting parameter in order to know for which utilisation load a certain solution performs within acceptable boundaries. Such results are useful feedback to systematically tune and re-engineer a solution for the AGV case study.
- Comparison and Evaluation of existing Decentralised Mechanisms: In today's self-organising emergent systems a number of decentralised mechanisms are used, of which a lot are inspired by nature [16] (e.g. pheromones, gradient fields, etc.). Evaluating each of the decentralised mechanisms with respect to for example non-

functional and other characteristics (scalability, flexibility, reaction-speed to changes, communication bandwidth used, etc.) allows a more precise comparison of those mechanisms. Also, each mechanism has a number of parameters one has to tune (e.g. the evaporation rate for digital pheromones). An analysis can be done of the influence of the parameters and in which context which parameter range is most suitable. Further, when engineering a self-organising emergent system, this information will guide the choice of appropriate mechanisms for the problem at hand and thus contribute to a more systematic engineering of such systems.

Exploiting such possibilities integrates the analysis approach into the engineering process such that a constant feedback loop between analysis and engineering shapes a self-organising emergent solution. In the end, a systematic simulation-based engineering process where scientific analysis and feedback are essential can be achieved.

5 Conclusion and Future Work

It is clear that the way to systematically build a self-organising emergent system remains an open issue. However, before even considering a self-organising emergent system in an industrial context, a systematic analysis approach is needed that gives guarantees that the required coherent macroscopic behaviour is achieved and maintained. Because formal or analytic proofs are infeasible and because mere observation of simulation results is not reliable and scientific enough, the proposed approach combines realistic agent-based simulations with existing scientific numerical analysis algorithms for dynamical systems. Compared to mere observation of simulation results, more reliable and valuable results are returned because the analysis algorithm itself obtains the results on the fly by steering and accelerating the simulation process according to the algorithm's goal. In order to also achieve a more systematic engineering approach, it was proposed to exploit the analysis approach during the engineering process such that a constant feedback loop between scientific analysis and engineering shapes a self-organising emergent solution.

Future work includes applying the approach extensively to the AGV case and other application domains. Also, there are a number of issues that need to be resolved in order to make the approach easy applicable. For example, there are no clear guidelines on how to chose suitable macroscopic variables that reflect the macroscopic properties under study. Next to making the analysis approach more straightforward, the integration of that analysis approach into a systematic engineering approach can also be worked out and be made explicit in for example an engineering road map. Such a road map would then indicate where in a traditional software engineering methodology the scientific analysis is situated and where feedback from that analysis is to be used to guide the engineering process.

Acknowledgments. This work is supported by the K.U.Leuven research council as part of the Concerted Research Action on Agents for Coordination and Control - AgCo2 (T. De Wolf,T. Holvoet). G. Samaey is a Research Assistant of the Fund for Scientific Research – Flanders (FWO). This work is supported by grants FWO G.0130.03, IUAP/V/22 (Belgian Federal Science Policy Office) (G. Samaey).

References

1. Egemin, DistriNet: Emc^2: Egemin Modular Controls Concept,. (IWT-funded project with participants: Egemin (http://www.egemin.be) and DistriNet (research group of K.U.Leuven)) Started on 1 March 2004, ending on 28 February 2006.
2. De Wolf, T., Holvoet, T.: Emergence and Self-Organisation: a statement of similarities and differences. In: Proceedings of the Second International Workshop on Engineering Self-Organising Applications, New York, USA (2004) 96–110
3. Wegner, P.: Why Interaction is More Powerful than Algorithms. Communications of the ACM **40** (1997) 80–91
4. Edmonds, B., Bryson, J.J.: The Insufficiency of Formal Design Methods - the necessity of an experimental approach for understanding and control of complex MAS. In: Proceedings of the 3rd Internation Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'04), New York, ACM Press (2004) 938–945
5. Edmonds, B.: Using the Experimental Method to Produce Reliable Self-Organised Systems. In Brueckner, S., Serugendo, G.D.M., Karageorgos, A., Nagpal, R., eds.: Engineering Self Organising Sytems: Methodologies and Applications. Volume 3464 of Lecture Notes in Artificial Intelligence., Springer (2004) (to appear spring 2005).
6. Anthony, R.J.: Emergence: A Paradigm for Robust and Scalable Distributed Applications. In: Proceedings of IEEE International Conference on Autonomic Computing (ICAC'04), New York (2004) 132–139
7. Parunak, H.V.D., Savit, R., Riolo, R.L.: Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide. In: MABS. (1998) 10–25
8. Wilson, W.: Resolving Discrepancies between Deterministic Population Models and Individual-Based Simulations. American Naturalist **151** (1998) 116–134
9. Kevrekidis, I.G., Gear, C.W., Hummer, G.: Equation-free: The computer-assisted analysis of complex, multiscale systems. AIChE Journal **50** (2004) 1346 – 1355
10. Kevrekidis, I.G., Gear, C.W., Hyman, J.M., Kevrekidis, P.G., Runborg, O., Theodoropoulos, C.: Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. Communications in Mathematical Sciences **1** (2003) 715 – 762 (available online at http://www.intlpress.com/CMS/).
11. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C: The Art of Scientific Computing. 2nd edn. Cambridge University Press, Cambridge (1992)
12. Guerin, S., Kunkle, D.: Emergence of Constraint in Self-Organizing Systems. NDPLS: Nonlinear Dynamics, Psychology, and Life Sciences **8** (2004) 131
13. De Wolf, T., Samaey, G., Holvoet, T., Roose, D.: Decentralised Autonomic Computing: Analysing Self-Organising Emergent Behaviour using Advanced Numerical Methods. In: Proceedings of IEEE International Conference on Autonomic Computing (ICAC'05), Seattle, USA (2005) (accepted).
14. Poulton, G., Guo, Y., James, G., Valencia, P., Gerasimov, V., Li, J.: Directed Self-Assembly of 2-Dimensional Mesoblocks using Top-down/Bottom-up Design. In: Proceedings of the Second International Workshop on Engineering Self-Organising Applications (ESOA04), New York, USA (2004) 137–149
15. Parunak, H.V.D., Brueckner, S.A.: Engineering Swarming Systems. In Bergenti, F., Gleizes, M.P., Zambonelli, F., eds.: Methodologies and Software Engineering for Agent Systems. Volume 11 of Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer (2004)
16. Nagpal, R.: A Catalog of Biologically-inspired Primitives for Engineering Self-Organization. In Serugendo, G.D.M., Karageorgos, A., Rana, O.F., Zambonelli, F., eds.: Engineering Self-Organising Systems, Nature-Inspired Approaches to Software Engineering. Volume 2977 of Lecture Notes in Computer Science., Springer (2004) 53–62

On the Role of Simulations in Engineering Self-organising MAS: The Case of an Intrusion Detection System in TuCSoN

Luca Gardelli, Mirko Viroli, and Andrea Omicini

Alma Mater Studiorum–Università di Bologna,
via Venezia 52, 47023 Cesena, Italy
{luca.gardelli, mirko.viroli, andrea.omicini}@unibo.it

Abstract. The intrinsic complexity of self-organising MASs (multi-agent systems) suggests the use of formal methods at early stages of the design process in order to predict global system evolutions. In particular, we evaluate the use of simulations of high-level system models to analyse properties of a design, which can anticipate the detection of wrong design choices and the tuning of system parameters, so as to rapidly converge to given overall requirements and performance factors.

We take intrusion detection (ID) as a case, and devise an architecture inspired by principles from human immune systems. This is based on the TuCSoN infrastructure, which provides agents with an environment of artifacts—most notably coordination artifacts and agent coordination contexts. We then use stochastic π-calculus for specifying and running quantitative, large-scale simulations, which allow us to verify the basic applicability of our ID and obtain a preliminary set of its main working parameters.

1 Introduction

The trend in today information systems engineering is toward an increasing degree of complexity and openness, leading to rapidly changing requirements and highly dynamic environments. As a consequence, the cost of system management is becoming comparable to the cost of the system itself [1]. This phenomenon has led to the challenge of discovering new ways of engineering systems inspired by social and natural sciences. For instance the Autonomic Computing initiative tries to face complexity by taking inspiration from the self-regulating behaviour of the biological processes [1, 2]. Other visions, such as Amorphous Computing [3] and Spray Computers [4] have sprung sharing the same objective. Self-organisation is a promising approach to tackle these issues without explicit pressure or constraints from outside the system: self-organisation is usually the result of the interaction and coordination at a local level of a set of agents, each simpler in structure than the global task achieved [5].

In this paper we conduct a preliminary study on methodological aspects of the engineering of self-organising MASs. Because of the intrinsic complexity of these systems, and the difficulties in predicting their behaviour and properties,

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 153–166, 2006.

we find it crucial to exploit formal tools for simulating their dynamics at the early stages of design. In the case of self-organising MASs, in fact, this approach appears to be almost unavoidable in order to nurture evolving ideas and design choices, and to effectively tune parameters of the final system.

Among the various formal models to specify quantitative aspects of MASs—based on process algebras [6], Petri Nets [7], and automata [8]—we promote the use of the stochastic π-calculus process algebra [9]. While it allows for a great expressiveness in representing key aspects such as interactions and concurrent activities, π-calculus also features full compositionality and modularity properties, which are crucial to scale up with system complexity.

This language is basically unexplored in the context of self-organising MASs: on the one hand, its simulation tools are relatively recent (see e.g. [10]), on the other, it was primarily inspired by the need of modelling biological systems [11]. However, we show that stochastic π-calculus can be fruitfully applied to the MAS paradigm as well: as far as stochastic aspects are concerned, the typical complexity of agent internal machinery can be suitably abstracted away, focussing instead on agent interactions and high-level activity changes. Then, tools like SpiM (Stochastic PI-calculus Machine [10]) can be effectively used to track the dynamics of global system properties in stochastic simulations, validating design directions, inspiring new solutions, and determining suitable system parameters.

In this paper, we apply these ideas to study an intrusion detection (ID) infrastructure for MASs, which detects malicious agents in an open context. The solution we consider is inspired by principles of human immune system [12], where agents resembling lymphocytes are dynamically created and updated with the goal of detecting malicious behaviours. The infrastructure we devise is based on the TuCSoN technology [13][1]. This allows us to structure a MAS not only in terms of agents, but also with *tuple centres* as *coordination artifacts* [14] and *agent coordination contexts* (ACC) as *boundary artifacts* [15]. Coordination artifacts are used to model resources in the environment on which (potentially) malicious agents act upon. ACCs specify and enact the access policies which each agent is subject to, and can be used to both *(i)* reify relevant information about the agent/artifacts interaction, which the lymphocyte agents can inspect to detect abnormal sequences of actions, and *(ii)* to abruptly deny malicious agents to access the MAS environment.

To evaluate the impact of different design choices and parameters of the ID infrastructure—such as inspection/detection rates, number of lymphocytes, upgrade policy for lymphocytes, and the like—we simulate the behaviour of different scenarios using SpiM specifications.

The rest of the article is structured as follows. In Section 2 we briefly highlight the main mechanisms and properties of intrusion detection and the human immune system, and apply them to a general architecture for a MAS based on TuCSoN. Section 3 motivates the use of π-calculus and its stochastic extension, providing an application example related to our ID domain using SpiM. In Section 4 we examine performance of several scenarios of the MASs infrastructure

[1] `http://tucson.sourceforge.net`

introduced. Finally, Section 5 concludes providing final remarks and listing main directions for future research.

2 Human Immune System and a MAS Architecture

In this section we first depict the main aspects of ID systems (IDSs), and then describe the structure and main functionalities of the human immune system. We are not concerned about accurately modelling or mimicking the human immune system, but we rather gather useful principles to engineer secure self-organising applications, which are then used to sketch a general architecture for MASs applying some of these concepts.

2.1 Security and IDSs in Information Systems

There are several mechanisms used to protect information systems, but usually only the basic ones are implemented: *(i) authentication*, the identity is proved by the knowledge of a secret (e.g. password) or a physical unique property (e.g. fingerprint, retina, voice); *(ii) authorisation*: user actions on the system are constrained by its role.

However, applications flaws typically cause these methods not to be sufficient alone [16]. For instance, in operating systems there is a need to use additional software such as firewalls, antivirus and many other specific tools that must be configured and kept updated to prevent unplanned attacks. Furthermore authorisation policies cannot account for all possible sequences of actions: a specific sequence might exhibit unexpected side-effects. In particular, it is in general too expensive and impractical (or even unfeasible) to intercept all emergent harmful paths at design-time.

As a consequence, automated tools are a very useful support for the detection of malicious behaviour. In this direction, many efforts have already been spent in developing IDSs. An IDS tries to detect abnormal behaviour and misuse of a target software system by observing it and deciding wether actions performed by a user/agent are symptomatic of an attack [17]. Efficiency of an IDS is evaluated by three parameters: *accuracy* (rate of false-alarms), *performance* (rate of audit processing), and *completeness* (rate of missed detection).

Usually IDSs are implemented as expert systems: they synthesise signatures of malicious behaviours from lists of sequences of actions. However, for generality, we do not focus here on specific techniques to handle signatures, for they mostly concern the IDS research community, and are typically very dependent on the application domain—they differ e.g. in the context of system calls [16, 12], TCP/IP connection addresses [18], memory accesses, and so on. Rather, we are concerned about the neat impact of such techniques, expressed in a stochastic manner, and how they can influence the design of a protection system.

2.2 Human Immune System Overview

The human immune system protects the body against foreign pathogens. We use the term *antigen* to refer to any foreign molecule that triggers an immune response

by the human body. The human immune system consists of many layers each exploiting several mechanisms to increase the degree of protection. A first kind of protection is provided by the physiological barriers, i.e. skin, temperature and pH, that are reactive and non-specific—i.e. not triggered by a specific class of antigens.

The human immune system also provides active mechanisms: the innate and the acquired immune system. The first is there since birth, and is composed of scavenger cells (e.g. *fagi*) wandering in the lymphatic system, which are able to detect and kill only a fixed subset of antigens. It is not adaptive, hence it is not able to protect the body from new kinds of antigens. This issue is instead the main concern of the acquired immune system: it improves during individual life, learning and memorising new antigens. The acquired immune system is composed of different types of cells: we consider only lymphocytes for they are responsible for the main form of immunity.

Lymphocytes are produced in the bone marrow and are sent to mature in the thymus. There, they are exposed to self-cells (body): if they bind the self-cells lymphocytes are eliminated. When the process of maturation ends, lymphocytes are released in the lymphatic systems. Lymphocyte surface is covered with different sets of receptors that are able to bind to different classes of antigens. This phenomenon, called dynamic coverage, lets the immune system cover part of the space of antigens (10^{16}) with a sensibly smaller set of lymphocytes (10^{10}). Receptors are randomly generated by a process of variation and selection. Lymphocytes have a short life (about two days), but if during this period they bind to several antigens they become "memory" and their life is extended. This strategy, in combination with receptor generation process, allows to discover new antigens and to apply a faster response if the antigen is met again.

2.3 A General Architecture for MAS Applications

In this section we describe a general architecture for MASs based on the TuCSoN coordination infrastructure [13], showing an approach to ensure security applying principles of the immune system—for space reasons we only sketch main design details.

We consider a system that provides agents with services encoded in terms of coordination artifacts, i.e. runtime abstractions encapsulating and providing a coordination service, to be exploited by agents in social contexts expressed by coordination rules and norms [14]. The basic model of coordination artifacts is characterized by *(i)* a usage interface, *(ii)* a set of operating instructions, and *(iii)* a coordination behaviour specification, which can be exploited by cognitive agents to rationally use a coordination artifact.

Accesses of agents to these resources is restricted by an authentication procedure. When an agent enters the system an authorisation policy limits its actions allowing the exploitation of a limited set of services and resources—e.g. those it has payed for. This is realised by the notion of Agent Coordination Context (ACC) [19, 15]. An ACC works as agent interface towards the environment: it is like a control room providing e.g. buttons and displays to an human, which are the only means by which he/she can interact with the environment. Thus, the

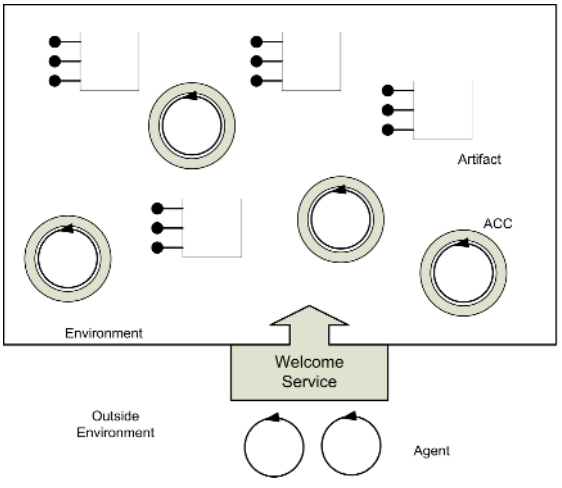


Fig. 1. A general architecture for a multi-agent system

ACC enables and rules the interaction between the agent and the environment [19], and it is then able to model security and organisation aspects in MASs [15]. In particular, the ACC is the right place to put authorisation policies, typically specified using a Role Based Access Control model (RBAC). The whole architecture is depicted in Figure 1.

Usually the two mechanisms of authentication and authorisation are considered to guarantee a sufficient degree of protection. However, as described in previous section, we instead promote the idea that a dynamic system is better protected by additional dynamic mechanisms. So we introduce other techniques inspired by the immune system and previous work on IDSs [16, 20].

As the human immune system has barriers, we consider authentication and authorisation procedures to be information systems barriers. We need an extra layer to cope with dynamic issues concerning harmful sequences of actions which cannot be statically identified as such. So we add a layer inspired by the acquired immune system of humans. Forrest et al. suggested a list of organising principles that should guide the design of a computer security system: distributability, multi-layering, no (totally) secure layer, diversity, disposability, autonomy, adaptability, and identity via behaviour—see [16] for more details on them.

We introduce a class of (distributed) agents, which we call agentLy, to model lymphocyte task: they should observe the actions performed by agents (autonomy) as reified by the ACC in charge of control them. When they detect a suspicious behaviour (identity via behaviour) they should dispatch an alert message to the authority that can trigger the proper response. In our case, this authority could invalidate the agent ACC, i.e. denying any further access to the system resources. Each agentLy is able to detect only a subset of the possible bad behaviours (diversity). If after a time interval it does not detect any attack then it can be replaced by another one (disposability). The security layer dynamically

covers the space of possible bad behaviours. The system should be able to learn and synthesise new signatures of abnormal behaviour (adaptability). As already stated we do not deal with synthesising these signature, but we encapsulate these skills in our agentLys and then abstract away from their details.

Now that we have sketched the system architecture and pointed out which properties could be useful, we are interested in predicting the behaviour of the system in large-scale scenarios, and the performance factors we should expect. We try to answer these questions in the following sections.

3 Simulations in π-Calculus

In this section we briefly introduce π-calculus [6] and its stochastic extension [9]. Then, we present an example of a simple program using the Stochastic Pi Machine (SpiM) [10].

3.1 The π-Calculus

The π-calculus is a formal model developed to reason about concurrency [6]: it is a language for describing and analysing systems consisting of agents (or processes) which interact with each other, and whose configuration of neighborhood is continually changing [21]. The basic entity is a *name*, which is used as an unstructured reference to a synchronous channel where messages can be sent and received. In its simpler version, a process is built from names according to the syntax:

$$P ::= 0 \mid \sum_{i \epsilon I} \pi_i.P_i \mid (P|Q) \mid !P \mid (\nu x)P \tag{1}$$

0 is the empty process. The summation $\sum_{i \epsilon I} \pi_i.P_i$ means that an agent might perform any prefix action π_i, and correspondingly continues as P_i behaviour: prefix forms π_i are of the kind $\bar{y}x$ (send the name x at channel y), $y(x)$ (wait for a name at channel y and rename it as x), and τ (perform a silent action). A composition $P|Q$ represents P and Q executed in interleaved concurrency. A replication $!P$ means that (infinitely) many copies of P can be executed concurrently (like $P|P|....$). Restriction $(\nu x)P$ creates the new name x and bounds its use in P.

The version of π-calculus that allows to send/receive a single name only to/from a channel is called monadic. The polyadic version of the π-calculus also allows more names in a single communication to be sent/received [21]. The semantics of π-calculus can be described by a transition system, where the transition relation $P \longrightarrow P'$—process P moving to P' by the occurrence of an inner interaction—is defined by operational rules [6].

3.2 On Stochastic Models

In general, each formal model whose semantics is given by a transition system can be extended to a stochastic version, resorting to the idea of Markov transition system [22]. There, each transition $P \xrightarrow{r} P'$ is labelled with a *rate* r, a nonnegative real value that scales how the transition probability between P and P' increases

with time. Stochastic models allow for quantitative simulations, for rates can be used to express aspects such as probability, speed, delays, and so on.

However, from an engineering perspective the choice of which language is used for describing processes is a crucial one, for the system to be simulated is to be effectively represented in the language.

Three basic options are available: *(i)* automata, like finite-state ones, where the system is described by state-changes and by supporting data-structures (such as stacks); *(ii)* nets, like Petri Net, where the system is described by a marking of tokens spread over a graph; and finally *(iii)* process algebras, like π-calculus, where the system is described by a composition of interacting entities. We find the third approach to be the best suited for describing quantitative aspects of complex MASs—such as self-organising ones. On the one hand, differently from automata, process algebras allow to express concurrent activities (agents in this case). On the other hand, differently from nets, process algebras allow for full compositionality: this property is particularly relevant, as it allows to express agents (and artifacts) with different roles separately, and then simply reuse such definitions to express the whole system model by composition (parallel composition, summation and replication).

3.3 Stochastic π-Calculus and π-Machine

Priami introduced a stochastic extension to π-calculus [9]. Each channel name is associated with an activity rate r: the delay of an interaction through that channel (representing the use of a resource [9]) is then a random variable with an exponential distribution defined by r. Exponential distributions are used because they enjoy the memoryless property, i.e. each transition is independent from the previous one [23]. Given a channel name x, the probability p_i of a transition $P \xrightarrow{r_i} P_i$ representing an interaction through x is the ratio between its rate r_i and the sum of rates of the n transitions through x enabled by P:

$$p_i = \frac{r_i}{\sum_{j=1..n} r_j}, \quad 1 \leq i \leq n \ . \tag{2}$$

We consider the SpiM implementation for the stochastic π-calculus interpreter [10]. As an example, we want to simulate a simple scenario where agents can enter and leave a system after being authenticated and authorised. The system parameters are *(i)* the number of agents within the system at $t = 0$, *(ii)* the "concentration" of malicious vs. legitimate agents, *(iii)* the rates at which legitimate agents enter and leave the system, and *(iv)* the rates at which malicious agents enter and leave the system.

We are simulating the system for 1000 time units. We want to keep the average number of agents constant so that the entering rate is equal to the leaving one. Code and simulation results are reported in Figure 2 and 3.

The initial part of the specification introduces channel names—we set e.g. the small rate 0.1 for agents entering and leaving the system (`delayEnter` and `delayLeave`), and a 70% ratio of legitimate (good) agents vs. malicious (bad) ones (`isGoodEA` and `isBadEA`). In the behavioural part, we instantiate 1000

```
1000.0 (* Simulation duration *)

(* Counters *)
new CountBad:1000000.0:<>               new CountGood:1000000.0:<>
new isBadEA:300000.0:<>                 new isGoodEA:700000.0:<>
new delayLeaveG:0.1:<>                  new delayLeaveB:0.1:<>
new delayEnterG:0.1:<>                  new delayEnterB:0.1:<>
new EA:<>        (* Agent*)
new BEA:<>       (* Malicious Agent*)      new GEA:<>    (* Legitimate Agent *)
new ActionLeaveG:<>                     new ActionLeaveB:<>
new ActionEnterG:<>                     new ActionEnterB:<>
new LB:<>        (* Signal: malicious agent must leave *)
new LG:<>        (* Signal: legitimate agent must leave *)
new InitEA:<int> (* Initialize agents *)
new Timer:<<>,<>>

(* Behaviour *)
( !GEA(); LG()
| !BEA(); LB()
    (* Manage the flow of agents (in/out the system) *)
    (* 4 timers:
        - legitimate agents entering and leaving
        - malicious agents entering and leaving  *)
| Timer<ActionEnterG, delayEnterG>
| !ActionEnterG(); (CountGood<> | GEA<> | Timer<ActionEnterG, delayEnterG>)
| Timer<ActionEnterB, delayEnterB>
| !ActionEnterB(); (CountBad<> | BEA<> | Timer<ActionEnterB, delayEnterB>)
| Timer<ActionLeaveG, delayLeaveG>
| !ActionLeaveG(); (CountGood() | LG<> | Timer<ActionLeaveG, delayLeaveG>)
| Timer<ActionLeaveB, delayLeaveB>
| !ActionLeaveB(); (CountBad() | LB<> | Timer<ActionLeaveB, delayLeaveB>)
    (* Initialize Lymphocytes, Good and Bad agents (t=0) *)
| !EA(); (isBadEA<>; (CountBad<> | BEA<> ) + isGoodEA<>; (CountGood<> | GEA<>))
| !isBadEA()
| !isGoodEA()
| !InitEA(n); if n>0 then ( EA<> | InitEA<n-1> )
| InitEA<1000>

(* Library *)
| !Timer(c,r); (r<>|r();c<> )
)
```

Fig. 2. Source code of the model of a simple system where malicious and legitimate agents can enter and leave

agents (`InitEA<1000>`) and then counted the number the evolution of legitimate and malicious agents (`CoundGood` and `CountBad`), which are those shown in the plot[2]. This example is used as basis for developing interesting dynamics in more complex cases, as developed in the following.

4 Simulating Self-organising Systems

In this section we discuss three scenarios, exemplifying an incremental design process for our IDS. We refer to the general architecture described in section 2.3 (figure 1). For space reasons the complete simulation code for these examples is not reported[3].

[2] The whole description of this code is avoided for brevity.

[3] The interested reader can download them from the site: `http://www.alice.unibo.it/download/spim/esoa.zip`

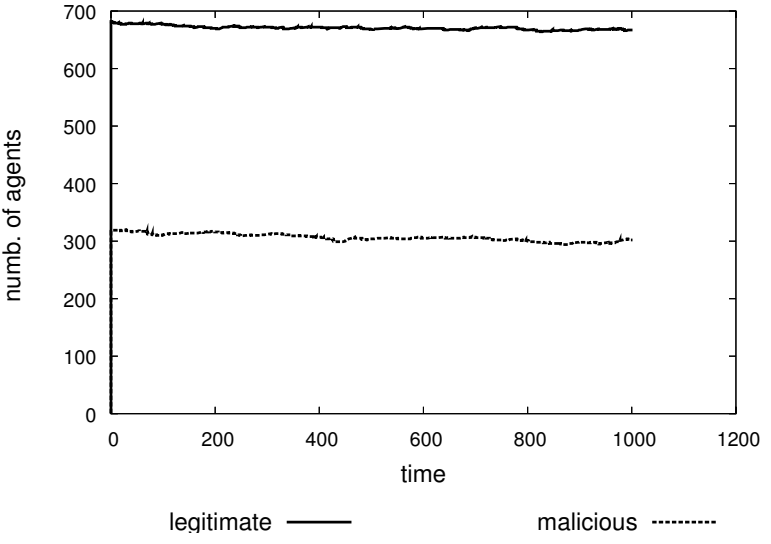


Fig. 3. Simulation of the system in Figure 2

4.1 Scenario 1

Starting from the previous example we add to the system lymphocyte agents (agentLy), which observe the behaviour of external agents. Its role, already described in Section 2.3, is to monitor ACC states to detect wrong behaviours. Each agentLy performs independently from the others, but they all share the same parameters. We add to the previous list the following parameters

- the number of agentLys (`10`)
- the rate at which an agentLy performs inspections (`delayInspection:0.5`)
- the probability that an agentLy, during inspection, detects an agent as malicious (10%, due to `matchP:100000.0` and `matchN:900000.0`).

The first and second parameter should be dynamically adjusted: for instance if the system is under attack it can raise its defenses. We consider them as a constant for the duration of the simulation. The third parameter instead should be regarded as a contract between the agentLy and the system. The system can replace agentLys that do not comply with the contract. This can be e.g. realised by making agentLys have their own ACC, and the infrastructure checking their detection rate.

The chart in Figure 4 shows the results of the simulation. With the chosen values for the parameters, we observe that the system is able to eliminate the activity of malicious agents within around 400 time units.

4.2 Scenario 2

In this scenario we introduce the idea that agentLys feature a limited lifetime. Furthermore we want to model two classes of agentLys which have different abilities to detect malicious activity. We call the ones which perform better

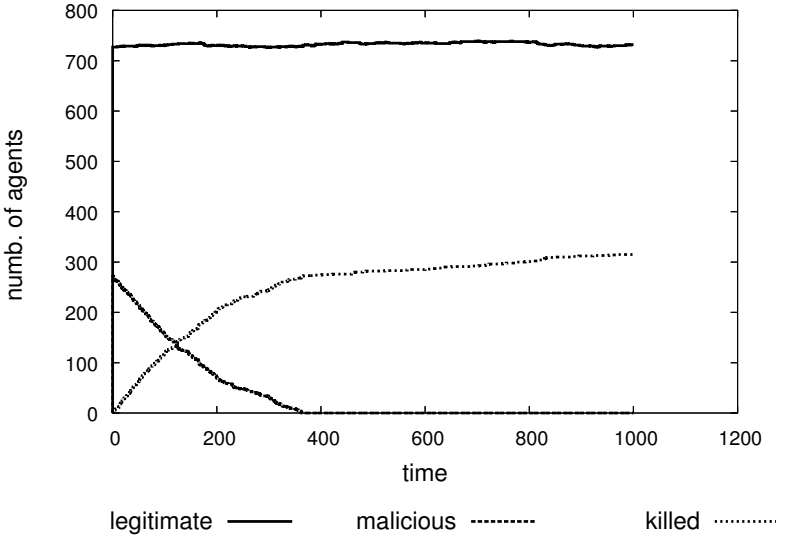


Fig. 4. A simulation of a simple system where malicious and legitimate agents can enter and leave. AgentLys limit the activity of malicious agents.

agentLyA, while the other ones agentLyB. Because an agentLyA performs better it will be rewarded with a longer lifetime, modelling the memory effect. So we add the following parameters:

- the probability that an agentLy belongs to class A rather than B (20% due to `isBadL:800000.0` and `isGoodL:200000.0`)
- the lifetime of the two classes of agentLys (`GL<200>` and `BL<100>`)
- the probability that an agentLyA, during inspection, detects an agent as malicious (30% due to `matchGP:300000.0` and `matchGN:700000.0`)
- the probability that an agentLyB, during inspection, detects an agent as malicious (10% due to `matchBP:100000.0` and `matchBN:900000.0`).

Figure 4 shows the results of the simulation. With the chosen values for the parameters the system is able to limit the activity of malicious agents. As expected, this system performs better than the previous, for now malicious agents are eliminated in less than 300 time units. Furthermore, by increasing the probability that an agentLy belongs to class A, we can expect the performance of detection to increase accordingly.

4.3 Scenario 3

In this scenario we introduce the hypothesis that the types of malicious behaviours are not uniformly distributed in the space of possible behaviours. So an agentLy that has a poor performance must learn from those who perform better. To account for this phenomenon we modelled the possibility for the system to clone an agentLyA to replace an agentLyB. This can be realised in our MAS by imposing in an agentLy's ACC contract that ineffective agentLys should accept

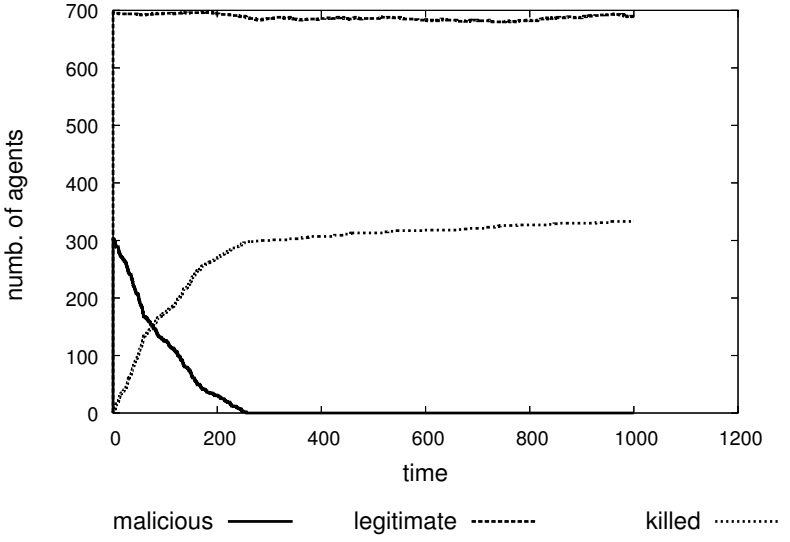


Fig. 5. A simulation of a simple system where malicious and legitimate agents can enter and leave. AgentLys limit the activity of malicious agents. An agentLyA perform better and has a longer lifetime than an agentLyB.

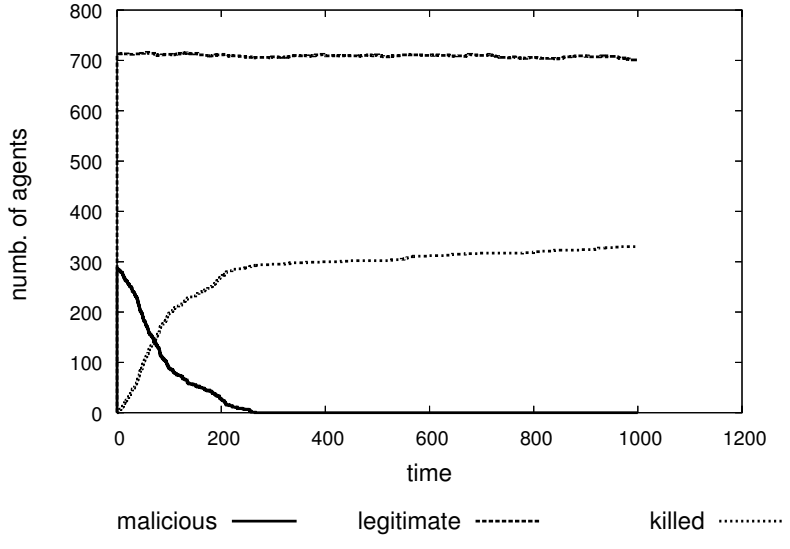


Fig. 6. A simulation of a simple system where malicious and legitimate agents can enter and leave. An agentLyA perform better and has a longer lifetime than an agentLyB. As agentLyA number increases it becomes more probable that a new agentLy will belong to class A.

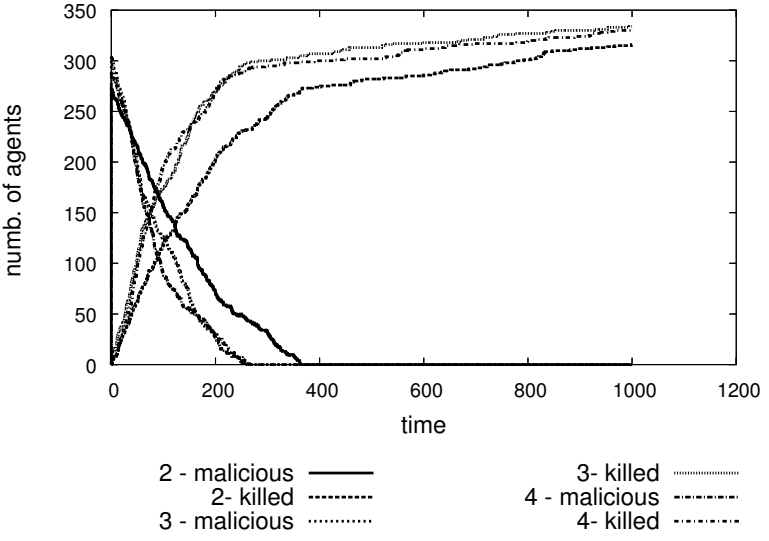


Fig. 7. A comparison in detection performance between the previous simulations

from the infrastructure upgrades on their behaviours—which the infrastructure takes from effective ones.

Figure 6 shows the results of the simulation. With the chosen values for the parameters the system is able to further limit the activity of malicious agents—within still remains similar to the previous case.

4.4 Results Comparison

In Figure 7, we merged the previous charts to evaluate the increase of detection performance. As we would expect there is an evident increase of performance between the first and second scenario, while a minor increase occurs between the second and the third. In general, while the third solution appears more promising, the actual results are strictly dependent from the value of parameters, hence more experiments are needed in order to evaluate whether the possible gains worth the considerable infrastructural support required to simulate lymphocyte learning. In fact, if the two classes of agents have a similar detection rate, the third solution might add only overhead to the system.

5 Conclusion

In this paper we have started putting together the elements of a framework for engineering self-organising applications: featuring MASs composed by agents and artifacts [14, 15], and simulations in a stochastic process algebra settings to tune system parameters at design-time. We used an intrusion detection system for TuCSoN as a mere explanatory case—as more comprehensive ones have already been explored (see e.g.[24, 12]).

The system depicted being based on the TuCSoN coordination infrastructure, it features the remarkable notion of ACCs, which enable to control agent actions, reify information on action sequences (to be read by the infrastructure and/or other agents), prevent agent actions from a given point in time. For the architecture and general principles we took inspiration from the human immune system. For the methodology, we relied to formal simulation and modelling via stochastic π-calculus, which—even though is a quite new language in the context of the MAS community—showed its effectiveness as a design tool.

Whereas the experiments we realised are still preliminary, we believe they generally emphasise the ability of the proposed approach to help the MAS developer to anticipate design decisions and strategies at the early stages of design—before actually developing prototypes and testing them.

Our plan for future works includes exploiting our approach to devise an actual implementation of intrusion detection systems on top of TuCSoN-based MASs. Other than evaluating the actual strategy to implement (as well as its distinctive parameters), we plan to explore the implications of extending such an approach to a network of nodes. In this paper we have only been concerned with self-organisation mechanisms: in future works we will explore the dynamics that causes new phenomena and behaviours to emerge. For example the uniqueness of the human immune system provide the human species with a greater probability to survive to a specific antigen. This is an emergent property which could be very important for distributed system. Integrating agent cognition and stochastic simulation models is a longer-term research direction as well.

References

1. Horn, P.: Autonomic computing: IBM's perspective on the state of information technology (2001)
2. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1) (2003) 41–50
3. Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Thomas F. Knight, J., Nagpal, R., Rauch, E., Sussman, G.J., Weiss, R.: Amorphous computing. Communications of the ACM **43**(5) (2000) 74–82
4. Zambonelli, F., Gleizes, M.P., Mamei, M., Tolksdorf, R.: Spray computers: Frontiers of self-organization for pervasive computing. In: 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'04), Washington, DC, USA, IEEE Computer Society (2004) 403–408
5. Heylighen, F.: The science of self-organization and adaptivity. In: Knowledge Management, Organizational Intelligence and Learning, and Complexity. The Encyclopedia of Life Support Systems. EOLSS Publishers (2003)
6. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, part I/II. Information and Computation **100**(1) (1992)
7. Petri, C.A.: Kommunikation mit Automaten. PhD thesis, Institut für Instrumentelle Mathematik, University of Bonn, Bonn, Germany (1962)
8. Bryans, J., Bowman, H., Derrick, J.: Model checking stochastic automata. ACM Trans. Comput. Logic **4**(4) (2003) 452–492
9. Priami, C.: Stochastic pi-calculus. Computer Journal **38**(7) (1995) 578–589

10. Phillips, A.: The stochastic Pi machine (SPiM) (2005) http://www.doc.ic.ac.uk/~anp/spim/.
11. Phillips, A., Cardelli, L.: Simulating biological systems in the stochastic pi-calculus (2004)
12. Forrest, S., Hofmeyr, S.A., Somayaji, A.: Computer immunology. Communications of the ACM **40**(10) (1997) 88–96
13. Omicini, A., Zambonelli, F.: Coordination for internet application development. Autonomous Agents and Multi-Agent Systems **2**(3) (1999) 251–269
14. Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C., Tummolini, L.: Coordination artifacts: Environment-based coordination for intelligent agents. In Jennings, N.R., Sierra, C., Sonenberg, L., Tambe, M., eds.: 3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004). Volume 1, New York, USA, ACM (2004) 286–293
15. Omicini, A., Ricci, A., Viroli, M.: RBAC for organisation and security in an agent coordination infrastructure. Electronic Notes in Theoretical Computer Science **128**(5) (2005) 65–85 2nd International Workshop on Security Issues in Coordination Models, Languages and Systems (SecCo'04), 30 August 2004. Proceedings.
16. Somayaji, A., Hofmeyr, S., Forrest, S.: Principles of a computer immune system. In: 1997 Workshop on New Security Paradigms (NSPW '97), New York, NY, USA, ACM Press (1997) 75–82
17. Debar, H., Marc, D., Andreas, W.: Towards a taxonomy of intrusion-detection systems. Computer Networks: The International Journal of Computer and Telecommunications Networking **31**(9) (1999) 805–822
18. Forrest, S., Hofmeyr, S.A., Anil, S., A., L.T.: A sense of self for Unix processes. In: 1996 IEEE Symposium on Security and Privacy, Los Alamitos, CA, USA, IEEE Computer Society (1996) 120–128
19. Omicini, A.: Towards a notion of agent coordination context. In Marinescu, D.C., Lee, C., eds.: Process Coordination and Ubiquitous Computing. CRC Press (2002) 187–200
20. Hofmeyr, S.A., Forrest, S.: Immunity by design: an artificial immune system. In: Genetic and Evolutionary Computation Conference (GECCO'99). (1999) 1289–1296
21. Milner, R.: The polyadic π-calculus: a tutorial. In Hamer, F., W., B., Schwichtenberg, H., eds.: International Summer School on Logic Algebra of Specification, Springer (1993)
22. Brinksma, E., Hermanns, H.: Process algebra and Markov chains. In: Lectures on formal methods and performance analysis: 1st EEF/Euro Summer School on Trends in Computer Science. Springer (2002) 183–231
23. Gillespie, D.: Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry **81**(25) (1977)
24. Hassas, S., Foukia, N.: Towards self-organizing computer networks: A complex system perspective. In Di Marzo Serugendo, G., Karageorgos, A., Rana, O.F., Zambonelli, F., eds.: 1st International Workshop on Engineering Self-Organising Applications (ESOA 2003). (2003) 77–83

Mesoscopic Modeling of Emergent Behavior - A Self-organizing Deliberative Minority Game

Wolfgang Renz and Jan Sudeikat

Multimedia Systems Laboratory,
Hamburg University of Applied Sciences,
Berliner Tor 7, 20099 Hamburg, Germany
Tel. +49-40-42875-8304
{wr, sudeikat}@informatik.haw-hamburg.de

Abstract. Recent research discussed several approaches to understand the relation between microscopic agent behavior and macroscopic multi–agent system (MAS) behavior. A structured methodology to derive these models will have impact on MAS design, evaluation and debugging. Current results have established the description of macroscopic behavior, including cooperation, by Rate Equations derived from markovian agent–states transitions. Emergent phenomena elude these descriptions. In this paper, we argue that mesoscopic modeling is needed to provide appropriate descriptions of emergent system behavior. The mesoscopic agent states reflect the emergent behavior and allow for a deliberative implementation of the rules and conditions which cause the MAS to self–organize as wanted. In a case study, we construct such a mesoscopic model for the socio-economic inspired Minority Game. The mesoscopic description leads us to a deliberative implementation, which exhibits equivalent self–organizing behavior, confirming our results.

1 Introduction

Multi–agent system (MAS) research has developed a number of reliable agent platforms, drawing attention to the methodical development of applications (as described in [1],[2]). Two paradigms for MAS design dominate current research efforts. A number of methodologies for the development of agent-based applications (surveyed in [3],[4],[5]) were proposed. The majority of these focus on *deliberative* agent architectures and therefore concentrate on modeling and analysis of individual agent behaviors. In opposition to these, *reactive* agents show adaptive and self–organized behavior, which leads to powerful applications [6],[7],[8],[9]. Recently the equivalence of these two approaches has been questioned by the observation of Universality [10].

Recent research has developed means to infer mathematical descriptions of the *macroscopic* system behavior from the *microscopic* agent behaviors. Being able to quantify the expected macroscopic behavior of MAS will have impact on system evaluation and verification. To enable the purposeful *engineering* of

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 167–181, 2006.

MAS, it is not only necessary to ensure the correct functionalities of the participating agents [11],[12], but also to verify that the system as a whole behaves as intended [13]. Current engineering approaches are based on systematic analysis of the macroscopic system behavior using simulations [14], to enable the identification of mechanisms, which ensure behavior and performance [15].

For MAS composed of homogeneous reactive agents, e. g. foraging robots [16], the microscopic agent behaviors directly reflect he overall system behavior. If agents have a fixed number of executable behaviors or states (e.g. searching and homing), the system can be described by the fraction of agents executing either of them. Therefore the rates of state transitions need to be defined. According to these simplifications, Rate Equations have been established to describe time dependent changes in the macroscopic behavior of MAS, which are composed of reactive and/or adaptive agents [16],[17],[18],[19]. These equations result from Master Equations describing the underlying Markov processes of these systems. These mathematical techniques have been successfully applied to study numerous examples, including aggregation [20], task allocation [17], foraging [21] and coalition formation [18].

Enhanced capabilities of the individual agents complicate these stochastic descriptions. In [16], modeling of agent populations using memory-based adaption [22] has been introduced. If decisions of agent are based on the past m states, the system can be represented as a generalized Markov process of order m. It has been shown that averaging over histories can be used to handle these kinds of systems [22].

Other complications to these kinds of stochastic descriptions arise from dynamically developed system behaviors. *Cooperation*, *self–organization* and *emergent phenomena* form during system development. The distinction between the later two is often complicated. Here, we adopt working definitions from [23]. Therefore, we understand self–organization as a dynamical and adaptive process, where structures arise and maintain, without external control. Systems exhibit emergence when microscopic interactions lead to novel artifacts (emergents), which dynamically arise at the macroscopic level.

In this paper we introduce *mesoscopic* modeling to derive mathematical descriptions of systems exhibiting emergent behavior. A mesoscopic modeling level introduces *hidden* agent states. We name these hidden, because they are not directly observable in the macroscopic behavior of the individual agent at each time step. This refinement to the above summarized techniques allow the description of emergent phenomena. In addition, we propose a method for construction of macroscopic equivalent reliable self–organized systems. The derived system uses a deliberative architecture for reactive planing and exhibits equivalent global behavior by self–organization. The construction is exemplified for a stochastic version of the well studied Minority Game (MG) [24], exhibiting working behavior which maximizes the macroscopic system performance to a great extend.

The next section gives a brief overview of recent approaches to model the macroscopic behavior of MAS, followed by an introduction of our framework to derive mesoscopic models of system behaviors. Section three introduces a case

study, the socio-economic inspired MG. We present a stochastic implementation of this game, its mesocopic description and a deliberative implementation which resembles the derived mesoscopic variables. Finally, we conclude and give prospects for further work.

2 Microscopic Models, Macroscopic Descriptions and Mesoscopic Modeling

In this section, the underlying conceptual ideas are introduced and briefly discussed. We summarize established techniques to infer macroscopic models and relate them to our mesoscopic approach. A more detailed description of mesoscopic modeling is given in a case study with Minority Games in the next section.

2.1 Macroscopic Description of Reactive Multi-agent Systems

Recent work [16],[17],[18],[19] showed how Rate Equations can be inferred from individual agent behavior. Reactive Agents can be modeled as Finite State Automaton (FSA). Agents can pursue different behaviors during their life-cycles, related to certain states in an imaginary FSA. In order to describe the overall system behavior, states are identified which have impact on the observables in the overall system. When it is possible to describe the probabilities of transitions between these states, Rate Equations for the observables can be written down directly. The global behavior of a MAS is characterized by the fractions of agents executing in certain microscopic states inside the FSA. To put it in other words, the FSA of the individual agents lead to an aggregate automaton, in which the states represent the numbers of agents executing that action [16],[25].

The Rate Equations for the occupation numbers of the microscopic model, namely the individual reactive agent, can be derived postulating an underlying time–continuous Markov process. These Rate Equations are impaired by three sources of complication. First, *correlations* between agent states vanishes. These can be spatial/time correlations or subtle structures of correlations. Secondly, the implicit mean–field approach eliminates *fluctuations* or variances of the number of agents in each state. Finally, *discrete time effects* are neglected. Fast and dynamic interchange between states can produce strong changes in occupation numbers on short time scales, leading to discrete time oscillatory behaviors relevant for the macroscopic description.

These complications prevent the direct description of emergent structures [23] by simple Rate Equations in many cases. E.g. in traffic systems wave phenomena emerge as soon as the stationary flow gets unstable. These can not be described by the average occupation numbers of cars, but spatio–temporal variations need to be included in the description. Similarly, freely movable swarms can be described by radial distribution functions for the agent distance only, whereas angular correlations have to be introduced when the agents obstruct each other and hinder free motion leading to spatial structures. Below we will examine the MG where persistent short time behaviors of agents are correlated with their

gameplay decisions. Quantities characterising the emergent structures have to be introduced in the system description. Thus, additional possibly hidden variables need to be included. Models keeping the relevant fluctuations but otherwise abstracting from microscopic details we call *mesoscopic* models in agreement with the statistical modeling community [26].

2.2 Mesoscopic Modeling

The dynamic evolution of MAS can exhibit emergent behavior not directly visible in its microscopic construction. As argued above, such behavior cannot be derived from macroscopic Rate Equation obtained from the Master Equation by a simple mean–field assumption [18]. But this is not our primary interest in the present paper.

Instead, we focus on the construction of less microscopic models with equivalent macroscopic behavior. Namely, software is *engineered* in controlled ways to *guaranty* certain macroscopic behaviors. To allow engineering, it is necessary to build–in the expected behavior into MAS rather than allowing it to emerge in a less policed way. This *build–in* requires a representation of the emergent behaviors inside the individual agents. In accordance to statistical physics modeling, we found that mesoscopic descriptions can fill this gap. It remains an open question which of the two approaches is more promising at the end.

For microscopic models exhibiting emergent behavior, we suggest a method, how to construct a mesoscopic model exhibiting equivalent macroscopic behavior. Relevant variables which are able to describe the emergent structures can be introduced into the original model. Then an averaging process can be assumed and mesoscopic states can be identified.

Such a model is still not a macroscopic model since it contains short–time fluctuations which in a macroscopic description have to be averaged out by self–organizing agent behavior. These interrelationships are described in figure 1.

The construction of the mesoscopic model provides the appropriate description level to allow the programmer to implement in a straight–forward way the rules and conditions for the deliberative agents to exhibit the emergent behavior by self–organization.

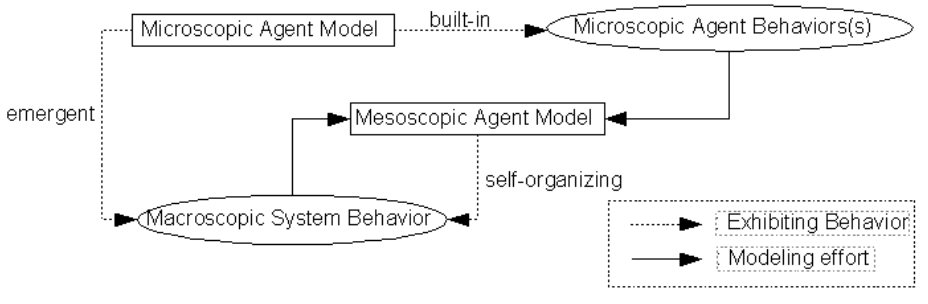


Fig. 1. The relation between micro–, meso– and macroscopic models

3 From Microscopic to Mesoscopic Modeling – A Case Study with Minority Games

The *El Farol Bar Problem* (EFBP) [27] is a famous setting to examine the impact and implications of inductive reasoning for populations of agents. Complex Systems Research has shown that *deductive* reasoning breaks down under complication, forcing agents to use *inductive* reasoning. Agents can get overwhelmed because (1) their *rationality is bounded* or (2) they need to make *assumptions* about the future behaviors of opponents, who do not behave rational [27]. This forces agents to use *inductive reasoning*.

The EFBP is named according to a bar in Santa Fe, which is only enjoyable, if it is not crowded – defined by a certain threshold. Agents are regularly forced to decide, either to *go* there or to *stay home*. In order to do so, they have to anticipate the current amount of customers. Their only source of information is the history of attendings from past evenings.

A socio–economically inspired, exact formulation of this setting is given in the so–called *Minority Game*(MG) [24],[28]. An odd number of N players have to make repeatedly binary decisions (e.g. yes or no, 1 or 0). In an economic interpretation, the players can be regarded as consumers deciding to buy from two suppliers. Only agents in the minority group are considered to be winners, they get rewarded by a score increment. With a memory size of m there are 2^m possible histories and 2^{2^m} possible *strategies* to predict the next choice of the majority. From an economical view–point the global systems behavior is optimized if both suppliers are chosen with approximately equal frequencies. Such globally optimal behavior can be obtained by synchronization to constant groups of winners ($\frac{N-1}{2}$) and losers ($\frac{N+1}{2}$). However to allow maximum individual profit, the groups should be mixed by fluctuating agents.

Several modifications of the game were introduced. These range from evolutionary approaches [29],[30] to stochastic simplifications [31]. Despite exhaustive theoretical investigations on the dynamic behavior and possible optimizations of populations (see [32] for an overview), there are still open questions concerning the simultaneous optimization of possibly deliberative behaviors [33]. We will return to these later. In the following sections, we will consider an *adaptive stochastic* version of the MG, which contains additional time evolving probabilities. In the spirit of models recently discussed [34],[35], our model displays a dynamical evolution of decision behavior of agents. Specific behaviors emerge, i. e. *alternating* or *supplier loyal* behavior, as the system evolves in time.

3.1 Stochastic Minority Game

Here, we summarize definition and results of an adaptive stochastic MG with dynamicly changing strategies we have introduced recently [36]. The game is round-based and consists of an odd number of N agents and two suppliers. Figure 2 summarizes the selection process inside individual agents. In order to choose one of the suppliers – 0 or 1 – at each time step, each agent i keeps a probability $p_i(t) \in (0, 1)$ to change the supplier, i.e. it stays with the same

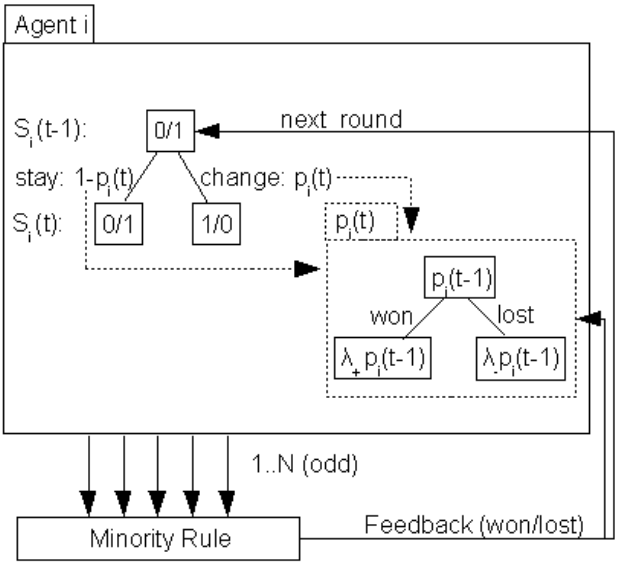


Fig. 2. Selection process inside stochastic agents

supplier as in the former step with probability $1 - p_i(t)$. The result of his choice is denoted by $s_i(t) \in \{0, 1\}$. The supplier chosen by the minority of the agents in time step t makes them winners, the majority having chosen the other supplier loose. At the beginning, suppliers $s_i(0)$ are selected randomly and the p_i are chosen uniformly over the interval $[0, 1]$, in our present study. The dynamical evolution consists in the change of the agents probability $p_i(t)$ after each round. After winning, the probability is multiplied by $\lambda_+ > 0$ whereas after loosing it is multiplied by $\lambda_- > 0$ thereby limiting the maximum of the probability to 1.0 in cases of $\lambda_+, \lambda_- > 1$. Depending on λ_+, λ_- the probability to change the supplier is increased or decreased after winning or loosing, resp. Different regimes of emergent behavior are summarized and discussed in the following subsection.

3.2 Emergent Behavior

The continuous range of probabilities $p_i \in [0, 1]$ defines an infinite set of agent states. Time evolution of the game play will change the initially uniformly distributed population of agents, depending on the values of λ_+, λ_-. The emerging behavior consists of either balancing the agents at mixed strategy or leads to the extreme cases of supplier–loyal agents $p_i \to 0$ or deterministic alternating agents $p_i = 1.0$. defined by ranges of λ_+, λ_-.

Supplier–loyal behavior occurs in particular for $\lambda_+, \lambda_- < 1$. After initial mixing, successive freezing of the agents into supplier–loyal behavior is observed and almost no agent changes side any more, since the $p_i(t)$ become exponentially small. Even if almost half of the agents cannot optimize there individual bahavior in this regime, almost optimal global behavior is still possible depending on parameters and initial conditions.

A similar time–evolution is observed also in the *deterministic–alternation freezing regime* $\lambda_+, \lambda_- > 1$. The difference is that, after the initial mixing stage, the freezing happens according to the fact that all $p_i = 1.0$, and all agents as well as the minority–supplier deterministically alternate.

Interestingly, optimization of global and individually fair behavior (almost equal sides, every agent wins almost half of the time steps) is obtained by mixing in the working regime $\lambda_+ > 1, \lambda_- < 1$, where winners increase their probability to change the supplier in contrast to losers, who increase the tendency to stay, as long as λ_- is not too small.

This emergent behavior cannot be derived directly from macroscopic Rate Equation obtained from the Master Equation by a simple mean–field assumption [18]. Instead, more sophisticated methods of statistical physics are needed. This is not our interest in the present paper.

Here, we focus on the question how emergent behavior can be used to *construct* a less microscopic model which exhibits equivalent macroscopic behavior but in a more controlled way, as explained in section 2.2.

3.3 Mesoscopic Model

So far, an adaptive stochastic MG with dynamically changing strategies has been reported with emergent behavior optimizing global performance as well as individual profit. A model exhibiting equivalent macroscopic behavior by building–in the expected behavior will implement states which directly represent the emergent structures thereby accounting for correlation effects, which are needed to allow for the self–organizing structures wanted. These three strategy states are assumed as representative of the agent behavior in the observed emergent structure, loyal, alternating or undetermined, cf. figure 3.

For the transitions between the states, a deliberative mechanism is introduced. In each state, the agent counts its number of wins and loses. As soon as these counters exceed certain thresholds, a transition into a neighboring state is done. Thus, transitions rates will depend on the individual agents history. In this model, the correlations between the three deliberative states and the supplier choice have to be kept and we end up with a six–state mesoscopic model. These

$s_i(t)$	probability	$s_i(t+1)$
0L		0L
1L		1L
1U	p , 1-p	1U, 0U resp.
0U	p , 1-p	0U, 1U resp.
0A		1A
1A		0A

Fig. 3. The mesoscopic state strategies: Loyal consumers ($1L$,$0L$) continue choosing their last supplier, alternating consumers ($1A$,$0A$) change definitely each time–step, whereas undetermined consumers ($1U$,$0U$) change their supplier with probability p

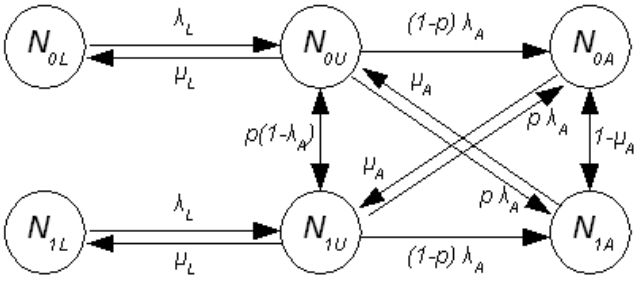


Fig. 4. The mesoscopic states and transitions between them. The transitions rates are time dependent according to our rule described in the text.

hidden states are not directly observable from the macroscopic behavior, which is described by the choice of the supplier solely. The evolving history of the winning supplier side will determine the time dependence of the transition rates shown in figure 4. The time evolution of the occupation numbers of the six states obeys

$$
\begin{aligned}
N_{0L}(t+1) &= (1-\lambda_L(t))N_{0L}(t) + \mu_L(t)N_{0U}(t) \\
N_{1L}(t+1) &= (1-\lambda_L(t))N_{1L}(t) + \mu_L(t)N_{1U}(t) \\
N_{0U}(t+1) &= \lambda_L(t)N_{0L}(t) + \left[(1-\lambda_A(t))(1-p) - \mu_L(t)\right] N_{0U}(t) \\
&\quad +(1-\lambda_A(t))pN_{1U}(t) + \mu_A(t)N_{1A}(t) \\
N_{1U}(t+1) &= \lambda_L(t)N_{1L}(t) + (1-\lambda_A(t))pN_{0U}(t) \\
&\quad + \left[(1-\lambda_A(t))(1-p) - \mu_L(t)\right] N_{1U}(t) + \mu_A(t)N_{0A}(t) \\
N_{0A}(t+1) &= \lambda_A(t)(1-p)N_{0U}(t) + \lambda_L(t)pN_{1U}(t) + (1-\mu_A(t))N_{1A}(t) \\
N_{1A}(t+1) &= \lambda_A(t)pN_{0U}(t) + \lambda_L(t)(1-p)N_{1U}(t) + (1-\mu_A(t))N_{0A}(t)
\end{aligned}
$$

These are time–discrete equations involving the time–dependent deterministic transition rates λ's and μ's. These depend on the internal counters and thereby on the winner side, which is self–consistently determined from

$$N_1 - N_0 = \sum_{j\in\{L,U,A\}} (N_{1j} - N_{0j}) \; .$$

The derivation of macroscopic time–continuous Rate Equations for the relevant slow variables by elimination of fast fluctuations has not been obtained so far.

This six–state mesoscopic model is not a microscopic model any more, since the multiple microscopic agent behaviors of the adaptive stochastic MG with dynamically evolving strategies have been averaged out and replaced be a few deliberative states representing the emergent structures of the microscopic model. On the other hand, it is not a macroscopic model since it is still a a time–discrete MG which has to find its optimum performance by self–organizing agents. These interrelationships are described in figure 1.

3.4 Deliberative Implementation

In the mesoscopic MG described above, the agents observable behavior can be regarded as mental attitudes. In [37] a similar semantic interpretation of agent behaviors, called *personalities*, in a MG like setting are given. The mesoscopic model proposes the available attitudes. The states N_{0L} and N_{1L} denote agents that are *loyal* to a past selection and stick to it. Opposed by the changeable agents in N_{0A} and N_{1A}, who alternate their former selection continuously. Finally, in N_{0U} and N_{1U} we find undecided agents, changing their selection with a given probability p. The observation that the mesoscopic states relate to mental attitudes leads to their implementation in a deliberative agent architecture.

A successful architecture to develop deliberative agents is the BDI model. Bratman [38] developed a theory of human practical reasoning, which describes rational behavior by the notions *Belief*, *Desire* and *Intention*. Implementations of this model introduced the concrete concepts of *goals* and *plans*, leading to a formal theory and an executable model [39],[40].

Beliefs represent the local information of agents about both the environment and its internal state. The structure of the beliefs defines a domain dependent abstraction of the actual environment. It can be regarded as the *view–point* of an agent toward its surrounding. The goals represent agent desires, commonly expressed by certain target states inside the beliefs. This general concept allows to implement both reactive and pro–active behavior. Reactive mechanisms are modeled by goals or plans which are triggered by the occurrence of certain events, while pro-active behavior is implemented by goals which are not directly triggered. Agents carry out these goals on their own (see [41] for a discussion of goals in BDI systems). Finally, plans are the executable means by which agents achieve their goals. In order to reach target states, agents deliberate which plans to execute. This is also known as *reactive planing*, because the precompiled plans are developed at design time. Single plans are not just a sequence of basic actions, but may also dispatch sub-goals.

The Jadex research project[1] [42],[43], provides the BDI–concepts on top of the well known JADE[2] Agent Platform [44]. A suite of tools facilitate the development, deployment and debugging of Jadex–based MAS. The single agents consist of two parts. First, they are described by the so–called *Agent Description Files* (ADF), which denote the structures of beliefs and goals together with other implementation dependent details in XML syntax. Secondly, the activities agents can perform are coded in plans, these are ordinary Java-classes. The goals and plans of the single agents can be described as a tree. Some AOSE and Requirements Engineering (RE) methodologies use corresponding trees to model behavior (see [45] for an overview). The nodes are goals and plans, which both can dispatch subgoals. The leafs of these trees are anyway plans, since they are the only means to perform activities. Figure 5 gives an overview of the goal hierarchy of individual agents in the mesoscopic adaptive MG. Rectangles denote the plans available to the agent, ovals the goals which are used in the delibera-

[1] http://vsis-ww.informatik.uni-hamburg.de/projects/jadex
[2] http://jade.tilab.com/

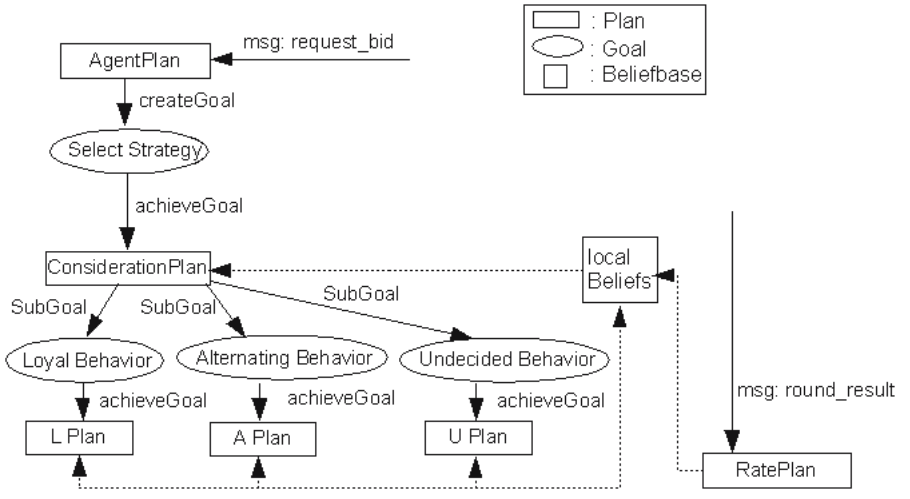


Fig. 5. Goal/Plan hierarchy of the BDI implementation

tive process to determine which plan to execute. The call (*msg:request_bid*) for the next round $(i + 1)$ is processed by a plan in each single agent (*AgentPlan*). Upon arrival a new goal (*Select Strategy*) is instantiated, which is achieved by a dedicated plan (*ConsiderationPlan*). This plan has access to the local beliefbase to determine the history, the current state and the values of λ, μ and p. Upon these data, the next state is selected and executed by dispatching a subgoal. The three available subgoals correspond to the three pairs of mesoscopic states described above. The associated plans access the beliefbase to determine the selection in round i and update the beliefbase to indicate the selection for the current round $i + 1$. Agents get informed about the result of each round by reception of a message (*msg:round_result*). A Plan (*RatePlan*) is responsible to process this message and to update of the local beliefs.

3.5 Comparison and Results

Our mesoscopic deliberative MG can balance agents at mixed strategy (mixing regime) or can lead to the extreme cases of supplier–loyal strategy or deterministic alternating strategy (freezing regimes), as the stochastic MG does. These behaviors are observed in different regimes defined by ranges of the parameters λ's, μ's and p of the six–states. This mesoscopic deliberative MG is found to exhibit equivalent macroscopic behavior as the microscopic adaptive stochastic MG with infinitely many states. This is shown in more detail in the rest of this section.

Both freezing regimes are found in a large range of parameter values in both the mesoscopic and the microscopic model. Also in these regimes, globally optimal behavior accompanied by segregation of the agent population is found. In the mixing regime global optimization is observed at *fair* individual behavior

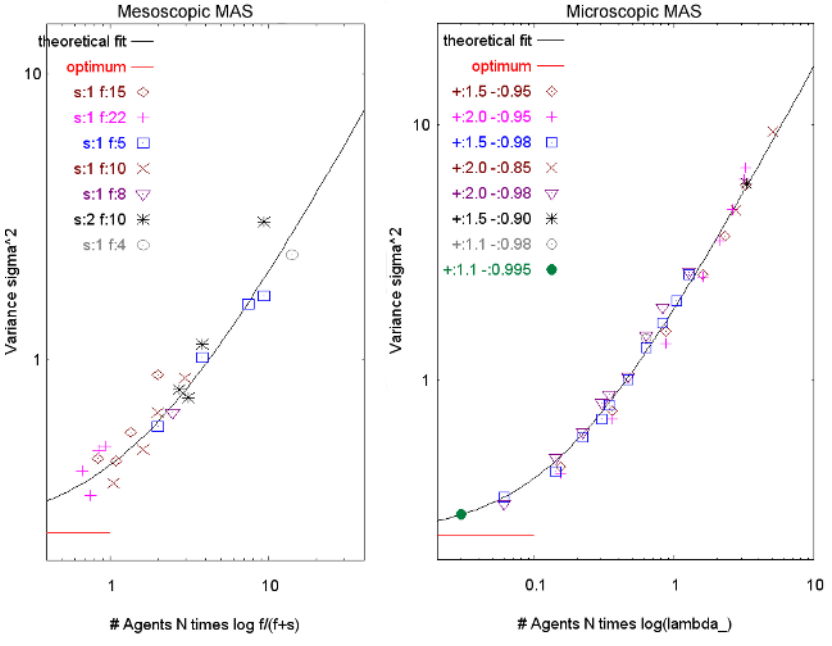


Fig. 6. The squared global loss is plotted as a function of the properly scaled number of agents. Using an information–theoretic inspired scaling with $\ln \lambda_-$ in the microscopic MG and with $\ln \frac{f}{f+s}$ in the mesoscopic MG, a data collaps is obtained with a crossover from the optimum solution at $\lambda_- \to 1$ or $f \to \infty$, resp., to the $O(N)$ behavior. Data of the microscopic MAS are taken from [36]. Further explantation see text.

in the following sense. Both suppliers are chosen by an almost equal number of agents, while every agent wins almost half of the time steps, the best individual solution without segregation. It is obtained in the region with $\lambda_j > \mu_j$ for $j = L, A$ in the mesoscopic model. and in part of the region $\lambda_+ > 1, \lambda_- < 1$ in the microscopic model. Also, *metastability* is observed as in our stochastic MG. Further details of the adaptive stochastic MG see [36].

The dynamic behavior of the MAS can be understood from the individual agents behavior. We have analyzed the mixing regimes in greater detail. Here, the agents prefer the two opposite groups in the alternating state. According to the deterministic transition rules, each agent returns to the undetermined state after a while, thus getting the chance to return to the winners side if it was on the losers side during its last alternating state. So, all agents rotate through winner and loser side thereby optimizing the global system behavior. This self–organizing macroscopic behavior is intended by the mesoscopic construction in contrast to the emergent behavior of the stochastic MG.

In figure 6 we compare results for our mesoscopic MG in the mixing regime with parameters $\lambda_L = 1$, $\lambda_A = 1/s$ if the agent won, $\mu_L = 1/10$, $\mu_A = 1/f$ if the agent lost and $p = 0.95$ with those or our stochastic MG in its mixing regime, i.e. the part of the region $\lambda_+ > 1, \lambda_- < 1$ with $1 - \lambda_- << \lambda_+ - 1$. Equivalent

behavior occurs over a hole range of parameters. The average loss of the system, i.e. the deviation from global optimum, is plotted as a function of the properly scaled number of agents. The scaling brings all simulated data of the chosen parameter range onto one curve described by a theoretical fit, which shows a crossover between the theoretic optimum and an $O(N)$ behavior. The data collapse is obtained by an information–theoretic inspired scaling with $\ln \lambda_-$ in the microscopic MG and with $\ln \frac{f}{f+s}$ in the mesoscopic MG for the parameters examined here. This means that our MGs generate globally optimal as well as individually fair behavior in the mixing regimes at arbitrarily large agent numbers N in the limits $\lambda_- \to 1$ or $f \to \infty$, respectively. Instead of the limit $f \to \infty$, other limits in parameter space can used for optimal behavior, e.g. $\mu_L \to 0$ in conjunction with $p \to 1$ is expected to work as well.

Further insight into the system behavior is expected from a study of the mesoscopic equations in section 3.3.

4 Conclusions

In this paper, we have shown how mesoscopic modeling, inspired by its usage in statistical physics [26], can be used to analyse and quantify MAS exhibiting emergent behaviors. In addition to the mathematical description of system behavior, the identified mesoscopic states lead to a deliberative implementation of agents, showing equivalent system behavior. Our model gives theoretical foundations for simulations using mentalistic abstractions in large scale simulations, e.g. [37].

The presented constructive approach deduces a number of mesoscopic agent states from observed emergent MAS behavior. By means of self–organization these states lead to similar behavior in a controlled and predictive way. The identified states directly reflect macropscopic emergent phenomena. Therefore we expect the number of mesoscopic states to scale proportional to the later. This will be a topic of further research.

Economic development efforts require developers to deliberate about the agent architecture to be employed under certain circumstances. Even when simple reactive or stochastic agents are used, development teams have a macroscopic behavior in mind when designing MAS agents. Therefore it would be valuable to examine if methodical ways are possible to derive microscopic states from deliberative mesoscopic models. Large numbers of agents would average out fluctuations, allowing developers to revise previously defined models of the desired agent behavior to a microscopic implementation of fairly simple agents.

The presented approach to the development of similar behavior does not only contribute to research of universality in MAS [10], but has also impact on MAS design. It allows to *construct* equivalent emergent phenomena in MAS. Furthermore, we expect that the deliberative model allows optimization of fitness–functions for individuals with coexistent optimized global system behavior. The simultaneous optimization of possibly deliberative behaviors is still an open question in the case of the MG [33]. How these abstractions can be used to *engineer* reliable self–organized MAS and limitations of this approach are topics of further investigation.

Acknowledgements

One of us (J.S.) would like to thank the *Distributed Systems and Information Systems* (VSIS) group at Hamburg University, particularly Winfried Lamersdorf, Lars Braubach and Alexander Pokahr for inspiring discussion and encouragement.

References

1. Jennings, N.R.: On agent-based software engineering. Artif. Intell. **117** (2000) 277–296
2. Jennings, N.R.: Building complex, distributed systems: the case for an agent-based approach. Comms. of the ACM **44 (4)** (2001) 35–41
3. Iglesias, C., Garrijo, M., Gonzalez, J.: A survey of agent-oriented methodologies. In: Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98). Volume 1555. (1999) 317–330
4. Tveit, A.: A survey of agent-oriented software engineering (2000)
5. Wei, G.: Agent orientation in software engineering. Knowledge Engineering Review **16(4)** (2002) 349–373
6. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers (2001)
7. Parunak, H.V.D.: Making swarming happen. In: Proceedings of Swarming and Networkenabled C4ISR. (2003)
8. Ando, Y., Masutani, O., Sasaki, H., Iwasaki, H., Fukazawa, Y., Honiden, S.: Pheromone model: Application to traffic congestion prediction. In: Proceedings of the International Workshop on Engineering Self-Organising Applications 2005. (2005)
9. Tatara, E., North, M., Hood, C., Teymour, F., Cinar, A.: Agent–based control of spatially distributed chemical reactor networks. In: Proceedings of the International Workshop on Engineering Self-Organising Applications 2005. (2005)
10. Parunak, H.V.D., Brueckner, S., Savit, R.: Universality in multi-agent systems. In: AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, IEEE Computer Society (2004) 930–937
11. Bordini, R.H., Fisher, M., Visser, W., Wooldridge, M.: Model checking rational agents. In: IEEE Intelligent Systems 19(5). (2004) 46–52
12. Poutakidis, D., Padgham, L., Winikoff, M.: Debugging multi-agent systems using design artifacts: The case of interaction protocols (2002)
13. Zambonelli, F., Mamei, M., Roli, A.: What can cellular automata tell us about the behavior of large multi-agent systems? In: Proceedings of SELMAS 2002. Volume 2603 of Lecture Notes in Computer Science., Springer (2002) 216–231
14. Wolf, T.D., Holvoet, T., Samaey, G.: Engineering self-organising emergent systems with simulation-based scientific analysis. In: Proceedings of the International Workshop on Engineering Self-Organising Applications 2005. (2005)
15. Brueckner, S., Parunak, H.V.D.: Information-driven phase changes in multi-agent coordination. In: Proceedings of the International Workshop on Engineering Self-Organising Applications 2005. (2005)
16. Lerman, K., Galstyan, A.: Automatically modeling group behavior of simple agents. In: Agent Modeling Workshop, AAMAS-04, New York, NY (2004)
17. Galstyan, A., Lerman, K.: Analysis of a stochastic model of adaptive task allocation in robots. In: AAMAS-04, New York, NY (2004)

18. Lerman, K., Galstyan, A.: A general methodology for mathematical analysis of multiagent systems (2001)
19. Lerman, K.: Design and mathematical analysis of agent-based systems. In: FAABS. (2000) 222–234
20. Agassounon, W., Martinoli, A., Easton, K.: Macroscopic modeling of aggregation experiments using embodied agents in teams of constant and time-varying sizes. In: Autonomous Robots Journal. (2003)
21. Lerman, K., Galstyan, A.: Mathematical model of foraging in a group of robots: Effect of interference. In: Autonomous Robots. Volume 13. (2002) 127–141
22. Lerman, K., Galstyan, A.: Agent memory and adaptation in multi-agent systems. In: AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, ACM Press (2003) 797–803
23. Wolf, T.D., Holvoet, T.: Emergence and self-organisation: a statement of similarities and differences. In: Proceedings of the International Workshop on Engineering Self-Organising Applications. (2004) 96–110
24. Challet, D., Zhang, Y.C.: Emergence of cooperation and organization in an evolutionary game. Physica A 246, 407 (1997)
25. Martinoli, A., Easton, K.: Modeling swarm robotic systems. In Siciliano, B., Dario, P., eds.: Proc. of the Eight Int. Symp. on Experimental Robotics ISER-02. Number 5 in Springer Tracts in Advanced Robotics (2003) 297–306
26. Van Kampen, N.G.: Stochastic Processes in Physics and Chemistry. Revised edn. Elsevier Science Pub Co (2001)
27. Arthur, W.B.: Inductive reasoning and bounded rationality. American Economic Review **84** (1994) 406–11 available at http://ideas.repec.org/a/aea/aecrev/v84y1994i2p406-11.html.
28. Li, Y., VanDeemen, A., Savit, R.: The minority game with variable payoffs. Physica A, 284(2000):461-477 (2000)
29. Metzler, R., Horn, C.: Evolutionary minority games: the benefits of imitation. In: Physica A329. (2003) 484–498
30. Johnson, N., Hui, P., Jonson, R., Lo, T.: Self-organized segregation within an evolving population. In: Physical Review Letters 82, 3360. (1999)
31. Reents, G., Metzler, R., Kinzel, W.: A stochastic strategy for the minority game. In: Physica A 299. (2001) 253–261
32. Challet, D., Marsili, M., Zhang, Y.C.: Minority Games - Interacting agents in financial markets. Number 0-19-856640-9 in Series: Oxford Finance Series. Oxford University Press (2004)
33. Challet, D.: Competition between adaptive agents: from learning to collective efficiency and back. In: chapter to appear in *Collectives and the design of complex systems*. cond-mat/0210319, Springer (2003)
34. Xie, Y., Wang, B.H., Hu, C., Zhou, T.: Global Optimization of Minority Game by Smart Agents. ArXiv Condensed Matter e-prints (2004)
35. Zhong, L.X., Zheng, D.F., Zheng, B., Hui, P.: Effects of contrarians in the minority game. In: cond-mat/0412524. (2004)
36. Renz, W., Sudeikat, J.: Modeling minority games with bdi agents - a case study. In: Proc. of the Third German Conference on Multi-Agent System Technologies (MATES 05),. Volume LNAI 3550., Springer (2005) 71–81
37. Bazzan, A.L., Bordini, R.H., Andrioti, G.K., Vicari, R.M.: Wayward agents in a commuting scenario (personalities in the minority game). In: Proc. of the Fourth Int. Conf. on Multi-Agent Systems (ICMAS'2000), Boston, IEEE Computer Science (2000)

38. M.E.Bratman: Intentions, Plans, and Practical Reason. Harvard Univ. Press. (1987)
39. Rao, A.S., Georgeff, M.P.: BDI-agents: from theory to practice. In: Proceedings of the First Intl. Conference on Multiagent Systems, San Francisco (1995)
40. Rao, A.S.: Agentspeak(l): Bdi agents speak out in a logical computable language. In: MAAMAW '96: Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away, Springer-Verlag New York, Inc. (1996) 42–55
41. Braubach, L., Pokahr, A., Lamersdorf, W., Moldt, D.: Goal representation for bdi agent systems. In Bordini, R.H., Dastani, M., Dix, J., Fallah-Seghrouchni, A.E., eds.: Second International Workshop on Programming Multiagent Systems: Languages and Tools. (2004) 9–20
42. Braubach, L., Pokahr, A., Lamersdorf, W.: Jadex: A short overview. In: Main Conference Net.ObjectDays 2004. (2004) 195–207
43. Pokahr, A., Braubach, L., Lamersdorf, W.: Jadex: Implementing a bdi-infrastructure for jade agents. EXP - in search of innovation (Special Issue on JADE) **3** (2003) 76–85
44. Bellifemine, F., Rimassa, G., Poggi, A.: Jade a fipa-compliant agent framework. In: In 4th International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-99), London, UK (1999) 97108
45. van Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour. In Proc. RE01 - Int. Joint Conference on Requirements Engineering (2001)

Pheromone Model: Application to Traffic Congestion Prediction

Yasushi Ando[1], Osamu Masutani[2], Hiroshi Sasaki[2], Hirotoshi Iwasaki[2], Yoshiaki Fukazawa[1], and Shinichi Honiden[3]

[1] Department of Science and Engineering,
Waseda University, 3-4-1 Okubo,
Shinjuku-ku, Tokyo, 169-8555, Japan
{ando, fukazawa}@fuka.info.waseda.ac.jp
[2] Research and Development Group,
DENSO IT Laboratory, Inc. 3-12-22 Shibuya,
Shibuya-ku, Tokyo, 150-0002, Japan
{omasutani, hsasaki, hiwasaki}@d-itlab.co.jp
[3] National Institute of Informatics,
2-1-2 Hitotsubashi Chiyoda-ku Tokyo, 101-8430, Japan
honiden@nii.ac.jp

Abstract. Social insects perform complex tasks without top-down style control, by sensing and depositing chemical markers called "pheromone". We have examined applications of this pheromone paradigm towards intelligent transportation systems (ITS). Many of the current traffic management approaches require central processing with the usual risk for overload, bottlenecks and delays. Our work points towards a more decentralized approach that may overcome those risks. In this paper, a car is regarded as a social insect that deposits (electronic) pheromone on the road network. The pheromone represents density of traffic. We propose a method to predict traffic congestion of the immediate future through a pheromone mechanism without resorting to the use of a traffic control center. We evaluate our method using a simulation based on real-world traffic data and the results indicate applicability to prediction of immediate future traffic congestion. Furthermore, we describe the relationship between pheromone parameters and accuracy of prediction.

1 Introduction

Many social insect species such as ants, bees, and wasp coordinate the activities of individuals in the colony without direct communication or complex reasoning. Instead, they deposit and sense chemical markers called "pheromone" in a shared physical environment (pheromone potential field) that participates actively in the system's dynamics. Here, we focus on the propagation model of the pheromone potential field. We will examine applications of this pheromone paradigm towards intelligent transportation systems (ITS).

ITS have spread widely recently. Beyond car navigation systems and traffic information systems[1], various types of sensors and communication devices are

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 182–196, 2006.

being introduced for various kinds of application. Solving traffic congestion is a serious problem among such applications.

One of the key technologies to solve this problem is traffic prediction. There are traffic information systems which are actually used, such as VICS (Vehicle Information and Communication System) [2] which started in Japan in 1996. Drivers can easily grasp the current surrounding traffic situations through VICS. However, the problem of VICS is that traffic information may be delayed due to communication between the road and data center or due to processing time at the data center. This leads to driver often receiving traffic congestion data that are different from the actual traffic.

Another problem is that drivers receive current traffic information only when they search for the shortest route to their destination. That current traffic data will become useless after a while on the route. Traffic prediction can solve this problem by predicting appropriate data for each stage of traffic data on the route. Traffic prediction techniques are widely studied in many institutions [3] and are already introduced in commercial car navigation systems [4]. Some of such techniques deal with long-term predictions such as predictions spanning 1 hour to 1 day based on statistical analysis of past data. However, long-term predictions cannot be applied to driving in a small area and cannot predict the innately sharp fluctuation of traffic. Short-term prediction is suitable to driving in a small area and to make predictions accurate.

Short-term prediction methods have also been studied [5],[6]. However most of the studies use statistical data of the past. As such, it does not work sufficiently with sharp or irregular fluctuation of traffic.

We propose a short-term traffic prediction using the pheromone model which has a relatively simple mechanism that doesn't require statistical data. In our method, a car is regarded as an agent or an ant. Each car would deposit some pheromone into a virtual space with its amount based on the congestion level. And the total pheromone amount at each location will be used to predict the traffic in the immediate future.

We experimented with the pheromone prediction method based on real traffic data to evaluate our method. The results indicate its applicability to prediction of traffic congestion in the immediate future. Furthermore we investigated the relationship between pheromone parameters and prediction accuracy.

2 Pheromone

2.1 Basic Mechanisms

The term "Pheromone" (Greek for "carrier of excitement") was named by P. Karlson and M. Luscher in 1950s [7]. Pheromone is defined as communication chemical that possesses certain meanings that are understood between individuals of the same species. Pheromones have various kinds of functions. For example, ants use trail pheromones to guide other ants to food, and bees use alert pheromones to inform of disturbance to other bees.

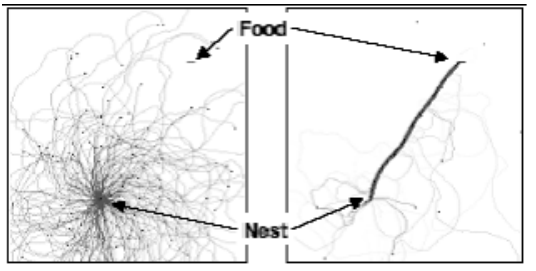


Fig. 1. Action pattern of ants which used trail pheromone. Quoted from document[9].

The trail pheromone mechanism has already been applied to certain areas to efficiently solve problems such as the Traveling Salesman Problem [8]. The mechanism works as follows: Ants find food and transport it to their nest depositing trails of pheromone on their returning path. The pheromone diffuses into the surrounding environment and its intensity weakens over time. However, the density of pheromone left on the path where ants have passed most frequently becomes condensed. Eventually, pheromone on the shortest path to the food becomes most dense (see Figure 1).

On the other hand, bees use alarm pheromone. A leading bee would inform of any danger using alarm pheromone and other bees that follow would avoid the alarm pheromone. This simple mechanism also has been applied to some technologies [10].

2.2 Characteristics

Pheromone is a simple chemical media for delivering information. Although each individual ant or bee possesses just a simple and local behavior rule, they are capable of enabling complex functions. This is largely related with the characteristics of "evaporation" and "propagation" [11]. "Evaporation" represents old information disappearing as time passes due to the volatile characteristics of pheromone. This characteristic enables other ants to receive the most recent information. "Propagation" represents the characteristics of pheromone spreading out towards the surrounding environment. This allows other ants in different places to receive this information.

3 Model for Traffic

This paper discusses the use of this dynamism to predict traffic. The model used here will regard cars as ants or bees and will deposit pheromone on the pheromone potential field. Cars equipped with speed sensors would deposit some amount of pheromone in proportion to their speed. Other cars that follow their route would avoid traffic congestion by checking the intensity of pheromone.

3.1 Brueckner's Model

We propose a traffic prediction pheromone model based on the pheromone transition model proposed by S. Brueckner [12]. He defined pheromone as having the following characteristics:

1. Pheromone is information deposited by agents when an event occurs.
2. Deposited pheromone is aggregated in place and represents the information potential of each location.
3. Pheromone evaporates over time thus decreasing its value.
4. Pheromone propagates towards the surrounding environment.
5. And each agent is affected by pheromone.

According to this definition, we defined traffic prediction pheromone as follows.

1. Pheromone is information deposited by cars based on the surrounding traffic situation.
2. Deposited pheromone is aggregated on the road and represents the traffic situation of each location.
3. Pheromone evaporates over time. This corresponds with traffic moving towards another location over time.
4. Pheromone is propagated towards nearby roads. This corresponds with traffic movement along flow.
5. And each driver is affected by the traffic prediction based on pheromone.

3.2 Transition Functions

Brueckner's Model is a kind of Coupled Map Lattice (CML) [13] or continuous cellular automaton. This section defines detail transition functions of aggregation, evaporation and propagation.

Aggregation. Each car is equipped with a speed sensor and a communication device and deposits (aggregates in place) pheromone according to congestion rate. Generally, speed is used to represent congestion rate, so car agent will deposit some amount of pheromone according to its speed.

Let aggregation of pheromone be $r(t, p)$ in time t and a location p. Value $v(t, p)$ is the speed of a car in time t and place p and α is the aggregation parameter. This function is represented as follows.

$$r(t, p) = \frac{\alpha}{v(t, p)} \tag{1}$$

Equation (1) has meanings as follows.

- The deposit increases with decreasing speed (Figure 2 (a)).
- The deposit decreases with increasing speed (Figure 2 (b)).

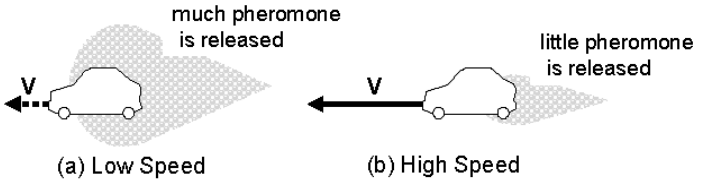


Fig. 2. The image of a car agent deposits pheromone

This model assumes existence of information servers at each location. Cars would send certain amounts of pheromone uplink to be aggregated at the information server. Information servers referred in this paper would be prepared for each road link[1] .

Evaporation. Aggregated pheromone decreases over time due to evaporation. Let the amount of pheromone be $s(t,p)$ in time t and place p. The predicted amount of pheromone $s(t+1,p)$ is represented as equation as follows.

$$s(t+1,p) = E \times s(t,p) + r(t,p) + q(t,p) \tag{2}$$

Value E ($E \in (0,1)$) is the evaporation parameter. $q(t,p)$ is the amount propagated from the neighboring servers as shown in the next section.

Propagation. Propagation of pheromone to neighbors is realized by communication among information servers. Transition function of the propagated amount $q(t,p)$ is shown as follows.

$$q(t+1,p) = \sum_{p' \in N(p)} \frac{F}{|N(p)|}(r(t,p') + q(t,p')) \tag{3}$$

$N(p)$ represents the upper stream neighbors of location p that affect the next status of location p. F is the propagation rate that is a function of the pheromone value of the upper stream in our model. In traffic flows in the road network, the movement of cars is only goes straight or turns. Thus, the propagation rate function from a neighboring location p' to a location p is described as follows using propagation parameters α and β.

$$F = f(p,p') = \begin{cases} \beta & \text{when p and p' are on straight line} \\ \frac{\beta}{s(t,p)}(s(t,p) > c) & \\ \gamma & \text{when p and p' aren't on straight line} \\ \frac{\gamma}{s(t,p)}(s(t,p) > c) & \end{cases} \tag{4}$$

[1] Judging from the situation that ITS spread, the assumption that a network of information servers would exist will become realistic. For example, for intellectual control of a signal, an idea to install a simple information server every each intersection is suggested[14]. On the other hand, technology to make cars servers temporarily is studied[15], [16] .

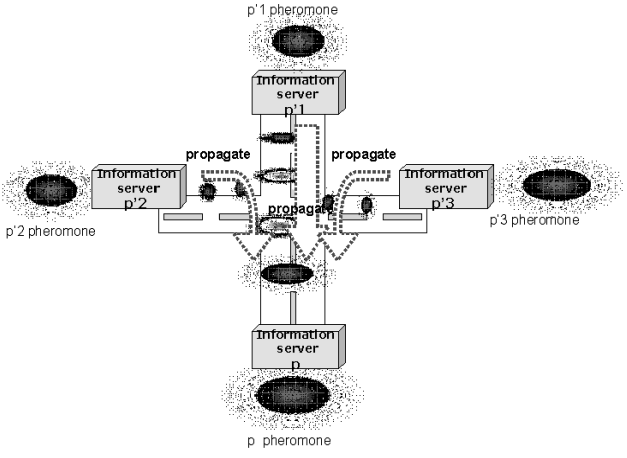


Fig. 3. Pheromone propagation

And we defined a congestion threshold c. When traffic congestion occurs down stream from p , propagation rate is an inverse proposition to the pheromone amount in p .

4 Evaluation

We conducted a simulation of our model using actual traffic benchmark data. Then we evaluated how accurate our model predicts traffic based on error indices. This chapter describes the data used in the simulation, assumptions of the simulation, and simulator specification.

4.1 Data

The data we chose has high time granularity and a relatively sharp fluctuation in traffic. The real data used in the simulation is "Kichijoji-Mitaka benchmark dataset" provided by i-Transport Lab. Co., Ltd. to the general public [17]. This dataset consists of road link information, complete trail data of approximately 12,000 cars in an area between Kichijoji and Mitaka station for about 2 hours on a certain day in 1996 (See Figure 4). These data was collected by human observers at about 100 observation points in the area. Each trail data includes license plate data and time data to the minute, representing the time when a car passed an observation points.

4.2 Assumptions for the Simulation

The simulation has been performed under the following assumptions.

- All cars are equipped with speed sensors and communication devices.
- Each lane of a bi-directional (ordinary) road has an information server of its own.

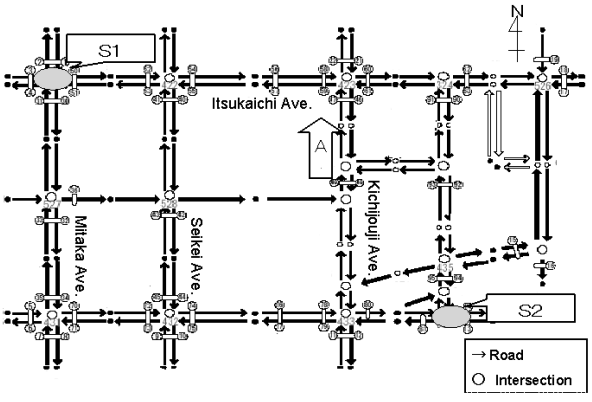


Fig. 4. Map of the area where actual survey data were observed

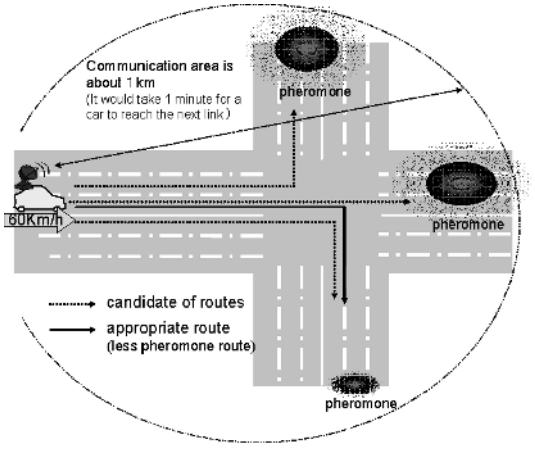


Fig. 5. A communication range of a car and relation of prediction time

– Cars send their speed information uplink to information servers corresponding to road links.
– Information servers are installed for every road link on the map and maintain local pheromone dynamics.
– Information servers can communicate with neighboring information servers.

Note that, as the pheromone value, we use link travel time (the time it takes for a car to travel a road link) which is calculated from the average speed of a car along the link distance rather than the speed of a car itself.

Average link travel time is around 1 minute since the average link length on the map is around 1 km and the speed of the fastest car is around 60 km/h. Therefore it would take 1 minute for a car to reach the next link after a prediction has been provided from a preceding car. Thus, in the simulation, transition time

step is set to 1 minute and the pheromone model predicts 1 minute into the future (See Figure 5).

4.3 Simulator

Our simulator is designed to evaluate the pheromone model and some other baseline model. And it calculates the error rate of these models. It also has visualization features to read actual traffic and predicted traffic through animation (See Figure 6).

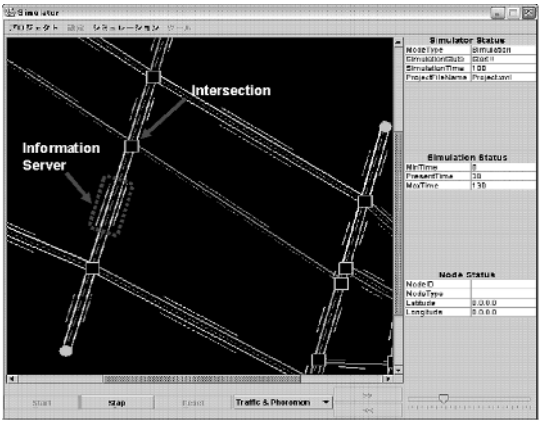


Fig. 6. Screenshot of simulator

5 Result of Experiments

Evaluation has been performed from the standpoint of prediction accuracy, relationship between parameters and accuracy, and effect on the search of shortest route to destination.

5.1 Experiment 1: Prediction Accuracy

Prediction accuracy is calculated as error rates between the predicted link travel time sequence and the actual link travel time sequence. Actual link travel time is a simple average of all passed cars within a single transition time step.

Predicted link travel time is a scaled pheromone amount based on a scaling parameter and length of each link. The scaling parameter is empirically set to 0.04. Let $Prt(t,p)$ be the predicted link travel time and S be the global scaling parameter and $l(t,p)$ be the link length (m) in a link p and a time t.

$$Prt(t,p) = s(t,p) \times S \times l(t,p) \tag{5}$$

The error rate indices we used are correlation coefficient and RMS (Root Mean Squared) error.

The prediction results are compared with two types of baseline predictions. One is a prediction based on a simple moving average time. The other is a prediction based on a persistent prediction model which assumes that the current situation will persist for some duration of time. For example, in a 5 minute persistent prediction model, the average value between 8:50 and 8:55 will persist until 9:00. The current VICS information can be received within 5 minute or more after sensing, so we can assume that a 5 minute persistent predication model corresponds with VICS data. The results of this experiment are also compared with a 1 minute persistent prediction model based on the assumption that the delay may be reduced in the future.

Result 1: Prediction Accuracy. Model parameters are empirically determined as follows: Aggregation parameter $\alpha = 2.0$, evaporation parameter $E = 0.8$, propagation parameter for straight link $\beta = 0.8$, propagation parameter for turning links $\gamma = 0.1$, congestion threshold $c = 10.0$.

For example, Figure 7 shows fluctuation of actual traffic and traffic predictions in the 2 hours between 7:50 and 10:00 at link A of Figure 4. Actual traffic is the average link travel time [seconds] of all cars that passed the link at the time the actual data was collected. Predicted link travel times [seconds] of 4 types of prediction methods are also drawn in the same graph.

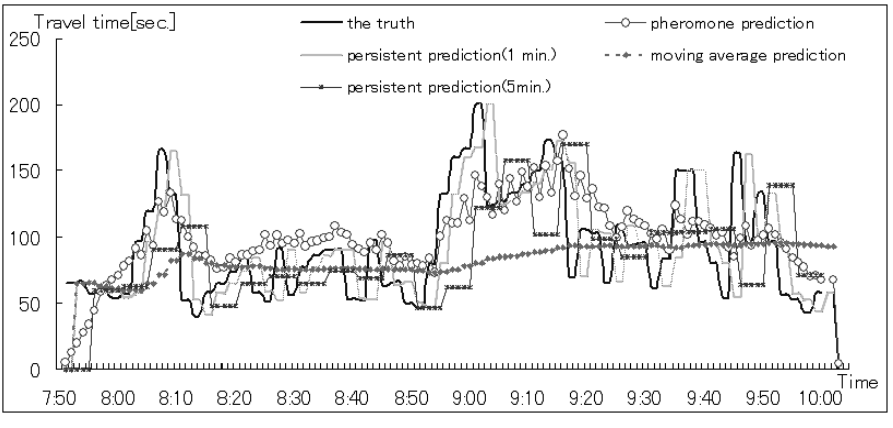


Fig. 7. Actual travel time, pheromone and persistent prediction in section A

Table 1. Accuracy of pheromone and persistent prediction(Section A)

	Correlation coefficient	RMS error[sec]
Pheromone	0.69	31.5
1 minute persistent	0.48	35.2
5 minute persistent	0.36	45.3
moving average	0.26	41.0

Table 2. Average accuracy of pheromone and persistent prediction

	Ave. Correlation coefficient	Ave. RMS error[sec]
Pheromone	0.59	45.6
1 minute persistent	0.41	56.5
5 minute persistent	0.41	46.6
moving average	0.30	49.2

The correlation coefficient and RMS error are shown in Table 1. These result shows that our method outperforms baseline predictions. We can say that baseline predictions become worse when traffic fluctuation becomes very sharp.

Additionally, overall result of all links on the map (Westside of Kichijoji street) as Table 2 shows that pheromone prediction has higher accuracy than the other methods.

5.2 Experiment 2: Relationship Between Parameters and Prediction Accuracy

Next we study the effect of different patterns of parameters. First, we assumed the following assumption regarding the aggregation parameter and evaluated the relationship.

> Existence of commercial vehicles such as buses or taxis would slow down the traffic since such vehicles stop abruptly or are hardly overtaken.

Based on this assumption, separate values for the aggregation parameter α were employed for commercial vehicles and other vehicles. The effect of the ratio of these aggregation parameters towards prediction accuracy is evaluated in this experiment.

Furthermore, relationship between the evaporation parameter and prediction accuracy is evaluated.

Result 2: Relationship between Parameters and Prediction Accuracy. Results of the sensitivity analysis of aggregation parameter are shown in Figure 8. The X axis represents $G : C$ ratio where G is the aggregation parameter of a general car and C is parameter of a commercial vehicle. The Y axis represents the number of links that have highest correlation coefficient with the parameter ratio. Assigning the same aggregation parameter values for the general and commercial vehicles marked the highest accuracy in most of the links. Furthermore, we confirmed that in such links, the $G : C$ ratio scarcely affected the accuracy. However, in some links, accuracy increased when the aggregation parameter value for commercial vehicles was increased. In such links, a ratio of 1:2 to 1:7 produced the best results. The number of buses or taxis in these links was higher than other links. Therefore the assumption that commercial cars would affect the traffic and slow it down was proved by this experiment.

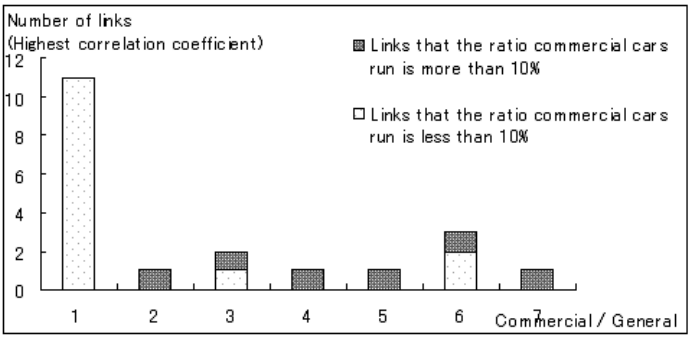


Fig. 8. Sensitivity of aggregation parameters varied in general cars and commercial cars

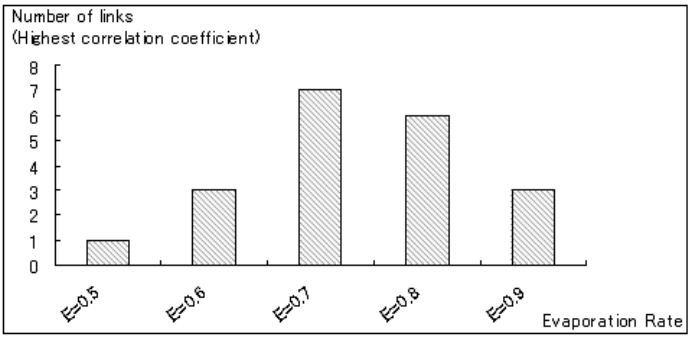


Fig. 9. Sensitivity of evaporation parameter

This result implies that car agents should vary its pheromone emission according to its type of usage (commercial or general). Such type information of vehicles can be easily provided if a car is equipped with a simple communication device and sends the information uplink. If such communication device is not available in a car, some types of car detector (such as video-based detector) can be used to detect the type of usage of a car.

Results of the sensitivity analysis of evaporation parameter are shown in Figure 9. The X axis represents the evaporation parameter setting and the Y axis represents the number of links which mark the highest correlation coefficient with the parameter. The optimal evaporation parameter is around 0.7. Furthermore, the parameter is not affected by the ratio of commercial vehicles.

5.3 Experiment 3: Improvement of Shortest Route Search

Finally we apply the prediction results to the shortest route search. The shortest route search can be improved by traffic prediction since actual travel time is affected by the situation of traffic. We compared the results of the shortest route

Table 3. The shortest time route selection rate(from S1 to S2)

	number	percentage[%]
Pheromone	63	54.3
1 minute persistent	42	36.2

Table 4. The shortest time route selection rate(from S2 to S1)

	number	percentage[%]
Pheromone	61	52.6
1 minute persistent	44	37.9

search with the prediction based on our method and the prediction based on the baseline prediction. A number of selection of the right shortest route by each search is used as indices.

In this experiment, the shortest route search is performed with one set of origin and destination designated as S1 and S2 in Figure 4. Searches for both directions were performed at each minute beginning from 7:53 until 9:48 (116 times). We adopted Dijkstra's shortest path algorithm to find the overall optimal solution. Dijkstra's algorithm is also generally used in commercial car navigation systems.

Note that this congestion-aware search is updated incrementally. A running car updates its estimated shortest route for each transition time step since predictions would change as the car runs.

Result 3: Improvement of Shortest Route Search. Table 3 and Table 4 show the results of the shortest route search with each prediction method. For both directions of the origin-destination set, the pheromone prediction method was better than the persistent prediction method. The absolute accuracy percentage of selection of shortest route was around 50%. The low overall accuracy is considered to be caused by the very sharp fluctuation of traffic in this area. The area used in this experiment is small and it does not have much variation in the selection of routes so we may have to experiment on larger maps.

6 Related Works

Application of the pheromone mechanism has been studied in the so-called swarm intelligence research field. On the other hand, traffic congestion prediction has various kinds of solutions such as the statistical method and simulation-based method. Such related works are discussed in the following sections.

6.1 Application of the Pheromone Mechanism

There are various studies concerning application of the pheromone mechanism. This paper regards mobile agents that deposit pheromone as cars, which is similar to the approaches used in actual agents for sensors or planes.

One example of an application is the unmanned plane in a war field [10][9] The unmanned plane would launch mobile agents which would deposit alarm pheromone when they encounter enemy missile sites. The plane would analyze the pheromone potential field and then decide upon a route to its destination.

Another example of an application is mobile sensor network.[18] The study of this application concerns methods to improve the efficiency of the overall monitoring task by making mobile sensors deposit alarm pheromone to indicate that it has observed a certain area and that other mobile sensors can avoid observing the same place again.

These researches use alarm (negative) pheromone while most of the researches regarding swarm intelligence use trail (positive) pheromone. These researches also deal with the flexible nature of pheromone appropriately. However, note that the researches are evaluated using virtual data and not actual data. Our study has performed evaluations using real traffic data to confirm the applicability to traffic prediction.

6.2 Traffic Congestion Prediction

There are two types of methods of short-term traffic congestion prediction in the field of traffic engineering research. One is based on a statistics of past traffic fluctuation patterns. The other is based on traffic simulation.

Statistical prediction is performed by matching current traffic patterns (time series data) to typical traffic patterns in the past. The most applicable pattern within the past traffic patterns is used for future prediction. Time span may vary from minutes to days depending on its application. The algorithms used in the matching process for short-term prediction are neural network [5] or Bayesian Network [6] and so on. The cost incurred for processing is significant since large amounts of history data of past traffic is used. Furthermore, prediction of irregular traffic patterns can not be achieved using only this pattern-matching based method.

On the other hand, traffic simulation is used for short-term traffic prediction [19]. The simulation model that describes the mutual interference of cars consists of running car dynamics, road type, congestion model and driver model. Thus, the simulation tends to be complicated and time consuming. Distributive methods of simulation such as cellular automaton (CA) are also introduced into this traffic prediction [20]. While prediction using CA has proved to be effective when information regarding incoming stream volume is available, it does not seem to be sufficient when the information is partially not available, such as when targeting normal road data.

Naturally, since these predictions are done based on information from a traffic information center, data will have low granularity (only three levels: "normal," "congestion," and "stacked"). Recently, probe car data is used as complementary data to the data from the information center.[21] Real-time data sent from the probe car is used to improve the accuracy of the prediction.

The traffic prediction method using pheromone which we have introduced does not require past traffic data and much processing cost, and it works with a relatively simple mechanism.

7 Conclusion and Future Works

This paper has proposed a traffic prediction method that employs the pheromone mechanism. Cars are regarded as agents that deposit pheromone towards places. The pheromone would then evaporate and propagate according to a modified version of the state transition model by Brueckner. A car would be able to predict traffic on the road ahead from the information provided by preceding cars. According to the results of the experiments that used real traffic data, we confirmed the applicability of our method towards short-term traffic prediction. Furthermore, we also confirmed optimal parameter settings through sensitivity analysis. Future studies would include revealing the relationship between other parameters and accuracy. Relationship between the optimal parameter and real data is also important in order to apply this method to the real world. Application towards congestion-aware shortest route selection might be improved by introducing of negotiation algorithms since if all cars use the same algorithm, the optimal route may become congested. Agent-based approach for shortest route search will realize global optimization of traffic system.

References

1. Traffic.com, http://www.traffic.com/
2. VICS(Vehicle Information and Communication System) Center, `http://www.vics.or.jp/`
3. E.Chung,Classification of Traffic Pattern, Proc. of the 11th World Congress on ITS,2003.
4. Honda Internavi system, http://premium-club.jp/PR/technology/tech3.html/
5. R. Yasdi. Prediction of Road Traffic using a Neural Network Approach, Neural Comput Applic (1999) Springer-Verlag London Limited,1999.
6. C. Zhang, S. Sun, G. Yu. Short-Term Traffic Flow Forecasting Using Expanded Bayesian Network for Incomplete Data, Lecture Notes in Computer Science Volume 3174,2004.
7. P. Karlson, M. Luscher. Pheromones': a new term for a class of biologically active substances, Nature,1959.
8. M. Dorigo, L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation, 1(1):53–66, 1997.
9. J. A.Sauter, R. Mattews, H.Van Dyke Parunak, S. Brueckner, Evolving Adaptive Pheromone Path Planning Mechanisms, Proc. First International Conference on Autonomous Agents and Multi-Agent System(AAMAS'2002), 2002
10. H.Van Dyke Parunak, Sven Brueckner, John Sauter, Jeff Prsdamer. Mechanisms and Military Applications for Synthetic Pheromones, Proceedings of Workshop on Autonomy Oriented Computation, Agents 2001,2001.
11. M. Dorigo,G. D. Caro,L. M. Gambardella, Ant Algorithms for Discrete Optimization, Arti.cial Life, MIT Press, 1999.
12. S. Brueckner. Return from the Ant: Synthetic Ecosystems for Manufacturing Control. Thesis at Humboldt University Department of Computer Science,2000.
13. K. Kaneko. Period-doubling of Kink-antikink Patterns, Quasi-periodicity in Antiferro-like Structures and Spatial Intermittency in Coupled Map Lattices — toward a prelude to a "Field Theory of Chaos"—,Prog. Theor. Phys. 72,1984

14. Takahiro Kazama, "Method of Estimating Travel Time by Exchanging Time-Series Traffic Tables between Intersections," Proc. 11th World Congress on Intelligent Transport System, October, 2004
15. Shin Kato, Sadayuki Tsugawa, "Information Transimission over Inter-Vehicle Communications," Technical Report of IEICE, ITS2002-31, vol.102, no.474, pp.13-18, Nov, 2002
16. Masashi Saito, Mayuko Funai, Takaaki Umedu, Teruo Higashino, "Inter-Vehicle Ad-Hoc Communication Protocol for Acquiring Local Traffic Information," Proc. 11th World Congress on Intelligent Transport System, October, 2004
17. R. Horiguchi, T. Yoshii, H. Akahane, M. Kuwahara, M. Katakura, H. Ozaki and T. Oguchi., A Benchmark DataSet for Validity Evaluation of Road Network Simulation Models, Proceedings of 5th World Congress on Intelligent Transport Systems, 1998.
18. M. Howard, D. Payton, R. Estkowski. Amorphous Predictive Nets, International Conference on Complex Systems (ICCS2002),2002.
19. M. Ben-Akiva, M. Bierlaire, H. Koutsopoulos, R. Mishalani. DynaMIT: a simulation-based system for traffic prediction, Paper presented at the DACCORD Short Term Forecasting Workshop February,1998.
20. E.G. Campari, G. Levi. A Realistic Simulation for Highway Traffic by the Use of Cellular Automata, ICCS2002,2002.
21. R. Jayakrishnan, SP Mattingly, MG McNally. Performance Study of SCOOT Traffic Control System with Non-ideal Detectorization: Field Operational Test in the City of Anaheim, 80th Annual Meeting of the Transportation Research Board,2001

How Bee-Like Agents Support Cultural Heritage

Martí Fàbregas, Beatriz López, and Josep Masana

University of Girona,
Campus Montilivi, edifici P4, 17071 Girona
{marti, blopez}@eia.udg.es
http://www.iiia.udg.es

Abstract. Elderly people are a great repository of knowledge, the majority of which has never been gathered by formal means. In this paper we introduce an application of multi-agent systems to support knowledge acquisition from this rich repository knowledge which is only available from elderly and experienced people. Our system provides the opportunity to complement different versions of the same knowledge produced in an extensive geographical and cultural region with the main objective of supporting Cultural Heritage. Users without much technological knowledge can search or leave information about some type of knowledge. Then, the system behaves like a swarm of bees, in this way the bee-like agents process the user contributions and the knowledge emerges from the system. Queen-like agents, honey-bee, drones and foragers have different roles inside the hive: looking for information resemblances, computing information confidence, checking the necessity of knowledge validation, and updating user's reliability. The system's feasibility has been tested on the specific area of ethnobotany, which concerns the ways in which specific societies name and classify plants.

1 Introduction

The speed at which the Internet is evolving can hide several questions related to our culture and traditions. Efforts in computer science in general, and agent technology in particular, are usually aimed at improving technology, so new capabilities are added to future systems. Most of the systems rely on formal knowledge, which is the main component of the systems. However we should be aware that most of our current traditional knowledge can disappear in the near future. We are not referring to formal forms of knowledge, but to the knowledge that only elderly people know.

As the Valencian writer Bernat Capó said: "When an elderly farmer dies it is as if a small library has burnt down" [4]. Elderly people are a great repository of knowledge the majority of which has never been gathered formally. Current technology has arrived to almost everywhere: city suburbs and social centers of any population have an access point to the net. However, the net is being used to bring information from new knowledge producers to the general public,

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 197–211, 2006.

and there is no flow in the opposite direction, that is, gathering and verifying information that people have and supplying it to the providers.

There are several examples of information gathering in specific knowledge fields. For example, in Spain, several authors have been supported by private and public local companies, for instance Joan Pellicer and his "Costumari botanic" [10]. This kind of work has been achieved by means of traditional field work, in which most people are involved in a manual process and the results are achieved in the long term. Our aim is to provide an alternative method to favor the conservation of a certain kind of information (traditional, coming from people) by means of the new technologies, and particularly, with the use of agent technology. Moreover, by gathering traditional knowledge in an automated way our system gives the opportunity to complement different versions of the same knowledge produced in an extensive geographical and cultural region.

In this paper we explain our multi-agent system En_C_Prou [1] where this traditional knowledge acquisition is performed. Our system is based on the paradigm of Swarm intelligence. Users without much technological knowledge can search or leave information about some type of knowledge. Then, the system behaves like a swarm, in such a way that bee-like agents process the user contributions and the knowledge emerges from the system. Queen-like agents, honey-bees, drones and foragers have different roles inside the hive: looking for information resemblances, computing information confidence, checking the necessity to validate knowledge, and updating user's reliability. The system's feasibility is being tested on the specific area of ethnobotany, which is concerned with the ways in which specific societies name and classify plants [10].

This paper is organized as follows. First, we introduce our approach to bee-like agents in section 2. Then we explain how the knowledge emerges from the swarm in section 3. We continue by shown our current experimental results in section 4. In section 5 we frame our system in previous works. And we end in section 6 with several conclusions and discussion.

2 Swarm-Like Multi-agent System

Swarm Intelligence is the name of a line of research with a lot of potential in the field of Artificial Intelligence [3, 2]. Swarm Intelligence refers to the capacity to solve complex problems with simple interactions. Insect colonies function in the following manner: without general supervision they undertake distributed work where each individual is concerned with a part of the work. From the work of all the individuals the solutions to complex problems emerge.

Most researchers have adopted the metaphor of the colony's social behavior to build artificial multi-agent systems. Thus complex behaviors are supported by simple agents, either in mass, time or scope [16]. Multi-agent systems based on swarm intelligence are self-organized systems that require interactions between agents (insects): direct interactions and indirect interactions. Direct interactions are agent-to-agent interactions, while indirect interactions are agent-

to-environment interactions, usually known as stigmergy [5, 2]. Two individuals interact indirectly when one of them modifies the environment and the other responds to the new environment at a later time.

We have taken this self-organization approach of multiagent systems to model our traditional knowledge gathering system. This approach enables knowledge to emerge from the information provided by a myriad of different and diverse users. In particular, we adopt the model of swarms of bees to implement the En_C_Prou system, so that each individual in the hive (working bees, drones, queens, foragers and honey bees) is represented by an agent, while users are beekeepers (see Figure 1).

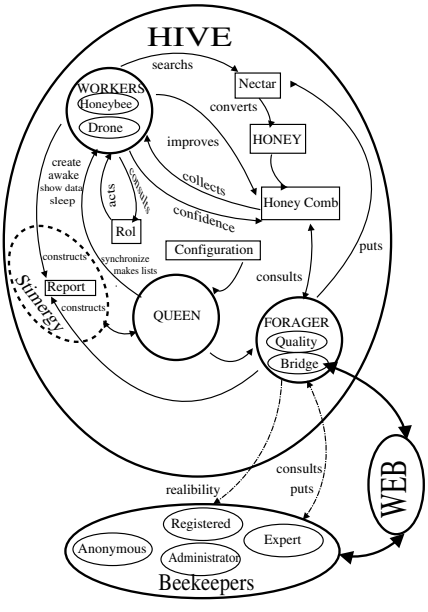


Fig. 1. Individual agents and their interaction in the EnCProu system

2.1 En_C_Prou General View

En_C_Prou is mainly concerned with the web interface and the hive. Users (beekeepers) access a web page where they can either carry out their consultations or make contributions. The user has a menu from which he/she can choose from among different actions: making a contribution to the subjects in the system, entering a new concept, creating a new attribute, querying a particular subject, validating the knowledge of the system, etc. Each action is enabled according to the user's reliability degree, which is represented by a numerical value in [0,1]. Depending on the chosen action, the user is guided step by step through other web pages in order to carry out the activity selected with the mouse. So the system has an easy interface accessible to non technological users, as elderly

people usually is[1]. The contents of the web pages are dynamic and represent the current information in the system.

For the sake of simplicity, in this paper we focus on the multi-agent system supporting the hive. The swarm processes information provided by users in the context of a traditional knowledge area, and keeps it in different honey combs. So from particular user contributions (nectar), bee-like agents produce knowledge (honey) that is kept in honey combs. At the end, users acting as beekeeper can access this knowledge.

The system always takes it for granted that beekeepers are honest when providing the information. However, it is necessary to note, that the knowledge contained in the system is not strictly academic, and therefore, it should be interpreted with the confidence that corresponds to any type of traditional knowledge. The model supposes that a user can have certain information on a subject but that he/she has not been able to relate this information with another subject with important degrees of coincidence because, for example, the terminology used by another user is different. Then, the system searches for similarities between the different information entered by the users in order to infer knowledge from all the information received. Moreover, the system keeps links between similar concepts to make cross-referencing knowledge easier. This makes accessing the system's knowledge easier for the user, saving time in revising information given by other users, favoring the incorporation of information, and allowing the final users to consult the knowledge kept in the hive quickly. This knowledge is validated a posteriori by the expert beekeepers. This validation process is performed periodically, depending on the amount of contributions provided by users and their reliability.

It is important to note, that user' reliability evolves according to the user-system interaction. Users with a low reliability can become experts if the system proves that their contributions have been useful to the swarm.

2.2 Bee-Like Agents

Agents build and maintain the honey combs, the spatiostructure regarding the information that they collect. That is, each honey comb contains knowledge related to a single concept in the application. For example, in the case of ethnobotany, each honey comb is a plant and the hive represents the complete knowledge of the application.

There are three main kinds of bees that are implemented as agents: the queen, workers and foragers. The queen is responsible for the brood, for keeping the hive with enough bees to deal with the amount of information currently available in the system. Workers, as suggested by their name, carry out the work related to converting information into knowledge. There are two main types of workers: drones and honey bees. Drones fertilize the information, in two ways:

[1] The use of the mouse, however, is not always easy for elderly people. So in the future we need to consider other kind of devices as eye-tracking devices, microphone and speakers, and so on.

1. First, they are aware of the quality and amount of information in the system, so they influence the activity of the hive in order to validate the information.
2. Second, they look for similarities between the different honey combs, so that the information can be cross-referenced (fertilization) and thus the knowledge in the hive enriched.

The honey bees are in charge of gathering information about a single concept. They get user information (nectar) and bring it up to the cells in the form of honey by using previous contributions of other users.

Finally, foragers are in charge of finding nectar sources. They have two main roles: Firstly, as bridge-foragers with the external environment, so they are aware of the current users in the system and their most recent contributions. Secondly, as quality-foragers, keeping information about the users' reliability.

2.3 Coordination

Coordination of the different agents is mainly achieved by indirect communication, that is, with the use of stigmergy mechanisms. Initially, beekeepers introduce information about a subject (plants) from the web page environment. Bridge foragers are aware of the information that the users are registering and they report such new information (nectar) on the information board of the hive.

Each worker decides its role (honey bee or drone) depending on the state of the hive. A honey bee enters the data introduced by the beekeeper (nectar) and converts it into knowledge (honey). To carry out this task, the honey bee searches in the immediate references provided by the foragers. Then it computes the confidence values of the contributions.

When the honey bee finishes its work it reports to the swarm about its performance. Then, with this report honey bees related to the new knowledge wake up. At the same time a drone finds out if there is information that needs to be verified as a consequence of the new knowledge incorporated into the system.

The drone reports the need to check the knowledge related to a single subject of the system together with the revision data. The drone is also the bee-like agent responsible for providing information about the concepts that have a high degree of coincidence (resemblance).

According to the drone's report about the need to check the knowledge, beekeepers validate the system's knowledge. External knowledge validation performed by users assures, to some extend, that the knowledge obtained by the swarm has a certain degree of quality.

A quality forager agent then revises the reliability degree of each user related to the validated knowledge. Users involved in the contributions that have converged on validated knowledge can increase his/her reliability; conversely, the user's reliability can also be decreased. If the reliability of a given user changes, the honey comb status depending on his/her information should be revised by the corresponding honey bees.

Reports and nectar are the stigma used for agent coordination. Occasionally, direct communication is also used between the queen and other agents. For example, when the queen creates a honey bee or drone.

To illustrate the complete system let us apply it in an ethnobotany context. In a given moment of time, the hive includes a honey comb related to the plant thyme (the concept). The knowledge stored in the honey comb is that thyme is a culinary and medical plant. The confidence of this information is 0.75 and 0.3 correspondingly. Moreover, there are some relationships with other plants (honey combs): fennel and oregano. A registered user with a 0.3 degree of reliability enters new information about thyme. In particular, the user says that the color of the thyme flower is violet. Bridge foragers bring this information as a report to the nectar repository. Then, honey bees related to thyme wake up, and convert the information "The color of the thyme flower is violet" to the knowledge "The color of the thyme flower is violet, confidence 0.3" at the honey comb. Now, a drone looks for similar flower colors in the hive and makes a report. Honey bees concerned with the report wake up and check the information contained in the thyme honey comb in order to improve its knowledge, and so on. The queen bee plays her role when the users enter information related to a new plant. Then she is in charge of creating a new honey bee that will start a new honey comb.

2.4 Hive

The hive is the database in which all the system's knowledge is stored. Each individual concept of knowledge is represented in a honey comb. Links between related concepts are established as well as links between the different honey combs. Each honey comb can have a different number of attributes. An attribute corresponds to a user contribution that is processed and transformed into knowledge, and stored in a cell of the honey comb. The initial set of attributes available to describe concepts is predefined. Each attribute can take on of the following two values: true or false. If an attribute has a true value is positive, it means that most of the users have confirmed the attribute. Otherwise, the attribute has a false value, the degree of confidence on the value is measured by the knowledge confidence on the attribute, defined in the interval [0, 1] and computed according to the knowledge emerging method explained in the following section.

Then, honey combs are modeled according to the different contributions that the users provide. Each honeycomb contains cells that collect the contributions about attributes, together with the confidence of the data contained in them (see Figure 2).

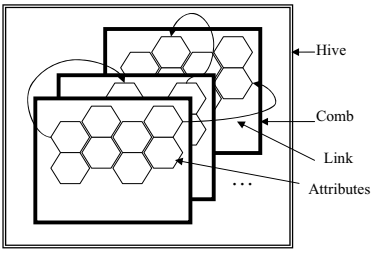


Fig. 2. Visual representation of the system's data base (hive)

For the ethnobotany application the user has several attributes predefined[2], that for the sake of management are grouped in 13 classes: application, uses, useful part, properties, doses, origin, fruit, predominant color, leafs, harvest, storage, ambient, and other features. In the properties group the following attributes are defined: Fever, liver, diuretic, anti-inflammatory, purgative, sedative, sleeping, laxative, cold, halitosis, burns, cough, repellant, etc. The thyme honeycomb can be modeled with the attributes: diuretic (yes, 0.3), cold (yes, 0.8), laxative (no, 0.7).

3 Emerging Knowledge

From the contributions made by the users the swarm acts, so we can say that the system infers knowledge from data. This process is not a centralized and sequential process, but the knowledge emerges as a consequence of the individual activities carried out by the agents in the system according to the following methods:

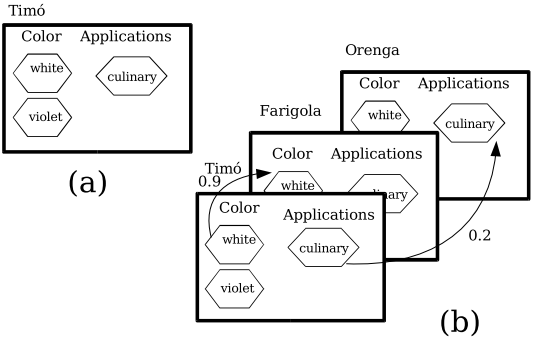


Fig. 3. (a) A collection of data. (b) The results of processing the information with En_C_Prou.

- The honey resemblance method, based on similarity measures
- The knowledge confidence method, based on certainty factors
- The degree of necessity to check information method, based on fuzzy logic
- The reliability update method, based on fuzzy logic too.

Figure 3 shows the difference between a system that gathers information and the knowledge kept in the En_C_Prou system.

In the next subsections the methods deployed by each bee-like agent are explained. Namely, the honey comb resemblance, the knowledge confidence, the degree of necessity to check information, and the reliability update methods.

3.1 Resemblance of Honeycombs

One of the most valuable functions that the system has for the user is that it compares honeycombs. In the case of ethnobotany comparing the honeycombs establishes resemblances between different plants. These honey comb comparisons

[2] More attributes can be added by users, if required.

Table 1. Examples of resemblances between different plants, P1 and P2. In columns corresponding to P1 and P2 the total number of attributes of each plant is shown.

P1	P2	#ccom	#ctot	#ceq	sim_1	sim_2	R
6	10	4	8	3	0.75	0.5	0.63
16	20	4	28	3	0.75	0.14	0.45
1	1	1	1	1	1	1	1
4	10	4	6	3	0.75	0.67	0.71

allow the drone to build links between different honey combs, so the knowledge from a honey comb is related to and supported by knowledge in other honey combs.

In artificial intelligence, resemblance methods are often called similarity methods. However, it is important to note that we are not looking for identical pieces of knowledge but for resemblances that allow the inference of knowledge to be supported.

There are many research papers by many researchers on similarity measures [17]. However, there is no universal guide on which method is most suitable. The choice of method depends a lot on the application domain.

Thus, for the particular case of ethnobotany, it is very complicated to find a single similarity criterion between plants. For one person, a plant A can be similar to a plant B because B performs a similar function, while for another person, the resemblance is due to the habitat that both plants share. The field of ethnobotany is then a clear example for understanding the implicit complications that the treatment of traditional knowledge can entail. Finally, after several studies, we have determined that two plants are similar according to the attributes that they share. This has been formalized in the following methodology.

Given two plants P1 and P2, the following measures are performed: the number of common attributes (#ccom), the union of attributes (#ctot), and the number of attributes with the same value in both plants (#ceq). Then, the resemblance between P1 and P2 is determined as follow:

$$R(P1, P2) = \alpha * sim_y 1(P1, P2) + \beta * sim_2(P1, P2) \quad (1)$$

where $sim_1(P1, P2) = \frac{\#ceq}{\#ccom}$ and $sim_2(P1, P2) = \frac{\#ccom}{\#ctot}$. In particular, we have experimentally chosen $\alpha = \beta = 0.5$. Table 1 shows an example of the application of the measure.

3.2 Knowledge Confidence

The information that remains as knowledge in the hive is the combination of all the contributions given by the users. There are different aggregation measures that are useful for this purpose [15], as well as other evidence combination methods as the Mycin certainty factors method [13]. We have experimentally tested some of them, and we have chosen the latter. The Mycin certainty factors combined by a sorting criteria have provided the best results (see section 4 and [14] for further details).

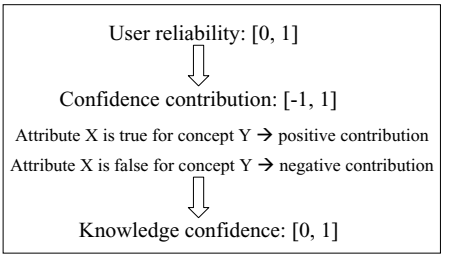


Fig. 4. Schema of the process from information to knowledge

Given a set contributions for the same attribute, $a_1, ..., a_n$, coming from different users $u_1, ..., u_n$, the procedure we propose to compute the knowledge confidence on the attribute is the following (see figure 4):

1. For each contribution a_i, the confidence contribution of a single user, c_i, is computed. This confidence contribution is defined in the [-1,1] interval and is computed as follows:
   - The absolute value corresponds to the user reliability.
   - The sign of the confidence contribution is positive, if the user confirms the attribute. Otherwise is negative.
2. Contributions are sorted according to their absolute value, getting an ordered list of contributions: $c'_1, ...c'_n$.
3. The certainty factor combination formulas are then applied to obtain the knowledge confidence of the attribute
   (a) Let be kc the knowledge confidence and set to $kc = c'_1$.
   (b) For each $i \in 2, ..., n$
      i. If $kc, c'_i > 0$, then $kc = kc + c'_i - kc * c'_i$
      ii. If $kc, c'_i < 0$, then $kc = kc + c'_i + kc * c'_i$
      iii. If $kc > 0$, $c'_i < 0$ or $kc < 0$, $c'_i > 0$): $kc = kc + c'_i$
4. The value of the the attribute is computed according to the sign of the resulting confidence as follows:
   - If the resulting confidence is positive, then the attribute value is "yes"
   - Otherwise, the attribute value is "no".

Note, that this method is not incremental. All the data is stored in the system and each time a new contribution arrives, all the steps are applied.

To illustrate the method with an example, let's suppose that the following contributions are available for the laurel subject:

- User 1 (reliability 0.1): Aromatic, yes
- User 2 (reliability 0.8): Aromatic, yes
- User 3 (reliability 0.3): Aromatic, no

After applying the first step, the the following confidence values are obtained: 0.1, 0.8, -0.3. Note that contributions with a positive value are confirming the

attribute, while contributions with a negative value are denying it. After sorting the contributions (step 2) the resulting order is: 0.1, -0.3, 0.8. Then, the first combination is carried out between 0.1 and -0. 3. Since both contributions are divergent it is necessary to apply case (c), the outcome of which is -0. 2. The next step is to combine this result -0. 2 with the 0.8 evidence. These contributions are also divergent, case (c) is applied again, obtaining the final confidence value of 0.6 for the Aromatic attribute of the Laurel concept. As the reader can observe, the contribution provided by the user with highest reliability (0.8) is the one that has the most influence in the final result. Finally, in step 4 the value of the attribute Aromatic is set to "yes" (with knowledge confidence 0.6).

3.3 The Degree of Necessity to Check Information

In the previous subsection, we described how honey bee agents decide about the confidence of an attribute from the contributions of the users. However, confidence of an attribute also depends on the number of contributions. That is, it is not the same to have a confidence value of 0.4 from 2 contributions as having a confidence value of 0.4 from 40 contributions. It is natural to think that in the second case it is necessary that an expert verifies the information. Two contributions are very few, and it is not necessary to burden a beekeeper with validating this knowledge. However, with more than 40 contributions we can say that there is a clear diversity of opinion and the approval of an expert is very necessary. Drones are responsible for monitoring this condition in relation to the knowledge stored in the hive and computing the degree of necessity to validate the knowledge.

Table 2. Fuzzy rules for determining the necessity of information checking

If KC is minimum and NC is very few, then checking degree is null
If KC is minimum and NC is few, then checking degree is normal
If KC is minimum and NC is average, then checking degree is high
If KC is minimum and NC is many, then checking degree is high

The need to check is an imprecise concept, there is a "high" need to check, or a "low" need. Therefore fuzzy logic allows us to model this type of knowledge [9]. We have defined a fuzzy decision system in which the degree of necessity to check the information is evaluated from two variables, the knowledge confidence of the attribute and the number of contributions. The variables have been modeled with the following labels:

- Knowledge confidence (KC): minimum, low, normal and high.
- Number of contributions (NC): very few, few, average, many
- Checking_degree (CD): null, low, normal, high.

Some rules of the system can be found in table 2. Drone-like agents apply these rules in order to determine if the intervention of an expert is necessary and reports to the swarm with the results.

3.4 User's Reliability Update

Confidence of the knowledge stored in the hive depends a lot on the users who interact with the system, all of whom have different backgrounds and thus different reliability degrees. For this reason, a mechanism to control and adapt the users' reliability according to their contributions is required. In particular, a user providing correct contributions should be rewarded by increasing their reliability; conversely, when a user provides incorrect contributions, their reliability should be decreased.

Defining correct contributions is somewhat difficult in terms of our application domain, that of traditional knowledge, in which the knowledge is never absolutely certain. However, we have defined this term on the basis of the opinion of the majority of users. Then, contributions are considered correct if they coincide with the current value of the corresponding knowledge to which they are contributing.

Then, a single user can be involved in different contributions, with different correctness values. Sometimes contributions are correct, sometimes they are not. So, to some extend, the user has shown a certain degree of coincidence between his/her contributions and the knowledge stored in the system. We model the degree of coincidence as follows:

$$hit\_degree = \frac{hit\_contributions}{new\_contributions} \tag{2}$$

Where hit_contributions is the number of correct contributions, while new_ contributions is the total number of contributions that the user has provided since the last reliability update. The hit_degree is 1 when all the user's contributions coincide with the rest of the users, however, this is not a typical situation and normally the hit_degree is under 1.

Then, the hit_degree of the user is compared with the total number of new contributions to the system, and a change in the reliability degree is computed. This certainty degree is once again an imprecise measure and we have used fuzzy logic to model it. On this occasion, input variables are hit degrees and system_contributions, while the outcome of the fuzzy system is the change_degree.

- Hit_degree: minimum, low, normal, high
- System_contributions: very few, few, average, many.
- Changing_degree: decrease, no_touch, increase.

Quality forager-like agents are responsible for updating the user's reliability according to the results of the fuzzy system.

4 Experiments and Results

The system En_C_Prou has been implemented in JADE, deploying Apache for the web service, using PHP and MYSQL for dealing with the database. In the user interface, there us a left menu that allows the user to query some information, to upload new information, and so on. If the user chooses "query", then the system shows all the information regarding a subject. When the user action is "upload",

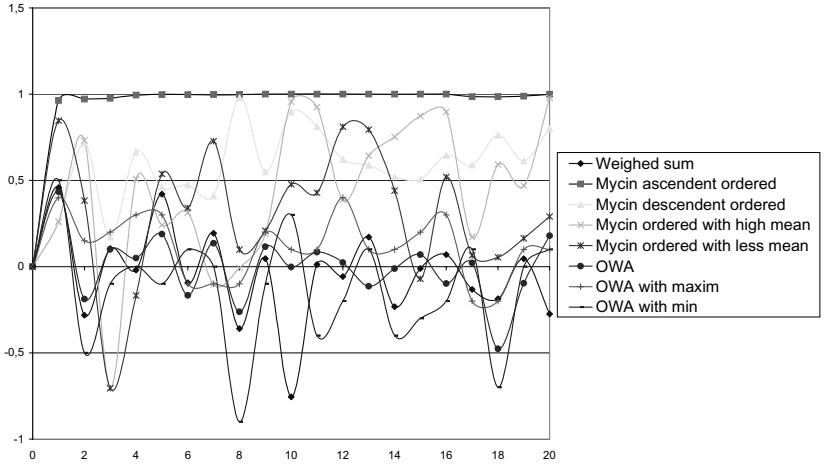


Fig. 5. Evolution dynamics of the confidence values

then the system prompts the user with the current attribute list, the hit can link with either existing or new concepts. The current prototype is accessible at the following URL: http://xixi.udg.es, it works in an experimental way. Current users are volunteers of our lab who are external to the system development.

With regards to the different methods proposed, we first studied the evolution dynamics of the confidence values of the knowledge based on system simulations. Experiments were carried out with different numbers of contributions in the same time instant, rating from 0 to 6, different confidence contributions (from -0.9 to 0.9), and up to 20 evolving times. Figure 5 shows the behavior of the system for different combination methods: weighted sum, Mycin with different orders, and the ordered weighted averaged operator (OWA). It is possible to see that for the Mycin certainty factors with ascending order (the method explained in this paper) the confidence remains stable when the user's reliability is high (around 0.9).

Second, we analyzed the method proposed to determined the degree of necessity to check the information. As shown in Figure 6 (left) the system sustains a balanced behavior regarding the amount of user's contributions and the knowledge confidence on the attributes.

Finally, the reliability updating method has also been analyzed experimentally with different hit degrees and contribution amounts (see Figure 6 right). The system has also shown balanced behavior depending on the system contributions of the user and the amount of contributions in the system. A user with a high hit degree and many contributions is likely to have his/her reliability increased. Conversely, the reliability of a user with a low hit degree is seen as defective, and so will have his/her reliability lowered.

Most of the experiments performed, however, are based on laboratory contributions and simulations. In order to get more appropriate conclusions we need to check En_C_Prou with a lot more users. Particularly, we are in contact with a public institution that is responsible of the public libraries around the local area.

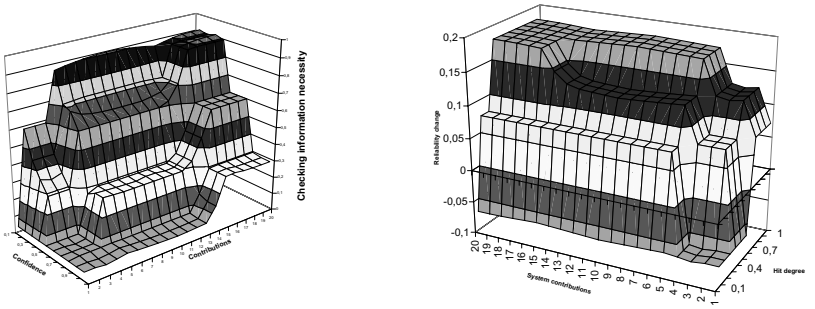


Fig. 6. Left: Results on the checking information necessity degree for several user's contributions. Right: Results on the reliability change analysis.

Our purpose is to use his infrastructure to deliver the system, so that elderly people can use them.

5 Related Work

There are several previous works related to biologically inspired multi-agent systems, and particularly regarding swarm intelligence. Parunak in [16] shows interesting properties of ant-like agents, as for example the idea of keeping agents small. Bonabeau, Dorigo and Thereaulaz [2] is also a good reference for applications based on such kind of swarm inspired systems. There is also an increasing interest on applying such kind of systems to deploy industrial applications. [6] and [12] are good examples about using swarm technology to modelling factory operations subjected to dynamic environment characteristics. Particularly, Hadeli and their colleagues [6] show how swarm intelligence provides reliability and reactivity to the systems without the need of a central coordination mechanism.

Regarding the specific domain of information gathering, [7] has also used the paradigm of swarm intelligence to explore and exploite web information. Such kind of exploration techniques have also been used to automatically build ontologies, as for example in [11]. We think that such kind of approaches are complementary to ours. So we are not exploring but analyzing the information that users are uploaded to our web with a given purpose. The work of Agassounon, regarding optimizating information retrieval can also be clustered in this kind of exploratory facet of information that differs from our approach.

A more closer approach is the one developed by Yang and Kamel [18]. They propose the use of different swarms so that each swarm learn a different cluster, and the ensemble of all of them results in a more efficient clustering. We believe that such approach can be useful in our system in a future work, so we can build several swarms according to different user communities (for example, geographically distributed) and finally combine the results.

Regarding the particularly properties of swarm-like algorithms, we will like to compare here our particular implementation for Cultural Heritage, and the approaches followed by Cultural algorithms. Cultural algorithms are a specific

form of evolutionary computation that utilizes culture as a vehicle for storing relevant information that is accessible to all members of the population over the course of many generations [12, 8]. The different steps of the algorithm comprises an evaluation and voting scheme, that we can compare to our knowledge inference mechanism, in which the degree of coincidence among users is taken into account for knowledge validation. We are in some sense surprising to find such similarities with our system and Cultural algorithm, being both kinds of systems developed for so different purposes.

Finally, there is a collaborative approach in the Internet for gathering information that is called wikipedia (http://www.wikipedia.org/). This is an on-line encyclopedia written by hundreds of volunteers. The computer tool that supports it is a simple edition program called Wiki, which follows the free code style. One of the most original elements of the wiki is that entries can be commented on and broadened by other users. Everybody is free to propose new definitions of terms. The edition is totally controlled by the user. Everybody can modify the definitions without any limitation, except for the prerequisite of being registered. Our proposal differs from the wikipedies since the information entered by users can only be modified by agents acting inside the system and, consequently, by the confidence and reliability methods deployed as well as the self-organization (resemblance, ties) of the information.

6 Conclusions

Traditional knowledge is a patrimony that we cannot permit ourselves to lose. The importance of preserving the knowledge of elderly people has motivated the development of our En_C_Prou system. We have shown throughout the paper how we can model a swarm of bee-like agents that gather the information of non technologically skilled users. Each bee-like agent performs its individual task of processing the information, so finally, a repository of traditional knowledge is achieved.

In future work we propose broadening the experimentation by installing the system in more powerful equipment. Consequently, it will be necessary to evaluate the evolution of the system when applied to great volumes of data coming from out of the lab information sources.

Acknowledgments

This research project has been partially funded by the Spanish MEC project TIN2004-06354-C02-02. Thanks to Eduard Muntaner and Maria de los Llanos Tena for their contribution in the system implementation.

References

1. En_c_prou. http://xixi.udg.es/.
2. E. Bonabeau, M. Dorigo, and G. Théraulaz. Swarm intelligence. from natural to artificial systems. Oxford University Press, 1999.

3. Théraulaz G. Bonabeau, E. Swarm smarts. Scientific American, 2000.
4. Bernat Capó. Costumari valencià 2. Edicions el Bullent, 1994.
5. V.A. Cicirello and S. F Smith. Wasp-like agents for distributed factory coordination. In *Autonomous Agents and Multi-Agent Systems*, volume 8, pages 237–266, 2004.
6. K. Hadeli, P. Valckenaers, B. Saint-Germain, P. Verstraete, C. BalaZamfirescu, and H. Van Brussel. Emergent forecasting using a stigmergy approach in manufacturing coordination and control. In *Proceedings ESOA Workshop*, 2004.
7. S. Hassas. Using swarm intelligence for dynamic web content organizing. In *Proc. Swarm Intelligence*, 2003.
8. Radu Iacoban, R.G. Reynolds, and J. Brewster. Cultural swarms: Modeling the impact of culture in social interactions and problem solving. In *Proc. Swarm Intelligence*, 2003.
9. Folger T. Klir, G. A. fuzzy sets, uncertainly, and information. Prentice Hall, 1992.
10. J. Pellicer. Costumari botànic. Picanya: Edicions del Bullent SL., 2000.
11. D. Rianyo, A. Moreno, D. Isern, J. Bocio, D. Sànchez, and L. Jiménez. Knowledge exploitation from the web. In *Proc. PAKM*, 2004.
12. N. Rychtyckyj, D. Ostrowski, G. Schleis, and R.G. Reynolds. Using cultural algorithms in industry. In *Proc. Swarm Intelligence*, 2003.
13. E.H. Shortliffe. Computer based medical consultation: Mycin. Elsevier, New York, 1976.
14. M.LL. Tena. Manteniment de la fiabilitat dels usuaris i de la confianca en la informació d'una base de coneixements populars autoorganitzada a través d'un sistema multiagent. In *BsCThesis, University of Girona (In Catalan)*, 2004.
15. A. Valls Mateu. Clusdm: A multiple criteria decision making method for heterogeneus data sets. In *PhDThesis, IIIA, CSIC, Bellaterra, Spain*, 2003.
16. H. Van Dyke Parunak. Go to the ant: Engineering principles from natural multiagent systems. In *Annals of Operations Research.*, volume 75, pages 69–101, 1997.
17. D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. In *Journal of Articial Intelligence Research*, volume 6:1, pages 1–34, 1997.
18. Y. Yang and M. Kamel. Clustering ensemble using swarm intelligence. In *Proc. Swarm Intelligence*, 2003.

Sift and Sort: Climbing the Semantic Pyramid

H.V.D. Parunak, Peter Weinstein,
Paul Chiusano, and Sven Brueckner

Altarum Institute,
3520 Green Court, Suite 300, Ann Arbor, MI 48105
{van.parunak, peter.weinstein, paul.chiusano,
sven.brueckner}@altarum.org

Abstract. Information processing operations in support of intelligence analysis are of two kinds. They may sift relevant data from a larger body, thus reducing its quantity, or sort that data, thus reducing its entropy. These two classes of operation typically alternate with one another, successively shrinking and organizing the available data to make it more accessible and understandable. We term the resulting construct, the "semantic pyramid." We sketch the general structure of this construct, and illustrate two adjacent layers of it that we have implemented in the Ant CAFÉ.

1 Introduction

In moving from raw data to finished policy recommendations, intelligence analysis must address two fundamental problems. First, the volume of available data is far more than can be processed by a policymaker, and must be reduced. Second, each item of data is susceptible to multiple interpretations, and must be related to other data to increase its semantic content.

Typically, these processes alternate. Sifting removes less relevant data. That which remains is sorted to increase its semantic content. Then further sifting reduces the data even more, permitting more sophisticated sorting. Eventually, a broad base of semantically poor data becomes a much smaller collection of data with rich associated semantics.

We motivate this view of analysis processes theoretically and illustrate it in Section 2. Then we use this distinction to describe a novel method for information retrieval in Section 3. Our sorting and sifting mechanisms are inspired by techniques used by ants to sort their nests and forage for food. Section 4 offers experimental evidence for the effectiveness of our mechanisms, and Section 5 concludes.

2 The Semantic Pyramid

The alternation of sifting and sorting is driven by a fundamental constraint from complexity theory: as the intricacy of analysis increases, the volume of data must decrease to permit timely processing.

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 212 – 221, 2006.

2.1 Computational Complexity

An important branch of theoretical computer science classifies various algorithms into different complexity classes [8]. This theory offers the following insights, which are critical for information processing in support of intelligence analysis.

The amount of time (computer cycles) or space (computer memory) needed to solve a problem depends on two things: the structure of the problem (finding records in a database vs. proving a logical theorem) and the size of the problem instance (processing a database of 10^3 records vs. one of 10^9 records). As one might expect, the larger the problem instance, the more time and/or space is needed to solve it. Less obviously, the mathematical function that translates the size of a problem into the time or space needed to solve it depends on the algorithm used to solve it, and thus on the nature of the problem. If n is the size of a problem, the time required to solve the problem might scale (for example) as log(n), or n^2, or n^{10000}, or e^n, or n!, depending on the structure of the problem.

Not surprisingly, processes that we think of as nontrivial (e.g., proving logical theorems) tend to be more sensitive to the size of the problem than simpler processes (e.g., finding the string "abc" in a document). Intelligence analysis requires processes across the complete range of such complexity. Complexity theory warns us that if we want answers in a timely fashion, we must limit the size of the problems that we ask these processes to solve, and that this limitation must be more severe for the more complex processes than for the simpler ones.

2.2 Climbing the Pyramid

Intelligence analysis typically begins with voluminous low-level data, and ends with closely reasoned analyses that support policy makers. Complexity theory shows that it is probably impossible to solve many problems perfectly in a reasonable amount of time. Thus, we must be satisfied with approximate rather than perfect answers. Furthermore, even approximate methods bog down when given problems that are too large. For example, the complex logical analysis that is required at the higher levels of analysis can only be applied to a small body of data. Thus sifting, or removing irrelevant or low-priority data, is an essential step in moving from raw data to finished product.

In general, we can identify at least four levels in the process, summarized in Figure 1.

At the lowest level, filtering (a sifting operation) applies very efficient techniques to separate a relatively small proportion of incoming data as potentially interesting. Ideally, the filtering process would touch every piece of incoming data. In practice, however, it may be necessary to sample the input stream instead, with the intensity of sampling increasing in response to positive tests.

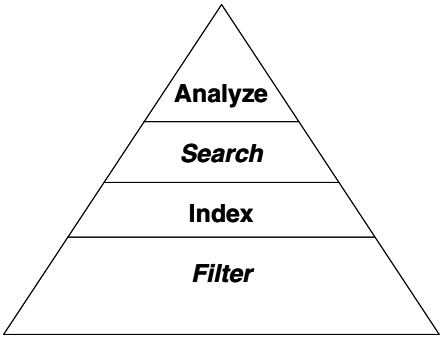


Fig. 1. The Semantic Pyramid—*Italicized* processes sift; nonitalicized processes sort

Sometimes, filter tests can be direct, for example, looking for the presence of certain terms. In other situations one must look for anomalies – deviations from expectations. (Anomaly detection does, however, require presorting: for example, to know that the data represents credit card purchases of individuals with purchasing histories).

Filtered data is then indexed, to enable it to be accessed quickly. The indices used by current internet search engines, for example, record the presence of words in each document. Semantic indexing, in comparison, strives to capture the meaning of words, phrases, sentences, and possibly other levels of meaning expressed by the data. Sorting processes like indexing do not reduce the size of the data, but make it more orderly, in order to support higher levels of processing.

Search uses the organization imposed by indexing to sift out the items of data that most closely match the analyst's requirements.

Analysis manipulates the smallest set of data: not the full volume of raw data, nor even the filtered data, but the subset that matches the analyst's current interests and requirements. It is another sorting process, detecting logical relationships among different assertions, constructing hypotheses, and verifying or refuting them. This level of analysis is the most complex computationally, and requires the previous levels of processing to reduce the amount of data to be handled.

These four levels are only illustrative. Closer analysis reveals further levels of distinct operations. Our fundamental claim is that however deep one carries this analysis, one will find an alternation of sifting and sorting, reducing the volume of data, then applying organizing processes to what remains. The structure provided by each sorting process enables more complex sifting at the next level, while the reduction in volume at each level of sifting permits application of more complex sorting algorithms at the next level.

3 Sorting and Sifting in Ant CAFÉ

We have been developing a system, called the Ant CAFÉ, that applies these concepts in doing information retrieval from massive docu-bases [16]. In this section we summarize the high-level behavior of the Ant CAFÉ, then discuss how it retrieves information of likely interest to the analyst (sifting) and how it organizes data to make this retrieval more efficient (sorting).

3.1 The Ant CAFÉ Feedback Loop

Figure 2 shows the basic concept of the Ant CAFÉ. The system has two main components. At the top of the figure, the Analyst Modeling Environment (AME) [1] maintains symbolic models at several levels.

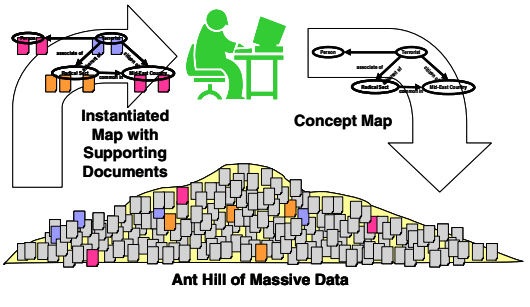


Fig. 2. Ant CAFÉ Concept

- Community models represent the interest of a group of analysts working on a common topic. They are the most general models and change the least rapidly.
- Tasking models represent a specific tasking assigned to an analyst.
- Analyst models represent the current state of an analyst's interests and hypotheses.
- Query models reflect the immediate information that an analyst wishes to access.

The AME constructs and maintains these models in response to observations of how the analyst interacts with information. Except for the query model, the analyst does not formulate these models explicitly. Currently, we represent the models as concept maps [4].

The models produced by the AME are passed to another component of the Ant CAFÉ, the Ant Hill (bottom of Figure 2). In contrast to the classic machine learning mechanisms of the AME, the Ant Hill uses techniques inspired by insect societies [12]. These techniques offer several benefits compared to more traditional algorithms:

- They are intrinsically distributed and decentralized. Thus the system can be scaled to deal with massive data by distributing it across multiple processors.
- Their processing model is dynamic, not query-driven. A query system does processing only when queried, and typically acts on a static system state. A dynamic system processes continuously, not just when a query is received, and continues to run while the system changes.
- They are anytime rather than input-process-output (IPO). An IPO system provides no information until the final answer is ready. An anytime system quickly produces approximate answers, and gives more refined information the longer the user waits.
- They are stochastic, driven by random sampling of data rather than complete enumeration. Thus users can dynamically trade accuracy against processing time by modulating the degree of coverage applied to the data.

This paper describes two such techniques. Dynamic granularity ant clustering (sorting) is inspired by the way that ants cluster the contents of their nests. Information foraging (sifting) is inspired by the pheromone mechanisms that ants use to construct paths between their nests and food sources.

3.2 Dynamic Granularity Ant Clustering

Ants sort the contents of their nests into piles of similar items [5]. As ants wander around, they maintain a short-term memory of the kinds of things they have recently encountered, and each time they encounter an object, they assess its similarity to other recently seen objects. An ant tends to pick up objects that are dissimilar to their surroundings, and to deposit objects that are similar to those in the ant's current environment. Over time, this distributed algorithm yields global clusters of high homogeneity.

Previously [13, 14], we eliminated the distinction between ants and documents, and gave each document the ability to move itself, based on its perception of its environment. Like the original algorithm, this refinement partitions the set of

documents into internally homogeneous sets, but does not provide any structure to guide subsequent retrieval operations.

Recently, we have developed a hierarchical version of this algorithm that can be distributed across multiple processors. Dynamic Granularity Ant Clustering (DGAC) repeatedly travels down its current hierarchical structure looking for nodes (sets of documents) that are internally cohesive, and moving these nodes to locations where they fit better. The resulting hierarchy reduces the complexity of subsequent retrieval.

The similarity function that drives the clustering is derived from the community model constructed by the Analyst Modeling Environment. In a previous filtering stage of processing, each document is indexed against the concepts represented in the community model. A document is considered to match a concept if it contains a morpheme that is subsumed by the concept in WordNet [7, 11]. The similarity of two documents is then measured by the standard cosine measure between their respective concept vectors.

Figure 3 illustrates the three main steps of this algorithm on a single machine. In the Figure, documents (at the bottom) are distinguished from higher-level nodes (circles), but the algorithm treats them all as "nodes." Each node maintains an estimate of its current cohesiveness in [0, 1] (the average pair-wise similarity across all documents that it currently subsumes) and a summary of its documents that it can use for comparing its similarity to other nodes (say, a vector average of its documents' concept vectors). The system also maintains a threshold $\theta \in [0,1)$, initially 0, whose function is similar to that of temperature in simulated annealing [10]. The cohesiveness of a document is 1, and its summary is its concept vector.

At any given cycle, one node is active. In Figure 3a, the current node is document H. In the "select" step, the node compares its cohesiveness c with θ. If $c <= \theta$, the node selects one of its children stochastically, favoring those that are least cohesive, and activates it. (Because document nodes have c = 1, this case never applies to them.) If $c > \theta$, the node leaves its current parent and is attached to its grandparent (A in Figure 3b). (In doing so, it is moving the entire sub-tree that descends from it.) Then it randomly chooses one of its new siblings (B in Figure 3b) and estimates whether a merger with this sibling would yield coherence greater than θ. If so, it merges with that sibling under a new node (K in Figure 3c), measures the actual similarity σ achieved, and updates θ with the Q-learning rule $\theta_{t+1} = (1 - \alpha)\theta_t + \alpha\sigma$ [15]. (α, the learning rate, is an arbitrary parameter in [0,1]. Our experiments use $\alpha = 0.02$.) Then the root becomes active, and the process repeats.

Initially, when the threshold θ is low, nodes readily climb up the tree and seek new partners. As θ increases, it becomes more difficult for nodes to relocate, leading the system to stabilize.

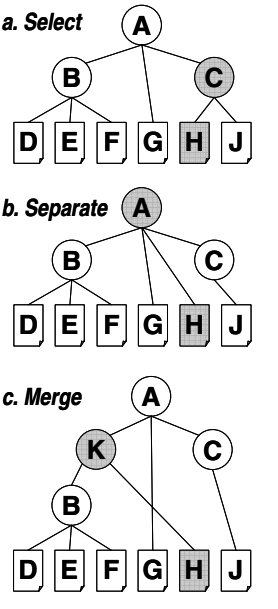


Fig. 3. Dynamic Granularity Ant Clustering

Hierarchical clustering is not new [2, 9], but the standard approaches have several limitations that this algorithm avoids:

- It does not require a centralized similarity matrix over all data items, and thus can readily be distributed across many machines. (In this case, each machine maintains its own θ, and periodically normalizes this value with those on other machines.)
- The population of data items can change dynamically as the algorithm runs, without restarting. The algorithm discovers misplaced documents and moves them dynamically. The similarity estimated when doing the merge can be less than that measured after the merge, particularly if the summary used in the estimation has been invalidated by the addition of new documents. So θ can decrease, permitting previously settled sub-trees to move.
- The similarity function used in the clustering operation (that is, the set of concepts attested in the community model) can also be changed incrementally without restarting the system. To the degree that the new function overlaps the old, the system reuses existing structure that has been constructed.
- Traditional hierarchical clustering is monotonic: once two nodes have merged into a higher-order node, they cannot separate again. This algorithm permits nodes to dissolve as well as to join, enabling the system to search continuously for the best fit to the data.
- This algorithm is stochastic, since node movements are evaluated by comparing a sample of their leaf nodes. As a result, users can trade accuracy against processing time.

Clustering runs continuously against the community model, even when there is no query from the user. It continuously organizes the underlying body of data to take account of changes in both the set of documents available and the slowly evolving community map.

3.3 Information Foraging

Ants construct efficient paths between their nests and food sources by depositing chemicals (pheromones) on the ground whenever they are carrying food, and by climbing the gradient of these chemicals whenever they are searching for food. As many ants discover food and deposit pheromones, discrete pheromone paths emerge that guide the ants' otherwise random movements toward food sources.

Dynamic Granularity Ant Clustering organizes sets of documents into hierarchical clusters. Each document is a leaf in the tree, and the similarity of documents under each node increases as one descends the tree from the root to the leaves. In information foraging, an analyst model or query model generates a stream of forager agents. Each forager agent begins at the root and descends the tree until it reaches a leaf. Then it evaluates a function that computes the relevance of the document that it has found to the higher-level query. This relevance score is deposited at the leaf, and propagates back up the tree toward the root, combining with any other relevance deposits from foragers representing the same model, and diminishing in strength with

each step. As successive foragers descend the tree, they select their path at each node stochastically by evaluating a Boltzmann-Gibbs distribution weighted by the relevance scores at each of the accessible next steps. The relevance scores function like ant pheromones, building up paths for later foragers to follow.

In general, searching for a data item in a population of size N requires time on the order of N (look at each item in turn until you find the one you want). If the items can be ordered linearly by their relevance, one can do the search in time logarithmic in N, but a single linear order is not realistic for most documents of interest to intelligence analysis. In our foraging system, the maximum length of the relevance path to documents of interest is the depth of the tree, which is logarithmic in the total number of documents (where the base of the logarithm is the mean branching factor at each node). Thus we achieve searching efficiencies comparable to those for linearly ordered data, even for data that cannot be usefully constrained to a total order.

The cost of this efficiency is constructing the hierarchical clustering of documents in the first place. Thus the sifting process of information foraging requires a preceding sorting process of constructing a hierarchical structure that will support the foraging. The Dynamic Granularity Ant Clustering algorithm described in Section 3.1 provides this structure.

4 Experimental Results

We have explored the potential of information foraging and hierarchical ant clustering with two sets of experiments. Both are based on a static set of 500 documents drawn from the CNS database [3] and hierarchically clustered by similarity. In principle, this kind of algorithm scales very well. We have used more mature ant clustering techniques to organize a dynamically changing population of more than 100,000 documents on a cluster of 16 processors [13, 14], but those techniques produced only a partitioning of the data, not a hierarchy, and so do not support the interactive experiments with foraging that we report here. We are currently preparing an experiment that will demonstrate a refinement of the hierarchical clustering algorithm to a document population on the order of 10^4.

4.1 Effectiveness of Foraging

To set up the experiment, we compute each document's relevance to the query and normalize this value so that the sum of relevance is 1. We compare the foraging process with a random process. While this benchmark may seem an unfair comparison, it is quite widely accepted in the pattern recognition community in the context of ROC (receiver operating characteristic) curves, a performance visualization to which our analysis is closely related [6]. At each step, the process under study selects a document, adds its relevance to the process's current total relevance, and then removes it from the population. Thus both processes begin with a relevance of 0, and end with a relevance of 1 (having seen all documents). In the random process, the documents are selected at random. In the foraging process, each successive forager takes advantage of the relevance paths laid down by previous foragers.

Figure 4 compares how the two processes accumulate relevance. Initially, before relevance paths develop, the processes are comparable. Then the rate of accumulating relevance accelerates dramatically for the foraging process, as subsequent foragers take advantage of the paths developed by previous ones.

4.2 Effectiveness of Clustering

To evaluate our clustering method, we use an artificial retrieval benchmark, called "seekers," that can readily be applied periodically during clustering. For each document in the population, we prepare 9 seekers with the same concept vector. Seekers follow the clustering hierarchy from the root to a leaf, following the branch at each level with the greatest similarity to a sample of leaves. The seeker metric is the proportion of seekers that terminate their search at a document that has a similarity of at least 0.95 with their concept vector.

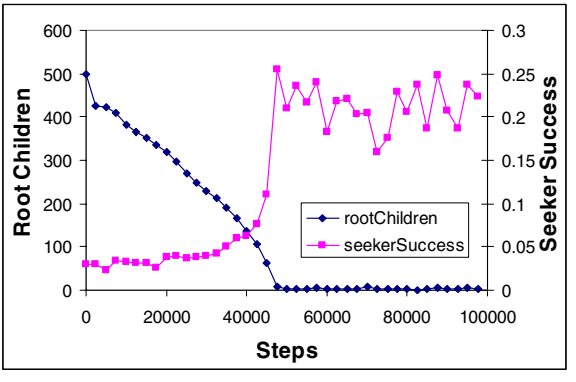


Fig. 4. Progress of Clustering

Figure 5 shows how seeker success (square symbols) increases as clustering progresses. (In this example, the collection of documents and the concept vector are static.) Like many stochastic processes with emergent behavior, the system exhibits a phase shift. Before 45000 cycles, there is little apparent improvement in seeker success. After 48000 cycles, seeker success jumps dramatically. The second series in the plot (lozenze symbols) shows the number of immediate descendants of the root node in the emergent hierarchy. At step 0, every document is a child of the root, but as the algorithm progresses, more stru-cture develops below the root, with a corresponding dec-rease in the number of immediate children of the root. This decrease cont-inues until 48000 cycles, when the multiplicity of the root levels off at an average of 3.7.

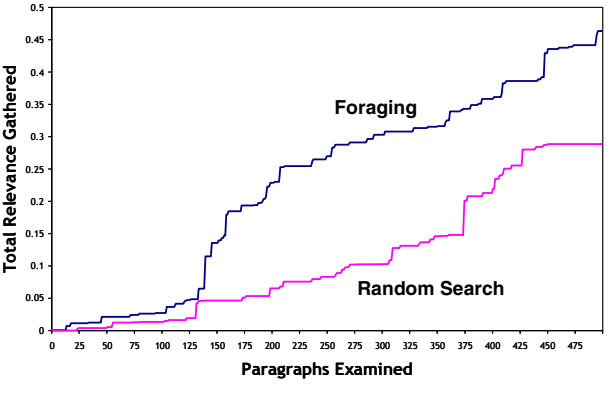


Fig. 5. Effectiveness of Foraging in Hierarchically Clustered Data

The stabilization of the number of root children, and the cor-

responding increase in the seeker success, show that the data hierarchy has reached a stable configuration that captures the intrinsic structure of the data. Because the clustering process is dynamic rather than query-driven, it can achieve this stable condition in the background, so that the hierarchy is immediately available for foragers to service a query. From an engineering perspective, the speed of this convergence is a critical design parameter that can help determine the number of parallel processors that are needed to accommodate a given rate of change of the document population or the community model.

The asymptotic success rate of 25% for the seekers reflects the stochastic nature of the algorithm, and the convergence imposed on the system by the linear learning rate of the threshold θ. The stochastic character of the algorithm is what permits the system to accommodate dynamic data, and one is willing to tolerate an imperfect hierarchy as the price of that flexibility. In addition, we are currently testing a more sophisticated learning mechanism for governing convergence, one that we will expect to yield higher asymptotic rates of seeker success.

5 Conclusion

Successful intelligence analysis requires both sifting masses of available data to reduce their quantity, and sorting them to increase their semantic content. These processes often alternate with one another. Sifting reduces the volume of data so that sorting can apply increasingly complex analyses within the bounds of available time, while sorting structures the data in a way that makes the next round of sifting more efficient.

This alternation is apparent in two insect-inspired processes that we are using to address the massive data problem. Experiments with these mechanisms demonstrate their promise in concurrently increasing the structure of data and reducing its volume in support of higher-level analyses.

Acknowledgements

This study was supported and monitored in part by the Advanced Research and Development Activity (ARDA) and the National Geospatial-intelligence Agency (NGA) under contract number NMA401-02-C-0020. The views, opinions, and findings contained in this report are those of the author(s) and should not be construed as an official Department of Defense position, policy, or decision, unless so designated by other official documentation.

References

[1] R. Alonso and H. Li. Model-Guided Information Discovery for Intelligence Analysis. In *Proceedings of CIKM '05*, Bremen, Germany, 2005.

[2] P. Berkhin. Survey Of Clustering Data Mining Techniques. Accrue Software, San Jose, CA, 2002. http://citeseer.ist.psu.edu/berkhin02survey.html.

[3] CNS. CNS WMD Databases. 2004. CD,

[4] J. W. Coffey, R. R. Hoffman, A. J. Cañas, and K. M. Ford. A Concept Map-Based Knowledge Modeling Approach to Expert Knowledge Sharing. In *Proceedings of IASTED International Conference on Information and Knowledge Sharing*, 2002. http://www.ihmc.us/users/acanas/Publications/IKS2002/IKS.htm.

[5] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The Dynamics of Collective Sorting: Robot-Like Ants and Ant-Like Robots. In J. A. Meyer and S. W. Wilson, Editors, *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 356-365. MIT Press, Cambridge, MA, 1991.

[6] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. HPL-2003-4, HP Laboratories, Palo Alto, CA, 2003. http://www.hpl.hp.com/techreports/2003/HPL-2003-4.pdf.

[7] C. Fellbaum, Editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. Cambridge, MA, MIT, 1998.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability*. San Francisco, CA, W.H. Freeman, 1979.

[9] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys*, 31(3), 1999.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671-80, 1983.

[11] G. A. Miller. WordNet: A Lexical Database for the English Language. 2002. Web Page, http://www.cogsci.princeton.edu/~wn/.

[12] H. V. D. Parunak. 'Go to the Ant': Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, 75:69-101, 1997. http://www.altarum.net/~vparunak/gotoant.pdf.

[13] H. V. D. Parunak, S. A. Brueckner, R. Matthews, and J. Sauter. Pheromone Learning for Self-Organizing Agents. *IEEE SMC*, 35(3 (May)):316-326, 2005. http://www.altarum.net/~vparunak/ParunakIEEE.pdf.

[14] H. V. D. Parunak, S. A. Brueckner, J. A. Sauter, and R. Matthews. Global Convergence of Local Agent Behaviors. In *Proceedings of Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS05)*, Utrecht, The Netherlands, pages 305-312, 2005. http://www.altarum.net/~vparunak/AAMAS05Converge.pdf.

[15] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, MIT Press, 1998.

[16] P. Weinstein, H. V. D. Parunak, P. Chiusano, and S. Brueckner. Agents Swarming in Semantic Spaces to Corroborate Hypotheses. In *Proceedings of AAMAS 2004*, New York, NY, pages 1488-1489, 2004. http://www.altarum.net/~vparunak/ AAMAS04Ant CAFE.pdf.

Agent-Based Control of Spatially Distributed Chemical Reactor Networks

Eric Tatara[1], Michael North[2], Cindy Hood[1], Fouad Teymour[1], and Ali Cinar[1]

[1] Illinois Institute of Technology, Chicago, IL 60616, USA
{tataeri, hood, teymour, cinar}@iit.edu
http://www.chee.iit.edu/~cinar
[2] Argonne National Lab, Argonne, IL
north@anl.gov

Abstract. Large-scale spatially distributed systems provide a unique and difficult control challenge because of their nonlinearity, spatialdistribution and generally high order. The control structure for these systems tend to be both discrete and distributed as well and contain discrete and continuous elements. A layered control structure interfaced with complex arrays of sensors and actuators provides a flexible supervision and control system that can deal with local and global challenges. An adaptive agent-based control structure is presented whereby local control objectives may be changed in order to achieve the global control objective. Information is shared through a global knowledge environment that promotes the distribution of ideas through reinforcement. The performance of the agent-based control approach is illustrated in a case study where the interaction front between two competing autocatalytic species is moved from one spatial configuration to another. The multi-agent control system is able to effectively explore the parameter space of the network and intelligently manipulate the network flow rates such that the desired spatial distribution of species is achieved.

1 Introduction

Large-scale spatially distributed systems provide a unique and difficult control challenge because of their nonlinearity, spatial distribution and generally high order. The control structure for these systems tend to be both discrete and distributed as well and contain discrete and continuous elements. A layered control structure interfaced with complex arrays of sensors and actuators provides a flexible supervision and control system that can deal with local and global challenges. Traditionally, research on control of nonlinear distributed processes has focused on distributed parameter systems involving mathematically complex model reduction and controller synthesis methodologies [1]. So-called hybrid control systems combine process dynamics and discrete control elements through the use of multiple linear models at different operating points [2]. One alternative approach is based on a hierarchical agent-based system with local and global control structures [3] that has been demonstrated on a network of interconnected continuous stirred tank reactors (CSTRs).

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 222–231, 2006.

Controlling the spatial distribution of autocatalytic species in a network of reactors requires simultaneous manipulation of interconnection flow rates. Numerical experiments suggest that individual CSTRs in networks are capable of hosting only a single dominant species, while other competing species may be present in trace quantities. Consequently, if the control objective calls for one species to be replaced with another, a nonlinear control scheme must be used.

For a single CSTR with competing autocatalytic species, the reactor residence time must first be modified such that the undesirable species is washed out of the system, and then set to an appropriate value that is favorable to the existence of the desired species [4]. This concept can be extended to systems with many reactors to effectively control the spatial distribution of autocatalytic species in the network. However, the control problem becomes complex because each CSTR has a feed and exit stream, as well as multiple interconnections to its neighbor(s). Obviously, each manipulated variable (interconnection flow rates, for example) requires an actuator and control structure, along with the appropriate number of sensors for each reactor.

One approach is to implement a nonlinear control scheme based on reduced order models. A switching technique for feedback controllers was proposed whereby a predetermined set of actuator configurations may be used to move the system from one state to another [5]. This methodology benefits from the inherent qualities of guaranteed closed-loop stability and operation within actuator constraints. Transitions between various actuator configurations is achieved through a set of switching rules. The method is only limited to the number of a priori determined set of actuator configurations and switching rules. A control objective may not be satisfied if a suitable configuration is not available. Alternatively, intelligent supervisory knowledge-based control systems have been implemented to control a distributed process with changing operating conditions in an adaptive manner [6]. A limitation of supervisory knowledge-based control and agent-based control techniques discussed below is lack of a priori determination of conditions that guarantee closed-loop stability by using well-established techniques. Instead, large number of simulations with preselected and random setpoint changes and disturbances are conducted to collect information about process behavior and identify control strategies and parameter values that have low likelihood of causing undesirable process behavior.

The operation of highly nonlinear systems like autocatalytic replicator networks may benefit from evolutionary self-organizing control because the optimal operating regime and the required control strategies may not be known a priori. Agent-based control systems provide the capability for localized and global control strategies that are both reactive in controlling disturbances and proactive in searching for better operational solutions [7]. This paper proposes an adaptive agent-based control system for a CSTR network. The performance of the agent-based control approach is illustrated in a case study where the interaction front between competing autocatalytic species is moved from one spatial configuration to another. Finding and maintaining operating states that

are both stable and efficient is an ongoing research challenge since the complex non-linear interactions between reactors often leads to undesirable or even unpredicted behavior. What is needed are adaptive control systems that can self-organize into productive patterns and self-correct in the face of unexpected deviations.

2 Agent-Based Control Framework

Multi-agent control system architectures have several properties that make them particularly attractive for use in supervising large, complex systems [8]. The first, and usually most important in critical systems, is a high level of reliability. Modularity, scalability and adaptability are also attractive features of multi-agent systems. The adaptive and self-regulatory nature of agent systems has only recently been investigated for solving control problems that are normally solved with traditional methods.

2.1 Design Process

The design procedure is a derivative of recent agent design methodologies based on the concept of the agent-services-acquaintance model [9] and the application to manufacturing control [10]. The goal of the design process is to develop an agent based control system for physically distributed industrial processes. Certain parts of the control system are generic because they are based on general concepts of industrial control system and operation of distributed processes.

Comprehensive studies of the physical process domain provide information regarding the processes' expected normal operating conditions, types of faults and disturbances that may occur, and control strategies. Additionally, the desired process operation and/or optimal conditions are expected to be known by the designers. Required agent types and roles are identified based on the requirements for controlling the physical system. After the agent model is specified, the services model can be derived. The services model describes all of the services and responsibilities provided by each agent. Two important distinctions in the agent responsibilities for process control applications are liveness responsibilities and safety responsibilities, the later of which play a critical role in operation of real-life process operations. Finally, the acquaintance model describes how the various agent types communicate with each other.

A hierarchical agent-based architecture has been recently developed for the control of spatially distributed chemical reactor networks [3]. The architecture consists of several sub-systems, each of which are highly modularized (Figure 1). At the process level, network elements such as reactors and valves interface with the higher-level agents via low-level agents. The lowest level of agents in the control system hierarchy include observation and actuation agents. Each reactor is monitored by an observation agent that is responsible for sampling data requested by other agents as well as storing the data in a history for some specified time. The interconnection flow rates are manipulated by actuation agents (not shown) that

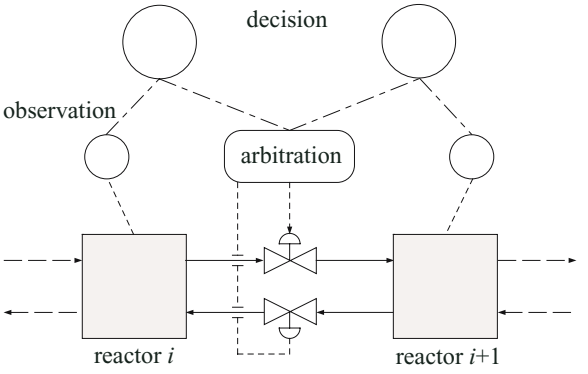


Fig. 1. Control agent architecture

receive commands from control arbitration agents. Arbitration agents may be local (focusing on the operation of a few adjacent reactors) or global (focusing on arbitration for all reactors).

The next layer in the control hierarchy is the local decision layer. Local decision agents are responsible for monitoring control functions and proactively improving the overall performance of the network based on the control objectives of the individual agents and the reactor network as a whole. Due to the number of control responsibilities of decision agents, each agent may use sub-agents. For example, the local control decision agent requires information regarding the state of the process. A sub-agent is therefore tasked with checking the state of a reactor or one of its neighbors.

During network operation, local decision agents attempt to satisfy their individual control objectives, for example to change the dominant autocatalytic species concentration from one species to another. However, in many cases, a decision agent may never fully reach its desired objective due to potential conflicts with other agents' control objectives. If an agent desires to modify the interconnection flow rate between a reactor and its neighbor to meet a control objective, the adjacent reactor will be affected as well. Naturally, disputes will arise as to the value of the interconnection flow rates between neighboring reactors.

Arbitration agents serve as a communication channel between decision agents as well as a means to resolve disputes between agents. The arbitration agents receive requested operational procedures from the local decision agents and then present a solution to them. For example, a decision agent must modify the residence time of the reactor to flush out an undesirable species by manipulating the interconnection flow rate. The decision agent sends a set of acceptable values for the manipulated variables to the arbitration agent which then tries to match the desired operational values for neighboring decision agents. Finally, supervision agents function as the top layer in the control system hierarchy. This layer is responsible for setting the desired global operating conditions for the network, for example the overall spatial distribution of autocatalytic species.

2.2 Global Knowledge Environment

Considering the nonlinearity of reactor networks, it is difficult to predict how the behavior of the system changes when the system parameters are manipulated. Consequently, one cannot easily predict how to change operating conditions of the network by manipulating the flow rates, nor what the localized operating conditions should in fact be, in order to satisfy a global objective. Several methods can be used to guide the decision agents in planning their control strategies including dynamic exploration of the parameter space, rule-based heuristic models, or first-principles based models.

Decision agents may exploit a model of the reactor network, say by knowing the precise location of stable branches and oscillatory regimes. For example, the complete bifurcation structure for a particular system would be quite valuable to decision agents in formulating a control strategy. However, since the number of steady states increases exponentially with the size of the system, this method is not scalable to larger systems. An effective solution is provided via rule-based heuristic models coupled with dynamic exploration techniques.

A heuristic model of the reactor network consists of rules that describe how the manipulated variables affect the system behavior. For example, the stable steady states occupy only certain portions of the diagram, or only certain spatial patterns of species concentration are stable. This information is provided to the decision agents in the form of rules to guide their control actions. Furthermore, the decision agents are allowed to "probe" the system by making small, temporary changes to the manipulated variables and observing the resulting system behavior. This dynamic exploration provides additional flexibility to the decision agents when the generalized heuristic model cannot explain system behavior.

Although information may be exchanged between agents via arbitrators, these interactions are local and inherently limit the amount and quality that can propagate through the agent control structures. The global knowledge representation (Figure 2) serves as an environment for indirect communication between agents. This concept builds upon the hierarchical structuring of the control system in Figure 1 by adding a mechanism for communication and reinforcement of ideas. The information in the knowledge space is divided into categories including local control objectives, control heuristics, and data-based models.

Information exchange occurs indirectly between agents because agents asynchronously read/write information to/from the knowledge space. For example, a particular agent may discover a local control strategy that works particularly well in meeting an objective set by a supervisor. This strategy is cataloged in the knowledge space by the originating agent. Other agents may read this strategy from the knowledge space and implement it to satisfy their particular control objective. The value of the strategy is then rated by the agents that adopt this new strategy such that its value relative to others is promoted. Similarly, outdated information in the knowledge space continuously decreases in value and eventually may be deleted from the knowledge space.

Although the stability of the agent dynamics cannot be guaranteed for every scenario, this methodology helps to reduce or prevent the emergence of dysfunc-

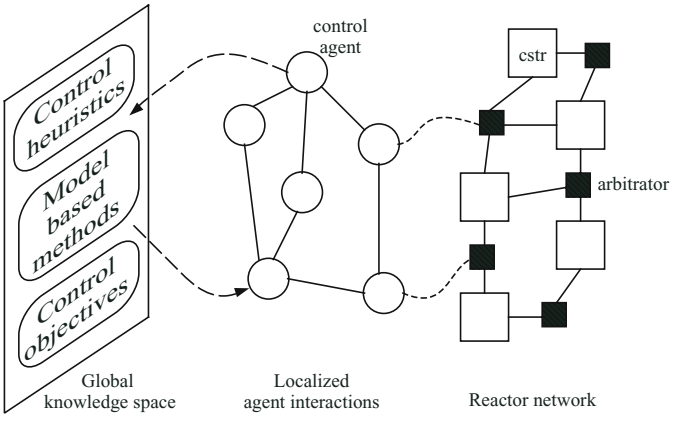


Fig. 2. Coupling of control agents with global knowledge space

tional agent dynamics by reinforcing "good" agent behavior, while punishing undesirable agent behavior. Furthermore, the agent system has been designed with the assumption that the agents' decision delays are small compared to the time scale of the physical process. This assumption generally holds for chemical processes in which operating changes are introduced infrequently and process dynamics are represented with time scales of tens of minutes or even hours. Traditional controllers are normally used in the event of very rapid localized dynamics and, while the agents may modify the setpoints of such controllers, the time-critical (first response) control actions are strictly outside the domain of the higher decision making agents.

3 Network Model

A network of interconnected isothermal CSTRs is modeled by specifying the material balance for each individual reactor in the network. The cubic autocatalytic reaction for N autocatalytic species is

$$R + 2P_n \xrightarrow{k_n} 3P_n \tag{1}$$

$$P_n \xrightarrow{k_{dn}} D \tag{2}$$

where R is the resource, P_n is the n^{th} species, and D is a dead (inert) species. Reaction rate constants k_n and k_{dn} characterize the growth and death rates of the n^{th} species.

Since the complexity of the system grows geometrically with the number of species and the number of reactors in the system [11], for networks larger than three reactors with two or more species, analytical solutions become practically intractable, although a single trivial steady state ($r = 1$, $p_n = 0$) exists for every combination of model parameter values. This trivial steady state is always stable and will always pose a threat to control efforts as it represents total extinction of the autocatalytic species in the system.

In the control examples detailed in the next section, the interconnection flow rates are used as manipulated variables. The system is operated with constant volume and constraint equations are formulated on the reactor flow rates to ensure that material is conserved. The reactor inputs include the reactor feed and the interconnection flows from neighboring reactors. Outflow rates from each reactor include the interaction with neighboring reactors as well as the drain. The constraints include a lower bound such that all flow rates are non-negative and an upper bound that ensures the equality of total inflow and outflow.

4 Case Study: Product Grade Transition in a Chemical Reactor Network

The performance of the agent-based control architecture is demonstrated in a case study to control the distribution of autocatalytic species in a network of 49 (7x7 grid) reactors hosting three autocatalytic species using the interaction flow rates as the manipulated variables. The spatial distribution of species determines the overall product composition that exits the network. By manipulating the spatial configuration of the species in the network, one can produce multiple grades of product by simply reconfiguring the network's connectivity.

The species that populate the reactor network have similar growth and death rates, such that one species does not have an unfair advantage over the others. When a particular interaction flow rate is manipulated, the outflows of the corresponding reactors are adjusted using constraint equations on the flow rates, thus keeping the volume of each reactor constant.

Moving the system from different initial spatial configurations to different final configurations requires individualized (and potentially unknown) control strategies for each region of the network. Although it is a difficult control problem for conventional controllers, an agent-based architecture can achieve spatial state transitions as illustrated below. The agent-based controller discretizes the reactor state space by identifying the dominant species in the reactor. Each reactor will only have one dominant species and several residual species that exist solely because of material exchange from neighboring reactors. The dominant species has generally an order of magnitude greater concentration than the other species.

4.1 RePast Implementation

The reactor network model and agent-based control system is implemented with the open source agent modeling and simulation environment RePast [12]. The RePast toolkit is a java-based framework for agent simulation and provides features such as an event scheduler and visualization tools.

The control agents created with RePast interact with virtual representations of the physical reactor network. The virtual network objects map the states of the physical system to objects that can be manipulated by the control objects. The interface between the physical network to the agent environment can take the form of a data acquisition system in the case of a real world control application, or in this case, a simulation of a chemical reactor network. The ordinary differential

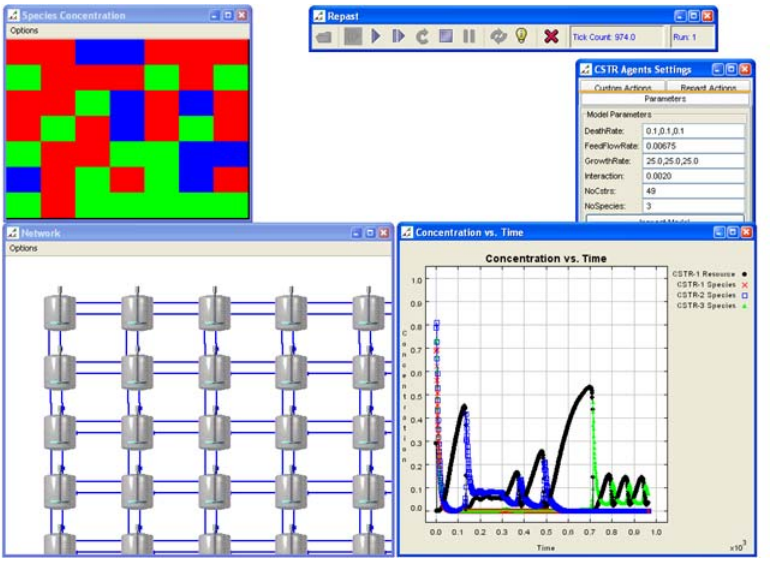
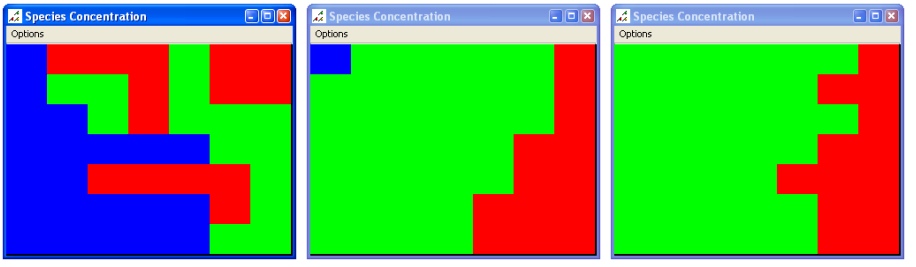


Fig. 3. Software implementation of agent control system with RePast

equations that describe the autocatalytic reactions in each CSTR are solved numerically using the CVODE solver [13]. The solver code is written in C and linked with RePast via the Java Native Interface (JNI).

Figure 3 illustrates the graphical user interface implemented in RePast. The upper left quadrant of Figure 3 contains a 2D map of the dominant species in each reactor, with red (dark gray) representing species 1, blue (black) species 2, and green (light gray) species 3. The map shows the distribution of each species in the network as is updated whenever the state of one of the reactors change, ie the dominant species switches from one species to another. The upper right quadrant of Figure 3 shows the control panels for starting/stopping the system as well as changing any of the available system parameters. The bottom left quadrant of Figure 3 contains the graphical representation of the physical reactor configuration, with respective icons corresponding to reactors and interconnecting pipes. Finally, the bottom right quadrant of Figure 3 shows a sample time series plot for the concentrations in an arbitrarily chosen reactor.

The spatial distribution of autocatalytic species in Figure 3 shows the open loop behavior of the system following the temporal evolution from a randomly generated set of initial conditions. Figure 4 shows the resulting changes in the species concentration profiles when the control system is tasked with creating different product grades starting from the initial condition shown in Figure 3. The desired product grade is specified by setting the percentage of the network average species concentrations (1,2,3) = (0.25,0.02,0.73) and the system goes through a series of transitionary states (Figure 4a) before successfully settling on the grade setpoint (Figure 4b). A second grade transition (0.3,0,0.7) demonstration is also successfully executed by the control system (Figure 4c).



(a) (b) (c)

Fig. 4. Evolution of 2D spatial concentration profile of the dominant species in each reactor, with red (dark gray) representing species 1, blue (black) species 2, and green (light gray) species 3

The specified concentration profiles only constrain the desired network average, thereby providing freedom to the agent-based control system to find any non-unique solution that satisfies the objective.

5 Conclusions

An adaptable, intelligent agent-based control system has been implemented to control the product grade transitions via spatial distribution of autocatalytic species in a reactor network by manipulating the interconnection flow rates. This methodology has been proposed as a real-time alternative to traditional nonlinear control schemes involving predetermined controller configurations or computationally expensive optimization techniques. The multi-agent control system is able to explore the parameter space of the network and intelligently manipulate the network flow rates such that the specified goal is achieved. The reactor control system presented in this paper is a self-organizing application (SOA) that can dynamically and automatically modify its configuration to maintain its stability and productivity. The control system achieves this goal using interlocking sets of simple local behaviors. The use of a global knowledge environment as a method for indirect communication between agents provides a means for the reinforcement of information. Furthermore, this system ontology permits the agents to organize their solutions independently of a top-level supervisor as long as the global objective is satisfied. This agent structure has direct correlations to other artificial and natural systems with dissipative information fields such as information sharing in ant colonies [14], [15] where information is reinforced in a self-catalyzed reinforcement process.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. CTS-0325378 of the ITR program.

References

1. Christofides, P.: Control of nonlinear distributed process systems: Recent developments and challenges. AIChE Journal **47** (2001) 514–518
2. Morari, M., Baotic, M., Borrelli, F.: Hybrid systems modeling and control. European Journal of Control **9** (2003) 177–189
3. Tatara, E., Birol, I., Teymour, F., Cinar, A.: Agent-based control of autocatalytic replicators in networks of reactors. Computers & Chemical Engineering **29** (2005) 807–815
4. Chaivorapoj, W., Birol, I., Cinar, A., Teymour, F.: Feedback control of a continuous-flow stirred tank reactor with competing autocatalators. Industrial and Engineering Chemistry Research **42** (2003) 3765–3785
5. El-Farra, N., Christofides, P.: Coordinating feedback and switching for control of spatially distributed processes. Computers and Chemical Engineering **28** (2004) 111–128
6. Kendra, S., Basila, M., Cinar, A.: Intelligent process control with supervisory knowledge-based systems. IEEE Control Systems **14** (1994) 37–47
7. Jennings, N., Bussmann, S.: Agent-based control systems. IEEE Control Systems **23** (2003) 61–73
8. Lesser, V.: Cooperative multiagent systems: A personal view of the state of the art. IEEE Transactions on Knowledge and Data Engineering **11** (1999) 133–142
9. Wooldridge, M., Jennings, N., Kinny, D.: A methodology for agent-oriented analysis and design. In: Proceedings of the Third International Conference on Autonomous Agents (Agents'99), Seattle, WA (1999)
10. Brueckner, S., Wyns, J., Peeters, P., Kollingbaum, M.: Designing agents for manufacturing control. In: Proceedings of the 2nd AI & Manufacturing Research Planning Workshop, Albuquerque, NM (1998)
11. Tatara, E., Birol, I., Teymour, F., Cinar, A.: Measuring complexity in reactor networks with cubic autocatalytic reactions. Industrial and Engineering Chemistry Research **44** (2005) 2781–2791
12. Collier, N., Howe, T., , North, M.: Onward and upward: The transition to repast 2.0. In: First Annual North American Association for Computational Social and Organizational Science Conference, Pittsburgh, PA. (2003)
13. Cohen, S., Hindmarsh, A.: CVODE User Guide. LLNL Report UCRL-MA-118618. (1994)
14. Di Marzo Serugendo, G., Foukia, N., Hassas, S., Karageorgos, A., Mostfaoui, S., Rana, O., Ulieru, M., Valckenaers, P., van Aart, C.: Self-organisation: Paradigms and applications. In: Workshop Notes: First International Workshop on Engineering Self-Organising Applications, at AAMAS. (2003)
15. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press (1999)

A Study of System Nervousness in Multi-agent Manufacturing Control System

Hadeli, Paul Valckenaers, Paul Verstraete, Bart Saint Germain, and Hendrik Van Brussel

MACC, PMA Division, Department of Mechanical Engineering, K.U. Leuven Celestijnenlaan 300 B, B-3001 Leuven, Belgium
{Hadeli.Karuna, Paul.Valckenaers, Paul.Verstraete, Bart.SaintGermain, Hendrik.VanBrussel}@mech.kuleuven.be

Abstract. This paper discusses a study on system nervousness in a multi-agent manufacturing control system. Manufacturing control systems, built along this approach, are able to generate short-term forecasts that predict both resource loads and order routings. These forecasts become known throughout the multi-agent system with some time delay. If the agents make their decisions based on these forecasts, proper measures need to be taken to account for these delays, especially when disturbances (rush orders, machine breakdowns) occur. If agents react too eagerly and swiftly, the forecasts become unreliable. This paper studies this issue and the measures in the control system design that address the problem. More precisely, the agents behave in a socially acceptable manner that reconciles adaptation to changed circumstances with predictability.

1 Introduction

This paper presents a study on control system nervousness that appears as a result of coordination mechanisms in multi-agent manufacturing coordination and control systems (see: [1], [2]). The research application discussed in this paper is a multi-agent system implementing manufacturing control. Previous developments ([1], [2], [3], [4]) have shown that this system exhibits several advantages including the ability to quickly respond to any disturbances in the system and the ability to foresee what is going to happen in the future (predictive/forecast ability). This forecasting ability has implications for the system developers. One of them is the need to balance the validity of the forecast information against the ability to react to the occurrence of changes and disturbances. When the system reacts aggressively and causes the forecast to become inaccurate, the system is said to be overly nervous. The art here is to design and implement a proper balance or so-called *social acceptable behavior* in the system.

This paper goes as follows. The next section concisely describes the multi-agent coordination and control system using stigmergy. Next, system nervousness issue is addressed in detail. Finally, the description of an experiment on how to balance a system and some closing remarks conclude this paper.

S.A. Brueckner et al. (Eds.): ESOA 2005, LNAI 3910, pp. 232–243, 2006.

2 The Design of the Multi-agent Manufacturing Control System

In this section, the mechanism of producing the forecast is explained. Our multi-agent manufacturing control comprises of three basic agents types, namely product agents, resource agents and order agents (implementing a PROSA architecture [5]).

- **Product agent.** It masters the know-how on how to make a product type with sufficient quality.
- **Order agent.** It manages the physical product being produced, the product state model, and all logistics information processing related to the job.
- **Resource agent.** It corresponds to a physical entity in a manufacturing system. It offers processing capacity and functionality to the surrounding agents. Its knowledge space is limited to self-monitoring and self-control. It also has a list of resources that connect directly to it. A resource agent is equipped with a blackboard on which other agents can put, observe and modify information. Information on this blackboard evaporates through time, distinguishing is from a normal blackboard [6].

2.1 Coordination Between the Agents

To properly operate a manufacturing systems control, the agents need to share knowledge and cooperate. In general, there are two basic ways to perform coordination, i.e. *coordination by direct communication* and *indirect communication*. In this research, agents utilize indirect communication by dropping *pheromone-like* information on blackboards at the resource agents. There are two main coordination mechanisms in this system, namely *an exploring mechanism* and *an intention propagation mechanism*. In addition, there is a feasibility information propagation mechanism (not discussed here).

Exploration mechanism. The aim of this mechanism is to ensure that all orders entering the system can find their way(s) to get all their operations (steps) executed. Orders, to which order agents attach, need to create and send out mobile agents called *exploring ants* to accomplish this task. Exploring ant agents run in the computation/simulation time mode. In addition, they inherit the problem solving behaviors from their creator. Exploring ants start their exploring activities from the position where the physical work piece resides, and continue until they virtually reach the end of the factory. To accomplish its task, the exploring ant agent observes the pheromone information at local blackboard attached to resource agents, and uses this information as a search guideline. The strategy to explore the available search space is a plug-in for the control system. Not every exploring ant agent uses the same strategy. While performing its task, an exploring ant agent records any important information (for instance the resource(s) it has visited, the estimated starting time, waiting time, duration of each processing step, cost of operation, etc.) at every resource it visits. At the end, the ant compiles them into a proposal of solution and reports back to the order agent. See Fig. 1.

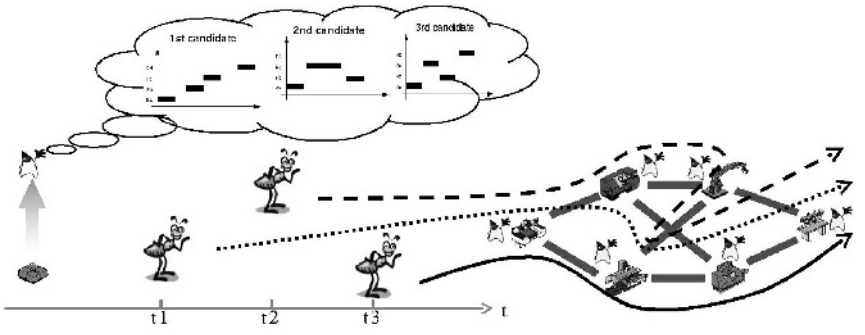


Fig. 1. Exploring mechanism

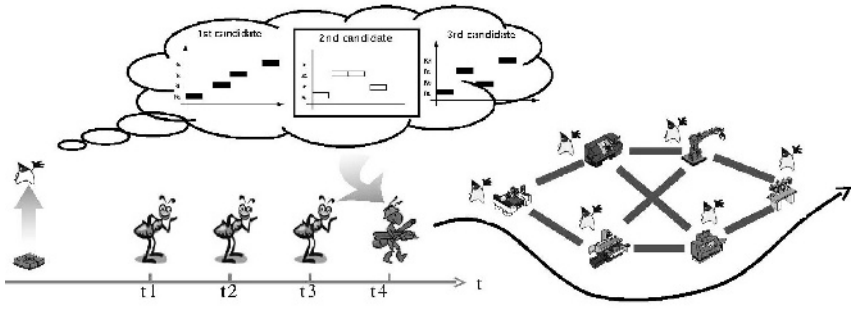


Fig. 2. Intention propagation mechanism

Intention propagation mechanism. In order to be processed at a certain resource, order agents have to express their intention to the resource agent. This part explains the mechanism to propagate the intentions of order agents to the resource agent(s). This task is done by an *intention ant*. An intention ant is created by the order agent, and it virtually executes the remaining routing or processing steps of the selected candidate solution. It visits all resources that are listed in the selected solution. Furthermore, at each resource, this ant expresses the intention of its order agent by requesting a time slot to be reserved. As a response, the resource agent checks its schedule list, blocks the slot and provides the intention ant with the most updated information on the starting time, operation duration and waiting time. During its journey, the intention ant agent collects and updates any necessary information from the resource(s) it has visited. Thus, changes promptly become visible during refresh. To maintain the validity of the information, the slot reservation information evaporates and disappears unless refreshed. As a consequence, if the order agent is serious in having its operation step(s) processed at certain resource(s), it must frequently propagate its intention to the corresponding resource(s). See Fig. 2.

Remark that in manufacturing control, the agent system has a single owner, who in principle decides about the agents allowed therein. Ongoing research is expanding this manufacturing control design toward supply network coordination. In that context, trust and reputation mechanisms are integrated into the design to handle the presence of multiple parties not sharing the same goals [7] to account for agents that may lie. This matter is outside the scope of this paper.

2.2 The Reinforcement of the Forecast Information

The whole exploring and intention propagation mechanism is explained in Fig. 3. This cycle can be named the *solution's reinforcement cycle*. When an order agent expresses its intention to be processed on one/several resources for its remaining processing step(s), the order agent sends out intention propagation ants to make its intention known to the resource agents. Based on this information, the resource agents build their load forecast information and schedule their own logistical activities such as maintenance, setup, etc. If a resource agent is able to work out a reliable forecast on its future load, it can provide an exploring ant agent with more accurate information about the possible starting time at that resource. Based on this information, the order agent can also have a more accurate proposal of solution. Accurate proposal of solutions will yield a more accurate intention. This cycle will then be repeated and the accuracy of the forecast and solution will be reinforced. Basically, it can be said that the accuracy of the solution created by the order agent is a function of the resources load forecast accuracy.

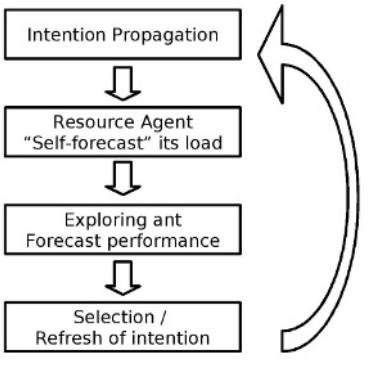


Fig. 3. The reinforcement of solution mechanism

2.3 The Forecast and Its Emergent Property

The interactions between the agents create a useful forecast information, and moreover, the forecast information appears emergently. The exploring and intention propagation mechanisms each produce a new information that cannot

be observed or found at the agent itself. This information appears as a result of interactions, exchanging information, between the agents at a micro level. For this reason, it can be said that these information are emergently appeared. Also remark that agents in this system only have detailed knowledge within their own scope (reflecting themselves). The agents possess no knowledge about the other agents, even for the same type of agent. For instance, one order agent has no information on solutions of the other order agents. There are two types of information that appear from these interactions, namely *a solution list at every order agent* and *a resource load forecast at every resource agent.* These emergently appeared information inform about the possible future state of the system. The solution list reflects the possible route that enable the order to be processed completely, and the resource load forecast informs the resource agent on the upcoming load in the future.

The ability of this new designed system is called *emergent forecasting.* As defined in [1], the working definition of emergent forecasting is: *The ability to foresee what is going to happen in the near future whereby the way in which the forecast appears is not predefined, but –emergently– appears as a result of local interaction of the agents in the system.*

3 System Nervousness

This section covers the system nervousness issue in this innovative manufacturing system design. Historically, practitioners of Material Requirement Planning (MRP) systems first coined the wording *system nervousness.* According to them, it generally refers to frequent changes in the due dates of open orders or, similarly, the instability in planned orders [8], [9]. When dealing with forecasting, accuracy and reliability are the most important properties. When forecast is neither accurate nor reliable, the process of decision making that based on the forecast will neither accurate nor optimal. As a result, the decision will change from one to another, and this cause the nervousness in the system. When the system is nervous, it is unable to make a good planning. In relation to our system, the working definition of system nervousness can be defined as: *the degree to which elements in the system react on the internal and external stimuli (e.g. rush order arrival);* increasing nervousness renders the behavior of the overall system more responsive but less predictable and eventually unstable and chaotic.

3.1 The Appearance of System Nervousness

In the MAS manufacturing control system, coordination and interaction between agents constitutes a source of dynamics and potential instability. As mentioned before, agents in this system coordinate amongst each other by observing pheromone-like information that is placed by other agents on the environment. Furthermore, agents interpret this information and react accordingly. There is no synchronization of information in this system; agents observe the most up-to-date information with some delay. By employing local knowledge, agents try

to interpret the global state of the system and furthermore try to draw a decision out of it. This is how order agents in this system try to find solution(s) to be finished. Exploring ants are sent out to explore the manufacturing network, then construct and report back the proposal for a solution. Next, order agents select one proposal to become their intention. These mechanisms are executed repeatedly during the lifetime of the order agents. Therefore, chances are that order agents encounter a better proposal of solution(s) and are willing to switch to the better one. When this switching happens to appear too frequently, the system behavior becomes very dynamic, and, possibly, even unstable. The forecast becomes obsolete and useless. Consequently, the whole plan collapse. The system loses its stability, and becomes difficult to control. Furthermore it becomes hard to predict what is going to happen in the future. For example, defining the delivery time of the product to the customer becomes difficult and so does observing the load at each resource. The system will have no clue when orders will be finished, resources will have no idea when can they perform the maintenance activities, rescheduling has to be conducted too often, and etc. Therefore, this behavior needs to be prevented by adequate dampening mechanisms.

3.2 Mechanisms to Have Those Agents Live in Harmonious Way

Since being too nervous can harm the whole system, this behavior should be controlled. There are several ways to handle this anomaly. According to Parunak, et.al [10], system performance often improves if individual agents reduce their activity. Hogg. et al. [11] also mentioned several ways to reduce the dynamics to avoid the chaos, namely increase the uncertainty in the agents, imposing a very slow decision making rate on the part of the agents and increase the diversity of the system by introducing additional types of agents which use different problem-solving methods.

In our manufacturing system, order agents should not change intention as soon as they encounter some new and better solution. Remark that a new solution can appear because of changes and disturbances happening in manufacturing environment (for instance rush orders, machine breakdowns, unavailability of materials). When order agents encounter a better solution, order agents must probabilistically wait for a while and observe what is the effect caused by the changes or disturbances and then react accordingly. Remark that the decision of one order agent affects decisions of other order agents. When order agents react immediately on changes, order agents take a decision based on immature/outdated/invalid information. Their decision will be misled by their interpretation of this no-longer-valid information.

3.3 Stabilize the System: *socially acceptable behavior*

In [1], a method called *socially acceptable behavior*, which is used to handle this nervousness issue, is explained briefly by presenting one example. This section discusses the concept underlying this behavior. The idea is simple: order agents

are not supposed to change their intention without sufficient reason. One possible way to handle this nervousness issue is composed of three modules. These modules are:

1. *Ensure that the new solution is better to the old one.* Order agents are allowed to switch to their new intention if and only if they are convinced that the performance of the new solution offered by their exploration team is *significantly better* relatively to the current intention (i.e. the estimated performance difference is well above the prevailing noise/uncertainty level). The idea here is to grant some reward for a new intention that promises a better performance. The implementation of the reward function is based on the idea that changing intentions brings a cost to the whole system. The benefit of changing intention has to be high enough to justify this cost. The example of the reward function can be an exponential function. As in [1], an exponential function is used so that small changes in performance improvement have a far less chance of causing an intention change of the order agent.
2. *Apply a probabilistic mechanism each time the order agent is willing to change intention.* As mentioned before, an order agent has its frequency rate to refresh its intention. The idea is to draw a certain random number, and if this random number is larger or equal to a certain defined threshold, then the order agent is allowed to change its intention. Remark, no matter how big is the value of the significance level, the system should guarantee that if the significance level stays the same all the time, then at the end of a finite time horizon, this change should be announced.
3. *Limit the frequency to change intentions.* This module imposes an upper bound on the frequency at which order agents can change their intentions. After changing its intention, an order agent's threshold for changing its new intention is raised. Then, this threshold decreases over time until its original value provided the order agents intention remains unchanged. This mechanism spaces successive intention changes over time.

3.4 The Detection of Nervousness

Being able to handle nervousness should be combined with the ability to sense the nervousness of the system itself. Nervousness in this system is caused by the behavior of order agents shifting towards a new and better solution to get finished. Consequently, this behavior should be mapped in an observable way; either by the order agents itself or the other types of agents, for example resource agents. In this paper, we propose three types of graphics to observe the nervousness, namely:

1. Frequencies of how often order agent changes its intention in a defined time interval. When the frequency is too high, it automatically shows that order agent is unable to control its desire to change to new better intention.
2. Performance value of each order agent can also become an indicator. When the performance values shows a dense oscillation, it can be seen as an indicator that order agent is always changing its intention.

3. As mentioned above, every order agent must expresses its intention to any resource(s) that it is likely to visit during its life cycle. Resource agents will then plot these intentions from order agents towards time horizon. By observing the fluctuation number of agents at this resource along certain time interval, order agents can define whether they needs to adjust their behavior.

These three sources of information should be interpreted wisely. Noticing that one graph shows a tendency to nervousness doesn't mean that the whole system is nervous. An order agent should study the whole information before taking any decision.

4 Implementation and Experimental Results

In order to observe the appearance of system nervousness, a simple experiment has been conducted. The setup of the experiment is as follows: the production system consists of two identical machining centers, and one arrival station (See Fig. 4). Remark that the use of two resources as an example is for the sake of simplicity in understanding the underneath events, behavior and the executed control mechanisms. In our system, the design is scalable.

Orders that arrive at the system first enter the arrival station and spend a fixed time period in that station before being dispatched to either of the two identical machines. While orders are in the arrival station, order agents start to send out exploring ant agents to find solutions. Each time an order agent sends out an exploring ant, it draws a random number to define to which resource it sends the ant (i.e. a very simple selection rule). Each resource has 0.5 probabilities to be visited. Upon arrival at the resource agent, the exploring ant agent checks the available capabilities it needs and then checks at what time it can start processing on that resource and what the duration will be. After collecting that information, it reports back to its parent. Next, the order agent takes a decision based on this information. This mechanism is repeatedly executed along the life cycle of the order agents. In this setup, order agents are set to be very naive. They switch to a new solution as soon as they find that the solution from another resource is better. The result of this behavior can be seen in Fig. 5. The left graph shows the number of order

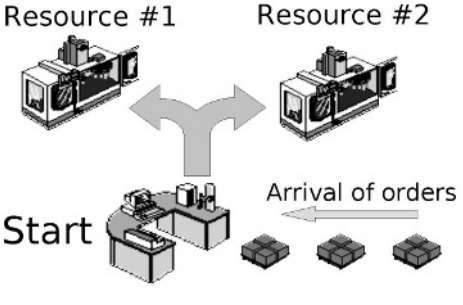


Fig. 4. Experimental setup

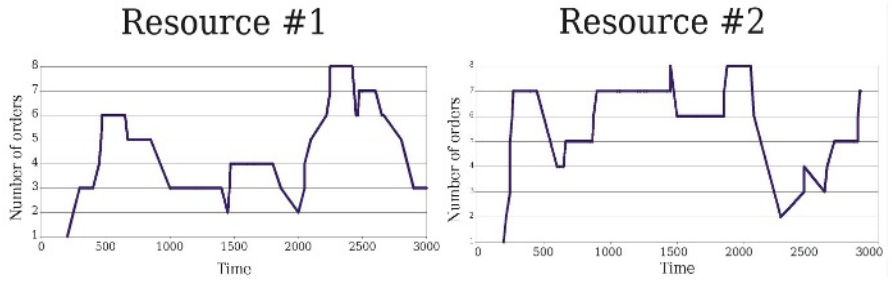


Fig. 5. Number of orders that intend to be processed in first and second resource

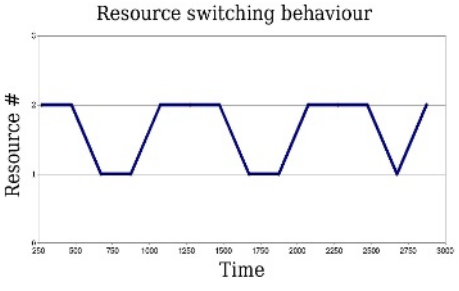


Fig. 6. Example of ever-changing intention behavior of order agent

agents that request to be processed at the first machine along time horizon. The right graph shows the same information for the second machine.

As revealed by the graph, at the beginning, the agents are more interested in the second resource, since more exploring ant agents visit the second resource (by chance). Later, the exploring ants discover the first resource, and some of them change their intention to the first resource. Between time stamp 1000 and 2000, a rush order arrives at the first resource and shifts the schedule of some orders backward, and this causes orders, which discover this, to change their intention to the second resource. Consequently, the second resource receives a lot of order's intentions; on the other hand, the first resource starts losing its contracts. However, fact shows that the rush order consumes less processing time than expected, it finishes earlier. Order agents that spot this changes will take this opportunity to change their intention. There are agents who previously switched to the second resource spotting this change; unfortunately it is too late, since their previous place has been taken by other orders. They have lost their opportunity to be finished earlier. Again, between time stamp 2000 and 3000 another rush order arrives at the second resource and shifts the schedule of some orders backward. As a result, order agents that notice this changes change their intention to the first resource. This ever-switching intention is unlikely to appear frequently. When this behavior appears frequently, then the system is nervous. Example on how an order agent changes its intention from the first resource to the second resource and vice versa can be seen in Fig. 6.

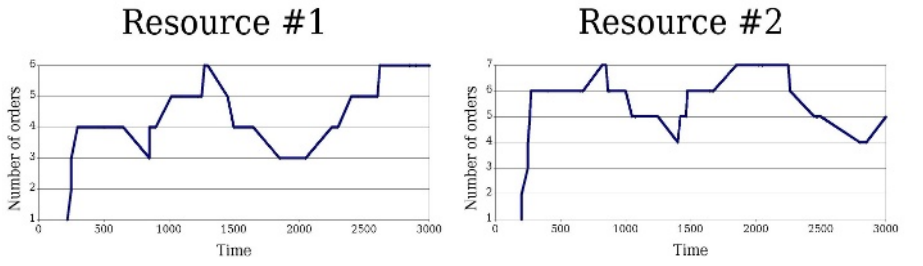


Fig. 7. Number of orders that intend to be processed in first and second resource. Threshold value = 0.5.

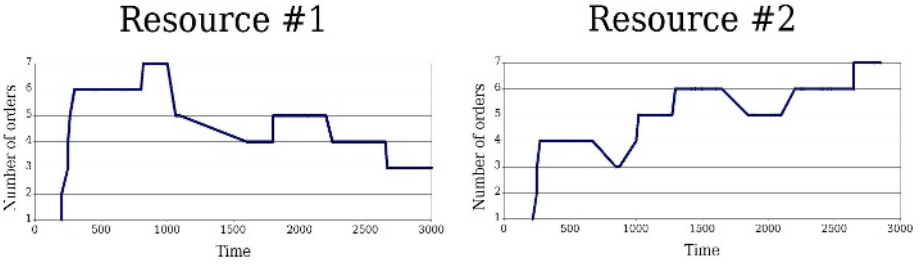


Fig. 8. Number of orders that intend to be processed in first and second resource. Threshold value = 0.3.

In order to control that behavior, socially acceptable behavior is applied to this experiment. The first step is to ensure that the new solution is significantly better than the current solution, and then when order agent is about to change intention, it draws random number. The result of this control mechanism can be seen on Fig. 7, where in this experiment the threshold value is set at 0.5, and Fig. 8, where in this experiment the threshold value is set at 0.3. As can be seen in the graphs, the behavior of the agents is more controllable.

Note that in simple examples, the opportunities to create undesirable system behaviors are limited. Indeed, the larger a system becomes, the more delay occurs in the propagation of information, and this delay plays a key role in the emergence of system nervousness and undesirable (chaotic?) behavior. Conversely, forecast reliability grows more important when it becomes the coupling and coordination medium between mechanisms that operate largely independently, which also is more likely when systems become bigger. In a manufacturing system, somehow independent planning and plan execution is to be expected for normal operations, rush orders, set-ups and changeovers, maintenance and cleaning, etc. Indeed, if only because human operators often are responsible for managing secondary operations, the forecasts will serve as coordination medium in a factory, and forecast reliability will be a necessary condition for it to be useful and workable. In other words, the above example probably is only the proverbial tip of the iceberg.

5 Discussion

System nervousness is a phenomenon that appears because elements in a multi-agent manufacturing control system react aggressively towards stimuli. System nervousness is not entirely deleterious; a manufacturing control system needs to be nervous to a certain degree to deal with changes and disturbances.

This paper presents this nervousness issue and dampening mechanisms, accompanying this re-activeness to changes and disturbances, to ensure that the forecasts remain usable. These dampening mechanisms only allow intention changes when the perceived improvement is sufficiently large and well above the noise levels. They impose intention changes dispersed probabilistically over many refresh cycles (a luxury enjoyed by a control system in cyber space that supervises a macro-mechanical system operating at much lower frequencies; an IT infrastructure management system typically cannot employ this mechanism), and impose a limit on the frequency at which individual order agents can change intentions.

Mechanisms to handle nervousness are not limited to those three mechanisms that are mentioned in the previous chapter. However, the most important mechanisms is to ensure that agents only change their intention if they are sure that the gained value is significant enough. For further research, this mechanism can be extended by varying the level of gained value, where the highest is awarded for intentions related to a more immediate future, and the lowest to the most distant one. Another issue that can be included in this mechanism is by varying the level of gained value for different type of orders. For example, rush orders can switch more easily than normal orders.

The detection of system nervousness is also an important issue in handling system nervousness. Since this system is not a centralized system, the information on how nervous the system is will not be detected and broadcasted in a centralized fashion. Agents have to observe this phenomenon by themselves. Therefore every agent can have a different interpretation on how nervous the system actually is. There are three mechanisms that are introduced in this paper, however the handling mechanism is not limited to them. The first two inputs are inputs that can be observed directly inside the order agent. This information gives an idea to the order agent of the impact of frequent changes of intention to its performance value. The third input, the number of orders that plan to request service(s) from certain resource, gives an idea to order agent of the behavior of other agents. At the very beginning (warm-up phase) of agent life-cycle, it has no idea on how nervous the system is, however as soon as it adjust its parameter concerning intention changes, it can start observing the result and adjust its behavior. Agents can start to learn what will be the effect by applying an aggressive behavior rather than calm behavior. How the agent uses this information about the system nervousness still is ongoing research.

Acknowledgements

This research is supported by the European Commission under GROWTH program and the K.U.Leuven research council (GOA-AgCo2).

References

1. Hadeli, Valckenaers, P., Saint Germain, B., Verstraete, P., Zamfirescu, C., Van Brussel, H.: Emergent Forecasting Using a Stigmergy Approach in Manufacturing Coordination and Control. to be appear in: Engineering-Self Organising Systems: Methodologies and Applications, *Lecture Notes in Artificial Intelligences*. **3464** (2005)
2. Valckenaers, P., Saint Germain, B., Vertraete, P., Hadeli, Zamfirescu, C., Van Brussel, H.: Ant Colony Engineering in Coordination and Control: How to engineer a short-term forecasting system. Proceedings of the International Workshop on Emergent Synthesis-IWES 2004 (2004) 125–132
3. Hadeli, Valckenaers, P., Zamfirescu, C., Van Brussel, H., Saint Germain, B., Holvoet, T., Steegmans, E.: Self-Organising in Multi-agent Coordination and Control Using Stigmergy. Engineering-Self Organising Systems: Nature-Inspired Approaches to Software Engineering, *Lecture Notes in Artificial Intelligences*. **2977** (2004) 105–123
4. Hadeli, Valckenaers, P., Kollingbaum, M., Van Brussel, H.: Multi-agent coordination and control using stigmergy. Computers In Industry. **53** (2004) 75–96
5. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference Architecture for Holonic Manufacturing Systems: PROSA. Computers in Industry, *Special issue on Intelligent Manufacturing System* **37** (1998) 255–274
6. Ciancarini, P., Omicini, A., Zambonelli, F.: Coordination Models for Multi-Agent Systems. AgentLink News **3** (1999)
7. Saint Germain, B., Valckenaers, P., Verstraete, P., Hadeli, Van Brussel, H.: Supply network control, an engineering perspective. Submitted to Control Engineering Practice (2005)
8. Blackburn, J.D., Kropp, D.H., Millen, R.A.: A Comparison of strategies to dampen nervousness in MRP systems. Management Science **32**-4 (1986) 413–429
9. Ho, C.J., Ireland, T.C.: Correlating MRP system nervousness with forecast errors. International Journal Production Research **36**-8 (1998) 2285–2299
10. Parunak, H.V.D., Brueckner, S.A., Matthews, R., Sauter, J.: How To Calm Hyperactive Agents. Proceedings AAMAS 2003,
11. Hogg, T., Huberman, B.A.: Controlling Chaos in Distributed Systems. IEEE Transaction on Systems, Man and Cybernatics **21** (1991) 1325–1332

Author Index