Advanced Information and Knowledge Processing

Mikhail Prokopenko *Editor*

Advances in Applied Self-Organizing Systems

Second Edition

Springer

Advances in Applied Self-Organizing Systems

Advanced Information and Knowledge Processing

Mikhail Prokopenko
Editor

Advances in Applied Self-Organizing Systems

Second Edition

Springer

Editor
Mikhail Prokopenko
ICT Centre
CSIRO
Marsfield, NSW
Australia

ISSN 1610-3947 Advanced Information and Knowledge Processing
ISBN 978-1-4471-5112-8 ISBN 978-1-4471-5113-5 (eBook)
DOI 10.1007/978-1-4471-5113-5
Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2013938780

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

It has been 60 years since the first time that a system was termed “self-organizing” in modern scientific literature.[1] During this time, the concept of self-organization developed in many directions and affected diverse fields, ranging from biology to physics to social sciences. For example, in his seminal book “At home in the Universe”, Stuart Kauffman argued that natural selection and self-organization are two complementary forces necessary for evolution: “If biologists have ignored self-organization, it is not because self-ordering is not pervasive and profound. It is because we biologists have yet to understand how to think about systems governed simultaneously by two sources of order … if ever we are to attain a final theory in biology, we will surely, surely have to understand the commingling of self-organization and selection”.[2] A similar dilemma can be re-phrased for various fields of engineering: If engineers have ignored self-organization, it is not because self-ordering is not pervasive and profound. It is because we engineers have yet to understand how to think about systems governed simultaneously by two sources of order: traditional design and self-organization.

Without claiming an undue comprehensiveness, this book presents state-of-the-practice of self-organizing systems, and suggests a high-level breakdown of applications into two general areas:

- Distributed Management and Control;
- Self-organizing Computation.

Each of these areas is exemplified with a selection of invited contributions, written and peer-reviewed by international experts in their respective fields, convincingly demonstrating achievements of self-organizing systems. The overall selection balances many aspects: modelling vs simulation vs deployment, as well as macro- vs micro-scale.

[1]Ashby, W. R. (1947). Principles of the self-organizing dynamic system. *Journal of General Psychology*, *37*, 125–128.

[2]Kauffman, S. (1995). *At home in the Universe* (p. 112). London: Oxford University Press.

We begin with more established fields of traffic management, sensor networks, and structural health monitoring, building up towards robotic teams, solving challenging tasks and deployed in tough environments. These scenarios mostly belong to macro-level, where multiple agents (e.g, robots) themselves may contain complicated components. Nevertheless, the main topic is self-organization within a multi-agent system, brought about by interactions among the agents. The second half of the book follows with a deeper look into the micro-level, and considers local interactions between agents such as particles, cells, and neurons. These interactions lead towards self-organizing resource management, scheduling, and visualization, as well as self-modifying digital circuitry, immunocomputing, memristive excitable automata, and eventually to Artificial Life.

We believe that the broad range of scales at which self-organizing systems are applied to real-world problems is one of the most convincing arguments for acceptance of the unifying theme—practical relevance and applicability of self-organization.

The second edition revisits these studies, providing concise summaries of the research during the last 5 years (the chapter "epilogues"), while offering new extensions for several important works. These extensions cover a diverse field, including a distributed thermal protection system for a spacecraft re-entering the atmosphere; self-configuring analog (Songline) processors; ad-hoc multi-robot systems with decentralized control; memristive cellular automata applicable to spintronic devices, neuromorphic circuits, and programmable electronics.

The progress demonstrated by the applied self-organizing systems that were developed in the last years is encouraging. An important general trend that emerged in the area since the first edition is *Guided Self-Organization* (GSO): an approach leveraging the strengths of self-organization while directing the outcome of the self-organizing process towards some specific goals. The approach builds up on earlier ideas of "design for emergence" and "emergent functionality" (as Luc Steels called it in late 1980s), aiming to provide a formal framework for studying and designing GSO systems. Equipped with such a unifying framework, designers may hope not only to produce the systems quicker and more reliably, but also to precisely verify and validate eventual performance of the outcomes—the key prerequisite to a wide-range adoption of self-organizing applications.

Sydney Mikhail Prokopenko
April 2007 (first edition)
January 2013 (second edition)

Contents

Contributors

Andrew Adamatzky University of the West of England, Bristol, UK

Gianluca Baldassarre Laboratory of Autonomous Robotics and Artificial Life, Istituto di Scienze e Tecnologie della Cognizione, Consiglio Nazionale delle Ricerche (LARAL-ISTC-CNR), Rome, Italy

Alla V. Borisova St. Petersburg Institute for Informatics and Automation, Russian Academy of Sciences, St. Petersburg, Russia

Fabio Boschetti Marine and Atmospheric Research, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Wembley, WA, Australia

Leon Chua EECS Department, University of California, Berkeley, Berkeley, CA, USA

Seung-Bae Cools Centrum Leo Apostel, Vrije Universiteit Brussel, Brussels, Belgium

Bart D'Hooghe Centrum Leo Apostel, Vrije Universiteit Brussel, Brussels, Belgium

Lisa J.K. Durbeck Cell Matrix Corporation, Blacksburg, VA, USA

Hugh Durrant-Whyte National ICT Australia (NICTA), Australian Technology Park, Eveleigh, NSW, Australia

Carlos Gershenson Centrum Leo Apostel, Vrije Universiteit Brussel, Brussels, Belgium

Randall Gray Marine and Atmospheric Research, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Hobart, TAS, Australia

Tad Hogg Hewlett-Packard Laboratories, Palo Alto, CA, USA

Nigel Hoschke Materials Science and Engineering, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Lindfield, NSW, Australia

Ryszard Kowalczyk Swinburne Centre for Information Technology Research, Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia

Young Choon Lee School of Information Technologies, The University of Sydney, Sydney, NSW, Australia

Nicholas J. Macias Cell Matrix Corporation, Blacksburg, VA, USA

George Mathews National ICT Australia (NICTA), Australian Technology Park, Eveleigh, NSW, Australia

Daniel Polani Adaptive Systems Research Group, Department of Computer Science, University of Hertfordshire, Hatfield, UK

Don C. Price Materials Science and Engineering, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Lindfield, NSW, Australia

Mikhail Prokopenko ICT Centre, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Epping, NSW, Australia

Tino Schlegel Swinburne Centre for Information Technology Research, Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia

D. Andrew Scott National Measurement Institute, Lindfield, NSW, Australia

Ivan Tanev Department of Information Systems Design, Doshisha University, Kyotanabe, Kyoto, Japan

Alexander O. Tarakanov St. Petersburg Institute for Informatics and Automation, Russian Academy of Sciences, St. Petersburg, Russia

Andrew Vande Moere Faculty of Architecture, Design and Planning, The University of Sydney, Sydney, NSW, Australia

Albert Y. Zomaya School of Information Technologies, The University of Sydney, Sydney, NSW, Australia

Part I
Introduction

Chapter 1
Design Versus Self-Organization

Mikhail Prokopenko

1.1 Introduction

The theory of self-organization has sufficiently matured over the last decades, and begins to find practical applications in many fields. Rather than analyzing and comparing underlying definitions of self-organization—the task complicated by a multiplicity of complementary approaches in literature; e.g., recent reviews (Boschetti et al. 2005; Prokopenko et al. 2009)—we investigate a possible design space for self-organizing systems, and examine ways to balance design and self-organization in the context of applications.

Typically, self-organization is defined as the evolution of a system into an organized form in the absence of external pressures. A broad definition of self-organization is given by Haken: "a system is self-organizing if it acquires a spatial, temporal or functional structure without specific interference from the outside. By 'specific' we mean that the structure or functioning is not impressed on the system, but that the system is acted upon from the outside in a non-specific fashion. For instance, the fluid which forms hexagons is heated from below in an entirely uniform fashion, and it acquires its specific structure by self-organization" (Haken 1988).

Another definition is offered by Camazine et al. in the context of pattern formation in biological systems: "Self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern" (Camazine et al. 2001).

In our view, these definitions capture three important aspects of self-organization. Firstly, it is assumed that the system has many interacting components (agents), and advances from a less organized state to a more organized state dynamically, over

M. Prokopenko (✉)
ICT Centre, Commonwealth Scientific and Industrial Research Organisation (CSIRO),
PO Box 76, Epping, NSW 1710, Australia
e-mail: mikhail.prokopenko@csiro.au

some time, while exchanging energy, matter and/or information with the environment. Secondly, this organization is manifested via global coordination, and the global behaviour of the system is a result of the interactions among the agents. In other words, the global pattern is not imposed upon the system by an external ordering influence (Bonabeau et al. 1997). Finally, the components, whose properties and behaviors are defined prior to the organization itself, have only local information, and do not have knowledge of the global state of the system—therefore, the process of self-organization involves some local information transfer (Polani 2003).

Self-organization within a system brings about several attractive properties, in particular, robustness, adaptability and scalability. In the face of perturbations caused by adverse external factors or internal component failures, a *robust* self-organizing system continues to function. Moreover, an *adaptive* system may reconfigure when required, degrading in performance "gracefully" rather than catastrophically. In certain circumstances, a system may need to be extended with new components and/or new connections among existing modules—without self-organization such *scaling* must be pre-optimized in advance, overloading the traditional design process.

It is interesting at this stage to contrast traditional engineering methods with biological systems that evolve instead of being built by attaching together separately pre-designed parts. Each biological component is reliant on other components and co-evolves to work even more closely with the whole. The result is a dynamic system where components can be reused for other purposes and take on multiple roles (Miller et al. 2000), increasing robustness observed on different levels: from a cell to an organism to an ant colony. Complementarity of co-evolving components is only one aspect, however. As noted by Woese (2004), "Machines are stable and accurate because they are designed and built to be so. The stability of an organism lies in resilience, the homeostatic capacity to reestablish itself." While traditionally engineered systems may still result in brittle designs incapable of adapting to new situations, "organisms are resilient patterns in a turbulent flow—patterns in an energy flow" (Woese 2004). It is precisely this homeostatic resilience that can be captured by self-organization.

However, in general, self-organization is a not a force that can be applied very naturally during a design process. In fact, one may argue that the notions of design and self-organization are contradictory: the former approach often assumes a methodical step-by-step planning process with predictable outcomes, while the latter involves non-deterministic spontaneous dynamics with emergent features.

Thus, the main challenge faced by designers of self-organizing systems is how to achieve and control the desired dynamics. Erring on the one side may result in over-engineering the system, completely eliminating emergent patterns and suppressing an increase in internal organization with outside influence. Strongly favoring the other side may leave too much non-determinism in the system's behaviour, making its verification and validation almost impossible. The balance between design and self-organization is our main theme, and we hope to identify essential causes behind successful applications, and propose guiding principles for future scenarios.

1.2 Background

Self-organization occurs in both biological and non-biological systems, ranging from physics and chemistry to sociology. In non-biological systems, it is produced by a flow of energy into or out of the system that pushes it beyond equilibrium: the winds that produce characteristic ripples in sand, the temperature gradients that produce Bénard convection cells in a viscous fluid, the thermodynamic forces that lead to crystal growth and characteristic molecular conformations are all examples of these external energy inputs. However, the nature of the outcomes depends critically on the interactions between the low-level components of the systems—the grains of sand, the molecules in the fluid, the atoms in the crystals and molecules—and these interactions are determined by the laws of nature and are immutable (Prokopenko et al. 2006c).

In biological systems, on the other hand, the interactions between components of a system may change over generations as a result of evolution. There are selection pressures shaping adaptation of the system (a biological organism) to the environment. These selection pressures lead to self-organization that is desirable for the survival of the system in the environment in which it has evolved, but which may be undesirable in other environments. Similarly, when using evolutionary methods for the design of applied self-organizing systems, there is a need to identify appropriate selection pressures (described more systematically in Sect. 1.3). These pressures constrain and channel components' interactions to produce desirable responses (Prokopenko et al. 2006c).

Self-organization is typically (but not necessarily) accompanied by emergence of new patterns and structures. An important distinction between two kinds of emergence is identified by Crutchfield (1994):

- pattern formation, referring to an external observer who is able to recognize how unexpected features (patterns) "emerge" during a process (e.g., spiral waves in oscillating chemical reactions)—these patterns may not have specific meaning within the system, but obtain a special meaning to the observer when detected;
- intrinsic emergence, referring to the emergent features which are important within the system because they confer additional functionality to the system itself, e.g. support global coordination and computation—for example, the emergence of coordinated behaviour in a flock of birds allows efficient global information processing through local interactions, which benefits individual agents.

In turn, the functional patterns emerging intrinsically can be further distinguished in terms of their usage: one may consider an *exploitation* of the patterns while the system is near an equilibrium, or an *exploration* of patterns during the system's shift away from an equilibrium. Examples of exploration include auto-catalytic processes leading to optimal paths' emergence, self-organized criticality phenomena, self-regulatory behaviour during co-evolution, etc., whereas exploitation may be used during traversing the optimal paths, self-assembly along optimal gradients, replication according to error-correcting encodings, and so on.

Self-organization has been the topic of many theoretical investigations, while its practical applications have been somewhat neglected. On the other hand there are many studies reporting various advances in the field, and the lack of a common design methodology for these applications across multiple scales indicates a clear gap in the literature. The following short review pinpoints relevant works which, nevertheless, may set the scene for our effort.

Zambonelli and Rana (2005) discussed a variety of novel distributed computing scenarios enabled by recent advances in microelectronics, communication, and information technologies. The scenarios are motivated by a number of challenges which, on the one hand, make it impossible for application components to rely on a priori information about their execution context, and on the other hand, make it very difficult for engineers to enforce a strict micro-level control over the components. These challenges call for novel approaches to distributed systems engineering, and a point is made that the industry has also realized the importance of self-organization and decentralized management approaches (the "Autonomic Computing" program at IBM Research, the "Dynamic Systems Initiative" at Microsoft, and the "Adaptive Enterprize" strategy from HP). Zambonelli and Rana conclude that "perhaps one of the barriers for real-world adoption is the lack of support in existing distributed systems infrastructure to enable these techniques to be utilized effectively" (Zambonelli and Rana 2005). One of the aims of our effort is to explore directions towards a better adoption of self-organization as a concept for engineering distributed systems. In addition, we intend to consider several novel applications, extending the range of applicability of self-organizing systems.

Sahin and Spears (2004) consider swarm robotics—the study of how a swarm of relatively simple physically embodied agents can be constructed to collectively accomplish tasks that are beyond the capabilities of a single one. Unlike other studies on multi-robot systems, swarm robotics emphasizes self-organization and emergence, while keeping in mind the issues of scalability and robustness. These emphases promote the use of relatively simple robots, equipped with localized sensing ability, scalable communication mechanisms, and the exploration of decentralized control strategies. While definitely very valuable in addressing the task in point, this work does not expand into related areas (which are out of its scope), thus leaving inter-scale relationships indistinct.

Self-organizing computation is another example of an emerging application domain. Czap et al. (2005) argue that since self-organization and adaptation are concepts stemming from nature, conventional self-organization and adaptation principles and approaches are prevented from being directly applicable to computing and communication systems, which are basically artificial systems. Their book discusses these challenges, as well as a range of state-of-the-art methodologies and technologies for the newly emerging area of Self-Organization and Autonomic Informatics. What may be lacking, however, is a well-grounded connection to other application areas, and identification of possible overlaps.

In summary, self-organization is a multi-faceted phenomenon, present in many fields, operating at multiple scales, and performing diverse roles. We hope that the practical case studies described in this book may not only illustrate the richness of the topic, but also provide guidance to the intricate area.

1.3 Evolutionary Design

One way to address the "design versus self-organization" dilemma is to consider possible design parameters that guide the design of a self-organizing system. Let us begin with a quotation from a topical review by Scaruffi, who considered the task of a "design without a designer" (Scaruffi 2003):

> The physicist Sadi Carnot, one of the founding fathers of Thermodynamics, realized that the statistical behavior of a complex system can be predicted if its parts were all identical and their interactions weak. At the beginning of the century, another French physicist, Henri Poincaré, realizing that the behavior of a complex system can become unpredictable if it consists of few parts that interact strongly, invented "chaos" theory. A system is said to exhibit the property of chaos if a slight change in the initial conditions results in large-scale differences in the result. Later, Bernard Derrida will show that a system goes through a transition from order to chaos if the strength of the interactions among its parts is gradually increased. But then very "disordered" systems spontaneously "crystallize" into a higher degree of order.

An important lesson here is that there are transitions separating ordered and chaotic regimes, and by varying *control parameters* (e.g., the system composition and the strength of interactions within it) one may trigger these transitions. This observation by itself is not sufficient to identify the generic design space. However, several approaches, also reviewed by Scaruffi, further develop this idea. In particular, *synergetics*—a theory of pattern formation in complex systems, developed by Haken (1983b)—is relevant. Following the Ginzburg-Landau theory, Haken introduced *order parameters* in explaining structures that spontaneously self-organize in nature. When energy or matter flows into a system typically describable by many variables, it may move far from equilibrium, approach a threshold (that can be defined in terms of some control parameters), and undergo a phase transition. At this stage, the behavior of the overall system can be described by only a few order parameters (degrees of freedom) that characterize newly formed patterns. In other words, the system becomes low-dimensional as some dominant variables "enslave" others, making the whole system to act in synchrony. A canonical example is laser: a beam of coherent light created out of the chaotic movement of particles.

The "enslaving principle" generalizes the order parameter concept: in the vicinity of phase transitions, a few *slower* and long-lasting components of the system determine the macroscopic dynamics while the *faster* and short-lasting components quickly relax to their stationary states (Jirsa et al. 2002). The fast-relaxing components represent stable modes, e.g., the chaotic motion of particles. The slower components represent unstable modes, i.e., the coherent macroscopic structure and behavior of the whole system (see also Chap. 2 Polani 2008). Thus, the order parameters can be interpreted as the amplitudes of these unstable modes that determine the macroscopic pattern and the dynamics of the enslaved fast-relaxing modes. In particular, the stationary states of fast-relaxing components are determined by the order parameters.

It can be argued that a layered hierarchical structure emerges where "higher" levels (the order parameters) "control" or "force order upon" lower levels (short-lasting and fast-relaxing components) (Liljenström and Svedin 2005). However, we may also point out circular nature of the mechanism: the dynamics of microscopic short-lasting components brings the system to the phase transition, forcing it over a threshold and stimulating macroscopic pattern formation. When new macroscopic patterns emerge, the order parameters enforce the downward enslavement (Haken 1983a):

> Because the order parameter forces the individual electrons to vibrate exactly in phase, thus imprinting their actions on them, we speak of their "enslavement" by the order parameter. Conversely, these very electrons generate the light wave, i.e. the order parameter, by their uniform vibration.

The collective synchronization of oscillators is another example of such circular causality. It is well-known that coupled limit-cycle oscillators tend to synchronize by altering their frequencies and phase angles (Kuramoto 1984; Pikovsky et al. 2001). Given a pulse, initially a few oscillators become synchronized, then a mean field forms that drives other oscillators which, in turn, contribute to the mean field. Thus, such self-organization can be better characterized in terms of tangled hierarchies exhibiting Strange Loops described by Hofstadter: "an interaction between levels in which the top level reaches back down towards the bottom level and influences it, while at the same time being itself determined by the bottom level" (Hofstadter 1989).

Importantly, tangled hierarchies with circular causality between microscopic and macroscopic levels result in stable behavior. For example, as noted by M.J.M. Volman in the context of neurobehavioral studies (Volman 1997),

> Enslaving provides a parsimonious solution to the degrees of freedom problem: only one or a few variables have to be controlled by the central nervous system. Two variables play an essential role: the order parameter or collective variable, and the control parameter. The collective variable captures the intrinsic order of the system. The control parameter is the parameter that induces a phase transition from one stable state of the system to another.

A self-organized low-dimensional system with fewer available configurations may be more efficient than a high-dimensional disorganized system which may, in principle, access more configurations. The reason for such higher efficiency is explained by Kauffman (2000) who suggested that the underlying principle of self-organization is the generation of constraints in the release of energy. According to this view, the constrained release allows for such energy to be controlled and channelled to perform some useful work. This work is "propagatable" and can be used in turn to create better and more efficient constraints, releasing further energy and so on. Importantly, the ability to constrain and control the release of energy provides the self-organized system with a variety of behaviors that can be selectively chosen for successful adaptation (Prokopenko et al. 2009), thus conforming with Ashby's Law of Requisite Variety.

These observations further advance our search for a suitable design space: control parameters become optimization variables, while order parameters contribute to (multi-)objective functions. The overall optimization is to be solved under the constraints generated by the release of energy from components in the system. The three elements (variables, objective functions, and constraints) constitute the design space. This approach suggests to consider *evolutionary design* as the methodology for designing self-organizing systems. Typically, evolutionary design may employ genetic algorithms in evolving optimal strategies that satisfy given fitness functions, by exploring large and sophisticated search-space landscapes (Crutchfield et al. 1998; Miller et al. 2000). Using selection and genetic variation of microscopic components, evolutionary design is capable to discover macroscopic patterns that correspond to order parameters.

There is a fundamental reason for employing evolutionary methods in designing self-organizing nonlinear systems. As pointed out by Heylighen (2000), many intricacies associated with nonlinearity (e.g., limit cycles, chaos, sensitivity to initial conditions, dissipative structures, etc.) can be interpreted through the interplay of positive and negative feedback cycles. In turn, both types of feedback provide a selective advantage: when variations positively reinforce themselves (e.g., autocatalytic growth) the number and diversity of configurations is increased to the point where resources may become insufficient, and competition may intensify. On the other hand, when variations reduce themselves via negative feedback, configurations become more stable. Therefore, a self-organizing system with both types of feedback is a natural target for evolutionary design.

Heylighen further notes that the increase in organization can be measured quantitatively as a decrease of statistical entropy, exported by the self-organizing system into its surroundings. Prigogine called systems which continuously export entropy in order to maintain their organization *dissipative structures* (Prigogine 1980), and formulated the minimum entropy production principle: stable near-equilibrium dissipative systems minimize their rate of entropy production. While the practical applicability of this principle is still a subject of an ongoing debate, we believe that it identifies a very generic guiding rule for evolutionary design, suggesting to incorporate minimization of entropy rate in the employed fitness functions.

Consequently, we may approach evolutionary design in two ways: via task-specific objectives or via generic intrinsic selection criteria (Prokopenko et al. 2006a, 2006b). The latter method—*information-driven evolutionary design*—essentially focuses on information transfer within specific channels, enabling "propagatable" work, i.e. self-organization. Various generic information-theoretic criteria may be considered, for example:

- maximization of information transfer in perception-action loops (Klyubin et al. 2004, 2005);
- minimization of heterogeneity in agent states, measured with the variance of the rule-space's entropy (Wuensche 1999; Prokopenko et al. 2005d) or Boltzmann entropy in agent states (Baldassarre et al. 2007); see also Chap. 7 (Baldassarre 2008);
- stability of multi-agent hierarchies (Prokopenko et al. 2005d);
- efficiency of computation (computational complexity);
- efficiency of communication topologies (Prokopenko et al. 2005b, 2005c); see also Chap. 4 (Hoschke et al. 2008, 2013);
- efficiency of locomotion and coordination of distributed actuators (Prokopenko et al. 2006a, 2006b; Der et al. 1999; Tanev et al. 2005); see also Chap. 6 (Tanev 2008), etc.

The solutions obtained by information-driven evolution can be judged by their degree of approximation of direct evolutionary computation, where the latter uses task-specific objectives. A good approximation indicates that the chosen criteria capture information dynamics of self-organization within specific channels.

In summary, design is possible even when the target is a far-from-equilibrium nonlinear system with multiple independent and interacting units. One should not avoid far-from-equilibrium dynamics and symmetry-breaking behavior but rather exploit the opportunities for creating stable patterns out of fluctuations.[1] The regions of macroscopic stability correspond to order parameters. These regions are separated by phase transitions that can be quantified via entropy rate, and induced by varying the control parameters. In short, one should design local rules of interaction among microscopic components (including the constraints and control variables) in such a way that macroscopic patterns (measured via the objective functions) self-organize globally, being then selected by information-driven evolution.

1.3.1 Example: Self-Organizing Locomotion

Different internal channels through which information flows within the system may be chosen for a specific analysis. For example, let us consider a modular robotic system modelling a multi-segment snake-like (salamander) organism, with actuators (“muscles”) attached to individual segments (“vertebrae”). A particular side-winding locomotion emerges as a result of individual control actions when the actuators are coupled within the system and follow specific evolved rules, as described in Chap. 6 (Tanev 2008), as well as by Tanev et al. (2005). There is no global coordinating component in the evolved system, and it can be shown that, as the modular robot starts to move across the terrain, the distributed actuators become more coupled when a coordinated side-winding locomotion is dominant. The periodicity of the side-winding locomotion can be related to order parameter(s). Faced with obstacles, the robot temporarily loses the side-winding pattern: the modules become less organized, the strength of their coupling (that can be selected as a control parameter) is decreased, and rather than exploiting the dominant pattern, the robot explores various alternatives. Such exploration temporarily decreases self-organization within the system. When the obstacles are avoided, the modules “rediscover” the dominant side-winding pattern by themselves, manifesting again the ability to self-organize without any global controller. Of course, the “magic” of this self-organization is explained by properties defined *a priori*: the control rules employed by the biologically-inspired actuators have been obtained by a genetic programming algorithm, while the biological counterpart (the rattlesnake *Crotalus cerastes*) naturally evolved over long time. Our point is simply that these transitions can be quantitatively measured within the channels of interest (e.g., via generalized

[1]According to Prigogine (1980), a thermodynamic system can be in a steady state while being not in equilibrium.

entropy rate and excess entropy, and used in information-driven evolutionary design (Prokopenko et al. 2006a, 2006b).

1.4 Information Dynamics

The generation of constraints in the release of energy explains why a low-dimensional self-organized system with fewer available configurations is more efficient than a high-dimensional disorganized system. One quantitative interpretation of this is that many actual configurations of a disorganized system may not be statistically different (Prokopenko et al. 2009). On the other hand, as the system self-organizes, it reaches a progressively larger number of statistically different configurations. In other words, an increase in organization can be measured via an increase in statistical complexity of the system's dynamics (Crutchfield 1994; Shalizi 2001; Shalizi et al. 2004). The latter approach is formalized within the Computational Mechanics methodology which equates statistically different configurations with *causal states*.[2]

Recently, Correia (2006) analyzed self-organization motivated by embodied systems, i.e. physical systems situated in the real world, and established four fundamental properties of self-organization: no external control, an increase in order, *robustness*,[3] and interaction. All of these properties are easily interpretable in terms of information dynamics (Prokopenko et al. 2009). Firstly, the absence of external control may correspond to spontaneous arising of information transfer within the system without any flow of information into the self-organizing system. Secondly, an increase in order or complexity reflects simply that the statistical complexity is increased internally within the system: $C_\mu^{system}(t_2) > C_\mu^{system}(t_1)$, for $t_2 > t_1$, where $C_\mu^{system}(t)$ is the statistical complexity at time t. In general, the distinction between these two requirements may be relaxed (Prokopenko et al. 2009), resulting in the requirement that in a self-organizing system the complexity of external influence $C_\mu^{influence}$ is strictly less than the gain in internal complexity, $\Delta C_\mu^{system} = C_\mu^{system}(t_2) - C_\mu^{system}(t_1)$, within the system:

$$C_\mu^{influence} < \Delta C_\mu^{system}$$

Thirdly, a system is robust if it continues to function in the face of perturbations (Wagner 2005)—in terms of information dynamics, robustness of a self-organizing system to perturbations means that it may interleave stages of an increased information transfer within some channels (dominant patterns are being exploited) with periods of decreased information transfer (alternative patterns are being explored).

[2]Statistical complexity is also an upper bound of predictive information, or structure, within the system (Bialek et al. 2001; De Wolf and Holvoet 2005).

[3]Although Correia refers to this as adaptability, he in fact defines robustness.

Finally, the interaction property is described by Correia (2006) as follows: minimization of local conflicts produces global optimal self-organization, which is evolutionarily stable. Following the review by Prokopenko et al. (2009), we note that minimization of local conflicts corresponds to a reduction of assortative noise (or non-assortativeness within the system, increasing therefore the information transfer within the system.

1.4.1 Example: Self-Organizing Traffic

In the context of pedestrian traffic, Correia (2006) argues that it can be shown that the "global efficiency of opposite pedestrian traffic is maximized when interaction rate is locally minimized for each component. When this happens two separate lanes form, one in each direction. The minimization of interactions follows directly from maximizing the average velocity in the desired direction." In other words, the division into lanes results from maximizing velocity (an overall objective or fitness), which in turn supports minimization of conflicts. A practical case study of self-organizing traffic, presented in Chap. 3 (Cools et al. 2008), considers ways to minimize conflicts as well, e.g., via "platoons" or "convoys" of cars that move together improving the traffic flow.

Another example is provided by ants: "Food transport is done via a trail, which is an organized behaviour with a certain complexity. Nevertheless, a small percentage of ants keeps exploring the surroundings and if a new food source is discovered a new trail is established, thereby dividing the workers by the trails (Hubbell et al. 1980) and increasing complexity" (Correia 2006). Here, the division into trails is again related to an increase in fitness and complexity.

These examples demonstrate that when local conflicts are minimized, the degree of coupling among the components (i.e. interaction) increases and the information flows easier, thus increasing the information transfer. This means that self-organization as a dynamic process tends to increase the overall diversity of a system (more lanes or trails), while keeping in check the interplay among different channels (the assortative noise within the system, the conflicts). In summary, self-organization can be quantitatively studied via information dynamics when the appropriate channels are identified (Prokopenko et al. 2009).

1.4.2 Example: Self-Organizing Computation

In illustrating the phenomenon of self-organizing computation we shall use Cellular Automata (CA): discrete spatially-extended dynamical systems that are often used as models of many computational, physical and biological processes—see, e.g. Mitchell et al. (1993) and Chap. 14 (Adamatzky 2008; Adamatzky and Chua

2013). The main conjecture within this application domain is that physical systems achieve the prerequisites for computation (i.e., transmission, storage, modification) in the vicinity of a phase transition between periodic and chaotic behavior (Langton 1991).

In classifying CA rules according to their asymptotic behavior, the following qualitative taxonomy is typically employed: class I (homogeneity); class II (periodicity); class III (chaos); class IV (complexity) (Wolfram 1984). The first class consists of CA that, after a finite number of time steps, produce a unique, homogeneous state (analogous to "fixed points" in phase space). From almost all initial states, such behaviour completely destroys any information on the initial state, i.e. complete prediction is trivial and complexity is low. The second class contains automata which generate a set of either stable or periodic structures (typically having small periods—analogous to "limit cycles" in phase space)—each region of the final state depends only on a finite region of the initial state, i.e. information contained within a small region in the initial state thus suffices to predict the form of a region in the final state. The third class includes CA producing aperiodic ("chaotic") spatiotemporal patterns from almost all possible initial states—the effects of changes in the initial state almost always propagate forever at a finite speed, and a particular region depends on a region of the initial state of an ever-increasing size (analogous to "chaotic attractors" in phase space). While any prediction of the "final" state requires complete knowledge of the initial state, the regions are indistinguishable statistically as they possess no structure, and therefore the statistical complexity is low. The fourth class includes automata that generate patterns continuously changing over an unbounded transient.

The fourth class CA existing near the phase transition between periodic and chaotic behavior was shown to be capable of universal computation (Wolfram 1984). These CA support three basic operations (information storage, transmission, and modification) through static, propagating and interacting structures (blinkers, gliders, collisions). Importantly, the patterns produced along the transient are different in terms of generated structure, and in fact, their structural *variability* is highest among all four classes—i.e. the statistical complexity of the class IV automata is highest.

Casti (1991) developed an analogy between the complex (class IV) automata and quasi-periodic orbits in phase space, while pursuing deeper interconnections between CA, dynamical systems, Turing Machines, and formal logic systems—in particular, the complex class IV automata were related to formal systems with undecidable statements (Gödel's Theorem). Such interconnections are also explored in Chap. 15 (Boschetti and Gray 2008), investigating the concept of *emergence* and limits of algorithmic approaches. As has been pointed out by Cooper, there is a "connection between the underlying basic causal structure (the 'design') and the emergent phenomenon", creating a certain level of robustness of the emergence (Cooper 2010). Following this view, one may argue that design is related to causality, and self-organization is related to emergence.

1.4.3 Adoption Roadblocks

Our analysis would be incomplete without discussing a few obstacles preventing a straightforward application of self-organizing systems in industry. The limits of algorithmic approaches to emergence studied in Chap. 15 (Boschetti and Gray 2008) may manifest themselves in many different ways. Firstly, the "numerous interactions among the lower-level components" (Camazine et al. 2001) that are essential for self-organization may often be costly in terms of communication overhead. For example, many decentralized multi-agent (e.g., peer-to-peer) clustering algorithms deployed in sensor networks form stable clusters only after a significant number of messages (Prokopenko et al. 2005a), potentially incurring a prohibitive cost. This highlights even more the role of selecting the most important information channels and communication topologies, reducing local conflicts (assortative noise) and maximizing information transfer.

Secondly (and this is probably the most principled impediment), self-organization results in non-deterministic outcomes. In fact, this is one of its strengths, and as noted earlier, far-from-equilibrium dynamics and symmetry-breaking behavior may lead to stable patterns that can and should be exploited. However, in order to be adopted by industry, non-determinism of self-organizing patterns requires an appropriate verification of the outcomes, and the search for most suitable verification methodology is still open. For example, Chap. 5 (Mathews et al. 2008; Mathews and Durrant-Whyte 2013) investigates self-organizing collaboration within a multi-agent system using analytical techniques, in an attempt to provide a verifiable optimal solution to a decentralised decision problem. Furthermore, Chaps. 8–13 explore different ways to harness the power of self-organization, while staying within some constrained design space (Hogg 2008; Macias and Durbeck 2008, 2013; Schlegel and Kowalczyk 2008; Lee and Zomaya 2008; Tarakanov 2008; Tarakanov and Borisova 2013; Vande Moere 2008).

Finally, a complete self-organizing system would typically depart too strongly from incremental advancements accepted by an industry. A more realistic approach suggests, instead, to deploy hybrid systems (e.g. such as the one described in Chap. 4 (Hoschke et al. 2008, 2013)), where self-organization is used within separate components, providing a convenient mechanism for managing communication overheads and verification requirements. Such hybrid systems are an intermediate step on the path towards complete self-organizing solutions, e.g., a completely self-organizing computation within reaction-diffusion media and memristive excitable automata, explored in Chap. 14 (Adamatzky 2008; Adamatzky and Chua 2013).

In summary, we believe that information-driven evolutionary design can produce self-organizing systems that can be as reliable as traditionally-engineered verifiable (provably-correct) systems, and as resilient as homeostatic biological organisms.

1.5 Epilogue

The topic of this chapter, ‘Design Versus Self-Organization’, has received much attention during the recent years, helping to shape the area of Guided Self-organization (GSO). The declared goal of GSO is to leverage the strengths of self-organization while still being able to direct the outcome of the self-organizing process (Prokopenko 2009; Ay et al. 2012b). Typically, GSO has the following features: (i) an increase in organization (structure and/or functionality) over some time; (ii) the local interactions are not *explicitly* guided by any external agent; (iii) task-independent objectives are combined with task-dependent constraints (Ay et al. 2012b). The last element is the one which incorporates design objectives into the process, purely at the level of global constraints/objectives, and without any direct encoding of local interactions. The process of self-organization is then expected to bring the system to an optimum, *guided* by both generic objectives and design characteristics.

This general approach has been applied in different contexts and at broad scales. Within the field of embodied intelligence, studies of self-organizing perception-action loops and sensorimotor processes of artificial agents have shown that cognition and action self-organize from interactions between brain, body, and environment under some appropriate constraints guiding the process (Polani et al. 2007; Klyubin et al. 2007; Ay et al. 2008; Martius and Herrmann 2012; Capdepuy et al. 2012). A novel information-theoretic framework for distributed computation was developed over the recent years (Lizier et al. 2008c, 2010, 2012b, 2012c), with applications to modular robotics (Lizier et al. 2008a), cascading failures in power grids (Lizier et al. 2009), computational neuroscience (Lizier et al. 2011a), machine learning (Ay et al. 2012a; Boedecker et al. 2012), swarm dynamics (Wang et al. 2012), etc.

Underpinning these successes is a new ability to rigorously quantify fundamental *computational* properties of self-organizing systems that exhibit very highly structured coherent computation in comparison to: (a) ordered systems, which are coherent but minimally interacting, and (b) chaotic systems, whose computations are dominated by high information transfer and interactions eroding any coherence (Lizier et al. 2012b).

Similar treatment extended to random and small-world Boolean networks (Gershenson 2012; Lizier et al. 2008b, 2011b; Wang et al. 2010, 2011; Prokopenko et al. 2011), with some of these studies demonstrating that critical dynamical regimes can also be characterized information-theoretically. In particular, transfer entropy (Schreiber 2000), active information storage (Lizier et al. 2008b, 2012c) and Fisher information (Prokopenko et al. 2011) were utilized in tracing phase transitions and divergent rates of change for the relevant order parameters.

The emerging understanding of critical dynamics in complex networks translates into possibilities for more efficient network design—still within the GSO guidelines. This is exemplified by recent investigations into network motifs and their role in contributing to information dynamics, network evolution and in-network information processing (Piraveenan et al. 2009a, 2009b; Lizier et al. 2012a). This, in turn, leads to more efficient strategies to counter spread of undesirable phenomena in complex networks, such as epidemics, etc. (Bauer and Lizier 2012; Piraveenan et al. 2012).

Thus, one may expect significant developments of specific self-organizing applications within the general framework of GSO.

Acknowledgements The Author would like to thank members of the discussion group on "Entropy and self-organisation in multi-agent systems", particularly Vadim Gerasimov, Nigel Hoschke, Joseph Lizier, George Mathews, Mahendra Piraveenan, and Don Price. The support of the CSIRO Emergence interaction task and the CSIRO Complex Systems Science Theme is gratefully acknowledged. Special thanks to Fabio Boschetti, Tony Farmer, Tomonari Furukawa, Carlos Gershenson, Geoff James, Antonio Lafusa, Ron Li, Abhaya Nayak, Oliver Obst, Daniel Polani, Geoff Poulton, Alex Ryan, Ivan Tanev, Tino Schlegel, Phil Valencia, and X. Rosalind Wang for their insightful comments.

References

Adamatzky, A. (2008). Emergence of traveling localizations in mutualistic-excitation media. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 333–354). London: Springer.

Adamatzky, A., & Chua, L. (2013). Memristive excitable automata: structural dynamics, phenomenology, localizations and conductive pathways. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (2nd ed.). London: Springer.

Ay, N., Bertschinger, N., Der, R., Güttler, F., & Olbrich, E. (2008). Predictive information and explorative behavior of autonomous robots. *The European Physical Journal. B, Condensed Matter Physics*, *63*, 329–339.

Ay, N., Bernigau, H., Der, R., & Prokopenko, M. (2012a). Information-driven self-organization: the dynamical system approach to autonomous robot behavior. *Theory in Biosciences*, *131*, 161–179.

Ay, N., Der, R., & Prokopenko, M. (2012b). Guided self-organization: perception-action loops of embodied systems. *Theory in Biosciences*, *131*, 125–127.

Baldassarre, G. (2008). Self-organization as phase transition in decentralized groups of robots: a study based on Boltzmann entropy. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 129–149). London: Springer.

Baldassarre, G., Parisi, D., & Nolfi, S. (2007). Measuring coordination as entropy decrease in groups of linked simulated robots. In Y. Bar-Yam (Ed.), *Proceedings of the 5th international conference on complex systems (ICCS2004)*.

Bauer, F., & Lizier, J. T. (2012). Identifying influential spreaders and efficiently estimating infection numbers in epidemic models: a walk counting approach. *Europhysics Letters*, *99*(6), 68007.

Bialek, W., Nemenman, I., & Tishby, N. (2001). Complexity through nonextensivity. *Physica. A*, *302*, 89–99.

Boedecker, J., Obst, O., Lizier, J., Mayer, N., & Asada, M. (2012). Information processing in echo state networks at the edge of chaos. *Theory in Biosciences*, *131*, 205–213.

Bonabeau, E., Theraulaz, G., Deneubourg, J.-L., & Camazine, S. (1997). Self-organisation in social insects. *Trends in Ecology & Evolution*, *12*(5), 188–193.

Boschetti, F., & Gray, R. (2008). A Turing test for emergence. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 355–370). London: Springer.

Boschetti, F., Prokopenko, M., Macreadie, I., & Grisogono, A.-M. (2005). Defining and detecting emergence in complex networks. In R. Khosla, R. J. Howlett, & L. C. Jain (Eds.), *Lecture notes* *in computer science: Vol. 3684. Proceedings of the 9th international conference on knowledge-based intelligent information and engineering systems, KES 2005, Part IV*, Melbourne, Australia, 14–16 September 2005 (pp. 573–580). Berlin: Springer.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., & Bonabeau, E. (2001). *Self-organization in biological systems*. Princeton: Princeton University Press.

Capdepuy, P., Polani, D., & Nehaniv, C. L. (2012). Perception-action loops of multiple agents: informational aspects and the impact of coordination. *Theory in Biosciences*, *131*(3), 149–159.

Casti, J. L. (1991). Chaos, Gödel and truth. In J. L. Casti & A. Karlqvist (Eds.), *Beyond belief: randomness, prediction, and explanation in science*. Boca Raton: CRC Press.

Cools, S.-B., Gershenson, C., & D'Hooghe, B. (2008). Self-organizing traffic lights: a realistic simulation. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 41–50). London: Springer.

Cooper, S. B. (2010). Incomputability, emergence and the Turing universe. *Theory Decis. Libr. A*, *46*, 135–153.

Correia, L. (2006). Self-organisation: a case for embodiment. In C. Gershenson & T. Lenaerts (Eds.), *Proceedings of the evolution of complexity workshop at artificial life X: the 10th international conference on the simulation and synthesis of living systems* (pp. 111–116).

Crutchfield, J. P. (1994). The calculi of emergence: computation, dynamics, and induction. *Physica. D*, *75*, 11–54.

Crutchfield, J. P., Mitchell, M., & Das, R. (1998). *The evolutionary design of collective computation in cellular automata* (Technical Report 98-09-080). Santa Fe Institute Working Paper, available at http://www.santafe.edu/projects/evca/Papers/EvDesign.html.

Czap, H., Unland, R., Branki, C., & Tianfield, H. (2005). *Frontiers in artificial intelligence and applications: Vol. 135. Self-organization and autonomic informatics (I)*. Amsterdam: IOS Press.

De Wolf, T., & Holvoet, T. (2005). Emergence versus self-organisation: different concepts but promising when combined. In S. Brueckner, G. D. M. Serugendo, A. Karageorgos, & R. Nagpal (Eds.), *Engineering self-organising systems* (pp. 1–15). Berlin: Springer.

Der, R., Steinmetz, U., & Pasemann, F. (1999). Homeokinesis—a new principle to back up evolution with learning. In *Concurrent systems engineering series* (Vol. 55, pp. 43–47).

Gershenson, C. (2012). Guiding the self-organization of random Boolean networks. *Theory in Biosciences*, *131*(3), 181–191.

Haken, H. (1983a). *Advanced synergetics: instability hierarchies of self-organizing systems and devices*. Berlin: Springer.

Haken, H. (1983b). *Synergetics, an introduction: nonequilibrium phase transitions and self-organization in physics, chemistry, and biology*. New York: Springer. 3rd rev. enl. ed.

Haken, H. (1988). *Information and self-organization: a macroscopic approach to complex systems*. Berlin: Springer.

Heylighen, F. (2000). Self-organization. In F. Heylighen, C. Joslyn, & V. Turchin (Eds.), *Principia Cybernetica web*. Brussels: Principia Cybernetica. Available at http://pespmc1.vub.ac.be/SELFORG.html.

Hofstadter, D. R. (1989). *Gödel, Escher, Bach: an eternal golden braid*. New York: Vintage Books.

Hogg, T. (2008). Distributed control of microscopic robots in biomedical applications. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 147–174). London: Springer.

Hoschke, N., Lewis, C. J., Price, D. C., Scott, D. A., Gerasimov, V., & Wang, P. (2008). A self-organizing sensing system for structural health monitoring of aerospace vehicles. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 51–75). London: Springer.

Hoschke, N., Price, D. C., & Scott, D. A. (2013). Self-organizing sensing of structures: monitoring a space vehicle thermal protection system. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (2nd ed.). London: Springer.

Hubbell, S. P., Johnson, L. K., Stanislav, E., Wilson, B., & Fowler, H. (1980). Foraging by bucket-brigade in leafcutter ants. *Biotropica*, *12*(3), 210–213.

Jirsa, V. K., Jantzen, K. J., Fuchs, A., & Kelso, J. A. (2002). Spatiotemporal forward solution of the eeg and meg using network modeling. *IEEE Transactions on Medical Imaging*, *21*(5), 493–504.

Kauffman, S. A. (2000). *Investigations*. Oxford: Oxford University Press.

Klyubin, A. S., Polani, D., & Nehaniv, C. L. (2004). Organization of the information flow in the perception-action loop of evolved agents. In *Proceedings of 2004 NASA/DoD conference on evolvable hardware* (pp. 177–180). Los Alamitos: IEEE Computer Society.

Klyubin, A. S., Polani, D., & Nehaniv, C. L. (2005). All else being equal be empowered. In M. S. Capcarrère, A. A. Freitas, P. J. Bentley, C. G. Johnson, & J. Timmis (Eds.), *Lecture notes of computer science: Vol. 3630. Proceedings of the 8th European conference on advances in artificial life, ECAL 2005*, Canterbury, UK, 5–9 September 2005 (pp. 744–753). Berlin: Springer.

Klyubin, A., Polani, D., & Nehaniv, C. (2007). Representations of space and time in the maximization of information flow in the perception-action loop. *Neural Computation*, *19*(9), 2387–2432.

Kuramoto, Y. (1984). *Chemical oscillations, waves, and turbulence*. Berlin: Springer.

Langton, C. (1991). Computation at the edge of chaos: phase transitions and emergent computation. In S. Forest (Ed.), *Emergent computation*. Cambridge: MIT Press.

Lee, Y. C., & Zomaya, A. Y. (2008). Immune system support for scheduling. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 247–270). London: Springer.

Liljenström, H., & Svedin, U. (2005). System features, dynamics, and resilience—some introductory remarks. In H. Liljenström & U. Svedin (Eds.), *MICRO-MESO-MACRO: addressing complex systems couplings* (pp. 1–16). Singapore: World Scientific.

Lizier, J. T., Prokopenko, M., Tanev, I., & Zomaya, A. Y. (2008a). Emergence of glider-like structures in a modular robotic system. In S. Bullock, J. Noble, R. Watson, & M. A. Bedau (Eds.), *Proceedings of the eleventh international conference on the simulation and synthesis of living systems (ALife XI)*, Winchester, UK (pp. 366–373). Cambridge: MIT Press.

Lizier, J. T., Prokopenko, M., & Zomaya, A. Y. (2008b). The information dynamics of phase transitions in random Boolean networks. In S. Bullock, J. Noble, R. Watson, & M. A. Bedau (Eds.), *Proceedings of the eleventh international conference on the simulation and synthesis of living systems (ALife XI)*, Winchester, UK (pp. 374–381). Cambridge: MIT Press.

Lizier, J. T., Prokopenko, M., & Zomaya, A. Y. (2008c). Local information transfer as a spatiotemporal filter for complex systems. *Physical Review E*, *77*(2), 026110.

Lizier, J. T., Prokopenko, M., & Cornforth, D. J. (2009). The information dynamics of cascading failures in energy networks. In *Proceedings of the European conference on complex systems (ECCS)*, Warwick, UK (p. 54). ISBN 978-0-9554123-1-8.

Lizier, J. T., Prokopenko, M., & Zomaya, A. Y. (2010). Information modification and particle collisions in distributed computation. *Chaos*, *20*(3), 037109.

Lizier, J. T., Heinzle, J., Horstmann, A., Haynes, J.-D., & Prokopenko, M. (2011a). Multivariate information-theoretic measures reveal directed information structure and task relevant changes in fMRI connectivity. *Journal of Computational Neuroscience*, *30*(1), 85–107.

Lizier, J. T., Pritam, S., & Prokopenko, M. (2011b). Information dynamics in small-world Boolean networks. *Artificial Life*, *17*(4), 293–314.

Lizier, J. T., Atay, F. M., & Jost, J. (2012a). Information storage, loop motifs, and clustered structure in complex networks. *Physical Review E*, *86*(2), 026110.

Lizier, J. T., Prokopenko, M., & Zomaya, A. Y. (2012b). Coherent information structure in complex computation. *Theory in Biosciences*, *131*(3), 193–203.

Lizier, J. T., Prokopenko, M., & Zomaya, A. Y. (2012c). Local measures of information storage in complex distributed computation. *Information Sciences*, *208*, 39–54.

Macias, N. J., & Durbeck, L. J. K. (2008). Self-organizing digital systems. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 177–215). London: Springer.

Macias, N. J., & Durbeck, L. J. K. (2013). Self-organizing computing systems: songline processors. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (2nd ed.). London: Springer.

Martius, G., & Herrmann, J. M. (2012). Variants of guided self-organization for robot control. *Theory in Biosciences*, *131*(3), 129–137.

Mathews, G., & Durrant-Whyte, H. (2013). Decentralised decision making for ad-hoc multi-agent systems. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (2nd ed.). London: Springer.

Mathews, G., Durrant-Whyte, H., & Prokopenko, M. (2008). Decentralized decision making for multi-agent systems. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 77–103). London: Springer.

Miller, J. F., Job, D., & Vassilev, V. K. (2000). Principles in the evolutionary design of digital circuits—Part I. *Genetic Programming and Evolvable Machines*, *1*(1), 8–35.

Mitchell, M., Hraber, P. T., & Crutchfield, J. P. (1993). Revisiting the edge of chaos: evolving cellular automata to perform computations. *Complex Systems*, *7*, 89–139.

Pikovsky, A., Rosenblum, M., & Kurths, J. (2001). *Synchronization: a universal concept in non-linear science*. Cambridge: Cambridge University Press.

Piraveenan, M., Prokopenko, M., & Zomaya, A. Y. (2009a). Assortativeness and information in scale-free networks. *The European Physical Journal. B, Condensed Matter Physics*, *67*, 291–300.

Piraveenan, M., Prokopenko, M., & Zomaya, A. Y. (2009b). Assortativity and growth of Internet. *The European Physical Journal. B, Condensed Matter Physics*, *70*, 275–285.

Piraveenan, M., Prokopenko, M., & Hossain, L. (2012). Percolation centrality: quantifying graph-theoretic impact of nodes during percolation in networks. *PLoS ONE*, *8*(1), e53095. doi:10.1371/journal.pone.0053095

Polani, D. (2003). Measuring self-organization via observers. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, & J. Ziegler (Eds.), *Advances in artificial life—proceedings of the 7th European conference on artificial life (ECAL)*, Dortmund (pp. 667–675). Heidelberg: Springer.

Polani, D. (2008). Foundations and formalizations of self-organization. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 19–37). London: Springer.

Polani, D., Sporns, O., & Lungarella, M. (2007). How information and embodiment shape intelligent information processing. In M. Lungarella, F. Iida, J. Bongard, & R. Pfeifer (Eds.), *Lecture notes in computer science: Vol. 4850. Proceedings of the 50th anniversary summit of artificial intelligence*, New York (pp. 99–111). Berlin: Springer.

Prigogine, I. (1980). *From being to becoming: time and complexity in the physical sciences*. San Francisco: Freeman.

Prokopenko, M. (2009). Guided self-organization. *HFSP Journal*, *3*(5), 287–289.

Prokopenko, M., Piraveenan, M., & Wang, P. (2005a). On convergence of dynamic cluster formation in multi-agent networks. In M. S. Capcarrère, A. A. Freitas, P. J. Bentley, C. G. Johnson, & J. Timmis (Eds.), *Lecture notes in computer science: Vol. 3630. Proceedings of the 8th European conference on advances in artificial life, ECAL 2005*, Canterbury, UK, 5–9 September 2005 (pp. 884–894). Berlin: Springer.

Prokopenko, M., Wang, P., Foreman, M., Valencia, P., Price, D. C., & Poulton, G. T. (2005b). On connectivity of reconfigurable impact networks in ageless aerospace vehicles. *Robotics and Autonomous Systems*, *53*(1), 36–58.

Prokopenko, M., Wang, P., & Price, D. C. (2005c). Complexity metrics for self-monitoring impact sensing networks. In J. Lohn, D. Gwaltney, G. Hornby, R. Zebulum, D. Keymeulen, & A. Stoica (Eds.), *Proceedings of 2005 NASA/DoD conference on evolvable hardware (EH-05)*, Washington DC, USA, 29 June–1 July 2005 (pp. 239–246). Los Alamitos: IEEE Computer Society.

Prokopenko, M., Wang, P., Price, D. C., Valencia, P., Foreman, M., & Farmer, A. J. (2005d). Self-organizing hierarchies in sensor and communication networks. *Artificial Life*, *11*(4), 407–426. Special Issue on Dynamic Hierarchies.

Prokopenko, M., Gerasimov, V., & Tanev, I. (2006a). Evolving spatiotemporal coordination in a modular robotic system. In S. Nolfi, G. Baldassarre, R. Calabretta, J. C. T. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, & D. Parisi (Eds.), *Lecture notes in computer science: Vol. 4095. From animals to animats 9: 9th international conference on the simulation of adaptive behavior (SAB 2006)*, Rome, Italy, 25–29 September 2006 (pp. 558–569).

Prokopenko, M., Gerasimov, V., & Tanev, I. (2006b). Measuring spatiotemporal coordination in a modular robotic system. In L. Rocha, L. Yaeger, M. Bedau, D. Floreano, R. Goldstone, & A. Vespignani (Eds.), *Artificial life X: proceedings of the 10th international conference on the simulation and synthesis of living systems* (pp. 185–191). Bloomington: MIT Press.

Prokopenko, M., Poulton, G. T., Price, D. C., Wang, P., Valencia, P., Hoschke, N., Farmer, A. J., Hedley, M., Lewis, C., & Scott, D. A. (2006c). Self-organising impact sensing networks in robust aerospace vehicles. In J. Fulcher (Ed.), *Advances in applied artificial intelligence* (pp. 186–233). Hershey: Idea Group.

Prokopenko, M., Boschetti, F., & Ryan, A. J. (2009). An information-theoretic primer on complexity, self-organization, and emergence. *Complexity*, *15*(1), 11–28.

Prokopenko, M., Lizier, J. T., Obst, O., & Wang, X. R. (2011). Relating Fisher information to order parameters. *Physical Review E*, *84*(4), 041116.

Sahin, E., & Spears, W. M. (2004). Revised selected papers. In *Lecture notes in computer science: Vol. 3342*. *Proceedings of SAB-2004 international workshop on swarm robotics*, Santa Monica, CA, USA, 17 July 2004.

Scaruffi, P. (2003). *Thinking about thought: a primer on the new science of mind, towards a unified understanding of mind, life and matter*. San Jose: Writers Club Press.

Schlegel, T., & Kowalczyk, R. (2008). Self-organizing nomadic services in grids. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 217–245). London: Springer.

Schreiber, T. (2000). Measuring information transfer. *Physical Review Letters*, *85*(2), 461–464.

Shalizi, C. (2001). *Causal architecture, complexity and self-organization in time series and cellular automata*. PhD thesis, University of Michigan. Available at http://www.cscs.umich.edu/~crshalizi/thesis/.

Shalizi, C. R., Shalizi, K. L., & Haslinger, R. (2004). Quantifying self-organization with optimal predictors. *Physical Review Letters*, *93*(11), 118701-1-4.

Tanev, I. (2008). Learning mutation strategies for evolution and adaptation of a simulated snakebot. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 105–127). London: Springer.

Tanev, I., Ray, T., & Buller, A. (2005). Automated evolutionary design, robustness, and adaptation of sidewinding locomotion of a simulated snake-like robot. *IEEE Transactions on Robotics*, *21*, 632–645.

Tarakanov, A. O. (2008). Formal immune networks: self-organization and real-world applications. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 271–290). London: Springer.

Tarakanov, A. O., & Borisova, A. V. (2013). Formal immune networks: self-organization and real-world applications. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (2nd ed.). London: Springer.

Vande Moere, A. (2008). A model for self-organizing data visualization using decentralized multi-agent systems. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 291–324). London: Springer.

Volman, M. J. M. (1997). *Rhythmic coordination dynamics in children with and without a developmental coordination disorder*. PhD dissertation, University of Groningen, available at http://irs.ub.rug.nl/ppn/163776687.

Wagner, A. (2005). *Robustness and evolvability in living systems*. Princeton: Princeton University Press.

Wang, X. R., Lizier, J. T., & Prokopenko, M. (2010). A Fisher information study of phase transitions in random Boolean networks. In H. Fellermann, M. Dörr, M. M. Hanczyc, L. L. Laursen, S. Maurer, D. Merkle, P.-A. Monnard, K. Stoy, & S. Rasmussen (Eds.), *Proceedings of the 12th international conference on the synthesis and simulation of living systems (Alife XII)*, Odense, Denmark (pp. 305–312). Cambridge: MIT Press.

Wang, X. R., Lizier, J. T., & Prokopenko, M. (2011). Fisher information at the edge of chaos in random Boolean networks. *Artificial Life*, *17*(4), 315–329.

Wang, X. R., Miller, J. M., Lizier, J. T., Prokopenko, M., & Rossi, L. F. (2012). Quantifying and tracing information cascades in swarms. *PLoS ONE*, *7*(7), e40084.

Woese, C. R. (2004). A new biology for a new century. *Microbiology and Molecular Biology Reviews*, *68*(2), 173–186.

Wolfram, S. (1984). Universality and complexity in cellular automata. *Physica D*, *10*.

Wuensche, A. (1999). Classifying cellular automata automatically: finding gliders, filtering, and relating space-time patterns, attractor basins, and the z parameter. *Complexity*, *4*(3), 47–66.

Zambonelli, F., & Rana, O. F. (2005). Self-organization in distributed systems engineering. Special Issue of *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, *35*(3).

Chapter 2
Foundations and Formalizations of Self-Organization

Daniel Polani

2.1 Introduction

In the study of complex systems, the relevance of the phenomenon of *self-organization* is ubiquitous. For example the stripe formation in morphogenesis (Meinhardt 1972, 1982), reaction-diffusion automata (Turing 1952), the reorganization of a self-organizing Kohonen map, the seemingly effortless distributed organization of work in an ant colony, the formation of flows in pedestrian movement patterns (Helbing et al. 2005), the maintenance and creation of the complexities involved with the maintenance of life in living cells all produce a behaviour that, in one way or another, can be called "organized" and if the source of organization is not explicitly identified outside of the system, it can be called "*self*-organized".

Strangely enough, as much agreement as there is about whether self-organization is present or absent in a system on visual inspection, as little agreement exists concerning the precise meaning of the word. In other words, while the phenomenology of the phenomenon is pretty much agreed upon, its formal foundations are not.

Among other problems, this causes a certain amount of confusion. For instance, the difference between the notions of emergence and self-organization are being strongly emphasized (Shalizi 2001), notwithstanding the frequent co-occurrence of these notions. On the other hand, without a clean formal definition of self-organization and emergence, it is difficult to make strong points in favour (or against) a separation of these notions.

With recent interest in the exploitation of phenomena of self-organization for engineering and other applications, the importance of characterizing and understanding the phenomenon of self-organization has even increased. It is no longer sufficient to characterize a system in "ivory-tower" fashion to be in one group or another according to some human classification. Rather it becomes necessary to

D. Polani (✉)
Adaptive Systems Research Group, Department of Computer Science, University of Hertfordshire, Hatfield, UK
e-mail: d.polani@herts.ac.uk

work towards a predictable and structured theory that will also allow to make useful predictions about the performance of a system. The advent of nanotechnology and bio-inspired engineering architectures increases the immediate practical relevance of understanding and characterizing self-organization.

In view of the various streams and directions of the field of self-organization, it is beyond the present introductory chapter to review all the currents of research in the field. Rather, the aim of the present section is to address some of the points judged as most relevant and to provide a discussion of suitable candidate formalisms for the treatment of self-organization. In the author's opinion, discussing formalisms is not just a vain exercise, but allows one to isolate the essence of the notion one wishes to develop. Thus even if one disagrees with the path taken (as is common in the case of not yet universally agreed upon formal notions), starting from operational formalisms helps to serve as a compass guiding one towards notions suitable for one's purposes. This is the philosophy of the present chapter.

The chapter is structured as follows: in Sect. 2.2, we will present several central conceptual issues relevant in the context of self-organization. Some historical remarks about related relevant work are then done in Sect. 2.3. To illustrate the setting, a brief overview over some classical examples for self-organizing processes is given in Sect. 2.4. In Sects. 2.5 and 2.6, introduces the two main information-theoretic concepts of self-organization that the present chapter aims to discuss. One concept, based on the ϵ-machine formalism by Crutchfield and Shalizi, introduces self-organization as an increase of (statistical) complexity with time. The other concept will suggest measuring self-organization as an increase of mutual correlations (measured by multi-information) between different components of a system. In Sect. 2.7, finally, important properties of these two measures as well as their distinctive characteristics (namely their power to identify temporal versus compositional self-organization) will be discussed, before Sect. 2.8 gives some conclusive remarks.

2.2 General Comments

In the vein of the comments made above, the present paper does not attempt to answer the question: "what is self-organization?" Rather, the question will be "where do we agree self-organization exists?" or "what are candidate characterizations of self-organization?". Thus, rather than attempting to give ultimate answers to that question, a number of suggestions will be presented that can form a starting point for the development of the reader's own notion.

The distinction of self-organization from emergence is emphasized time and time again (Shalizi 2001); this is sometimes confusing to outsiders, given that both often appear in similar contexts and that there is no universally accepted formal definition for each of them. This distinction emphasizes, though, that there is a general desire to have them carry different meanings.

Self-organization, the main focus of the present chapter,[1] is a phenomenon under which a dynamical system exhibits the tendency to create organization "out of itself", without being driven by an external system, in particular, not in a "top-down" way. This requires clarification of several questions:

1. What is meant by organization?
2. How and when to distinguish system and environment?
3. How can external drives be measured?
4. What does top-down mean?

Question 1 is clearly related to the organizational concept of *entropy*. However, it is surprisingly not straightforward to adapt entropy to be useful for measuring self-organization, and some additional efforts must be made (Polani 2003). This is the main question the present chapter concentrates upon. All the other questions are only briefly mentioned here to provide the reader with a feel of what further issues in the context of formalization of self-organization could and should be addressed in future.

Question 2 is, basically, about how one defines the boundaries of the system; this is related with the question of autonomy (Bertschinger et al. 2006). Once they are defined, we can ask ourselves all kinds of questions about the system under investigation and its environment, its "outside". A complication is brought into the discussion through the fact that, if organization is produced in the "inner" system out of itself (the "self" in self-organization), in a real physical system, disorder has to be produced in the outside world, due to the conservation of phase space volume (Adami 1998).

However, for many so-called self-organizing systems the physical reality is quite detached, i.e. conservation or other thermodynamic laws are not (and need not be) part of the model dynamics, so Landauer's principles (Landauer 1961; Bennett and Landauer 1985) are irrelevant in the general scenario: for instance, a computer emulating a self-organizing map produces much more total heat in its computation than the minimum amount demanded by Landauer's principle due to the entropy reduction of the pure computation. In other words, the systems under consideration may be arbitrarily far away from thermal equilibrium and worse, there may be a whole hierarchy of different levels of organization whose constraints and invariants have to be respected before one can even consider coming close to the Landauer limit.[2]

Therefore, unless we are aiming for understanding nano-systems where issues of Landauer's principle could begin to play a role, we can and will ignore issues of the "compulsory" entropy production of a real physical system that exhibits self-organization. In particular, the systems we will consider in the following are general dynamical systems. We will not require them to be modeled in a thermodynamically or energetically consistent way.

[1] Emergence is briefly discussed in Sects. 2.5.2 and 2.6.1.

[2] As an example, the energy balance of real biological computation process will operate at the ATP metabolism level and respect its restrictions—but this is still far off the Landauer limit.

As for question 3, it is not as straightforward to respond to, a difficulty that is conceded in Shalizi et al. (2004). There, it has been suggested to study causal inference as a possible formalism to investigate the influence of an environment onto a given system. In fact, the concept of *information flow* has been recently introduced to address exactly this question (Ay and Wennekers 2003; Klyubin et al. 2004; Ay and Krakauer 2007; Ay and Polani 2008), providing an information-theoretic approach to measure causal inference. At this point these notions are still quite fresh and not much is known about their possible relevance for characterizing self-organizing systems, although it will be an interesting avenue for future work.

Question 4, again introduces an interesting and at the same time unclear notion of "top-down". Roughly, top-down indicates a kind of downward causation (Emmeche et al. 2000), where one intervenes to influence some coarse-grained, global, degrees of freedom as to achieve a particular organizational outcome; or else, the external experimenter "micro-manages" the system into a particular state. For this latter view, one could consider using a formalization of an agent manipulating its environment (Klyubin et al. 2004). This is again outside of the scope of the present paper. Nevertheless, it is hoped that this section's brief discussion of general conceptual issues highlights some related open questions of interest that may prove amenable to treatment by a consistent theoretical framework.

2.3 Related Work and Historical Remarks

Shalizi et al. (2004) track back the first use of the notion of "self-organizing systems" to Ashby (1947). The bottom-up cybernetic approach of early artificial intelligence (Walter 1951; Pask 1960) devoted considerable interest and attention to the area of self-organizing systems; many of the questions and methods considered relevant today have been appropriately identified almost half a century ago (e.g. Yovits and Cameron 1960).

The notions of *self-organization* and the related notion of *emergence* form the backbone for the studies of dynamical hierarchies, and in particular those types of dynamics that lead to climbing the ladder of complexity as found in nature. Notwithstanding the importance and frequent use of these notions in the relevant literature, a both precise and useful mathematical definition remains elusive. While there is a high degree of intuitive consensus on what type of phenomena should be called "self-organizing" or "emergent", the prevailing strategy of characterization is along the line of "I know it when I see it" (Harvey 2000).

Specialized formal literature often does not go beyond pragmatic characterizations; e.g. Jetschke (1989) defines a system as undergoing a self-organizing transition if the symmetry group of its dynamics changes to a less symmetrical one (e.g. a subgroup of the original symmetry group, Golubitsky and Stewart 2003), typically occurring at phase transitions (Reichl 1980). This latter view relates self-organization to phase transitions. However, there are several reasons to approach the definition of self-organization in a different way. The typical complex system is not in thermodynamic equilibrium (see also Prigogine and Nicolis 1977). One possible extension of the formalism is towards nonequilibrium thermodynamics, identifying phase transitions by *order parameters*. These are quantities that characterize the "deviation" of the system in a more organized state (in the sense of Jetschke) from the system in a less organized state, measured by absence or presence of symmetries. Order parameters have to be constructed by explicit inspection of the system since a generic approach is not available, although an ϵ-machine based approach such as in Shalizi (2001) seems promising. Also, in complex systems, one can not expect the a priori existence or absence of any symmetry to act as universal indicators for self-organization; in general such a system will exhibit, at best, only approximate or imperfect symmetries, if at all. Without a well-founded concept of characterizing such imperfect "soft" symmetries, the symmetry approach to characterize self-organization is not sufficient to characterize general complex systems.

2.4 Examples for Self-Organization

We now consider a number of examples of systems which are typically regarded as self-organizing. Since the field lacks a consensus on suitable formal definitions, it is helpful to consider examples of the phenomenon at hand, where there is less controversy whether or not they exhibit the desired behaviour.

2.4.1 Bifurcation

Consider a dynamical system with state $\mathbf{x}(t) \in \mathbb{R}^n$ at time t, whose state dynamics governed by a differential equation

$$\dot{\mathbf{x}} = F(\mathbf{x}, \mu) \tag{2.1}$$

where $F : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is a smooth function, and $\mu \in \mathbb{R}$ is a so-called *bifurcation parameter*, and the dot denotes the usual time derivative. For a fixed μ, this defines a particular dynamical system which, amongst other, exhibits a particular *fixed point* profile, i.e. a set of points $\{\mathbf{x}^0 \in \mathbb{R}^n \mid F(\mathbf{x}^0) = 0\}$. The existence of fixed points engenders a possibly highly intricate structure of the system state space $\mathbb{R}^n$. Of particular importance are the so-called *stable* and *unstable manifolds*. The *stable manifold* of a fixed point $\mathbf{x}^0$ is the continuation (forward in time) of the local eigenspaces of the Jacobian $DF|_{\mathbf{x}^0}$ for negative eigenvalues, the *unstable manifold* is the continuation (backward in time) for positive eigenvalues (see Jetschke 1989, or any good book about dynamical systems for details). The important part of this message is that the structure of the invariant (stable and unstable) manifolds structures the state space in a characteristic way. A very simple example is shown in Fig. 2.1: even in this simple example, the state space is split into four regions. With

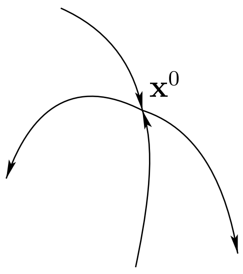


Fig. 2.1 Stable and unstable manifold of a fixed point $\mathbf{x}^0$

a more intricate fixed points (or attractor) structure, that profile can be quite more complex.

The importance of above observation stems from a number of facts. In the example above (as shown in Fig. 2.1), there are only positive or negative eigenvalues of the Jacobian $DF|_{\mathbf{x}^0}$. However, if we consider μ a parameter that is scanned through, the eigenvalue spectrum of the Jacobian changes smoothly, and eigenvalues may change sign, i.e. may crossing the 0 level. Generically, an eigenvalue changing sign on changing μ will travel with nonzero speed through 0, i.e. for which $DF_\mu|_{\mathbf{x}^0} \neq 0$, where DF_μ is the partial derivative of the Jacobian with respect to μ. If this is the case, the number or character of the fixed points may change, sometimes dramatically, and with it the whole split of the space into attractor regions of different character. This process is known as *bifurcation*. In systems which have a fast dynamics F parameterized by a slow-varying (and perhaps externally controlled) parameter μ, the appearance of new fixed points is often interpreted as a process of self-organization.

2.4.2 Synergetics

The concept of a slow varying parameter has been made part of the above analysis by a number of approaches, most notably the synergetics approach (Haken 1983), but is also known under the name of *slow manifold* and *fast foliation* (Mees 1981). If we consider a dynamical system where the Jacobian of F has a few eigenvalues very close to 0, and a large number of strongly negative eigenvalues, those components of x which fall into the degrees of freedom of the strongly negative eigenvalues will vanish quickly, reducing the essential dynamics of the system to the low-dimensional submanifold of the whole system which corresponds to the "slow" degrees of freedom of the system. In the more colourful language of synergetics, these "slow" degrees of freedom are the "master" modes that "enslave" the fast, quickly decaying modes belonging to the strongly negative eigenvalues.

Synergetics provided an historically early formal and quantitative approach for the treatment of self-organization phenomena by decomposing a possibly large system into hierarchically separate layers of dynamics. It has been successfully applied to a number of physical systems and models, including laser pumping, superconductivity, the Ginzburg-Landau equations and the pattern formation and the Bénard instabilities in fluids (Haken 1983). However, it only works properly under certain conditions (namely the particular structure of the eigenvalue spectrum) and there are self-organization phenomena it fails to capture fully (Spitzner and Polani 1998).

2.4.3 Pattern Formation in Spatial Media

To define the dynamical system concept from Sects. 2.4.1 and 2.4.2 one requires the concept of for smooth manifolds, i.e. a space with consistent differentiable structures. If one adds the requirement that F obeys given symmetries, i.e. that there is a symmetry group $\varGamma$ such that $\gamma \in \varGamma$ operates on $\mathbb{R}^n$ and (2.1) obeys

$$(\dot{\gamma \mathbf{x}}) = F(\gamma \mathbf{x}, \mu)$$

for all $\gamma \in \varGamma$, then it can be shown this imposes restrictions on the solution space, including the bifurcation structure (Golubitsky and Stewart 2003).

A special, but important case is a spatial symmetry governing the state space of the dynamics. In the most generic case, the state space is not the finite dimensional space $\mathbb{R}^n$, but rather the space $C^m(\mathbb{R}^k, \mathbb{R})$ of sufficiently smooth functions on $\mathbb{R}^k$, and the symmetries are the Euclidean symmetries (translations and orthogonal rotations) on $\mathbb{R}^k$, and (2.1) actually becomes a partial differential equation.[3] In a discrete approximation one can replace the space $\mathbb{R}^k$ by a *lattice* $L = \{\sum_{i=1}^k l_i \mathbf{v}_i \mid l_i \in \mathbb{Z}\}$ for linear independent $\mathbf{v}_i \in \mathbb{R}$ (Hoyle 2006), and thus reobtain a finite-dimensional version of (2.1), this time the n components of space not anymore forming an unstructured collection, but rather being organized as a lattice and subject to its symmetries. Here, the system dynamics, together with the symmetries governing the system give rise to particular stable states and attractors where, due to their easily visualizable spatial structure it is easy to detect phenomena of self-organization.

A classical example for such a model is Turing's reaction-diffusion model (Turing 1952). He was among the first to study the dynamics of reaction-diffusion systems as possible model for computation; his particular interest was to study pattern formation in biological scenarios. In a time where there was still a debate which would be the most adequate model for computation, Turing's suggestion of a spatially organized computational medium with an activator and an inhibitor substance with differing diffusion coefficients, provides a wide range of spatial organization dynamics. Depending on the chemical dynamics, there are different instances of reaction-diffusion machines. Figure 2.2 shows some Turing patterns emerging from having an activator and an inhibitor with concentrations a, b, respectively, computed from the definition of their rates of change

$$\frac{\partial a}{\partial t} = \delta_1 \Delta a + k_1 a^2/b - k_2 a \tag{2.2}$$

[3]Here we ignore technical details necessary to properly define the dynamics.

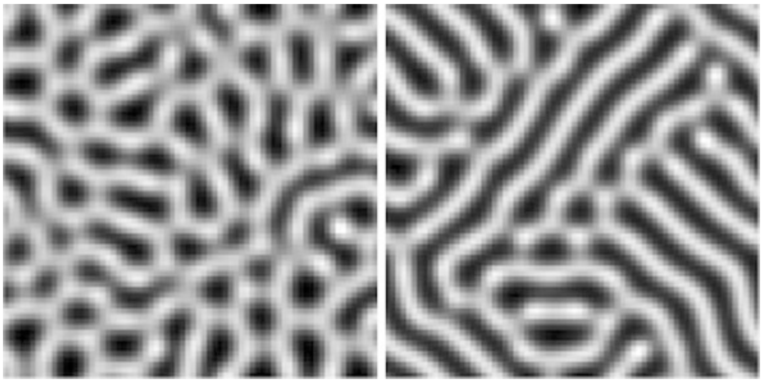


Fig. 2.2 Turing patterns of the concentration a of the activator, as emerging from (2.2) for different parameters k. See Bar-Yam (1997) for discussion how to obtain different patterns

$$\frac{\partial b}{\partial t} = \delta_2 \Delta b + k_3 a^2 - k_4 b \tag{2.3}$$

in the discrete approximation of a square lattice (Bar-Yam 1997). The starting configuration were randomly initialized concentration fields for a and b. The simulations show the dynamics is constrained to produce structured patterns from virtually arbitrary initial states.

Other reaction-diffusion systems exhibiting spatial self-organization phenomena are, for instance the Belousov-Zhabotinsky reaction which can be implemented in vitro. Reaction-diffusion processes are believed to influence the morphogenesis processes in living beings (Meinhardt 1982) and, as such, of central importance to understand how morphological complexity can be obtained by "unpacking" the relatively compact genotype.

2.5 Information-Theoretic Approaches to Self-Organization

The models from Sect. 2.4 have in common that they require the dynamical system to "live" on a differentiable manifold to describe self-organization. In addition, the synergetics model of self-organization requires a particular grouping of the eigenvalues of the Jacobian and the pattern formation models require the presence of a spatially structured state space. These are relatively specific requirements. In a unified treatment of self-organization, it would be very limiting to exclude self-organization in discrete, not differentiable, worlds. Similarly, it would be inappropriate to assume that systems must have a Euclidian spatial organization similar to reaction-diffusion systems. It is easy to envisage scenarios where a system may possess other topological/structural properties, such as social or food web networks.

In addition, the example systems from Sect. 2.4 were implicitly assumed to be deterministic, which is usually far too strong an assumption. Neither this assumption nor the assumption from the last paragraph needs to hold: one can imagine self-organization in an agent system (such as relevant for engineering problems) which is neither deterministic nor organized on a regular spatial structure, and certainly nothing can be assumed in terms of distributions of eigenvalues close to fixed points (if these at all exist).

Can anything at all be analyzed in such structurally impoverished scenarios? Indeed, it turns out that still a lot can be said with much less structure, and the toolbox of choice is information theory. In the following, we will outline two approaches to model self-organization using information theory.

Information theory operates on probability distributions. These require only minimal structure (a probability measure) on the space of interest, and make no assumption about differentiability or spatial structure. Information theory has crystallized as a promising common language for the study of general systems, to tackle issues of complex phenomena exhibiting a wide variety of seemingly incompatible properties.

2.5.1 *Notation*

Due to space limitations, the formalization of the exposition is restricted to a minimum. Consider a random variable X assuming values $x \in \mathcal{X}$, $\mathcal{X}$ the set of possible values for X. For simplicity, assume that $\mathcal{X}$ is finite. Define the *entropy* of X by

$$H(X) := -\sum_{x \in \mathcal{X}} p(x) \log p(x),$$

the *conditional entropy* of Y as

$$H(Y|X) := \sum_{x \in \mathcal{X}} p(x) H(Y|X = x)$$

with

$$H(Y|X = x) := -\sum_{y \in \mathcal{Y}} p(y|x) \log p(y|x)$$

for $x \in \mathcal{X}$. The *joint entropy* of X and Y is the entropy $H(X, Y)$ of the random variable (X, Y). The *mutual information* of random variables X and Y is $I(X; Y) := H(Y) - H(Y|X) = H(X) + H(Y) - H(X, Y)$. A generalization is the *intrinsic information*: for random variables $X_1, \ldots, X_k$, the intrinsic or multi-information is

$$I(X_1; \ldots; X_k) := \left[\sum_{i=1}^{k} H(X_i)\right] - H(X_1, \ldots, X_k).$$

This notion is also known e.g. as *integration* in Tononi et al. (1994). Similar to the mutual information, it is a measure for the degree of dependence between the different X_i.

2.5.2 Self-Organization as Increasing Statistical Complexity

One influential approach to study complex systems and the notions of self-organization and emergence is based on the ϵ-machine formalism which provides a model to describe complex temporal processes (Crutchfield and Young 1989). Using this formalism, Shalizi (2001) develops a quantifiable notion of self-organization. In the following, we briefly describe the ϵ-machine formalism and the ensuing model for self-organization.

Consider a stochastic process (with, say, infinite past and future):

$$\mathbf{X} = \dots X^{(t-3)}, X^{(t-2)}, X^{(t-1)}, X^{(t)}, X^{(t+1)}, X^{(t+2)}, X^{(t+3)}, \dots.$$

Denote the (ordered) sequence of variables up to $X^{(t)}$ by $\overleftarrow{X}$ (*past*) and the sequence of variables from $X^{(t+1)}$ upwards by $\overrightarrow{X}$ (*future*). Consider the equivalence relation that identifies all pasts $\overleftarrow{x}$ for which the probability distribution $P(\overrightarrow{X} \mid \overleftarrow{x})$ of the possible futures is exactly the same. This equivalence relation partitions the pasts into disjoint sets, which, for the sake of simplicity, we name $\tilde{x}$. Any past $\overleftarrow{x}$ is member of exactly one equivalence class $\tilde{x}$.

To construct an ϵ-machine from a given process $\mathbf{X}$, define an automaton such that its states are identified one-to-one by the equivalence classes $\tilde{x}$ arising from above procedure. When a transition from t to $t+1$ is made, it means replacing a past $\dots X^{(t-3)}, X^{(t-2)}, X^{(t-1)}, X^{(t)}$ by a past $\dots X^{(t-2)}, X^{(t-1)}, X^{(t)}, X^{(t+1)}$, and thus is acts as a transition from an equivalence class $\tilde{x}$ to an equivalence class $\tilde{x}'$, corresponding to the new past. Together with labeling the transition by the realization $x^{(t+1)}$ of $X^{(t+1)}$, this defines the automaton.

The ϵ-machine, when it exists, acts as the unique minimal maximally predictive model of the original process (Shalizi and Crutchfield 2002), including highly non-Markovian processes which may contain a significant amount of memory.[4] It allows to define the concept of *statistical complexity* as the entropy $H(\tilde{X}) = -\sum_{\tilde{x}} p(\tilde{x}) \log p(\tilde{x})$ of the states of the ϵ-machine. This is a measure of the memory required to perform the process $\mathbf{X}$.

Note that the statistical complexity is, in general, different from another important quantity, known, among other, as *excess entropy* and which is given by $\eta(\mathbf{X}) := I(\overleftarrow{X}; \overrightarrow{X})$ (see e.g. Grassberger 1986). One always has $\eta(\mathbf{X}) \leq H(\tilde{X})$ (Shalizi 2001). The interpretational distinction between statistical complexity and excess entropy is subtle. Of the selection of interpretations available, the author prefers the view inspired by the "information bottleneck" perspective (Tishby et al. 1999; Shalizi and Crutchfield 2002): The excess entropy is the actual information

[4]Note that, in general, the construction of an ϵ-machine from the visible process variables $\mathbf{X}$ is not necessarily possible, and the reader should be aware that the Shalizi/Crutchfield model is required to fulfil suitable properties for the reconstruction to work. I am indebted to Nihat Ay and Wolfgang Löhr for pointing this out to me.

contained in the complete past (for a given time) about the complete future *as it could be reconstructed if the complete past were available as a whole*. As opposed to that, to obtain the statistical complexity one has to force this information through the "bottleneck" given by the ϵ-machine state at the *present* time slice which has to provide a sufficient statistics about the past concerning the future constrained to the current time. Because of the constraint of this bottleneck to the present time slice it is, in general, less parsimonious in description than the "idealized" excess entropy that, in principle, assumes availability of the whole process (past and future) to produce its prediction. In the bottleneck picture, we have the sequence

$$\tilde{X} \longleftarrow \overleftarrow{X} \longleftrightarrow \overrightarrow{X}$$

where the left arrow indicates the projection of $\overleftarrow{X}$ to the ϵ-machine state and the arrows between the past and future variables indicate their informational relation. The process of "squeezing" their mutual information into the bottleneck variable $\tilde{X}$ produces in general a variable with a larger entropy than the actual mutual information content between $\overleftarrow{X}$ and $\overrightarrow{X}$ (this is a general property of the information bottleneck, of which the relation between statistical complexity and excess entropy is just a special case).

In computing the ϵ-machine, the picture of an infinite time series is idealized. In the empirical case one will consider finite time series, giving rise to a "localized" ϵ-machine, operating on a certain window size. Here, one will expect to encounter a slow drift superimposed on the fast dynamics of the process which will slowly change the localized ϵ-machine with the passing of time. Shalizi (2001) calls a system self-organizing if this statistical complexity grows with time. We will call this flavour of self-organization *SC-self-organization* ("SC" for statistical complexity).

This approach basically considers organization to be essentially the same as complexity. In this model, self-organization is an intrinsic property of the system and unambiguously measurable. In particular, this approach makes the relation to emergence unambiguously clear, as Shalizi gives a definition of emergence based on the ϵ-machine notion in the same work. The essence is that in the ϵ-machine perspective emergence is present if there is a coarse-grained description of the system that is more predictively efficient than the original description of the process, i.e. if it has a higher ratio $\eta(\mathbf{X})/H(\tilde{X})$ of excess entropy vs. statistical complexity, a better ratio between the total amount of process information that ideally needs to be processed and the process memory that is actually necessary to achieve that.

This approach to emergence is descriptive, as it characterizes a property of the particular description (i.e. perspective or "coordinate system") through which one looks into a system. As opposed to that, self-organization in the ϵ-machine model is a purely intrinsic property of the system. Through this split into description and intrinsic properties, Shalizi (2001) argues that while emergence may allow to simplify descriptions of a system, there may be cases of self-organization which humans do not recognize as such because there is no appropriate simplified (emergent) coordinate system through which the self-organization would become apparent. It is only visible through the ϵ-machine construction. This provides a transparent picture how self-organization and emergence turn out to be two mathematically distinct concepts that represent different aspects of a system. While this is a motif that one finds repeatedly emphasized in the scientific discourse, it is rarely formulated in an as compelling and crisp language.

One interesting aspect of the ϵ-machine view is how it reflects the bifurcation concept from Sect. 2.4.1. Consider as process an iterator map for $t \to \infty$. As long as there is only a single fixed point attractor, the ϵ-machine will (asymptotically) have only one state. As a bifurcation into two fixed point attractors occurs, these two attractors will be reflected by the ϵ-machine. With the bifurcation behaviour becoming more intricate (as would happen, say, in the logistic map example with an adiabatically slowly growing bifurcation parameter), the ϵ-machine also grows in complexity. In this way, the ϵ-machine can grow significantly in size and with it the statistical complexity.

2.5.3 *Observer-Induced Self-Organization*

Conceptually, pattern formation which we gave in Sect. 2.4.3 as a prominent example of self-organization, does not fit smoothly into the picture of growing statistical complexity. One reason for that is that statistical complexity by its very foundations is a concept that operates on an process that has no a priori structure on the $X^{(t)}$, except for the ordered-ness (and, implied, directed-ness) of time.

Quite different from that, spatial pattern formation inextricably requires a spatial structure on its state space $\mathcal{X}$. The patterns that develop during the experiments form in space, and space has strong structural constraints. For this purpose, in Shalizi (2001), a spatial ϵ-machine is developed to deal specifically with this problem. Thus, the spatial structure is brought in explicitly as a part of the model.

An alternative approach to model self-organization using information theory is suggested in Polani (2003). This approach no longer considers an unstructured dynamical system on its own, but adds the concept of an *observer* which acts as a particular "coordinate system" through which the given system is represented at a given time step. For this model of self-organization, an observer or coordinate system needs to be specified *in addition* to the dynamics of the system. The suspicion that observers may be of importance to characterize complex systems has been voiced quite a few times in the past (Crutchfield 1994; Harvey 2000; Baas and Emmeche 1997; Rasmussen et al. 2001). In formalizing this idea here, we follow the particular flavour from Polani (2003).

2.6 Organization via Observers

A *(perfect) observer* of a (random) variable X is a collection $X_1, X_2, \ldots, X_k$ of random variables allowing full reconstruction of X, i.e. for which $H(X|X_1, X_2, \ldots, X_k)$

vanishes. We define the *organization information* with respect to the observer as the multi-information $I(X_1; \dots; X_k)$. We call a system *self-organizing* (with respect to the given observer) if the organization information increase with respect to the observer variables is positive as the system dynamics progresses with time. $I(X_1; \dots; X_k)$ quantifies to which extent the observer variables $X_1, X_2, \dots, X_k$ depend on each other. We call this flavour of self-organization *O-self-organization* ("O" for observer-based).

The set of observer variables can often be specified in a natural way. For instance, systems that are composed by many, often identical, individual subsystems, have a canonical observer, defined via the partition of the system into subsystems. For instance, the observer variables could denote the states of agents that collectively make up a system. An increase in the multi-information of the system with respect to the agent states then indicates an increasing degree of coordination between the agents: this is consistent with our intuitive understanding of self-organization. Reaction-diffusion systems are also naturally described in this framework. Each point in space becomes an observer variable; in the attractor states with spot-and-wave patterns, these observer variables are intrinsically correlated.

Note, however, that for the multi-information not to vanish, it is still necessary that the whole system has some degree of freedom and that there is not just a single fixed pattern the system converges to. This makes sense, since otherwise one would just be talking about a single-attractor, and thus trivial, system.

Using the Self-Organizing Map as model system, and the individual neuron weights as observer variables, Polani (2003) discusses in detail the advantage of multi-information as a measure for self-organization as compared to other information-theoretic candidates for such a measure (except for the comparison with SC-self-organization, which is discussed in the present chapter for the first time). Many of the arguments carry over to other typical scenarios.

Note that compared to SC-self-organization (Sect. 2.5.2), O-self-organization is different in several respects. SC-self-organization does not require observers, and arises from the intrinsic dynamics of the system. This is orthogonal to the view of O-self-organization. In principle, the need for fewer assumptions by SC-self-organization is a conceptual advantage. On the other hand, to model the appearance of regular patterns (e.g. of a reaction-diffusion system) as a self-organization process one must anyway specify the extra spatial structure in which the patterns appear. In O-self-organization, this can be directly made a part of the specification. Thus, O-self-organization would be a natural candidate for these types of scenarios.

2.6.1 Observer Dependence

For the observer-based measure, a central question is how the measure changes as one moves from one observer to another, i.e. what happens to the measure on change of the "coordinate system". It turns out that it is possible to formulate an interesting relation between fine-grained observers and a coarse-graining of the very same observers. We will discuss this relation in the following.

Let $X_i, i = 1 \ldots k$ be a collection of jointly distributed random variables; this collection forms the *fine-grained observer*. Obtain the *coarse-grained observer* by grouping the X_i according to

$$\underbrace{X_1, \ldots, X_{k_1}}_{\tilde{X}_1}, \underbrace{X_{k_1+1}, \ldots, X_{k_2}}_{\tilde{X}_2}, \ldots, \underbrace{X_{k_{\tilde{k}-1}+1}, \ldots, X_k}_{\tilde{X}_{\tilde{k}}}, \tag{2.4}$$

i.e. each of the coarse-grained variables $\tilde{X}_j, j = 1 \ldots \tilde{k}$ is obtained by grouping several of the fine-grained variables X_i together.

Then the multi-information of the fine-grained observer can be expressed as[5]

$$I(X_1; X_2; \ldots; X_k) = I(\tilde{X}_1; \tilde{X}_2; \ldots; \tilde{X}_{\tilde{k}}) + \sum_{j=1}^{\tilde{k}} I(X_{k_{j-1}+1}; \ldots; X_{k_j}), \tag{2.5}$$

(where we adopt the convention $k_0 := 0$ and $k_{\tilde{k}} := k$). Relation (2.5) states that the multi-information as measure of self-organization in the fine-grained case can be expressed as the multi-information for the set of coarse-grained variables, corrected by the *intrinsic* multi-information of all these coarse-grained variables, or to put it snappily, "the fine-grained system is more than the sum of its coarse-grained version". The proof is sketched in Appendix.[6]

Equation (2.5) states how the intrinsic information of a system changes under a "change of coordinates" by regrouping the random variables that represent the system. This "bookkeeping" of multi-information while changing the basis for the system description in general only applies to regrouping of variables, but not to recoding. Under recoding of variables (i.e. re-representing the variables X_i by random variables $Y_i = f_i(X_1, \ldots, X_i, \ldots, X_k)$ where f_i is some deterministic function), there is no canonical way of transforming multi-information in a simple and transparent manner.

To see that, note that recoding may entirely remove dependencies between the recoded X_i (e.g. in Independent Component Analysis, Comon 1991). In fact, further requirements can be added to the component independence; indeed, this has been proposed as a way of discovering degrees of freedom representing *emergent levels of description* (Polani 2004, 2006). In this sense, O-self-organization is distinct from and effectively conjugate to the "emergent descriptions" concept. This dichotomy mirrors the analogous dichotomy exhibited by the formalization of self-organization and emergence using the ϵ-machine formalism (Sect. 2.5.2).

[5]This is a generalization of Eq. (3) from Tononi et al. (1994) for the bipartite case to the multipartite case.

[6]This property is related to a property that can be proven for graphical models, see e.g. Proposition 2.1 in Slonim et al. (2001).

2.7 Discussion

2.7.1 SC- and O-Self-Organization

We have emphasized that self-organization is a phenomenon often discussed in conjunction with complex systems. While there is a manifold selection of processes that are associated with this phenomenon, most notions used to characterize self-organization are either too vague to be useful, or too specific to be transferable from a system to another. The information-theoretic notions of (statistical complexity) SC-self-organization (Shalizi 2001; Shalizi et al. 2004) and that of (observer-based) O-self-organization (Polani 2003) strive to fill this gap. While similar to each other in the general philosophy, they are essentially "orthogonal" as quantities.

SC-self-organization takes in a single time series and measures the growth in statistical complexity during time. O-self-organization requires an observer, i.e. a set of variables through which the system state is observed. Such a set of variables is often naturally available, for instance, in multi-agent systems. Similar to SC-self-organization, O-self-organization seems to capture essential aspects of self-organization—for instance, the freezing of seemingly unrelated degrees of freedom (the observer variables) into highly coordinated global behaviour.

While SC-self-organization concentrates on the complexity of the temporal dynamics, O-self-organization concentrates on the compositional aspects of the system (this compositionality can, but need not be spatial). This distinction is also what indicates the use of each of the measures. If one focuses on the temporal dynamics, SC-self-organization may be more relevant, if on the spatial or compositional dynamics, O-self-organization may be the measure of choice. As the precise mathematical conceptualizations of self-organization are relatively recent, it will take some time until enough experience is accumulated to make an informed decision which measure is appropriate to use in a given constellation, or whether still other, more suitable measures will need to be discovered.

A word of caution at this point: the calculation of multi-information is difficult if the number of variables is large (Slonim et al. 2005). Particularly in unorganized and random states a Monte-Carlo estimate of the entropies and multi-information is likely to be undersampled and to overestimate the organization of the system in that state. Research is underway to develop general methods that are able to properly estimate the organization of unstructured states (and to distinguish it from the organized states).

2.7.2 Introducing Observers

For O-self-organization, we have assumed the existence of natural observers. What if none exist? Which ones to introduce and to use? The multi-information term consists of the entropies of the individual variables as well as the entropy of the joint variables. The latter depends only on the total system, not the observer. It is therefore the entropies of the *individual* variables that will change if we choose different observers. In general, it is quite possible to choose them in such a way as to make them independent—while this choice of observer is interesting (it essentially corresponds to Independent Component Analysis, Sect. 2.6.1), it makes the system maximally un-self-organized. This clearly shows that O-self-organization is not intrinsic. O-self-organization is "in the eye of the beholder" (Harvey 2000), but in a formally precise way.

Now, for O-self-organization to be present at all, the whole system must have some degrees of uncertainty, otherwise the individual variable entropies will also collapse and the multi-information will vanish. This is a property of the whole system. Thus, one could consider a natural observer one that *maximizes* the multi-information (as opposed to minimizing it), and thus making the system as self-organized as possible. If this is the case, O-self-organization could be viewed as the opposite to independent component decomposition.

But there is yet another way of constructing a natural observer: if one considers units (agents) that operate in the system and which possess sensors and actuators, the former which attain information about the system, and the latter which act upon and modify the system, then the perception-action loop of these agents forms a structured information channel. It can be shown (Klyubin et al. 2004) that maximizing the information flow through this channel allows the units to extract features from the system that are pertinent to the structure of the system.

This view is particularly interesting since it does not look at a system with a pre-ordained temporal dynamics, but rather the units (agents) have the option to choose their own actions. Nevertheless, once they perform the information flow maximization, they attain perceptual properties specially appropriate for the system at hand. The thus attained filters or feature detectors could act as another form of natural observer variables for the given system. Similarly, principles of informational organization can lead to a joint coordination of a sensorimotor device (Klyubin et al. 2005; Prokopenko et al. 2006) and direct systems to an equipment with embodiment-appropriate pattern detector loops.

2.8 Conclusion

The present chapter discussed the general problem of defining self-organization and presented two operational approaches, both based on information-theoretic principles. One approach, based on the ϵ-machine formalism, defines self-organization as an intrinsic property of a system, as a growth of the memory required to process a time series of random variable. The other approach defines self-organization via an observer, in typical cases realized as a family of variables of more-or-less similar type; a growing coordination between these variables with time is then identified as self-organization. Depending on one's aims, one will choose one or the other model to identify self-organization in a system. In particular, SC-self-organization will be the notion of choice if one is interested in characterizing the increase in complexity of the temporal dynamics, while O-self-organization emphasizes the self-organization process in a system composed of many individual subsystems.

The advantage of using information-theoretic notions for quantifying self-organization is that they provide a precise language for identifying the conditions of self-organization and the underlying assumptions, as opposed to vague or uncomputable qualitative characterizations. The quantitative character of information measures also allows one to actively search for "more self-organized" systems in a given context, rather than just state whether a system possesses or does not possess this property (as e.g. an algebraic characterization would do). In addition, the information-theoretic language forces one to specify exactly the assumptions and requirements underlying the notions one is using.

In short, information theory proves to be a powerful language to express self-organization and other central concepts relevant to complex systems. Even if one ultimately should prefer a different route to characterize self-organization in a complex system, it is probably a good first bet to strive towards a formulation that profits from the clarity and transparence of the information-theoretic language.

Appendix: Proof of Relation Between Fine and Coarse-Grained Multi-Information

Proof First, note that a different way to write the composite random variables $\tilde{X}_j$ is $\tilde{X}_j = (X_{k_{j-1}+1}, \ldots, X_{k_j})$ for $j = 1 \ldots \tilde{k}$, giving

$$H(\tilde{X}_j) = H(X_{k_{j-1}+1}, \ldots, X_{k_j}) \,. \tag{2.6}$$

Similarly, the joint random variable $(\tilde{X}_1, \ldots, \tilde{X}_{\tilde{k}})$ consisting of the composite random variables $\tilde{X}_j$ can be seen as a regrouping of the elementary random variables $X_1, \ldots, X_k$.Therefore the joint random variable constructed from the $\tilde{X}_j$ and that constructed from the X_i have both the same entropy:

$$H(\tilde{X}_1, \ldots, \tilde{X}_{\tilde{k}}) = H(X_1, \ldots, X_k) \,. \tag{2.7}$$

For consistency of notation, write $k_0 = 0$ and $k_{\tilde{k}} = k$. One then obtains

$$I(\tilde{X}_1; \tilde{X}_2; \ldots; \tilde{X}_{\tilde{k}}) + \sum_{j=1}^{\tilde{k}} I(X_{k_{j-1}+1}; \ldots; X_{k_j})$$

$$= \sum_{j=1}^{\tilde{k}} H(\tilde{X}_j) - H(\tilde{X}_1, \ldots, \tilde{X}_{\tilde{k}})$$

$$+\sum_{j=1}^{\tilde{k}}\left(\sum_{j'=k_{j-1}+1}^{k_j} H(X_{j'}) - H(X_{k_{j-1}+1},\dots,X_{k_j})\right)$$

$$=\underbrace{\sum_{j=1}^{\tilde{k}}\sum_{j'=k_{j-1}+1}^{k_j} H(X_{j'})}_{=\sum_{i=1}^{k} H(X_i)}+\sum_{j=1}^{\tilde{k}}\underbrace{\big(H(\tilde{X}_j) - H(X_{k_{j-1}+1},\dots,X_{k_j})\big)}_{=0}$$

$$-\underbrace{H(\tilde{X}_1,\dots,\tilde{X}_{\tilde{k}})}_{=H(X_1,\dots,X_k)}$$

where the first term results from a regrouping of summands, the second term results from Eq. (2.6) and the third from rewriting the whole set of random variables from the coarse-grained to the fine-grained notation, thus giving

$$=\sum_{i=1}^{k} H(X_i) - H(X_1,\dots,X_k)$$
$$=I(X_1;\dots;X_k)$$

which proves the equation. □

References

Adami, C. (1998). *Introduction to artificial life*. New York: Springer.

Ashby, W. R. (1947). Principles of the self-organizing dynamic system. *The Journal of General Psychology*, *37*, 125–128.

Ay, N., & Krakauer, D. C. (2007). Geometric robustness theory and biological networks. *Theory in Biosciences*, *125*(2), 93–121.

Ay, N., & Polani, D. (2008). Information flows in causal networks. *Advances in Complex Systems*, *11*(1), 17–41.

Ay, N., & Wennekers, T. (2003). Dynamical properties of strongly interacting Markov chains. *Neural Networks*, *16*(10), 1483–1497.

Baas, N. A., & Emmeche, C. (1997). On emergence and explanation. *Intellectica*, *2*(25), 67–83.

Bar-Yam, Y. (1997). *Dynamics of complex systems. Studies in nonlinearity*. Boulder: Westview Press.

Bennett, C. H., & Landauer, R. (1985). The fundamental limits of computation. *Scientific American*, *253*(1), 48–56.

Bertschinger, N., Olbrich, E., Ay, N., & Jost, J. (2006). Autonomy: an information theoretic perspective. In *Proc. workshop on artificial autonomy at Alife X*, Bloomington, Indiana (pp. 7–12).

Comon, P. (1991). Independent component analysis. In *Proc. intl. signal processing workshop on higher-order statistics*, Chamrousse, France (pp. 111–120).

Crutchfield, J. P. (1994). The calculi of emergence: computation, dynamics, and induction. *Physica D*, 11–54.

Crutchfield, J. P., & Young, K. (1989). Inferring statistical complexity. *Physical Review Letters*, *63*, 105–108.

Emmeche, C., Køppe, S., & Stjernfelt, F. (2000). Levels, emergence, and three versions of downward causation. In P. B. Andersen, C. Emmeche, N. O. Finnemann, & P. V. Christiansen (Eds.), *Downward causation. minds, bodies and matter* (pp. 13–34). Århus: Aarhus University Press.

Golubitsky, M., & Stewart, I. (2003). *The symmetry perspective*. Basel: Birkhäuser.

Grassberger, P. (1986). Toward a quantitative theory of self-generated complexity. *International Journal of Theoretical Physics*, *25*, 907–938.

Haken, H. (1983). *Advanced synergetics*. Berlin: Springer.

Harvey, I. (2000). *The 3 Es of artificial life: emergence, embodiment and evolution*. Invited talk at Artificial Life VII, 1–6 August, Portland, Oregon.

Helbing, D., Buzna, L., Johansson, A., & Werner, T. (2005). Self-organized pedestrian crowd dynamics: experiments, simulations, and design solutions. *Transportation Science*, *39*(1), 1–24.

Hoyle, R. (2006). *Pattern formation*. Cambridge: Cambridge University Press.

Jetschke, G. (1989). *Mathematik der Selbstorganisation*. Braunschweig: Vieweg.

Klyubin, A. S., Polani, D., & Nehaniv, C. L. (2004). Organization of the information flow in the perception-action loop of evolved agents. In *Proceedings of 2004 NASA/DoD conference on evolvable hardware* (pp. 177–180). Los Alamitos: IEEE Computer Society.

Klyubin, A. S., Polani, D., & Nehaniv, C. L. (2005). Empowerment: a universal agent-centric measure of control. In *Proc. IEEE congress on evolutionary computation (CEC 2005)*, Edinburgh, Scotland, 2–5 September 2005 (pp. 128–135). New York: IEEE.

Landauer, R. (1961). Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, *5*, 183–191.

Mees, A. I. (1981). *Dynamics of feedback systems*. New York: Wiley.

Meinhardt, H. (1972). A theory of biological pattern formation. *Kybernetik*, *12*, 30–39.

Meinhardt, H. (1982). *Models of biological pattern formation*. San Diego: Academic Press.

Pask, G. (1960). The natural history of networks. In M. C. Yovits & S. Cameron (Eds.), *Computer science and technology and their application. Self-organizing systems—proceedings of an interdisciplinary conference*, 5–6 May 1959 (pp. 5–6). New York: Pergamon.

Polani, D. (2003). Measuring self-organization via observers. In W. Banzhaf, T. Christaller, J. Ziegler, P. Dittrich, J. T. Kim, H. Lange, T. Martinetz, & F. Schweitzer (Eds.), *Advances in artificial life. Proc. 7th European conference on artificial life*, Dortmund, 14–17 September. Berlin: Springer.

Polani, D. (2004). Defining emergent descriptions by information preservation. *InterJournal Complex Systems*, *1102*.

Polani, D. (2006). Emergence, intrinsic structure of information, and agenthood. *InterJournal Complex Systems*, *1937*.

Prigogine, I., & Nicolis, G. (1977). *Self-organization in non-equilibrium systems: from dissipative structures to order through fluctuations*. New York: Wiley.

Prokopenko, M., Gerasimov, V., & Tanev, I. (2006). Evolving spatiotemporal coordination in a modular robotic system. In S. Nolfi, G. Baldassarre, R. Calabretta, J. C. T. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, & D. Parisi (Eds.), *Lecture notes in computer science: Vol. 4095. From animals to animats 9: 9th international conference on the simulation of adaptive behavior (SAB 2006)*, Rome, Italy (pp. 558–569). Berlin: Springer.

Rasmussen, S., Baas, N., Mayer, B., Nilsson, M., & Olesen, M. W. (2001). Ansatz for dynamical hierarchies. *Artificial Life*, *7*, 329–353.

Reichl, L. (1980). *A modern course in statistical physics*. Austin: University of Texas Press.

Shalizi, C. R. (2001). *Causal architecture, complexity and self-organization in time series and cellular automata*. PhD thesis, University of Wisconsin-Madison.

Shalizi, C. R., & Crutchfield, J. P. (2002). Information bottlenecks, causal states, and statistical relevance bases: how to represent relevant information in memoryless transduction. *Advances in Complex Systems*, *5*(1), 91–95.

Shalizi, C. R., Shalizi, K. L., & Haslinger, R. (2004). Quantifying self-organization with optimal predictors. *Physical Review Letters*, *93*(11), 118701.

Slonim, N., Friedman, N., & Tishby, T. (2001). Agglomerative multivariate information bottleneck. In *Neural information processing systems (NIPS 01)*, La Jolla (pp. 929–936).

Slonim, N., Atwal, G. S., Tkačik, G., & Bialek, W. (2005). Estimating mutual information and multi-information in large networks. arXiv:cs.IT/0502017.

Spitzner, A., & Polani, D. (1998). Order parameters for self-organizing maps. In L. Niklasson, M. Bodén, & T. Ziemke (Eds.), *Proc. of the 8th int. conf. on artificial neural networks (ICANN 98)*, Skövde, Sweden (Vol. 2, pp. 517–522). Berlin: Springer.

Tishby, N., Pereira, F. C., & Bialek, W. (1999). The information bottleneck method. In *Proc. 37th annual Allerton conference on communication, control and computing*, Urbana-Champaign, IL.

Tononi, G., Sporns, O., & Edelman, G. M. (1994). A measure for brain complexity: relating functional segregation and integration in the nervous system. *Proceedings of the National Academy of Sciences of the United States of America*, *91*, 5033–5037.

Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, *327*, 37–72.

Walter, W. G. (1951). A machine that learns. *Scientific American*, *185*(2), 60–63.

Yovits, M. C. & Cameron, S. (Eds.) (1960). *Computer science and technology and their application. Self-organizing systems—proceedings of an interdisciplinary conference*, 5–6 May 1959. New York: Pergamon.

Part II
Distributed Management and Control

Chapter 3
Self-Organizing Traffic Lights: A Realistic Simulation

Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe

3.1 Introduction: Catch the Green Wave? Better Make Your Own!

Everybody in populated areas suffers from traffic congestion problems. To deal with them, different methods have been developed to mediate between road users as best as possible. Traffic lights are not the only pieces in this puzzle, but they are an important one. As such, different approaches have been used trying to reduce waiting times of users and to prevent traffic jams. The most common consists of finding the appropriate phases and periods of traffic lights to quantitatively optimize traffic flow. This results in "green waves" that flow through the main avenues of a city, ideally enabling cars to drive through them without facing a red light, as the speed of the green wave matches the desired cruise speed for the avenue. However, this approach does not consider the current state of the traffic. If there is a high traffic density, cars entering a green wave will be stopped by cars ahead of them or cars that turned into the avenue, and once a car misses the green wave, it will have to wait the whole duration of the red light to enter the next green wave. On the other hand, for very low densities, cars might arrive too quickly at the next intersection, having to stop at each crossing. This method is certainly better than having no synchronization at all, however, it can be greatly improved.

Traffic modelling has enhanced greatly our understanding of this complex phenomenon, especially during the last decade (Prigogine and Herman 1971; Wolf et al. 1996; Schreckenberg and Wolf 1998; Helbing et al. 2000; Helbing 1997; Helbing

B. D'Hooghe is deceased.

S.-B. Cools · C. Gershenson (✉) · B. D'Hooghe
Centrum Leo Apostel, Vrije Universiteit Brussel, Krijgskundestraat 33, 1160 Brussels, Belgium
e-mail: cgg@unam.mx

S.-B. Cools
e-mail: secools@vub.ac.be

and Huberman 1998), suggesting different improvements to the traffic infrastructure. One of these consists of adapting the traffic lights to the current traffic conditions. Indeed, modern "intelligent" advanced traffic management systems (ATMS) use learning methods to adapt phases of traffic lights, normally using a central computer (Federal Highway Administration 1998; Hunt et al. 1981). The self-organizing approach we present here does not need a central computer, as the global synchronization is adaptively achieved by local interactions between cars and traffic lights, generating flexible green waves on demand.

We have previously shown in an abstract simulation (Gershenson 2005) that self-organizing traffic lights can greatly improve traffic flow for any density. In this chapter, we extend these results to a realistic setting, implementing self-organizing traffic lights in an advanced traffic simulator using real data from a Brussels avenue. In the next section, a brief introduction to the concept of self-organization is given. The SOTL control method is then presented, followed by the moreVTS simulator. In Sect. 3.5, results from our simulations are shown, followed by Discussion, Future Work, and Conclusions.

3.2 Self-Organization

The term *self-organization* has been used in different areas with different meanings, as is cybernetics (von Foerster 1960; Ashby 1962), thermodynamics (Nicolis and Prigogine 1977), biology (Camazine et al. 2003), mathematics (Lendaris 1964), computing (Heylighen and Gershenson 2003), information theory (Shalizi 2001), synergetics (Haken 1981), and others (Skår and Coveney 2003) (for a general overview, see Heylighen 2003). However, the use of the term is subtle, since any dynamical system can be said to be self-organizing or not, depending partly on the observer (Gershenson and Heylighen 2003; Ashby 1962): If we decide to call a "preferred" state or set of states (i.e. attractor) of a system "organized", then the dynamics will lead to a self-organization of the system.

It is not necessary to enter into a philosophical debate on the theoretical aspects of self-organization to work with it, so a practical notion will suffice (Gershenson 2006):

> A system *described* as self-organizing is one in which elements *interact* in order to achieve *dynamically* a global function or behavior.

This function or behavior is not imposed by one single or a few elements, nor determined hierarchically. It is achieved *autonomously* as the elements interact with one another. These interactions produce feedbacks that regulate the system. If we want the system to solve a problem, it is useful to describe a complex system as self-organizing when the "solution" is not known beforehand and/or is changing constantly. Then, the solution is dynamically sought by the elements of the system. In this way, systems can adapt quickly to unforeseen changes as elements interact locally. In theory, a centralized approach could also solve the problem, but in practice such an approach would require too much time to compute the solution and would not be able to keep the pace with the changes in the system and its environment.

In engineering, a self-organizing system would be one in which elements are designed to *dynamically* and *autonomously* solve a problem or perform a function at the system level. Our traffic lights are self-organizing because each one makes a decision based only on local information concerning its own state. Still, they manage to achieve robust and adaptive global coordination.

3.3 Self-Organizing Traffic Lights: The Control Method

In the SOTL method (originally named *sotl-platoon* in Gershenson (2005)), each traffic light, i.e. intersection, keeps a counter κ_i which is set to zero when the light turns red and then incremented at each timestep by the number of cars approaching *only* the red light (i.e. the next one a car will reach) independently of the status or speed of the cars (i.e. moving or stopped). When κ_i (representing the integral of cars over time) reaches a threshold θ, the green light at the same intersection turns yellow, and the following time step it turns red with $\kappa_i = 0$, while the red light which counted turns green. In this way, if there are more cars approaching or waiting behind a red light, it will turn to green faster than if there are only few cars. This simple mechanism achieves self-organization in the following way: if there are single or few cars, these will be stopped for more time behind red lights. This gives time for other cars to join them. As more cars join the group, cars will wait less time behind red lights. With a sufficient number of cars, the red lights will turn green even before they reach the intersection, generating “green corridors”. Having “platoons” or “convoys” of cars moving together improves traffic flow, compared to a homogeneous distribution of cars, since there are large empty areas between platoons, which can be used by crossing platoons with few interferences.

The following constraint is considered to prevent traffic lights from switching too fast when there are high densities: A traffic light will not be changed if the number of time steps is less than a minimum phase, i.e. $\varphi_i < \varphi_{\min}$ (φ_i is the time since the light turned green).

Two further conditions are taken into account to regulate the size of platoons. Before changing a red light to green, the controller checks if a platoon is crossing through, in order not to break it. More precisely, a red light is not changed to green if on the crossing street there is at least one car approaching within a distance ω from the intersection. This keeps crossing platoons together. For high densities, this condition alone would cause havoc, since large platoons would block the traffic flow of intersecting streets. To avoid this, we introduce a second condition. Condition one is not taken into account if there are more than μ cars approaching the green light. Like this, long platoons can be broken, and the restriction only comes into place if a platoon will soon be through an intersection.

The SOTL method is formally summarized in Algorithm 3.1.

Algorithm 3.1: Self-organizing traffic lights (SOTL) controller

```
1: foreach (timestep) do
2:     $\kappa_i += cars_{approachingRed}$ in $\rho$
3:     if ($\varphi_i \geq \varphi_{\min}$) then
4:         if not ($0 < cars_{approachingGreen}\ in\ \omega < \mu$) then
5:             if ($\kappa_i \geq \theta$) then
6:                 $switchlight_i()$
7:                 $\kappa_i = 0$
8:             end
9:         end
10:    end
11: end
```

This method has no phase or internal clock. If there are no cars approaching a red light, the complementary one can stay green. We say that this method is self-organizing because the global performance is given by the local rules followed by each traffic light: they are "unaware" of the state of other intersections and still manage to achieve global coordination.

The method uses a similar idea to the one used by Porche and Lafortune (1999), but with a much simpler implementation. There is no costly prediction of arrivals at intersections, and no need to establish communication between traffic lights to achieve coordination and there are not fixed cycles.

3.4 A Realistic Traffic Simulator: moreVTS

Our simulator (moreVTS 2006) (A More Realistic Vehicle Traffic Simulator) is the third of a series of open source projects building on the previous one, developed in Java. Green Light District (GLD) was developed by the Intelligent Systems Group at the University of Utrecht (GLD 2001; Wiering et al. 2004). Then, GLD was improved by students in Argentina within the iAtracos project, which we used as a starting point for our simulator, which introduces realistic physics into the simulation. Among other things, acceleration was introduced, and the scale was modified so that one pixel represents one meter and one cycle represents one second.

The simulator allows the modelling of complex traffic configurations, enabling the user to create maps and then run simulations varying the densities and types of road users. Multiple-lane streets and intersections can be arranged, as well as spawn and destination frequencies of cars. For implementation details of moreVTS, the reader is referred to Cools (2006).

The self-organizing traffic light controller described in the previous section was implemented in moreVTS. Using data provided by the Brussels Capital Region, we were able to build a detailed simulation of the Rue de la Loi/Wetstraat, a four-lane

Table 3.1 Average vehicle count per hour at the beginning of the Wetstraat. Data kindly provided by the Brussels Capital Region

0	1	2	3	4	5	6	7	8	9	10	11
476	255	145	120	175	598	2933	5270	4141	4028	3543	3353

12	13	14	15	16	17	18	19	20	21	22	23
3118	3829	3828	3334	3318	3519	3581	3734	2387	1690	1419	1083

westwards one-way avenue in Brussels which gathers heavy traffic towards the centre of the city. We used the measured average traffic densities per hour on working days for 2004 (shown in Table 3.1) and the current "green wave" method, which has a period of 90 seconds, with 65 seconds for the green phase on the Wetstraat, 19 for the green phase on side streets, and 6 for transitions. This enabled us to compare our self-organizing controller with a standard one in a realistic setting. Figure 3.1 shows the simulation view of the Wetstraat and its surrounding streets.

The data from Table 3.1 is for the cars entering the Wetstraat on the East, so the spawn rates for the two nodes in the simulation representing this were set according to this data. For the other nodes, the spawn and destination frequencies were set based on a field study we performed in May 2006, comparing the percentage of cars that flow through the Wetstraat and those that flow through side streets, enter, or leave the Wetstraat. These percentages were kept constant, so that when the density of cars entering the Wetstraat changed, all the other spawn rates changed in the same proportion. On average, for each five cars flowing through a side street, one hundred flow through the Wetstraat. This is not the case of the Kuststraat, a two way avenue at the West of the Wetstraat (second and third crossing streets from left to right on Fig. 3.1), where for 100 cars driving through the Wetstraat, about 40 turn right, 40 turn left, and only 20 go straight, while 20 more drive through the Kuststraat (about 10 in each direction). The precise spawn rates and destination frequencies are given in Cools (2006, pp. 55–57).

3.5 Results

To measure the performance of the current green wave method and our self-organizing controller, we used the average trip waiting times (ATWT). The trip waiting time for one car is the travel time minus the minimum possible travel time (i.e. travel distance divided by the maximum allowed speed, which for the Wetstraat simulation is about sixty seconds).

Several simulation runs were performed to find the best parameters for the SOTL method. For each parameter and traffic density, five simulation runs representing one hour, i.e. 3600 cycles, were averaged. The results were robust and consistent, with

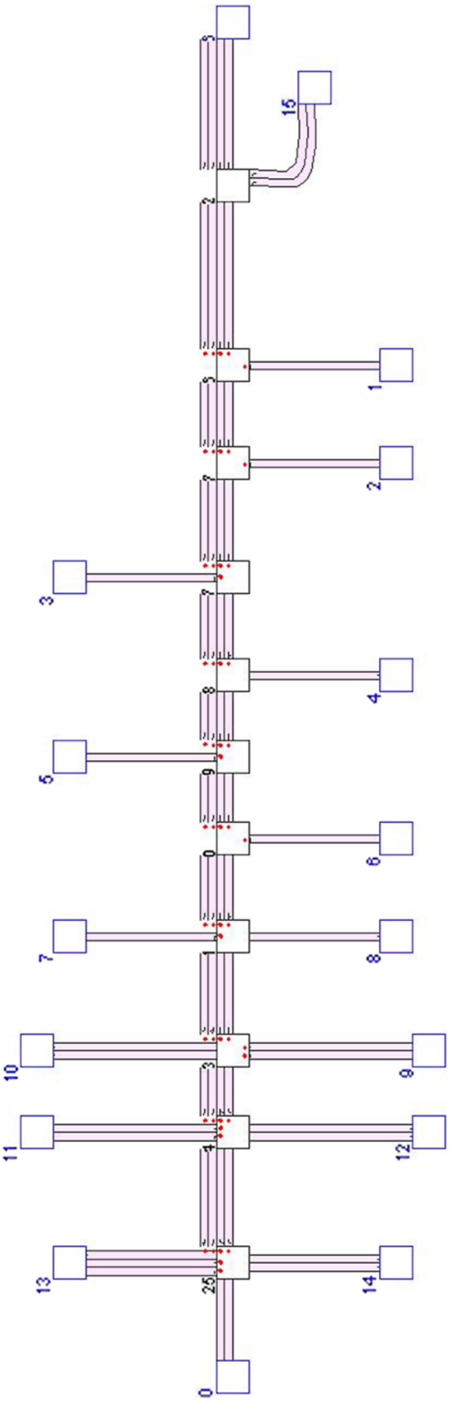


Fig. 3.1 Simulation of the Wetstraat and intersecting streets. Cars flow westward on the Wetstraat. *Red dots* represent traffic lights for each incoming lane at intersections (Color figure online)

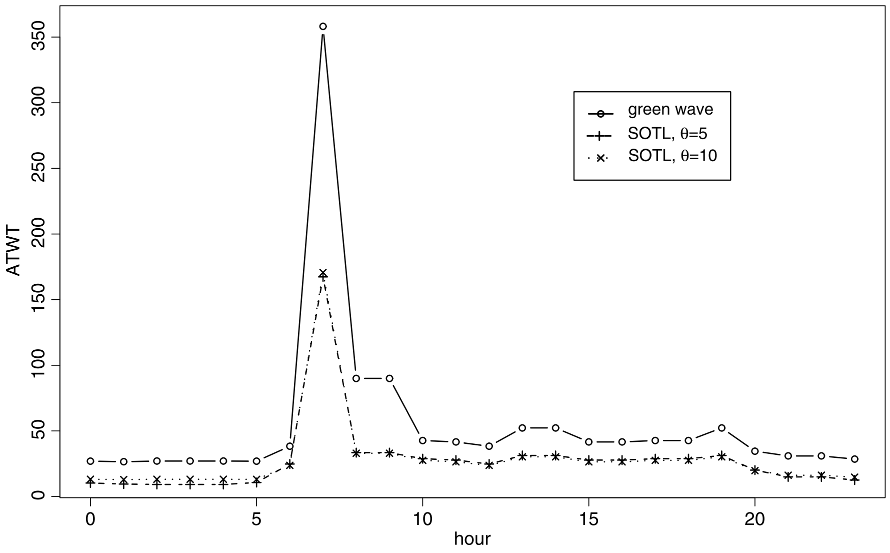


Fig. 3.2 Average trip waiting times (ATWT) at different hours of the day, green wave controller and SOTL controller with $\varphi_{\min} = 5$ and $\theta = 5; 10$

SOTL performing better than the green wave method for a wide range of parameters θ and $\varphi_{\min}$ (Cools 2006). Only the best ones are shown in Fig. 3.2, together with the results for the green wave method. The cruise speed used was 14 m/s, $\omega = 25$ and $\mu = 3$. Since some densities from Table 3.1 are very similar, we averaged and considered the same densities for 2:00, 3:00 and 4:00; 8:00 and 9:00; 10:00, 17:00 and 18:00; 11:00, 15:00 and 16:00; 13:00, 14:00 and 19:00; and 21:00 and 22:00.

As Fig. 3.2 shows, there is considerable reduction in ATWT using SOTL instead of the current green wave method. The ATWT for the densities at different hours using SOTL were from 34 % to 64 % of the ATWT for the green wave method, and on average 50 %. Since the minimum travel time for the Wetstraat is about one minute, while the overall ATWT for the green wave method is also about one minute and for SOTL about half, the improvement in the average total travel times would be of about 25 %, i.e. cars under a green wave method would take 33 % more time to reach their destination than those under SOTL. This shows with a realistic simulation that SOTL improves greatly traffic flow compared to the current green wave method.

3.6 Discussion

The green wave method works well for regulating traffic on the Wetstraat, since most of the traffic flows through it. Still, having no consideration about the actual state of the traffic has several drawbacks. It can give a green light to a side street even if there are no cars on it, or when a group of cars is about to cross. Also, if the traffic density is high, the speed of the cars will be slower than that of the green wave. And when a car misses a green wave, it will have to wait a full cycle to enter the next one.

Having actual information about the traffic state enables SOTL to adapt to the current situation: it only gives green lights on demand, so time will not be wasted for streets without cars, and the street with more cars will have more demand, thus more green lights. Cars do have to wait behind red lights, but since while doing so they are demanding to cross, it is very unlikely that a car will have to wait more than $\phi_{\min}$. Moreover, when a car is stopped, a platoon is likely to be formed, accelerating the switching of green lights.

Another advantage of platoons is that they reduce entropy in the city, defined via the probability of finding a car in any part of the city. If there is maximal entropy, there is the same probability of finding a car anywhere on the city. This increases the probability of interference, i.e. that two cars will meet at an intersection, thus one needs to stop. The opposite extreme is less desirable: if we have a certainty of the position of every car, it is because they are stopped, i.e. in a traffic jam. However, platoons offer a useful balance: there is a high probability that a car will be close to another car, i.e. in a group. Thus, there are many free spaces left between platoons, that other platoons can exploit to cross without interferences. There will be interferences, but these will be minimal.

3.7 Future Work

The following list summarizes the future work.

- A method similar to SOTL has been used successfully in the United Kingdom for some time, but only for isolated intersections (Vincent and Young 1986). Indeed, it is not obvious to expect that traffic lights without direct communication would be able to coordinate robustly. In any case, the technology to implement it is already available, so a pilot study could be quickly deployed in a real city. Since the traffic lights are adaptive, only a few intersections could be changed, which would adapt to the control method used in the rest of the city. This also would make it easy to incrementally introduce them in large cities.
- We have observed that there is a monotonic relation between the best θ and the traffic density (Cools 2006). Exploring this relation better could allow us to set a variable θ depending on the current traffic density measured by the traffic lights. However, since SOTL performs very well for a broad range of parameters, it does not require the calculation of precise parameters. In other words, SOTL is not sensitive to small changes in parameters, making it a robust method.
- The SOTL method could also be used to give preference to certain users, e.g. public transport or emergency vehicles. Simply, a weight would be given to each vehicle in the count κ_i, so that vehicles with preference would be able to trigger green lights by themselves. They would be equivalent to a platoon of cars, thus

being seamlessly integrated into the system. This might be a considerable improvement compared to current methods where some vehicles (e.g. buses in London, trams in Brussels) have preference and the rest of the users are neglected, in some cases even when there are no preferred vehicles nearby.

- The "optimal" sizes of platoons, depending on different features of a city, is an interesting topic to research. The parameters of SOTL can be regulated to promote platoons of a certain size, so knowing which size should be aimed at would facilitate the parameter search.
- It would be interesting to compare SOTL with the Dresden method (Helbing et al. 2005; Lämmer et al. 2006), which couples oscillators using self-organization, whereas SOTL has no internal phases nor clocks.

3.8 Conclusions

In this chapter we presented results showing that a self-organizing traffic lights control method considerably improves the traffic flow compared to the current green wave method, namely reducing on average waiting times by half. These results are encouraging enough to continue refining and exploring similar traffic light controllers and to implement them in real cities, starting with pilot studies. However, we would not like to motivate further the use of cars with an efficient traffic control, since this would increase traffic densities and pollution even more. Any city aiming at improving its traffic flow should promote in parallel alternative modes of transportation, such as cycling, walking, car pooling, or using public transport.

3.9 Epilogue

After this chapter was initially published (Gershenson 2008), the self-organizing method was improved with three more rules (Gershenson and Rosenblueth 2012a), now achieving close to optimal performance for all densities. This was evaluated with an elementary cellular automaton model of city traffic (Rosenblueth and Gershenson 2011). This abstract model is useful for comparing methods against a theoretical optimum and for detecting phase transitions. Since the self-organizing method is close to the theoretical optimum, there is little which can be further improved in traffic light coordination. We have also generalized the algorithm to complex intersections (Gershenson and Rosenblueth 2012b), implementing the elementary cellular automaton model on an hexagonal grid. The green wave method performs even worse when more streets have to be coordinated, while the self-organizing method manages to scale to the increased complexity of the traffic light coordination. In one case, as vehicle density varied, six phase transitions were identified. In another case, ten phase transitions occurred. Currently we are planning a pilot study to test the effectiveness of the self-organizing method with real traffic.

Self-organization has also been applied to other aspects of urban mobility (Gershenson 2012). One interesting example involves public transportation systems (Gershenson 2011), where self-organization manages to achieve better than optimal performance. Self-organizing systems are promising not only for improving mobility, but for a broad variety of urban problems (Gershenson 2013). To contribute to this goal, a methodology for designing and controlling self-organizing systems has been developed (Gershenson 2007).

Acknowledgements We should like to thank the Ministerie van het Brussels Hoofdstedelijk Gewest for their support, providing the data for the Wetstraat.

References

Ashby, W. R. (1962). Principles of the self-organizing system. In H. V. Foerster & G. W. Zopf Jr. (Eds.), *Principles of self-organization* (pp. 255–278). New York: Pergamon.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., & Bonabeau, E. (2003). *Self-organization in biological systems*. Princeton: Princeton University Press.

Cools, S. B. (2006). *A realistic simulation for self-organizing traffic lights*. Unpublished BSc Thesis, Vrije Universiteit Brussel.

Federal Highway Administration (1998). *Traffic control systems handbook*. Washington: U.S. Department of Transportation.

Gershenson, C. (2005). Self-organizing traffic lights. *Complex Systems*, *16*(1), 29–53.

Gershenson, C. (2006). *A general methodology for designing self-organizing systems* (Technical Report 2005-05). ECCO.

Gershenson, C. (2007). *Design and control of self-organizing systems*. Mexico: CopIt Arxives. http://tinyurl.com/DCSOS2007.

Gershenson, C. (2008). Self-organizing traffic lights: a realistic simulation. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (1st ed.). London: Springer.

Gershenson, C. (2011). Self-organization leads to supraoptimal performance in public transportation systems. *PLoS ONE*, *6*(6), e21469.

Gershenson, C. (2012). Self-organizing urban transportation systems. In J. Portugali, H. Meyer, E. Stolk, & E. Tan (Eds.), *Complexity theories of cities have come of age: an overview with implications to urban planning and design* (pp. 269–279). Berlin: Springer.

Gershenson, C. (2013, in press). Living in living cities. *Artificial Life*.

Gershenson, C., & Heylighen, F. (2003). When can we call a system self-organizing? In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, & J. Ziegler (Eds.), *LNAI: Vol. 2801. 7th European conference on advances in artificial life, ECAL 2003* (pp. 606–614). Berlin: Springer.

Gershenson, C., & Rosenblueth, D. A. (2012a). Adaptive self-organization vs. static optimization: a qualitative comparison in traffic light coordination. *Kybernetes*, *41*(3), 386–403.

Gershenson, C., & Rosenblueth, D. A. (2012b). Self-organizing traffic lights at multiple-street intersections. *Complexity*, *17*(4), 23–39.

GLD (2001). Green Light District.

Haken, H. (1981). Synergetics and the problem of selforganization. In G. Roth & H. Schwegler (Eds.), *Self-organizing systems: an interdisciplinary approach* (pp. 9–13). New York: Campus.

Helbing, D. (1997). *Verkehrsdynamik*. Berlin: Springer.

Helbing, D., & Huberman, B. A. (1998). Coherent moving states in highway traffic. *Nature*, *396*, 738–740.

Helbing, D., Herrmann, H. J., Schreckenberg, M., & Wolf, D. E. (Eds.) (2000). *Traffic and granular flow '99: social, traffic, and granular dynamics*. Berlin: Springer.

Helbing, D., Lämmer, S., & Lebacque, J.-P. (2005). Self-organized control of irregular or perturbed network traffic. In C. Deissenberg & R. F. Hartl (Eds.), *Optimal control and dynamic games* (pp. 239–274). Dordrecht: Springer.

Heylighen, F. (2003). The science of self-organization and adaptivity. In L. D. Kiel (Ed.), *The encyclopedia of life support systems*. Oxford: EOLSS.

Heylighen, F., & Gershenson, C. (2003). The meaning of self-organization in computing. *IEEE Intelligent Systems*, 72–75.

Hunt, P. B., Robertson, D. I., Bretherton, R. D., & Winton, R. I. (1981). *SCOOT—a traffic responsive method of coordinating signals* (Technical report, TRRL).

Lämmer, S., Kori, H., Peters, K., & Helbing, D. (2006). Decentralised control of material or traffic flows in networks using phase-synchronisation. *Physica. A*, *363*(1), 39–47.

Lendaris, G. G. (1964). On the definition of self-organizing systems. *Proceedings of the IEEE*, *52*(3), 324–325.

moreVTS (2006). A more realistic vehicle traffic simulator.

Nicolis, G., & Prigogine, I. (1977). *Self-organization in non-equilibrium systems: from dissipative structures to order through fluctuations*. Chichester: Wiley.

Porche, I., & Lafortune, S. (1999). Adaptive look-ahead optimization of traffic signals. *Journal of Intelligent Transportation Systems*, *4*(3), 209–254.

Prigogine, I., & Herman, R. (1971). *Kinetic theory of vehicular traffic*. New York: Elsevier.

Rosenblueth, D. A., & Gershenson, C. (2011). A model of city traffic based on elementary cellular automata. *Complex Systems*, *19*(4), 305–322.

Schreckenberg, M. & Wolf, D. E. (Eds.) (1998). *Traffic and granular flow '97*, Singapore. Berlin: Springer.

Shalizi, C. R. (2001). *Causal architecture, complexity and self-organization in time series and cellular automata*. PhD thesis, University of Wisconsin at Madison.

Skår, J. & Coveney, P. V. (Eds.) (2003). Self-organization: the quest for the origin and evolution of structure. *Philosophical Transactions of Royal Society of London A*, *361*(1807). Proceedings of the 2002 Nobel Symposium on self-organization.

Vincent, R. A., & Young, C. P. (1986). Self optimising traffic signal control using microprocessors—the TRRL MOVA strategy for isolated intersections. *Traffic Engineering & Control*, *27*(7–8), 385–387.

von Foerster, H. (1960). On self-organizing systems and their environments. In M. C. Yovitts & S. Cameron (Eds.), *Self-organizing systems* (pp. 31–50). New York: Pergamon.

Wiering, M., Vreeken, J., Veenen, J. V., & Koopman, A. (2004). Simulation and optimization of traffic in a city. In *IEEE intelligent vehicles symposium (IV'04)* (pp. 453–458). New York: IEEE.

Wolf, D. E., Schreckenberg, M., & Bachem, A. (Eds.) (1996). *Traffic and granular flow '95*. Singapore: World Scientific.

Chapter 4
Self-Organizing Sensing of Structures: Monitoring a Space Vehicle Thermal Protection System

Nigel Hoschke, Don C. Price, and D. Andrew Scott

4.1 Introduction

This Chapter describes the development and operation of an experimental structural health monitoring system whose functionality is based on self-organization in a complex multi-agent system. Self-organization within a system of many interacting components is generally understood to mean the formation of global patterns, or the production of coordinated global behaviours, solely from the interactions among the lower-level components of the system. The important characteristics are that the resulting patterns or behaviours occur at a larger scale than the individual system components, and that the interactions between the components are not influenced by a central controller or by reference to the emergent pattern or behaviour: they are purely local interactions. Self-organization in biological systems has been defined and discussed by Camazine et al. (2001), and Prokopenko et al. (2008) have discussed self-organization from an information-theoretic perspective.

The system that will be described in this Chapter consists of a large number ($\sim$200) of semi-autonomous local sensing agents, each of which can sense, process data, and communicate with its neighbours. In this context self-organization means that the agents will produce a system-level response to external events or damage that is produced entirely by the local communications between the agents, and is not influenced by a central controller or by any system-level design. The main benefits

D.C. Price is deceased.

N. Hoschke (✉) · D.C. Price
Materials Science and Engineering, Commonwealth Scientific and Industrial Research Organisation (CSIRO), PO Box 218, Lindfield, NSW 2070, Australia
e-mail: nigel.hoschke@csiro.au

D.A. Scott
National Measurement Institute, PO Box 264, Lindfield, NSW 2070, Australia
e-mail: andy.scott@csiro.au

of this approach lie in scalability (the system performance is not limited by the computational and communication capability of a central controller) and in robustness (there is no single point of vulnerability, such as would be represented by a central controller).

4.1.1 The Requirements of Structural Health Monitoring

Structural Health Monitoring (SHM) is a new approach to monitoring the integrity and functionality of structures. It is expected to enhance, and ultimately replace, the traditional approach of periodic inspection for the maintenance of, for example, aerospace and other transport vehicles, bridges, buildings, and other safety-critical infrastructure. SHM uses information from sensors permanently embedded in the structure to detect events or conditions that may lead to damage, and/or to detect damage at an early stage and monitor its development with time. Initially, SHM systems will be used to plan maintenance as required, rather than the current practice of maintenance at pre-determined intervals, but ultimately it is likely to be used to manage and monitor materials and structures with self-repair capabilities.

SHM systems will be required to monitor very large numbers of sensors, to use the information deduced from the sensor data, to diagnose damaging situations and consequent damage, to form a prognosis of the future safety and functionality of the structure, and to initiate and monitor mitigation and repair procedures as required. Different forms of damage can develop on very different spatial and temporal scales (compare, for example, the effects of a sudden major impact with those of slowly-developing corrosion or fatigue), and the SHM system must be able to respond effectively in all cases.

Of paramount importance for SHM of safety-critical structures are the requirements for system robustness and scalability. The system must be capable of continuing to operate effectively in the presence of damage to itself and/or to the structure, and it must be able to operate efficiently, both locally and globally, even though it may be monitoring very large numbers of sensors. These requirements mitigate against the use of traditional centrally-controlled engineered systems, with their inherent points of vulnerability and communications limitations, in favour of distributed adaptive systems.

4.1.2 The Approach to SHM System Development

CSIRO, with support from NASA, has been developing an experimental structural health monitoring concept demonstrator (CD) test-bed system for the detection of high-velocity impacts, such as may occur due to the impact of a micrometeoroid on a space vehicle. The distinguishing feature of this system is that its architecture is based on a complex multi-agent system and its behaviours and responses are developed through self-organization. It has no central controller. This approach endows the system with a high degree of robustness, adaptability and scalability.

The test-bed has been built as a tool for research into sensor design, sensing strategies, communication protocols, and distributed processing using multi-agent systems. Each of the semi-autonomous sensing agents contains a suite of sensors to enable it to gather data related to the state of the structure (generally, but not necessarily, referring to the agent's local region), a facility to perform some processing of this data, and the ability to communicate with neighbouring agents. These agents may also have the capability to perform active tasks (e.g. repair functions), or there may be other agents to perform these functions. Agents may be embedded in the structure, eventually being integrated into the materials, or may be mobile and free to roam regions of the structure.

A number of recent articles have described the development of the hardware and software of the basic CD system of embedded piezoelectric sensors and processing electronics distributed throughout the structure, along with some multi-agent algorithms to characterize impacts and subsequent damage, see e.g. Prokopenko et al. (2006), Hoschke et al. (2006), Price et al. (2004), and Scott et al. (2005). The CD system has been designed to be highly flexible: by replacing the sensors and their associated interface and data acquisition electronics, the system can be readily reconfigured for other applications.

This Chapter provides more detail on the system described in our earlier works, with an emphasis on the communications between agents, the development of the tracking field algorithm for this system, and the incorporation of a robotic mobile agent which can move over the surface of the CD skin as an independent agent of the self-organizing system. This mobile agent can carry out sensing functions that the embedded agents cannot, and it is able to communicate with embedded agents of the CD structure in its vicinity. It is envisaged to be the forerunner of a swarm of small robots that will cooperatively perform both inspection and repair functions. The essential point is that the functions and behaviours of the mobile agent (or agents) are determined by self-organization of the entire system of fixed and mobile sensing agents.

In addition, an application of the principles of the CD to a practical damage scenario, the detection and evaluation of impact damage to the thermal protection systems (TPS) of spacecraft that re-enter the atmosphere, is described.

4.1.3 Overview of the Experimental System Operation

The CD system consists of a basic structure to be monitored, which is a rigid framework in the form of a hexagonal prism ($\sim$1 m in height and $\sim$1 m across the hexagonal section) covered by an aluminium skin 1 mm thick. Bonded to the inner surface of the aluminium are 768 small (2.5 mm diameter) piezoelectric sensors in groups of four, in the form of a regular array. A block of electronics that constitutes an agent

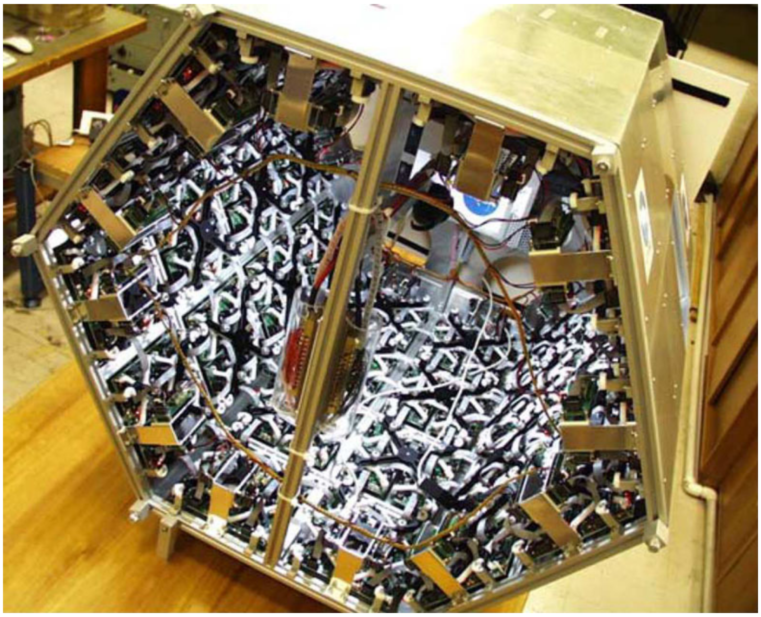


Fig. 4.1 The hexagonal prism physical implementation of the CD test-bed structure, lying on its side with the end open to reveal the cellular internal structure of the electronics

of the system is connected to each group of four sensors. There are 192 such agents, distributed in a square array with 32 on each of the hexagonal faces of the structure. Each agent monitors its group of four sensors, acquires and analyses the data they produce, and communicates with its four neighbours. Figure 4.1 is a photograph of this structure, showing the array of agents covering the inner surface. An agent, and the region of the skin that it monitors, is referred to as a cell, though the terms agent and cell will be referred to interchangeably throughout this Chapter.

An impact on the surface of the aluminium skin excites elastic waves which are detected by the piezoelectric sensors. High-velocity impacts are simulated using short laser pulses and/or steel spheres fired using a light gas-gun. Repeatable low-velocity impacts are produced using a pendulum. The agents that detect the impact also locate its position and estimate the damage severity from the sensor signals.

A number of multi-agent algorithms have been developed (Prokopenko et al. 2005b, 2006; Hoschke et al. 2006; Price et al. 2004; Scott et al. 2005) to identify impacts, the extent of damage caused, and networks of multiple impacts. The state of the system at any time is monitored on an external computer (the system visualizer) which acts as another agent of the system that requests and displays state information from the embedded agents. The visualizer does not have any control function: it simply monitors and displays the states of the other agents.

A robot has been developed to move around the outer surface of the skin to provide a mobile sensing capability. At this stage it carries a video camera to record visual images of damage, but it could carry other sensors. This robot can communicate through the aluminium skin, using ultrasonic signals, with the agent in its immediate vicinity. It is guided towards damage sites by information it obtains from the embedded agents as it moves: the necessary information to guide the robot is produced collectively by the multi-agent system—it is a self-organized response to the detection of an impact by a local agent. The robotic agent is not controlled centrally: it navigates purely by the local information it obtains from the agents in the underlying structure as it passes by, and therefore the robot movement is a self-organized response of the system of agents to the impact. This agent-based response of the robot is robust, scalable and adaptable, similar to the agent-based response of the system to impact detection, and the overarching principles of the project.

4.1.4 Structure of the Chapter

The next sections contain more detailed descriptions of the two major components of the system, the fixed structure with its embedded agents, and the mobile robotic agent. Section 4.2 describes the fixed structure of the CD and its embedded sensing agents, including details about the sensors, the impacts and their diagnosis, the development of self-organizing algorithms for robot guidance, and details about the ultrasonic signals used by the embedded agents to communicate with the robotic agent. Section 4.3 describes the robotic agent—its hardware, modes of motion and communications with the fixed agents on the CD structure. The communications protocol is outlined in Sect. 4.3, followed by a description of the way in which the communications with the embedded agents are used by the robot for stepping and navigation. These sections are followed by a short description of the system visualizer (Sect. 4.4), a description of an application of the principles of the CD system to a practical damage scenario (Sect. 4.5) and a summary of what has been achieved along with an indication of the next steps in this development program.

4.2 CD Embedded System Components: Hardware and Software

4.2.1 CD Architecture and Hardware

The initial goal of the test-bed is to detect and characterize impacts on the skin, and to form a diagnosis of the accumulated damage. The skin consists of 48 aluminium panels (eight on each side of the hexagon), each of which contains four cells (see Fig. 4.2). Cells are the fundamental building blocks of the system: they are the electronic modules containing the sensing, processing and communication electronics, along with the sensors and the region of skin they monitor. Each cell is an agent of the distributed multi-agent system. It communicates with its four immediate neighbours.

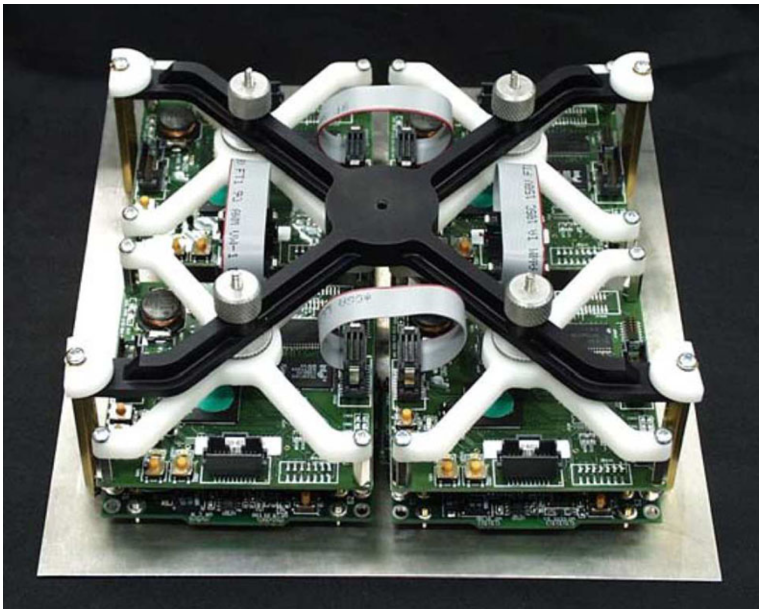


Fig. 4.2 Aluminium panel containing four cells. Each cell consists of a data acquisition sub–module (DAS) below a network application sub-module (NAS). Each cell is connected to its four immediate neighbours, via the ribbon cables that can be seen in the photograph, to form a square network array

Each cell occupies an area of ∼100 mm × 100 mm of the skin, mounted on the inside of which are four piezoelectric polymer (PVDF) sensors to detect the acoustic waves that propagate through the skin as a result of an impact. Thus the complete test-bed contains 192 cells. One of the panels, and its four cells, is shown in Fig. 4.2.

The cell electronics are constructed as two sub-modules mounted directly on top of each other. One of the sub-modules, called the network application sub-module (NAS), contains the communications and processing hardware, while the data acquisition sub-module (DAS) contains the analogue electronics and digitization hardware specific to the attached sensors. A benefit of this division is that the NAS is flexible enough for almost any SHM sensor network application, and only the DAS needs to be changed to accommodate the different sensors that may be required in different applications. Further details of the electronics can be found in Hedley et al. (2003).

4.2.2 Piezoelectric Sensors

The aluminium panels that form the CD skin have dimensions 200 mm × 220 mm × 1 mm. Each panel has an array of piezoelectric sensors bonded to one side. This consists of sixteen polyvinylidene fluoride (PVDF) discs (110 μm in thickness, 2.5 mm in diameter, coated on one side with silver ink, while the other side is bonded to the aluminium) and four lead zirconate titanate (PZT) discs (0.5 mm in thickness, 2.5 mm in diameter, with fired-on silver electrodes on the two flat faces). The sensors that detect the elastic waves caused by impacts were made from PVDF, as this piezoelectric material has high sensitivity as a receiver (but is a relatively poor transmitter), is inexpensive, is relatively easy to fabricate into transducers, and would not significantly mass-load the surface. The four associated with each cell are bonded to the aluminium in a square of 75 mm on each side, directly under the corresponding DAS board. The single transducer at the centre of each cell, designed primarily as a transmitter to communicate with the robot and, at a later stage, to generate signals for damage evaluation, is made from PZT, which is a more efficient transmitter than PVDF. In the fully populated Demonstrator there are 48 panels, and therefore 768 PVDF and 192 PZT transducers.

The initial role of the PVDF sensors was simply to enable the detection, characterization and location of impacts on the aluminium skin. Each group of four PVDF sensors in each cell would detect elastic waves resulting from an impact travelling on the aluminium plate, and the agent would use this information to determine the location and severity of the impact. This role has now been expanded: these sensors also act as the receivers of communication signals from the mobile robotic agent.

The PZT transducers, located in the centre of each group of four PVDF sensors, fulfil a number of functions. They can be used as transmitters to send ultrasonic waves through the panel for subsequent detection by the PVDF sensors before and after impacts on the panels. This allows firstly the calibration of the PVDF sensor (or at least a measurement of their sensitivity), and secondly the possible determination of the state of damage of the panel after the impact has occurred. The PZT transducer in each cell is also used to transmit communications from the embedded agent to the mobile agent when it is positioned on the skin in the region of that cell.

4.2.3 Impacts and Simulated Damage

As outlined above, the CD's present function is to detect impacts and diagnose the level of damage to its operation. The software system has been designed to distinguish between hard impacts (those with a high impulsive force) resulting in damage to the panel (a critical impact, requiring attention by the robot, or mobile agent), lesser impacts (low impulsive force) resulting in damage that does not need immediate repair (a non-critical impact), and electronics and/or communications failures not caused by an impact. During system development so far, real damage has been avoided, to avoid costly replacement of panels. The simulation of the two levels of damage has been done by using a pendulum to strike the panel with either a high impulsive force (for a critical or hard impact), or a low impulsive force (for a non-critical or soft impact). Neither of these types of impact cause damage to the panel, but they do generate elastic waves in the panel of higher or lower amplitude respectively. By manually attaching a red marker to the cell that has suffered a hard impact (or a green marker for a low impact), a visual difference between critically and non-critically damaged cells is provided. This allows the secondary inspection system (using the robot with a small video camera and simple frame-grabbing software to determine the colour of the marker) to visually distinguish between the consequences of hard and soft impacts by colour recognition rather than by visually detecting real damage such as a penetration of the panel. The location and diagnosis of the damage by this secondary inspection technique may then be compared with estimates made from the piezoelectric sensor data.

During system development and testing, repeatable hard and soft impacts were applied by using a pendulum apparatus, which could be set to deliver any impulsive force repeatably and reliably, only limited by the highest potential (and hence kinetic) energy obtainable from the pendulum. The apparatus was held against a panel (using three felt-covered stand-offs), and the pendulum drawn back to the desired and repeatable height for striking the panel with the selected impulsive force.

Following such an impact, the PVDF sensor data was used to estimate the position (using the triangulation method outlined in Prokopenko et al. 2006) and severity (using self-organized maps, as described in next sub-section) of the measured impact.

4.2.4 Impact Signals and Sensor-Based Diagnosis: Use of Self-Organized Maps (SOMs)

A general discussion of the approach to damage diagnosis by self-organization is given in Price et al. (2004). In this case self-organizing maps (Kohonen maps, see Kohonen 2001, 2003) have been implemented to classify impact severity, distinguishing critical impacts for which the skin has been ruptured, from non-critical impacts. A previous discussion of the application of SOMs to the analysis of impact data is given in Prokopenko et al. (2005b). Electronic failures, which are detected when a cell loses its communication capability, are distinguished from critical impacts that have damaged the electronics by the absence of an impact recorded by a neighbouring cell.

The aim of this signal-based diagnosis is to identify high- and low-severity impacts in different regions of a panel: specifically, whether an impact has occurred within the cell that has recorded the signals or within one of the other three cells of the panel. In general, a cell's sensors will detect an impact that occurs anywhere on the panel on which the cell is located, but usually not if the impact occurs on another panel. The diagnosis should be able to unequivocally identify on which cell the impact occurred (even if that cell has been damaged to the extent that it can no longer communicate), an approximate position within that cell, and whether the impact was of high or low severity.

Training of the SOMs was done on a single panel, using hard and soft impacts produced by the pendulum apparatus at a number of locations within each of the four cells on the panel.

The initial aim was to use the SOM to identify the following four conditions:

- A soft impact occurred within the cell.
- A hard impact occurred within the cell.
- A soft impact occurred outside the cell.
- A hard impact occurred outside the cell.

If this diagnosis can be made for each cell on a panel that has suffered an impact, then the cell on which the impact occurred can be identified unambiguously.

For a particular cell on the panel, 100 soft and 100 hard impacts at random positions within the area of the cell were sampled. These samples covered most of the cell's area thoroughly, giving roughly two impacts per square centimetre within the cell. Further, for impacts outside the cell, 100 soft and 100 hard impacts were sampled from the areas of the three remaining cells.

An impact event is recognized when a sensor signal threshold is exceeded. The cell's DAS board then acquires 256 samples from each of its four sensors. It cannot be assumed that any of the signals reflect an accurate time of arrival of the impact pulse, because of the use of a constant threshold. In order to reduce the memory requirement for each SOM, and to maximize the size of the SOM array that can be stored, the string length was reduced to 64 by 4-point sub-sampling of each 256-sample signal. A data input vector used for training the SOM consists of the 64-point data string from each of its four sensors in the relevant cell. This allowed a 10×10 SOM array to be stored in binary format in 50 kB of flash memory on each agent.

In order to evaluate the accuracy with which signals can be identified with the resulting SOM, only half of the training signals were used to train the SOM and the other half were used to evaluate the accuracy of recall and precision. Several separate runs showed a consistent accuracy of $\sim 93 \pm 1$ %, independent of the signal trigger threshold over a range from ~ 1.2 % to ~ 3.6 % of the maximum signal amplitude. Given that each impact is detected by four cells, the probability of an incorrect assignment of an impact location is very small.

For greater accuracy, the SOM on each agent could have been trained separately —this would have individualized the SOMs to take account of the inevitably different sensitivities of the sensors in different cells. Ultimately, it is expected that SOMs will learn on-line, so even if they are initially trained with a common data set, the subsequent learning will develop a set of individualized SOMs.

So far the SOM-based impact diagnoses have proved to be highly reliable, but tests to date have all been carried out using the same impact mechanism with which they were trained. It remains to be seen whether the SOMs will retain this high accuracy with impacts of different origin but similar spectral characteristics.

4.2.5 Self-Organized Robot Guidance: Tracking Field Algorithms

There are two related methods by which navigation of the robot may be achieved, both directed by self-organization. Firstly, algorithms based on ant colony optimization (ACO) (Dorigo and Di Caro 1999; Dorigo and Stützle 2004; Prokopenko et al. 2005a) have been developed in earlier work to link sub-critical impact locations by a simulated pheromone trail and a dead reckoning scheme (DRS) that form a minimum spanning tree (Prokopenko et al. 2005a). The decentralized ACO-DRS algorithm has low communication cost, is robust to information loss within any individual cells, and allows navigation around critically damaged regions in which communication capability has been lost.

An alternative scheme evaluated in Prokopenko et al. (2005a) is a distributed dynamic programming algorithm, employing a gradient field (GF) set by each impact cell. In this Chapter this field is known as the tracking field (TF) as it determines the track the robot will follow: the robot moves along the steepest gradient of the tracking field.[1]

While the concept of the robot following the self-organized pheromone trails produced by ACO is appealing, there is a trade-off between the low communication cost of the ACO-DRS algorithm and a better quality of the minimum spanning tree approximation computed by the gradient-based algorithm (Prokopenko et al. 2005a). The TF algorithm also ensures that each cell in the system has a valid field value: the ACO-DRS algorithm does not guarantee a pheromone value in every cell.

The approach that has been implemented is the tracking field (TF) algorithm. The basic principle of field propagation is very simple. All cells are initiated with a high integer value for the tracking field (255). When a cell detects an impact, its field value is set to zero. At each time-step, each cell compares its field value with that of each of its neighbours. If a cell finds a neighbour with a field value lower than its own, it takes this value plus one as its new field value. This is repeated at subsequent time-steps until a stable tracking field is produced over the whole array of cells, i.e. the field produced by the impact propagates throughout the system of agents. It is independent of the number of neighbours each cell has, so it is robust in the presence of failed cells. Multiple impacts produce a tracking field with multiple minima, analogous to a topographic map of a surface with minima that correspond to impact sites, or the surface of a trampoline that has a number of separated weights on it.

Figure 4.3 illustrates a grid of 23×15 simulated cells each containing a number representing the value of the gradient field (at that cell position) that has resulted from three impacts (denoted by black cells). The positions of two robots are denoted by the two white cells, and the shortest paths found by the robots from these positions to the impact sites are shown in dark grey. Non-operational cells that are unavailable for the robot to traverse are denoted by crosses. As each damaged cell is attended to, the gradient field will be updated and the robot(s) will move to the remaining damaged site(s).

The modification to the tracking field (when the damage is repaired or inspection shows that it may be ignored) is achieved as follows. The previously impacted cell resets its impact flag, and then increases its field to a value that is one greater than that of its neighbour with the lowest field value. All cells then check all their neighbours. If a cell has a field value lower than those of all its neighbours, and it hasn't

[1]The tracking field referred to here is identical to the gradient field defined by Prokopenko et al. (2005a). The robot's path is determined by the gradient of this field.

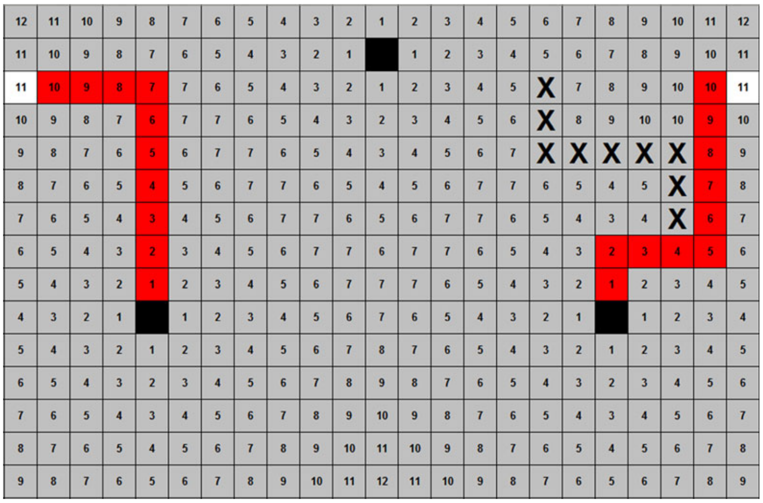


Fig. 4.3 A grid of simulated cells showing the value of the gradient field in each cell resulting from three impacts (*black cells*). The initial positions of two robots are shown with *white squares*. Crosses denote cells that are not operational. The paths taken by the robots to the impact sites are shown in *dark grey*

been impacted, its field value is incremented by one. The new field values reach equilibrium in less than the time it takes the robot to perform one step. Repetition of this algorithm removes the minimum associated with the repaired cell. This action is analogous to one of several separated weights being removed from the surface of a trampoline.

To allow the robot to identify and respond to different classes of event, a tracking field could be set up with minima of varying depths corresponding to the severity/importance of each event. However, steps would have to be taken to avoid the complication of the robot being attracted to nearby minima in preference to more distant but perhaps more important minima. A simple solution to this problem is to model different classes of event on separate tracking fields, and set all minima on a single tracking field to be equally important (deep). The multiple classes of tracking field may all model different routes to their respective sites of importance. This approach means that the robot makes the decisions on which field to follow. One way to process this tracking information is for the robot to respond to events in order of priority by exhausting one class of field before switching to a field class with lower priority, and so on. It should be noted that each time the robot makes contact with the panel, the value of the tracking field for each type of damage is communicated to it. If, for example, the robot is following a lower priority tracking field such as the non-critical impacts field, and a critical impact occurs, the appearance of the updated critical impact field values will cause the robot to follow this higher priority field on its next and subsequent steps.

This multiple gradient field solution has been implemented and currently the robot responds to four classes of gradient/event:

- High severity impacts, representing critical damage—perhaps penetrating impacts and/or cell destruction.
- Low severity impacts, representing non-critical damage—non-penetrating impacts and damage which does not affect system behaviour.
- Damage not caused by an impact, such as communications and/or electronic failure.
- Dock. Models the shortest route to the robot's docking station for re-charging, downtime, etc.

The robot can adopt different criteria to prioritize the order in which it visits impact sites. A possible modification to the behaviour outlined above would be to allow the robot to visit sites of lesser importance if they are on or near the intended path to a site of greater importance.

Major benefits of this algorithm are its stability and its fast dynamic response. A mobile agent in the system may determine the direction of the shortest path to the nearest impact location through interrogation of the local cell group. This is analogous to a ball on a slope; the ball need not know where the bottom of the hill is to know which way to roll—simply knowing the gradient of its local piece of hill is sufficient. In the TF algorithm, the shortest distance to an impact location can be determined simply from the value of the field in a particular cell, since each cell increments the lowest gradient value of its group of neighbours by one unit. Note that this algorithm, with a separate tracking field for each type of damage, does not suffer the 'local minima problem'. Within each tracking field each damage site has equal priority, with the same 'depth' or field value—damage of different priority is represented by values contained in different fields.

Figure 4.4 shows a real tracking field produced on the CD as a result of two non-critical impacts, located in the black cells. The field value at each cell is indicated by the shade of grey. White cells are those with which the robot cannot communicate. This may be due to electronic failure or, as is the case here, sensors have not yet been fitted.

4.2.6 Communications with the Robot: Distinguishing Impact and Communication Signals

Communication between an embedded agent (cell) and the robot utilizes ultrasonic signals propagated through the aluminium skin of the cell. The robot transmits and receives ultrasonic signals using transducers mounted in the centre of each foot. Further details about the robot's transducers and communications are given in the next section. The agent transmits via the PZT element in the centre of the cell, and receives signals through one of the four PVDF elements that are used for impact detection.

A communication is initiated when the robot places one of its feet on the region of skin monitored by agent A (say). In order to initiate a communication sequence,

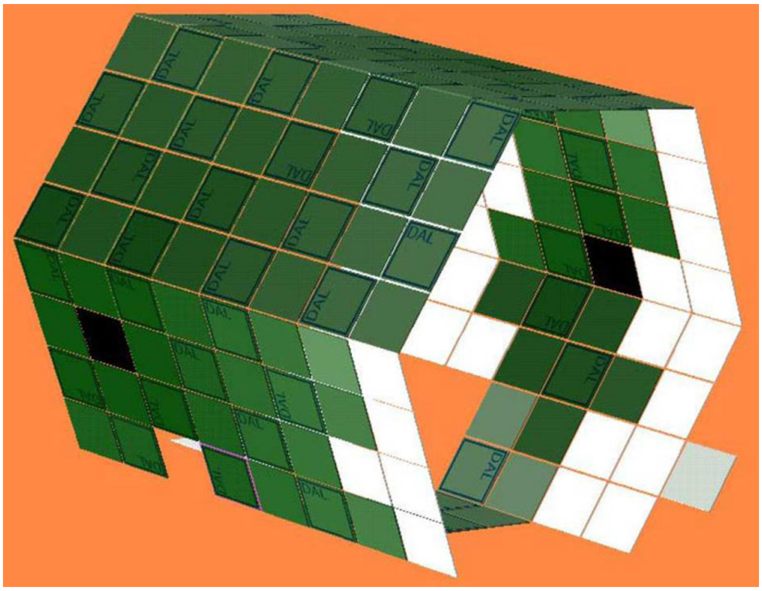


Fig. 4.4 An image of the tracking field produced by two non-critical impacts that occurred at the *black cells*. The *squares* indicate the cells on the surface of the hexagonal prism test-bed. The field values are shown as shades of *grey*, with a lower field value being darker. The *white cells* are those with which the robot cannot communicate. Absent cells in the image are those that have been physically removed from the CD

the robot then transmits a tone burst from the ultrasonic transducer in this foot, which consists of 5 cycles at a driving frequency of 400 kHz. This corresponds approximately to the lowest order radial mode of the robot transducer disc, and it excites the zeroth order anti-symmetric (A_0) guided elastic wave mode of the aluminium plate (see e.g. Rose 1999). The agent *A* distinguishes this signal from that due to an impact on the basis of its spectral content.

A 5-cycle tone burst has a spectral width of ~20 % of the centre frequency, and this will be increased by the spectral response of the transmitting and receiving transducers. Nevertheless, it is expected to have significantly different spectral characteristics to an impact-generated signal. At this stage the agents use a simple combination of the responses of two band-pass filters with different cut-off frequencies to distinguish impact events from communications events, but more sophisticated processing could be readily implemented.

A communications sequence is completed when the agent receives a specific acknowledgement signal from the robot. The issue of an impact that occurs during a communications sequence has not yet been dealt with: at this stage it may result in a corrupted communications packet, but will not otherwise be recorded. This is not an urgent issue for the present system, since the robot foot would shield the cell during a communications sequence, though an impact might damage the robot. However, it raises potential issues of impact damage to the robot, and impact damage to the cell when much smaller robots are in use.

More details of the cell-robot communications are given in the next section.

4.3 The Mobile Robotic Agent

An important feature of the CD system is an ability to support mobile (robotic) agents that can roam the exterior surface of the test-bed, communicating with the fixed agents embedded in the underlying structure. The function and operation of such an agent will be described in this section, and it should be emphasized that it is not controlled centrally, but cooperatively with the network of fixed local agents with which it communicates.

It should also be emphasized that the system described here is no more than a test-bed, whose primary purpose is for investigation of the practicality of the self-organized complex system approach to damage diagnosis and response. Thus, details of the specific hardware implementation (such as the use of air suction for the robot's attachment, which is obviously inconsistent with a space-based application) are not considered to be important at this stage. While the present implementation of the robot is bulky and represents a single point of failure, the eventual aim is to develop a swarm of very small robots that can perform internal or external tasks cooperatively. The work described in this Chapter represents a first step towards that ultimate goal.

Why is a robotic agent needed in an SHM system? When sensing impacts using passive sensors, the information received may be insufficient to characterize the damage, and where damage is detected it may need to be repaired. One approach to obtaining additional damage data, and to providing a crude repair capability, is the development of a mobile robot that can move around the outside skin (Fig. 4.5).

The robot moves rather like an inch-worm, with its design based on an articulated box section with six degrees of freedom. The joints are driven by commercial model aircraft servos and have no position feedback to the controlling processor. The robot is equipped with six suction cups on each of its two feet, and a pneumatic system with a variable speed vacuum pump and electrically controlled valves that allow it to selectively attach and detach its feet to and from the surface. To allow the robot to find the surface and attach to it reliably there are two optical rangefinders on each foot that measure the distance to the surface and the angle of the surface. A lithium polymer battery supplies power to the robot for approximately 30 minutes of operation before recharging is necessary.

The robot has two modes of locomotion. The first mode is very much like an inch-worm, as mentioned above: to move forward the robot alternately stretches out and contracts whilst detaching and attaching its feet in sequence. The second mode requires the robot to detach one foot, pivot 180° around the other (attached) foot and then reattach the first. It can change direction by pivoting through any angle up to 360°. Initially the robot will carry two small video cameras, one on

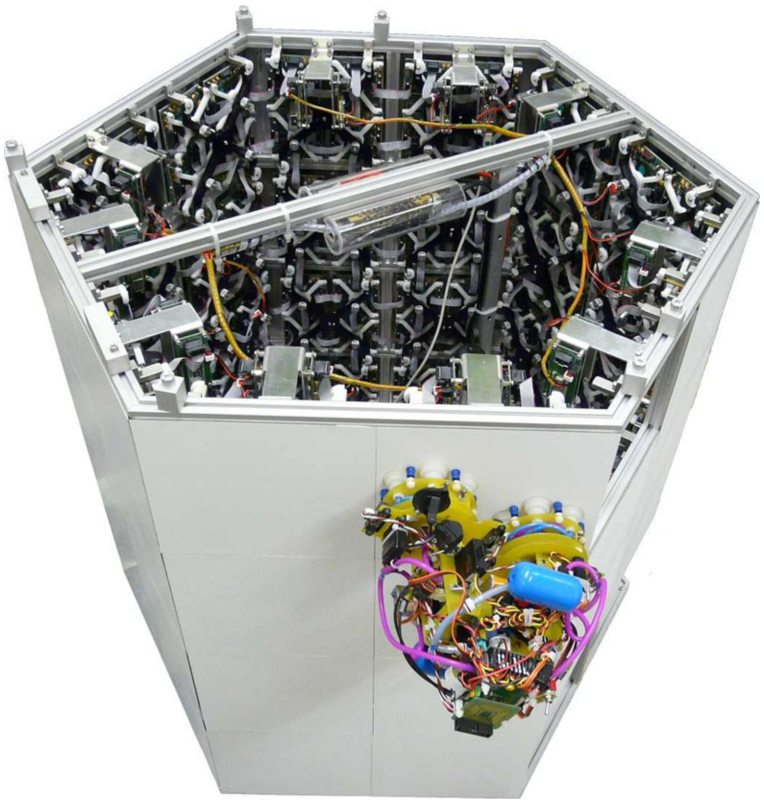


Fig. 4.5 The robot on a vertical face of the CD test-bed

each foot, which will send images back to the network for further analysis. In the future other sensors may be included, such as an active ultrasonic transducer that can interact with the piezoelectric receivers embedded in the skin for ultrasonic damage evaluation.

The robot communicates with the fixed agents in the network using piezoceramic (PZT) ultrasonic transducers in both feet to pass messages through the aluminium skin to the underlying cells. The fixed agents receive messages via one of the four piezoelectric polymer sensors that are used for detecting impacts. A fifth transducer, in this case a piezoceramic, has been added at the centre of each cell for transmission of messages from the cell to the robot.

Further details concerning the physical communications mechanisms and the communications sequences have been given by Hoschke et al. (2008).

Because the robot has no global navigation capabilities and can only move from one cell to the next using dead reckoning, large positional errors could rapidly accumulate as the robot moves over the surface. To avoid such positional errors, the underlying cell measures the robot's foot position by triangulation, as described above, and reports it to the robot. The robot can then either physically correct the foot position, or take it into account in calculating the step required for the next move.

A complication with this method of positional feedback is that the cell and the robot must have a common knowledge of the orientation of the cell relative to other cells and the structure. One way to avoid this issue is to build the CD structure with all cells in a prescribed orientation. However, it can be argued that this solution is inconsistent with the concept of an adaptive, self-organizing structure, and a more satisfactory solution involves the cells cooperatively determining their relative orientations. This is also necessary for algorithms such as ant colony optimization.

Nevertheless, there is a need for the robot to have some basic knowledge about the structure, since it cannot be allowed to step on the gaps between panels, and it needs to know where the face edges of the prism are located in order to be able to step from one face to another. A robot with more computational power than the present one could, for example, use its video camera to resolve local issues such as these, but for the time being the robot has been given this basic knowledge of the cell layouts.

The robot's navigation and functions are determined cooperatively with the local agents embedded in the test-bed skin with which it is in contact. The robot navigates around the surface of the test-bed using tracking field data available from the underlying cell to which it is attached at the time. This data is specific to the cell's local neighbourhood, and does not contain any global information about the system. Further information about the tracking fields was provided in Sect. 4.2 above.

4.4 The System Visualizer

As described in earlier reports (Prokopenko et al. 2006; Price et al. 2004; Hoschke et al. 2008), the system visualizer is a computer that is used for initializing the multi-agent system and displaying the state of the system, but which plays no essential role in the operation of the system. It can be attached via a USB port at a number of points around the edge of the CD system (in principle it could be anywhere), and its function is to request state information from the embedded agents and display this information.

The visualization function also shows the robot position, and this is illustrated in Fig. 4.6. This information is not obtained from the robot itself, which in principle doesn't need to know its absolute position on the structure, but from the agents with which the robot is communicating. Thus, an observer doesn't need to be able to see the robot to be able to monitor its activities. This principle can of course be extended to more than one robot.

Four views of the Concept Demonstrator have been developed to display and debug the gradient algorithm. Each view uses a different colour and represents the value of the gradient field on a particular cell by the shade of colour (Fig. 4.4). Lighter shades represent higher gradient values, which are further away from impact locations. Cells that the robot cannot reach because they have no DAS board

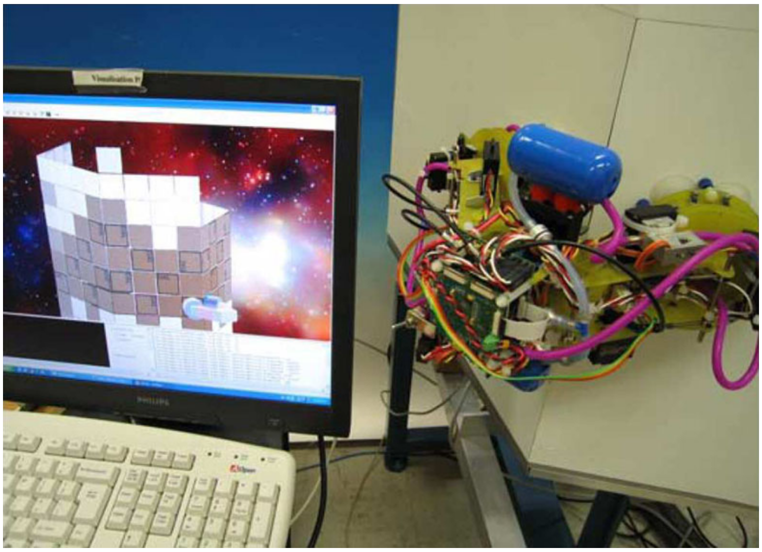


Fig. 4.6 Visualizer screen (*left*), showing an animation of the robot in its actual position on the CD structure (*right*)

attached are displayed in white. By double-clicking on a cell in the display the numerical values of the cell's gradients are displayed.

While the damage sensing capability of the system is currently limited to a video camera mounted on the robot, in the longer term it will be desirable to make a diagnostic decision based on data from a range of sensors, possibly with the assistance of material or structural models. The use of data from multiple sensing modalities for damage diagnosis and prognosis will be the subject of future work.

In cases when major damage occurs suddenly, which will usually be the result of an external influence such as an impact, it may be necessary to initiate a response to an initial indicator (such as the sensing of the impact) without waiting for a subsequent detailed inspection and diagnosis. In such a perceived emergency situation a precautionary principle must clearly be adopted: rapid action should be taken first and a more detailed diagnosis made later.

4.5 A Practical Damage Scenario: Impact Damage in Thermal Protection Systems

4.5.1 Introduction

Although the task described in the preceding sections required further work to enable the robot to perform functions required to gather diagnostic information, such as the manipulation of a video camera or some other sensing function, subsequent work has concentrated on adaptation of the system to apply it to a practical damage scenario: the detection and evaluation of impact damage in the thermal protection systems of spacecraft.

The thermal protection system (TPS) of a vehicle that travels at high velocity in a planetary atmosphere, such as a spacecraft re-entering the atmosphere, or perhaps a future hypersonic aircraft operating in the atmosphere, is the material that protects the vehicle from frictional heat damage or incineration. There are a number of types of materials that have been (and will be) used for thermal protection of space vehicles. These may be usefully categorised as re-useable (or passive) and non-reusable. The ceramic foam tiles and carbon-carbon composites used on the NASA Space Shuttle fall into the first category, while the ablative materials such as that used on the Apollo spacecraft (e.g. Venkatapathy et al. 2010) and a number of others, fall into the second. The relatively new structurally integrated TPS materials are intended to be re-useable.

The main source of potential damage is considered to be impacts by foreign objects such as micrometeoroids or orbital debris (MMOD). MMOD consists of millions of man-made debris particles and naturally occurring micrometeoroids orbiting in and around Earth's space environment at hypervelocity speeds, typically $\sim$10 km/s (22,000 mph). At these velocities even small particles can produce significant damage to the TPS layer. Impact damage during launch, such as occurred to the Space Shuttle Columbia during STS-107, must also be considered a possible hazard. Thus, the primary source of damage for which the SHM system is designed is that due to impact by high-velocity (or high-energy) objects.

The aim of the SHM system is to detect and evaluate damage to the TPS prior to entry into the atmosphere. It is assumed that if the TPS is undamaged on entry, and if the entry is made at an appropriate velocity, angle and vehicle orientation, it will function correctly and protect the vehicle from catastrophic failure. Once atmospheric entry is under way it is generally considered to be too late to respond to malfunction of the TPS.

A straightforward application of the system outlined in the preceding sections could, in principle, have satisfied the requirements of this task, but it was decided to adopt a different approach to characterising the damage by monitoring its effect on the relevant functional properties of the TPS material. The primary concern, at least for passive materials, is whether any damage to the TPS has adversely affected its thermal resistance properties, although its ability to retain structural integrity in the presence of the large thermal and mechanical stresses of re-entry are also important. The initial focus of this work will be on monitoring the thermal resistance of the TPS layer, i.e. on monitoring the functional consequences of damage rather than the damage itself.

The general requirements of the SHM system are as follows:

1. It should detect impacts on the TPS materials, locate the positions of impacts, and evaluate the severity of damage caused. The effect of the damage on the thermal properties of the TPS is of primary concern.

2. A further requirement is the ability to scan the TPS surface to verify the thermal integrity of the layer prior to re-entry. This could be carried out as a routine safety check, regardless of whether or not an impact had been detected during the flight.
   As described in Sect. 4.1.1, further requirements are:
3. Robustness to sensor failure and structural damage. Components of a sensing system can fail for a variety of reasons, including damage that is consequential to structural damage of the TPS (e.g. impact-induced damage). It is a requirement that the TPS health monitoring system should continue to operate effectively in the presence of moderate amounts of component failure or damage.
4. The system should be scalable, or capable of monitoring a large number of sensors without loss of performance or response time.

With these requirements in mind, the SHM system was designed following the general approach outlined in Sects. 4.1 and 4.2 above. Impacts are detected and located using a network of passive piezoelectric acoustic emission sensors that monitor the elastic waves produced in the TPS material by the impacting particle. Damage evaluation is conducted as a second stage of sensing, by monitoring the thermal conductivity of the material in the region of the impact. This is done using temperature sensors embedded within the TPS material, which measure the temperature rise produced by an externally-applied heat source. The heat source could be naturally occurring, such as the Sun, or a localised heat source applied by, for example, the mobile robot described above. This process is effectively a thermographic technique using embedded thermal sensors. Damage will appear as a local anomaly in the measured temperature distribution in the region of the impact.

In order to satisfy requirements 3 and 4, a local agent-based system architecture is employed as outlined above. The local agents control and monitor the piezoelectric impact sensors, they "order" the application of the heat source, monitor the thermal sensors, and collectively evaluate the presence and significance of a thermal anomaly associated with impact damage.

The purpose of the work undertaken to date is to design and build a laboratory-scale demonstration that will establish the feasibility of this novel, robust SHM system. This is not yet complete, but the remainder of this section will outline fundamental aspects of the design and operation of the system, and describe results of the testing carried out on fabricated components of the demonstration system. Aspects of this work have been reported by Scott et al. (2009), Hoschke and Price (2012) and Hoschke et al. (2013).

4.5.2 *Design Fundamentals of the Agent-Based Thermal Sensing System*

The earlier sections of this Chapter have described the design and implementation of an agent-based network of piezoelectric sensors for detecting and locating impacts. The general issues discussed there apply to this application, with the exception that the acoustic properties of TPS materials must be taken into account in designing sensor placements and the impact location algorithms. These issues will be discussed in the next sub-section. The main focus of this sub-section is on the network of temperature sensors required to evaluate the effects of impact damage.

The most obvious approach to measuring temperature distributions in this situation is to use an array of local electronic temperature sensors attached to each agent, similarly to the piezoelectric acoustic emission sensors. Thermocouples or resistance sensors would be likely candidates. However, this approach has the following draw-backs:

- It is likely that a higher density of temperature sensors than piezoelectric sensors will be required. Elastic waves propagate over relatively large distances, but the lateral spatial extent of a damage-related thermal anomaly will depend on the location of thermal sources/sinks and the thermal conductivity of the material, and may be relatively small. This means that a relatively large number of sensors would be required to be connected to each agent.
- Each sensor would need to be separately wired to its local agent. Individual wiring of large numbers of sensors will add both significant weight and complexity to the structure, neither of which is desirable.
- Importantly, for TPS materials with very low thermal conductivity and specific heat, the presence of a web of wiring is likely to significantly affect the thermal properties of the structure, and seriously perturb the measurements required of the sensor network, i.e. the presence of the sensors may complicate, or even compromise, the interpretation of their measurements.

An alternative means of measuring temperature within or on the surface of a material with a high sensor density is through the use of optical fibre Bragg grating (FBG) sensors. Bragg gratings written in optical fibres are sensitive to temperature through the thermal expansion of the fibre and the temperature-dependence of the refractive index of the fibre core material, the latter being the dominant effect. Optical fibres can contain a large number of sensors on a single fibre (typically at $\sim$1 cm spacing), and the thermal conduction of an optical fibre is very low: the fibres are thin (typically $\sim$150 μm diameter for a single-mode clad fibre) and the thermal conductivity is orders of magnitude less than for a metal wire. Therefore, both the weight and thermal conduction problems associated with electronic sensors are greatly reduced by the use of FBG sensors.

However, the use of FBG sensors for temperature measurement in a local sensing architecture is not straightforward, and presents the following challenges.

1. The first and foremost challenge is that a fibre containing FBG sensors is essentially a distributed rather than a local sensing system. A single fibre may contain a large number (up to several thousand) of sensors and a sophisticated optical measurement instrument is required to measure the properties of each individual grating. While a single FBG is a local sensor, the principal advantage of FBGs it to obtain a large number (density) of sensors—a distributed system. At this stage of technology development it is not practical to have a separate sensing and measurement system incorporated in each local agent.

2. While one of the strengths of FBG sensing is that a single fibre can contain many sensors, it is also a weakness in situations where there is potential for damage and a robust sensing system is required: a single break in a fibre can remove many sensors from the system.
3. FBGs are sensitive to strain as well as temperature. For accurate measurement of temperature, steps must be taken to eliminate effects of strain on the measurement. Various methods have been developed and reported in the literature to deal with this issue, so it will not be considered further here.

A novel approach has been developed in this work to address points 1 and 2. The critical idea, introduced by Scott and Price (2007, 2009), is to employ a switched network of optical fibres in which the routing of the light through the fibres in the network is controlled electronically by the local agents. In this way the agents control in which areas of the structure sensing takes place, and broken segments of fibre can be by-passed by re-routing light around them. A schematic showing the principle of the approach is shown in Fig. 4.7.

It is clear from the schematic of Fig. 4.7 that the local agents control the routing of the light from and back to the measurement instrument. The multiplexer (large box at top) is controlled by another agent. Operation of the system is as follows.

When an impact has been detected and its location identified, the agents act collectively to establish the shortest path for laser light from the measurement instrument to reach the region of the impact. Segments of fibre known to be damaged are avoided. This is a similar procedure to the guiding of the mobile robot to the site of an impact, as described in Sect. 4.2.5 above, and the algorithm developed to achieve it is outlined below. Because the light path is established by self-organization, the measurement instrument neither has nor requires any information about it.

Results of the measurement take the form of temperature vs path length, or grating number, along the measured fibre path. This data is returned to the agents, being communicated by a back-propagation method. The data string is communicated back along the route of the light path. Each agent along the path holds information about which of its fibre segments was active, and the number of gratings in that segment. The agent controlling the first segment along the path keeps the data measured from its fibre segment(s) and passes the remainder of the data string to the next agent along the path. This process continues until all agents along the path have extracted the data that corresponds to their active fibre segments. This data is then used to construct the temperature distribution in the region near the impact.

When used as described above, the essentially non-local FBG measurement system can be operated as a group of local sensors controlled by the local agents. Furthermore, the robustness of the system is enhanced because a fibre breakage will cause the loss of only the sensors in one segment of fibre rather than in an entire fibre. While the measurement instrument represents a single point of failure in the system, this vulnerability could be mitigated by the use of multiple instruments injecting their light into different points of the network.

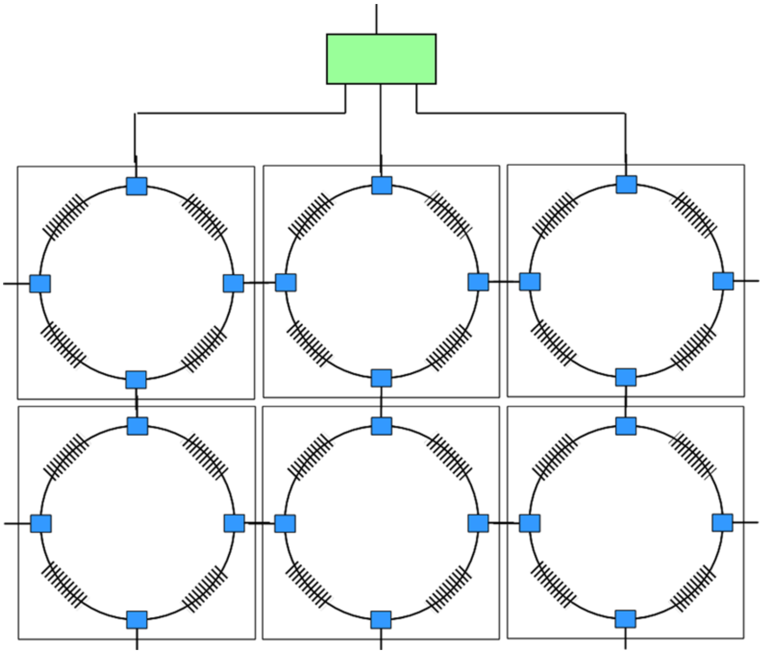


Fig. 4.7 Schematic of a switched FBG sensing network. The *six square cells* are regions of structure monitored by a single agent. Each contains four segments of optical fibre containing FBG sensors, arranged in a loop and joined by *square grey boxes* that each represent an electronically-controlled optical switch. Each switch has three optical inputs/outputs—the direction of light propagation is immaterial—and electronic controls to allow any two to be connected together leaving the third unconnected (open). Each switch is connected to two fibre segments on its own cell and one, without, FBGs, that connects it to a neighbouring cell. The configuration of each switch is controlled electronically by the agent on the local cell. The *large box at the top* is an optical multiplexer that connects the measurement instrument (not shown) to a number of different points in the network

4.5.3 Specific Design and Operating Issues

The demonstrator to be built will be based on a passive TPS that is circular in shape. Because none of the TPS materials developed by NASA were available for this work, a commercial alumina-based ceramic foam, with properties similar to those of the Space Shuttle tile material (see, for example Alers and Zimmerman 1980) has been used. This material, designated ZAL 15 (Zircar 2012), is manufactured by Zircar Ceramics Inc. (NY, USA) and has specified density $\rho = 0.24\ \mathrm{g\,cm^{-3}}$, thermal conductivity $k = 0.06\ \mathrm{W\,(m\,K)^{-1}}$ and specific heat $C = 1047\ \mathrm{J\,(kg\,K)^{-1}}$. It is composed of alumina fibres with 15 % high-purity silica as binder. An advantage of this material over some other available ceramic foams is its fine open pore structure with homogeneous microstructure and consistent binder distribution. It is supplied as boards, of various thicknesses from 0.50 in (12.7 mm) to 1.50 in (38.1 mm), with

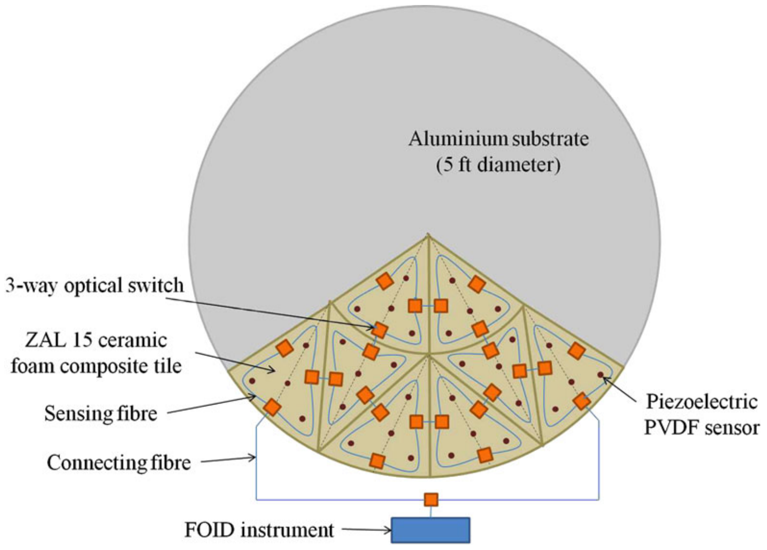


Fig. 4.8 A schematic plan view of part of the proposed cellular structure The four triangular areas that make up each 60° segment of the circular TPS demonstration structure each represents a cell (or tile) whose sensors and optical switches are controlled by a single local agent. Within each cell, the *dark circles* represent piezoelectric AE sensors, the *square elements* represent 3-way optical switches and the *light lines* joining them represent the optical sensing fibres (though the fibre layouts are more complex that shown in this schematic). Each agent controls the routing of the light through the fibre network: communications between the agents enable self-organizing routes to be established between the measurement instrument (FOID) and an impact site

the alumina fibres preferentially oriented in the plane of the board. It is therefore expected to exhibit anisotropic elastic and thermal properties (the thermal conductivity quoted above is for the through-thickness (z) direction of the boards).

The proposed network structure, in this case designed for monitoring a circular heat shield, is shown schematically in Fig. 4.8. The local agents that manage the piezoelectric sensors also control the optical switches that route the light through the fibre network. The instrument that monitors the FBGs is referred to as the fibre optic interrogation device (FOID).

4.5.4 Realisation of Switched Fibre Network

Measurement of temperature in materials with low thermal conductivity and low heat capacity is challenging. If the amount of heat removed by conduction by the sensor is comparable with or greater than the in-flow of heat through the material, the temperature measured by the sensor will not be representative of the temperature in the material in the absence of the sensor. In order that the sensors provide a reliable temperature measurement, the following conditions are required:

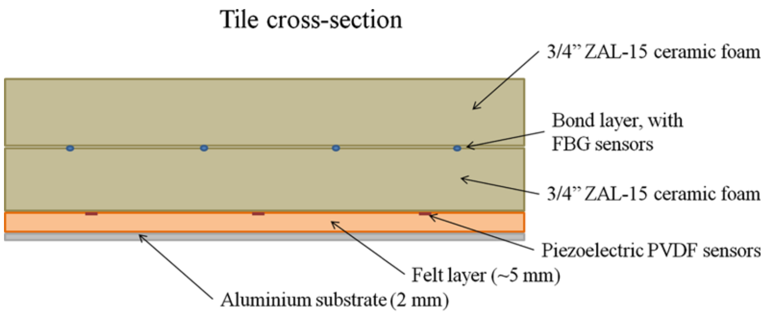
1. The sensors are in good thermal contact with the TPS material.
2. The sensor has a relatively low heat capacity, so its temperature can follow that of the measurement point with minimal heat transfer.
3. The connections to the sensors have low thermal conductivity and small cross-sectional area, to reduce heat transfer from the sensing point.
4. The connections to the sensors are in good thermal contact with a heat bath close to the temperature of the measurement point so that the thermal gradient along the connections is small.

Relevant thermal properties of fused silica, the starting material of the optical fibres, are: thermal conductivity $k \sim 1.3\ \mathrm{W\,(m\,K)^{-1}}$, specific heat $C \sim 703\ \mathrm{J\,(kg\,K)^{-1}}$ and fibre diameter (incl. core, cladding and polymer coating) $\sim$150 µm. Therefore, silica optical fibres satisfy requirements 2 and 3 quite well, and certainly better than do metallic sensors/connectors.

Conditions 1 and 4 are addressed by bonding the fibres between two sheets of ZAL 15 board using an alumina cement, designated AL-CEM, produced by Zircar Inc (Zircar 2012) specifically for bonding these ceramic foam materials. Its thermal conductivity when cured is not known, but it should ensure intimate contact between the fibres and the boards and forms a thin (typically $\sim$250 µm thick) bond layer. The fibre layout, designed to satisfy condition 4 and to provide a high spatial density of FBG sensors, is shown in Fig. 4.9.

The fibres are sandwiched between two ZAL 15 boards, each $\frac{3}{4}''$ (19 mm) thick to ensure they are in good thermal contact with the TPS material and well-insulated from the surrounding environment. Particular care has been taken to provide a long entry/exit path for the fibres from the bond region, via the sides of the tiles (in the gap between the tiles), to the optical switches that are mounted on the substrate below the TPS. The arrangement of the fibres shown in Fig. 4.9 (lower image) was chosen to provide spatial redundancy. The three separate fibre segments on each tile run approximately parallel and each provides sensing distributed over the whole tile area. Therefore if one, or even two, fibre segments are damaged, some sensing capability over the area of the tile is provided by the undamaged segment(s). This would not be as effective if each segment was localised in a different region of the tile.

The FBG sensors are monitored using the technique of optical frequency domain reflectometry (OFDR), initially developed at NASA Langley Research Center (Froggatt 1996; Froggatt and Moore 1998) and further refined at the NASA Dryden Flight Research Center (Richards et al. 2012). This technique is capable of monitoring up to several thousand FBG sensors on a single fibre, all written with the same nominal wavelength, at a rate of $\sim$ 20 Hz. Feasibility testing has been carried out (see below) to ensure that the technique is capable of providing accurate measurements in a switched network, which will have points of attenuation at the switches and fibre connections. Connections may be either permanent splices or removable connectors, but both impose some attenuation on the propagating light.



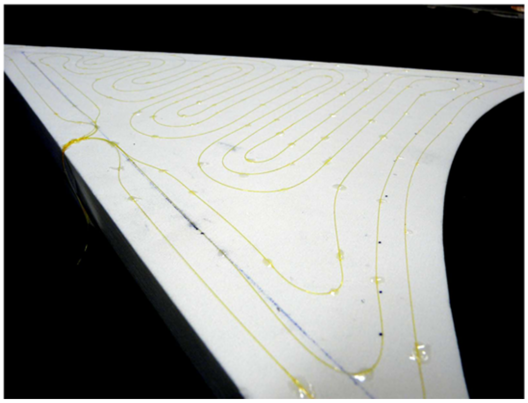


Fig. 4.9 A schematic diagram of the optical fibre sensing layout within a typical TPS tile. The *upper image* is a schematic view of the tile cross-section, and the *lower image* shows the fibre layout prior to bonding the upper ZAL 15 board in place. The *lower image* shows how the three fibre segments are run in parallel, as described, and how the fibres enter/exit the tile

4.5.5 AE Sensor Network

In order to design the layouts of the sensor networks, knowledge of the elastic and thermal properties of the material is required. Detailed measurements of the elastic properties have been carried out, in part using a high-energy focussed laser pulse to generate elastic waves in the material. Such pulses provide a reasonable simulation of high-energy particle impacts, so the measurements give a good insight into the detectability of particle impacts. This work has been described by Hoschke and Price (2012) and Hoschke et al. (2013).

Measurements of the times of arrival of either the first minimum or the first maximum of the received pulse yield a propagation velocity for longitudinal waves in this direction of 0.40 mm/µs. This is very small, and compares with the velocity of sound in dry air at room temperature of $\sim$0.34 mm/µs. Similar measurements for propagation in the (in-plane) x- and y-directions indicate propagation velocities of longitudinal waves in both cases of $\sim$1.89 mm/µs. Thus, the elastic properties of the material are highly anisotropic, with effective isotropy in the plane of the boards: the effective elastic symmetry is orthotropic with planar isotropy. This suggests that, as expected from our knowledge of the material microstructure, the alumina fibres lie in the x, y plane, i.e. in the plane of the boards, with random orientation.

The elastic constant tensor for materials of this symmetry contains only five independent constants, which in the Voigt or reduced matrix notation (see e.g. Nye 1985; Rose 1999), are: C_{11}, C_{33}, C_{12}, C_{13}, C_{44}, with the only other non-zero elements being $C_{22} = C_{11}$, $C_{23} = C_{13}$, $C_{55} = C_{44}$, $C_{66} = \frac{1}{2}(C_{11}–C_{12})$ and $C_{ij} = C_{ji}$ (Nye 1985; Rose 1999; Hoschke and Price 2012). Measurements yielded estimates of all five elastic constants, which are:

$C_{11} = 857\ \mathrm{MN\,m^{-2}}$
$C_{33} = 40\ \mathrm{MN\,m^{-2}}$
$C_{44} = 40\ \mathrm{MN\,m^{-2}}$
$C_{13} = 40\ \mathrm{MN\,m^{-2}}$ or $-120\ \mathrm{MN\,m^{-2}}$ (ambiguous from velocity measurements)
$C_{12} \approx 330\ \mathrm{MN\,m^{-2}}$ (insufficient measurements for reliable estimation).

The fact that $C_{44} \approx C_{33}$ means that shear waves propagate in the z-direction with the same phase velocity as longitudinal waves, another unusual feature of this material.

Alers and Zimmerman (1980) reported measurements of the elastic constants of the ceramic foam used in the Space Shuttle TPS tiles, which is also orthotropic with planar isotropy, as $C_{11} = 240\ \mathrm{MN\,m^{-2}}$, $C_{33} = 46\ \mathrm{MN\,m^{-2}}$, $C_{44} = 21\ \mathrm{MN\,m^{-2}}$, $C_{12} = 145\ \mathrm{MN\,m^{-2}}$ and $C_{13} = 82\ \mathrm{MN\,m^{-2}}$ (with the C_{13} value corrected to account for an apparent error in the authors analysis). Thus, this material is also highly anisotropic, but not quite to the same extent as ZAL 15.

An important consequence of the large elastic anisotropy of the TPS material is that the arrival time at a sensor of the first elastic pulse produced by an impact is almost independent of the distance of the impact from the sensor over a range of the order of 1.5 times the thickness of the material. Calculations of wave arrivals at points at a range of angles from an impact site are shown in Fig. 4.10, for the waves of the three orthogonal polarizations, showing an almost angle-independent first arrival time out to an incident angle of $\sim 60°$. Further details of these calculations are given by Hoschke and Price (2012).

This implies that for tiles of the structure shown in Fig. 4.9 there will be a region of ~ 55 mm radius around each sensor within which the time of arrival of the signal from an impact will be effectively independent of the impact location. This will place a limit on the accuracy with which impact locations can be determined, unless signals can be detected on multiple sensors.

The piezoelectric acoustic emission (AE) sensors are PVDF discs, similar to those described in Sect. 4.2.2 above. There are four on each tile, and therefore four for each of the electronic agents. They are placed with one approximately at the centroid of the tile and the other three on lines that bisect the angles of the tile corners.

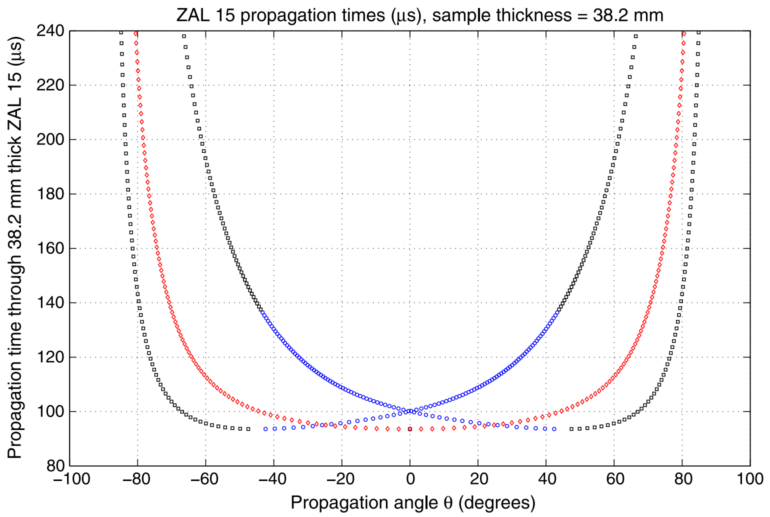


Fig. 4.10 Arrival times of the three principal modes at a range of angles to the sample z-axis for a ZAL 15 panel 38.2 mm thick, with z-axis normal to the sample surface

4.5.6 Self-Organizing Agent-Based Algorithm and Simulation

An initial algorithm to establish the optical path connecting the FBG measuring instrument (FOID) to the impact region has been developed and trialled in a computer simulation. This algorithm is based on the tracking field algorithm developed for guiding a mobile robot to a damage site, described in Sect. 4.2.5 above. This basic algorithm will be described briefly, results of simulations shown, and aspects that require modification outlined.

Within the distributed multi-agent system architecture, the FBG network requires distributed local algorithms (i.e. algorithms that reside and run on the individual local agents) to control the optical switches that will determine the path by which the probe laser light will propagate through the network structure. The goal of the algorithm is to form a connected path through the network from the OFDR measurement system (FOID) to an agent that had detected an impact. The FOID, which controls the probe laser and processes the returned optical signals, is required to communicate with the local sensing agents on the TPS. Just as in the work described in previous sections, in which an autonomous robot was treated as another agent in the multi-agent system, in this case the FOID will be treated as an agent with communication links to some "neighbouring" agents.

The tracking field algorithm described in Sect. 4.2.5 results in a contour map set up by the local agents. This map essentially maps the distance of an agent and its physical cell from the damage site, and is generated by two rules; firstly if an agent senses an impact in its own cell it sets its tracking field value to 0; all other agents set their field values to 1 higher than the minimum value of their neighbours. This

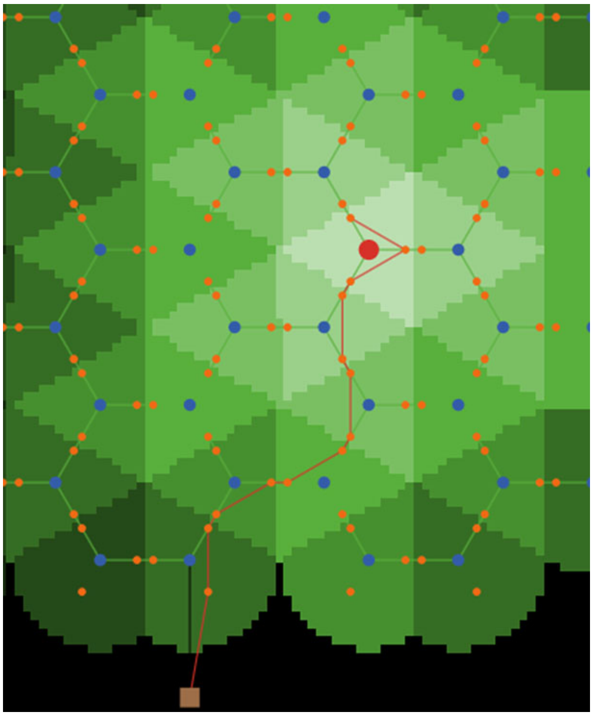


Fig. 4.11 A connected optical path from the FOID (*square at the bottom*) to an impacted agent (*large grey circle*) for triangular cells. The agents that control the cells are shown as *dark grey circles*, and the optical switches as *smaller light grey circles*. Cells filled in *lighter shades of grey* are closer to the impacted cell

means agents directly adjacent to impacted agents will have a field value of 1, agents connected to them will have a value of 2 and so on (see Fig. 4.3).

The optimal path through the fibre network is then determined using an algorithm that employs a connection message—an information packet that can be communicated around the network. The FOID sends a message that travels through the network following the maximum gradient of the contour map and configures switches appropriately as it goes. When an agent receives the connection message it passes it on to its neighbour with the lowest tracking field value. In the case that two neighbours have equal field values the choice is made randomly (this is an issue that needs further attention, and will be discussed further below). Once the message has reached an impact site, for which the agent has a tracking value of 0, the connection message follows the path it has established back to the FOID, and the measurement process for the sensors along the path is initiated.

As shown in Sect. 4.2.5, the tracking field algorithm can handle multiple impacts if required, leading to more complicated contour maps. After an impact region has been inspected using the optical fibre sensors the impact can be cleared as described in Sect. 4.2.5. The contour map then reorganizes and the cycle can start again to connect the FOID to the next impact location. Figure 4.11 shows a path formed with the algorithm using the connection agent

The measurement process results in a set of temperature (in this case) measurements as a function of the propagation time, or equivalently of the distance, along the probed fibre. The FOID has no information that allows it to associate these readings with physical locations in the structure, since it has no information about the path the light has taken through the network. However, these measurements are returned to the network, along the switched optical path, and the agents can deduce the data readings appropriate to their own sensors. Therefore, the measurement process results in the local agents receiving local sensed information, even though a wide-area sensing system was used to obtain the data.

An important requirement of the system is that it must be able to form connected paths, and preferably minimal connected paths, that avoid damaged sections of the network. The detection and avoidance of failed agents has been discussed in Sect. 4.2.5, but there is also a requirement in this system to detect and avoid damaged segments of fibre.

Damage to the optical fibre network can be detected either by periodic scanning of the entire network, or more localised scanning following the detection of an impact. Scanning is done by a procedure that systematically simulates an impact at all cells in turn. Optical paths joining the FOID to each agent would be sequentially established, and sensor information returned to the agents along each path. A damaged fibre segment would be revealed as a large broadband reflector (assuming that physical damage to a fibre, such as a break, would reflect all wavelengths in the relatively narrow pass-band of the fibre), and the relevant agent would recognise this as fibre damage.

Another possibility, yet to be investigated, may be to incorporate a small laser diode and photodiode detector into each agent, to enable the agent to carry out a direct interrogation of the integrity of the fibre segments within its cell. This would come at some cost in additional hardware and power demand across the structure, but would provide the agents with direct information about their sensing integrity. The addition of a basic capability on the agents to inject light into the network and detect optical signals from it would allow the fibre network to be used for communications as well as sensing, and thus avoid the need for a parallel communications network.

The method of dealing with damage in the network depends on where in the network it has occurred. If the segment of fibre that connects one cell to another is damaged, it is managed using the tracking field values and contour map. In this situation cells that have a damaged connection ignore each other when finding the minimum of their neighbours tracking field values. The damaged connections result in a discontinuity in the gradient across the cells, as can be seen in Fig. 4.12. In the simulation the damaged connecting links are represented with short black lines between switches on the adjoining cells.

Damage in the sensing segments of the fibres within a cell is dealt with by the agent on that cell, and the strategy depends on the extent of the damage. If a single segment of fibre is damaged, the agent can simply route the light around its subnetwork in the other direction. If more segments are damaged the solution is more complicated and will not be elaborated here.

An issue that is still to be resolved is ensuring that the optical path in the vicinity of the impact site provides sufficient sensor data to enable any resultant thermal

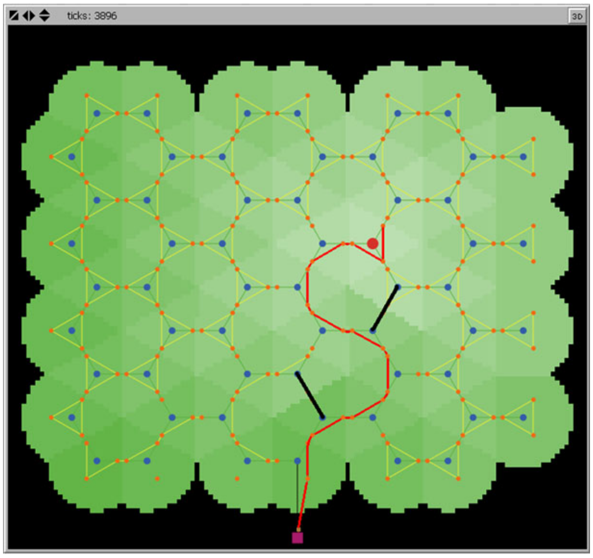


Fig. 4.12 Distance contour map for an array of triangular cells, and the optimal path from a damaged cell, indicated by the *large grey circle*, to the FOID (*grey square at bottom*), in the case of damage to several interconnect fibre segments. The damaged fibre segments are indicated by *thick solid black lines* joining optical switches (*small light grey circles*). The optical path established is shown

anomaly to be detected. This may require the agent at the impact site to select which fibre segments are interrogated, possibly including some on neighbouring cells if an impact occurs near the edge of a cell. As the thermal conductivity in the plane of the TPS materials has not yet been measured—it is assumed to be anisotropic as are the elastic properties, but the relative anisotropy is unknown—the lateral extent of the thermal anomaly associated with point impact damage is not yet known.

4.5.7 Fabrication and Testing

As stated in Sect. 4.5.1 above, fabrication of a demonstrator to establish the feasibility of the techniques outlined above is not yet complete, but solutions have been developed and tested for most of the significant design, operational and fabrication issues. System fabrication is in progress.

An important issue has been establishing the feasibility of the proposed switched optical fibre network. As is evident from the network schematics of Figs. 4.7 and 4.8, the network requires electronically-controlled 3-way switches, capable of connecting any of the three input/output optical ports to either of the other two to provide a bidirectional optical path with low attenuation. Suitable MEMS switches were sourced from DiCon Fiberoptics, Inc (CA, USA) but these are not yet an off-the-shelf item. For the purpose of the demonstrator, the three switches associated with each agent/cell are mounted on the agent circuit board. The sensing fibres have a 5 μm silica core, with 1542 nm wavelength Bragg gratings 5 mm long written with 10 mm spacing. They were obtained from Luna Technologies Inc. (VA, USA).

Feasibility testing has been carried out to ensure that the OFDR technique is capable of providing accurate measurements in a switched network, which will have points of attenuation at the switches and fibre joins. The tests, described by Hoschke et al. (2012, 2013), showed that the method provided accurate FBG measurements in the presence of attenuation in the network, limited only by the signal-to-noise ratio in the measurement instrument. For the FOID instrument used for the tests, this limit was reached when the added attenuation from switches, fibre connections, etc. was ∼7.5 dB, but it is expected that this can be increased substantially by increasing the laser power and reducing system noise. The switches each contribute ∼0.24 dB attenuation, but fibre splices and standard FC/APC optical fibre connectors also contribute significantly.

The fibre networks have been laid out on two tiles, and the two boards that sandwich the fibres have been bonded using the alumina cement Zircar AL-CEM, as shown in Fig. 4.9. Preliminary tests of impact detection, using a short (∼6.3 ns) 250 mJ pulse from a Q-switched Nd:YAG laser to simulate a fast particle impact, have been carried out.

4.5.8 Summary and Future Directions

The work completed so far on this application has shown how to extend the system described in the earlier sections in two significant ways. Firstly, it has incorporated a second embedded sensing modality and demonstrated its use in a simple sequential two-stage sensing strategy. Secondly, it has developed a technique to enable an inherently distributed sensing modality to be accommodated within a local sensing architecture. The control aspect of the switched optical fibre network is closely analogous to the control of the mobile robot described in Sects. 4.2 and 4.3 above, but the robot was envisaged to carry local sensors whose measured data could be transmitted directly back to the local agent concerned. The difference with the FBG network is that data will be obtained for all fibre segments along the selected path, and this can be communicated to all the agents concerned. This can be very useful if the thermal effects of the impact damage extend over more than the area of a single tile. Furthermore, it may be noted that the mobile robot could be incorporated into the present TPS monitoring system to apply a source of heat (a point source or a broad-area lamp) and/or to carry other sensors to probe the impact damage, such as a video camera as envisaged earlier.

Future directions for this application are initially to complete development of the damage diagnosis algorithm and fabrication of the demonstrator hardware. Attractive possibilities to enhance the sensing capability of the system are to use the mobile robot as described above, and to use the FBG network to measure strain as well as temperature (see, e.g. Richards et al. 2012) as this could be utilised to evaluate the effect of damage on the structural integrity of the TPS, an issue that has not been addressed so far. Application to other current or prospective TPS materials, such as the ablative and newer structurally integrated materials (Venkatapathy et al. 2010) is desirable, though this would be more of an adaptation than a new direction.

4.6 Conclusions

This Chapter has described the development of the first stages of an experimental structural health monitoring system whose operation is based on self-organization in a complex multi-agent system. Damage identification, location and the first stage of evaluation have been demonstrated, as has the deployment of a secondary robotic inspector. This is all achieved without central control.

A key feature has been the development of a mobile robotic agent, and the hardware and software modifications and developments required to enable the fixed and mobile agents to operate as a single, self-organizing, multi-agent system. This single-robot system is seen as the forerunner of a system in which larger numbers of small robots perform inspection and repair tasks cooperatively, by self-organization.

Recent work, outlined in Sect. 4.5, has been directed towards application of the principles of the original demonstrator to a realistic damage scenario: health monitoring of the thermal protection systems of spacecraft that re-enter the Earth's atmosphere. This work has involved extension of the system to two sensing modalities, and the development of a novel approach for incorporating a distributed optical fibre sensing system into the local agent architecture, which results in a much more robust fibre network.

The eventual goal of demonstrating self-organized damage diagnosis using information from multiple sensors has not yet been achieved, but this recent work is an important step in that direction. Concurrent work on corrosion monitoring at "hot spots" in aircraft is in progress (Trego et al. 2005; Muster et al. 2005; Cole et al. 2009), and this utilizes multi-sensor data to form data-driven damage models that are used for prognostic purposes.

While the present demonstration systems are clearly not suitable for large-scale implementation in a current aerospace vehicle, it is envisaged that the sensing, computation and self-repair functions of the embedded system will eventually be integrated into advanced materials. Recent advances in materials science and nanotechnology give confidence that this will be achieved in the foreseeable future, as will the development of micro-sized, intelligent robots. We believe that the basic approach outlined in this Chapter, of developing self-organizing, adaptive solutions in distributed multi-agent systems, will form the basis of future developments in this area.

It is our view that structural health monitoring is an interesting and fertile application area in which to study engineered self-organization. The wide range of spatial and temporal scales on which events can occur and damage develop, and the consequent variety of responses and response requirements, ensure that this general application will provide a more complete challenge for self-organized sensing and response than many others.

Acknowledgements It is a pleasure to acknowledge the continued support for this work of Drs. Ed Generazio and Bill Prosser of NASA Langley Research Center, and of Dr. Lance Richards of NASA Dryden Flight Research Center. We also gratefully acknowledge the contributions of Adam Batten, Graeme Edwards, Tony Farmer, Peter Isaacs and Chris Lewis (all from CSIRO Materials Science and Engineering), as well as Mikhail Prokopenko and Peter Wang (CSIRO ICT Centre) to this work.

References

Alers, G. A., & Zimmerman, R. M. (1980). Ultrasonic characterization of the thermal protection tiles for the space shuttle. In B. R. McAvoy (Ed.), *1980 ultrasonics symposium proceedings* (pp. 894–896). New York: IEEE Press.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., & Bonabeau, E. (2001). *Self-organization in biological systems*. Princeton: Princeton University Press.

Cole, I. S., Corrigan, P. A., Edwards, G. C., Ganther, W., Muster, T. H., Patterson, D., Price, D. C., Scott, D. A., Followell, D., Galea, S., & Hinton, B. (2009). A sensor-based learning approach to prognostics in intelligent vehicle health monitoring. *Materials Forum*, *33*, 27–35 [in Proceedings of the 2nd Asia-Pacific workshop on structural health monitoring (2APWSHM), Melbourne, December 2008].

Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In *Proc. 1999 congress on evolutionary computation*, Washington DC, July 1999 (pp. 1470–1477).

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge: MIT Press.

Froggatt, M. (1996). Distributed measurement of the complex modulation of a photoinduced Bragg grating in an optical fiber. *Applied Optics*, *35*, 5162–5164.

Froggatt, M., & Moore, J. (1998). Distributed measurement of static strain in an optical fiber with multiple Bragg gratings at nominally equal wavelengths. *Applied Optics*, *37*, 1741–1746.

Hedley, M., Johnson, M. E., Lewis, C. J., Carpenter, D. A., Lovatt, H., & Price, D. C. (2003). Smart sensor network for space vehicle monitoring. In *Proceedings of the international signal processing conference*, Dallas, Texas, March 2003. http://www.gspx.com/GSPX/papers_online/papers_list.php.

Hoschke, N., & Price, D. C. (2012). *Monitoring of thermal protection systems using robust self-organizing optical fibre sensing networks* (Report 3: Completion of Design Study). CSIRO Materials Science & Engineering.

Hoschke, N., Lewis, C. J., Price, D. C., Scott, D. A., Edwards, G. C., & Batten, A. (2006). A self-organising sensing system for structural health management. In B. Gabrys, R. J. Howlett, & L. C. Jain (Eds.), *Lecture notes in artificial intelligence: Vol. 4253. Proceedings of 10th international conference on knowledge-based intelligent information and engineering systems, Part III, KES 2006*, Bournemouth, UK, 9–11 October 2006 (pp. 349–357). Berlin: Springer.

Hoschke, N., Lewis, C. J., Price, D. C., Scott, D. A., Gerasimov, V., & Wang, P. (2008). A self-organizing sensing system for structural health monitoring of aerospace vehicles. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (1st ed.). London: Springer.

Hoschke, N., Price, D. C., Wood, A., & Walker, D. (2012). Fibre Bragg grating networks for robust sensing systems. In *Proceedings of the 37th Australian conference on optical fibre technology (ACOFT 2012)*, Sydney, December 2012.

Hoschke, N., Price, D. C., Scott, D. A., & Richards, W. L. (2013, to be published). Structural health monitoring of space vehicle thermal protection systems. *Key Engineering Materials* [in Proceedings of the 4th Asia-Pacific Workshop on Structural Health Monitoring (2APWSHM), Melbourne, December 2012].

Kohonen, T. (2001). *Self-organizing maps* (3rd ed.). Berlin: Springer.

Kohonen, T. (2003). Self-organized maps of sensory events. *Philosophical Transactions of the Royal Society of London, Series A: Mathematical and Physical Sciences*, *361*, 1177–1186.

Muster, T., Cole, I., Ganther, W., Paterson, D., Corrigan, P., & Price, D. (2005). Establishing a physical basis for the in-situ monitoring of airframe corrosion using intelligent sensor networks. In *Proceedings of the 2005 tri-service corrosion conference*, Florida, USA, November 2005.

Nye, J. F. (1985). *Physical properties of crystals*. London: Oxford University Press.

Price, D. C., Batten, A., Edwards, G. C., Farmer, A. J. D., Gerasimov, V., Hedley, M., Hoschke, N., Johnson, M. E., Lewis, C. J., Murdoch, A., Prokopenko, M., Scott, D. A., Valencia, P., & Wang, P. (2004). Detection, evaluation and diagnosis of impact damage in a complex multi-agent structural health management system. In *Proceedings of the 2nd Australasian workshop on structural health monitoring*, Melbourne, Australia, December 2004 (pp. 16–27).

Prokopenko, M., Wang, P., Foreman, M., Valencia, P., Price, D., & Poulton, G. (2005a). On connectivity of reconfigurable impact networks in ageless aerospace vehicles. *Robotics and Autonomous Systems*, *53*, 36–58.

Prokopenko, M., Wang, P., Scott, D. A., Gerasimov, V., Hoschke, N., & Price, D. C. (2005b). On self-organising diagnostics in impact sensing networks. In R. Khosla, R. J. Howlett, & L. C. Jain (Eds.), *Lecture notes in computer science: Vol. 3684*. *Proceedings of 9th international conference on knowledge-based intelligent information and engineering systems, Part IV, KES 2005*, Melbourne, Australia, 14–16 September 2005 (pp. 170–178). Berlin: Springer.

Prokopenko, M., Poulton, G., Price, D. C., Wang, P., Valencia, P., Hoschke, N., Farmer, A. J. D., Hedley, M., Lewis, C., & Scott, D. A. (2006). Self-organising impact sensing networks in robust aerospace vehicles. In J. Fulcher (Ed.), *Advances in applied artificial intelligence* (pp. 186–233). Hershey: Idea Group.

Prokopenko, M., Boschetti, F., & Ryan, A. J. (2008). An information-theoretic primer on complexity, self-organisation and emergence. *Complexity*, *15*, 11–28.

Richards, W. L., Parker, A. R., Ko, W. L., Piazza, A., & Chan, P. (2012) *Flight test instrumentation series: Vol. 22*. *Application of fiber optic instrumentation, NATO RTO AGARDograph 160*. http://www.cso.nato.int/Pubs/rdp.asp?RDP=RTO-AG-160-V22.

Rose, J. L. (1999). *Ultrasonic waves in solid media*. Cambridge: Cambridge University Press.

Scott, D. A., & Price, D. C. (2007). *Health monitoring of thermal protection systems. Report 1: preliminary measurements and design specifications* (NASA Contractor Report NASA/CR-2007-215092). NASA, Washington DC, USA.

Scott, D. A., Batten, A., Edwards, G. C., Farmer, A. J., Hedley, M., Hoschke, N., Isaacs, P., Johnson, M., Murdoch, A., Lewis, C., Price, D. C., Prokopenko, M., Valencia, P., & Wang, P. (2005). An intelligent sensor system for detection and evaluation of particle impact damage. In D. E. Chimenti (Ed.), *Review of progress in quantitative nondestructive evaluation* (Vol. 24, pp. 1825–1832). New York: AIP

Scott, D. A., Price, D. C., Hoschke, N., & Richards, W. L. (2009). Structural health monitoring of thermal protection systems. *Materials Forum*, *33*, 457–464 [in Proceedings of the 2nd Asia-Pacific workshop on structural health monitoring (2APWSHM), Melbourne, December 2008].

Trego, A., Price, D., Hedley, M., Corrigan, P., Cole, I., & Muster, T. (2005). Development of a system for corrosion diagnostics and prognostics. In *Proceedings of 1st World congress on corrosion in the military: cost reduction strategies*, Sorrento, Italy, June 2005.

Venkatapathy, E., Szalai, C. E., Laub, V., Hwang, H. H., Conley, J. L., & Arnold, J. (2010). *Thermal protection system technologies for enabling future sample return missions* (White paper submitted to the NRC Planetary Science Decadal Survey, Primitive Bodies Sub-panel). http://www.lpi.usra.edu/decadal/sbag/topical_wp/EthirajVenkatapathy.pdf.

Zircar (2012). Information in http://www.zircarceramics.com/pages/rigidmaterials/specs/zal15.htm. http://www.zircarceramics.com/pages/cem-rig/specs/al-cem.htm.

Chapter 5
Decentralised Decision Making for Ad-hoc Multi-Agent Systems

George Mathews and Hugh Durrant-Whyte

5.1 Introduction

Decision making in large distributed multi-agent systems is a fundamental problem with a wide range of applications including distributed environmental monitoring, area search and surveillance, and coordination of transportation systems. In general, for an agent to make an good decision, it must consider the decisions of all the other agents in the system. This coupling between decision makers has two main causes: (i) the agents share a common objective function (e.g. they operate as a team), or (ii) the agents have individual goals but share constraints (e.g. they must cooperate in sharing a finite resource). This chapter is focused on the first issue, and assumes the agents are designed to operate as a team.

The classical approach to solve this type of decision or planning problem is to collect all the information from the agents in a single location and solve the resulting optimisation problem (e.g. see Furukawa et al. 2003). However, this centralised approach has two main problems:

- The required communication bandwidth grows at least linearly with the number of agents.
- The resulting optimisation complexity is in general exponential in the number of agents.

Thus, for a sufficiently large number of agents this problem becomes impractical to solve in a centralised fashion.

However for physically distributed systems, sparsity in the interdependencies between the agents should be exploited and the agents allowed to cooperate or self-

G. Mathews (✉) · H. Durrant-Whyte
National ICT Australia (NICTA), Australian Technology Park, 13 Garden St, Eveleigh, NSW 2015, Australia
e-mail: George.Mathews@nicta.com.au

H. Durrant-Whyte
e-mail: h.durrant-whyte@cas.edu.au

organise in solving the distributed decision problem. The main issue with this decentralised approach now becomes identifying what local processing is required, which agents need to communicate, what information should be sent and how frequently.

This chapter approaches the multi-agent collaboration problem using distributed optimisation techniques and presents results on the structure of the decision problem and how to exploit sparseness. Although distributed optimisation methods have been studied for over two decades (Baudet 1978; Bertsekas and Tsitsiklis 1989, 1991; Tsitsiklis et al. 1986; Tseng 1991; Patriksson 1997; Camponogara and Talukdar 2007), the existing results and algorithms generally require significant prior configuration that defines the structure of the local interactions and are generally not suitable ad-hoc multi-agent systems.

This chapter defines a simple and intuitive decentralised optimisation algorithm which enables multiple decision makers to propose and refine decision to optimise a given team objective function. A subsequent convergence analysis of this procedure provides an intuitive relationship between the communication frequency, transmission delays and the inherent inter-agent coupling in the system. From this, an approximate algorithm is defined that can be easily implementable in an ad-hoc team without the need to define the coupling information up front. Finally, a generalisation of the objective function is introduced that allows the specific capabilities and individual models of the agents to be abstracted away.

The algorithm is applied to the control of multiple mobile robots undertaking an information gathering task. The specific scenario considered requires the robotic agents to actively localise a group of objects. For this scenario the inter-agent coupling loosely relates to the amount of overlap between the information two agents will receive when undertaking their respective plans. This requires communications only between coupled agents and results in a scalable system with sparse interactions.

Section 5.2 defines the multi-agent decision problem and introduces a general distributed optimisation method to solve it. Section 5.3 defines the convergence conditions and introduces an inter-agent coupling metric, and links it to the structure of inter-agent communications. Section 5.4 extends the optimisation algorithm to ah-hoc systems by defining an on-line approximation method for the inter-agent coupling. The resulting algorithm is applied to a simple example problem in Sect. 5.5. Section 5.6 extends the algorithm further by considering a decomposition of the objective function that explicitly specifies what local information is required about the capabilities and models of other agents. Section 5.7 describes the general multi-agent information gathering problem and formulates it as a distributed sequential decision problem. This is specialized for an object localisation problem in Sect. 5.7 with results given in Sect. 5.8. Section 5.9 provides a summary and directions for future work.

5.2 Distributed Optimisation

Distributed algorithms can be categorised into synchronous and asynchronous (Bertsekas and Tsitsiklis 1991) methods. Synchronous algorithms require iterations to be executed in a predetermined order and can be broken down into Gauss-Seidel and Jacobi types (see Fig. 5.1). The iterations in a Gauss-Seidel algorithm must be computed sequentially, possibly causing the agents to be idle while waiting for their turn. The underlying optimisation algorithm presented in Inalhan et al. (2002), and used for collision avoidance, is an example of this type.

A Jacobi type algorithm allows the agents to work on subproblems in parallel, but still requires each agent to wait until information from all other agents has been received prior to starting a new iteration. The formation control algorithm presented in Raffard et al. (2004) falls in this category.

An asynchronous algorithm is similar to a Jacobi type, but does not require each agent to wait before starting the next iteration, an agent simply uses all the information it has available at the time. This type of algorithm allows each agent to compute and communicate at different rates without the overall progress being limited by the slowest agent.

Asynchronous algorithms can be further separated into totally asynchronous and partially asynchronous (Bertsekas and Tsitsiklis 1991). For totally asynchronous algorithms, the age of the information an agent has about another can become arbitrary large. While a partially asynchronous algorithm requires this to be bounded. The ADOPT algorithm (Modi et al. 2005) for distributed constraint optimization problems with discrete decisions is an example of a totally asynchronous algorithm.

The remainder of this section defines the general team decision problem that will be considered in this paper and introduces the proposed asynchronous solution method.

5.2.1 Problem Formulation

Consider a team of p agents, where each agent $i \in \{1, \dots, p\}$ is in charge of a local decision variable $\mathbf{v}_i$ that is constrained to be an element of the feasible local decision set $\mathcal{V}_i \subseteq \Re^{d_i}$. The team decision vector[1] is given by collection of decisions from each of the agents

$$\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_p]$$

and is defined on the product set $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_p \subseteq \Re^d$, where $d = d_1 + \cdots + d_p$. In regards to notation, normal italics is used for scalars, bold type for vectors and matrices, and calligraphic for sets.

Assumption 1 (Decisions) The set of feasible decisions $\mathcal{V}_i$ is convex for all agents $i \in \{1, \dots, p\}$.

[1]All vectors are assumed to be column vectors. For simplicity the 'transpose' has been omitted. Formally, $\mathbf{v}$ should be defined as $[\mathbf{v}_1^T, \dots, \mathbf{v}_p^T]^T$.

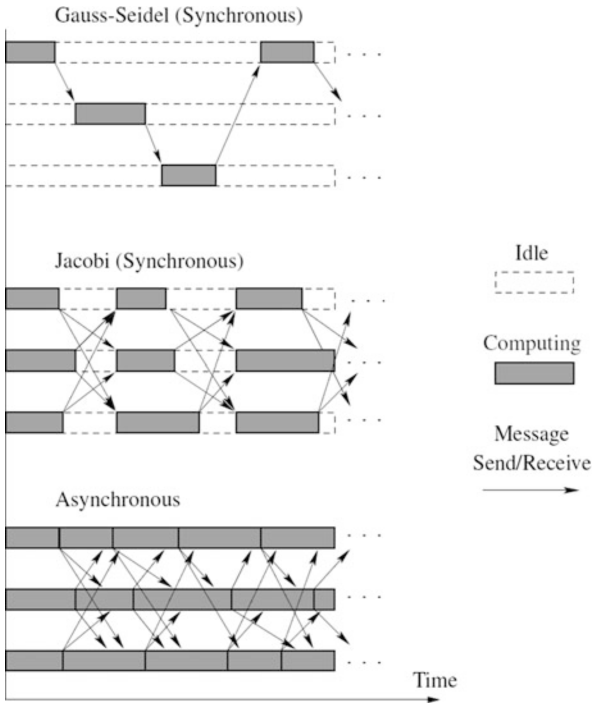


Fig. 5.1 Synchronous vs asynchronous iterations for a system of three agents

As the joint decision set is defined as $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_p$, Assumption 1 also guarantees $\mathcal{V}$ is convex.

The team decision problem of interest requires the agents to jointly select decisions such that a given team objective function $J : \mathcal{V} \to \Re$ is minimised

$$\mathbf{v}^* = \arg\min_{\mathbf{v}\in\mathcal{V}} J(\mathbf{v}), \tag{5.1}$$

where $\mathbf{v}^*$ is the desired optimal joint decision. It is noted that this definition of the decision problem, incorporates local constraints on the individual decisions of the agents but excludes hard inter-agent constraints. Coupling between the agents only through the team objective function J.

Assumption 2a (Objective Function) The objective function J has a minimum, is continuous and twice differentiable.

Assumption 2b (Convexity) The objective function J obeys Assumption 2a and is also convex.

Under the convexity conditions of Assumptions 1 and 2b, a necessary and sufficient condition (Bertsekas 1999) for $\mathbf{v}^*$ to be optimal is

$$\left(\boldsymbol{\nu} - \mathbf{v}^*\right)^T \nabla J\left(\mathbf{v}^*\right) \geq 0, \quad \text{for all } \boldsymbol{\nu} \in \mathcal{V}, \tag{5.2}$$

where $\nabla J(\mathbf{v}) \in \Re^d$ is the gradient vector of $J(\mathbf{v})$. In terms of each agents local decision, this can be rewritten as

$$\left(\boldsymbol{\nu}_i - \mathbf{v}_i^*\right)^T \nabla_i J\left(\mathbf{v}^*\right) \geq 0, \quad \text{for all } \boldsymbol{\nu}_i \in \mathcal{V}_i, \tag{5.3}$$

for all i, where $\mathbf{v}_i^*$ is the ith agent's component of the optimal decision $\mathbf{v}^*$ and $\nabla_i J(\mathbf{v}^*) \in \Re^{d_i}$ is the gradient vector with respect to the ith agent's decision, evaluated at $\mathbf{v}^*$.

It is noted that if only Assumption 2a holds (i.e. the objective function is not convex), (5.3) corresponds to the first order necessary condition for the point $\mathbf{v}^*$ to be a local minimum (Bertsekas 1999).

5.2.2 Asynchronous Optimisation Model

The asynchronous solution method pursued in this chapter is based on Tsitsiklis et al. (1986) and allows each agent to propose an initial decision and then to incrementally refine it, while intermittently communicating these refinements to the team. Under the condition of asynchronous execution and communication, it is initially required that each agent maintains a local copy of the team decision vector, containing the (possibly out of date) decision of each agent. The copy maintained by agent i at discrete time $t \in \{0, 1, 2, \dots\}$ is denoted by

$$^i\mathbf{v}(t) = \left[{}^i\mathbf{v}_1(t), {}^i\mathbf{v}_2(t), \dots, {}^i\mathbf{v}_p(t)\right]. \tag{5.4}$$

Here, pre-superscripts represents a copy held by a specific agent, while subscripts represent a specific agents decision, e.g. ${}^i\mathbf{v}_j(t)$ represents agent i's local copy of agent j's decision. Here, the variable t is used to represent when discrete events take place, such as when an agent computes an update or communicates.

To represent the effects of communication delays and asynchronous execution, the variable $\tau_{ji}(t)$ will be used to denote when agent j's local copy ${}^j\mathbf{v}_i(t)$ was generated by agent i and hence ${}^j\mathbf{v}_i(t) = {}^i\mathbf{v}_i(\tau_{ji}(t))$. It is assumed that $\tau_{ii}(t) = t$ and thus agent i always has the latest copy of its own decision. Using this notation, (5.4) can be rewritten as

$$^i\mathbf{v}(t) = \left[{}^1\mathbf{v}_1\left(\tau_{i1}(t)\right), \dots, {}^p\mathbf{v}_p\left(\tau_{ip}(t)\right)\right]. \tag{5.5}$$

5.2.2.1 Local Decision Update

To formalise the notion of *decision refinement* a local update rule $f_i : \mathcal{V} \to \mathcal{V}_i$ is defined for each agent i that modifies its local decision ${}^i\mathbf{v}_i$, based on its copy of the team decision vector ${}^i\mathbf{v}$. To allow each agent to perform updates asynchronously a set of times $T_i^U \subseteq \{0, 1, 2, \dots\}$ is associated with agent i that represents when the agent computes a local update

$$ {}^i\mathbf{v}_i(t+1) = \begin{cases} f_i({}^i\mathbf{v}(t)) & \text{if } t \in T_i^U, \\ {}^i\mathbf{v}_i(t) & \text{else.} \end{cases} \tag{5.6}$$

To explicitly define the capabilities of each agent, and ensure the complexity of the local decision refinement is only dependent on the size of the local decision space, the following restriction is made on what basic operations an agent can perform on the objective function.

Assumption 3a (Local Operations) Each agent can only compute the gradient of the objective function with respect to its own decision vector $\mathbf{v}_i$, $\nabla_i J(\mathbf{v}) \in \Re^{d_i}$.

An extension, also considered in this work, is defined in the following assumption.

Assumption 3b (Local Operations, ext.) In addition to Assumption 3a, each agent can compute the second order derivatives of the objective function with respect to its own decision vector $\mathbf{v}_i$, corresponding to $\nabla_{ii}^2 J(\mathbf{v})$ the $d_i \times d_i$ submatrix of the full Hessian.

Based on these constraints it is proposed to use a scaled gradient projection update method. Thus, for an agent i to update its decision, it first determines the local gradient vector $\nabla_i J({}^i\mathbf{v}(t))$ and applies a positive definite scaling matrix $\mathbf{A}_i(t)$ (e.g. the local Hessian $\nabla_{ii}^2 J({}^i\mathbf{v}(t))$ or simply the identity matrix $\mathbf{I}$), producing an update direction $\mathbf{d}_i = [\mathbf{A}_i(t)]^{-1}\nabla_i J({}^i\mathbf{v}(t))$. The decision is then updated by moving it in this direction and, if needed, projecting it back onto the feasible decision set

$$ f_i\big({}^i\mathbf{v}(t)\big) = \mathrm{Proj}_{\mathcal{V}_i}^{\mathbf{A}_i(t)}\big({}^i\mathbf{v}_i(t) - \gamma_i\big[\mathbf{A}_i(t)\big]^{-1}\nabla_i J\big({}^i\mathbf{v}(t)\big)\big), \tag{5.7}$$

where γ_i is a step size to be defined and $\mathrm{Proj}_{\mathcal{V}_i}^{\mathbf{A}_i}(\cdot)$ denotes the projection onto the set $\mathcal{V}_i$ under the scaling of $\mathbf{A}_i$. This projection is defined for any vector $\boldsymbol{\mu}_i \in \Re^{d_i}$ as

$$ \mathrm{Proj}_{\mathcal{V}_i}^{\mathbf{A}_i}(\boldsymbol{\mu}_i) = \arg\min_{\boldsymbol{\nu}_i \in \mathcal{V}_i} (\boldsymbol{\nu}_i - \boldsymbol{\mu}_i)^T \mathbf{A}_i (\boldsymbol{\nu}_i - \boldsymbol{\mu}_i). \tag{5.8}$$

The projection operation requires only local knowledge of the feasible set $\mathcal{V}_i$ and the scaling matrix $\mathbf{A}_i(t)$.

There is a slight, but very important, difference between this formulation of the update rule and those presented in Bertsekas and Tsitsiklis (1989), Tsitsiklis et al. (1986), Tseng (1991), and Patriksson (1997). The difference lies in the assignment of a possibly different step size γ_i to each agent and will have significant practical implications.

To ensure that the interval between updates computed by each agent is bounded and thus, as t tends to ∞ the number of updates computed by each agent also tends to ∞ the following assumption is required.

Assumption 4 (Continuous Computation) There exists a finite positive constant Q, such that for all t and i

$$\exists q \in T_i^U, \quad \text{such that } q \in [t, t+Q]. \tag{5.9}$$

5.2.2.2 Communication

Inter-agent communication is modelled in a high level fashion and requires all agents be able to send and receive messages from all other agents. Although the physical communication network is not likely to be fully connected, this high-level model allows the specific routing protocol and topology to be ignored, while retaining the important effects of asynchronous execution and message delivery delays. In this model, communications are initiated by an agent i sending a message at some time $t \in T_{ij}^C \subseteq \{0, 1, 2, \dots\}$ to another agent j containing its latest decision ${}^i\mathbf{v}_i(t)$. After some communication delay $b_{ij}(t)$ agent j receives it and incorporates it into its local copy of the team decision vector. Thus, when the message is received ${}^j\mathbf{v}_i(t + b_{ij}(t)) = {}^i\mathbf{v}_i(t)$ and hence $\tau_{ji}(t + b_{ij}(t)) = t$.

There are two basic assumptions that can be made about the sets of communication times $\{T_{ij}^C : \forall i, j \neq i\}$ and delivery delays $\{b_{ij}(t) \colon \forall l, i, j \neq i\}$. These will be made indirectly by simply considering the age of the information one agent has about another, represented by the variables $\tau_{ji}(t)$.

Assumption 5a (Arbitrary Delays) Given any time t_1, there exists a time $t_2 > t_1$ such that for all i and j

$$\tau_{ji}(t) > t_1, \quad \text{for all } t > t_2. \tag{5.10}$$

Thus, the delays in the system can become arbitrarily large, while enforcing that after a sufficiently long time (t_2) all old information (from before t_1) is removed from the system.

Assumption 5b (Bounded Delays) There exists finite positive constants B_{ij} for all i and j such that

$$t - \tau_{ji}(t) \leq B_{ij}, \quad \text{for all } t. \tag{5.11}$$

Informally, this can be relaxed so that B_{ij} represents the maximum difference, measured in numbers of updates computed by agent i, between ${}^i\mathbf{v}_i(t)$ and ${}^j\mathbf{v}_i(t)$.

It is noted that Assumption 5b implies Assumption 5a. The difference between these two assumptions differentiate totally and partially asynchronous algorithms (Bertsekas and Tsitsiklis 1989).

It is noted that the delay bounds in Assumption 5b can be determined by knowing: (i) the maximum rate iterations are computed by agent i, denoted by R_i; (ii) the minimum rate messages are communicated from i to j, denoted by C_{ij}; and (iii) the maximum delivery delay between agent i sending and j receiving a message, denoted by D_{ij}. This results in the relation

$$B_{ij} = \frac{R_i}{C_{ij}} + R_i D_{ij}. \tag{5.12}$$

5.3 Convergence Results

The previous section defined a general gradient-based optimisation method that allows each agent to incrementally refine its local decision, while intermittently communicating with the rest of the team. What is of interest now is if such an approach will converge to an optimal solution of the original problem defined in (5.1). This section will present two sufficient conditions under which the algorithm method will converge to a team decision obeying the optimality condition (5.3). The first requires a restrictive assumption on the structure of the problem, but allows very simple algorithms to be create. While the second can be applied to any problem, but leads to slightly more complex algorithms.

5.3.1 Weak Coupling Convergence

The first condition requires the system to be *weakly coupled*, which will be defined below. For simplicity the following restrictions will be made in this section: (i) decisions are scalar ($\mathcal{V}_i \subseteq \Re$ for all i); (ii) unity scaling is employed ($\mathbf{A}_i = 1$ for all i); and (iii) all agents share a common step size ($\gamma_i = \gamma$ for all i).

Assumption 6 (Weak Coupling) The Hessian of the objective function is diagonally dominate over all feasible decisions. That is, for all i and all $\mathbf{v} \in \mathcal{V}$

$$\nabla^2_{ii} J(\mathbf{v}) > \sum_{j \neq i} \left| \nabla^2_{ij} J(\mathbf{v}) \right|, \tag{5.13}$$

where the second order derivatives are scalars, i.e. $\nabla^2_{ii} J(\mathbf{v})$, $\nabla^2_{ij} J(\mathbf{v}) \in \Re$, and $|\cdot|$ denotes the absolute value.

Informally, this condition requires the gradient of the objective function with respect to the ith agent, $\nabla_i J(\mathbf{v})$, to have a greater dependence on the local decision $\mathbf{v}_i$ than the sum of all dependencies from other agents. It is noted that this is satisfied by all 2 dimensional strictly convex functions but does not necessarily hold for higher dimensional convex functions ($p > 2$).

Under this Assumption, convergence can be demonstrated using a contraction mapping approach that ensures, after successive iterations, the team decision moves closer to the optimum.

Theorem 1 (Weak Coupling Convergence (Bertsekas and Tsitsiklis 1989)) *The Assumptions 1, 2b, 4, 5a and 6 provide sufficient conditions for the asynchronous optimisation algorithm, defined by (5.6) and (5.7), to converge to the global minimum of J, provided the step size γ obeys*

$$0 < \gamma < \frac{2}{K}, \tag{5.14}$$

where K is related to the maximum curvature of J and is given by[2]

$$K = \max_{\mathbf{v}\in\mathcal{V}} \left\| \nabla^2 J(\mathbf{v}) \right\|. \tag{5.15}$$

Here, $\|\cdot\|$ represents the induced L_2 matrix norm and $\nabla^2 J(\mathbf{v})$ denotes the Hessian matrix of the objective function J.

This theorem can be used to guarantee convergence for a class of algorithms based on the generalisation of the Jacobi method for solving linear equations. Utilising Assumption 5a, if each agent i only communicates after they have minimised the objective function with respect to their local decision $\mathbf{v}_i$, the totally asynchronous version of the nonlinear Jacobi algorithm is obtained.

Corollary 1 (Nonlinear Jacobi Method (Bertsekas and Tsitsiklis 1989)) *Under the same assumptions of Theorem 1, the asynchronous nonlinear Jacobi algorithm, defined by (5.6) with the local update rule given by*

$$f_i\left({}^i\mathbf{v}(t)\right) = \arg\min_{\mathbf{v}_i\in\mathcal{V}_i} J\left({}^i\mathbf{v}_1(t), \ldots, {}^i\mathbf{v}_{i-1}(t), \mathbf{v}_i, {}^i\mathbf{v}_{i+1}(t), \ldots, {}^i\mathbf{v}_p(t)\right), \tag{5.16}$$

will also converge to the global minimum of J.

The form of the nonlinear Jacobi algorithm is very simple and it is unsurprising it has been applied to cooperative control problems in multi-robot systems (Grocholsky 2002; Bourgault et al. 2004). However, the only known sufficient condition guaranteeing convergence requires the system to be weakly coupled and possess a convex objective function, conditions which are often not satisfied. For this reason algorithms based on Theorem 1 will not be pursued further.

5.3.2 General Convergence

Due to the restrictions imposed by Assumptions 2b and 6, more general results have been developed (Tsitsiklis et al. 1986; Tseng 1991; Patriksson 1997), which

[2]In Bertsekas and Tsitsiklis (1989), K is defined using a Lipschitz continuity condition of ∇J. Here, a simpler, but less general, definition is used by considering the Hessian.

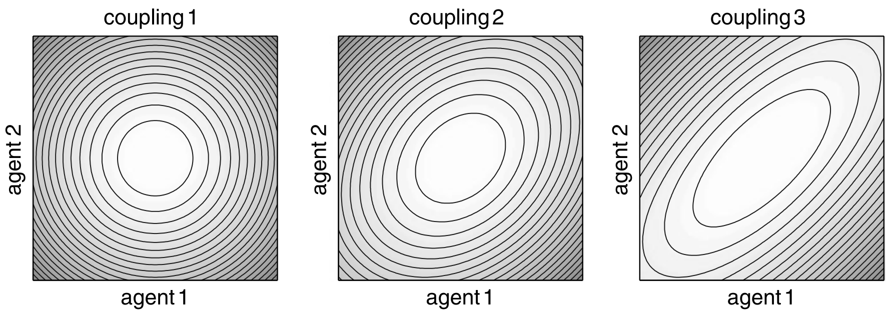


Fig. 5.2 Visualisation of the inter-agent coupling K_{ij} for a quadratic function with scalar decisions

are applicable to a wider range of problems. A key difference of these methods is they require the time between communications be bounded, utilising Assumption 5b instead of Assumption 5a.

These partially asynchronous methods are guaranteed to converge provided the step size is small enough. However, previous results are generally only concerned with the existence of a bound on the step size and not in the specific form of it. The condition developed here extends the results of Tsitsiklis et al. (1986) and presents a bound on the step size of each agent that only depends on the local coupling and communication properties of that agent. As will be shown, this provides significant insight into the structure of the problem and the requirements of which agents actually need to communicate.

To start, a scalar coupling metric is defined for each pair of agents that captures the maximum curvature of the objective function in the subspace of their decisions.

Assumption 7 (Coupling) For every i and j, there exists a finite positive constant K_{ij}, such that

$$K_{ij} = \max_{\mathbf{v} \in \mathcal{V}} \left\| \nabla_{ij}^2 J(\mathbf{v}) \right\|, \tag{5.17}$$

where again, $\|(\|\cdot)$ is the induced L_2 matrix norm and $\nabla_{ij}^2 J(\mathbf{v})$ corresponds to the $d_i \times d_j$ submatrix of the Hessian of the objective function.

The term K_{ii} will be referred to as the internal coupling of agent i, while for $j \neq i$ the term K_{ij} will be referred to as the inter-agent coupling of agents i and j. The inter-agent coupling represents the strength of the dependency of the gradient of one agent's decision on the decision of another and is depicted for a quadratic function in Fig. 5.2.

Theorem 2 (General Convergence) *The Assumptions* 1, 2b, 4, 5b *and* 7 *provide sufficient conditions for the asynchronous optimisation algorithm defined by* (5.6) *and* (5.7) *to converge to a limit point obeying the optimality criterion* (5.3), *provided*

$\gamma_i \in (0, \Gamma_i)$ *where*

$$\Gamma_i = \frac{2a_i}{K_{ii} + \sum_{j \neq i} K_{ij}(1 + B_{ij} + B_{ji})} \tag{5.18}$$

and a_i *is a scaling normalisation factor given by*

$$a_i = \min_t \underline{s}\big(\mathbf{A}_i(t)\big). \tag{5.19}$$

Here, $\underline{s}(\mathbf{M})$ *represents the minimum singular value of matrix* $\mathbf{M}$. *See the Appendix for a proof.*

This Theorem is applicable to any problem with any coupling structure, however, it becomes increasingly conservative for weakly coupled systems. In addition, under the weaker condition of Assumption 2a, any limit point of the algorithm will be guaranteed to satisfy the first order necessary condition (5.3), such as a local minimum.

Based on the definition of Γ_i in Theorem 2, an algorithm can be developed by defining the step size as

$$\gamma_i = \frac{\beta a_i}{K_{ii} + \sum_{j \neq i} K_{ij}(1 + B_{ij} + B_{ji})}, \tag{5.20}$$

for any $\beta \in (0, 2)$.

It is noted that the step size is inversely related to the product of the inter-agent coupling to an agent j (K_{ij}) and the associated maximum round trip communication delay ($B_{ij} + B_{ji}$). This provides a unified way of relating the inherent inter-agent coupling of the decision problem and communication structure of the team, to the rate at which each agent can refine its local decision.

5.3.3 *Efficient Communication Policies*

For the general problem under consideration, for each agent to receive the decisions of each other agent, there must exist a communication channel between all agents in the system. Regardless of the implementation, the only relevant features of this communication network are the inter-agent communication frequencies and transmission delays. It is important to capture the effects of the communication network on the step size γ_i as the magnitude of the step size used in gradient based algorithms have a critical impact on their overall convergence speed. This is known for typical centralised or synchronous algorithms (Bertsekas 1999), and asynchronous algorithms (Tseng 1991). Although a detailed analysis of the convergence rate is beyond the scope of this work.

For constant computation rates R_i, communication rates C_{ij} and transmission delays D_{ij}, the delay terms B_{ij} are given by (5.12), allowing (5.20) to be rewritten as the relation

$$\frac{\beta}{\gamma_i} = K_{ii} + \sum_{j \neq i} K_{ij} \left(1 + \frac{R_i}{C_{ij}} + R_i D_{ij} + \frac{R_j}{C_{ji}} + R_j D_{ji} \right). \tag{5.21}$$

Although both the communication frequencies and transmission delays both influence the step size, only the communication frequency is generally controllable by the agents (up to a limit), while the transmission delays are determined by the topology of the communication network, routing policies and the nature of the physical communication medium. Thus, the remainder of this section will focus on defining a communication policy that determines appropriate communication frequencies, based on the local coupling structure.

5.3.3.1 Communication Rate

Now, consider the computation rates (R_i and R_j) and inter-agent communication delays ($D_{j \to i}$ and $D_{i \to j}$) as fixed and uncontrollable, (5.21) can be rewritten as

$$\frac{\beta}{\gamma_i} = K_{ii} + \sum_{j \neq i} K_{ij} \left(W_{ij} + \frac{R_i}{C_{ij}} + \frac{R_j}{C_{ji}} \right) \tag{5.22}$$

where $W_{ij} = 1 + R_i D_{ij} + R_j D_{ji}$ and is now a fixed constant. Thus, the only terms that are controllable by the agents are the communication rates. From (5.22), the step size γ_i is maximised when all the communication rates are minimised, that is every agent communicates to every other agent after every local iteration.

However, this policy is impractical for large systems containing many agents. Potential this can be overcome by allowing each pair of agents to communicate at a rate proportional to the coupling, i.e.

$$C_{ij} = \eta_i K_{ij} \tag{5.23}$$

for some constant η_i. However, this will also be impractical for large systems since the step size will become directly related to the number of agents. This can be demonstrated by considering all agents to have a fixed computation rate $R_i = R$ and proportionality constant $\eta_i = \eta$, and substituting (5.23) into (5.22)

$$\frac{\beta}{\gamma_i} = K_{ii} + \sum_{j \neq i} K_{ij} W_{ij} + 2(p-1)\frac{R}{\eta}. \tag{5.24}$$

Thus, for large p, the last term will dominate causing the step size γ_i to approach 0 regardless of how small the actual inter-agent coupling may be. Thus a communication rate proportional to the coupling is in general too low.

To overcome this it is proposed to set the communication rate proportional to the square root of the coupling.

$$C_{ij} = \eta_i \sqrt{K_{ij}}. \tag{5.25}$$

This represents a compromise between fast convergence, requiring a high communication rate, and the practical requirements of a slow communication rate. The step size γ_i, and hence the possible convergence rate, is now only dependent on the coupling, which is in turn determined by the inherent complexity of the problem. This can be demonstrated by substituting (5.25) into (5.22)

$$\frac{\beta}{\gamma_i} = K_{ii} + \sum_{j \neq i} \big(K_{ij} W_{ij} + \sqrt{K_{ij}} (\eta_i R_i + \eta_j R_j) \big) \tag{5.26}$$

The constant η_i can be chosen such that the strongest coupled agent is sent a message after every local iteration, or to satisfy some bandwidth limitations. More complex considerations could be taken into account when choosing η_i, but this will not be considered here.

5.3.4 Algorithm

The overall optimisation algorithm built from Theorem 2 and the communications policy defined above, defines a set of local procedures for each agent and is outlined in Algorithm 5.1. A significant drawback of this algorithm is the requirement to have access to inter-agent coupling values K_{ij} prior to running the algorithm. This is a strong condition and will require some external mechanism that can calculate them. Such a situation will be unlikely to exist in ad-hoc teams and approximation methods will be introduced in the next section.

5.4 Ad-hoc Implementation and Coupling Estimation

To overcome the strong requirement of knowing the inter-agent coupling structure, this section defines an approximation method that allows each agent to estimate the coupling on-line using only locally available information.

Consider the Taylor expansion of J about a decision vector $\mathbf{v}$ with a perturbation $\Delta\mathbf{v} = [\Delta\mathbf{v}_1, \ldots, \Delta\mathbf{v}_p]$

$$J(\mathbf{v} + \Delta\mathbf{v}) \approx J(\mathbf{v}) + \sum_{i=1}^{p} \Delta\mathbf{v}_i^T \nabla_i J(\mathbf{v}) + \frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \Delta\mathbf{v}_i^T \nabla_{ij}^2 J(\mathbf{v}) \Delta\mathbf{v}_j. \tag{5.27}$$

The use of the coupling term K_{ij} gives a maximum bound on the value of the last term (see (5.17)). Thus, it is proposed to approximate the inter-agent coupling by estimating this term over successive iterations.

Algorithm 5.1: Local algorithm for agent i. This algorithm requires the inter-agent coupling to be calculated before execution

```
Input: $J, \{K_{ij}\}_{j=1}^{p}, \mathbf{v}^0, \eta_i, \beta$
Output: $\mathbf{v}^*$
 1: ${}^i\mathbf{v} \Leftarrow \mathbf{v}^0$  // Initialise local copy of team decision vector
 2: for all $j \neq i$ do
 3:     $C_{ij} = \eta_i \sqrt{K_{ij}}$                    // Comm. rate to agent $j$
 4:     Initialise communication link to $j$
 5:     Send (receive) computation rate $R_i$ ($R_j$)
 6:     Send (receive) communication rate $C_{ij}$ ($C_{ji}$)
 7:     Determine transmission delays $D_{ij}$ and $D_{ji}$
 8:     $B_{ij} = R_i/C_{ij} + R_i D_{ij}$     // Total delay in info. to agent $j$
 9:     $B_{ji} = R_j/C_{ji} + R_j D_{ji}$     // Total delay in info. from agent $j$
10: end for
11: $\gamma_i = \frac{\beta a_i}{K_{ii} + \sum_{j \neq i} K_{ij}(1 + B_{ij} + B_{ji})}$   // Calculate step size
12: repeat
13:     $\mathbf{A}_i \Leftarrow \nabla_{ii}^2 J({}^i\mathbf{v})$ or $\mathbf{I}$            // Generate scaling matrix
14:     $\mathbf{d}_i \Leftarrow -\mathbf{A}_i^{-1} \nabla_i J({}^i\mathbf{v})$         // Evaluate update direction
15:     ${}^i\mathbf{v}_i \Leftarrow \mathrm{Proj}_{\mathcal{V}_i}^{\mathbf{A}_i}({}^i\mathbf{v}_i + \gamma_i \mathbf{d}_i)$       // Update local plan
16:     for all $j \neq i$ do {Manage communications}
17:         if Req. to send msg to $j$ then {Determined from $C_{ij}$}
18:             Send $m_{ij} = {}^i\mathbf{v}_i$ to $j$
19:         end if
20:         if Msg $m_{ji} = {}^j\mathbf{v}_j$ received from $j$ then
21:             ${}^i\mathbf{v}_j \Leftarrow {}^j\mathbf{v}_j$                  // Update local copy
22:         end if
23:     end for
24: until Converged
25: return ${}^i\mathbf{v}$
```

If only perturbations in the decisions of agents i and j are considered, then the cross derivative term can be estimated using

$$\Delta\mathbf{v}_i^T \nabla_{ij}^2 J(\mathbf{v}) \Delta\mathbf{v}_j \approx \Delta\mathbf{v}_i^T \nabla_i J(\mathbf{v} + \Delta\mathbf{v}_j) - \Delta\mathbf{v}_i^T \nabla_i J(\mathbf{v}).$$

With some abuse of notation, the vector $\mathbf{v} + \Delta\mathbf{v}_j$ represents $\mathbf{v} + [0, \ldots, 0, \Delta\mathbf{v}_j, 0, \ldots, 0]$. The inter-agent coupling K_{ij} can now be estimated using

$$K_{ij} \approx \frac{1}{\|\Delta\mathbf{v}_j\|} \left| \frac{\Delta\mathbf{v}_i^T}{\|\Delta\mathbf{v}_i\|} \nabla_i J(\mathbf{v} + \Delta\mathbf{v}_j) - \frac{\Delta\mathbf{v}_i^T}{\|\Delta\mathbf{v}_i\|} \nabla_i J(\mathbf{v}) \right|,$$

where the absolute value is used to maintain a positive estimate. By defining $\mathbf{e}_i = \frac{\Delta\mathbf{v}_i}{\|\Delta\mathbf{v}_i\|}$, the above equation can be considered as the difference between the gradient of J in the direction $\mathbf{e}_i$ before and after a perturbation in agent j's decision.

By evaluating this at each iteration, and defining $\mathbf{e}_i$ and $\Delta\mathbf{v}_j$ to lie in the update directions of agents i and j respectively, the estimate will track the local curvature of the objective function in the vicinity of the actual team decision and in directions where the decisions are actively being refined. This is in contrast to using an absolute maximum over all positions and directions, as suggested in (5.17), and will result in a more appropriate value.

Thus, if agent i maintains the two most recent decisions communicated from agent j, ${}^i\mathbf{v}_j$ and ${}^i\mathbf{v}_j^{\text{old}}$, the coupling can be estimated by agent i at each iteration t using

$$ {}^i\hat{K}_{ij}(t) = \frac{1}{\|\Delta\mathbf{v}_j\|} \left| \mathbf{e}_i^T \nabla_i J\left({}^i\mathbf{v}\right) - \mathbf{e}_i^T \nabla_i J\left({}^i\mathbf{v}^{\text{old},j}\right) \right|, \tag{5.28} $$

where $\Delta\mathbf{v}_j = {}^i\mathbf{v}_j - {}^i\mathbf{v}_j^{\text{old}}$, ${}^i\mathbf{v}^{\text{old},j} = {}^i\mathbf{v} - [0,\dots,\Delta\mathbf{v}_j,\dots,0]$, and $\mathbf{e}_i$ is chosen to lie in the direction the local decision will be updated in, defined in (5.6) as $-\mathbf{A}_i^{-1}\nabla_i J({}^i\mathbf{v})$.

This allows the inter-agent coupling to be evaluated locally by each agent using only gradient evaluations. Furthermore, only the inner product between the gradient and a unit vector is required, which can be significantly cheaper than a direct gradient evaluation.

5.4.1 Internal Coupling

The same method can be used by agent i to produce an estimate ${}^i\hat{K}_{ii}$ of its internal coupling. However, if the stronger condition of Assumption 3b holds, the Hessian submatrix $\nabla_{ii}^2 J(\mathbf{v})$ is available locally and the internal coupling can be estimated directly using

$$ {}^i\hat{K}_{ii}(t) = \mathbf{e}_i^T \nabla_{ii}^2 J\left({}^i\mathbf{v}\right)\mathbf{e}_i. \tag{5.29} $$

5.4.2 Dynamic Communication Rates

Through the communication policy (5.25), the communication rate is determined by the inter-agent coupling, which is now estimated on-line and may change throughout the optimisation procedure in response to the local curvature of the objective function. Thus, the communication rate can also be updated on-line based on the current value of the coupling estimate

$$ C_{ij}(t) = \eta_i \sqrt{{}^i\hat{K}_{ij}(t)}. \tag{5.30} $$

However, this dynamic communication rate will prevents the delay bounds B_{ij} from being calculated accurately. To address this it is proposed to simply let agent i approximate on-line the delays to agent j using the current communication rates

$$^{i}\hat{B}_{ij}(t) = \frac{R_i}{C_{ij}(t)} + R_i D_{ij}. \tag{5.31}$$

The delays from agent j can be estimated using a communication rate C_{ji} determined from the time between received messages

$$^{i}\hat{B}_{ji}(t) = \frac{R_j}{C_{ji}} + R_j D_{ji}. \tag{5.32}$$

5.4.3 Scaling Normalisation Approximation

This idea of producing estimates of quantities based only on their current values, can be extended to the scaling normalisation constant a_i

$$\hat{a}_i(t) = \mathbf{e}_i^T \mathbf{A}_i(t)\mathbf{e}_i. \tag{5.33}$$

This leads to an estimate that more appropriately reflects the normalisation required for the current scaling matrix in the update direction that will be used. It is noted that if the scaling matrix $\mathbf{A}_i(t)$ is the local Hessian submatrix $\nabla^2_{ii} J(^{i}\mathbf{v})$, the scaling normalisation $\hat{a}_i(t)$ is equal to the internal coupling estimate $^{i}\hat{K}_{ii}(t)$.

5.4.4 Dynamic Step Size

Finally, these approximations are used to calculate the step size γ_i of the current iteration

$$\gamma_i(t) = \frac{\beta \hat{a}_i(t)}{^{i}\hat{K}_{ii}(t) + \sum_{j \neq i} {^{i}\hat{K}_{ij}(t)}(1 + {^{i}\hat{B}_{ij}(t)} + {^{i}\hat{B}_{ji}(t)})}. \tag{5.34}$$

This step size will vary in response to the coupling estimates, which in turn are dependent on the properties of the objective function at the current iteration, and influence the inter-agent communication rates.

5.4.5 Approximate Algorithm

Incorporating the above approximations, a new distributed optimisation procedure is defined by Algorithm 5.2. This algorithm determines the value of η_i such that agent i communicates once per iteration to the agent it has the greatest coupling to, however this could also be set to obey some bandwidth constraint. It is assumed the

Algorithm 5.2: Local algorithm for agent i. This algorithm approximates the inter-agent coupling terms on-line

```
Input: $J, \mathbf{v}^0, \beta$
Output: $\mathbf{v}^*$
 1: ${}^i\mathbf{v} \Leftarrow \mathbf{v}^0$ // Initialise local copy of team decision vector
 2: for all $j \neq i$ do
 3:     Initialise communication link to $j$
 4:     Send (receive) computation rate $R_i$ ($R_j$)
 5:     Determine transmission delays $D_{ij}$ and $D_{ji}$
 6: end for
 7: repeat
 8:     $\mathbf{g}_i \Leftarrow \nabla_i J({}^i\mathbf{v})$ // Local gradient
 9:     $\mathbf{A}_i \Leftarrow \nabla^2_{ii} J({}^i\mathbf{v})$ or $\mathbf{I}$ // Generate scaling matrix
10:     $\mathbf{d}_i \Leftarrow -\mathbf{A}_i^{-1}\mathbf{g}_i$ // Update direction
11:     $\mathbf{e}_i \Leftarrow \mathbf{d}_i/\|\mathbf{d}_i\|$ // Unit vector in update direction
12:     $\hat{a}_i \Leftarrow \mathbf{e}_i^T\mathbf{A}_i\mathbf{e}_i$ // Scaling normalisation
13:     $g_{\mathbf{e}_i} \Leftarrow \mathbf{e}_i^T\mathbf{g}_i$ // Gradient in update direction
14:     ${}^i\hat{K}_{ii} \Leftarrow \mathbf{e}_i^T\nabla^2_{ii} J({}^i\mathbf{v})\mathbf{e}_i$ // Or via finite diff. approx.
15:     for all $j \neq i$ do {Calculate coupling and delay terms}
16:         ${}^i\mathbf{v}^{\text{old},j} \Leftarrow {}^i\mathbf{v} - [0,\ldots,\Delta\mathbf{v}_j,\ldots,0]$ // Perturbed decision
17:         $g^{\text{old}}_{\mathbf{e}_i} \Leftarrow \mathbf{e}_i^T\nabla_i J({}^i\mathbf{v}^{\text{old},j})$ // Perturbed gradient
18:         ${}^i\hat{K}_{ij} \Leftarrow |g_{\mathbf{e}_i} - g^{\text{old}}_{\mathbf{e}_i}|/\|\Delta\mathbf{v}_j\|$ // Inter-agent coupling†
19:         $\hat{B}_{ij} \Leftarrow R_i/C_{ij} + R_i D_{ij}$ // Inter-agent delay term to $j$
20:         $\hat{B}_{ji} \Leftarrow R_j/C_{ji} + R_j D_{ji}$ // Requires comm. rate of $j$
21:     end for
22:     $\gamma_i \Leftarrow \dfrac{\beta\hat{a}_i}{{}^i\hat{K}_{ii} + \sum_{j\neq i}{}^i\hat{K}_{ij}(1 + {}^i\hat{B}_{ij} + {}^i\hat{B}_{ji})}$ // Step size
23:     ${}^i\mathbf{v}_i \Leftarrow \text{Proj}^{\mathbf{A}_i}_{\mathcal{V}_i}({}^i\mathbf{v}_i + \gamma_i\mathbf{d}_i)$ // Update local decision
24:     for all $j \neq i$ do {Manage communications}
25:         $\eta_i \Leftarrow R_i/\max_{j\neq i}\sqrt{{}^i\hat{K}_{ij}}$ // Or via bandwidth constraint
26:         $C_{ij} \Leftarrow \eta_i\sqrt{{}^i\hat{K}_{ij}}$
27:         if Req. to send msg to $j$ then {Determined from $C_{ij}$}
28:             Send $m_{ij} = {}^i\mathbf{v}_i$ to $j$
29:         end if
30:         if Msg $m_{ji} = {}^j\mathbf{v}_j$ received from $j$ then
31:             $\Delta\mathbf{v}_j \Leftarrow {}^i\mathbf{v}_j - {}^j\mathbf{v}_j$ // Determine change in decision of $j$
32:             ${}^i\mathbf{v}_j \Leftarrow {}^j\mathbf{v}_j$ // Update local copy
33:             Use time between messages to estimate $C_{ji}$
34:         end if
35:     end for
36: until Converged
37: return ${}^i\mathbf{v}$
```

† The inter-agent coupling ${}^i\hat{K}_{ij}$ can only be estimated after two messages have been received from j, prior to this it can be set to zero. For simplicity the additional logic to deal with this has been ignored.

message delivery delays (D_{ij}) are fixed and estimated at the start of the algorithm, when the local computation rates are exchanged, however these estimates could also be refined during execution.

It is noted that the inter-agent coupling terms ${}^{i}\hat{K}_{ij}$ are estimated using information from the two previous messages communicated by j, and thus, before this information has been received, no estimate can be made. During this time the estimate can be set to zero, or a value similar to the internal coupling, ${}^{i}\hat{K}_{ij} = {}^{i}\hat{K}_{ii}$. For simplicity, the extra logic required to deal with this situation has been ignored.

The results of Theorem 2 only guarantee convergence when the step sizes are smaller than the limit defined in (5.18). However, the proposed approximations may lead to step sizes that do not obey this condition. Thus, even for convex objective functions, Algorithm 5.2 is not strictly guaranteed to converge and it is expected that some tuning of the parameter β may be necessary. This issue will be explored empirically in Sect. 5.5 for a quadratic objective function. Alternative methods are also defined in Mathews (2008).

5.5 Example: Quadratic Optimisation

This section explores the convergence properties of Algorithm 5.2 to understand it's performance and allow suitable values for β can be determined. Consider a quadratic objective function of the form

$$J(\mathbf{v}) = \frac{1}{2}\mathbf{v}^T\mathbf{Q}\mathbf{v} + \mathbf{b}\mathbf{v}, \tag{5.35}$$

where $\mathbf{Q}$ is a positive definite matrix of dimension $d \times d$. For simplicity and without loss of generality the coefficient $\mathbf{b}$ will be set to 0. Now, the gradient and Hessian are given by

$$\nabla J(\mathbf{v}) = \mathbf{Q}\mathbf{v}, \quad \nabla^2 J(\mathbf{v}) = \mathbf{Q}. \tag{5.36}$$

The vector $\mathbf{v}$ is partitioned into p components with the ith component of dimension d_i. The partitioned elements of the gradient and Hessian are given by

$$\nabla_i J(\mathbf{v}) = \sum_{j=1}^{p} \mathbf{Q}_{ij}\mathbf{v}_j, \quad \nabla_{ij}^2 J(\mathbf{v}) = \mathbf{Q}_{ij} \tag{5.37}$$

where $\mathbf{Q}_{ij}$ is the appropriate $d_i \times d_j$ submatrix of $\mathbf{Q}$.

Since the Hessian matrix is constant, the coupling between agent i and j is given by

$$K_{ij} = \|\mathbf{Q}_{ij}\|. \tag{5.38}$$

5.5.1 Numerical Convergence Results

The specific example considered here consists of three agents, each in control of a 2 dimensional local decision. The Hessian is defined as

$$\mathbf{Q} = \left[\begin{array}{c|c|c} \mathbf{Q}_{11} & \mathbf{Q}_{12} & \mathbf{Q}_{13} \\ \hline \mathbf{Q}_{21} & \mathbf{Q}_{22} & \mathbf{Q}_{23} \\ \hline \mathbf{Q}_{31} & \mathbf{Q}_{32} & \mathbf{Q}_{33} \end{array}\right] = \left[\begin{array}{cc|cc|cc} 4 & 1 & 1 & 2 & 1 & 1 \\ 1 & 4 & 2 & 2 & 2 & 1 \\ \hline 1 & 2 & 5 & 3 & 2 & 4 \\ 2 & 2 & 3 & 5 & 2 & 3 \\ \hline 1 & 2 & 2 & 2 & 6 & 2 \\ 1 & 1 & 4 & 3 & 2 & 6 \end{array}\right].$$

For this system the weak coupling condition is not obeyed and the nonlinear Jacobi algorithm, defined in Corollary 1, diverges.

Although not used in Algorithm 5.2, the coupling can be easily calculated using (5.38)

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \approx \begin{bmatrix} 5 & 3.56 & 2.62 \\ 3.56 & 8 & 5.73 \\ 2.62 & 5.73 & 8 \end{bmatrix}. \tag{5.39}$$

In this system agents 2 and 3 are strongly coupled ($K_{23} \approx 5.73$), while agents 1 and 2 have a medium amount of coupling ($K_{12} \approx 3.56$) and agents 1 and 3 are weakly coupled ($K_{13} \approx 2.62$).

The algorithm is initialised with the normalised sum of the eigenvectors

$$\mathbf{v}^0 = \frac{\sum_{\ell=1}^{6} \boldsymbol{\epsilon}_\ell(\mathbf{Q})}{\|\sum_{\ell=1}^{6} \boldsymbol{\epsilon}_\ell(\mathbf{Q})\|} \approx \begin{bmatrix} 0.09 \\ -0.44 \\ 0.66 \\ -0.29 \\ 0.25 \\ 0.46 \end{bmatrix}, \tag{5.40}$$

where $\boldsymbol{\epsilon}_\ell(\mathbf{Q})$ is the ℓth eigenvector of the matrix $\mathbf{Q}$. This guarantees that the initial guess does not lie along some degenerate direction, with a simplistic solution.

Each agent computes iterations at the same rate, with a system wide delivery delay, i.e. $D_{ij} = D$, for all i, $j \neq i$. Figure 5.3 plots the convergence ratio of the algorithm for different delivery delays D and values of β. The theory accurately predicts convergence up to $\beta = 2$, after which the algorithm may become unstable and for most $\beta > 3$ diverges. This result suggests that to achieve a reasonable convergence speed, without the possibility of diverging, a suitable choice for β is 1.5. This value will be used in the numerical examples presented in Sect. 5.10.

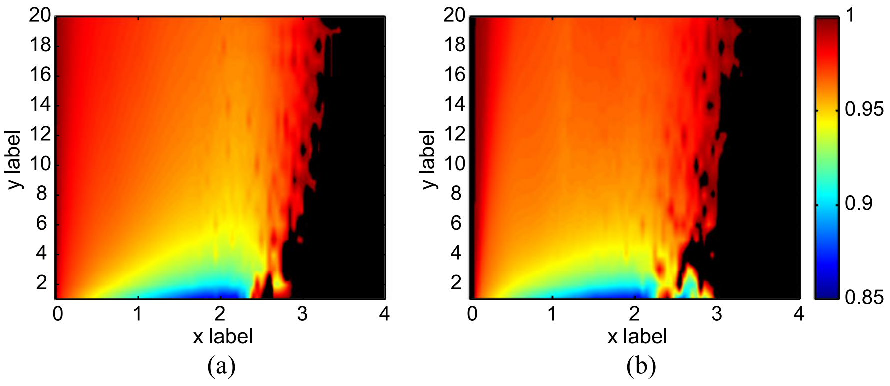


Fig. 5.3 Convergence rate, averaged over 300 iterations, for different β values and communication delays (D, defined relative to the time to compute a single iteration), using (**a**) the scaled and (**b**) steepest descent variants. *Black* represents the algorithm diverged

5.6 Heterogeneous System and Modularity

Until now it has been an implicitly requirement that each agent has a full representation of the systems objective function. In general this requires detailed operational information of the other agents, for instance if the agents are mobile robots the operational information will include the state and sensor/actuator models. This may not be a problem for a small number of agents, but poses a significant issue for a large distributed system.

Ideally, each agent should only require a detailed knowledge about itself and relatively little knowledge of the other agents. This issue has been examined previously in Mathews et al. (2006, 2009) and a composable or partially separable form of the objective function introduced that enables the separation of information about the local agent and more abstract information about other agents. This section will introduce this concept and reformulate the previous optimisation algorithm to build on this framework.

5.6.1 Partial Separability

An objective function in partially separable form allows the agent specific operations that are required to be performed on its local decision to be separated from operations that allow the effects, or impacts, of multiple agents to be combined to form a single scale objective function value. This will be useful in further defining what information each agent needs to know about the others, and also save computational resources by allowing each agent to perform these local operations before communicating decision updates, which will remove the need for all receiving agents to perform a same set of operations.

Definition 1 (Partial Separability) A partially separable system has an objective function of the form

$$J(\mathbf{v}) = \psi\big(\Upsilon_1(\mathbf{v}_1) * \Upsilon_2(\mathbf{v}_2) * \cdots * \Upsilon_p(\mathbf{v}_p)\big), \tag{5.41}$$

where $\Upsilon_i : \mathcal{V}_i \to \mathcal{I}$ is the ith agents *impact function* and maps a decision to an element of an *impact space* $\mathcal{I}$. An element $\alpha \in \mathcal{I}$ of this space will be referred to as an *impact*. The *composition operator* $* : \mathcal{I} \times \mathcal{I} \to \mathcal{I}$ allows impacts to be summarised without losing any task relevant information. The *generalised objective function* $\psi : \mathcal{I} \to \Re$ maps a combined team impact to a scalar cost.

The impact function of a mobile robotic agent would define an abstraction of its state and sensor/actuator models and maps a given decision onto a task specific impact space. It is assumed each agent i only has knowledge of its own impact function Υ_i and thus requires the impacts $\alpha_j = \Upsilon_j(\mathbf{v}_j)$ from each other agent $j \neq i$ for the objective function to be evaluated. Thus, instead of each agent maintaining a local copy of each agents decision vector $\mathbf{v}_j$, it simply maintains their impact α_j. This definition allows the information about the operational models of other agents to be abstracted out and defines a common language of impacts that the agents use to communicate.

For simplicity, the cumulative composition operator $\bigodot$ will be used, such that (5.41) can be written as

$$J(\mathbf{v}) = \psi\left(\bigodot_{i=1}^{p} \Upsilon_i(\mathbf{v}_i)\right). \tag{5.42}$$

5.6.1.1 Example: Collision Avoidance

To provide an example of an impact, consider a multi-agent path planning scenario with a separation requirement. For this task the objective function will be dependent on the separation between the agents, which in turn requires the individual paths (possibly represented by a fixed number of points) from all the agents. In this case each agents path abstracts away its motion model and corresponding control inputs. Thus, an agents path can be consider its impact and the composition operator simply collects the paths. The generalised objective function is used to calculate the associated cost of these specific paths.

It is noted that for this example the size of the impact space is proportional to the number of agents (the number of paths is the same as the number of agents). This differs from the example presented in Sect. 5.8, which has a composition operator given by addition. For this case the size of the impact space is independent of the number of agents (the sum of many numbers is still a number).

5.6.2 Modular Decision Refinement

Using the partially separable form of the objective function, the local decision refinement process, presented in Sect. 5.2.2.1, can be modified such that each agent i is only required to maintain a local copy of the impact of each other agent

$$\left\{{}^{i}\alpha_{j}(t) : \forall j \neq i\right\}.$$

The explicit maintenance of the team decision vector, as defined in (5.4), is no longer required. The locally stored impacts are updated via messages from the other agents in the system and may contain delayed information (according to Assumption 5b).

From this modification, the local decision refinement equations for agent i (corresponding to (5.7)) can be rewritten as

$$f_i\left({}^{i}\mathbf{v}(t)\right) = \mathrm{Proj}_{\mathcal{V}_i}^{\mathbf{A}_i(t)}\left({}^{i}\mathbf{v}_i(t) - \gamma_i\left[\mathbf{A}_i(t)\right]^{-1}\nabla_i J\left({}^{i}\mathbf{v}(t)\right)\right), \tag{5.43}$$

where the local gradient $\nabla_i J({}^{i}\mathbf{v}(t))$ is now calculated according to given by

$$\nabla_i J\left({}^{i}\mathbf{v}(t)\right) = \nabla_i \psi\left(\Upsilon_i\left({}^{i}\mathbf{v}_i(t)\right) * \bigodot_{j\neq i} {}^{i}\alpha_j(t)\right). \tag{5.44}$$

With this straight forward change, the decision refinement process can be modified to use the communicated impacts instead of the raw decisions.

5.6.3 Coupling Estimation

Previously, the inter-agent coupling terms were estimated locally by each agent from the two most recent decisions received from each other agent using (5.28). If only the impacts of the remote agents are known and not the specific decisions, the gradient calculations required for this formula can be calculated in a similar fashion to (5.44). The only additional information required to estimate the coupling terms is the distance the underlying decision vector moved between the two communication events. Unfortunately, this information cannot be calculated from the abstract impacts and must be communicated separately. Thus, for each communicated impact an agent i sends to agents j, it must also send the distance the underlying decision moved since the last message was sent.

5.6.4 Modular Algorithm

With the above changes define the major modifications required to allow Algorithm 5.2 to exploit the abstractions of a partially separable objective function. The new algorithm is defined in Algorithm 5.3.

Algorithm 5.3: Local algorithm for agent i that exploits partially separable objective function that abstracts away the details of other agents

```
Input: $\phi$, $\Upsilon_i$, $\mathbf{v}_i^0$, $\beta$
Output: $\mathbf{v}_i^*$
 1: ${}^i\mathbf{v}_i \Leftarrow \mathbf{v}_i^0$  // Initialise local decision vector
 2: for all $j \neq i$ do
 3:     Initialise communication link to $j$
 4:     Send (receive) computation rate $R_i$ ($R_j$)
 5:     Determine transmission delays $D_{ij}$ and $D_{ji}$
 6:     ${}^i\alpha_j \Leftarrow \emptyset$  // Initialise impacts of remote agents with the null impact
 7: end for
 8: repeat
 9:     $\mathbf{g}_i \Leftarrow \nabla_i \psi(\Upsilon_i({}^i\mathbf{v}_i) * \bigodot_{j\neq i} {}^i\alpha_j)$  // Local gradient
10:     $\mathbf{A}_i \Leftarrow \nabla^2_{ii} \psi(\Upsilon_i({}^i\mathbf{v}_i) * \bigodot_{j\neq i} {}^i\alpha_j)$ or $\mathbf{I}$  // Generate scaling matrix
11:     $\mathbf{d}_i \Leftarrow -\mathbf{A}_i^{-1}\mathbf{g}_i$  // Update direction
12:     $\mathbf{e}_i \Leftarrow \mathbf{d}_i / \|\mathbf{d}_i\|$  // Unit vector in update direction
13:     $\hat{a}_i \Leftarrow \mathbf{e}_i^T \mathbf{A}_i \mathbf{e}_i$  // Scaling normalisation
14:     $g_{\mathbf{e}_i} \Leftarrow \mathbf{e}_i^T \mathbf{g}_i$  // Gradient in update direction
15:     ${}^i\hat{K}_{ii} \Leftarrow \mathbf{e}_i^T \nabla^2_{ii} \psi(\Upsilon_i({}^i\mathbf{v}_i) * \bigodot_{j\neq i} {}^i\alpha_j)\mathbf{e}_i$  // Or via finite diff. approx.
16:     for all $j \neq i$ do {Calculate coupling and delay terms}
17:         $g^{\text{old}}_{\mathbf{e}_i} \Leftarrow \mathbf{e}_i^T \nabla_i \psi(\Upsilon_i({}^i\mathbf{v}_i) * {}^i\alpha_j^{\text{old}} * \bigodot_{k\neq i,j} {}^i\alpha_k)$  // Perturbed gradient
18:         ${}^i\hat{K}_{ij} \Leftarrow |g_{\mathbf{e}_i} - g^{\text{old}}_{\mathbf{e}_i}| / \delta_j$  // Inter-agent coupling$^\dagger$
19:         $\hat{B}_{ij} \Leftarrow R_i / C_{ij} + R_i D_{ij}$  // Inter-agent delay term to $j$
20:         $\hat{B}_{ji} \Leftarrow R_j / C_{ji} + R_j D_{ji}$  // Requires comm. rate of $j$
21:     end for
22:     $\gamma_i \Leftarrow \dfrac{\beta \hat{a}_i}{{}^i\hat{K}_{ii} + \sum_{j\neq i} {}^i\hat{K}_{ij}(1 + {}^i\hat{B}_{ij} + {}^i\hat{B}_{ji})}$  // Step size
23:     ${}^i\mathbf{v}_i \Leftarrow \mathrm{Proj}^{\mathbf{A}_i}_{\mathcal{V}_i}({}^i\mathbf{v}_i + \gamma_i \mathbf{d}_i)$  // Update local decision
24:     ${}^i\alpha_i = \Upsilon_i({}^i\mathbf{v}_i)$  // Compute local impact
25:     for all $j \neq i$ do {Manage communications}
26:         $\eta_i \Leftarrow R_i / \max_{j\neq i} \sqrt{{}^i\hat{K}_{ij}}$  // Or via bandwidth constraint
27:         $C_{ij} \Leftarrow \eta_i \sqrt{{}^i\hat{K}_{ij}}$
28:         if Req. to send msg to $j$ then {Determined from $C_{ij}$}
29:             $\delta_i = \|{}^i\mathbf{v}_i - {}^i\mathbf{v}_i^{\text{old},j}\|$  // Distance raw decision has moved since last
                msg
30:             Send $m_{ij} = ({}^i\alpha_i, \delta_i)$ to $j$
31:             ${}^i\mathbf{v}_i^{\text{old},j} = {}^i\mathbf{v}_i$  // Store current decision
32:         end if
33:         if Msg $m_{ji} = ({}^j\alpha_j, \delta_j)$ received from $j$ then
34:             Update $\delta_j$ using communicated value
35:             ${}^i\alpha_j^{\text{old}} \Leftarrow {}^i\alpha_j$  // Save current impact for use in coupling
                calculation
36:             ${}^i\alpha_j \Leftarrow {}^j\alpha_j$  // Update local copy of impact
37:             Use time between messages to estimate $C_{ji}$
38:         end if
39:     end for
40: until Converged
41: return ${}^i\mathbf{v}$
```

† The inter-agent coupling ${}^i\hat{K}_{ij}$ can only be estimated after two messages have been received from j, prior to this it can be set to zero. For simplicity the additional logic to deal with this has been ignored.

5.7 Active Information Gathering

The application considered in this work consists of multiple mobile robotic agents undertaking a reconnaissance or information gathering task. This type of scenario requires the agents to actively gather information on a particular external random variable $\mathbf{x} \in \mathcal{X}$. In general this may include the positions and identities of stationary or mobile objects, terrain properties of a given region, or any other state of interest. In Sect. 5.7 this will be specialised for the active localisation of a group of objects.

5.7.1 Agent Models

The mobile robotic agents are modelled as discrete time dynamical systems, with the ith agents state given by $\mathbf{s}_i \in \mathcal{S}_i$. The agent is controlled from discrete time $k-1$ to k by applying a particular control input $\mathbf{u}_i^k \in \mathcal{U}_i$. In general this causes the agents state to change according to the probabilistic discrete time Markov motion model $P(\mathbf{s}_i^k|\mathbf{s}_i^{k-1}, \mathbf{u}_i^k)$. However, for simplicity it is assumed that the agent's motion is know with precision, i.e. $\mathbf{s}_i^k = \mathbf{f}_i(\mathbf{s}_i^{k-1}, \mathbf{u}_i^k)$. The joint system state and transition model is given by $\mathbf{s}^k = \mathbf{f}(\mathbf{s}^{k-1}, \mathbf{u}^k) = \{\mathbf{f}_1(\mathbf{s}_1^{k-1}, \mathbf{u}_1^k), \ldots, \mathbf{f}_p(\mathbf{s}_p^{k-1}, \mathbf{u}_p^k)\} = \{\mathbf{s}_1^k, \ldots, \mathbf{s}_p^k\}$.

The observations made by the ith agent regarding the variable of interest $\mathbf{x}$, are modelled by the conditional density $P(\mathbf{z}_i^k|\mathbf{x}^k; \mathbf{s}_i^k)$ which describes the probability of obtaining a particular observation $\mathbf{z}_i^k$ given the external state $\mathbf{x}^k$ and agents state $\mathbf{s}_i^k$. The notation $\mathbf{z}^k$ will denote the set of observations made by all the agents at time step k, i.e. $\mathbf{z}^k = \{\mathbf{z}_1^k, \ldots, \mathbf{z}_p^k\} \in \mathcal{Z} = \prod_{i=1}^p \mathcal{Z}_i$. Assuming that the observations are conditionally independent given the states $\mathbf{x}^k$ and $\mathbf{s}_i^k$, the combined sensor model can be written as $P(\mathbf{z}^k|\mathbf{x}^k; \mathbf{s}^k) = \prod_{i=1}^p P(\mathbf{z}_i^k|\mathbf{x}^k; \mathbf{s}_i^k)$.

It is desired that the agents are controlled such that the combined observations they receive produce the most informative (or least uncertain) belief about the value of the random variable $\mathbf{x}$. To accomplish this the distribution's entropy will be used as a measure of its associated uncertainty. The entropy of a distribution $P(\mathbf{x})$, is the negative of the expectation of its logarithm

$$H_{P(\mathbf{x})} = -\mathrm{E}_{\mathbf{x}}\big\{\log P(\mathbf{x})\big\}. \tag{5.45}$$

This can be used for continuous variables, where $P(\mathbf{x})$ is the probability density function, or for discrete variables, where $P(\mathbf{x})$ is the probability distribution function. For a detailed description of the properties of this metric see Cover and Thomas (1991).

5.7.2 Bayes Filtering

This section details the process of maintaining an accurate belief (as a probability) about the state $\mathbf{x}$. Further details of this process can be found in Mathews (2008). Bayes Filtering will be used for two purposes: (i) to maintain a current belief for the state $\mathbf{x}$ that is updated as new observations are made, and (ii) to predict the effect future observations will have on a future belief of $\mathbf{x}$. The Bayesian approach allows the system to be given some prior belief, however, if nothing is known this may simply be a noninformative uniform distribution. Once this has been established the belief at any later stage can be constructed recursively. To avoid potential confusion, instantiated variables (i.e. random variables with a known value) will be denoted using a tilde, e.g. $P(\tilde{\mathbf{x}}) \equiv P(\mathbf{x} = \tilde{\mathbf{x}})$.

Consider the system at a given time step k. The system's state is given by $\tilde{\mathbf{s}}^k$ and the belief about $\mathbf{x}^k$, conditioned on all past observations and agent configurations, is $P(\mathbf{x}^k|\tilde{\mathbf{Z}}^k;\tilde{\mathbf{S}}^k)$, where $\tilde{\mathbf{Z}}^k = \{\tilde{\mathbf{z}}^1, \ldots, \tilde{\mathbf{z}}^k\}$ and $\tilde{\mathbf{S}}^k = \{\tilde{\mathbf{s}}^1, \ldots, \tilde{\mathbf{s}}^k\}$ and $\tilde{\mathbf{Z}}^0 = \tilde{\mathbf{S}}^0 = \{\emptyset\}$.

When a joint control action, $\tilde{\mathbf{u}}^{k+1}$ is taken, the new state of the agents becomes $\tilde{\mathbf{s}}^{k+1} = \mathbf{f}(\tilde{\mathbf{s}}^k, \tilde{\mathbf{u}}^{k+1})$, and an observation $\tilde{\mathbf{z}}^{k+1}$ is received. To update the belief about $\mathbf{x}^{k+1}$, it must first be predicted forward in time using the Chapman-Kolmogorov equation

$$P\big(\mathbf{x}^{k+1}|\tilde{\mathbf{Z}}^k;\tilde{\mathbf{S}}^k\big) = \int_{\mathcal{X}} P\big(\mathbf{x}^{k+1}|\mathbf{x}^k\big) P\big(\mathbf{x}^k|\tilde{\mathbf{Z}}^k;\tilde{\mathbf{S}}^k\big) d\mathbf{x}^k. \tag{5.46}$$

The belief can now be updated using Bayes rule

$$P\big(\mathbf{x}^{k+1}|\tilde{\mathbf{Z}}^{k+1};\tilde{\mathbf{S}}^{k+1}\big) = \frac{1}{\mu} P\big(\mathbf{x}^{k+1}|\tilde{\mathbf{Z}}^k;\tilde{\mathbf{S}}^k\big) \prod_{i=1}^{p} P\big(\tilde{\mathbf{z}}_i^{k+1}|\mathbf{x}^{k+1};\tilde{\mathbf{s}}_i^{k+1}\big) \tag{5.47}$$

where $\tilde{\mathbf{z}}^{k+1} = \{\tilde{\mathbf{z}}_1^{k+1}, \ldots, \tilde{\mathbf{z}}_p^{k+1}\}$ are the actual observations taken by the agents. The term $P(\tilde{\mathbf{z}}_i^{k+1}|\mathbf{x}^{k+1}, \tilde{\mathbf{s}}_i^{k+1})$ is the ith agents observation model evaluated at the actual observation and agent configuration, resulting in a likelihood over $\mathbf{x}^{k+1}$. The normalisation constant μ is given by

$$\begin{aligned} \mu &= P\big(\tilde{\mathbf{z}}^{k+1}|\tilde{\mathbf{Z}}^k;\tilde{\mathbf{S}}^{k+1}\big) \\ &= \int_{\mathcal{X}} P\big(\mathbf{x}^{k+1}|\tilde{\mathbf{Z}}^k;\tilde{\mathbf{S}}^k\big) \prod_{i=1}^{p} P\big(\tilde{\mathbf{z}}_i^{k+1}|\mathbf{x}^{k+1};\tilde{\mathbf{s}}_i^{k+1}\big) d\mathbf{x}^{k+1}. \end{aligned} \tag{5.48}$$

For each agent to maintain this belief, each agent must communicate the observation likelihood function $\lambda(\mathbf{x}^{k+1}) = P(\tilde{\mathbf{z}}_i^{k+1}|\mathbf{x}^{k+1}, \tilde{\mathbf{s}}_i^{k+1})$ after each observation is made. The field of Decentralised Data Fusion examines efficient ways for this to communicated around a sensor network (Liggins et al. 1997; Manyika and Durrant-Whyte 1994), however for this work it is assumed each agent simply communicates it to every other agent.

The key problem of interest in this process is deciding on the systems control inputs $\mathbf{u}^k$ such that the future uncertainty in the state $\mathbf{x}$ is minimised.

5.7.3 Control Parameterisation

Although the goal of the system is to minimise its uncertainty in its joint belief about $\mathbf{x}$. There are many ways to formally define this control problem, the best being a discounted infinite horizon dynamic programming problem (Bertsekas 2005). However, for any relatively complex scenario this becomes intractable and approximate techniques must be used.

Thus, a finite look ahead will be considered and an open loop control policy, or plan, for each agent developed. To accommodate feedback, a rolling time horizon will be employed. This requires the open loop control policies to be recomputed at short intervals to keep the look ahead approximately constant and allows the system to adapt to changes.

The control policy shall be parameterised by a piecewise constant function, defined over N equal time partitions of M times steps each (Goh and Teo 1988). This results in a look ahead of NM time steps. Thus, the open loop control policy for a time interval $[k+1, k+NM]$ can be specified with the parameters $\mathbf{v}_i^k = \{\mathbf{v}_i^k(1), \ldots, \mathbf{v}_i^k(N)\} \in \mathcal{V}_i = (\mathcal{U}_i)^N$ with actual controls given at each time step $k+q$ by $\mathbf{u}_i^{k+q} = \mathbf{v}_i^k(\lceil \frac{q}{M} \rceil)$, where $q \in \{1, \ldots, NM\}$ and $\lceil \cdot \rceil$ represents the round up operator.

5.7.4 Objective Function

For a given time k, the utility of a joint open loop control policy, or joint plan, $\mathbf{v}^k = \{\mathbf{v}_1^k, \ldots, \mathbf{v}_p^k\} \in \mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_p$ and observation series $\mathbf{z}^{k+1:k+NM} = \{\mathbf{z}^{k+1}, \ldots, \mathbf{z}^{k+NM}\}$ is defined by the amount of uncertainty in the resulting posterior belief at time $k+NM$. Actions and observations that produce a smaller uncertainly in the posterior belief are favored over others. The cost function used to measure uncertainty is the entropy of the posterior distribution

$$\begin{aligned} &C^k\big(\mathbf{z}^{k+1:k+NM}, \mathbf{v}^k\big) \\ &\quad = H_{P(\mathbf{x}^{k+NM} | \mathbf{z}^{k+1:k+NM}, \mathbf{s}^{k+1:k+NM}(\mathbf{v}^k), \tilde{\mathbf{Z}}^k, \tilde{\mathbf{S}}^k)} \\ &\quad = -\mathrm{E}_{\mathbf{x}^{k+NM}}\big\{\log P\big(\mathbf{x}^{k+NM} \big| \mathbf{z}^{k+1:k+NM}, \mathbf{s}^{k+1:k+NM}\big(\mathbf{v}^k\big), \tilde{\mathbf{Z}}^k, \tilde{\mathbf{S}}^k\big)\big\}. \end{aligned} \tag{5.49}$$

However, the actual observations that will be made in the future will not be known in advance. Thus, an expectation over all possible future observations must be performed, resulting in the expected team cost or objective function

$$J^k\big(\mathbf{v}^k\big) = \mathrm{E}_{\mathbf{z}^{k+1:k+NM}}\big\{C^k\big(\mathbf{z}^{k+1:k+NM}, \mathbf{v}^k\big)\big\}. \tag{5.50}$$

The finite horizon optimal control problem, at time k, becomes the parameter optimisation problem

$$\mathbf{v}^{k*} = \arg\min_{\mathbf{v}^k \in \mathcal{V}} J^k\big(\mathbf{v}^k\big). \tag{5.51}$$

For a rolling time horizon, this must be resolved every $N_r \leq NM$ time steps.

This parameter optimisation problem translates directly to the distributed decision problem discussed earlier and, provided the objective function is partially separable, can be solved using the distributed optimisation algorithm defined in Algorithm 5.3. It is noted that this approach can only be applied if the time required to compute the solution is significantly less that the time scale of the system dynamics.

5.8 Example: Object Localisation

The proposed distributed control strategy has been implemented in a simulated object localisation task. For this scenario, robots equipped with bearing only sensors (e.g. Fig. 5.4) and moving in a 2D plane, are required to cooperatively localise a collection of stationary point objects.

The multi-agent control problem for this type of scenario has been previously examined by Grocholsky et al. (2003). In this work each agent shared past observations but developed a plan independently. This section extends the work of Grocholsky by applying the distributed optimisation procedure to find the optimal joint plans.

5.8.1 Modelling

5.8.1.1 Objects

The state $\mathbf{x}$ is separated into m independent objects, thus

$$\mathbf{x} = \{\mathbf{x}_{o_1}, \ldots, \mathbf{x}_{o_m}\}. \tag{5.52}$$

Since the objects are stationary the time index has been dropped.

Each object o_j is specified by a 2D position $\mathbf{x}_{o_j} = [x_{o_j}, y_{o_j}]^T$ that is independent of all other objects.

5.8.1.2 Agent Motion

The agents are described by their position and orientation, $\mathbf{s}_i^k = [x_i^k, y_i^k, \theta_i^k]^T$, travel at a constant velocity $V_i = 50$ m/s and are controlled via a single scalar defining the robots rate of turn $\mathbf{u}_i^k = \dot{\theta}_i^k$. Thus, the deterministic motion model $\mathbf{s}_i^{k+1} = \mathbf{f}_i(\mathbf{s}_i^k, \mathbf{u}_i^{k+1})$ is given by

$$x_i^{k+1} = x_i^k + \frac{2V_i}{\mathbf{u}_i^{k+1}} \sin\left(\frac{1}{2}\mathbf{u}_i^{k+1}\Delta t\right) \cos\left(\theta_i^{k+1} + \frac{1}{2}\mathbf{u}_i^{k+1}\Delta t\right) \tag{5.53a}$$

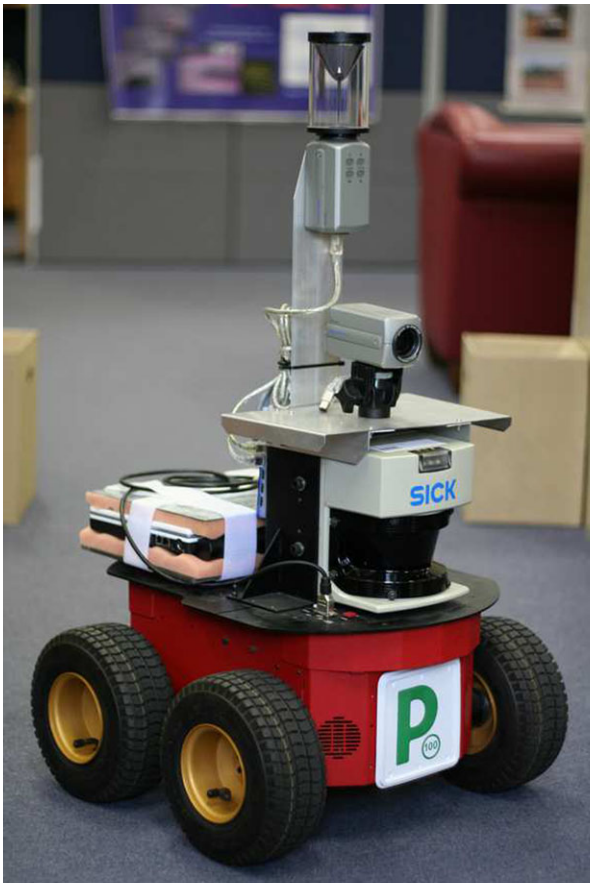


Fig. 5.4 Typical mobile robot equipped with a bearing only sensor (vertically mounted panoramic camera)

$$y_i^{k+1} = x_i^k + \frac{2V_i}{\mathbf{u}_i^{k+1}} \sin\left(\frac{1}{2}\mathbf{u}_i^{k+1}\Delta t\right) \sin\left(\theta_i^{k+1} + \frac{1}{2}\mathbf{u}_i^{k+1}\Delta t\right) \tag{5.53b}$$

$$\theta_i^{k+1} = \theta_i^k + \mathbf{u}_i^{k+1}\Delta t \tag{5.53c}$$

where Δt is the time between k and $k+1$.

5.8.1.3 Observations

It is assumed each agent i receives an independent bearing observation $\mathbf{z}_{i,o_j}^k$ from each object o_j at each time k, thus $\mathbf{z}_i^k = \{\mathbf{z}_{i,o_j}^k : \forall j\}$. The independency assumptions allows the observations of the objects to be modelled separately.

The ith agents observation model for object o_j, defined as the conditional probability density, is assumed Gaussian and is given by

$$P\big(\mathbf{z}_{i,o_j}^k \big| \mathbf{x}_{o_j}; \mathbf{s}_i^k\big) = N\big(\mathbf{z}_{i,o_j}^k; \mathbf{h}_{i,o_j}\big(\mathbf{x}_{o_j}, \mathbf{s}^k\big), \mathbf{R}_{i,o_j}^k\big). \tag{5.54}$$

Here, the notation $N(\xi; \mathbf{m}_\xi, \mathbf{C}_\xi)$ represents a Gaussian (or normal) density defined on the state ξ with a mean of $\mathbf{m}_\xi$ and variance $\mathbf{C}_\xi$.

The mean observation for a given object o_j with state $\mathbf{x}_{o_j}$ when the agent is in state $\mathbf{s}_i^k$ is given by the nonlinear function

$$\begin{aligned}\bar{\mathbf{z}}_{i,o_j}^k &= \mathbf{h}_{i,o_j}\big(\mathbf{x}_{o_j}, \mathbf{s}_i^k\big) \\ &= \tan^{-1}\left(\frac{y_{o_j} - y_i^k}{x_{o_j} - x_i^k}\right).\end{aligned} \tag{5.55}$$

The variance of the bearing observations is set to $\mathbf{R}_{i,o_j}^k = 25$ (degrees)^2 for all agents and objects.

5.8.2 *Filtering*

Consider some time $k-1$, where the teams belief about $\mathbf{x}_{o_j}$ is Gaussian with mean $\bar{\mathbf{x}}_{o_j}^{|k-1}$ and covariance $\mathbf{P}_{o_j}^{|k-1}$

$$P\big(\mathbf{x}_{o_j} \big| \tilde{\mathbf{Z}}^{k-1}; \tilde{\mathbf{S}}^{k-1}\big) = N\big(\mathbf{x}_{o_j}; \bar{\mathbf{x}}_{o_j}^{|k-1}, \mathbf{P}_{o_j}^{|k-1}\big). \tag{5.56}$$

Here the notation $(\cdot)^{|k-1}$ represents "given information up to $k-1$".

Since all the objects are independent, the belief is simply given by the product of the individual probability densities for each object

$$P\big(\mathbf{x}^{k-1} \big| \tilde{\mathbf{Z}}^{k-1}; \tilde{\mathbf{S}}^{k-1}\big) = \prod_{j=1}^{m} P\big(\mathbf{x}_{o_j}^{k-1} \big| \tilde{\mathbf{Z}}^{k-1}; \tilde{\mathbf{S}}^{k-1}\big). \tag{5.57}$$

Thus, only the belief $P(\mathbf{x}_{o_j}^{k-1} | \mathbf{Z}^{k-1}; \mathbf{S}^{k-1})$ of a single object needs to be considered.

Due to the static nature of the objects, the prediction step (corresponding to (5.46)) can be skipped. If the update step, defined in (5.47), is implemented directly, the nonlinearity in the observation model, will cause the posterior to become non-Gaussian (even if the prior is Gaussian). The extended Kalman filter (Maybeck 1979–1982) overcomes this by linearising the observation model about a nominal state ${}_n\mathbf{x}_{o_j}^k$, usually defined by the prior mean $\bar{\mathbf{x}}_{o_j}^{|k-1}$.

Using a first order Taylor expansion on $\mathbf{h}_{i,o_j}(\mathbf{x}_{o_j}, \mathbf{s}_i^k)$ about ${}_n\mathbf{x}_{o_j}^k$ yields

$$\mathbf{h}_{i,o_j}\big(\mathbf{x}_{o_j}, \mathbf{s}_i^k\big) \approx {}_n\mathbf{z}_{i,o_j}^k + \mathbf{H}_{i,o_j}^k\big[\mathbf{x}_{o_j} - {}_n\mathbf{x}_{o_j}^k\big] \tag{5.58}$$

where ${}_n\mathbf{z}^k_{i,o_j} = \mathbf{h}_{i,o_j}({}_n\mathbf{x}^k_{o_j}, \mathbf{s}^k_i)$ is the nominal observation and the matrix $\mathbf{H}^k_{i,o_j} = \nabla_\mathbf{x}\mathbf{h}_{i,o_j}({}_n\mathbf{x}^k_{o_j}, \mathbf{s}^k_i)$ is the Jacobian of the observation function evaluated at the nominal object state and agent state.

For a bearing observation these become

$$ {}_n\mathbf{z}^k_{i,o_j} = \tan^{-1}\left(\frac{{}_n y^k_{o_j} - y^k_i}{{}_n x^k_{o_j} - x^k_i}\right) \tag{5.59}$$

and

$$\mathbf{H}^k_{i,o_j} = \frac{1}{{}_n r^k_{i,o_j}}\left[-\sin\left({}_n\mathbf{z}^k_{i,o_j}\right), \cos\left({}_n\mathbf{z}^k_{i,o_j}\right)\right] \tag{5.60}$$

where ${}_n r^k_{i,o_j} = \sqrt{({}_n y^k_t - y^k_i)^2 + ({}_n x^k_{o_j} - x^k_i)^2}$ is the range from the agent to the nominal object state.

With this linearised model the posterior will remain Gaussian for a Gaussian prior. The following equations define the update step and take as input the prior mean and covariance and an observation, and produce the associated posterior mean and covariance. They are given in information form, which propagates the Fisher information matrix $\mathbf{Y}$, defined as the inverse covariance matrix $\mathbf{Y} \equiv \mathbf{P}^{-1}$ instead of the covariance matrix

$$\mathbf{Y}^{|k}_{o_j} = \mathbf{Y}^{|k-1}_{o_j} + \sum_{i=1}^{p}\left(\mathbf{H}^k_{i,o_j}\right)^T\left(\mathbf{R}^k_{i,o_j}\right)^{-1}\mathbf{H}^k_{i,o_j}, \tag{5.61}$$

and

$$\begin{aligned}\mathbf{Y}^{|k}_{o_j}\bar{\mathbf{x}}^{|k}_{o_j} &= \mathbf{Y}^{|k-1}_{o_j}\bar{\mathbf{x}}^{|k-1}_{o_j} \\ &\quad + \sum_{i=1}^{p}\left(\mathbf{H}^k_{i,o_j}\right)^T\left(\mathbf{R}^k_{i,o_j}\right)^{-1}\left(\tilde{\mathbf{z}}^k_{i,o_j} - {}_n\mathbf{z}^k_{i,o_j} + \mathbf{H}^k_{i,o_j}\bar{\mathbf{x}}^{|k-1}_{o_j}\right).\end{aligned} \tag{5.62}$$

An interesting property of this representation is that the updated or posterior information matrix $\mathbf{Y}^{|k}_{o_j}$ (and hence covariance $\mathbf{P}^{|k}_{o_j}$) is independent of the actual observations, $\mathbf{z}^k_i$, taken (see (5.61)). This is an important property and will allow the expected entropy required for the objective function to be calculated very efficiently.

5.8.3 Objective Function

The objective function, defined in Sect. 5.7.4, represents the expected posterior entropy of the team belief at the end of an NM step time horizon. Since the objects are independent (the density can be decomposed into the product of the densities of each object alone) the entropy becomes a sum of the entropies of each individual object, thus

$$H_{P(\mathbf{x}|\tilde{\mathbf{Z}}^{k+NM};\tilde{\mathbf{S}}^{k+NM})} = \sum_{j=1}^{m} H_{P(\mathbf{x}_{o_j}|\tilde{\mathbf{Z}}^{k+NM};\tilde{\mathbf{S}}^{k+NM})}. \tag{5.63}$$

Where the entropy of the Gaussian density, for the object state, is given by

$$H_{P(\mathbf{x}_{o_j}|\tilde{\mathbf{Z}}^{k+NM};\tilde{\mathbf{S}}^{k+NM})} = -\frac{1}{2}\log\bigl((2\pi e)^{d_x}\bigl|\mathbf{Y}_{o_j}^{|k+NM}\bigr|\bigr) \tag{5.64}$$

where $d_x = 2$ is the dimension of the state $\mathbf{x}_{o_j}$.

It is noted that the entropy is only dependent on the information matrix $\mathbf{Y}_{o_j}^{|k+NM}$. By examining the modelling and filtering equations of Sect. 5.8.2, it can be seen that the observations only influence the covariance or information matrix (hence the posterior entropy), by changing the point at which the observation model is linearised (through changing the prior densities mean).

Thus, to remove this dependency, and the requirement to perform the expectation in (5.50), the nominal state about which the observation model is linearised will be given by the mean object state at time k

$${}_{n}\mathbf{x}_{o_j}^{k+l} = \bar{\mathbf{x}}_{o_j}^{k}, \quad \forall l \in \{1, \ldots, NM\}. \tag{5.65}$$

Hence, the posterior information matrix will be independent of the observation sequence $\tilde{\mathbf{z}}_{o_j}^{k+1:k+NM}$ and may be evaluated directly using

$$\mathbf{Y}_{o_j}^{|k+NM} = \mathbf{Y}_{o_j}^{|k} + \sum_{i=1}^{p}\sum_{l=1}^{NM}\mathbf{I}_i^{k+l}\bigl(\mathbf{v}_i^k\bigr) \tag{5.66}$$

where $\mathbf{I}_{i,o_j}^{k+l}(\mathbf{v}_i^k)$ is the observation information matrix and is given by

$$\mathbf{I}_{i,o_j}^{k+l}\bigl(\mathbf{v}_i^k\bigr) = \bigl(\mathbf{H}_{i,o_j}^{k+l}\bigr)^T\bigl(\mathbf{R}_{i,o_j}^{k+l}\bigr)^{-1}\mathbf{H}_{i,o_j}^{k+l}. \tag{5.67}$$

The posterior entropy of object o_j is now given by

$$\begin{aligned} J_{o_j}^{k}(\mathbf{v}^k) &= H_{P(\mathbf{x}_{o_j}|\tilde{\mathbf{Z}}^{k+NM};\tilde{\mathbf{S}}^{k+NM})} \\ &= -\frac{1}{2}\log\bigl((2\pi e)^{d_x}\bigl|\mathbf{Y}_{o_j}^{|k+NM}\bigr|\bigr) \end{aligned} \tag{5.68}$$

and the final team objective function, corresponding to the joint entropy of all objects, is

$$J^k\bigl(\mathbf{v}^k\bigr) = \sum_{j=1}^{m} J_{o_j}^{k}\bigl(\mathbf{v}^k\bigr). \tag{5.69}$$

5.8.3.1 Partial Separability

For each agent to evaluate the objective function, it requires the sum of the observation information matrices from all other agent and over all steps in the planning horizon. The actual observation models, $\mathbf{H}^k_{i,o_j}$ and $\mathbf{R}^k_{i,o_j}$, and the position of the agents $\mathbf{s}^k_i$ are irrelevant once this information is obtained.

- *Impact Space:* Due to this structure, an impact space can be defined that is the product set of the m copies of the vector space $\mathcal{M}^S_{2\times 2}$, which contains all symmetric 2×2 matrices

$$\mathcal{I} = \times^m_{j=1} \mathcal{M}^S_{2\times 2} \tag{5.70}$$

- *Impact Function:* The impact function for an agent i maps a decision onto an element of this space by summing the individual information matrices $\mathbf{I}^{k+l}_{i,o_j}(\mathbf{v}^k_i)$ for the entire planning horizon $l \in \{1, \dots, NM\}$ for each object $j \in \{1, \dots, m\}$ and, i.e.

$$\Upsilon_i\big(\mathbf{v}^k_i\big) = \left\{ \sum_{l=1}^{NM} \mathbf{I}^{k+l}_{i,o_j}\big(\mathbf{v}^k_i\big) : \forall j \in \{1, \dots, m\} \right\}. \tag{5.71}$$

- *Composition Operator:* This operator combines impacts from different agents. It is given by matrix addition and simply adds corresponding observation information matrices. Thus, if $\alpha^k_a = \Upsilon_a(\mathbf{v}^k_a)$ and $\alpha^k_b = \Upsilon_b(\mathbf{v}^k_b)$, the composition operator is given by

$$\alpha^k_a * \alpha^k_b = \left\{ \sum_{l=1}^{NM} \mathbf{I}^{k+l}_{a,o_j} + \sum_{l=1}^{NM} \mathbf{I}^{k+l}_{b,o_j} : \forall j \in \{1, \dots, m\} \right\}. \tag{5.72}$$

- *Generalised Objective Function:* This function evaluates the cost (expected posterior entropy) of the agents decisions directly from the combined system impact. Consider the combined impact $\alpha^k_T = \bigodot^p_{i=1} \Upsilon_i(\mathbf{v}^k_i)$ given as

$$\alpha^k_T = \left\{ \alpha^k_{T,o_j} : \forall j \in \{1, \dots, m\} \right\} \tag{5.73}$$

where $\alpha^k_{T,o_j} = \sum^p_{i=1} \sum^{NM}_{l=1} \mathbf{I}^{k+l}_{i,o_j}$. Now, the generalised objective function $\psi^k(\alpha^k_T)$ is given by

$$\psi^k\big(\alpha^k_T\big) = -\sum_{j=1}^{m} \frac{1}{2} \log\big((2\pi e)^{d_x} \big|\mathbf{Y}^{|k}_{o_j} + \alpha^k_{T,o_j}\big|\big). \tag{5.74}$$

5.8.4 Collaborative Control

With the above definition the multi-agent decision problem becomes for a given time k becomes

$$\mathbf{v}^{k*} = \arg\min_{\mathbf{v}^k \in \mathcal{V}} \psi^k\bigl(\alpha_T^k\bigr) \tag{5.75}$$

where $\alpha_T^k = \bigodot_{i=1}^{p} \Upsilon_i(\mathbf{v}_i^k)$, and can be solved using Algorithm 5.3. Once this is generated and implemented by the team, the problem is resolved after N_r time steps.

5.9 Results

5.9.1 Two Agents—Single Object

To demonstrate the workings of the distributed optimisation algorithm, a system comprising two agents observing a single stationary object is considered. The initial configuration is shown in Fig. 5.5. For this scenario each agent has to decide on an open loop control policy consisting of a single control parameter ($N = 1$) that defines its rate of turn over a planning horizon of 12 s.

The optimal joint plans are found by each agent executing Algorithm 5.3. Although this procedure is designed to allow asynchronous execution, it was executed synchronously for demonstration purposes. Agents communicated after every local iteration with an imposed communication delay corresponding to three iterations.

As each agent only has a bearing sensor, individually they have a very poor ability to localise the object. However, if they cooperate and observe the object from perpendicular directions they can greatly minimise the position uncertainty of the object. However, there is also a dependency on the range at which they observe the object. such that a smaller range will give a smaller uncertainty in the measured position.

These seemingly opposing objectives are capture by the single objective function defined in (5.69). As shown in Fig. 5.5, the optimal decisions cause the agents to move toward the object and separate such that a better triangulation angle is obtained.

Although this resulting behaviour is intuitive to the observer, each agent cannot reason about this sort of global behaviour. Each agent only knows about the other through the communication of abstract *impacts*, the position of the other agent and its planned trajectory is completely unknown.

Figure 5.6(a) displays the evolution of the agents decisions throughout the optimisation procedure. Although the communication delay causes a significant difference in the perceived trajectories through the decision space, the system still converges to the optimum. It is noted, each agent never knows what the actual decision of the other agent is, it only knows its impact, Fig. 5.6(a) simply plots the decision corresponding to this impact. Figure 5.6(b) plots the inter-agent coupling, as

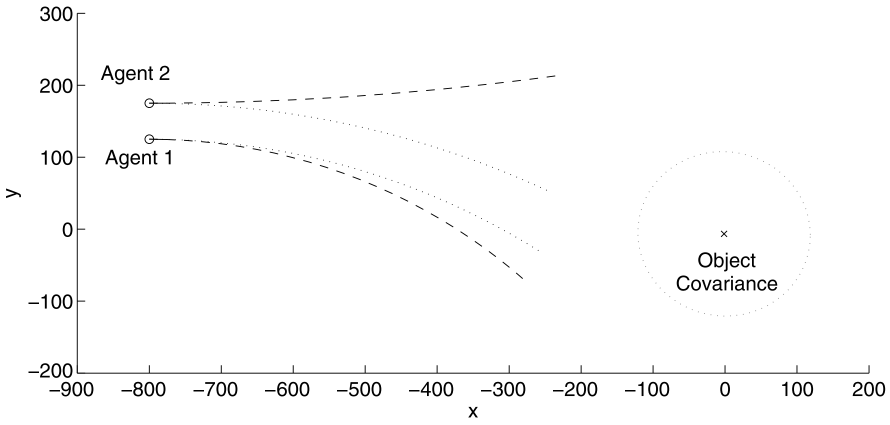


Fig. 5.5 The *dashed trajectories* corresponding to the jointly optimal plans, under the defined control parameterisation. The *dotted trajectories* represent the optimal solution to the corresponding single agent problem and was used to initialise the negotiated solution. The prior probability density of the position of the object is given by a Gaussian with mean given by the cross ($\times$) and variance by the *dotted circle*

approximated by both agents. The curves have similar shapes, but are not identical. This occurs because they are measuring the curvature of the objective function at different points in the decision space.

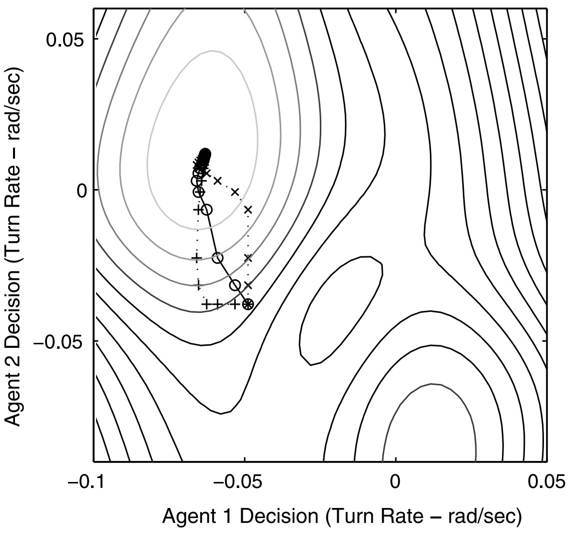
5.9.2 Nine Agents—Eighteen Objects

This scenario consists of 9 agents cooperatively localising 18 objects. The agents start from the left side of Fig. 5.7(a), in an arrow formation. The agents take observations at a rate of 2 Hz and plans a trajectory 16 s into the future (corresponding to an 800 m path). The trajectory is parameterised by 4 variables defining the required turn rates for each quarter of the trajectory. This corresponds to $N = 4$, $M = 8$ and $\Delta t = 0.5$ s and results in a optimal planning problem consists of 36 parameters distributed over the 9 agents.

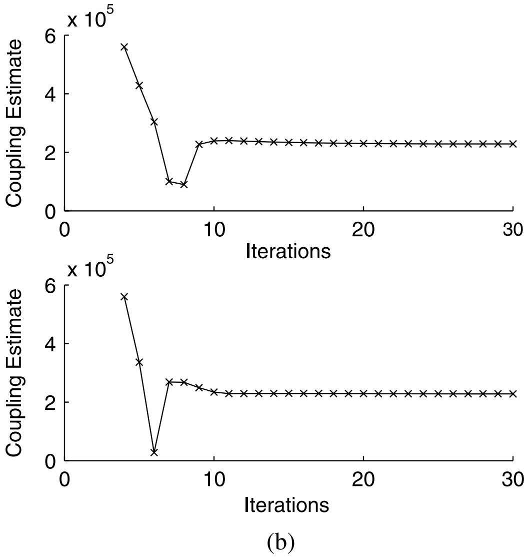
A rolling planning horizon is employed, requiring a new plan to be developed every second (i.e. $N_r = 2$). When generating the very first plan, the agents initialise their decisions using the locally optimal decision, however at all later stages the decisions are initialised using the solution from the previous decision problem.

The snapshots of the system are shown in Fig. 5.7(a)–(d). Figure 5.7(d) shows the final state of the system after all the objects have been sufficiently localised, however, for completeness the current optimal plans are also shown.

Figure 5.8 shows data of a typical run of the optimisation algorithm. It displays the estimated coupling terms, communication events and the evolution of the decision parameters from the perspective of a single agent. This data is for agent 6 for the very first decision problem (the results of which are shown in Fig. 5.7(a))



(a)



(b)

Fig. 5.6 (**a**) Evolution of agents decisions through the global decision space $\mathcal{V}$ during the optimisation procedure, overlaid with contours of the objective function. The true path $\mathbf{v} = [{}^1\mathbf{v}_1, {}^2\mathbf{v}_2]^T$ is shown with *circles* (∘), while the *pluses* (+) and *crosses* (×) represent the perceived path ${}^i\mathbf{v} = [{}^i\mathbf{v}_1, {}^i\mathbf{v}_2]^T$ for agents $i = 1, 2$ respectively. The difference is caused by the communication delays. (**b**) Coupling estimates ${}^1\hat{K}_{12}$ (*top*) and ${}^2\hat{K}_{21}$ (*bottom*) calculated by each agent 1 and 2 respectively

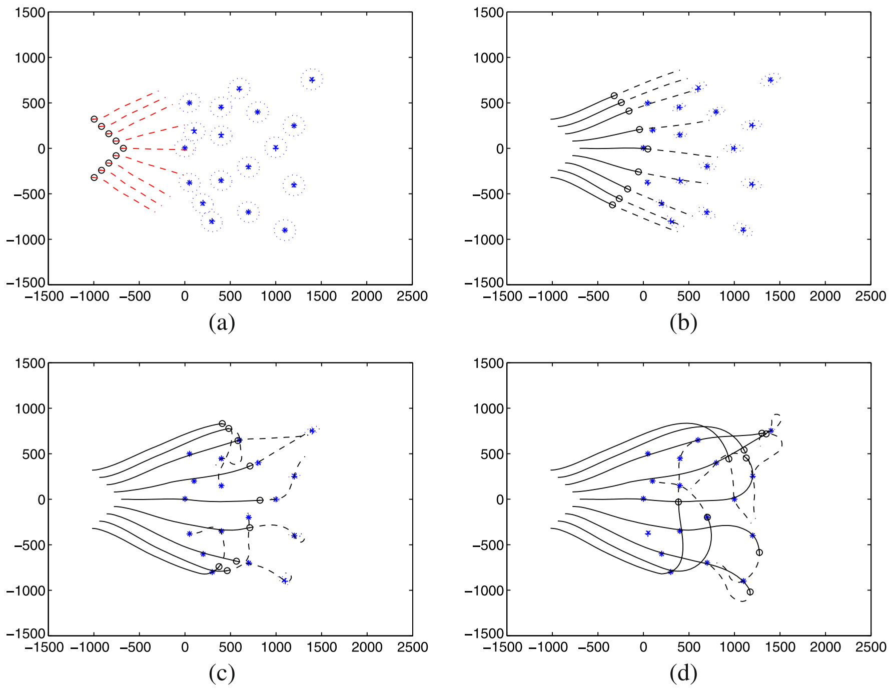


Fig. 5.7 Snap shots throughout the scenario at (**a**) $k = 0$, (**b**) $k = 30$, (**c**) $k = 60$, and (**d**) $k = 90$. Current agent positions are shown with a *circle* ($\circ$), and the optimal future planned trajectory with a *dotted line*. The current probability density of the location of each object is represented by its mean ($\times$) and covariance (*dotted circle*)

and demonstrates the relation between coupling (top graph) and communication frequency (middle graph). The agent communicates at a high frequency to agent 5, which it is coupled to the most and at a much lower frequency to other agents (especially agent 1) where the inter-agent coupling is smaller.

Figure 5.9 displays the inter-agent coupling for the whole system for each snap shot in Fig. 5.7. The ith row of each matrix represents the average of ${}^{i}\hat{K}_{ij}$ over the all iterations of Algorithm 1 for each other agent j. As expected, the matrix is reasonably symmetric (the coupling terms correspond to cross derivatives, which by definition are symmetric) and shows a large amount of structure. The matrix in Fig. 5.9(a) displays the intuitive result that agents close to each other are highly coupled (due to the diagonal structure). However, agent separation is not directly important, the coupling loosely measures the sensitive in the information content of one agents future observations on the decisions (and observations) of another. This is demonstrated in the last matrix corresponding to Fig. 5.7(d). At this time in the mission all the objects are well localised and for the agents to gather more information (and reduce the uncertainty) about the positions of the objects, the agents must travel very close to them. Thus, only agents with planned trajectories passing by a common object are coupled, e.g. agents 8 and 9; and agents 3 and 4.

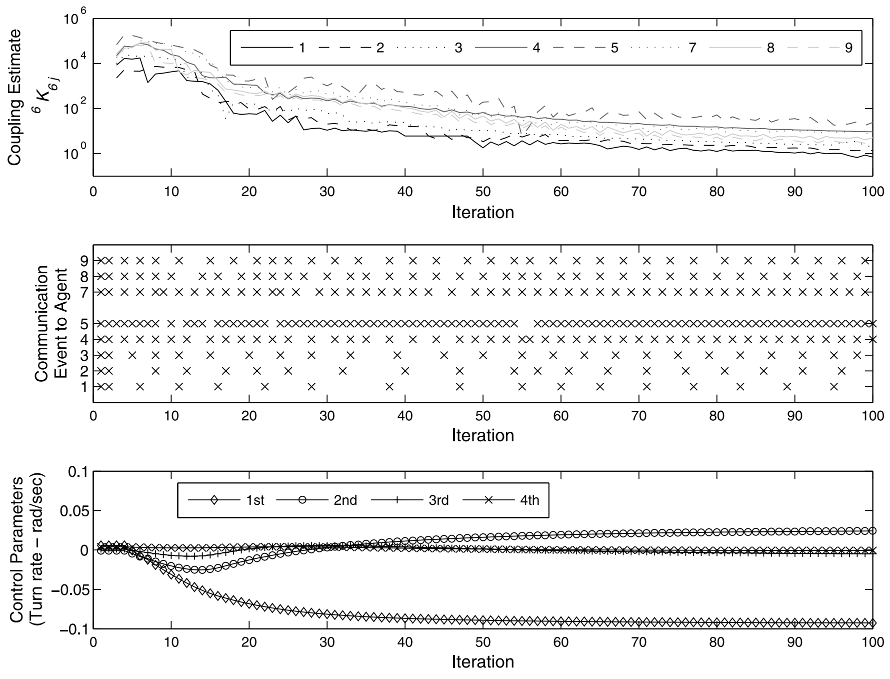


Fig. 5.8 Typical data during a single run of the distributed optimisation algorithm. This data corresponds to agent 6 for $k = 0$, corresponding to Fig. 5.7(a). *Top*: Estimated coupling terms ${}^6\hat{K}_{6j}$ for $j \in \{1, 2, 3, 4, 5, 7, 8, 9\}$. *Middle*: Each *cross* indicates the time a message to each specific agent in the system (the frequency of these events are determined by the coupling metric, according to (5.25)). *Bottom*: Evolution of the agents decision parameters (corresponding to the agents rate of turn during the planning horizon)

This coupling metric captures how the underlying agent models interact through the systems objective function, and in turn defines which agents should communicate and how often.

5.10 Discussion and Future Work

This chapter has approached the problem of multi-agent decision making and planning using the tools of asynchronous distributed optimisation. This analytical approach lead to the definition of a coupling metric that intuitively links the rate at which an agent may refine its local decision to its inter-agent communication frequency and transmission delays. The coupling is determined by the cross derivatives of the objective function and captures how the underlying agent models interact with the definition of the systems goals.

This decentralised negotiation algorithm was used to control a simulated multi-agent system involving multiple mobile robots undertaking an object localisation

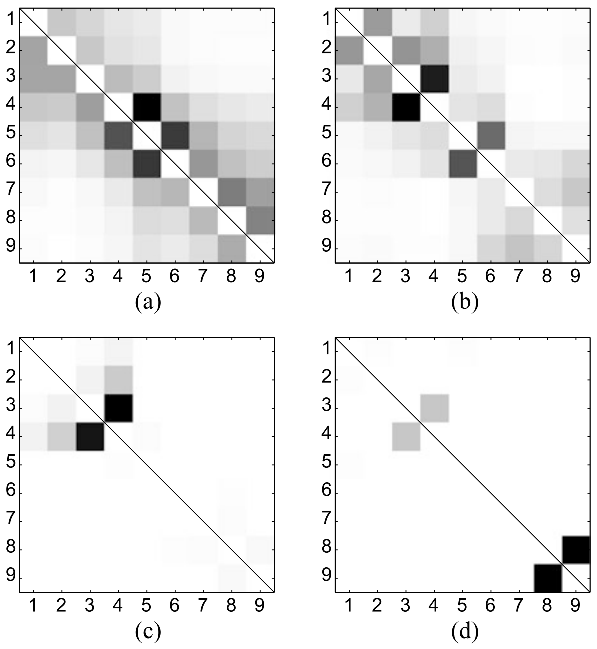


Fig. 5.9 Coupling matrix for the initial decision problem. The checker board type appearance (especially the two minor diagonals) represent that generally agents closer to each other are more strongly coupled (see Fig. 5.7(a))

task. This example demonstrated that agent are only required to communicate often to other agent that it is coupled to.

5.10.1 Toward Network Design

The algorithms presented in this chapter requires each agent to communicate to every other (coupled) agent. For simplicity this work assumed that lower level structure of the communications network was fixed and could not be controlled. The ability to link the performance of the optimisation algorithm with the structure of the communication system, via the step size and convergence speed has significant potential that has not been explored here. It is envisaged that the communication system design problem can be formulated as a higher level optimisation problem, which will enable communication networks and routing algorithms to be selected that maximises the convergence speed of the lower level distributed decision making algorithm. For this system level design problem to be fully defined, further analysis will be needed to extend the process outlined in Sect. 5.3.3 to enable an accurate link between the convergence speed and the properties of the communications system.

Acknowledgements This work is partly supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government; and by CSIRO Industrial Physics.

Appendix

Lemma 1 *For all scalars a and b the following holds*

$$ab \le \frac{1}{2}\left(a^2 + b^2\right), \tag{5.76}$$

which can easily be proved from the identity $(a-b)^2 \ge 0$.

Lemma 2 *For any vector* $\boldsymbol{\mu}_i \in \Re^{d_i}$

$$\left(\boldsymbol{\nu}_i - \operatorname{Proj}_{\mathcal{V}_i}^{\mathbf{A}_i}(\boldsymbol{\mu}_i)\right)^T \mathbf{A}_i \left(\boldsymbol{\mu}_i - \operatorname{Proj}_{\mathcal{V}_i}^{\mathbf{A}_i}(\boldsymbol{\mu}_i)\right) \le 0, \quad \textit{for all } \boldsymbol{\nu}_i \in \mathcal{V}_i. \tag{5.77}$$

Proof This can be proved by considering the definition of the projection (5.8) with the optimality condition (5.2) and noting

$$\nabla_{\boldsymbol{\nu}_i}\left((\boldsymbol{\nu}_i - \boldsymbol{\mu}_i)^T \mathbf{A}_i (\boldsymbol{\nu}_i - \boldsymbol{\mu}_i)\right) = 2\mathbf{A}_i(\boldsymbol{\nu}_i - \boldsymbol{\mu}_i). \qquad \square$$

For ease of notation, the following abbreviations will be made

$$J(t) \triangleq J\left({}^G\mathbf{v}(t)\right),$$
$$J_i(t) \triangleq \nabla_i J\left({}^G\mathbf{v}(t)\right),$$
$${}^iJ_i(t) \triangleq \nabla_i J\left({}^i\mathbf{v}(t)\right),$$

where ${}^G\mathbf{v}(t) = [{}^1\mathbf{v}_1(t), \ldots, {}^p\mathbf{v}_p(t)]$ is the global decision vector, constructed using the latest decision from each agent.

Define the actual update direction of the ith agent, for all $t \in T_i^U$, as

$$\mathbf{s}_i(t) = \frac{1}{\gamma_i}\left(\operatorname{Proj}_{\mathcal{V}_i}^{\mathbf{A}_i(t)}\left({}^i\mathbf{v}_i(t) - \gamma_i \mathbf{A}_i(t)^{-1}\,{}^iJ_i(t)\right) - {}^i\mathbf{v}_i(t)\right) \tag{5.78}$$

and for $t \notin T_i^U$ as $\mathbf{s}_i(t) = 0$. Thus, the update equation (5.6) can be written as

$${}^i\mathbf{v}_i(t+1) = {}^i\mathbf{v}_i(t) + \gamma_i \mathbf{s}_i(t), \quad \text{for all } t. \tag{5.79}$$

Lemma 3 *For all i and t*

$$\mathbf{s}_i(t)^{T\,i}J_i(t) \le -a_i \left\|\mathbf{s}_i(t)\right\|^2. \tag{5.80}$$

Proof This can be proved by considering the definition of $\mathbf{s}_i(t)$ and substituting into (5.77), $\mathbf{A}_i \triangleq \mathbf{A}_i(t)$, $\boldsymbol{\nu}_i \triangleq {}^i\mathbf{v}_i(t)$ and $\boldsymbol{\mu}_i \triangleq {}^i\mathbf{v}_i(t) - \gamma_i \mathbf{A}_i(t)^{-1\,i}J_i(t)$ and noting $\mathbf{s}_i(t)^T\mathbf{A}_i(t)\mathbf{s}_i(t) \ge a_i\|\mathbf{s}_i(t)\|^2$ where a_i is defined in (5.19). $\square$

Proof of Theorem 2 Consider the 2nd order Taylor expansion of $J({}^G\mathbf{v}(t+1))$ about ${}^G\mathbf{v}(t)$ and using Assumption 7

$$J(t+1) \leq J(t) + \sum_{i=1}^{p} \gamma_i \mathbf{s}_i(t)^T J_i(t) + \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p} \left\| \gamma_i \mathbf{s}_i(t) \right\| K_{ij} \left\| \gamma_j \mathbf{s}_j(t) \right\|. \quad (5.81)$$

The third term of (5.81) can be bounded from above using (5.76)

$$\begin{aligned} &\frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p} \left\| \gamma_i \mathbf{s}_i(t) \right\| K_{ij} \left\| \gamma_j \mathbf{s}_j(t) \right\| \\ &\quad \leq \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p} K_{ij} \frac{1}{2}\left(\gamma_i^2 \left\| \mathbf{s}_i(t) \right\|^2 + \gamma_j^2 \left\| \mathbf{s}_j(t) \right\|^2\right) \\ &\quad = \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p} K_{ij} \gamma_i^2 \left\| \mathbf{s}_i(t) \right\|^2. \end{aligned} \quad (5.82)$$

Examining the second term of (5.81) and using (5.80)

$$\sum_{i=1}^{p} \gamma_i \mathbf{s}_i(t)^T J_i(t) \leq -\sum_{i=1}^{p} \gamma_i a_i \left\| \mathbf{s}_i(t) \right\|^2 + \sum_{i=1}^{p} \gamma_i \mathbf{s}_i(t)^T \left(J_i(t) - {}^i J_i(t) \right). \quad (5.83)$$

The final term in (5.83) can be further bounded using (5.17) and (5.76) and noting ${}^G\mathbf{v}(t) = [{}^1\mathbf{v}_1(t), \ldots, {}^p\mathbf{v}_p(t)]$

$$\begin{aligned} &\sum_{i=1}^{p} \gamma_i \mathbf{s}_i(t)^T \left(J_i(t) - {}^i J_i(t) \right) \\ &\quad \leq \sum_{i=1}^{p} \gamma_i \left\| \mathbf{s}_i(t) \right\| \sum_{j=1}^{p} K_{ij} \left\| {}^j\mathbf{v}_j(t) - {}^i\mathbf{v}_j(t) \right\| \\ &\quad \leq \sum_{i=1}^{p} \gamma_i \left\| \mathbf{s}_i(t) \right\| \sum_{j=1}^{p} K_{ij} \sum_{m=t-B_{ij}}^{t-1} \gamma_j \left\| \mathbf{s}_j(m) \right\| \\ &\quad \leq \frac{1}{2}\sum_{i=1}^{p}\sum_{j=1}^{p}\sum_{m=t-B_{ij}}^{t-1} K_{ij} \left(\gamma_i^2 \left\| \mathbf{s}_i(t) \right\|^2 + \gamma_j^2 \left\| \mathbf{s}_j(m) \right\|^2\right). \end{aligned} \quad (5.84)$$

Substituting (5.82), (5.83) and (5.84) into (5.81) and summing the inequalities from $t = 0$ to $t = n$

$$J(n+1) \leq J(0) - \sum_{t=0}^{n}\sum_{i=1}^{p} \gamma_i a_i \left\| \mathbf{s}_i(t) \right\|^2 + \frac{1}{2}\sum_{t=0}^{n}\sum_{i=1}^{p}\sum_{j=1}^{p} K_{ij} \gamma_i^2 \left\| \mathbf{s}_i(n) \right\|^2$$

$$+\frac{1}{2}\sum_{t=0}^{n}\sum_{i=1}^{p}\sum_{j=1}^{p}\sum_{m=t-B_{ij}}^{t-1} K_{ij}\left(\gamma_i^2\left\|\mathbf{s}_i(t)\right\|^2+\gamma_j^2\left\|\mathbf{s}_j(m)\right\|^2\right). \tag{5.85}$$

By expanding the summation over t, the last term can be bounded by

$$\frac{1}{2}\sum_{t=0}^{n}\sum_{i=1}^{p}\sum_{j=1}^{p}\sum_{m=t-B_{ij}}^{t-1} K_{ij}\left(\gamma_i^2\left\|\mathbf{s}_i(t)\right\|^2+\gamma_j^2\left\|\mathbf{s}_j(m)\right\|^2\right)$$
$$\leq \frac{1}{2}\sum_{t=0}^{n}\sum_{i=1}^{p}\sum_{j=1}^{p}\gamma_i^2\left\|\mathbf{s}_i(t)\right\|^2(B_{ij}K_{ij}+B_{ji}K_{ji}). \tag{5.86}$$

It is noted that this is a strict inequality for non-zero $\mathbf{s}_j(t)$ when $t\in\{n-B_{ij},\ldots,n\}$. Using $K_{ij}=K_{ji}$ and (5.86), inequality (5.85) can be written as

$$\sum_{i=1}^{p}E_i\sum_{t=0}^{n}\left\|\mathbf{s}_i(t)\right\|^2\leq J\big(\mathbf{v}(0)\big)-J\big(\mathbf{v}(n+1)\big), \tag{5.87}$$

where $E_i=\gamma_i(a_i-\frac{1}{2}\gamma_i\sum_{j=1}^{p}K_{ij}(1+B_{ij}+B_{ji}))$. Let

$$\begin{aligned}\Gamma_i&=\frac{2a_i}{\sum_{j=1}^{p}K_{ij}(1+B_{ij}+B_{ji})}\\&=\frac{2a_i}{K_{ii}+\sum_{j\neq i}K_{ij}(1+B_{ij}+B_{ji})},\end{aligned}$$

then for all $\gamma_i\in(0,\Gamma_i)$, $E_i>0$. Furthermore, the objective function J is bounded from below (Assumption 2b), and thus, allowing $n\to\infty$ and restricting $\gamma_i\in(0,\Gamma_i)$, (5.87) must imply

$$\sum_{t=0}^{\infty}\left\|\mathbf{s}_i(t)\right\|^2<\infty,\quad\text{for all } i,$$

which requires $\lim_{t\to\infty}\|\mathbf{s}_i(t)\|=0$. This, along with Assumptions 4 and 5b, requires each agents local copy of the team decision to converge to the same limit. Let this fixed point be $\mathbf{v}^*$

$$\lim_{t\to\infty}{}^i\mathbf{v}(t)=\mathbf{v}^*,\quad\text{for all } i.$$

For this fixed point, the following must hold

$$\mathbf{v}_i^*=\mathrm{Proj}_{\mathcal{V}_i}^{\mathbf{A}_i(t)}\left(\mathbf{v}_i^*-\gamma_i\mathbf{A}_i(t)^{-1}\nabla_i J\big(\mathbf{v}^*\big)\right),\quad\text{for all } i.$$

Using this and (5.77), the fixed point $\mathbf{v}^*$ obeys the optimality condition (5.3)

$$\big(\boldsymbol{\nu}_i-\mathbf{v}_i^*\big)^T\nabla_i J\big(\mathbf{v}^*\big)\geq 0,\quad\text{for all } \boldsymbol{\nu}_i\in\mathcal{V}_i\text{ and all } i.$$

This corresponds to the first order condition for $\mathbf{v}^*$ to be a minimum of J. If the stronger condition of Assumption 2b holds the objective function is convex), the limit point $\mathbf{v}^*$ is a global minimum. $\square$

References

Baudet, G. M. (1978). Asynchronous iterative methods for multiprocessors. *Journal of the ACM*, *25*(2), 226–244.

Bertsekas, D. P. (1999). *Nonlinear programming*. Belmont: Athena Scientific.

Bertsekas, D. P. (2005). *Dynamic programming and optimal control*. Belmont: Athena Scientific.

Bertsekas, D. P., & Tsitsiklis, J. N. (1989). *Parallel and distributed computation: numerical methods*. New York: Prentice-Hall.

Bertsekas, D. P., & Tsitsiklis, J. N. (1991). Some aspects of parallel and distributed iterative algorithms—a survey. *Automatica*, *27*(1), 3–21.

Bourgault, F., Furukawa, T., & Durrant-Whyte, H. F. (2004). Decentralized Bayesian negotiation for cooperative search. In *Proc. IEEE/RSJ int. conf. on intelligent robots and systems*.

Camponogara, E., & Talukdar, S. (2007). Distributed model predictive control: synchronous and asynchronous computation. *IEEE Transactions on Systems, Man and Cybernetics*, *37*(7), 732–745.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. New York: Wiley.

Furukawa, T., Dissanayake, G., & Durrant-Whyte, H. F. (2003). Time-optimal cooperative control of multiple robot vehicles. In *Proc. IEEE int. conf. on robotics and automation*.

Goh, C. J., & Teo, K. L. (1988). Control parametrization: a unified approach to optimal control problems with general constraints. *Automatica*, *24*(1), 3–18.

Grocholsky, B. (2002). *Information-theoretic control of multiple sensor platforms*. PhD thesis, Univ. of Sydney, Australia.

Grocholsky, B., Makarenko, A., Kaupp, T., & Durrant-Whyte, H. F. (2003). Scalable control of decentralised sensor platforms. In *Int. workshop on information processing in sensor networks*, Palo Alto, CA, USA.

Inalhan, G., Stipanovic, D., & Tomlin, C. (2002). Decentralized optimization, with application to multiple aircraft coordination. In *Proc. IEEE conf. on decision and control*.

Liggins, M. E. II, Chong, C.-Y., Kadar, I., Alford, M. G., Vannicola, V., & Thomopoulos, S. (1997). Distributed fusion architectures and algorithms for target tracking. In *Proceedings of the IEEE*, New York.

Manyika, J., & Durrant-Whyte, H. F. (1994). *Data fusion and sensor management: a decentralized information-theoretic approach*. New York: Ellis Horwood.

Mathews, G. M. (2008). *Asynchronous decision making for decentralised autonomous systems*. PhD thesis, The University of Sydney.

Mathews, G. M., Durrant-Whyte, H. F., & Prokopenko, M. (2006). Scalable decentralised decision making and optimisation in heterogeneous teams. In *Proc. int. conf. on multisensor fusion and integration for intelligent systems*.

Mathews, G., Durrant-Whyte, H., & Prokopenko, M. (2009). Decentralised decision making in heterogeneous teams using anonymous optimisation. *Robotics and Autonomous Systems*, *57*(3), 310–320.

Maybeck, P. S. (1979–1982). *Stochastic models, estimation and control*. New York: Academic Press.

Modi, P., Shen, W., Tambe, M., & Yokoo, M. (2005). ADOPT: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, *161*(1–2), 149–180.

Patriksson, M. (1997). Decomposition methods for differentiable optimization problems over Cartesian product sets. *Computational Optimization and Applications*, *9*(1), 5–42.

Raffard, R. L., Tomlin, C. J., & Boyd, S. P. (2004). Distributed optimization for cooperative agents: application to formation flight. In *Proc. IEEE conf. on decision and control*.

Tseng, P. (1991). On the rate of convergence of a partially asynchronous gradient projection algorithm. *SIAM Journal on Optimization*, *1*(4), 603–619.

Tsitsiklis, J. N., Bertsekas, D. P., & Athens, M. (1986). Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, *31*(9), 803–812.

Chapter 6
Learning Mutation Strategies for Evolution and Adaptation of a Simulated Snakebot

Ivan Tanev

6.1 Introduction

Wheelless, limbless snake-like robots (Snakebots) feature potential robustness characteristics beyond the capabilities of most wheeled and legged vehicles—ability to traverse terrain that would pose problems for traditional wheeled or legged robots, and insignificant performance degradation when partial damage is inflicted. Some useful features of Snakebots include smaller size of the cross-sectional areas, stability, ability to operate in difficult terrain, good traction, high redundancy, and complete sealing of the internal mechanisms (Dowling 1997; Hirose 1993).

Robots with these properties open up several critical applications in exploration, reconnaissance, medicine and inspection. However, compared to the wheeled and legged vehicles, Snakebots feature (i) more difficult control of locomotion gaits and (ii) inferior speed characteristics. In this work we intend to address the following challenge: how to automatically develop control sequences of Snakebot's actuators, which allow for achieving the fastest possible speed of locomotion.

In principle, the task of designing the code of a Snakebot could be formalized and the formal mathematical models could be incorporated into direct programmable control strategies. However, the eventual models would feature enormous complexity and such models are not recognized to have a known analytically exact optimal solution. The complexity of the model stems from the large number of degrees of freedom of the Snakebot, which cannot be treated independent of each other. The dynamic patterns of position, orientation, velocity vectors, and the points and instances of contact with the surface (and consequently—the vectors of resulting traction forces, which propel the Snakebot) of each of the morphological segments of the Snakebot have to be considered within the context of other segments. Furthermore, often the dynamic patterns of these parameters cannot be deterministically

I. Tanev (✉)
Department of Information Systems Design, Doshisha University, 1-3 Miyakodani, Tatara, Kyotanabe, Kyoto 610-0321, Japan
e-mail: itanev@mail.doshisha.ac.jp

inferred from the desired velocity characteristics of the locomotion of the Snakebot. Instead, the locomotion of the Snakebot is viewed as an emergent property at higher level of consideration of a complex hierarchical system, comprising many relatively simple entities (morphological segments). Higher-level properties of the system as a whole and the lower-level properties of entities comprising it cannot be induced from each other. Evolutionary approaches (Mahdavi and Bentley 2003; Takamura et al. 2000) are considered an efficient way to tackle such ill-posed problems due to their ability to find a near-optimal solution in a reasonable runtime.

As an instance of evolutionary algorithms, Genetic Algorithms (GA) differ from Genetic Programming (GP) mainly in the genotypic representation (i.e., the chromosome) of potential solutions (Goldberg 1989). Instead of representing the solution as a computer program (usually—a parse tree) featuring arbitrary structure and complexity as in GP, a GA employs a fixed-length linear chromosome. This difference implies a favorable computational advantage of the GA over GP for simple problems, because linear chromosomes are computationally more efficient to manipulate and interpret. For complex tasks however, such as evolution of locomotion gaits of a Snakebot the runtime overhead associated with manipulation of the genotype is negligible compared to the more significant overhead of the fitness evaluation of evolved (either simulated or real) artifacts. Moreover, an efficient GA (in terms of computational effort, or number of fitness evaluations) often requires incorporation of computationally heavy probabilistic learning models (Pelikan et al. 1999) aimed at discovering and maintaining the complex interrelations between variables in the genome. In addition, the fixed-length genome usually implies that the latter comprises various carefully encoded domain-dependent parameters of the solution with an *a priori* known structure and complexity. This might be a concern if no such knowledge is available in advance, but rather needs to be automatically and autonomously discovered by the evolving artifact. The latter is especially true when the artifact has to perform in unanticipated, uncertain environmental conditions or under its own (possibly degraded) mechanical abilities.

An example of the successful implementation of evolution (using a GA) and adaptation (via hierarchical, two-layered Q-learning) of locomotion gaits of real-world snake-like robot is demonstrated by Ito et al. (2003). Each gene in the linear chromosome represents the angle of the corresponding joint in the snake, and the number of genes is the same as the number of joints. This work demonstrates the efficiency of a canonical GA for the particular complexity of the task—evolution of two-dimensional gaits of a snake having five joints. The efficiency of the GA is adequate even without the need to consider either the scalability problem (the inflation of search space with the increase of the number of joints) or the linkage problem (the correlation between the genes in linear chromosomes). Several similar methods of adaptation combining evolutionary algorithms (either GA or GP for off-line evolution of a model of the artifact) and learning (for on-line adaptation of the real artifact) have been recently proposed (Kamio et al. 2003; Kimura et al. 2001). The learning component in these approaches is usually intended to tune, in an online mode, the solution obtained off-line via simulated evolution. Such approaches, as an on-line parametric optimization of solution via local search, are efficient when changes to the fitness landscape are assumed to be insignificant. However, adaptation to unanticipated, unknown or uncertain changes in fitness landscapes might require the discovery of completely new solutions, which often could not be achieved by parametric optimization of already existing solutions. We assume that GP alone, which is able to balance inherently the exploration (of completely new solutions) and exploitation (of previously obtained solutions) by maintaining a diverse population of solutions, offers good opportunities to discover these new solutions.

An inverse approach, based on the evolution (estimation) of the morphology of a potentially damaged robot given a controller (instead of evolving a controller given a morphology of the damaged robot) allows to evolve a damage hypothesis after failure and then to re-evolve a compensatory neural controller to restore the robots functionality (Bongard and Lipson 2004). Conversely, in our work we adhere to the conventional approaches of employing simulated evolution to develop the compensatory traits of a controller given the unanticipated changes of morphology due to partial damage of the Snakebot.

The *objectives* of our work are (i) to explore the feasibility of applying GP for efficient automatic design of the fastest possible locomotion of realistically simulated Snakebot and (ii) to investigate the adaptation of such locomotion to challenging environment and degraded abilities (due to partial damage) of simulated Snakebot. In Tanev and Ray (2005) we discussed the very feasibility of applying an evolutionary approach for automated development of locomotion gaits of Snakebots. Later we demonstrated the evolution of non-periodic postures of the Snakebot and verified the versatility of genetic programming for evolution and adaptation to environmental challenges and to damages (Tanev et al. 2005). In this work we discuss the biologically plausible (Caporale 2003; Kirschner and Gerhart 2005) non-random mutations implemented through learning mutation strategies (LMS) in GP. We are especially interested in the implications of LMS on the efficiency of evolution and adaptation of Snakebot.

Presented approach of incorporating LMS is implemented via learning probabilistic context-sensitive grammar (LPCSG), employed to express the preferable syntactical bias of mutation operation in GP. LPCSG is related to approaches of using a grammar to define the allowed syntax of the evolved artifacts. Examples of such approaches are the grammatical evolution (GE) (O'Neill and Ryan 2003), grammar-based genetic programming (Wong 2005) and grammar-based genetic algorithms (Antonisse 1991). The genotype, evolved via GE encodes the sequence of grammar rules, which should be applied during the simulated gene expression phase in order to generate the phenotype.

From another perspective, our work is also related to the incorporation of estimation of distribution algorithms (EDA) for biased mutations in evolutionary computations, mostly in GA (Pelikan et al. 1999). Motivated by the demonstrated advantages of both the GE and EDA in GA, we intend to merge both approaches in a way that allows for the biased mutation in GP (rather than GA, as in EDA) to be implemented via adjustable, learned preferences (rather than "hard coded" in the chromosome, as in GE) in choosing the appropriate rule from the set of alternative, potentially applicable grammar rules. Although a few grammar-based EDAs have been recently proposed (Bosman and de Jong 2004; Shan et al. 2004), in neither of these methods the incorporation of LPCSG in GP has been explored.

The remainder of the chapter is organized as follows. Section 6.2 emphasizes the main features of GP proposed for evolution of locomotion of the Snakebot. Section 6.3 introduces the proposed approach of incorporating LPCSG in GP. Section 6.4 presents the empirically obtained results of efficiency of evolution and adaptation of Snakebot to challenging environment and partial damage. Section 6.5 discusses an alternative, interactive mechanism of learning the mutation strategies. Section 6.5 draws a conclusion.

6.2 Genetic Programming for Design of Gaits of the Snakebot

6.2.1 Morphology of Snakebot

Snakebot is simulated as a set of identical spherical morphological segments ("vertebrae"), linked together via universal joints. All joints feature identical (finite) angle limits and each joint has two attached actuators ("muscles"). In the initial, standstill position of Snakebot the rotation axes of the actuators are oriented vertically (vertical actuator) and horizontally (horizontal actuator) and perform rotation of the joint in the horizontal and vertical planes respectively. Considering the representation of Snakebot, the task of designing the fastest locomotion can be rephrased as developing temporal patterns of desired turning angles of horizontal and vertical actuators of each segment, that result in fastest overall locomotion of Snakebot. The proposed representation of Snakebot as a homogeneous system comprising identical morphological segments is intended to significantly reduce the size of the search space of the GP. Moreover, because the size of the search space does not necessarily increase with the increase of the complexity of Snakebot (i.e. the number of morphological segment), the proposed approach allows achievement of favorable scalability characteristics of GP.

In the representation considered, we use a pyramid approximation of the Coulomb isotropic friction model. No anisotropic (directional) friction between the morphological segments and the surface is considered. Despite the anticipated ease of simulation and design of eventual morphological segments featuring anisotropic friction with the surface (using passive wheels (Hirose 1993; Ito et al. 2003) or "belly" scales), such an approach would have the following drawbacks:

(i) Wheels, attached to the morphological segments are mainly effective in two-dimensional locomotion gaits. However, neither the fastest gaits in unconstrained environments nor the adaptive gaits in challenging environments (narrow tunnels, obstacles etc.) are necessarily two-dimensional. In three-dimensional locomotion gaits the orientation (the pitch, roll and yaw angles) of morphological segments at the instant of contact with the surface is arbitrary, which renders the design of effective wheels for such locomotion gaits a non-trivial engineering task.

(ii) Wheels may compromise the potential robustness characteristics of the Snakebot because they can be trapped easily in the challenging environments (rugged terrain, obstacles, etc.).
(iii) Wheels potentially reduce the application areas of the Snakebot because their engineering design implies lack of complete sealing of all mechanisms.
(iv) Belly scales would not promote any anisotropic friction if the Snakebot operates on a smooth, flat, clean and/or loose surface. Therefore the generality of locomotion gaits and their robustness with respect to various environmental conditions would be compromised.
(v) Belly scales are efficiently utilized as a source of anisotropic friction in some locomotion gaits of natural snakes. However, these gaits usually require an involvement of large amount of complex muscles located immediately under the skin of the snake. These muscles lift the scales off the ground, angle them forward, and then push them back against the surface. In the Snakebot, implementing actuators which mimic such muscles of natural snakes would be too expensive and thus infeasible from an engineering point of view.

6.2.2 GP

6.2.2.1 Algorithmic Paradigm

GP (Koza 1992) is a domain-independent problem-solving approach in which a population of computer programs (individuals' genotypes) is evolved to solve problems. The simulated evolution in GP is based on the Darwinian principle of reproduction and survival of the fittest. The fitness of each individual is based on the quality with which the phenotype of the simulated individual is performing in a given environment.

6.2.2.2 Set of Functions and Terminals

In applying GP to evolution of Snakebot, the genotype is associated with two algebraic expressions, which represent the temporal patterns of desired turning angles of both the horizontal and vertical actuators of each morphological segment. Because locomotion gaits, by definition, are periodical, we include the periodic functions `sin` and `cos` in the function set of GP in addition to the basic algebraic functions. Terminal symbols include the variables `time`, `index` of the segment of Snakebot, and two constants: `Pi`, and `random` constant within the range [0, 2]. The main parameters of the GP are shown in Table 6.1.

The introduction of variable `time` reflects our objective to develop *temporal* patterns of turning angles of actuators. The choice of the trigonometric functions `sin` and `cos` reflects our intention to verify the hypothesis, as first expressed by Miturich in 1920s (Andrusenko 2001), that undulate motion mechanisms could yield efficient gaits of snake-like artifacts operating in air, land, or water.

Table 6.1 Main parameters of GP

Category	Value
Function set	{sin, cos, nop, +, −, ∗, /}
Terminal set	{time, segment_ID, Pi, random constant, ADF}
Population size	200 individuals
Selection	Binary tournament, selection ratio 0.1, reproduction ratio 0.9
Elitism	Best 4 individuals
Mutation	Random subtree mutation, ratio 0.01
Fitness	Velocity of simulated Snakebot during the trial
Trial interval	180 time steps, each time step account for 50 ms of "real" time
Termination criterion	(Fitness > 100) or (Generations > 40)

From another perspective, the introducing the trigonometric functions we would attempt to mimic (at functional, rather than neurological level) the central pattern generator (CPG) in the central nervous system. The CPG, usually located in the ganglia or spinal cord of animals, is believed to be necessary and sufficient for the generation of rhythmic patterns of activities (Levitan and Kaczmarek 2002). CPG for robot control typically comprises coupled neural controllers, which generate (without the need of external feedback) the motion pattern of actuators in the respective morphological segments of the artifact. The approach of employing CPG for developing the locomotion gaits of the Snakebot would be based on an iterative process (e.g., employing the machine learning and/or evolutionary computations paradigms) of tuning the main parameters of CPG including, for example, the common single frequency across the coupled oscillators, the fixed phase-relationship between the oscillators, and the amplitude of each of oscillations. The proposed approach of applying GP for evolution of locomotion gaits of Snakebot shares some of the features of CPG-based approaches such as the open-loop, sensorless control scheme and the incorporation of coupled oscillators. Conversely to the CPG-based approaches however, the proposed method incorporates too little domain-specific knowledge about the task.

The rationale of employing automatically defined functions (ADF) is based on empirical observation that although the straightforward, independent encoding of the desired turning angles of both horizontal and vertical actuators allows GP to adequately explore the huge search space and ultimately, to discover the areas which correspond to fast locomotion gaits in solution space, the evolvability of such encoding is relatively poor. We discovered that (i) the motion patterns of horizontal and vertical actuators of each segment in fast locomotion gaits are highly correlated (e.g., by frequency, direction, etc.) and that (ii) discovering and preserving such correlation by GP is associated with enormous computational effort. ADF, as a way of limiting the search space by introducing modularity and reuse of the evolved code is employed in our approach to allow GP to explicitly evolve the correlation between motion patterns of horizontal and vertical actuators in a form of shared fragments in algebraic expressions of desired turning angles of the actuators. Moreover, we observed that the best result is obtained by (i) allowing the use of ADF as a terminal symbol in algebraic expression of desired turning angle of vertical actuator only, and (ii) by evaluating the value of ADF by equalizing it to the value of the currently evaluated algebraic expression of the desired turning angle of the horizontal actuator.

6.2.2.3 Context-Free Grammar for Canonical GP

The context-free grammar (CFG) G, usually employed to define the allowed syntax of individuals in GP consists of (N, Σ, P, S) where N is a finite set of nonterminal symbols, Σ is a finite set of terminal symbols that is disjoint from N, S is a symbol in N that is indicated as the start symbol, and P is a set of production rules, where a rule is of the form

$$V \to w$$

where V is a non-terminal symbol and w is a string consisting of terminals and/or non-terminals. The term “context-free” comes from the feature that the variable V can always be replaced by w, in no matter what context it occurs. The set of non-terminal symbols of G of GP, is employed to develop the temporal patterns of desired turning angles of horizontal and vertical actuators of segments, that result in fastest overall locomotion of Snakebot, is defined as follows:

```
N = {GP, STM, STM1, STM2, VAR, CONST_x10, CONST_PI, OP1, OP2}
```

where `STM` is a generic algebraic statement, STM1—a generic unary (e.g., `sin`, `cos`, `nop`) algebraic statement, `STM2`—a generic binary (dyadic, e.g. $+$, $-$, $*$, and $/$) algebraic statement, `VAR`—a variable, `OP1`—an unary operation, `OP2`—a binary (dyadic) operation, `CONST_x10` is a random constant within the range $[0\ldots20]$, and `CONST_PI` equals either 3.1416 or 1.5708. The set of terminal symbols is defined as:

$$\Sigma = \{\texttt{sin}, \texttt{cos}, \texttt{nop}, +, -, *, /, \texttt{time}, \texttt{segment\_id}\}$$

where `sin`, `cos`, `nop`, $+$, $-$, $*$ and $/$ are terminals which specify the functions in the generic algebraic statements. The start symbol is GP, and the set of production rules expressed in Backus-Naur form (BNF) are as shown in Fig. 6.1. GP uses the defined production rules of G to create the initial population and to mutate genetic programs. In the canonical GP the production rules with multiple alternative right-hand sides (such as rules 2, 4, 6, 7 and 9, shown in Fig. 6.1) are usually chosen *randomly* during these operations.

6.2.2.4 Fitness Evaluation

The fitness function is based on the velocity of Snakebot, estimated from the distance, which the center of the mass of Snakebot travels during the trial. Fitness of

```
(1)          GP → STM
(2.1-2.5)    STM → STM1|STM2|VAR|CONST_x10|CONST_PI
(3)           STM1 → OP1 STM
(4.1-4.6)      OP1 → sin|cos|nop|-|sqr|sqrt
(5)           STM2 → OP2 STM STM
(6.1-6.4)      OP2 → +|-|*|/
(7.1-7.2)     VAR  → time|segment_id
(8)           CONST_x10 → 0..20
(9.1-9.2)     CONST_PI  → 3.1416|1.5708
```

Fig. 6.1 BNF of production rules of the context free grammar G of GP, employed for automatic design of locomotion gaits of Snakebot. The following abbreviations are used: `STM`—generic algebraic statement, `STM1`—unary algebraic statement, `STM2`—binary (dyadic) algebraic statement, `VAR`—variable, `OP1`—unary operation, and `OP2`—binary operation

100 (the one of termination criteria shown in Table 6.1) is equivalent to a velocity, which displaced Snakebot a distance equal to twice its length.

6.2.2.5 Representation of Genotype

Inspired by its flexibility, and the recently emerged widespread adoption of document object model (DOM) and extensible markup language (XML) (Bray et al. 2000), we represent the evolved genotypes of the Snakebot as DOM-parse trees featuring equivalent flat XML-text. Both (i) the calculation of the desired turning angles during fitness evaluation and (ii) the genetic operations are performed on DOM-parse trees via API of the off-the shelf DOM-parser.

6.2.2.6 Genetic Operations

Selection is a binary tournament. Crossover is defined in a strongly typed way in that only the DOM-nodes (and corresponding DOM-subtrees) of the same data type (i.e. labeled with the same tag) from parents can be swapped. The sub-tree mutation is allowed in strongly typed way in that a random node in genetic program is replaced by syntactically correct sub-tree. The mutation routine refers to the data type of currently altered node and applies the chosen rule from the set of applicable rewriting rules as defined in the grammar of GP. The selection of the grammar rule, which should be applied to the currently altered tree node during the mutation is *random* in the canonical implementation of GP; and *biased* in the proposed approach of applying LMS as shall be elaborated in the following Sect. 6.3.

6.2.2.7 Open Dynamic Engine

We have chosen Open Dynamics Engine (ODE) (Smith 2006) to provide a realistic simulation of physics in applying forces to phenotypic segments of the Snakebot. ODE is a free, industrial quality software library for simulating articulated rigid body dynamics. It is fast, flexible and robust, and it has built-in collision detection.

6.3 Incorporating LMS in GP

6.3.1 Learning Probabilistic Context-sensitive Grammar

The proposed approach is based on the idea of introducing bias in applying the most preferable rule from the grammar rules with multiple, alternative right-hand sides (RHS). We presume that the preferences of applying certain production rules depend on the surrounding grammatical context, defining which rules have been applied before. The probability distributions (PD) $p_1^i, p_2^i, \ldots, p_N^i$ for each $context_i$ for each grammar rule with multiple RHS are initially uniform, and then learned (tuned) incrementally at each generation from the subset of the best performing Snakebots. The learned PD is then used as a bias to steer the mutation of Snakebots.

In the proposed approach, the learning probabilistic context-sensitive grammar (LPCSG) G^* is proposed as a formal model describing such mutation. G^* is introduced as a set of the same attributes (N^*, Σ^*, P^*, S^*) as the CFG G defined in Sect. 2.2. The attributes N^*, Σ^*, and S^* are identical to the corresponding attributes N, Σ, and S of G. The set of production rules P^* of G^* are derived from P of G as follows:

(i) Production rules of P_S ($P_S \subset P$) of G which have a single right-hand side are defined in the same way in P^* as in P, and
(ii) Production rules in P_M ($P_M \subset P$) of G, which feature multiple right-hand side alternatives $V \rightarrow w_1|w_2|\ldots|w_N$ are re-defined for each instance i of the context as follows:

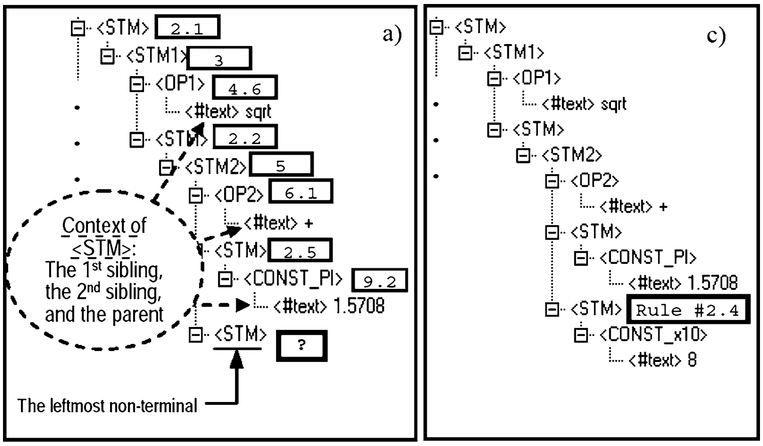
$$
\begin{aligned}
&context_i\, V \rightarrow context_i\ w_1 \quad (p_1^i)\\
&context_i\, V \rightarrow context_i\ w_2 \quad (p_2^i)\\
&\ldots\\
&context_i\, V \rightarrow context_i\ w_N \quad (p_N^i)
\end{aligned}
$$

where $p_1^i, p_2^i, \ldots, p_N^i$ ($\sum_1^N p_n^i = 1$) are the probabilities of applying each alternative rule with the left-hand side non-terminal V for the given $context_i$.

Applying the IF-THEN stimulus-response paradigm, which usually expresses the reactive behavioral strategies of intelligent entities in AI (e.g., software agents, robots, etc.) to such biased mutation operations in GP, and viewing the evolved genotype not only as an evolving, but also as a learning intelligent entity, the above considered sample rule of G^* could be modeled by the following behavioral IF-THEN statement:

```
if   Context_of_[$V$] is [$context_i$]  then
       Apply_Rules_With_Probabilities($p_1^i, p_2^i, \ldots, p_N^i$)
```

The LMS strategy in our approach comprises the dynamic set of IF-THEN rules created and tuned by parsing the syntax of the best performing Snakebots of the current generation. A sample of biased application of production rules of G^* according to the learned PD and the corresponding IF-THEN rule of LMS for the considered leftmost non-terminal and the context are shown in Fig. 6.2.



```
IF-THEN Rule of the LMS                                            b)
---------------------------------------------
IF (Context_of_[STM] is [1.57, +, sqrt])
THEN Apply_Rules_With_Probabilities [0.34, 0, 0.07, 0.43, 0.16]

#   Rule in LPCSG      Probability   For the Context
--- ----------------- ------------- ----------------
2.1 STM —> STM1       0.34          [1.57, +, sqrt]
2.2 STM —> STM2       0             [1.57, +, sqrt]
2.3 STM —> VAR        0.07          [1.57, +, sqrt]
2.4 STM —> CONST_x10  0.43          [1.57, +, sqrt]
2.5 STM —> CONST_PI   0.16          [1.57, +, sqrt]
```

Fig. 6.2 Sample of biased application of production rules of G^*: the current leftmost non-terminal, as shown in (**a**) is STM, which requires applying one of the production rules 2.1–2.5 (refer to Fig. 6.1). For the considered context (**a**), the LMS of applying rules 2.1–2.5 (**b**) suggests a highest probability for applying the production rule 2.4, yielding the genetic program as shown in (**c**)

6.3.2 GP Incorporating LMS

The principal steps of algorithm of GP incorporating LMS via LPCSG are shown in Fig. 6.3. As Fig. 6.3 illustrates, additional Steps 6 and 9 are introduced in the canonical algorithm of GP. The LMS is updated on Step 6, and the new offspring, created applying the proposed biased mutation via LPCSG on Step 9 are inserted into already reproduced—via canonical crossover (Step 7) and mutation (Step 8)—growing new population of Snakebots. The parameter K_{LMS} defines the ratio of the number of offspring $\#N_{LMS}$ created via biased mutation using LMS and the number of offspring $\#N_{CO}$ created via canonical crossover. K_{LMS} is dynamically tuned on Step 6 based on the stagnation counter C_S, which maintains the number of most recent generations without improvement of the fitness value. In our implementation, K_{LMS} is kept within the range [0, 5]. It is defined according to the following rule:

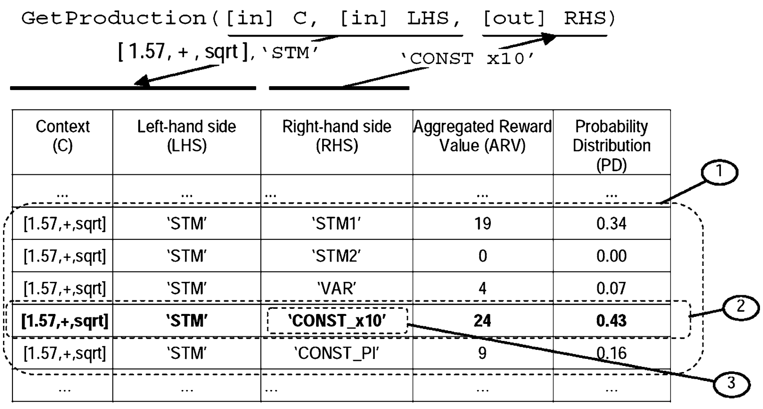
```
Step 0:  Creating Initial Population and Clearing PDD;
Step 1:  While (true) do begin
Step 2:      Evaluating Population;
Step 3:      Ranking Population;
Step 4:      if Termination Criteria then Go to Step 10;
Step 5:      Selecting the Mating Pool;
Step 6:      Updating LMS and $K_{LMS}$;
Step 7:      Creating #$N_{CO}$ offspring via canonical crossover;
Step 8:      Mutating current population via canonical mutation;
Step 9:      Creating #$N_{LMS}$ offspring via mutation of mating pool using LMS;
Step 10: end;
```

Fig. 6.3 Algorithm of GP incorporating LMS. Steps 6 and 9 are specific for the proposed approach. Steps 0, 2-5, 7 and 8 are common principal steps of canonical GP

$$K_{LMS} = 5 - \texttt{smaller\_of}(5, C_S)$$

Lower values of K_{LMS} in stagnated population (i.e., for $C_S > 0$) favor the reproduction via canonical random genetic operations over the reproduction using biased mutation via LMS. As we empirically investigated, the low values of K_{LMS} facilitate avoiding premature convergence by increasing the diversity of population and consequently, accelerating the escape from the (most likely) local optimal solutions, discovered by the steering bias of the current LMS. Conversely, replacing the usually random genetic operations of canonical GP with the proposed biased mutation when K_{LMS} is close to its maximum value (i.e., for $C_S = 0$) can be viewed as a mechanism for growing and preserving the proven beneficial building blocks in evolved solutions rather than destroying them by the usual random crossover and mutation

Updating (Fig. 6.3, Step 6) and applying LMS during the biased mutation (Fig. 6.3, Step 9) implies maintaining a table, which represents the set of learned IF-THEN rules. Each entry in the table stores the context, the left-hand side non-terminal, the list of right-hand side symbols, the aggregated reward values and the calculated probability of applying the given production rule for the given context. A new entry is added or the aggregated reward value of existing entry is updated by extracting the syntactic features of the best performing genetic programs (the mating pool) of the current generation. The outdated entries, added 4 or more generations before are deleted, keeping the total number of entries in the table between 300 and 500. The string of characters, comprising the right-hand side RHS of given production rule that should be applied to the current leftmost non-terminal (i.e. the corresponding left-hand symbol in production rule, LHS) for the given context C is obtained by the function GetProduction([in] C, [in] LHS, [out] RHS) which operates on LMS table as shown in Fig. 6.4.



```
GetProduction([in] C, [in] LHS, [out] RHS)
```

Context (C)	Left-hand side (LHS)	Right-hand side (RHS)	Aggregated Reward Value (ARV)	Probability Distribution (PD)
...	...	...	...	...
[1.57,+,sqrt]	'STM'	'STM1'	19	0.34
[1.57,+,sqrt]	'STM'	'STM2'	0	0.00
[1.57,+,sqrt]	'STM'	'VAR'	4	0.07
[1.57,+,sqrt]	**'STM'**	**'CONST_x10'**	**24**	**0.43**
[1.57,+,sqrt]	'STM'	'CONST_PI'	9	0.16
...	...	...	...	...

Fig. 6.4 Obtaining the most preferable right-hand side (RHS) of production rule of LPCSG that should be applied to the left-most non-terminal (i.e. left-hand symbol, LHS), and the context (C) according to a sample `IF-THEN rule` of the current LMS: (1) Selecting the set of entries associated with the entries featuring the given LHS and C, (2) Choosing an entry from the obtained result set with a probability, proportional to the learned PD, and (3) returning the RHS of the chosen production rule. The sample `IF-THEN` rule of the LMS, shown here is the same as depicted in Fig. 6.2

6.4 Results

This section discusses empirically obtained results verifying the effects of incorporating LMS on the efficiency of GP applied for the following two tasks: (i) *evolution* of the fastest possible locomotion gaits of Snakebot for various fitness conditions and (ii) *adaptation* of these locomotion gaits to challenging environment and degraded mechanical abilities of Snakebot. These tasks, considered as relevant for successful accomplishment of anticipated exploration, reconnaissance, medicine or inspection missions, feature different fitness landscapes. Therefore, the experiments discussed in this section are intended to verify the versatility and the scope of applicability of the proposed approach. In all of the cases considered, the fitness of Snakebot reflects the low-level objective (i.e. *what* is required to be achieved) of Snakebot in these missions, namely, to be able to move fast regardless of environmental challenges or degraded abilities. The experiments discussed illustrate the ability of the evolving Snakebot to learn *how* (e.g. by discovering beneficial locomotion traits) to accomplish the required objective without being explicitly taught about the means to do so. Such *know-how* acquired by Snakebot automatically and autonomously can be viewed as a demonstration of emergent intelligence (Angeline 1994), in that the task-specific knowledge of *how* to accomplish the task emerges in the Snakebot from the interaction of the problem solver and the fitness function.

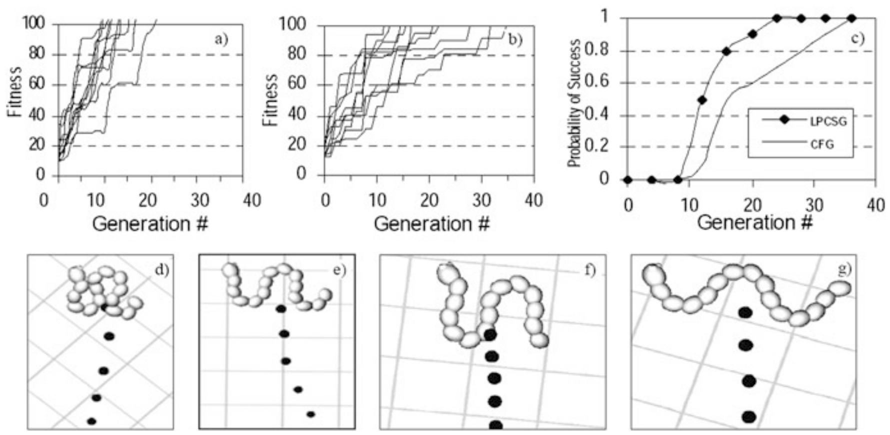


Fig. 6.5 Evolution of locomotion gaits for cases where fitness is measured as velocity in any direction: fitness convergence characteristics of 10 independent runs of GP with LMS (**a**), canonical GP (**b**), probability of success (**c**), and snapshots of sample evolved via GP with LMS best-of-run sidewinding Snakebots (**d**), (**e**), (**f**) and (**g**). The *dark trailing circles* in (**d**), (**e**), (**f**) and (**g**) depict the trajectory of the center of the mass of Snakebot

6.4.1 Evolution of Fastest Locomotion Gaits

Figure 6.5 shows the results of evolution of locomotion gaits for cases where fitness is measured as velocity in any direction. Despite the fact that fitness is unconstrained and measured as velocity in *any* direction, *sidewinding* locomotion (defined as locomotion predominantly perpendicular to the long axis of Snakebot) emerged in all 10 independent runs of GP, suggesting that it provides superior speed characteristics for considered morphology of Snakebot. As Fig. 6.5c illustrates, incorporating LMS in GP is associated with computational effort (required to achieve probability of success 0.9) of about 20 generations, which is about 1.6 times faster than canonical GP with CFG. Sample snapshots of evolved best-of-run sidewinding locomotion gaits are shown in Fig. 6.5d–g.

In order to verify the superiority of velocity characteristics of sidewinding we compared the fitness convergence characteristics of evolution in unconstrained environment for the following two cases: (i) unconstrained fitness measured as velocity in any direction (as discussed above and illustrated in Fig. 6.5), and (ii) fitness, measured as velocity in forward direction only. The results of evolution of forward (rectilinear) locomotion, shown in Fig. 6.6 indicate that non-sidewinding motion, compared to sidewinding, features much inferior velocity characteristics. The results also demonstrate that GP with LMS in average converges almost 4 times faster and to higher values than canonical GP. Snapshots taken during the motion of a sample evolved best-of-run sidewinding Snakebot are shown in Fig. 6.6c and 6.6d.

The results of evolution of rectilinear locomotion of simulated Snakebot confined in narrow "tunnel" are shown in Fig. 6.7. As the fitness convergence characteristics of 10 independent runs (Fig. 6.7a and Fig. 6.7b) illustrate, GP with LMS is almost

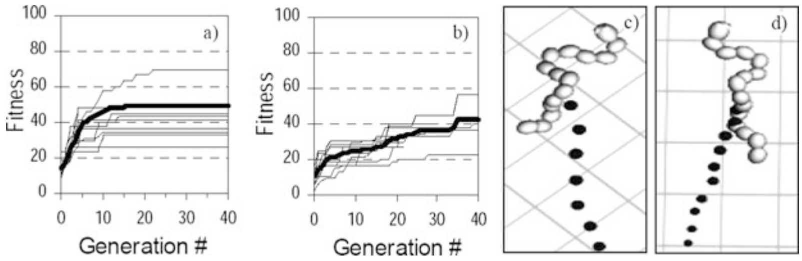


Fig. 6.6 Evolution of locomotion gaits for cases where fitness is measured as velocity in forward direction only. Fitness convergence characteristics of 10 independent runs of GP with LMS (**a**), canonical GP (**b**), and snapshots of sample evolved via GP with LMS best-of-run forward locomotion (**c** and **d**)

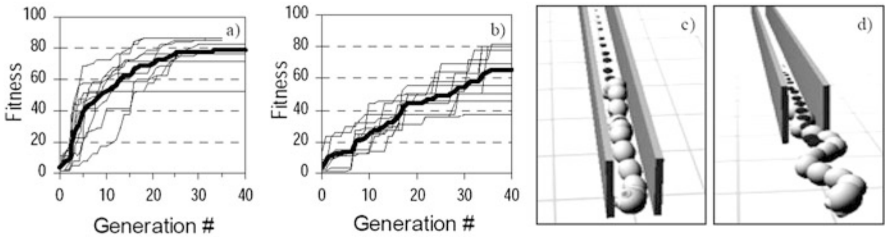


Fig. 6.7 Evolution of locomotion gaits of Snakebot confined in narrow "tunnel": fitness convergence characteristics of 10 independent runs of GP with LMS (**a**), canonical GP (**b**), and snapshots of sample evolved best-of-run gaits at the intermediate (**c**) and final stages of the trial (**d**)

twice faster than canonical GP. Compared to forward locomotion in unconstrained environment (Fig. 6.6), the velocity in this experiment is superior, and even comparable to the velocity of sidewinding (Fig. 6.5). This, seemingly anomalous phenomenon demonstrates a case of emergent intelligence—i.e. an ability of evolution to discover a way to utilize the walls of "tunnel" as (i) a source of extra grip and as (ii) an additional mechanical support for fast yet unbalanced locomotion gaits (e.g., vertical undulation) in an eventual unconstrained environment.

6.4.2 Adaptation to Unanticipated Challenging Terrain. Generality of Adapted Gaits

Adaptation in Nature is viewed as an ability of species to discover the best phenotypic (i.e. pertaining to biochemistry, morphology, physiology, and behavior) traits for survival in continuously changing fitness landscape. The adaptive phenotypic traits are result of beneficial genetic changes occurred during the course of evolution (phylogenesis) and/or phenotypic plasticity (ontogenesis—learning, polymorphism, polyphenism, immune response, adaptive metabolism, etc.) occurring during

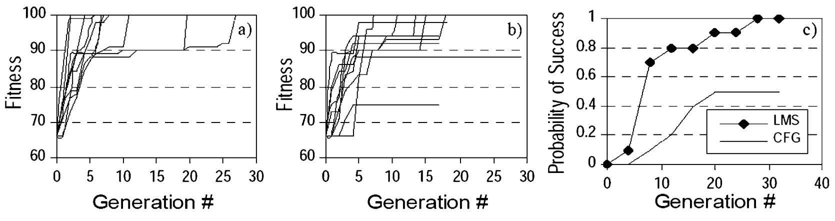


Fig. 6.8 Adaptation of sidewinding locomotion to challenging environment: fitness convergence characteristics of 10 independent runs of GP with LMS (**a**), canonical GP (**b**), and probability of success (**c**)

the lifetime of the individuals. In our approach we employ GP with LMS for adaptation of Snakebot to changes in the fitness landscape caused by (i) challenging environment and (ii) partial damage to 1, 2, 4 and 8 (out of 15) morphological segments. In all of the cases of adaptation, GP is initialized with a population comprising 20 best-of-run genetic programs, obtained from 10 independent runs of evolution of Snakebot in unconstrained environment, plus additional 180 randomly created individuals.

The challenging environment is modeled by the introduction of immobile obstacles comprising 40 small, randomly scattered boxes, a wall with height equal to the 0.5 diameters of the cross-section of Snakebot, and a flight of 3 stairs, each with height equal to the 0.33 diameters of the cross-section of Snakebot. The results of adaptation of Snakebot, shown in Fig. 6.8 demonstrate that the computational effort (required to reach fitness values of 100 with probability of success 0.9) of GP with LMS is about 20 generations. Conversely, only half of all runs of canonical GP achieve the targeted fitness value, implying that the corresponding probability of success converges to the value of 0.5. Snapshots illustrating the performance of Snakebot initially evolved in unconstrained environment, before and after the adaptation (via GP with LMS) to challenging environment are shown in Fig. 6.9.

The additional elevation of the body, required to faster negotiate the obstacles represents the emergent know-how in the adapting Snakebot. As Fig. 6.10 illustrates, the trajectory of the central segment around the center of the mass of sample adapted Snakebot (Fig. 6.10b) is twice higher than before the adaptation (Fig. 6.10a).

The generality of the evolved via GP with LMS robust sidewinding gaits is demonstrated by the ease with which Snakebot, evolved in known challenging terrain overcomes various types of unanticipated obstacles such as a pile of boxes, a burial under boxes, and small walls, as illustrated in Figs. 6.11, 6.12, and 6.13.

6.4.3 Adaptation to Partial Damage

The adaptation of sidewinding Snakebot to partial damage to 1, 2, 4 and 8 (out of 15) segments by gradually improving its velocity is shown in Fig. 6.14. Demonstrated results are averaged over 10 independent runs for each case of partial damage to 1,

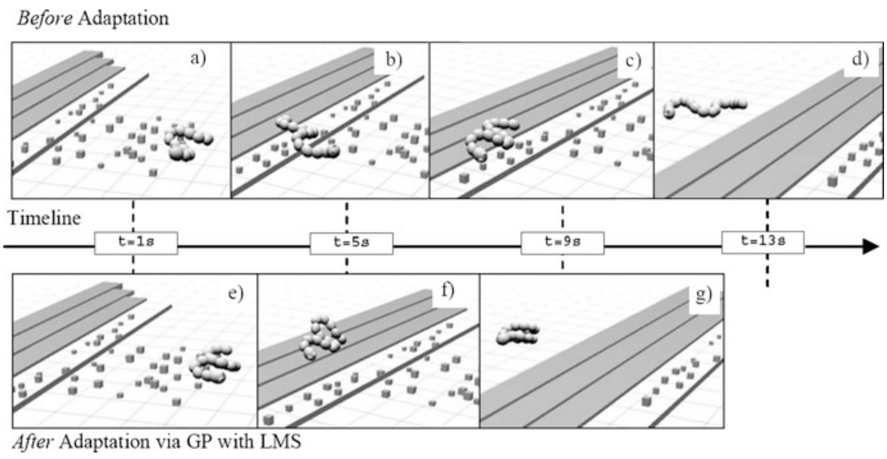


Fig. 6.9 Snapshots illustrating the sidewinding Snakebot, initially evolved in unconstrained environment, before the adaptation—initial (**a**), intermediate (**b** and **c**) and final stages of the trial (**d**), and after the adaptation to challenging environment via GP with LMS—initial (**e**), intermediate (**f**) and final stages of the trial (**g**)

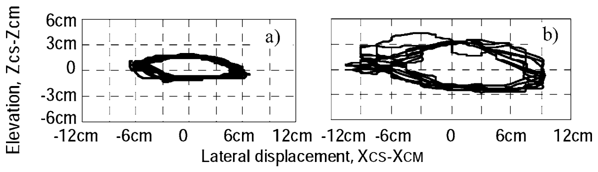


Fig. 6.10 Trajectory of the central segment (cs) around the center of mass (cm) of Snakebot for sample best-of-run sidewinding locomotion before (**a**) and after the adaptation (**b**) to challenging environment

2, 4 and 8 segments. The damaged segments are evenly distributed along the body of Snakebot. Damage inflicted to a particular segment implies a complete loss of functionality of both horizontal and vertical actuators of the corresponding joint.

As Fig. 6.14 depicts, Snakebot recovers completely from the damage to single segment attaining its previous velocity in 25 generations with canonical GP, and only in 7 generations with GP with LPCSG, resulting in a mean real-time of adaptation of a few hours of runtime on PC featuring an Intel® 3 GHz Pentium® 4 microprocessor and 2 GB RAM under Microsoft Windows XP OS. Snakebots recovers to average of 94 % (canonical GP) and 100 % (GP with LMS) of its previous velocity in the case where 2 segments are damaged. With 4 and 8 damaged segments the degree of recovery is 77 % (Canonical GP) and 92 % (GP with LMS), and 68 % (Canonical GP) and 72 % (GP with LMS) respectively. In all of the cases considered incorporating LMS contributes to faster adaptation of Snakebot, and in all cases the Snakebot recovers to higher values of velocity of locomotion. The snapshots of sidewinding Snakebot immediately after damage, and after having recovered from the damage of 1, 2 , 4 and 8 segments are shown in Fig. 6.15. The views of the

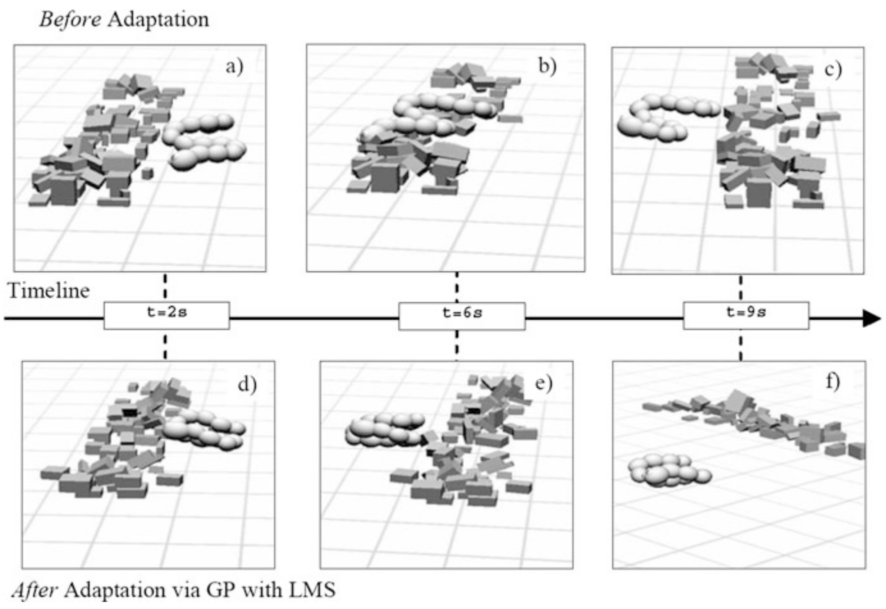


Fig. 6.11 Snapshots illustrating the generality of sidewinding Snakebot adapted to the known challenging environment as depicted in Fig. 6.9. Before the adaptation to the known challenging environment the Snakebot overcomes an unanticipated pile of boxes slower (**a**, **b** and **c**) than after the adaptation (**d**, **e**, and **f**) via GP with LMS

recovered Snakebot (Fig. 6.15b, 6.15d, 6.15f and 6.15h) reveal the emergent tendency of increasing the winding angle of locomotion. Moreover, the frontal view of the Snakebot before (Fig. 6.16a) and after the adaptation (Fig. 6.16b) to the damage of single segment demonstrates the additional elevation of the adapted Snakebot in a way analogous to the adaptation to the challenging environment as illustrated in Fig. 6.10.

6.5 Discussion

The efficiency of discussed approach of incorporating LS in GP depends on several factors such as the adequacy of the genetic representation of solution, the size of search space and the characteristics of the fitness landscape. Considering the latter issue, it is believed that the gradients towards the global optimums are a relevant prerequisite for an efficient evolution. These non-deceptive fitness gradients seemed to appear in the tasks of evolving and adapting the Snakebot as elaborated in the preceding sections. However, in some cases (as illustrated in Figs. 6.9, 6.11 and 6.12) the artifact might be temporarily trapped by obstacles in challenging environment. Consequently, the eventual evolutionary modifications to the locomotion patterns of the artifact might temporarily yield a negligible velocity of locomotion, and con-

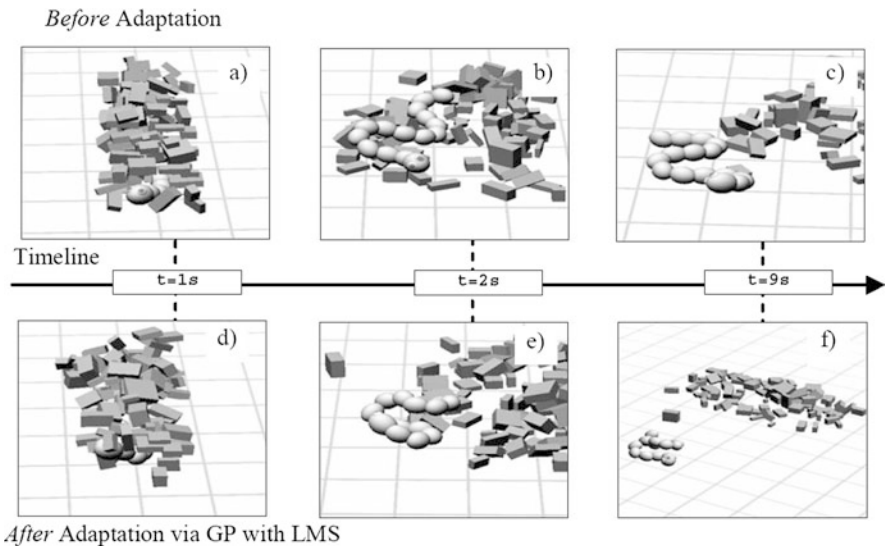


Fig. 6.12 Snapshots illustrating the generality of sidewinding Snakebot adapted to the known challenging environment as depicted in Fig. 6.9. Before the adaptation to the known challenging environment the Snakebot emerges from an unanticipated burial under pile of boxes slower (**a**, **b** and **c**) than after the adaptation (**d**, **e**, and **f**) via GP with LMS

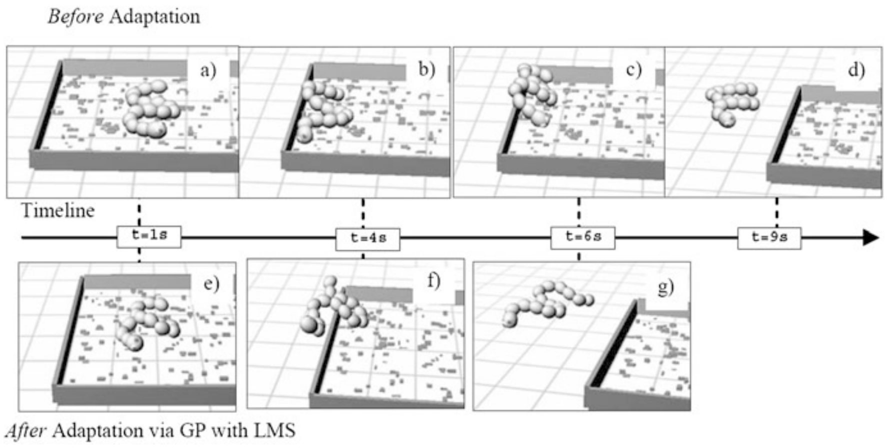


Fig. 6.13 Snapshots illustrating the generality of sidewinding Snakebot adapted to the known challenging environment as depicted in Fig. 6.9. Before the adaptation to the known challenging environment the Snakebot clears an unanticipated walls forming pen slower (**a**, **b**, **c** and **d**) than after the adaptation (**e**, **f**, and **g**). The walls are twice higher than in the known challenging terrain, and their height is equal to the diameter of the cross-section of Snakebot

sequently, negligible small fitness values, providing virtually no insight to the evolution about the promising areas in the explored search space. The corresponding

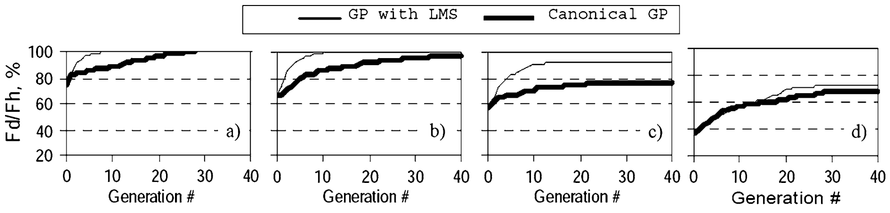


Fig. 6.14 Adaptation of Snakebot to damage of 1 (**a**), 2 (**b**), 4 (**c**) and 8 (**d**) segments. Fd is the best fitness in evolving population of damaged Snakebots, and Fh is the best fitness of 20 best-of-run healthy sidewinding Snakebots

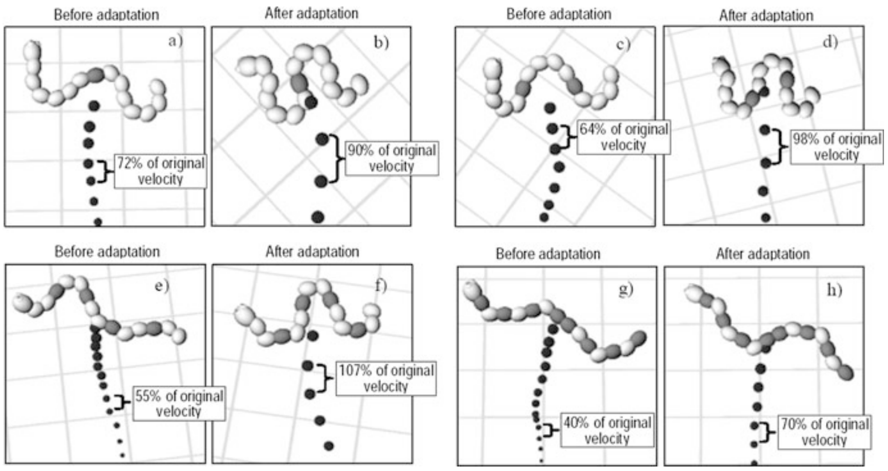


Fig. 6.15 Snapshots of the sidewinding Snakebot, immediately after damage to 1 (**a**), 2 (**c**), 4 (**e**), and 8 (**g**) segments, and after having recovered from the damage (**b**, **d**, **f**, and **h**) by adaptation via GP with LPCSG. The damaged segments of Snakebot are shown in a dark color

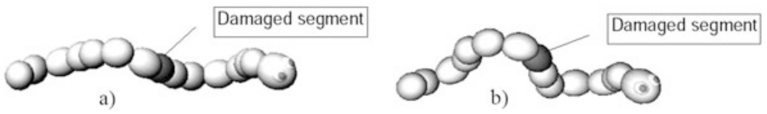


Fig. 6.16 The frontal view of the Snakebot before (**a**) and after the adaptation (**b**) to the damage of single segment. The corresponding views from above of the sidewinding Snakebot are shown in Fig. 6.15a and 6.15b respectively

fitness landscape would feature wide areas covered by low plateaus, which might render the simulated evolution to a poorly-guided or even a random search with relatively low computational efficiency. The information-driven evolutionary design, which introduces spatiotemporal measures of coordination of the modules that indirectly approximate the fitness function, is promising to be an interesting way to address such a problem (Prokopenko et al. 2006).

An alternative approach to address the issue of evolving an initially trapped Snakebot is to employ a derivation of the above discussed GP with LMS, in which the probabilities of applying the production rules are learned interactively from the parsed syntax of the Snakebots that, according to the human observer, are believed to exhibit behavioral features that are relevant for overcoming the obstacles (e.g., symmetrical shape, well-synchronized motion of segments, elevation of the body, etc.). Because these features might not be necessarily exhibited by the current best-performing Snakebots, they would provide the evolution with an additional insight about the promising areas in the fitness landscape. The preliminary results indicate that employing such an interactive feature-oriented GP via LMS can be associated with improved efficiency in that the locomotion gaits of Snakebot evolve faster and to higher velocities than those of the canonical GP (Tanev 2006).

6.6 Conclusion

In this work we propose an approach of incorporating LMS implemented via LPCSG in GP and verified it on the efficiency of evolution and adaptation of locomotion gaits of simulated Snakebot. We introduced a biased mutation in which the probabilities of applying each of particular production rules with multiple right-hand side alternatives in the LPCSG depend on the context, and these probabilities are "learned" from the aggregated reward values obtained from the evolved best-of-generation Snakebots. Empirically obtained results verify that employing LMS contributes to the improvement of computational effort of both (i) the evolution of the fastest possible locomotion gaits for various fitness conditions and (ii) adaptation of these locomotion gaits to challenging environment and degraded mechanical abilities of Snakebot.

Recent discoveries in molecular biology and genetics suggest that mutations do not happen randomly in Nature (Caporale 2003; Kirschner and Gerhart 2005) Instead, some fragments of DNA tend to repel the mutations away, while other fragments seem to attract them. It is assumed that the former fragments are related to the very basics of life, and therefore, any mutation within them can be potentially fatal to the species. We consider the ability of the Snakebot to move as an analogy of these very basics of life. Preserving the corresponding genotypic areas from mutations and focusing on genetic changes that facilitate the discovery of the beneficial phenotypic properties (e.g., additional elevation of the body and increased winding angle) of the already evolved fast locomotion gaits improves the efficiency of evolution and adaptation of the Snakebot to challenging environments and partial damages. The proposed approach contains no domain specifics and therefore can be incorporated into genetic programming for solving a wide range of problems from various problem domains.

Considering the situational awareness as a necessary condition for any intelligent autonomous artifact, in our future work would like to investigate the feasibility of incorporating sensors that allow the Snakebot to explicitly perceive the surrounding environment. We are especially interested in sensors that do not compromise the robustness characteristics of the Snakebot—such as, for example, Golgi's tendon receptors, incorporated inside the potentially completely sealed Snakebot.

Acknowledgements The author thanks Katsunori Shimohara, Thomas Ray and Andrzej Buller for their support of this work.

References

Andrusenko, Y. (2001). Russian culture navigator: Miturich-Khlebnikovs: art trade runs in the family. Available at http://www.vor.ru/culture/cultarch191_eng.html.

Angeline, P. J. (1994). Genetic programming and emergent intelligence. In K. E. Kinnear Jr. (Ed.), *Advances in genetic programming* (pp. 75–98). Cambridge: MIT Press.

Antonisse, J. (1991). A grammar-based genetic algorithm. In G. J. E. Rawlins (Ed.), *Foundations of the genetic algorithm workshop (FOGA)* (pp. 193–204). San Francisco: Morgan Kaufmann.

Bongard, J. C., & Lipson, H. (2004). Automated damage diagnosis and recovery for remote robotics. In *IEEE proceedings of the 2004 international conference on robotics and automation (ICRA 2004)* (pp. 3545–3550). New York: IEEE.

Bosman, P., & de Jong, E. (2004). Learning probabilistic tree grammars for genetic programming. In *Proceedings of the 8th international conference on parallel problem solving from nature PPSN-04* (pp. 192–201). London: Springer.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., & Maler, E. (2000). Extensible Markup Language (XML) 1.0, Second Edition, W3C Recommendation. Available at http://www.w3.org/TR/REC-xml/.

Caporale, L. H. (2003). *Darwin in the genome: molecular strategies in biological evolution*. New York: McGraw-Hill/Contemporary Books.

Dowling, K. (1997). *Limbless locomotion: learning to crawl with a snake robot* (Doctoral dissertation, Technical Report CMU-RI-TR-97-48). Robotics Institute, Carnegie Mellon University.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading: Addison-Wesley.

Hirose, S. (1993). *Biologically inspired robots: snake-like locomotors and manipulators*. Oxford: Oxford University Press.

Ito, K., Kamegawa, K., & Matsuno, F. (2003). Extended QDSEGA for controlling real robots—acquisition of locomotion patterns for snake-like robot. In *IEEE proceedings of IEEE international conference on robotics and automation (ICRA 2003)* (pp. 791–796). New York: IEEE.

Kamio, S., Mitsuhashi, H., & Iba, H. (2003). Integration of genetic programming and reinforcement learning for real robots. In E. Cantú-Paz, J. A. Foster, K. Deb, L. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. K. Standish, G. Kendall, S. W. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. A. Dowsland, N. Jonoska, & J. F. Miller (Eds.), *Proceedings of the genetic and evolutionary computations conference (GECCO 2003)* (pp. 470–482). Berlin: Springer.

Kimura, H., Yamashita, T., & Kobayashi, S. (2001). Reinforcement learning of walking behavior for a four-legged robot. In *Proceedings of 40th IEEE conference on decision and control*, Orlando, USA (pp. 411–416).

Kirschner, M. W., & Gerhart, J. C. (2005). *The plausibility of life: resolving Darwin's dilemma*. New Haven: Yale University Press.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*. Cambridge: MIT Press.

Levitan, I. B., & Kaczmarek, L. K. (2002). *The neuron: cell and molecular biology*. New York: Oxford University Press.

Mahdavi, S., & Bentley, P. J. (2003). Evolving motion of robots with muscles. In *Proceedings of EvoROB2003, the 2nd European workshop on evolutionary robotic (EuroGP 2003)*, Essex, UK (pp. 655–664).

O'Neill, M., & Ryan, C. (2003). *Grammatical evolution: evolutionary automatic programming in an arbitrary language*. Norwell: Kluwer.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: the Bayesian optimization algorithm. In *Proceedings of the genetic and evolutionary computation conference (GECCO-99)*, Orlando, USA (pp. 525–532).

Prokopenko, M., Gerasimov, V., & Tanev, I. (2006). Evolving spatiotemporal coordination in a modular robotic system. In S. Nolfi, G. Baldassarre, R. Calabretta, J. C. T. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, & D. Parisi (Eds.), *Lecture notes in computer science: Vol. 4095. From animals to animats 9: 9th international conference on the simulation of adaptive behavior (SAB 2006)*, Rome, Italy, 25–29 September 2006 (pp. 558–569). Berlin: Springer.

Shan, Y., McKay, R. I., & Baxter, R. (2004). Grammar model-based program evolution. In *Proceedings of the 2004 IEEE Congress on evolutionary computation*, 20–23 June, Portland, Oregon (pp. 478–485).

Smith, R. (2006). Open dynamics engine. Available at http://q12.org/od.

Takamura, S., Hornby, G. S., Yamamoto, T., Yokono, J., & Fujita, M. (2000). Evolution of dynamic gaits for a robot. In *Proceedings of the IEEE international conference on consumer electronics*, Los Angeles (pp. 192–193).

Tanev, I. (2006). Interactive learning of mutation strategies in genetic programming. In *Proceedings of the 5th joint symposium between Chonnam National University and Doshisha University*, Kwangju, Korea (pp. 83–87). Chonnam: Chonnam University Press.

Tanev, I., & Ray, T. (2005). Evolution of sidewinding locomotion of simulated limbless, wheelless robots. *Artificial Life and Robotics*, *9*, 117–122.

Tanev, I., Ray, T., & Buller, A. (2005). Automated evolutionary design, robustness and adaptation of sidewinding locomotion of simulated snake-like robot. *IEEE Transactions on Robotics*, *21*, 632–645.

Wong, M. L. (2005). Evolving recursive programs by using adaptive grammar based genetic programming. *Genetic Programming and Evolvable Machines 6*, 421–455.

Chapter 7
Self-Organization as Phase Transition in Decentralized Groups of Robots: A Study Based on Boltzmann Entropy

Gianluca Baldassarre

7.1 Introduction

An important goal of collective robotics (Dudek et al. 1996; Cao et al. 1997; Dorigo and Sahin 2004; Dorigo et al. 2004) is the development of multi-robot systems capable of accomplishing collective tasks without centralized coordination (Kube and Zhang 1993; Holland and Melhuish 1999; Ijspeert et al. 2001; Quinn et al. 2003). From an engineering point of view, decentralized multi-robot systems have several advantages vs. centralized ones in some tasks. For example, they are more robust with respect to the failure of some of their composing robots, do not require a control system or robot with sophisticated computational capabilities to manage the centralized control (Kube and Bonabeau 2000), have a high scalability with respect to the whole system's size (Baldassarre et al. 2006, 2007a), and tend to require simpler robots due to the low requirements of communication as they often can rely upon implicit coordination (Beckers et al. 1994; Trianni et al. 2006).

Decentralized coordination is usually based on self-organizing principles. Very often research on decentralized multi-robot systems makes a general claim on the presence of these principles underlying the success of the studied systems, but it does not conduct a detailed analysis of which specific principles are at work, nor it attempts to measure their effects in terms of the evolution of the system's organization in time or to analyze the robustness of its operation versus noise (e.g. see Holland and Melhuish 1999; Krieger et al. 2000; Kube and Bonabeau 2000; Quinn et al. 2003). This paper studies some of these issues in a multi-robot system presented in detail elsewhere (Baldassarre et al. 2003, 2006, 2007a, 2007b). This system is formed by robots that are physically connected and have to coordinate their direction of motion to explore an open arena without relying on a centralized coordination. The robots are controlled by an identical neural network whose weights

G. Baldassarre (✉)
Laboratory of Autonomous Robotics and Artificial Life, Istituto di Scienze e Tecnologie della Cognizione, Consiglio Nazionale delle Ricerche (LARAL-ISTC-CNR), Rome, Italy
e-mail: gianluca.baldassarre@istc.cnr.it

are evolved through a genetic algorithm. Through this algorithm the system develops the capacity to solve the task on the basis of self-organizing principles. The goal of this paper is to present some preliminary results that show how such principles lead the organization of the system, measured through a suitable index based on Boltzmann entropy, to arise in a quite abrupt way if the noise/signal ratio related to the signal that allows the robots to coordinate is slowly decreased. With this respect, the paper argues, on the basis of theoretical arguments and experimental evidence, that such sudden emergence of organization shares some properties with the phase transitions exhibited by some physical system studied in physics (Anderson 1997).

The rest of the paper is organized as follows. Section 7.2 presents a qualitative description of the mechanisms that are usually behind self-organization and introduces an index, based on Boltzmann entropy, that can be used to measure the *synchronic* level of order of a system composed of many dynamical parts. Section 7.3 illustrates the robots forming the multi-robot system considered here, the collective task tackled with it, the neural controller of the robots, and the genetic algorithm used to evolve it. Section 7.4 analyzes the behavior of the single robots developed by the genetic algorithm, and the effects it has at the collective level. Section 7.5 uses the entropy index to show that, when the noise/signal ratio related to the signal used by the robots to coordinate is slowly decreased, the level of order of the robotic system behaves as some global organization parameters observed in phase transitions of some physical systems. Finally, Sect. 7.6 draws the conclusions.

7.2 Mechanisms of Self-Organization, Phase Transitions, and Indexes to Measure the Organization Level of Collective Systems

Prokopenko et al. (2009) (see also Chap. 1 Prokopenko 2008) suggest that self-organization is characterized by three features: (a) it causes the parts forming a collective system to acquire global coordination; (b) this coordination is caused by the local interactions and information exchange between the parts composing the system and not by a centralized ordering mechanism; (c) the system passes from less organized states to more organized states. This section first tackles points (a) and (b) from a qualitative perspective, by presenting three basic mechanisms that usually underlie self-organization. Then it presents an index based on Boltzmann entropy that can be used to measure the level of order of a collective system at a given instant of time. This index can be used, as illustrated in the succeeding sections, to measure the level of organization of a multi-robot system under the action of self-organizing processes and hence to study point (c). Finally the section presents some theoretical arguments in favor of the hypothesis for which in some cases the dynamics of order exhibited by self-organizing multi-robot systems, as the one considered here, might have the features of phase transitions studied in physics. These arguments are supported by the experimental results presented in Sect. 7.5.

7.2.1 Qualitative Mechanisms of Self-Organization

Self-organizing processes regard systems composed of several and usually similar components. Self-organizing processes usually (always?) rely upon three basic principles (Camazine et al. 2001): (a) random fluctuations; (b) positive feedback; (c) negative feedback. These principles are now illustrated in detail.

The elements composing self-organizing systems are usually dynamic in the sense that they can assume one state among a certain number of possible states at each time step, and pass from state to state in time. Fully disorganized systems are those where each component passes from state to state in a random fashion. A typical feature of such systems is that the distribution of the components over the possible states tends to be uniform, that is *symmetric* (e.g., a school of fish randomly swimming in an aquarium tend to have a uniform distribution in the aquarium's water).

The symmetry of a collective system formed by components driven by random dynamics tends to be imperfect in the sense that it tends to have *random fluctuations* in time due to noise (e.g., there are some areas of the aquarium with a slightly higher density of fish). Now consider the possibility that each component of the system does not move (only) randomly, but tends to assume the states assumed by some other components of the system, that is it individually follows a *conformist rule* of the kind "I do what you do" (e.g., fish move to portions of space where other fish are located, so as to minimize the chance of being found alone by predators). In this condition, it might happen that some *random fluctuations are amplified*: indeed, the larger the number of components that assume a certain state vs. other states, the more intensely the remaining components will tend to imitate their state, so causing an exponential avalanche effect with a consequent *symmetry break* of the initial uniform distribution (e.g., the fish tend to cluster and form a whole school). The process that leads to this amplification is called *positive feedback*. In all real systems, the action of positive feedback tends to be counterbalanced by *negative feedback*. The latter might assume the form of an active process (e.g., the fish tend to cluster to avoid predators, but they also tend to keep at a certain minimal distance to avoid collisions) or a passive process (e.g., all fish have converged to the same zone in space) so the process of convergence stops. Starting from an initial uniform distribution, and after a first exponential convergence of the elements of the system to similar states due to positive feedback, negative feedback will start to slow down the process of convergence. With this respect, negative feedback tends to operate with a strength positively related to the number of elements that have already converged to the same states (e.g., to avoid collisions the fish "repulsion" behavior might be implemented with more vigor in space areas with higher densities of conspecifics as such densities correspond to smaller distances and higher chances of collision). For this reason negative feedback usually increases to levels that fully counterbalance the effect of positive feedback. At this point usually the system's overall state tends to reach equilibrium (e.g., the density of the fish school remains within a certain range; for examples of simulations of flocks, herds and schools of animals, see the seminal paper of Reynolds (1987), and the literature that followed it linked in the web page http://www.red3d.com/cwr/boids/).

7.2.2 An Index to Measure the Synchronous Level of Organization of Collective Systems Based on Boltzmann Entropy

The index used to measure the level of order of the group of robots studied here is based on Boltzmann entropy. Note that the index can be used to measure the level of organization of a collective system independently of the fact that such organization is the result of the action of self-organizing or of centralized coordination mechanisms. Boltzmann entropy has been proposed in mechanical statistics to measure the level of disorder that characterizes a system formed by a set of N gas molecules that occupy a given portion of space. This portion of space is divided into an arbitrary number C of cells each having a constant volume (in general the number of cells will influence the outcome of the application of the index, but, as we will see, the index can be suitably normalized to avoid this problem). The index is based on the assumption that the elements composing the system move randomly. This implies that at any time step an element can occupy any cell with a constant probability $1/C$ (the cell occupied by the element will constitute the element *state*). To give an example of this, consider the case of the robotic system studied here. This system is composed of $N = 40$ robots. Each robot can assume a given direction of motion ranging over a 1D closed space that ranges over $[0°, 360°]$ degrees. If this space is divided into $C = 8$ *cells* of constant size, at each time step the probability that an element occupies a given cell is equal to $1/8$.

The computation of the index is based on the so called microstates and macrostates of the system. A *microstate* of the system corresponds to all individual states of the elements in a given time step. For example, in a system with $N = 2$ and $C = 2$, the microstate is the vector (c_1, c_2) where c_n is the cell occupied by the element n. Note that the microstate is a vector and not a simple set, that is the order of the c_n states of the elements is relevant: this is a consequence of the fact that the *identity of the elements is assumed to be distinguishable*. So, for example, given a system with $N = 2$ and $C = 2$, the microstate where the first element occupies the first cell and the second element occupies the second cell is different from the microstate where the first element occupies the second cell and the second element occupies the first cell, even if in both cases the system has one element in the first cell and one element in the second cell. As each element can be in one of C possible different states, the number of different possible microstates is C^N.

Indicating with N_i the number of elements in cell i, a *macrostate* of the system is defined as the *distribution* $(N_1, N_2, \ldots, N_i, \ldots, N_C)$ of the elements over the cells, *without considering the identity* of the elements. An example of distribution for the system with $N = 2$ and $C = 2$ is $(0, 2)$, this meaning that there are zero elements in the first cell and two elements in the second cell. Each macrostate is (usually) composed of several possible microstates as the distribution of elements over the cells that correspond to it can be obtained in different ways. For example, in the $N = 2, C = 2$ system, the macrostate $(1, 1)$ with one element in each cell is composed of two microstates, that is $(1, 2)$ and $(2, 1)$. The other two macrostates $(2, 0)$ and $(0, 2)$, respectively with both elements in the first and the second cell, are each composed of only one microstate each, respectively $(1, 1)$ and $(2, 2)$.

Boltzmann entropy E_m refers to the macrostate m of the system at a given time step and is defined as follows:

$$E_m = k \ln[w_m] \tag{7.1}$$

where w_m is the number of microstates of m, $\ln[\cdot]$ is the natural logarithm and k is a scaling constant.

As at any time-step the probability of having any microstate is constant and equal to $1/C^N$. The probability that the system is in a given macrostate is proportional to the number of microstates that compose it: this probability is equal to w_m/C^N. Now consider the possibility that an *ordering mechanism* (e.g., a flow of energy that goes trough the system) starts to operate on the elements of the system previously subject only to noise. This mechanism is "ordering" in the sense that it drives the system towards macrostates composed of fewer microstates, so it operates *against the noise*, that is against the evolution that the system would undergo if only driven by randomness. The important point for Boltzmann entropy is that as the elements of the system wander across the different states *due to noise*, and hence the system wanders across the different corresponding microstates, at a given time step the system has a *high probability* of being in *macrostates* that are *formed by many microstates* vs. macrostates that are formed by few microstates. As Boltzmann entropy is positively related to the number of microstates that compose the macrostate of the system, it can be considered a *measure of the disorder of the system* caused by the random forces acting on its composing elements and operating against the ordering mechanisms eventually existing within it. This also implies that Boltzmann entropy can be used as an index to detect the *presence* and *level* of effectiveness of ordering mechanisms operating in the system: the lower the value of the index, the stronger the effectiveness of such mechanisms.

Notice that highly disordered macrostates correspond to situations where the elements of the system tend to be more equally distributed over the cells (these are macrostates composed by many microstates), hence to situations where the system is highly *symmetric*, whereas ordered macrostates correspond to situations where the system is *asymmetric*, for example macrostates where the system's elements gather in few cells (these are macrostates composed by relatively few microstates). With this respect, ordering mechanisms operating on the system tend to lead it from symmetric to asymmetric global states.

The reader should note an important feature of the index of disorder used here: it allows computing the level of disorder of a dynamical system *at a given time step*, whereas many other indexes applied to dynamical systems, such as the entropy rate and the excess entropy, are used to capture the regularities of the states visited by the systems in time (Feldman 1998; Prokopenko et al. 2006). This property allows the use of the index to study how the level of order of systems evolves in time, as done here and in Baldassarre et al. (2007a). Intuitively, the reason why the index can compute the level of disorder of a system at an instant of time, i.e., on the basis of a "synchronic picture" of it, is that unlike other indexes it does not need to compare the states that system assumes in time in order to estimate the probabilities of such states. But it rather computes such probabilities on the basis of the *potential microstates* that the system *might* have assumed if driven by sheer random forces.

Calculating the specific value of the index for a particular macrostate m assumed by a system requires computing the number w_m of microstates that compose it. This number can be obtained as follows:

$$w_m = \frac{N!}{N_1!N_2!\cdots N_C!} \quad \sum_{i=1}^{C} N_i = N \tag{7.2}$$

where N_i is the number of elements in the cell c, and "!" is the factorial operator. The formula relies upon the fact that there are $((N)(N-1)\cdots(N-N_1+1))/N_1!$ different possible sets of elements that can occupy the first cell, there are $((N-N_1)(N-N_1-1)\cdots(N-N_1-N_2+1))/N_2!$ different sets of elements that can occupy the second cell for each set of elements occupying the first cell, and so on. The expression for w_m is given by the multiplication of these elements referring to all the C cells. Substituting Eq. (7.2) into Eq. (7.1) of the index one has:

$$E_m = k\ln[w_m] = k\ln\left[\frac{N!}{N_1!N_2!\cdots N_C!}\right] = k\left(\ln[N!] - \sum_{i=1}^{C}\ln[N_i!]\right) \tag{7.3}$$

Once N and C are given, the maximum entropy is equal to the entropy of the macrostate where the N elements are equally distributed over the cells. This allows setting k to one divided by the maximum entropy, obtaining, from Eq. (7.3), a normalized entropy index ranging in [0, 1]:

$$\begin{aligned} E_m = k\ln[w_m] &= \frac{1}{\ln[\frac{N!}{((N/C)!)^C}]}\ln[w_m] \\ &= \frac{1}{\ln[N!] - C\ln[(N/C)!]}\left(\ln[N!] - \sum_{i=1}^{C}\ln[N_i!]\right) \end{aligned} \tag{7.4}$$

Last, the calculation of the index can avoid the computation of the factorials, which becomes unfeasible for increasing integers, by using the Stirling's approximation:

$$\ln[n!] \approx \left(n + \frac{1}{2}\right)\ln[n] - n + \ln\left[\sqrt{2\pi}\right] \tag{7.5}$$

Stirling's approximation gives increasingly good approximations for integers n of increasing size (e.g., the error of approximation goes below 0.5 % for $n > 20$).

7.2.3 An Hypothesis: Self-Organization of Multi-Robot Systems as a Phase Transition

One of the main contributions of this paper is to present some results that suggest that the self-organization of robotic systems as those considered here might have

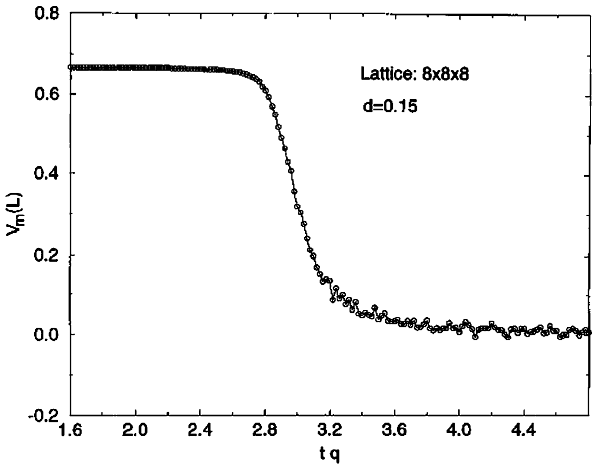


Fig. 7.1 Example of phase transition studied in physics. Y-axis: a measure of magnetization (fourth-order cumulant) in a spin-1 Ising model. X-axis: temperature. Reported from Tsai and Salinas (1998: copyright of the Brazilian Journal of Physics)

the features of phase transitions as those studied in physics. According to Wikipedia (2008) (http://en.wikipedia.org/wiki/Phase_transition), a phase transition can be defined as follows: "In physics, a phase transition, or phase change, is the transformation of a thermodynamic system from one phase to another. The distinguishing characteristic of a phase transition is an *abrupt sudden change* in one or more *physical properties*, in particular the heat capacity, with a *small change* in a thermodynamic *variable* such as the temperature" (*Italics* added). The distinguishing feature of a phase transition is hence the fast change of a variable related to the collective level of a system (e.g., the heat capacity of a gas, that is the capacity of a whole gaseous system to absorb energy when temperature changes of a certain amount) when a variable related to the behavior of the composing elements (e.g., the average noisy movement of the molecules of a gas, captured by the temperature) is slowly changed and passes a *critical value* that characterizes the phase transition.

The diagram of Fig. 7.1 shows an example of phase transition in a physical system, illustrated through a result obtained in physics with a spin-1 Ising model related to finite spin systems (Tsai and Salinas 1998). This example shows how the magnetization properties of the spin system undergoes an abrupt change when the temperature of the system is slowly decreased below a critical value.

Here we suggest that the dynamics of organization generated by self-organizing principles in multi-robot systems might share some features with that of the global organization exhibited by some physical systems undergoing a phase transition. The suggestion stems from the following considerations. The behavior of individual robots is affected by noise that influences their sensors' reading and actuators' performance. This noise causes the robots to act in a random disorganized fashion. On the other side, the controller of the robots might implement an "ordering mechanism" of the kind "I do what you do" that tends to generate self-organization within the system. However, in order to lead the whole system to successfully self-organize (i.e., all robots converge on the same behavior), the ordering mechanism has to overcome the effects of noise. This requires three conditions: (a) the signal that is perceived by the robots through the sensors, that informs them on the behavior of the other robots (i.e., that allows the robots to know "what you do"), is *sufficiently* high with respect to noise; (b) the commands issued to the motors (i.e., the "I do" part) are *sufficiently* effective and succeed to overcome the noise affecting actuator's response; (c) the controller is capable of implementing a "conformist principle" that self-organization needs to function (i.e., to implement the causation "what you do $\rightarrow$ I do").

These considerations suggest the following prediction: in the case the actuators are sufficiently reliable, the controllers are sufficiently effective, and the controller produces a conformist behaviour, if the noise/signal ratio related to the robots sensors is slowly decreased starting from high values, then the organization of the system generated by self-organizing principles should *abruptly* emerge, as in phase transitions studied in physics. The fact that such order should emerge "abruptly" is due to the fact that once self-organization succeeds to amplify some random fluctuations vs. noise, that is to overcome the "noise barrier" that initially prevents the emergence of the system's organization by continuously disrupting the asymmetries generated by the random fluctuations, then the positive feedback mechanism generates a self-reinforcing process that further strengthens the signal that enforces the robots to adopt the same behavior. Consequently, such signal definitely overcomes noise and the system "remains locked" in the organized phase and resists external perturbations due to noise. Section 7.5 will present some preliminary results that support this prediction and the related explanation.

7.3 Robots and Task

The scenario used for the experiments consists of a group of simulated robots (from 4 to 36, see Figs. 7.2 and 7.6, the latter explained later) set in an open arena. The robots are physically linked (they are manually assembled before the experiment) and their controller is evolved with a genetic algorithm. The task of the robots is to harmonize their direction of motion in order to move together as far as possible from the initial position in a given amount of time.

The simulation of the robots was carried out with a C++ program based on Vortex™ SDK, a set of commercial libraries that allow programming realistic simulations of dynamics and collisions of rigid bodies in three dimensions. The simulation of each robot was based on the prototype of a hardware robot that was built within the project SWARM-BOTS funded by the European Union (Mondada et al. 2004; see Fig. 7.2). Each robot was composed of a cylindrical turret with a diameter of 5.8 cm and a chassis with two motorized wheels at the two sides and two caster wheels at the front and at the rear for stability. The simulated robot was half the size of the hardware robot: this decreased the weights of the simulated bodies and so allowed decreasing the simulation time step of Vortex and decreasing the computational burden of the simulations (see below).

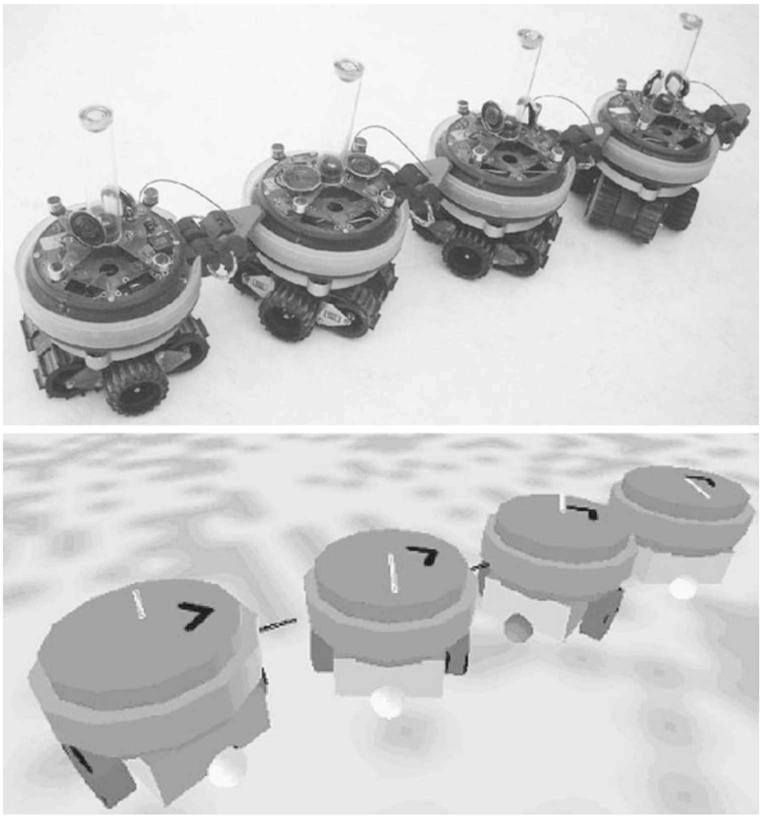


Fig. 7.2 *Top*: The hardware robots. *Bottom*: The simulated robots. Each simulated robot is made up by a chassis having two motorized cylindrical wheels and two smaller caster wheels (the visible dark-gray caster wheel marks the front of the chassis). The chassis supports a cylindrical turret (the arrow on the turret indicates its orientation)

The chassis was capable of freely rotating with respect to the turret through a further motor. This motor was activated on the basis of the difference of the activation of the motors of the two side wheels to ease the robots' turning while being physically linked to other robots (see Baldassarre et al. 2006, for details). The turret was provided with a gripper through which the robot could grasp other robots: this gripper was simulated through a rigid joint connecting the robots since our work focused on the behavior of groups of robots that were physically linked between them during the whole duration of the experiments. The gravitational acceleration coefficient was set at 9.8 cm/s^2 and the maximum torque of the wheels' motors was set at 70 dynes/cm. These low parameter settings, together with the small size of the robots, allowed the use of a relatively fast integration time step in Vortex lasting 100 ms. This was desirable since simulations based on Vortex are computationally very heavy. The speed of the wheels was updated by the robots' controllers every 100 ms and could vary within ± 5 rad/s.

Each robot had only a sensor, a special sensor called *traction sensor* (introduced for the first time in Baldassarre et al. 2003). This sensor was placed between the turret and the chassis. The sensor indicated to the robot the angle (with respect to the chassis orientation) and the intensity of the force that the turret exerted on the chassis. During the tests this force was caused by the physical interactions between the robots, in particular by the mismatch of the direction of movement of the chassis of the robot with respect to the movement of its turret and hence of the robots attached to it. Notice that if one assumes a perfect rigidity of the physical links, the turrets and the links of the robots of the group formed a whole solid body, so the traction measured the mismatch of movement between the robot's chassis and the rest of the group. Traction, seen as a vector, was affected by a 2D noise of ± 5 % of its maximum length (computed based on a simulation where one robot tries to move at maximum speed and the group is still).

The controller of each robot was a two-layer feed-forward neural network. The input layer was composed of four sensory units that encoded the traction force from four different preferential orientations with respect to the chassis orientation (rear, left, front and right). When the angle was within $\pm 90°$, each of these units had an activation proportional to the cosine of the angle between the unit's preferential orientation and the traction direction. With angles different from $\pm 90°$, the units had a zero activation. The units' activation was also multiplied by the intensity of traction normalized in [0, 1] based on its maximum value. The last unit of the input layer was a bias unit that was constantly activated with 1. The output of the neural network was formed by two sigmoid output units. These units were used to activate the wheels' motors by mapping their activation onto the range of the desired speed motor commands that varied in ± 5 rad/s.

The connection weights of the neural controllers were evolved through an evolutionary algorithm (Nolfi and Floreano 2001). Initially the algorithm created a population of 100 random genotypes. Each genotype contained a binary encoding of the ten connection weights of the neural controller (the weights ranged over ± 10). The neural controller encoded by a genotype was duplicated for a number of times equal to the number of robots forming a group, and these identical controllers were used to control the robots themselves (so the robots were "clones").

Groups of four robots connected to form a line were used to evolve the controllers. Each group was tested in five epochs each lasting 150 cycles (15 s). At the beginning of each epoch the robots were assigned random chassis' orientations. The 20 genotypes corresponding to the groups with the best performance of each generation were used to generate five copies each. Each bit of these copies was mutated (flipped) with a probability of 0.015. The whole cycle composed of these testing, selecting, and reproducing phases was repeated 100 times (generations). The whole evolutionary process was replicated 30 times by starting with different populations of randomly generated genotypes. Notice that in this evolutionary algorithm one genotype corresponds to one robots' group (so the group is the unit of selection of the genetic algorithm), and the robots' groups compete and are selected as wholes. This allows obtaining groups composed of highly cooperating individuals so avoiding the risk of the emergence of "free rider" individuals within them.

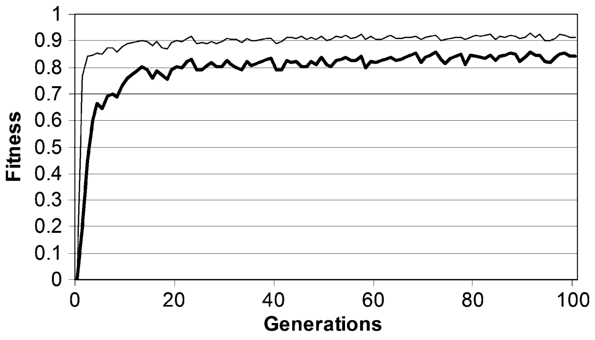


Fig. 7.3 The fitness (y-axis) of the best robots' group (*thin curve*), and average of the whole population (*bold curve*), across the 100 generations of one of the best evolutionary processes (x-axis)

The genetic algorithm selected the best 20 genotypes (groups) of the population of each generation on the basis of a fitness criterion capturing the ability of the groups to move as straight and as fast as possible. In particular, the Euclidean distance covered by each group from the starting point to the point reached at the end of the epoch was measured and averaged over the five epochs. To normalize the value of the fitness within [0, 1] the distance averaged over the five epochs was divided by the maximum distance covered by a single robot moving straight at maximum speed in 15 s (one epoch).

7.4 Analysis of the Emerged Self-Organizing Behavior at the Individual and Collective Level

The graph of Fig. 7.3 shows how the fitness of the best group and the average fitness of the whole population of 100 groups increase throughout the generations in one evolutionary run. Testing the best groups of the last generation of each of the 30 evolution replications for 100 epochs showed that the best and worst group have a performance of respectively 0.91 and 0.81. This means that all the evolutionary runs produce groups that are very good in coordinating and moving together.

Now the functioning of the evolved behavior will be described at the individual level and then at the collective level, focussing on the controller emerged in the 30th run of evolution (one with top fitness). Overall, the behavior of single robots can be described as a "conformist behavior": the robots tend to follow the movement of the group as signaled by their traction sensors. Figure 7.4 shows more in detail the commands that the controller issues to the motors of the wheels in correspondence to different combinations of intensities and angles of traction. If a robot is moving towards the same direction of motion of the group, the robot perceives a zero or low traction from the front (around 180°): in this case the robot keeps moving straight. If the robot is moving in one direction and the group moves towards its left hand side, it tends to perceive a traction from the left (around 90°) and as a consequence turns left. Similarly, if the robot is moving in one direction and the group moves towards its right hand side, it tends to perceive a traction from the right (around 270°) and as a consequence turns right. Finally, if the robot moves in the opposite direction with

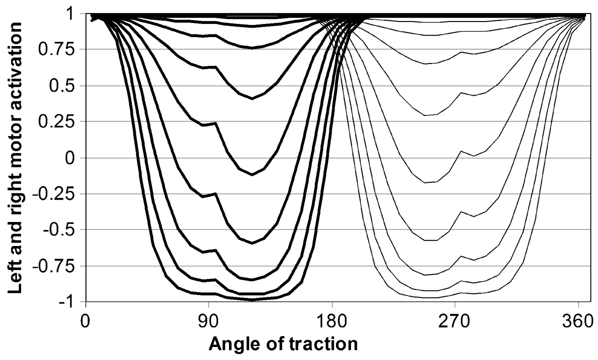


Fig. 7.4 The graph shows how a robot's left motor (*bold curves*) and right motor (thin curves) react to a traction force with eleven different levels of intensity (different *bold* and *thin lines*) and angles measured clockwise from the rear of the chassis of the robot (x-axis). The speed of the wheels (y-axis) is scaled between -1 (that corresponds to a wheel's maximum backward speed) and $+1$ (wheel's maximum forward speed)

respect to the group's movement, it perceives a traction from the rear (around 0°): in this case the robot tends to move straight, but since this is an unstable equilibrium state situated between the behaviors of turning left and right, the robot soon escapes it due to noise.

When the evolved robots are tested together, one can observe that they start to pull and push in different directions selected at random. In fact initially there is symmetry in the distribution of the motion directions over 360°. Noise causes some robots to move toward similar directions. If one of these random fluctuations eventually gains enough intensity, so that the other robots feels a traction in that direction, it breaks the initial symmetry: other robots start to follow such bearing, and in so doing they further increase the traction felt by the non-aligned robots toward the same direction. The whole group will hence rapidly converge toward the same direction of motion: the positive feedback mechanism succeeds in amplifying one of the initial random fluctuations so causing an avalanche effect that rapidly leads the whole group to coordinate.

It is important to note that the common direction of motion that emerges in one coordinated motion test is the result of a collective decision based on the amplification of some fluctuations that depend on the robots' initial random orientations. As a consequence, as shown in Fig. 7.5, if the test is repeated more times the group's direction of motion that emerges is always different.

Similarly important, in some tests where the robots' chassis have particular initial orientations, the group starts to rotate around its geometrical center. This collective behavior is a stable equilibrium for the group since the robots perceive a slight traction towards the center of the group itself, which makes them to keep moving in circle around it. The experiments show that the stronger the symmetry of the group with respect to its center, the more likely that it falls into this stable state.

The illustrated robots' behavior indicates that the distributed coordination performed by the evolved robots' controller relies upon the self-organizing mechanism

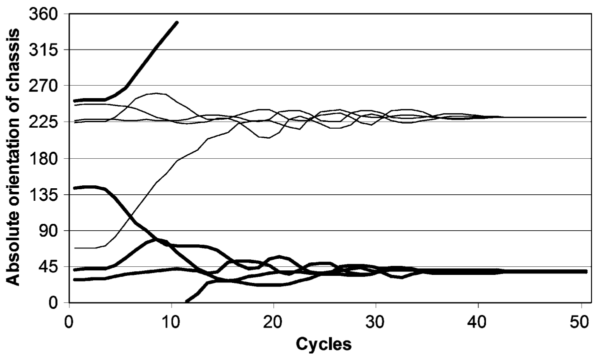


Fig. 7.5 The absolute angles (with respect to the environment) of the chassis' orientations of the four robots forming a group (y-axis) measured in two tests (respectively *bold* and *thin curves*) where the initial orientations are randomly selected

of positive feedback. Indeed, the behavior that the robots exhibit at the individual level is of the type "conform to the behavior of the group", as requested by the positive feedback mechanism (see Sect. 7.2.1). Moreover at the collective level, as illustrated in Fig. 7.5, this behavior leads the robots to amplify some random fluctuations that eventually move the system away from the initial symmetric state. As a consequence the system achieves a complete asymmetric ordered state corresponding to a very good alignment and coordination of the robots.

7.5 The Emergence of Organization vs. Noise: A Phase Transition?

This section presents some results that suggest that the organization generated by the self-organizing mechanisms presented in the previous sections might have some features in common with the organization observed in phase transitions of physical systems. Notice that to gain stability of the data, the tests reported in this section were carried out with a group of robots formed by far more individuals than those that composed the group with which the controller was evolved, precisely 36 (Fig. 7.6). This was possible because, as shown in detail elsewhere (Baldassarre et al. 2006, 2007b), the evolved controller has very good scaling properties due to the self-organizing mechanisms it relies upon.

First of all, let us see how the entropy index was applied to the robotic system. The possible orientation angle of each robot, within the range $[0°, 360°]$ (this was considered as the state space of the elements of the system), was divided into eight "cells" of 45° each. The 0° angle was set to correspond to 22.5° clockwise with respect to the absolute angle of one particular robot chosen as "pivot" (the angles of the other robots were then computed anticlockwise with respect to this origin angle). Notice that while the origin angle on the basis of which the cells are computed is arbitrary, the selection done here assured that when the group achieved high coordination, the chassis' orientations of the robots were located close to the center of the first cell and inside it (minimum entropy). Moreover, as the pivot robot was

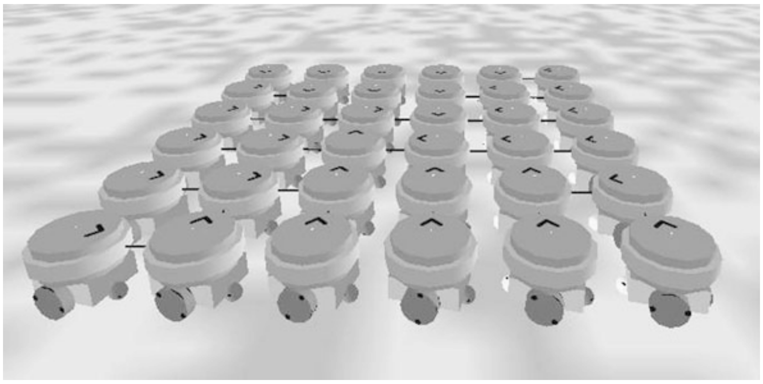


Fig. 7.6 A group of 36 robots engaged in the coordinated motion task. The black segments between the turrets of robots' couples represent the physical connection between them

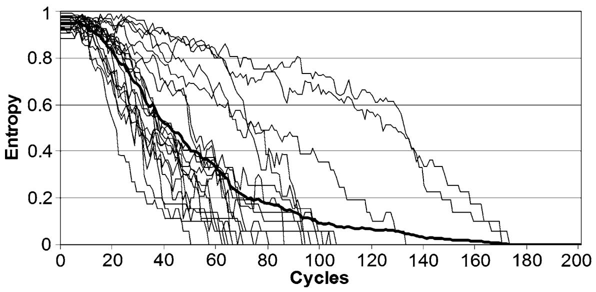


Fig. 7.7 Entropy of a group formed by 36 robots engaged in a coordinated motion task. The *thin lines* refers to the entropy measured in 20 tests that lasted 200 cycles each and were run with different initial random orientations of the robots' chassis; the *bold line* is the average of the 20 tests

always in the first cell, the number of microstates used to compute the entropy was computed with respect to $N - 1 = 35$ and not N robots.

In order to normalize E_m within [0, 1], the scaling constant k of the index was set to one divided by the maximum value that $\ln[w_m]$ (see Eq. (7.1)) could assume for the studied system, corresponding to a uniform distribution of the chassis' orientations over the eight cells. In particular, given the low number of robots, for greater accuracy instead of considering (7.4) the maximum value was directly computed on the basis of Eq. (7.2) by considering the most uniform distribution that could be obtained with the 35 robots composing the system:

$$k = 1/\ln\left[35!/(5!\,5!\,5!\,4!\,4!\,4!\,4!\,4!)\right] \approx 1/\ln\left[7.509 * 10^{26}\right] \approx 1/61.8843 \approx 0.01615 \tag{7.6}$$

The graph in Fig. 7.7 illustrates the functioning of the index by reporting the level of entropy measured *during* 20 coordinated motion tests run with the system formed by 36 robots shown in Fig. 7.6. The figure shows how the disorganization of the group initially decreases exponentially and then stabilizes at a null value when all the robots have converged to the same direction of motion (see Baldassarre et al. 2007a for a statistical analysis and further considerations on these results).

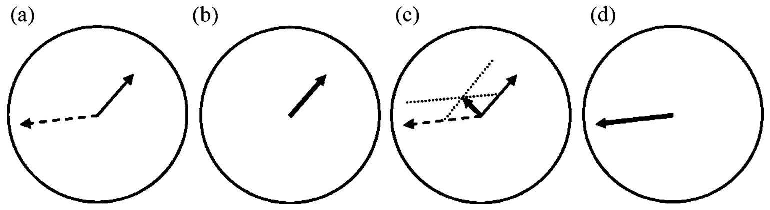


Fig. 7.8 Scheme of how the signal perceived by each robot was corrupted by noise at each time step of the tests depending on the noise/signal ratio: (**a**) an example of traction signal (*continuous arrow*) and noise (*dashed arrow*) represented as vectors; (**b**) if the ratio is equal to zero, the signal is not corrupted by noise (the signal perceived by the robot is represented by the *bold arrow*); (**c**) if the ratio has an intermediate value, for example 0.5 as in this case, the signal is partially corrupted by noise; (**d**) if the ratio is equal to one, the signal is completely substituted by noise

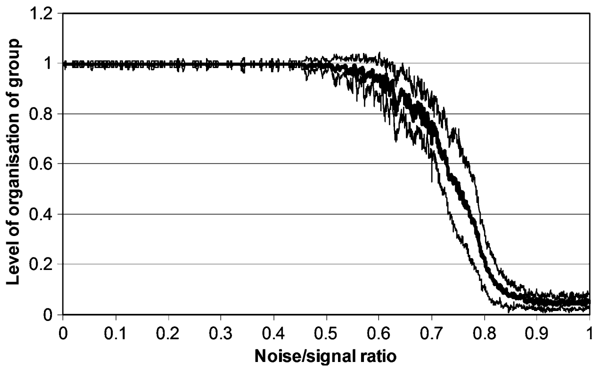


Fig. 7.9 Relationship between the noise/signal ratio and the level of organization of the group (equal to the complement to one of the normalized entropy) measured while slowly lowering the noise/signal ratio from one to zero. Average (*bold line*) ± standard deviation (*thin lines*) of the results obtained in 20 replications of the experiment

The tests directed to evaluate if the self-organization of the robotic system has the properties of a phase transition relied upon a slow progressive decrease of the ratio between noise and the signal returned by the traction sensor (recall from Sect. 7.3 that such signal is used by the robots to "know" the direction of movement of the other robots so as to conform to it). In particular, the noise/signal ratio was built through the following procedure (see Fig. 7.8): (a) At each time step, a 2D vector similar to the signal's vector was randomly generated (this vector had a random direction and a length ranging in [0, 1]). (b) The controller of the robot received as input a vector equal to a weighted average of the random vector and the signal vector (this average vector was obtained by multiplying the length of the two vectors by the respective "weights" of the average, and then by computing the sum of the resulting vectors with the parallelogram rule). (c) The weights of this weighted average were respectively equal to $\varepsilon \in [0, 1]$ and to $(1 - \varepsilon)$ for the noise and the signal: the "noise/signal ratio" manipulated in the experiments presented below was ε.

This computation of the ratio allowed running 20 tests with the 36-robots system where the noise/signal ratio ε was linearly lowered from one to zero during 20,000 time steps. During these tests the entropy of the group was measured. Figure 7.9 reports the results of these measurements in terms of the relationship between the

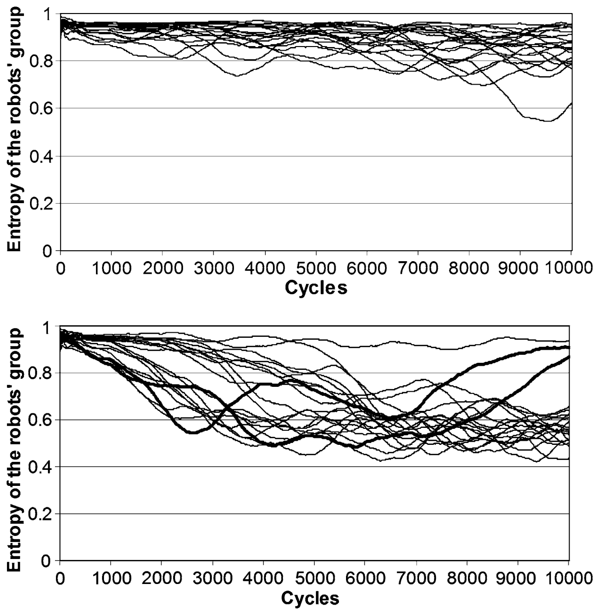


Fig. 7.10 Level of entropy (100-step moving average) of the 36-robot system in 20 tests lasting 10,000 steps each, when the noise/signal ratio is set at two different fixed levels, namely 0.80 and 0.75 for the top and bottom graph respectively (the level of the noise/signal ratio is indicated on the y-axis of each graph by the *bold arrow*). The *two bold lines of the bottom graph* refer to two tests where the system first reached an ordered state and then lost it

noise/signal ratio and the level of *order* of the group (i.e. the complement to one of the normalized entropy index).

A first relevant fact highlighted by the figure is that the system starts to organize at a very high level of noise/signal ratio, about 0.8, indicating a surprising robustness vs. noise of the self-organizing mechanisms employed by the system. Previous work (Baldassarre et al. 2006) already gave some indications in such direction but this result overcomes prior expectations and furnishes a quantitative measure of the level of such robustness.

The second relevant fact is that when the noise/signal ratio is progressively lowered, organization does not increase linearly but rather reaches its maximum level quite abruptly in correspondence to levels of noise/signal ratio ranging approximately between 0.6 and 0.8. This suggests that there is a critical noise/signal level in correspondence to which the system exhibits a transition from a disorganized to an organized state.

To further investigate the possible existence of such critical value, groups of 20 tests where carried out by setting the noise/signal level to fixed values chosen in the range between 0.9 and 0.6, at intervals of 0.05, and by measuring the level of entropy of the system in 10,000 cycles of simulation. The goal of these tests was to verify if there was a critical level of noise/signal ratio above and below which the system exhibited a discontinuous behavior in terms of overall organization. The outcome of these tests suggested that this might be the case. In particular Fig. 7.10, that shows the outcome of these tests for three levels of noise/signal ratio, indicates that this critical level might be within (0.75, 0.80). In fact, if the noise/signal value is set at 0.80 the entropy of the system fluctuates in the range of (0.80, 1.00), that is around its maximum values (in evaluating the level of order corresponding to such noise/signal values, consider that a level of entropy of 0.9 corresponds to quite uniform distributions of the robots on the cells, for example: $5, 6, 6, 6, 6, 5, 1, 0$). On the contrary, for noise/signal values set at 0.75 in 18 out of 20 experiments the entropy level of the system initially decreases from about 0.95 to about 0.55, indicating that the system self-organizes, and then stabilizes at values ranging in $(0.45, 0.65)$ (in evaluating the level of order corresponding to such noise/signal values, consider that a level of entropy of 0.55 corresponds to quite concentrated distributions of the robots on the cells, for example: $0, 1, 6, 20, 7, 1, 0, 0$). Once the system "gets locked" in the ordered state, it tends to resist noise perturbations, as predicted by the considerations presented in Sect. 7.2.1. Indeed, entropy raised again to high values only in 2 out of 20 cases after the system reached the ordered state (see bold lines in the bottom graph of Fig. 7.10).

7.6 Conclusions

This paper presented a multi-robot system guided by a decentralized control system evolved with a genetic algorithm. The control system is capable of coordinating the robots so as to accomplish a collective task relying upon a minimal implicit communication between them and self-organizing mechanisms. These self-organizing mechanisms were first described at the level of individual and collective behavior, and then the effects they produced on the level of organization of the whole system were quantitatively analyzed on the basis of an index based on Boltzmann entropy. This analysis showed that, when one slowly decreases the noise/signal ratio related to the signal that the robots use to coordinate, the dynamics of the self-organization exhibited by the system resembles the self-organization characterizing physical systems undergoing phase-transitions. In particular, the order of the system tends to emerge quite abruptly when the ratio is lowered below a critical value.

The hypothesis that the dynamics of the level of order of self-organized multi-robot systems might have the features of a phase transition would have important implications if confirmed. In fact it would imply that self-organization of collective systems tends to manifest in an all-or-nothing fashion depending on the quality of the signals exchanged by the elements forming the system. Moreover, when such quality overcomes a critical value, even of a *small* amount, the organization produced by the self-organizing mechanisms becomes fully effective and robust vs. noise (as the system "locks in" in its state of order). These implications are relevant for engineering purposes. For example identifying the critical noise-signal level that characterizes a distributed multi-robot system might allow adjusting the physical set-up of the latter so as to achieve a reliable level of robustness of its self-organization. The implications are also important for scientific purposes, for example for investigating self-organization in collective biological systems (Bonabeau et al. 1999; Camazine et al. 2001; Anderson et al. 2002). In fact in some of such systems self-organization emerges quite abruptly if some parameters of the system change beyond certain thresholds. For example, trail formation in ants requires that the number of ants that compose the group, and hence the amount of pheromone released on the ground, reaches a certain level for the organization of the group to emerge. Indeed, given that the laid pheromone trace slowly vanishes in time, if the number of ants, and hence the level of the released pheromone, is not enough, the signal that it furnishes to the ants is too weak to allow them to self-organize.

The added value of the paper resides also in the techniques it presented. In particular such techniques might not only be used to measure the level of organization of decentralized (and also centralized) systems, as done here, but it might also be directly used as fitness function to evolve systems that exhibit useful behaviors (for some examples of this, that use entropy indexes different from those used here, see Prokopenko et al. 2006), or to explore the self-organization potential of systems. Moreover, the identification of the critical noise/signal ratio that characterizes a decentralized robotic system might be a way to furnish a quantitative *measure of the robustness* of the self-organizing principles that govern it.

Notwithstanding the relevance of all these implications, we recognize that the results presented in the paper, in particular those related to the hypothesis according to which in some conditions self-organization of some multi-robot systems might behave as a phase transition, are preliminary under many respects. For example, further research is needed to corroborate or falsify the hypothesis itself, to better understand the behavior of the system in correspondence to the critical level of the noise/signal ratio, and to better understand the relationship existing between the level of order of the system and the role that it plays in its functioning (e.g., in its capacity to displace in space). Moreover, it might be useful to build a mathematical abstract model of the system to carry out an analytical study directed to ascertain at a more formal level if it posses the properties that characterize phase transitions. For example, this analysis might identify some quantities associated with the self-organization of the robotic system that behave similarly to "free energy" or "latent heat" in phase transitions of physical systems (for an introduction on these topics, see http://en.wikipedia.org/wiki/Phase_transition).

A last observation is that experiments similar to those conducted here by slowly lowering the noise/signal ratio might be also conducted on the actuator's noise and on the controller's effectiveness. With this respect it might be possible to envisage a way to regulate the "noise/effectiveness level" of actuators, or the "level of effectiveness" of the controller in ways similar to the one used here to regulate the noise/signal ratio of sensors. These experiments might show that also these two manipulations lead to phase-transitions at the level of the system's overall organization.

7.7 Epilogue

The multi-robot system presented in the first edition (Baldassarre 2008) has been further developed in two follow-up works. The first work (Baldassarre and Nolfi 2009) further investigated the robotic system used here to show how the controller evolved with the genetic algorithm can be captured with a simple mathematical function linking the direction and strength of the traction sensor to the motor commands. The parameters of the mathematical function can be found based on a non-linear regression of the input-output points of the original controller and the whole technique has a general applicability to simple controllers. The paper shows that, in general, this transformation allows "bridging" the controllers develop with evolutionary techniques with more standard robotic controllers, such as behaviour based (e.g., schema-based) controllers, so allowing the exploitation of the strengths of both. Once this transformation is done, thanks to the robustness of the original controller capable of exploiting self-organisation, the resulting function offers a number of advantages. These go from a higher transparency with respect to the original neural network, to the possibility of changing its parameters by hand, and to the possibility of using the function to build more-complex compound controllers capable of solving a number of different tasks. With respect to the issues discussed here, the function-based description of the controller might also facilitate the application of formal and principled tools to investigate the self-organising principles underlying evolved controllers.

The second work (Ferrauto et al. 2013) studies again a multi-robot decentralised system of robots engaged in navigation tasks (here groups are formed only by two robots). However, in this case the work focusses on different possible genetic algorithms that might be used to evolve the robots so to lead them to solve two different tasks requiring either specialisation or dynamic role-taking. Based on these tasks, the work analyses the most important genetic algorithms proposed so far to evolve collective systems showing their strengths and weaknesses for the two types of tasks. The different genetic algorithms vary with respect to the unit of selection, the number of populations used, and the test of each robot within a fixed or variable group. The relevance of this work for the issues faced here resides in the fact that the controllers evolved with the different genetic algorithms tend to exploit different self-organisation principles such as symmetry breaking in role allocation and self-organised behaviour generated by robots with different controllers.

Although promising, no further work has been carried out on the specific issue tackled here and related to self-organisation principles of multi-robot systems analysed in quantitative and formal ways (the author research has diverged to the study of behaviour and brain of single organisms). However, the author is still convinced that the research presented in this chapter contributed to open a very important new research thread within the study of self-organising multi-robot systems. The reason is that the "methodological message" of this paper is still very important. Such message can be summarised in three points as follows:

- Multi-robot systems exploiting self-organisation principles are very robust, effective, and simple. This makes them very interesting from a scientific point of view, and potentially very useful from an engineering point of view.
- To fully understand and exploit self-organising principles in multi-robot systems, and to be cumulative in doing so, we need to study such self-organising principles in a quantitative/formal fashion where theory and empirical tests go hand in hand.
- The theoretical and formal apparatus needed for doing this can be borrowed from physics and information theory: these can furnish the needed ideas, principles,

formalisms, and metrics to investigate self-organising principles in a quantitative and principled fashion.

Acknowledgements This research has been supported by the *SWARM-BOTS* project funded by the Future and Emerging Technologies program (IST-FET) of the European Commission under grant IST-2000-31010. I thank Stefano Nolfi and Domenico Parisi with which I designed, developed and studied extensively the robotic setup studied in the paper.

References

Anderson, P. (1997). *Basic notions of condensed matter physics*. Cambridge: Perseus.

Anderson, C., Theraulaz, G., & Deneubourg, J.-L. (2002). Self-assemblages in insect societies. *Insectes Sociaux*, *49*, 1–12.

Baldassarre, G. (2008). Self-organization as phase transition in decentralized groups of robots: a study based on Boltzmann entropy. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (1st ed.). London: Springer.

Baldassarre, G., & Nolfi, S. (2009). Strengths and synergies of evolved and designed controllers: a study within collective robotics. *Artificial Intelligence*, *173*, 857–875.

Baldassarre, G., Nolfi, S., & Parisi, D. (2003). Evolution of collective behaviour in a group of physically linked robots. In G. Raidl, A. Guillot, & J.-A. Meyer (Eds.), *Applications of evolutionary computing—proceedings of the second European workshop on evolutionary robotics* (pp. 581–592). Berlin: Springer.

Baldassarre, G., Parisi, D., & Nolfi, S. (2006). Distributed coordination of simulated robots based on self-organization. *Artificial Life*, *12*(3), 289–311.

Baldassarre, G., Parisi, D., & Nolfi, S. (2007a). Measuring coordination as entropy decrease in groups of linked simulated robots. In A. Minai & Y. Bar-Yam (Eds.), *Proceedings of the fifth international conference on complex systems (ICCS2004)*, Boston, MA, USA, 16–21 May 2004 (pp. e1–e14). http://www.necsi.edu/events/iccs/2004proceedings.html.

Baldassarre, G., Trianni, V., Bonani, M., Mondada, F., Dorigo, M., & Nolfi, S. (2007b). Self-organised coordinated motion in groups of physically connected robots. *IEEE Transactions on Systems, Man and Cybernetics*, *37*(1), 224–239.

Beckers, R., Holland, O. E., & Deneubourg, J.-L. (1994). From local actions to global tasks: stigmergy and collective robotics. In R. A. Brooks & P. Maes (Eds.), *Proceedings of the 4th international workshop on the synthesis and simulation of living systems (Artificial Life IV)* (pp. 181–189). Cambridge: MIT Press.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. New York: Oxford University Press.

Camazine, S., Deneubourg, J. L., Franks, N. R., Sneyd, J., Theraulaz, G., & Bonabeau, E. (2001). *Self-organization in biological systems*. Princeton: Princeton University Press.

Cao, Y. U., Fukunaga, A. S., & Kahng, A. B. (1997). Cooperative mobile robotics: antecedents and directions. *Autonomous Robots*, *4*, 1–23.

Dorigo, M., & Sahin, E. (2004). Swarm robotics—special issue editorial. *Autonomous Robots*, *17*(2–3), 111–113.

Dorigo, M., Trianni, V., Sahin, E., Gross, R., Labella, T. H., Baldassarre, G., Nolfi, S., Denebourg, J.-L., Floreano, D., & Gambardella, L. M. (2004). Evolving self-organizing behavior for a swarm-bot. *Autonomous Robots*, *17*(2–3), 223–245.

Dudek, G., Jenkin, M., Milios, E., & Wilkes, D. (1996). A taxonomy for multi-agent robotics. *Autonomous Robots*, *3*, 375–397.

Feldman, P. D. (1998). *A brief introduction to: Information theory, excess entropy and computational mechanics* (Technical report). Department of Physics, University of California.

Ferrauto, T., Parisi, D., Di Stefano, G., & Baldassarre, G. (2013, in press). Different genetic algorithms and the evolution of specialisation: a study with groups of simulated neural robots. *Artificial Life*.

Holland, O., & Melhuish, C. (1999). Stimergy, self-organization, and sorting in collective robotics. *Artificial Life*, *5*, 173–202.

Ijspeert, A. J., Martinoli, A., Billard, A., & Gambardella, L. M. (2001). Collaboration through the exploitation of local interactions in autonomous collective robotics: the stick pulling experiment. *Autonomous Robots*, *11*, 149–171.

Krieger, M. J. B., Billeter, J. B., & Keller, L. (2000). Ant-like task allocation and recruitment in cooperative robots. *Nature*, *406*, 992–995.

Kube, R. C., & Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, *30*, 85–101.

Kube, C. R., & Zhang, H. (1993). Collective robotics: from social insects to robots. *Adaptive Behavior*, *2*(2), 189–219.

Mondada, F., Pettinaro, G., Guignard, A., Kwee, I., Floreano, D., Denebourg, J.-L., Nolfi, S., Gambardella, L. M., & Dorigo, M. (2004). Swarm-bot: a new distributed robotic concept. *Autonomous Robots*, *17*(2–3), 193–221.

Nolfi, S., & Floreano, D. (2001). *Evolutionary robotics. the biology, intelligence, and technology of self-organizing machines*. Cambridge: MIT Press.

Prokopenko, M. (2008). Design versus self-organization. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (pp. 3–18). London: Springer.

Prokopenko, M., Gerasimov, V., & Tanev, I. (2006). Evolving spatiotemporal coordination in a modular robotic system. In S. Nolfi, G. Baldassarre, R. Calabretta, J. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, & D. Parisi (Eds.), *Lecture notes in computer science: Vol. 4095. From animals to animats 9: proceedings of the ninth international conference on the simulation of adaptive behavior (SAB-2006)* (pp. 558–569). Berlin: Springer.

Prokopenko, M., Boschetti, F., & Ryan, A. J. (2009). An information-theoretic primer on complexity, self-organization, and emergence. *Complexity*, *15*(1), 11–28.

Quinn, M., Smith, L., Mayley, G., & Husbands, P. (2003). Evolving controllers for a homogeneous system of physical robots: structured cooperation with minimal sensors. *Philosophical Transactions - Royal Society. Mathematical, Physical and Engineering Sciences*, *361*, 2321–2344.

Reynolds, C. W. (1987). Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, *21*(4), 25–34.

Trianni, V., Nolfi, S., & Dorigo, M. (2006). Cooperative hole-avoidance in a swarm-bot. *Robotics and Autonomous Systems*, *54*(2), 97–103.

Tsai, S., & Salinas, S. R. (1998). Fourth-order cumulants to characterize the phase transitions of a spin-1 Ising model. *Brazilian Journal of Physics*, *28*(1), 58–65.

Chapter 8
Distributed Control of Microscopic Robots in Biomedical Applications

Tad Hogg

8.1 Microscopic Robots

The development of molecular electronics, motors and chemical sensors could enable constructing large numbers of devices able to sense, compute and act in micron-scale environments. Such microscopic robots, of sizes comparable to bacteria, could simultaneously monitor entire populations of cells individually in vivo. Their small size allows the robots to move through the tiniest blood vessels, so could pass within a few cell diameters of most cells in large organisms via their circulatory systems to perform a wide variety of biological research and medical tasks. For instance, robots and nanoscale-structured materials inside the body could significantly improve disease diagnosis and treatment (Freitas 1999; Morris 2001; NIH 2003; Keszler et al. 2001). Initial tasks for microscopic robots include in vitro research via simultaneous monitoring of chemical signals exchanged among many bacteria in a biofilm. The devices could also operate in multicellular organisms as passively circulating sensors. Such devices, with no need for locomotion, would detect programmed patterns of chemicals as they pass near cells. More advanced technology could create devices able to communicate to external detectors, allowing real-time in vivo monitoring of many cells. The devices could also have capabilities to act on their environment, e.g., releasing drugs at locations with specific chemical patterns or mechanically manipulating objects for microsurgery. Extensive development and testing is necessary before clinical use, first for high-resolution diagnostics and later for programmed actions at cellular scales.

Realizing these benefits requires fabricating the robots cheaply, in large numbers and with sufficient capabilities. Such fabrication is beyond current technology. Nevertheless, ongoing progress in engineering nanoscale devices could eventually enable production of such robots. One approach to creating microscopic pro-

T. Hogg (✉)
Hewlett-Packard Laboratories, Palo Alto, CA, USA
e-mail: tad.hogg@hp.com

grammable machines is engineering biological systems, e.g., bacteria executing simple programs (Andrianantoandro et al. 2006), and DNA computers responding to logical combinations of chemicals (Benenson et al. 2004). However, biological organisms have limited material properties and computational speed. Instead we focus on machines based on plausible extensions of current molecular-scale electronics, sensors and motors (Berna et al. 2005; Collier et al. 1999; Craighead 2000; Howard 1997; Fritz et al. 2000; Montemagno and Bachand 1999; Soong et al. 2000; Wang and Williams 2005). These devices could provide components for stronger and faster microscopic robots than is possible with biological organisms. Thus the focus here is on nonbiological robots containing nanoscale sensors and electronics, along with a power source, within a protective shell. As technology improves, such robots could be supplemented with other capabilities such as communication and locomotion.

Because we cannot yet fabricate microscopic robots with molecular electronics components, estimates of their performance rely on plausible extrapolations from current technology. The focus in this paper is on biomedical applications requiring only modest hardware capabilities, which will be easier to fabricate than more capable robots. Designing controls for microscopic robots is a key challenge: not only enabling useful performance but also compensating for their limited computation, locomotion or communication abilities. Distributed control is well-suited to these capabilities by emphasizing locally available information and achieving overall objectives through self-organization of the collection of robots. Theoretical studies allow developing such controls and estimating their performance prior to fabrication, thereby indicating design tradeoffs among hardware capabilities, control methods and task performance. Such studies of microscopic robots complement analyses of individual nanoscale devices (McCurdy et al. 2002; Wang and Williams 2005), and indicate even modest capabilities enable a range of novel applications.

The operation of microscopic robots differs significantly from that of larger robots (Mataric 1992), especially for biomedical applications. First, the physical environment is dominated by viscous fluid flow. Second, thermal noise is a significant source of sensor error and Brownian motion limits the ability to follow precisely specified paths. Third, relevant objects are often recognizable via chemical signatures rather than, say, visual markings or specific shapes. Fourth, the tasks involve large numbers of robots, each with limited abilities. Moreover, a task will generally only require a modest fraction of the robots to respond appropriately, not for all, or even most, robots to do so. Thus controls using random variations are likely to be effective simply due to the large number of robots. This observation contrasts with teams of larger robots with relatively few members: incorrect behavior by even a single robot can significantly decrease team performance. These features suggest reactive distributed control is particularly well-suited for microscopic robots.

Organisms contain many microenvironments, with distinct physical, chemical and biological properties. Often, precise quantitative values of properties relevant for robot control will not be known a priori. This observation suggests a multi-stage protocol for using the robots. First, an information-gathering stage with passive robots placed into the organism, e.g., through the circulatory system, to measure relevant properties (Hogg and Kuekes 2006). The information from these robots, in conjunction with conventional diagnostics at larger scales, could then determine appropriate controls for further actions in subsequent stages of operation.

For information gathering, each robot notes in its memory whenever chemicals matching a prespecified pattern are found. Eventually, the devices are retrieved and information in their memories extracted for further analysis in a conventional computer with far more computational resources than available to any individual microscopic robot. This computer would have access to information from many robots, allowing evaluation of aggregate properties of the population of cells that individual robots would not have access to, e.g., the number of cells presenting a specific combination of chemicals. This information allows estimating spatial structure and strength of the chemical sources. The robots could detect localized high concentrations that are too low to distinguish from background concentrations when diluted in the whole blood volume as obtained with a sample. Moreover, if the detection consists of the joint expression of several chemicals, each of which also occurs from separate sources, the robot's pattern recognition capability could identify the spatial locality, which would not be apparent when the chemicals are mixed throughout the blood volume.

Estimating the structure of the chemical sources from the microscopic sensor data is analogous to computerized tomography (Natterer 2001). In tomography, the data consists of integrals of the quantity of interest (e.g., density) over a large set of lines with known geometry selected by the experimenter. The microscopic sensors, on the other hand, record data points throughout the tissue, providing more information than just one aggregate value such as the total number of events. However, the precise path of each sensor through the tissue, i.e., which vessel branches it took and the locations of those vessels, will not be known. This mode of operation also contrasts with uses of larger distributed sensor networks able to process information and communicate results while in use.

Actions based on the information from the robots would form a second stage of activity, perhaps with specialized microscopic robots (e.g., containing drugs to deliver near cells), with controls set based on the calibration information retrieved earlier. For example, the robots could release drugs at chemically distinctive sites (Freitas 1999, 2006) with specific detection thresholds determined with the information retrieved from the first stage of operation. Or robots could aggregate at the chemical sources (Casal et al. 2003; Hogg 2007) or manipulate biological structures based on surface chemical patterns on cells, e.g., as an aid for microsurgery in repairing injured nerves (Hogg and Sretavan 2005). These active scenarios require more advanced robot capabilities, such as locomotion and communication, than needed for passive sensing. The robots could monitor environmental changes due to their actions, thereby documenting the progress of the treatment. Thus the researcher or physician could monitor the robots' progress and decide whether and when they should continue to the next step of the procedure. Using a series of steps, with robots continuing with the next step only when instructed by the supervising person, maintains overall control of the robots, and simplifies the control computations each robot must perform itself.

To illustrate controls for large collections of microscopic robots, this paper considers a prototypical diagnostic task of finding a small chemical source in a multicellular organism via the circulatory system. To do so, we first review plausible capabilities for microscopic robots and the physical constraints due to operation in fluids at low Reynolds number, diffusion-limited sensing and thermal noise from Brownian motion. We then discuss techniques for evaluating the behavior of large collections of robots, and examine a specific task scenario. The emphasis here is on feasible performance with plausible biophysical parameters and robot capabilities. Evaluation metrics include minimizing hardware capabilities to simplify fabrication and ensuring safety, speed and accuracy for biological research or treatment in a clinical setting.

8.2 Capabilities of Microscopic Robots

This section describes plausible robot capabilities based on currently demonstrated nanoscale technology. Minimal capabilities needed for biomedical tasks include chemical sensing, computation and power. Additional capabilities, enabling more sophisticated applications, include communication and locomotion.

8.2.1 Chemical Sensing

Large-scale robots often use sonar or cameras to sense their environment. These sensors locate objects from a distance, and involve sophisticated algorithms with extensive computational requirements. In contrast, microscopic robots for biological applications will mainly use chemical sensors, e.g., the selective binding of molecules to receptors altering the electrical characteristics of nanoscale wires. The robots could also examine chemicals inside nearby cells (Xie et al. 2006).

Microscopic robots and bacteria face similar physical constraints in detecting chemicals (Berg and Purcell 1977). The diffusive capture rate γ for a sphere of radius a in a region with concentration C is (Berg 1993)

$$\gamma = 4\pi DaC \tag{8.1}$$

where D is the diffusion coefficient of the chemical. Even when sensors cover only a relatively small fraction of the device surface, the capture rate is almost this large (Berg 1993). Nonspherical devices have similar capture rates so Eq. (8.1) is a reasonable approximation for a variety of designs.

Current molecular electronics (Wang and Williams 2005) and nanoscale sensors (Li et al. 2005; Patolsky and Lieber 2005; Sheehan and Whitman 2005) indicate plausible sensor capabilities. At low concentrations, sensor performance is primarily limited by the time for molecules to diffuse to the sensor and statistical fluctuations in the number of molecules encountered is a major source of noise.

Moreover, other chemicals with similar binding properties to the chemical of interest would give additional noise. Thus the selectivity of the sensor is important in setting the noise level, and may trade-off with the time used to determine whether a detection occurred (Alon 2007).

8.2.2 Timing and Computation

With the relevant fluid speeds and chemical concentrations described in Sect. 8.4, robots pass through high concentrations near individual cells on millisecond time scales. Thus identifying significant clusters of detections due to high concentrations requires a clock with millisecond resolution. This clock need not be globally synchronized with other devices.

In a simple scenario, devices just store sensor detections in their memories for later retrieval. In this case most of the computation to interpret sensor observations takes place in larger computers after the devices are retrieved. Recognizing and storing a chemical detection involves at least a few arithmetic operations to compare sensor counts to threshold values stored in memory. An estimate on the required computational capability is about 100 elementary logic operations and memory accesses within a 10 ms measurement time. This gives about 10^4 logic operations per second. While modest compared to current computers, this rate is significantly faster than demonstrated for programmable bacteria (Andrianantoandro et al. 2006) but well within the capabilities of molecular electronics.

8.2.3 Communication

A simple form of one-way communication is robots passively sensing electromagnetic or acoustic signals from outside the body. An example is using radio frequency to produce local heating in metal nanospheres attached to the devices (Hamad-Schifferli et al. 2002). Such signals could activate robots only within certain areas of the body at, say, centimeter length scales. Using external signals to localize the robots more precisely is difficult due to variations in propagation through tissues, so such localization requires more complex technology (Freitas 1999) than considered in this discussion.

Additional forms of communication, between nearby robots and sending information to detectors outside the organism, are more difficult to fabricate and require significant additional power. For example, since the robots considered here have chemical sensors, a natural approach to communication is using an onboard storage of specific chemicals they could release for detection by other nearby robots. Such diffusion-mediated signals are not effective for communicating over distances beyond a few microns but could mark the environment for detection by other robots that pass nearby later, i.e., stigmergy (Bonabeau et al. 1999). Acoustic signals provide more versatile communication. Compared to fluid flow, acoustic signals are essentially instantaneous, but power constraints limit their range to about 100 μm (Freitas 1999).

8.2.4 Locomotion

Biomedical applications will often involve robots operating in fluids. Viscosity dominates the robot motion, with different physical behaviors than for larger organisms and robots (Purcell 1977; Vogel 1994; Fung 1997; Karniadakis and Beskok 2002; Squires and Quake 2005). The ratio of inertial to viscous forces for an object of size s moving with velocity v through a fluid with viscosity η and density ρ is the Reynolds number $\mathrm{Re} \equiv s\rho v/\eta$. Using typical values for density and viscosity (e.g., of water or blood plasma) in Table 8.1 and noting that reasonable speeds for robots with respect to the fluid (Freitas 1999) are comparable to the fluid flow speed in small vessels, i.e., $\sim$1 mm/s, motion of a 1-micron robot has $\mathrm{Re} \approx 10^{-3}$, so viscous forces dominate. Consequently, robots applying a locomotive force quickly reach terminal velocity in the fluid, i.e., applied force is proportional to velocity as opposed to the more familiar proportionality to acceleration of Newton's law $F = ma$. By contrast, a swimming person has Re about a billion times larger.

Flow in a pipe of uniform radius R has a parabolic velocity profile: velocity at distance r from the axis is

$$v(r) = 2v_{\mathrm{avg}}\bigl(1 - (r/R)^2\bigr) \tag{8.2}$$

where v_{avg} is the average speed of fluid in the pipe.

Robots moving through the fluid encounter significant drag. For instance, an isolated sphere of radius a moving at speed v through a fluid with viscosity η has a drag force

$$6\pi a\eta v \tag{8.3}$$

Although not quantitatively accurate near boundaries or other objects, this expression estimates the drag in those cases as well. For instance, a numerical evaluation of drag force on a 1 μm-radius sphere moving at velocity v with respect to the fluid flow near the center of a 5 μm-radius pipe, has drag about three times larger than given by Eq. (8.3). Other reasonable choices for robot shape have similar drag.

Fluid drag moves robots in the fluid. An approximation is robots without active locomotion move with the same velocity as fluid would have at the center of the robot if the robot were not there. Numerical evaluation of the fluid forces on the robots for the parameters of Table 8.1 show the robots indeed move close to this speed when the spacing between robots is many times their size. Closer packing leads to more complex motion due to hydrodynamic interactions (Hernandez-Ortiz et al. 2005; Riedel et al. 2005).

Table 8.1 Parameters for the environment, robots and the chemical signal. The robots are spheres with radius a. The chemical signal concentrations correspond to a typical 10 kilodalton chemokine molecule, with mass concentrations near the source and background (i.e., far from the source) equal to 3×10^{-5} kg/m^3 and 10^{-7} kg/m^3, respectively. C_{source} is equivalent to a three nanomolar solution. The $\rho_{\text{vessel}} V$ small vessels in the tissue volume occupy about a fraction $\rho_{\text{vessel}} \pi R^2 L \approx 4\ \%$ of the volume

Parameter	Value
Tissue, vessels and source	
Vessel radius	$R = 5$ µm
Vessel length	$L = 1000$ µm
Number density of vessels in tissue	$\rho_{\text{vessel}} = 5 \times 10^{11}$/m^3
Tissue volume	$V = 10^{-6}$ m^3
Source length	$L_{\text{source}} = 30$ µm
Fluid	
Fluid density	$\rho = 10^3$ kg/m^3
Fluid viscosity	$\eta = 10^{-3}$ kg/m/s
Average fluid velocity	$v_{\text{avg}} = 10^{-3}$ m/s
Fluid temperature	$T = 310$ K
Robots	
Robot radius	$a = 1$ µm
Number density of robots	$\rho_{\text{robot}} = 2 \times 10^{11}$ robot/m^3
Robot diffusion coefficient	$D_{\text{robot}} = 7.6 \times 10^{-14}$ m^2/s
Chemical signal	
Production flux at target	$F_{\text{source}} = 5.6 \times 10^{13}$ molecule/s/m^2
Diffusion coefficient	$D = 10^{-10}$ m^2/s
Concentration near source	$C_{\text{source}} = 1.8 \times 10^{18}$ molecule/m^3
Background concentration	$C_{\text{background}} = 6 \times 10^{15}$ molecule/m^3

8.2.5 Additional Sensing Capabilities

In addition to chemical sensing, robots could sense other properties to provide high-resolution spatial correlation of various aspects of their environment. For example, nanoscale sensors for fluid motion can measure fluid flow rates at speeds relevant for biomedical tasks (Ghosh et al. 2003), allowing robots to examine in vivo microfluidic behavior in small vessels. In particular, at low Reynolds number, boundary effects extend far into the vessel (Squires and Quake 2005), giving an extended gradient in fluid speed with higher fluid shear rates nearer the wall. Thus, several such sensors, extending a small distance from the device surface in various directions, could estimate shear rates and hence the direction to the wall or changes in the vessel geometry. Another example for additional sensing is optical scattering in cells, as has been demonstrated to distinguish some cancer from normal cells in vitro (Gourley et al. 2005).

8.2.6 Power

To estimate the power for robot operation, each logic operation in current electronic circuits uses 10^4–10^5 times the thermal noise level $k_B T = 4 \times 10^{-21}$ J at the fluid temperature of Table 8.1, where k_B is the Boltzmann constant. Near term molecular electronics could reduce this to $\approx 10^3 k_B T$, in which case 10^4 operations per second uses a bit less than 0.1 pW. Additional energy will be needed for signals within the computer and with its memory.

This power is substantially below the power required for locomotion or communication. For instance, due to fluid drag and the inefficiencies of locomotion in viscous fluids, robots moving through the fluid at $\approx 1\,\text{mm/s}$ dissipate a picowatt (Berg 1993). However, these actions may operate only occasionally, and for short times, when the sensor detects a signal, whereas computation could be used continuously while monitoring for such signals.

For tasks of limited duration, an on-board fuel source created during manufacture could suffice. Otherwise, the robots could use energy available in their environment, such as converting vibrations to electrical energy (Wang and Song 2006) or chemical generators. Typical concentrations of glucose and oxygen in the bloodstream could generate ≈ 1000 pW continuously, limited primarily by the diffusion rate of these molecules to the device (Freitas 1999). For comparison, a typical person at rest uses about 100 watts. Beyond simply generating the required power, the robots require effective machines to distribute and use the energy, with several possible approaches (Freitas 1999).

8.3 Evaluating Collective Robot Performance

Because the microscopic robots can not yet be fabricated and quantitative biophysical properties of many microenvironments are not precisely known, performance studies must rely on plausible models of both the machines and their task environments (Drexler 1992; Freitas 1999; Requicha 2003). Microorganisms, which face physical microenvironments similar to those of future microscopic robots, give some guidelines for feasible behaviors.

Cellular automata are one technique to evaluate collective robot behavior. For example, a two-dimensional scenario shows how robots could assemble structures (Arbuckle and Requicha 2004) using local rules. Such models can help understand structures formed at various scales through simple local rules and some random motions (Whitesides and Grzybowski 2002; Griffith et al. 2005). However, cellular automata models either ignore or greatly simplify physical behaviors such as fluid flow. Another analysis technique considers swarms (Bonabeau et al. 1999), which are well-suited to microscopic robots with their limited physical and computational capabilities and large numbers. Most swarm studies focus on macroscopic robots or behaviors in abstract spaces (Gazi and Passino 2004) which do not specifically include physical properties unique to microscopic robots. In spite of the simplified physics, these studies show how local interactions among robots lead to various collective behaviors and provide broad design guidelines.

Simulations including physical properties of microscopic robots and their environments can evaluate performance of robots with various capabilities. Simple models, such as a two-dimensional simulation of chemotaxis (Dhariwal et al. 2004), provide insight into how robots find microscopic chemical sources. A more elaborate simulator (Cavalcanti and Freitas 2002) includes three-dimensional motions in viscous fluids, Brownian motion and environments with numerous cell-sized objects, though without accounting for how they change the fluid flow. Studies of hydrodynamic interactions (Hernandez-Ortiz et al. 2005) among moving devices include more accurate fluid effects.

Another approach to robot behaviors employs a stochastic mathematical framework for distributed computational systems (Hogg and Huberman 2004; Lerman et al. 2001). This method directly evaluates average behaviors of many robots through differential equations determined from the state transitions used in the robot control programs. Direct evaluation of average behavior avoids the numerous repeated runs of a simulation needed to obtain the same result. This approach is best suited for simple control strategies, with minimal dependencies on events in individual robot histories. Microscopic robots, with limited computational capabilities, will likely use relatively simple reactive controls for which this analytic approach is ideally suited. Moreover, these robots will often act in environments with spatially varying fields, such as chemical concentrations and fluid velocities. Even at micron scales, the molecular nature of these quantities can be approximated as continuous fields with behavior governed by partial differential equations. For application to microscopic robots, this approximation extends to the robots themselves, treating their locations as a continuous concentration field, and their various control states as corresponding to different fields, much as multiple reacting chemicals are described by separate concentration fields. This continuum approximation for average behavior of the robots will not be as accurate as when applied to chemicals or fluids, but nevertheless gives a simple approach to average behaviors for large numbers of robots responding to spatial fields. One example of this approach is following chemical gradients in one dimension without fluid flow (Galstyan et al. 2005).

Cellular automata, swarms, physically-based simulations and stochastic analysis are all useful tools for evaluating the behaviors of microscopic robots. One example is evaluating the feasibility of rapid, fine-scale response to chemical events too small for detection with conventional methods, including sensor noise inherent in the discrete molecular nature of low concentrations. This paper examines this issue in a prototypical task using the stochastic analysis approach. This method allows incorporating more realistic physics than used with cellular automata studies, and is computationally simpler than repeated simulations to obtain average behaviors.

This technique is limited in requiring approximations for dependencies introduced by the robot history, but readily incorporates physically realistic models of sensor noise and consequent mistakes in the robot control decisions. The stochastic analysis indicates plausible performance, on average, and thereby suggests scenarios suited for further, more detailed, simulation studies.

8.4 A Task Scenario

As a prototypical task for microscopic robots, we consider their ability to respond to a cell-sized source releasing chemicals into a small blood vessel. This scenario illustrates a basic capability for the robots: identifying small chemically-distinctive regions, with high sensitivity due to the robots' ability to get close (within a few cell-diameters) to a source. This capability would be useful as part of more complex tasks where the robots are to take some action at such identified regions. Even without additional actions, the identification itself provides extremely accurate and rapid diagnostic capability compared to current technology.

Microscopic robots acting independently to detect specific patterns of chemicals are analogous to swarms (Bonabeau et al. 1999) used in foraging, surveillance or search tasks. Even without locomotion capabilities, large numbers of such devices could move through tissues by flowing passively in moving fluids, e.g., through blood vessels or with lymph fluid. As the robots move, they can monitor for preprogrammed patterns of chemical concentrations.

Chemicals with high concentrations are readily detected with the simple procedure of analyzing a blood sample. Thus the chemicals of interest for microscopic robot applications will generally have low concentrations. With sufficiently low concentrations and small sources, the devices are likely to only encounter a few molecules while passing near the source, leading to significant statistical fluctuations in number of detections.

We can consider this task from the perspective of stages of operation discussed in the introduction. For a diagnostic first stage, the robots need only store events in their memory for later retrieval, when a much more capable conventional computer can process the information. For an active second stage where robots react to their detections, e.g., to aggregate at a source location or release a drug, the robots would need to determine themselves when they are near a suitable location. In this later case, the robots would need simple control decision procedure, within the limits of local information available to them and their computational capacity. Such a control program could involve comparing counts to various thresholds, which were determined by analysis of a previous diagnostic stage of operation.

8.4.1 Example Task Environment

Tissue microenvironments vary considerably in their physical and chemical properties. As a specific example illustrating the capabilities of passive motion by micro-

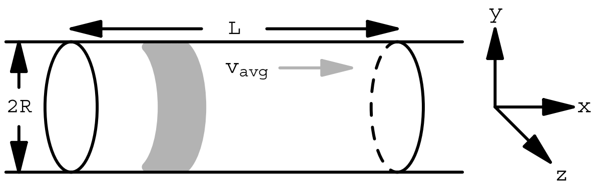


Fig. 8.1 Schematic illustration of the task geometry as a vessel, of length L and radius R. Fluid flows in the positive x-direction with average velocity v_{avg}. The *gray area* is the source region wrapped around the surface of the pipe

scopic sensors, we consider a task environment consisting of a macroscopic tissue volume V containing a single microscopic source producing a particular chemical (or combination of chemicals) while the rest of the tissue has this chemical at much lower concentrations. This tissue volume contains a large number of blood vessels, and we focus on chemical detection in the small vessels, since they allow exchange of chemicals with surrounding tissue, and account for most of the surface area. A rough model of the small vessels is each has length L and they occur throughout the tissue volume with number density ρ_{vessel}. Localization to volume V could be due to a distinctive chemical environment (e.g., high oxygen concentrations in the lungs), an externally supplied signal (e.g., ultrasound) detectable by sensors passing through vessels within the volume, or a combination of both methods. The devices are active only when they detect they are in the specified region.

Robots moving with fluid in the vessels will, for the most part, be in vessels containing only the background concentration, providing numerous opportunities for incorrectly interpreting background concentration as source signals. These false positives are spurious detections due to statistical fluctuations from the background concentration of the chemical. Although such detections can be rare for individual devices, when applied to tasks involving small sources in a large tissue volume, the number of opportunities for false positive responses can be orders of magnitude larger than the opportunities for true positive detections. Thus even a low false positive rate can lead to many more false positive detections than true positives. The task scenario examined in this paper thus includes estimating both true and false positive rates, addressing the question of whether simple controls can achieve a good trade-off of both a high true positive rate and low false positive rate.

For simplicity, we consider a vessel containing only flowing fluid, robots and a diffusing chemical arising from a source area on the vessel wall. This scenario produces a static concentration of the chemical throughout the vessel, thereby simplifying the analysis. We examine the rate at which robots find the source and the false positive rate as functions of the detection threshold used in a simple control rule computed locally by each robot.

Figure 8.1 shows the task geometry: a segment of the vessel with a source region on the wall emitting a chemical into the fluid. Robots continually enter one end of the vessel with the fluid flow. We suppose the robots have neutral buoyancy and move passively in the fluid, with speed given by Eq. (8.2) at their centers. This approximation neglects the change in fluid flow due to the robots, and is reasonable for estimating detection performance when the robots are at low enough density to be spaced apart many times their size, as is the case for the example presented here. The robot density in Table 8.1 corresponds to 10^9 robots in the entire 5-liter blood volume of a typical adult, an example of medical applications using a huge number of microscopic robots (Freitas 1999). These robots use only about 10^{-6} of the vessel volume, far less than the 20 %–40 % occupied by blood cells. The total mass of all the robots is about 4 mg.

The scenario for microscopic robots examined here is detecting small areas of infection or injury. The chemicals arise from the initial immunological response at the injured area and enter nearby small blood vessels to recruit white blood cells (Janeway et al. 2001). We consider a typical protein produced in response to injury, with concentration near the injured tissue of about 30 ng/ml and background concentration in the bloodstream about 300 times smaller. These chemicals, called chemokines, are proteins with molecular weight around 10^4 daltons (equal to about 2×10^{-23} kg). These values lead to the parameters for the chemical given in Table 8.1, with chemical concentrations well above the demonstrated sensitivity of nanoscale chemical sensors (Patolsky and Lieber 2005; Sheehan and Whitman 2005). This example incorporates features relevant for medical applications: a chemical indicating an area of interest, diffusion into flowing fluid, and a prior background level of the chemical limiting sensor discrimination.

8.4.2 Diffusion of Robots and Chemicals

Diffusion arising from Brownian motion is somewhat noticeable for microscopic robots, and significant for molecules. The diffusion coefficient D, depending on an object's size, characterizes the resulting random motion, with root-mean-square displacement of $\sqrt{6Dt}$ in a time t. For the parameters of Table 8.1, this displacement for the robots is about $0.7\sqrt{t}$ microns with t measured in seconds. Brownian motion also randomly alters robot orientation.

The chemical concentration C is governed by the diffusion equation (Berg 1993)

$$\frac{\partial C}{\partial t} = -\nabla \cdot \mathbf{F} \tag{8.4}$$

where $\mathbf{F} = -D\nabla C + \mathbf{v}C$ is the chemical flux, i.e., the rate at which molecules pass through a unit area, and $\mathbf{v}$ is the fluid velocity vector. The first term in the flux is diffusion, which acts to reduce concentration gradients, and the second term is motion of the chemical with the fluid.

We suppose the source produces the chemical uniformly with flux F_{source}. To evaluate high-resolution sensing capabilities, we suppose the chemical source is small, with characteristic size L_{source} as small as individual cells. Total target surface area is $2\pi RL_{\text{source}} \approx 9.4 \times 10^{-10}$ m^2, about the same as the surface area of a single endothelial cell lining a blood vessel. The value for F_{source} in Table 8.1 corresponds to 5×10^4 molecule/s from the source area as a whole. This flux was chosen to make

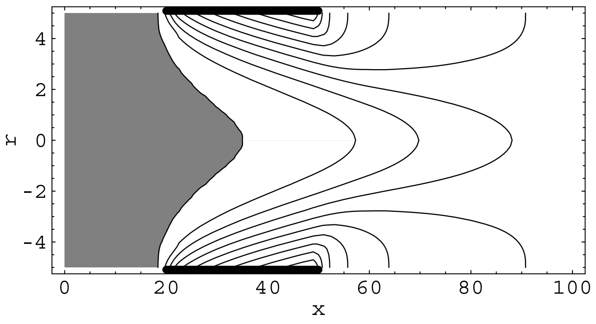


Fig. 8.2 Concentration contours on a cross section through the vessel, including the axis ($r = 0$) in the middle and the walls ($r = \pm R$) at the top and bottom. The *gray area* shows the region where the concentration from the target is below that of the background concentration. The *thick black lines* along the vessel wall mark the extent of the target region. The vertical and horizontal scales are different: the cross section is 10 μm vertically and 100 μm horizontally. Numbers along the axes denote distances in microns

the concentration at the source area equal to that given in Table 8.1. The background concentration is the level of the chemical when there is no injury.

Objects, such as robots, moving in the fluid alter the fluid's velocity to some extent. For simplicity, and due to the relatively small volume of the vessel occupied by the robots, we ignore these changes and treat the fluid flow as constant in time and given by Eq. (8.2). Similarly, we also treat detection as due to absorbing spheres with concentration C at the location of the center of the sphere for Eq. (8.1), assuming the robot does not significantly alter the average concentration around it. Figure 8.2 shows the resulting steady-state concentration from solving Eq. (8.4) with $\partial C/\partial t = 0$. The concentration decreases with distance from the source and the high concentration contours occur downstream of the source due to the fluid flow.

8.4.3 Control

The limited capabilities of the robots and the need to react on millisecond time scales leads to emphasizing simple controls based on local information. For chemical detection at low concentrations, the main sensor limitation is the time required for molecules to diffuse to the sensor. Thus the detections are a stochastic process. A simple decision criterion for a robot to determine whether it is near a source is if a sufficient number of detections occur in a short time interval. Specifically, a robot considers a signal detected if the robot observes at least $C_{\text{threshold}}$ counts in a measurement time interval T_{measure}. The choice of measurement time must balance having enough time to receive adequate counts, thereby reduce errors due to statistical fluctuations, while still responding before the robot has moved far downstream of the source where response would give poor localization of the source or, if robots are to take some action near the source, require moving upstream against the fluid

Table 8.2 Robot control parameters for chemical signal detection

Parameter	Value
Measurement time	$T_{\text{measure}} = 10$ ms
Detection threshold	$C_{\text{threshold}}$

flow. Moreover, far downstream of the source the concentration from the target is small so additional measurement time is less useful. A device passing through a vessel with a source will have about $L_{\text{source}}/v_{\text{avg}} = 30$ ms with high concentration, so a measurement time of roughly this magnitude is a reasonable choice, as selected in Table 8.2. A low value for $C_{\text{threshold}}$ will produce many false positives, while a high value means many robots will pass the source without detecting it (i.e., false negatives). False negatives increase the time required for detection while false positives could lead to inappropriate subsequent activities (e.g., releasing a drug to treat the injury or infection at a location where it will not be effective).

8.4.4 Analysis of Behavior

Task performance depends on the rate robots detect the source as they move past it, and the rate robots incorrectly conclude they detect the source due to background concentration of the chemical.

From the values of Table 8.1, robots enter any given vessel at an average rate

$$\omega_{\text{robot}} = \rho_{\text{robot}} \pi R^2 v_{\text{avg}} \approx 0.016/\text{s} \tag{8.5}$$

and the rate robots enter (and leave) any of the small vessels within the tissue volume is

$$\Omega_{\text{robot}} = \rho_{\text{vessel}} V \omega_{\text{robot}} \approx 8 \times 10^3/\text{s} \tag{8.6}$$

A robot encounters changing chemical concentration as it moves past the source. The expected number of counts a robot at position $\mathbf{r}$ has received from the source chemical during a prior measurement interval time T_{measure} is

$$K(\mathbf{r}) = 4\pi D a \int_0^{T_{\text{measure}}} C\big(\mathbf{r}'(\mathbf{r},\tau)\big) d\tau \tag{8.7}$$

where $\mathbf{r}'(\mathbf{r},\tau)$ denotes the location the robot had at time τ in the past. During the time the robot passes the target, Brownian motion displacement is $\sim$0.1 μm, which is small compared to the 10 μm vessel diameter. Thus the possible past locations leading to $\mathbf{r}$ are closely clustered and for estimating the number of molecules detected while passing the target, a reasonable approximation is the robot moves deterministically with the fluid. In our axially symmetric geometry with fluid speed given by Eq. (8.2), positions are specified by two coordinates $\mathbf{r} = (r, x)$ so $\mathbf{r}'((r,x),\tau) = (r, x - v(r)\tau)$ when the robot moves passively with the fluid and

Brownian motion is ignored. During this motion, the robot will, on average, also encounter

$$k = 4\pi DacT_{\text{measure}} \tag{8.8}$$

molecules from the background concentration, not associated with the source.

With diffusive motion of the molecules, the actual number of counts is a Poisson distributed random process. The detection probability, i.e., having at least E events when the expected number is μ, is

$$\Pr(\mu, E) = 1 - \sum_{n=0}^{E-1} \text{Po}(\mu, n)$$

where $\text{Po}(\mu, n) = e^{-\mu}\mu^n/n!$ is the Poisson distribution.

Taking the devices to be ideal absorbing spheres for the chemical described in Table 8.1, Eq. (8.1) gives the capture rates $\gamma \approx 8/\text{s}$ at the background concentration and $\approx 2300/\text{s}$ near the source. Detection over a time interval Δt is a Poisson process with mean number of detections $\mu = \gamma \Delta t$. Consider a robot at $\mathbf{r}$. During a small time interval Δt the probability to detect a molecule is $4\pi Da(C(\mathbf{r}) + c)\Delta t \ll 1$. For a robot to first conclude it has detected a signal during this short time it must have $C_{\text{threshold}} - 1$ counts in the prior $T_{\text{measure}} - \Delta t$ time interval and then one additional count during Δt. Thus the rate at which robots first conclude they detect a signal is

$$4\pi Da(C + c)\frac{\text{Po}(K + k, C_{\text{threshold}} - 1)}{\sum_{n<C_{\text{threshold}}} \text{Po}(K + k, n)} \tag{8.9}$$

In Eq. (8.9) C and K depend on robot position and the last factor is the probability the robot has $C_{\text{threshold}} - 1$ counts in its measurement time interval, given it has not already detected the signal, i.e., the number of counts is less than $C_{\text{threshold}}$. This expression is an approximation: ignoring correlations in the likelihood of detection over short time intervals. Equation (8.9) also gives the detection rate when there is no source, i.e., false positives, by setting C and K to zero.

To evaluate the rate robots detect the source as they pass it, we view the robots as having two internal control states: MONITOR and DETECT. Robots are initially in the MONITOR state, and switch to the DETECT state if they detect the chemical, i.e., have at least $C_{\text{threshold}}$ counts during time T_{measure}. Using the stochastic analysis approach to evaluating robot behavior, the steady-state concentrations of robots monitoring for the chemical, R_{m}, is governed by Eq. (8.4) for R_{m} instead of chemical concentration, with the addition of a decay due to robots changing to the DETECT state, i.e.,

$$\nabla \cdot (D_{\text{robot}} \nabla R_{\text{m}} - \mathbf{v} R_{\text{m}}) - \alpha_{\text{m}\to\text{d}} R_{\text{m}} = 0 \tag{8.10}$$

The signal detection transition, $\alpha_{\text{m}\to\text{d}}$, is given by Eq. (8.9) and depends on the choice of threshold $C_{\text{threshold}}$ and robot position.

The rate sensors detect the source using a threshold $C_{\text{threshold}}$ is

$$\omega_{\text{source}} = \omega_{\text{robot}} \int \alpha_{\text{m}\to\text{d}} R_{\text{m}} dV \tag{8.11}$$

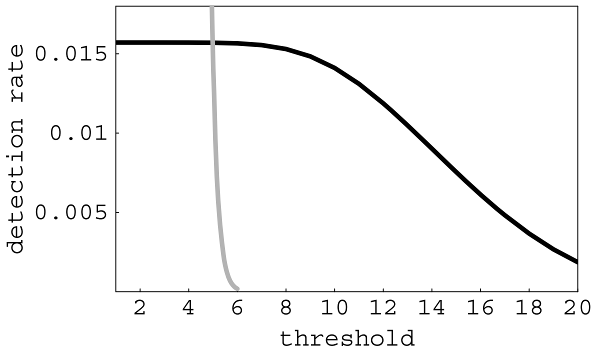


Fig. 8.3 Rates robots detect a signal for those passing through the single small vessel with the source (*black*) and all the others in the tissue volume, i.e., the false positives (*gray*) as a function of threshold $C_{\text{threshold}}$ used for detection during the T_{measure} time interval

where ω_{robot} is given by Eq. (8.5) and the integral is over the interior volume of the vessel containing the source.

The background concentration can give false positives, i.e., occasionally producing enough counts to reach the count threshold in time T_{measure}. The background concentration extends throughout the tissue volume giving many opportunities for false positives. With the parameters of Table 8.1, the expected count from background in T_{measure} is $\mu_{\text{background}} = 0.08$. Since a sensor spends $\approx L/v_{\text{avg}} = 1$ s in a small vessel in the tissue volume, the sensor has about 100 independent 10 ms opportunities to accumulate counts toward the detection threshold $C_{\text{threshold}}$. The rate of false positive detections is then

$$\omega_{\text{background}} \approx 100\, \Omega_{\text{robot}} \Pr(\mu_{\text{background}}, C_{\text{threshold}}) \tag{8.12}$$

For a diagnostic task, we can pick a detection threshold $C_{\text{threshold}}$ and a time T for sensors to accumulate counts. The expected number of sensors reporting detections from the source and from the background are then $\omega_{\text{source}} T$ and $\omega_{\text{background}} T$, respectively. The actual number is also a Poisson process, so another decision criterion for declaring a source detected is the minimum number of sensors n reporting a detection. Since expected count rate near the source is significantly larger than the background rate, the contributions to the counts from the source and background are nearly independent, so the probability for detecting a source is

$$\Pr\big((\omega_{\text{source}} + \omega_{\text{background}})T, n\big)$$

and similarly for the false positives with counts based only on $\omega_{\text{background}}$.

8.4.5 Detection Performance

Figure 8.3 shows the values of ω_{source} and $\omega_{\text{background}}$ for the parameters of Table 8.1 for various choices of the control parameter $C_{\text{threshold}}$. Despite the much larger number of opportunities for false positives compared to the single vessel with the source, the ability of robots to pass close to the source allows selecting $C_{\text{threshold}} \approx 10$ for

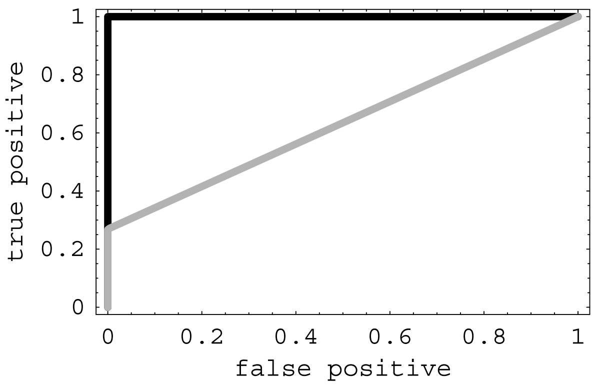


Fig. 8.4 Probabilities of detecting a single source (true positive) and mistaking background concentration for a source (false positive) after sampling for $T = 20$ and 1000 seconds (*gray* and *solid curves*, respectively). Each curve corresponds to changing the minimum threshold $C_{\text{threshold}}$ of counts during the time interval T_{measure} required to indicate a detection event

which false positive detections are small while still having a significant rate of true positives.

For diagnostics, an indication of performance is comparing the likelihood of true and false positives. In particular, identifying choices of control parameters giving both a high chance of detecting a source and a low chance of false positives. Figure 8.4 illustrates the tradeoff for the task considered here. The curves range from the lower-left corner (low detection rates) with a high threshold to the upper-right corner (high detection and high false positive rate) with a low threshold. Robots collecting data for only about 20 minutes allow high performance, in this case with $C_{\text{threshold}}$ around 10. This corresponds to the behavior seen in Fig. 8.3: $C_{\text{threshold}} \approx 10$ is high enough to be rarely reached with background concentration alone (in spite of the much larger number of vessels without the source than the single vessel with the source), but still low enough that most devices passing through the single vessel with the source will detect it.

If the robots are to act at the source, ensuring at least one detection may not be enough. For instance, if the robots release a chemical near the source, or aggregate near the source (e.g., to stick to vessel wall near the source to provide enhanced imaging or to mechanically alter the tissue), then it may be necessary to ensure a relatively large number of robots detect the source. At the same time, we wish to avoid inappropriate actions due to false positives. A criterion emphasizing safety is having a high chance the required number detect the source before a single robot has a false positive detection, so not even a single inappropriate action takes place. Figure 8.5 shows we can achieve high performance by this criterion: the robots circulate for about a day to have enough time for 1000 to detect the source while still having a low chance for any false positives.

Another motivation for requiring more than one detection at the source is to account for sensor failures, e.g., requiring detection by several sensors as independent confirmation of the source. Occasional spurious extra counts by the sensors amount to an increase in the effective background concentration. As long as these extra counts are infrequent, and not significantly clustered in time, such errors will not significantly affect the overall accuracy of the results. Conversely, sensor failures

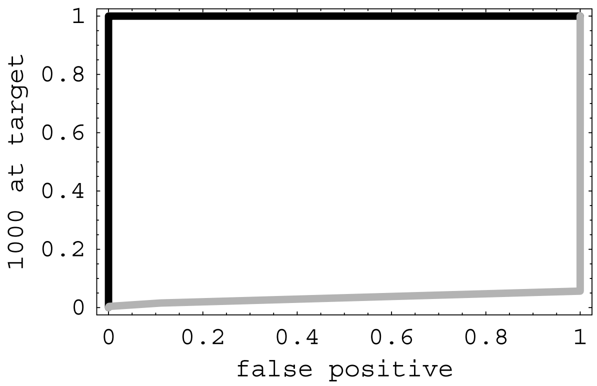


Fig. 8.5 Probability of at least 1000 robots detecting the source vs. the probability at least one robot mistakes background concentration for a source (false positive) after sampling for $T = 6 \times 10^4$ and 10^5 seconds (*gray* and *solid curves*, respectively). Each curve corresponds to changing the minimum threshold $C_{\text{threshold}}$ of counts during the time interval T_{measure} required to indicate a detection event

leading to missed counts will decrease the average detection rate, thereby requiring correspondingly longer operation times.

In summary, simple control allows fast and accurate detection of even a single cell-sized source within a macroscopic tissue volume. The key feature enabling this performance is the robots' ability to pass close to individual cells, where concentration from released chemicals is much higher than in fluid far from the cell.

For comparison, instead of using microscopic sensors, one could use a blood sample extracted from the body, which is a routine medical procedure. Conventional laboratory analysis outside the body could then attempt to detect the chemical in the sample. However, such a sample will represent the average concentration of the chemical in the full blood volume. For a small chemical source, e.g., the size of a single cell, this average amounts to a significant dilution of the chemical, resulting in considerably smaller concentrations than are available to microscopic sensors passing close to the source. As an example, suppose a single source described above produces the chemical for one day and all this production is delivered to the blood without any degrading before a sample is taken. The source producing $\sim 5 \times 10^4$ molecule/s builds up to a concentration in the 5 liter blood volume of about 7×10^{11} molecule/m^3 after a day. This concentration is about 10^{-4} of the background concentration, so the additional chemical released by the source would be difficult to detect against small variations in background concentration.

These performance estimates also indicate behavior in other scenarios. For instance, with fewer sensors, detection times would be correspondingly longer, or would only be sensitive to a larger number of sources. For instance, with 10^6 sensors, a factor of 1000 fewer than in Table 8.1, achieving the discrimination shown in Fig. 8.4 would take a thousand times longer. Alternatively, for 10^6 sensors with 1000 sources distributed randomly in the tissue volume instead of just one source, performance would be similar to that shown in the figure. The stochastic analysis

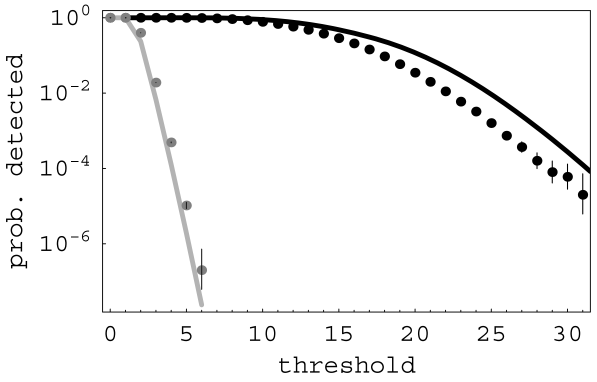


Fig. 8.6 Chemical detection probabilities vs. threshold $C_{\text{threshold}}$ for detection during the T_{measure} time interval. Two cases are shown: a robot passing the source (*black*) and a robot in a small vessel without the source, i.e., false positive detection (*gray*). Curves are the theoretical estimates, and points are from simulations. Each point includes a line showing the 95 % confidence interval of the probability estimate, which in most cases is smaller than the size of the plotted point. The points are averages from 10^5 and 10^7 discrete simulations of single robots passing through vessels with and without a source, respectively

approach used here could also estimate other aspects of robot performance, such as the average distance to the source if and when a passing robot detects it.

The stochastic analysis approach to evaluating robot behaviors in spatially varying fields makes various simplifying approximations to obtain ω_{source} and $\omega_{\text{background}}$. These include independence of the number of detection counts in different time intervals, characterizing the robots by a continuous concentration field rather than discrete objects, and ignoring how the robots (and other objects, such as blood cells) change the fluid flow and the concentration distribution of the chemical in the fluid. In principle, the analysis approach can readily include objects in the fluid, but the numerical solution becomes significantly more complicated. Instead of the parabolic velocity profile Eq. (8.2), fluid flow changes with time as the objects move through the vessel. Simultaneously, the object motion is determined by the drag force from the fluid. Evaluating the flow requires solving the Navier-Stokes equation, which can also include more complicated geometries for the vessels and source than treated here. Despite this additional numerical complexity, analyzing robot behaviors is formally the same.

On the other hand, history-dependent behavior of the robots, such as checking for a certain number of counts within a specified time interval, is difficult to include in the formalism, leading to the simplifying independence assumptions used here. As a check on the accuracy of these assumptions, a discrete-event simulation for the same task evaluates robot behavior without making those assumptions, but requires significantly more compute time to evaluate average behaviors. Figure 8.6 shows this comparison for the key quantities used here: the detection probabilities for true and false positives. We see a fairly close correspondence between the two methods, but with a systematic error. The correspondence preserves the key qualitative difference between the curves: many orders of magnitude difference in detection probabilities over a range of thresholds (in this case around 10 to 15) where the detection probability for true positives is fairly high. This large difference allows finding a choice of threshold and measurement time giving high detection probability with low false positives, as illustrated in Fig. 8.4.

8.5 Applications for Additional Robot Capabilities

High-resolution in vivo chemical sensing with passive motion in fluids is well-suited to robots with minimal capabilities. Additional hardware capabilities allow collecting more information communicating with external observers during operation, or taking actions such as aggregating at the chemical source, releasing chemicals or mechanically manipulating the cells. This section presents a number of such possibilities.

8.5.1 Improved Inference from Sensor Data

The example in Fig. 8.4 uses a simple detection criterion based on a single choice of threshold $C_{\text{threshold}}$. More sophisticated criteria could improve accuracy. One example is matching the temporal distribution of detections to either that expected from a uniform background or a spatially localized high concentration source. This procedure would also be useful when the source and background concentrations are not sufficiently well known a priori to determine a suitable threshold. Instead, the distribution of counts could distinguish the low background rate encountered most of the time from occasional clusters of counts at substantially higher rates.

Robots could use correlations in space or time for improved sensitivity. For instance, if a source emits several chemicals together, each of which has significant background concentration from a variety of separate sources, then detecting all of them within a short period of time improves statistical power of the inference. Similarly if the chemical is released in bursts, sensors nearby during a burst would encounter much higher local concentrations than the time averaged concentration. Stochastic temporal variation in protein production is related to the regulatory environment of the genes producing the protein, in particular the number of proteins made from each transcript (McAdams and Arkin 1997). Thus temporal information could identify aspects of the gene regulation within individual cells.

Correlations in the measurements could distinguish between a strong source and many weak sources throughout the tissue volume producing the chemical at the same total rate. The strong source would give high count rates for a few devices (those passing near the source) while multiple weak sources would have some detection in a larger fraction of the sensors.

As another example, using fluid flow sensors would allow correlating chemical detections with properties of the flow and the vessel geometry (e.g., branching and changes in vessel size or permeability to fluids).

8.5.2 Correlating Measurements from Multiple Devices

Determining properties of the chemical sources would improve if data from devices passing different sources is distinguished from multiple devices passing the same source. One possibility for correlating information from different devices is if sources, or their local environments, are sufficiently distinct chemically so measuring similar ratios of various chemicals in different devices likely indicates they encountered the same source. Common temporal variation could also suggest data from the same source.

With less distinctive sources or insufficient time to collect enough counts to make the distinction, devices capable of communicating with others nearby could provide this correlation information directly. With the parameters of Table 8.1, typical spacing between devices is several hundred microns, which is beyond a plausible communication distance between devices. Thus direct inter-device communication requires reducing the spacing with either a larger total number of devices or temporary local aggregation in small volumes.

Devices could achieve this aggregation if they can alter their surface properties to stick to the vessel wall (Lahann and Langer 2005). With such a programmable change, a device detecting an interesting event could stick when it next encounters the wall. The parameters of Table 8.1 give, on average, about one device passing through a given small vessel each minute. Thus a device on the wall could remain there for a few minutes, broadcasting its unique identifier to other devices as they pass. Devices would record the identifiers they receive, with a time stamp, thereby correlating their detections.

Further inferences could be made from devices temporarily on the vessel wall where small vessels merge into larger ones. In this case, the identifier received from a device on the wall enables correlating events in nearby vessels, i.e., those that merge into the same larger vessel. Similarly, if robots aggregate in upstream vessels, before they branch into smaller ones, the robots could record each others' unique identifiers. Subsequent measurements over the next few hundred milliseconds would be known to arise from the same region, i.e., either the same vessel or nearby ones.

The ability to selectively stick to vessel walls would allow another mode of operation: the devices could be injected in larger blood vessels leading into a macroscopic tissue volume of interest and then stick to the vessel walls after various intervals of time or when specific chemicals are detected. After collecting data, the devices would release from the wall for later retrieval.

8.5.3 Reporting During Operation

The devices could carry nanoscale structures with high response to external signals. Such structures could respond to light of particular wavelengths when near the skin (Wang et al. 2005), or give enhanced imaging via MRI or ultrasound (Liu et al. 2006). Such visualization mechanisms combined with a selective ability to stick to vessel walls allows detecting aggregations of devices at specified locations near the surface of the body (Service 2005).

This visualization technique could be useful even if the tissue volume of interest is too deep to image effectively at high resolution. In particular, robots could use various areas near the skin (e.g., marked with various light or ultrasound frequencies) at centimeter scales as readout regions during operation. Devices that have detected the chemicals could aggregate at the corresponding readout location, which would then be visible externally. Devices could choose how long to remain at the aggregation points based on how high a concentration of the chemical pattern they detected. This indication of whether, and (at a coarse level) what, the devices have found could help decide how long to continue circulating to improve statistics for weak chemical signatures. These aggregation points could also be used to signal to the devices, e.g., instructing them to select among a few modes of operation.

8.5.4 Detection of Chemicals Inside Cells

The task described above relies on detecting chemicals released into the bloodstream. However, some chemicals of interest may remain inside cells, or if released, be unable to get into the bloodstream. In that case, sensors in the bloodstream would not detect the chemical even when they pass through vessels near the cells.

However, an extension of the protocol could allow indirect detection of intracellular chemicals. For example, current technology can create molecules capable of entering cells, and, if they encounter specific chemicals, changing properties to emit signals or greatly enhance response to external imaging methods. Such molecules can indicate a variety of chemical behaviors within cells (Xie et al. 2006). Thus if the microscopic devices include sensors for these indicator signals, they could indirectly record the activity of the corresponding intracellular chemicals in nearby cells. Such sensing from within nearby blood vessels would complement current uses of these marker molecules with much larger scale whole body imaging. In this extended protocol, the microscopic devices would provide the same benefits of detecting chemicals directly by instead detecting a proxy signal that is able to reach the nearby blood vessels.

8.5.5 Modifying Microenvironments

Beyond the diagnostic task discussed above, robots able to locate chemically distinctive microenvironments in the body could have capabilities to modify those environments. For instance, the devices could carry specific drugs to deliver only to cells matching a prespecified chemical profile (Freitas 1999, 2006) as an extension of a recent in vitro demonstration of this capability using DNA computers (Benenson et al. 2004).

With active locomotion, after detecting the chemicals the devices could follow the chemical gradient to the source, though this would require considerable energy to move upstream against the fluid flow. Alternatively, with sufficient number of robots so they are close enough to communicate, a robot detecting the chemical could acoustically signal upstream devices to move toward the vessel wall. In this cooperative approach, the detecting device does not itself attempt to move to the source, but rather acts to signal others upstream from the source to search for it. These upstream devices would require little or no upstream motion. Furthermore, with a large number of devices, even if only a small fraction move in the correct direction to the source after receiving a signal, many would still reach the source. This approach of using large numbers and randomness in simple local control is analogous to that proposed for collections of larger reconfigurable robots (Bojinov et al. 2002; Rus and Vona 1999; Salemi et al. 2001). The behavior of microscopic active swimmers (Dreyfus et al. 2005) raises additional control issues to exploit the hydrodynamic interactions among swimming objects as they aggregate so the distance between devices becomes only a few times their size (Hernandez-Ortiz et al. 2005).

Robots aggregated at chemically identified targets could perform precise microsurgery at the scale of individual cells. Since biological processes often involve activities at molecular, cell, tissue and organ levels, such microsurgery could complement conventional surgery at larger scales. For instance, a few millimeter-scale manipulators, built from micromachine (MEMS) technology, and a population of microscopic devices could act simultaneously at tissue and cellular size scales. An example involving microsurgery for nerve repair with plausible biophysical parameters indicates the potential for significant improvement in both speed and accuracy compared to the larger-scale machines acting alone (Sretavan et al. 2005; Hogg and Sretavan 2005).

8.6 Discussion

Plausible capabilities for microscopic robots suggest a range of novel applications in biomedical research and medicine. Sensing and acting with micron spatial resolution and millisecond timing allows access to activities of individual cells. The large numbers of robots enable such activities simultaneously on a large population of cells in multicellular organisms. In particular, the small size of these robots allows access to tissue through blood vessels. Thus a device passes within a few tens of microns of essentially every cell in the tissue in a time ranging from tens of seconds to minutes, depending on the number of devices used. As described above, this access to cells allows rapid, chemical sensing with much higher resolution than possible with in vitro sample analysis. Furthermore, the precision of localization and the robots' programmability gives them a degree of flexibility to alter microenvironments, e.g., by releasing drugs, well beyond that possible with either large scale surgery or nonprogrammable chemically-targeted drug delivery. The full range of biomedical situations that could benefit from this flexibility, e.g., nerve repair (Hogg and Sretavan 2005), remains to be seen.

With many devices in the tissue but only a few in the proper context to perform task, false positives are a significant issue. In some situations, these false positives may just amount to a waste of resources (e.g., power). But in other cases, too many false positives could be more serious, e.g., leading to aggregation blocking blood vessels or incorrect diagnosis.

The sensing task described in this paper highlights key control principles for microscopic robots. Specifically, by considering the overall task in a series of stages, the person deploying the robots remains in the decision loop, especially for the key decision of whether to proceed with manipulation (e.g., release a drug) based on diagnostic information reported by the devices. Information retrieved during treatment can also indicate how well the procedure is proceeding and provide high-resolution documentation of what was done to help improve future treatments. More generally, this hybrid control illustrates an important approach to using self-organized systems: use local, distributed control to achieve robust self-organized behaviors on small scales in space and time, combined with feedback from a slower, larger central control (e.g., a person) to verify performance and consider global constraints not easily incorporated within local controllers.

The performance estimates for the sensing task show devices with limited capabilities—specifically, without locomotion or communication with other devices—can nevertheless rapidly detect chemical sources as small as a single cell. The devices use their small size and large numbers to allow at least a few to get close to the source, where concentration is much higher than background. This paper also illustrates use of an analysis technique for average behavior of microscopic robots that readily incorporates spatially variable fields in the environment. Such fields are of major significance for microscopic robots, in contrast to their usual limited importance for larger robots.

As a caveat on the results, the model examined here treats the location of the chemical source as independent of the properties and flow rates in the vessel containing the source. Systematic variation in the density and organization of the vessels will increase the variation in detected values. For instance, the tissue could have correlations between vessel density and the chemical sources (e.g., if those chemicals enhance or inhibit growth of new vessels). Accurate inference requires models of how the chemicals move through the tissue to nearby blood vessels. Chemicals could react after release from the source to change the concentrations with distance from the source. Furthermore, other chemicals, unrelated to the desired detection task, may have similar sensor binding characteristics as the desired chemical. These other chemicals will increase the effective background rate, depending on the selectivity of the sensor. Nevertheless, the simple model discussed here indicates the devices could have high discrimination for sources as small as single cells. Thus even with some unmodeled sources of variability, good performance could still be achieved by extending the sensing time or using more sophisticated inference methods. Moreover, with some localization during operation, the devices themselves could estimate some of this variation (e.g., changes in density of vessels in different tissue regions), and these estimates could improve the inference instead of relying on average or estimated values for the tissue structure.

Further open questions include the effect of higher diffusion from mixing due to motion of cells in fluid, for both chemicals and robots. For instance, the hydrodynamic effect of blood cells moving in the fluid greatly increases the diffusion coefficient of smaller objects in the fluid, to about 10^{-9} m^2/s (Keller 1971).

Instead of flowing with fluid, the sensors could be implanted at specific locations of interest to collect data in their local environments, and later retrieved. This approach does not take full advantage of the sensor size: it could be difficult to identify interesting locations at cell-size resolutions and implant the devices accurately. Nevertheless, such implants could be useful by providing local signals to indicate regions of interest to other sensors passing nearby in a moving fluid.

Safety is important for medical applications of microscopic robots. Thus, evaluating a control protocol should consider its accuracy allowing for errors, failures of individual devices or variations in environmental parameters. For the simple distributed sensing discussed in this paper, statistical aggregation of many devices' measurements provides robustness against these variations, a technique recently illustrated using DNA computing to respond to patterns of chemicals (Benenson et al. 2004). Furthermore, the devices must be compatible with their biological environment (Nel et al. 2006) for enough time to complete their task. Appropriately engineered surface coatings and structures should allow sufficient biocompatibility during robot operation (Freitas 2003; Schrand et al. 2007). However, even if individual devices are inert, too many in the circulation would be harmful. From Table 8.1, sensors occupy a fraction $(4/3)\pi a^3 \rho_{\text{robot}} \approx 10^{-6}$ of the volume inside the vessels. This value is well below the fraction, about 10^{-3}, of micron-size particles experimentally demonstrated to be safely tolerated in the circulatory system of at least some mammals (Freitas 2003). Thus the number of robots used in the protocol of this paper is unlikely to be a safety issue.

Despite the simplifications used to model sensor behavior, the estimates obtained in this paper with plausible biophysical parameters show high-resolution sensing is possible with passive device motion in the circulatory system, even without communication capabilities. Thus relatively modest hardware capabilities could provide useful in vivo sensing capabilities far more flexible and specific than larger scale devices. Research studies of tissue microenvironments with such robots will improve knowledge of their biophysical parameters, and hence enable better inferences from the data collected by these devices. The improved understanding will, in turn, indicate distributed controls suitable for more capable devices and appropriate tradeoffs between scale and capability for hybrid systems combining coarse centralized control with the flexibility of self-organization within the biological microenvironments.

8.7 Epilogue

Microscopic robots present challenges for both hardware and software development. Recent years, since the first edition of this chapter (Hogg 2008), have seen progress in both areas.

As one example of hardware development, clinical magnetic resonance imaging (MRI) can be used to move microrobots containing ferromagnetic particles through blood vessels (Martel et al. 2007; Olamaei et al. 2010). Other demonstrated micromachines use flagellar motors to move through fluids, and may be useful for minimally invasive surgery in parts of the body beyond the reach of catheters (Behkam and Sitti 2007; Fernandes and Gracias 2009).

Modifying biological organisms or using their components is an important method for creating cell-sized robots. One approach to creating such robots is engineering biological systems, e.g., RNA-based logic inside cells (Win and Smolke 2008), bacteria attached to nanoparticles (Martel et al. 2008), executing simple programs via the genetic machinery within bacteria (Ferber 2004; Andrianantoandro et al. 2006), DNA computers responding to Boolean combinations of chemicals (Benenson et al. 2004) and artificial DNA-based structures capable of self-locomotion (Smith 2010; Sanchez and Pumera 2009; Douglas et al. 2012). In addition to specific examples of microscopic devices, this work improves our engineering understanding of mechanical and chemical mechanisms used in biology. This is important for guiding designs of new devices since molecular machines operate in environments quite different from conventional machines, particularly including significant thermal fluctuations and the dominance of viscous over inertial forces (Nelson 2008).

This progress is impressive and is expanding our capabilities to engineer microscopic machines that include sensing, computing and acting on their environments. These are three key aspects of autonomous robots. Nevertheless, these recently demonstrated devices remain a long way from a fabrication technology able to make robots with significant capabilities beyond biological cells. These extensions including using stronger materials, having significant on-board programmable computation and communicating with other robots and human supervisors of the robots' operations.

Computational studies are investigating performance and design choices for robots that cannot yet be built. Such studies include how much power microscopic robots could generate with oxyglucose fuel cells (Mano et al. 2002; Rapoport et al. 2012) using available glucose and oxygen in tissue (Hogg and Freitas 2010). A variety of method for powering MEMS devices (Cook-Chennault et al. 2008) may suggest additional approaches to powering smaller robots. Power is likely to be a significant constraint on such machines, so quantifying possible power sources is important for constraining designs and applications of the robots. The development of methods to deploy networks of small sensors (Mahfuz and Ahmed 2005) could also eventually apply to more sophisticated machines. Communication is another important capability by which robots can perform a wide range of coordinated activities, and respond to commands from human controllers. Numerical and analytic models can determine the tradeoffs among range, bit rate and power consumption, e.g., for acoustic communication among the devices (Hogg and Freitas 2012).

In the near term, we can expect continued progress on fabricating novel nanoparticles with some ability to determine actions through a small number of logical operations, and more capable biology-based devices. These developments provide practical constraints for computational studies of distributed control using such devices. Such studies can help design swarm behaviors in settings corresponding to environments of microscopic robots, particularly thermal fluctuations, rapid diffusion as the main transport mode over short distances and viscous fluids. These improving hardware and distributed control designs could pave the way to fabrication and use of larger collections of more capable microscopic robots.

Acknowledgements I have benefited from discussions with Philip J. Kuekes and David Sretavan.

References

Alon, U. (2007). *An introduction to systems biology: design principles of biological circuits*. London: Chapman and Hall.

Andrianantoandro, E., Basu, S., Karig, D. K., & Weiss, R. (2006). Synthetic biology: new engineering rules for an emerging discipline. *Molecular Systems Biology*, *2*, 2006.0028. doi:10.1038/msb4100073.

Arbuckle, D., & Requicha, A. A. G. (2004). Active self-assembly. In *Proceedings of the IEEE international conference on robotics and automation*, New York (pp. 896–901).

Behkam, B., & Sitti, M. (2007). Bacterial flagella-based propulsion and on/off motion control of microscale objects. *Applied Physics Letters*, *90*, 023902.

Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., & Shapiro, E. (2004). An autonomous molecular computer for logical control of gene expression. *Nature*, *429*, 423–429.

Berg, H. C. (1993). *Random walks in biology* (2nd ed.). Princeton: Princeton Univ. Press.

Berg, H. C., & Purcell, E. M. (1977). Physics of chemoreception. *Biophysical Journal*, *20*, 193–219.

Berna, J., et al. (2005). Macroscopic transport by synthetic molecular machines. *Nature Materials*, *4*, 704–710.

Bojinov, H., Casal, A., & Hogg, T. (2002). Multiagent control of modular self-reconfigurable robots. *Artificial Intelligence*, *142*, 99–120. arXiv:cs.RO/0006030.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. Oxford: Oxford University Press.

Casal, A., Hogg, T., & Cavalcanti, A. (2003). Nanorobots as cellular assistants in inflammatory responses. In J. Shapiro (Ed.), *Proceedings of the 2003 Stanford biomedical computation symposium (BCATS2003)*, Stanford, CA (p. 62). Available at http://bcats.stanford.edu.

Cavalcanti, A., & Freitas, R. A. Jr. (2002). Autonomous multi-robot sensor-based cooperation for nanomedicine. *International Journal of Nonlinear Sciences and Numerical Simulation*, *3*, 743–746.

Collier, C. P., et al. (1999). Electronically configurable molecular-based logic gates. *Science*, *285*, 391–394.

Cook-Chennault, K. A., Thambi, N., & Sastry, A. M. (2008). Powering MEMS portable devices—a review of non-regenerative and regenerative power supply systems with special emphasis on piezoelectric energy harvesting systems. *Smart Materials and Structures*, *17*, 043001.

Craighead, H. G. (2000). Nanoelectromechanical systems. *Science*, *290*, 1532–1535.

Dhariwal, A., Sukhatme, G. S., & Requicha, A. A. G. (2004). Bacterium-inspired robots for environmental monitoring. In *Proceedings of the IEEE international conference on robotics and automation*, New York (pp. 1436–1443).

Douglas, S. M., Bachelet, I., & Church, G. M. (2012). A logic-gated nanorobot for targeted transport of molecular payloads. *Science*, *335*, 831–834.

Drexler, K. E. (1992). *Nanosystems: molecular machinery, manufacturing, and computation*. New York: Wiley.

Dreyfus, R., et al. (2005). Microscopic artificial swimmers. *Nature*, *437*, 862–865.

Ferber, D. (2004). Microbes made to order. *Science*, *303*, 158–161.

Fernandes, R., & Gracias, D. H. (2009). Toward a miniaturized mechanical surgeon. *Materials Today*, *12*(10), 14–20.

Freitas, R. A. Jr. (1999). *Nanomedicine, volume I: basic capabilities*. Georgetown: Landes Bioscience. Available at www.nanomedicine.com/NMI.htm.

Freitas, R. A. Jr. (2003). *Nanomedicine, volume IIA: biocompatibility*. Georgetown: Landes Bioscience. Available at www.nanomedicine.com/NMIIA.htm.

Freitas, R. A. Jr. (2006). Pharmacytes: an ideal vehicle for targeted drug delivery. *Journal of Nanoscience and Nanotechnology*, *6*, 2769–2775.

Fritz, J., et al. (2000). Translating biomolecular recognition into nanomechanics. *Science*, *288*, 316–318.

Fung, Y. C. (1997). *Biomechanics: circulation* (2nd ed.). New York: Springer.

Galstyan, A., Hogg, T., & Lerman, K. (2005). Modeling and mathematical analysis of swarms of microscopic robots. In P. Arabshahi & A. Martinoli (Eds.), *Proceedings of the IEEE swarm intelligence symposium (SIS2005)*, New York (pp. 201–208).

Gazi, V., & Passino, K. M. (2004). Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man and Cybernetics. Part B. Cybernetics*, *34*, 539–557.

Ghosh, S., et al. (2003). Carbon nanotube flow sensors. *Science*, *299*, 1042–1044.

Gourley, P. L., et al. (2005). Ultrafast nanolaser flow device for detecting cancer in single cells. *Biomedical Microdevices*, *7*, 331–339.

Griffith, S., Goldwater, D., & Jacobson, J. M. (2005). Robotics: self-replication from random parts. *Nature*, *437*, 636.

Hamad-Schifferli, K., et al. (2002). Remote electronic control of DNA hybridization through inductive coupling to an attached metal nanocrystal antenna. *Nature*, *415*, 152–155.

Hernandez-Ortiz, J. P., Stoltz, C. G., & Graham, M. D. (2005). Transport and collective dynamics in suspensions of confined swimming particles. *Physical Review Letters*, *95*, 204501.

Hogg, T. (2007). Coordinating microscopic robots in viscous fluids. *Autonomous Agents and Multi-Agent Systems*, *14*(3), 271–305.

Hogg, T. (2008). Distributed control of microscopic robots in biomedical applications. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (1st ed.). London: Springer.

Hogg, T., & Freitas, R. A. Jr. (2010). Chemical power for microscopic robots in capillaries. *Nanomedicine*, *6*, 298–317. arXiv:0906.5022.

Hogg, T., & Freitas, R. A. Jr (2012). Acoustic communication for medical nanorobots. *Nano Communication Networks*, *3*, 83–102.

Hogg, T., & Huberman, B. A. (2004). Dynamics of large autonomous computational systems. In K. Tumer & D. Wolpert (Eds.), *Collectives and the design of complex systems* (pp. 295–315). New York: Springer.

Hogg, T., & Kuekes, P. J. (2006). Mobile microscopic sensors for high-resolution in vivo diagnostics. *Nanomedicine*, *2*, 239–247.

Hogg, T., & Sretavan, D. W. (2005). Controlling tiny multi-scale robots for nerve repair. In *Proceedings of the 20th national conference on artificial intelligence (AAAI2005)* (pp. 1286–1291). Menlo Park: AAAI Press.

Howard, J. (1997). Molecular motors: structural adaptations to cellular functions. *Nature*, *389*, 561–567.

Janeway, C. A., et al. (2001). *Immunobiology: the immune system in health and disease* (5th ed.). New York: Garland.

Karniadakis, G. E. M., & Beskok, A. (2002). *Micro flows: fundamentals and simulation*. Berlin: Springer.

Keller, K. H. (1971). Effect of fluid shear on mass transport in flowing blood. In *Proceedings of Federation of American Societies for Experimental Biology* (pp. 1591–1599).

Keszler, B. L., Majoros, I. J., & Baker, J. R. Jr. (2001). Molecular engineering in nanotechnology: structure and composition of multifunctional devices for medical application. In *Proceedings of the ninth foresight conference on molecular nanotechnology*, Palo Alto, CA.

Lahann, J., & Langer, R. (2005). Smart materials with dynamically controllable surfaces. *MRS Bulletin*, *30*, 185–188.

Lerman, K., et al. (2001). A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life*, *7*, 375–393.

Li, Z., et al. (2005). Silicon nanowires for sequence-specific DNA sensing: device fabrication and simulation. *Applied Physics. A, Materials Science & Processing*, *80*, 1257–1263.

Liu, J., et al. (2006). Nanoparticles as image enhancing agents for ultrasonography. *Physics in Medicine and Biology*, *51*, 2179–2189.

Mahfuz, M. U., & Ahmed, K. M. (2005). A review of micro-nano-scale wireless sensor networks for environmental protection: prospects and challenges. *Science and Technology of Advanced Materials*, *6*, 302–306.

Mano, N., Mao, F., & Heller, A. (2002). A miniature biofuel cell operating in a physiological buffer. *Journal of the American Chemical Society*, *124*, 12962–12963.

Martel, S., Mathieu, J.-B., Felfoul, O., Chanu, A., Aboussouan, E., Tamaz, S., & Pouponneau, P. (2007). Automatic navigation of an untethered device in the artery of a living animal using a conventional clinical magnetic resonance imaging system. *Applied Physics Letters*, *90*, 114105.

Martel, S., et al. (2008). Flagellated bacterial nanorobots for medical interventions in the human body. In D. Meldrum & O. Khatib (Eds.), *Proceedings of 2nd IEEE conference on biomedical robotics and biomechatronics* (pp. 264–269).

Mataric, M. (1992). Minimizing complexity in controlling a mobile robot population. In *Proceedings of the 1992 IEEE intl. conf. on robotics and automation*, New York (pp. 830–835).

McAdams, H. H., & Arkin, A. (1997). Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Sciences of the United States of America*, *94*, 814–819.

McCurdy, C. W., et al. (2002). *Theory and modeling in nanoscience* (Workshop report). US Dept. of Energy. www.science.doe.gov/bes/reports/files/tmn_rpt.pdf.

Montemagno, C., & Bachand, G. (1999). Constructing nanomechanical devices powered by biomolecular motors. *Nanotechnology*, *10*, 225–231.

Morris, K. (2001). Macrodoctor, come meet the nanodoctors. *The Lancet*, *357*, 778.

Natterer, F. (2001). *The mathematics of computerized tomography*. Philadelphia: SIAM.

Nel, A., et al. (2006). Toxic potential of materials at the nanolevel. *Science*, *311*, 622–627.

Nelson, P. (2008). *Biological physics: energy, information, life*. New York: W.H. Freeman.

NIH (2003). National Institutes of Health roadmap: nanomedicine. Available at http://nihroadmap.nih.gov/nanomedicine/index.asp.

Olamaei, N., Cheriet, F., Beaudoin, G., & Martel, S. (2010). MRI visualization of a single 15 μm navigable imaging agent and future microrobot. In *Proceedings of the 2010 conf. on engineering in medicine and biology society* (pp. 4355–4358). New York: IEEE.

Patolsky, F., & Lieber, C. M. (2005). Nanowire nanosensors. *Materials Today*, *8*, 20–28.

Purcell, E. M. (1977). Life at low Reynolds number. *American Journal of Physics*, *45*, 3–11.

Rapoport, B. I., Kedzierski, J. T., & Sarpeshkar, R. (2012). A glucose fuel cell for implantable brain-machine interfaces. *PLoS ONE*, *7*(6), e38436.

Requicha, A. A. G. (2003). Nanorobots, NEMS and nanoassembly. *Proceedings of the IEEE*, *91*, 1922–1933.

Riedel, I. H., et al. (2005). A self-organized vortex array of hydrodynamically entrained sperm cells. *Science*, *309*, 300–303.

Rus, D., & Vona, M. (1999). Self-reconfiguration planning with compressible unit modules. In *Proceedings of the conference on robotics and automation (ICRA99)* (pp. 2513–2520). New York: IEEE.

Salemi, B., Shen, W.-M., & Will, P. (2001). Hormone controlled metamorphic robots. In *Proceedings of the international. conference on robotics and automation (ICRA2001)*, New York (pp. 4194–4199).

Sanchez, S., & Pumera, M. (2009). Nanorobots: the ultimate wireless self-propelled sensing and actuating devices. *Asian Journal of Chemistry*, *4*, 1402–1410.

Schrand, A. M., et al. (2007). Are diamond nanoparticles cytotoxic? *Journal of Physical Chemistry. B*, *111*, 2–7.

Service, R. F. (2005). Nanotechnology takes aim at cancer. *Science*, *310*, 1132–1134.

Sheehan, P. E., & Whitman, L. J. (2005). Detection limits for nanoscale biosensors. *Nano Letters*, *5*(4), 803–807.

Smith, L. M. (2010). Molecular robots on the move. *Nature*, *465*, 167–168.

Soong, R. K., et al. (2000). Powering an inorganic nanodevice with a biomolecular motor. *Science*, *290*, 1555–1558.

Squires, T. M., & Quake, S. R. (2005). Microfluidics: fluid physics at the nanoliter scale. *Reviews of Modern Physics*, *77*, 977–1026.

Sretavan, D., Chang, W., Keller, C., & Kliot, M. (2005). Microscale surgery on axons for nerve injury treatment. *Neurosurgery*, *57*(4), 635–646.

Vogel, S. (1994). *Life in moving fluids* (2nd ed.). Princeton: Princeton Univ. Press.

Wang, Z. L., & Song, J. (2006). Piezoelectric nanogenerators based on zinc oxide nanowire arrays. *Science*, *312*, 242–246.

Wang, S.-Y. & Williams, R. S. (Eds.) (2005). *Nanoelectronics* (Vol. 80). New York: Springer. Special issue of Applied Physics A.

Wang, H., et al. (2005). In vitro and in vivo two-photon luminescence imaging of single gold nanorods. *Proceedings of the National Academy of Sciences of the United States of America*, *102*, 15752–15756.

Whitesides, G. M., & Grzybowski, B. (2002). Self-assembly at all scales. *Science*, *295*, 2418–2421.

Win, M. N., & Smolke, C. D. (2008). Higher-order cellular information processing with synthetic RNA devices. *Science*, *322*, 456–460.

Xie, X. S., Yu, J., & Yang, W. Y. (2006). Living cells as test tubes. *Science*, *312*, 228–230.

Part III
Self-Organizing Computation

Chapter 9
Self-Organizing Computing Systems: Songline Processors

Nicholas J. Macias and Lisa J.K. Durbeck

9.1 Introduction

Theory is at the threshold of understanding how to translate self-organizing principles and processes to human-formed systems. However, practice lags behind theory. This chapter endeavors to provide inroads into the application of self-organization principles to one aspect of electronics systems, namely, digital logic.

Digital circuitry proliferated from the early transistor-Transistor Logic (TTL) circuits of the 1960s to the now mass markets of computers, mobile phones, T.V.s, and numerous other consumer products. The fundamental component of digital circuitry is the logic gate from which complex functions can be derived and explained with the use of digital logic. Due to its widespread use and complex application, digital logic is arguably a good target for applying concepts from self-organizing systems.

The ultimate goal of applying self-organization concepts to digital logic is to devise theory and practice as to how digital logic could be constructed and operated as a self-organizing system. Our approach has been to devise reconfigurable logic hardware and an architecture that permits self-organizing processes, and then to begin methodically developing self-organization concepts and their translation to practice within this framework. This work requires changing the way digital logic is both designed and built, providing so-called *primitives*, or fundamental behaviors, for self-organizing systems, along with a way to build upon these primitives to conceive of, compose, and orchestrate self-organized digital logic.

To achieve an inherently self-organizing infrastructure, a number of departures from conventional digital logic design are required. These include:

- reworking how the system is controlled by placing the control within the componentry of the system itself, that is, if the system is composed entirely of digital

N.J. Macias (✉) · L.J.K. Durbeck
Cell Matrix Corporation, 1901 Gardenspring Drive, Blacksburg, VA 24060-6015, USA
e-mail: nmacias@cellmatrix.com

L.J.K. Durbeck
e-mail: ld@cellmatrix.com

logic, then its digital logic must have the ability to control digital logic, creating a new class of digital logic that is able to dynamically change, and able to modify both itself and its neighbors;

- the need to incorporate this self-inspecting, self-modifying power into systems without again introducing the hierarchy of controller and controlled, so that they are both loosely onto each other, or interchangeable; and
- the need to conceive of and develop strategies that build upon these core capabilities to re-conceive system monitoring and control as a distributed process largely enacted within the confines of the system itself, and composed of many simple, localized activities with significant autonomy in their ability to decide and act on local information.

The work described below will further ground this discussion of objectives and insights with concrete examples as to how these properties are included in the hardware architecture we are developing, and gives some insight by example as to how these simple, foundational or underlying processes can be used to compose digital logic that is self-organizing and dynamically self-modifying.

As this is a new approach digital logic design, it is unlikely that we have developed all the primitives necessary for every class of problem. We therefore anticipate that as we apply this work to more classes of problems, the need for other primitives is likely. We have also not yet developed the full set of useful mid-level behaviors, built from the primitives, that are likely to be necessary for self-organizing digital logic designs. However, we report here on the current state of the art and outline areas in which this work will be further applied.

Several separate aspects of digital logic production may benefit from the application of principles of self-organization, both in the structure and function of digital logic circuits. In the case of an FPGA, a self-organizing process could be used to fabricate the physical hardware; the primitive functions of the hardware could use and enable self-organization; and any logical level could do the same for the logical layer above it by supplying primitives that the upper layers can employ. In this chapter we present research done on the design of the FPGA and its low level logic behavior to develop self-organizing primitives that can be used to structure the logical levels above it, or can be invoked by those upper logical levels. We view this as foundational work toward the eventual integration of self-organizing behavior into digital logic.

A key aspect of design at the hardware architecture and low-level system structure level is the data path and the control path of the computational architecture. Much of the discussion below will refer to the system *control*. This should be understood to be all those processes that direct the operation of the overall system, such as those scheduling activities or processes, synchronizing the actions of disparate processes, and maintaining the overall system and all subprocesses of it in proper working order.

System maintenance is currently done largely by people rather than processes on the machine because of the need for physical action in many cases such as the replacement of failing memory cards and disk drives. However, in concept, the act of inspecting the equipment for failure can be integrated into the design of the processes that run on the equipment. When systems scale upward to the point that they contain 10^{17} or more logical devices, there will likely be sufficient incentive to reorganize hardware inspection as a distributed, localized process as well, on account of the unavoidably high frequency of hardware upsets. Similarly, the process of inspecting the initial constructed hardware for defects may also become tedious enough that it has similar incentive to reorganize hardware inspection as a distributed, localized process running on the hardware itself.

To invest the low level architecture of digital logic systems with properties of self-organizing systems, it appears to be critical to change the typical computer control path into one that is based instead upon strictly local interactions, and to then conceive of the overall control path as being a complex distributed process that emerges out of many local control actions. Our work provides this new kind of control path, and we detail a number of examples of how it is used to recast control as a highly localized process, and then describe examples in which we have built up increasingly complex systems that are composed out of these small, highly localized processes. Management of the actions of processes is thus transformed from the typical centralized control of a manager that is making decisions and controlling the system outside the system itself to distributed management and control.

Self-organization may be an antidote to the fact that the complexity of controlling and managing systems has been at least proportional to the size of the system, the number of components under management. Managing 10^{18} components using traditional methodology does not appear to be a tenable proposition. A hypothesis that underlies the work presented here is that approaches to deal with the complexity of a very large system likely require at least an equally large system as the manager, and that it may be preferable that the manager be not separate but integral with the system under management, since, for example, the details requiring management decisions come down to a very large number of small details that may be extremely difficult to output every nanosecond. Achieving this co-functioning of manager and process under management appears possible with a particular type of infrastructure to the underlying system, one that combines sufficient flexibility with an inherently self-referential structure that makes introspection and autonomous self-modification feasible. The architecture presented here has the necessary properties to test this hypothesis. Section 9.2 will describe a particular system called the Cell Matrix (Macias 1999; Durbeck and Macias 2001d) that possesses these necessary characteristics.

9.1.1 Background on the Concept of Self-Organization

The concept of self-organization has application to a number of domains. In the domain of living systems, self-organization is a central theme in many areas (Darwin 1859; Kauffman 1993). Philosophical considerations date back even further (Haldane 1931; Lennox 2001). In the domain of artificial systems, many facets of self-organization have been explored, including autonomous behavior and self-repair (Aspray and Burks 1987).

Approaches to self-organization can be grouped into at least two main categories: one we will call a *statistical* approach, and the other an *engineered* approach. Statistical approaches seek to manage complexity by *discovering* algorithms or techniques. Such approaches are not necessarily concerned with *how* the given task is accomplished, only with how well it is presently being accomplished. Examples of statistical approaches include neural networks (e.g., Abdi 1994) and genetic algorithms (e.g., Koza 1992). Genetic algorithms have been extensively applied to the development of electronic circuits, in a field called Evolvable Hardware (e.g., Thompson 1996). It should be noted that much of this work draws inspiration from biology (Darwin 1859).

In contrast to statistical approaches, engineered approaches seek to more-deliberately achieve some set of goals by following more of a pre-defined algorithm. Interestingly, many engineered approaches also draw inspiration from biology, including the Electronic Embryology (Embryonics) work of LSL (Prodan et al. 2003; Ortega-Sanchez et al. 2000), and the Supercell work of Cell Matrix Corporation (Macias and Durbeck 2004). The discussion in this chapter falls into the domain of the engineered approach.

9.1.2 Chapter Organization

The remainder of this chapter is organized as follows: Sect. 9.2 will describe in detail a particular target processing architecture called the Cell Matrix, which possesses an inherently self-organizing infrastructure; Sect. 9.3 will describe simple examples of self-modifying circuitry on the Cell Matrix, which will form the building blocks for larger-scale self-organizing systems; Sect. 9.4 will discuss such larger systems; and Sect. 9.6 will conclude with discussions of implementation details, including manufacturing and CAD issues related to the Cell Matrix.

9.2 Target Platform: The Cell Matrix

This chapter discuses the distributed management and control of electronic circuitry implemented on a specific reconfigurable platform called the Cell Matrix (Macias 1999; Durbeck and Macias 2001d). While there are a number of commercially-available reconfigurable devices (*Field Programmable Gate Arrays*, or FPGAs), including many from Xilinx, Inc., most available devices are essentially externally-controlled, i.e., they require intervention from an outside system (e.g., a PC) in order to be configured or re-configured (Xilinx 2006). Moreover, even among devices that can hold several simultaneous configurations (Trimberger 1998), those configurations are generally pre-created, externally, again using a PC or other extra-FPGA system. In contrast, the Cell Matrix is fundamentally an *internally*-configured device: the configuration of each cell is written—and read—by those cells connected

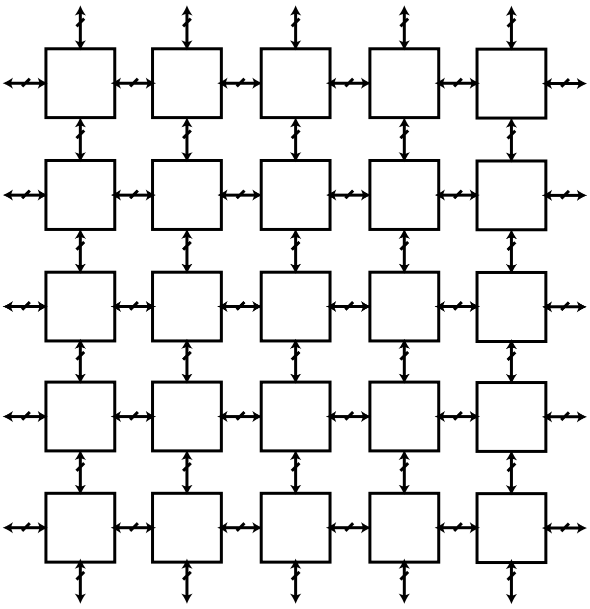


Fig. 9.1 5 × 5 collection of two-dimensional, four-sided Cell Matrix cells

to it, its immediate adjacent neighbors. Only cells situated on the perimeter of the matrix (which are thus missing one or more neighbors) are accessible from outside the system. This is fundamentally different from most other devices, where typically *every* cell can be accessed from outside the system by simply sending a long configuration string throughout the device.

While it may seem unusual, and perhaps disadvantageous, to have such limited access to cells from outside the system, this is in fact a critical characteristic of the Cell Matrix, and is directly linked to its ability to implement autonomous, self-organizing circuitry. Having local-only cell control also allows the system to scale, without specific regard for scaling the control structures.

9.2.1 Basic Cell Structure

A Cell Matrix is a regularly-tiled collection of simple reconfigurable elements called *cells*. These cells are arranged in a fixed, identical topology throughout the matrix, and that topology defines a notion of a cell's *neighbors*: the neighbors of a cell "X" are all those cells that are immediately connected to X. Each cell receives a single input bit (called its "D Input") from each of its neighbors, and generates a single output bit (its "D Output") to each of those neighbors. Figure 9.1 shows a two-dimensional collection of four-sided cells. In this topology, each cell has four immediate neighbors. We will mainly be discussing two-dimensional, four-sided cells in this chapter. However, (useful) two-dimensional cells can have as few as three sides, or may have more than four, though four is the most typical number.

Cells can also be three-dimensional, having as few as four sides, but more typically six. Higher-dimensional cells are also possible, though anything higher than three dimensions ceases to be (architecturally) infinitely-scalable because there is no way to organize the cells topologically that puts all neighbors a finite, very small distance from each other.

9.2.2 Cell Structure

Each cell contains a small memory which stores a *truth table*. The truth table maps input combinations to outputs: given the set of incoming bits from all of a cell's neighbors, the cell's outputs are precisely determined by the information in the cell's truth table. This mechanism allows a single cell to implement simple combinatorial functions, such as basic logic gates, single-bit adders or multiplexers. Cells can also act as simple wire: a block for passing data from one side of itself to another—or, viewed differently, a block for allowing two non-adjacent cells to share data with each other. Implementing wires is a major use of cells.

9.2.3 Cell Configuration

The act of loading truth table information into a cell is called *cell configuration* or "configuring a cell." Similarly, the act of loading truth table information into a number of cells is called "Configuring the Cell Matrix." While single-cell circuits are necessarily extremely simplistic, configuring a group of cells appropriately leads to multi-cell circuits, which can be arbitrarily complex. Because single cells can implement any fixed input-to-output mapping, and cells can be interconnected via intervening cells, any circuitry that can be implemented using traditional digital circuit design can also be implemented on a Cell Matrix.

Figure 9.2 shows a more-detailed view of a single cell. As can be seen, each side has two input lines and two output lines, which connect it to each of its immediately-adjacent neighbors. One line is labeled "D" and the other "C." The D inputs are used to select information from the cell's truth table, and the D outputs are set based on the truth table's values, as described above. *But this is the case only if all the C inputs are 0.*

When all C inputs are 0, the cell is said to be in "D" mode. If, however, any C inputs are set to 1, then the cell is in "C" mode. C mode is the configuration mode of a cell: it is the mode in which a cell's truth table can be modified. In C mode, a cell's truth table can also be examined. A cell that is asserting one of its own C outputs, and is thus asserting a neighboring cell's C input, is able to read and write that neighboring cell's truth table. Note that if more than one C input is set to 1, then the cell's truth table is sent to multiple neighbors, and its new truth table is determined by a combination (logical ORing) of its neighbors outputs.

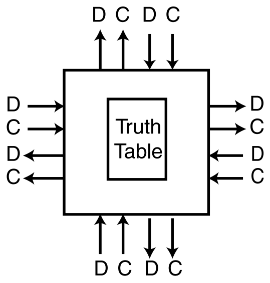


Fig. 9.2 Four-sided Cell Matrix cells. Each neighboring cell reads and writes two bits (D and C). C inputs determine the mode of the cell. D inputs either select C and D output values from the Truth Table (in D mode) or supply new values for the Truth Table (in C mode)

This C-mode operation is like the *unit measure* of self-organization for the entire architecture; using it, cells are configured by a neighboring cell. The extreme locality of this operation makes the entire architecture fine-grained in its reconfigurability, and makes configuration a distributed, local process.

Cell configuration is the only inherently clocked operation in the Cell Matrix: a single system-wide clock is used to serially shift out the current contents of a cell's truth table, and to serially shift in new truth table bits. These bits are read-from and written-to the D output and input lines, respectively, on the same side on which the cell's C input is asserted (called the *active side*). Figure 9.3 shows an example of adjacent-cell interactions in D and C modes. In Fig. 9.3a, Cell Y is in D mode, since all of its C inputs are 0. It thus uses its four D inputs to select a single row in the 16×8 truth table memory, and sends the selected eight output values to its eight outputs (four C and four D).

In Fig. 9.3b, one of Cell Y's C input is asserted (the one supplied by Cell X). This places Cell Y into C mode, the mode in which its truth table is read and written. Each time the system-wide clock ticks, the D input supplied by Cell X is loaded into Cell Y's truth table, at a position that changes with each tick (in Fig. 9.3b, the bit in the third row, second column is being written). Additionally, the previous value stored in that location is made available on the D output to Cell X. All other D outputs from Cell Y are forced to 0, as are all of Cell Y's C outputs (so that a cell being configured cannot itself simultaneously configure another cell). By convention, if two or more of a cell's C inputs are asserted, the bit value loaded into the cell's truth table is the logical OR of the D inputs on all active sides.

Using this simple interaction scheme, it is possible for any cell to read and write any neighboring cell's truth table. Since cells along the edge of the matrix have some of their inputs and outputs unconnected, as shown in Fig. 9.4, those edge cells can be configured from outside the matrix, if their C and D inputs (and, perhaps, outputs) are made available. In Fig. 9.4, all edge cells have their inputs and outputs accessible from the edge of the matrix on at least one cell side (corner cells are accessible from two sides).

Figures 9.5, 9.6, and 9.7 show the full details of a cell configuration operation: Fig. 9.5 shows a single cell configured as a NOR gate; Fig. 9.6 shows the cell's corresponding truth table; and Fig. 9.7 shows a timing diagram for configuring the cell. The cell is first placed into C mode by raising one of its C inputs. As soon as the cell enters C mode, the current value of the cell's first truth table bit is sent to

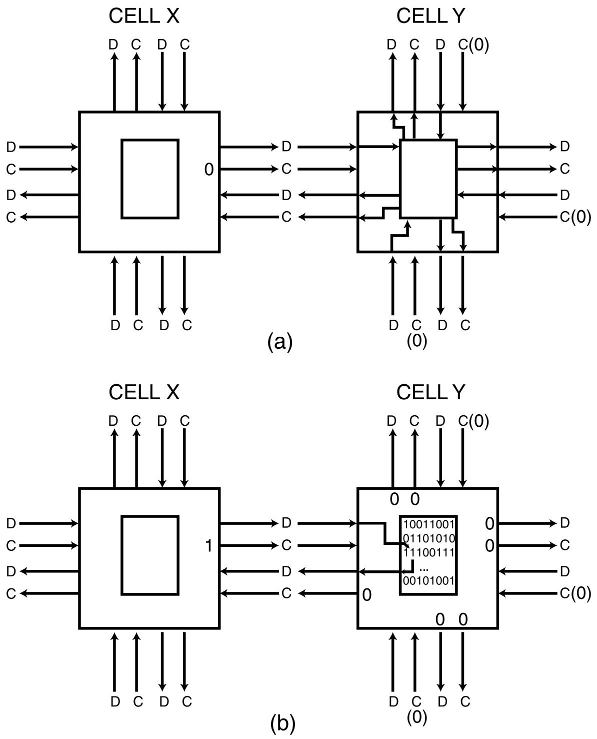


Fig. 9.3 The two mode of cell operation. In (**a**), cell Y is in D Mode (all C inputs are 0). Its four incoming D values are used to select 8 output values from its truth table. Those output values are sent to the cell's 8 output lines (4 C lines and 4 D lines). In (**b**), cell X is asserting a 1 to one of cell Y's C inputs, and thus cell Y is in C-mode. In this mode, the D input from Cell X supplies new values for Cell Y's truth table. Each time the system clock ticks, a new incoming bit value is sampled, and loaded into Cell Y's truth table. Cell Y's current truth table bits are simultaneously sent out the D output to cell X. In the figure above, the second bit in the third row of the truth table is being read and written by Cell X. All of Cell Y's other outputs (C and D) are forced to 0

the corresponding D output (not shown in the figure). On the next rising edge of the system clock, the D input is sampled and latched. On the next falling edge, the latched value is loaded into the truth table, and the current truth table's next bit is sent to the D output. Note that this timing makes the truth table's current bit values available on the D output half a cycle before the new bit value must be presented to the D input. This makes it simple to read a truth table bit and then re-write the same bit, thus performing a non-destructive read.

Before the 4th clock tick, the D input is raised. This "1" value is latched/loaded into the cell's truth table on the next rising/falling edge of the system clock. Similarly, a "1" is loaded during the 12th cycle following the cell's entry into C mode. All other incoming bit values are 0.

A few cycles later, the cell's C input is set to 0, and the cell returns to D mode. Assuming its truth table initially contained all 0s, its truth table is now as shown in Fig. 9.6.

9.2.4 Self-Configuration

Because cells are able to read and write other cells' truth tables, the Cell Matrix can be configured from *inside* the Cell Matrix itself. This makes the Cell Matrix

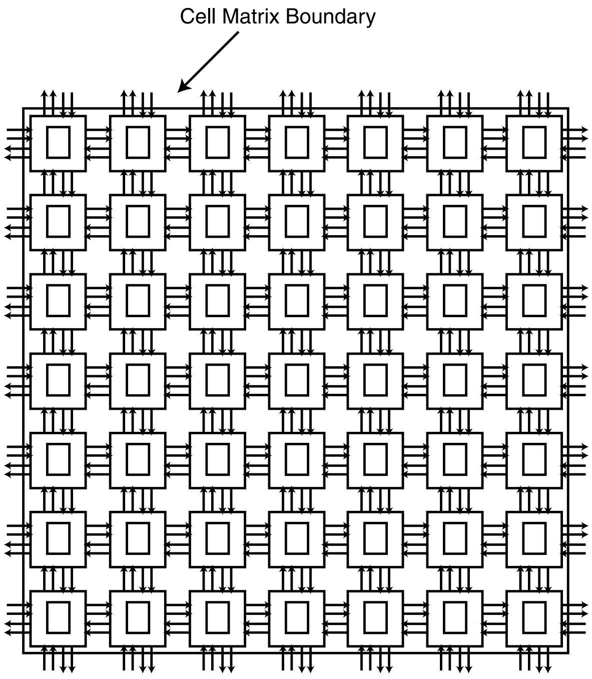


Fig. 9.4 7 × 7 Cell Matrix. Edge cells have their D and C inputs and outputs accessible from outside the matrix. Corner cells have I/O accessible on two of their sides

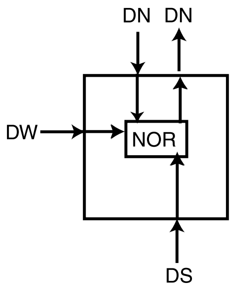


Fig. 9.5 Single cell implementing a three-input NOR gate

a *self-configurable* system, i.e., circuits can be constructed that read and write cell configurations, and thus can analyze and change circuitry within the matrix. Circuitry constructed on the Cell Matrix can process data that represents logical values, characters, integers, floating point numbers, or any sort of data structure. But additionally, circuitry constructed on the Cell Matrix can also process a unique type of data: circuit configuration information. And because the mapping between circuit configuration and circuit behavior is very straightforward, one can construct circuits that effectively *process other circuits*.

Moreover, there is no hardware-level or architectural difference between a cell that is being configured and the one that is configuring it. Figure 9.8 shows some of the possibilities resulting from this fact. In Fig. 9.8a, Cell X, Cell Y and Cell Z are all in D mode, since their C inputs are all 0 (all inputs are assumed to be 0 unless

Fig. 9.6 Truth Table corresponding to Fig. 9.5

INPUTS				OUTPUTS							
DN	DS	DW	DE	CN	CS	CW	CE	DN	DS	DW	DE
0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0

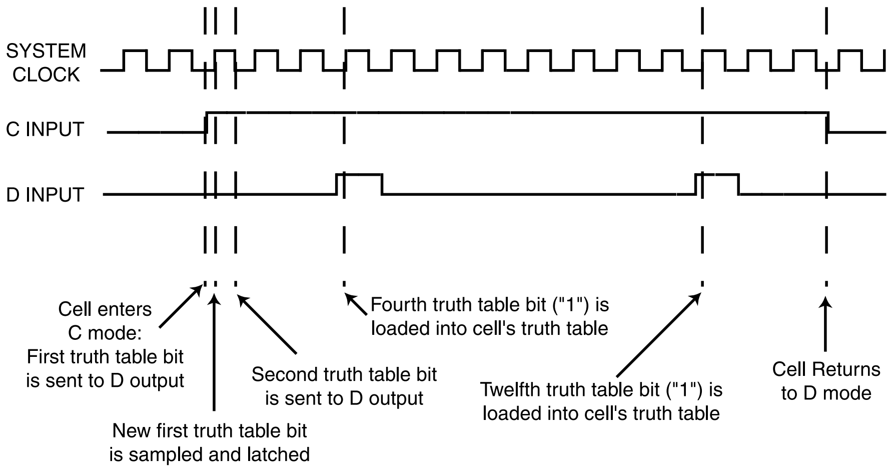

SYSTEM CLOCK
C INPUT
D INPUT
Cell enters C mode: First truth table bit is sent to D output
New first truth table bit is sampled and latched
Second truth table bit is sent to D output
Fourth truth table bit ("1") is loaded into cell's truth table
Twelfth truth table bit ("1") is loaded into cell's truth table
Cell Returns to D mode

Fig. 9.7 Truth Table programming sequence. After cell is placed into C mode, a “1” bit is loaded on the 4th and 12th ticks of the system clock. The cell is returned to D mode two ticks later. This loads 14 bits into the cell’s Truth Table: 0001 0000 0001 00

shown otherwise). Each cell is simply receiving D inputs, using them to address their internal truth table, and producing D and C outputs accordingly.

In Fig. 9.8b, Cell X is asserting a 1 on its C output to Cell Y. This places Cell Y into C mode. In this mode, Cell Y’s truth table is being configured by Cell X, by sampling the D outputs sent from Cell X to Cell Y.

In Fig. 9.8c, cell X has returned its C output to 0, and thus Cell Y returns to D mode. Cell Y is thus asserting its outputs based on the (new) contents of its truth table. In this example, the truth table indicates that Cell Y’s C output to Cell Z is to be set to 1. This places Cell Z into C-mode, and Cell Y is now configuring Cell Z.

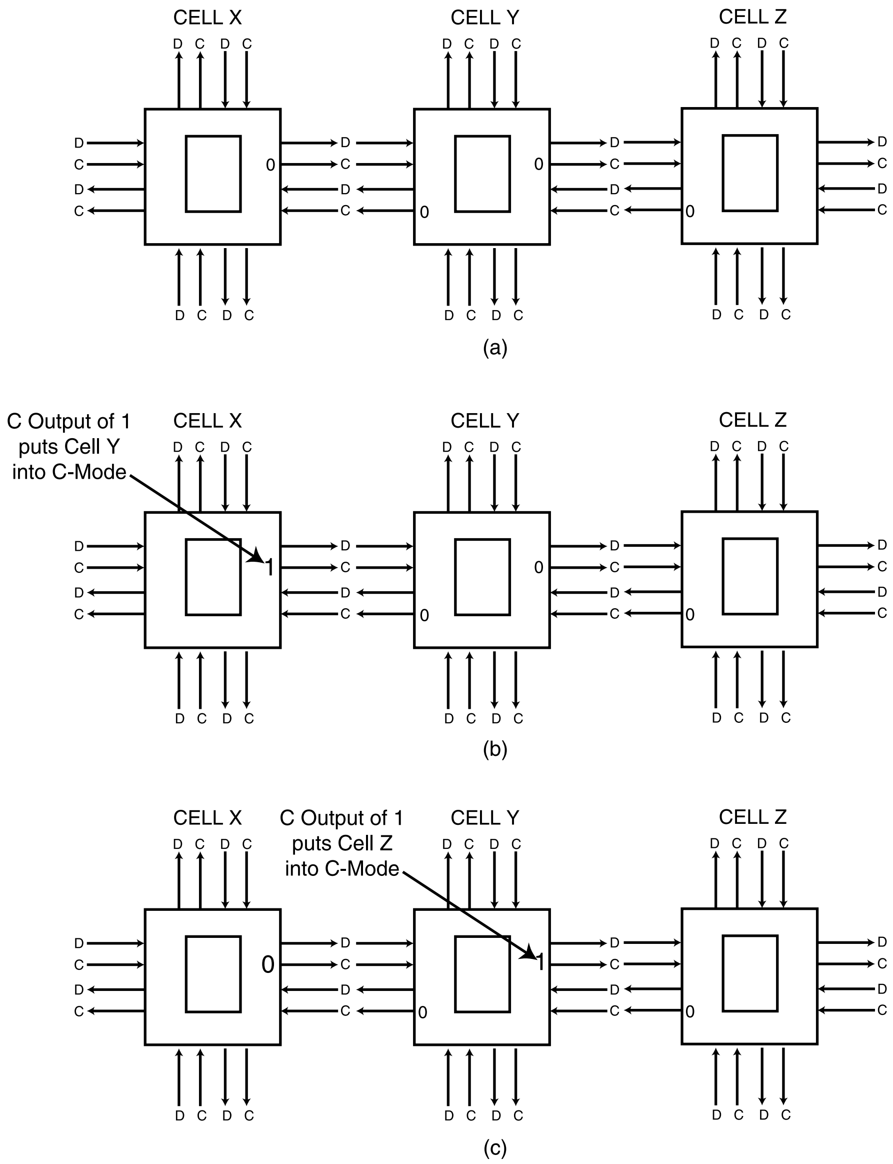

CELL X
CELL Y
CELL Z
D C D C
0
1
C Output of 1 puts Cell Y into C-Mode
C Output of 1 puts Cell Z into C-Mode
(a)
(b)
(c)

Fig. 9.8 Interaction of Cells' Modes. In (**a**), all three cells are in D Mode: each cell is reading inputs and producing outputs based on its current truth table contents and its D inputs. In (**b**), an input change (not shown) in cell X's D inputs has caused cell X to assert its C output to cell Y. Cell X has thus placed cell Y into C mode, and cell Y's truth table is now being configured. In (**c**), cell X is again outputting a 0 on its C output to cell Y. Cell Y has thus been returned to D mode. Based on cell Y's new truth table, cell Y is now asserting its C output to cell Z, and has thus placed cell Z into C mode. Cell Z is now being configured by cell Y

This example illustrates a very typical case: Cell Y was previously configured by a neighbor, but it is now itself configuring another neighbor. Within the Cell Matrix, there is a perfect interchangeability between subjects and objects of configuration operations. This is the essence of self-configuration and self-modification within the Cell Matrix.

9.2.5 Implications

There are a number of immediate implications arising from the architecture described above. The Cell Matrix architecture is infinitely scalable. Because only power and a single clock line are distributed throughout the matrix, there is no architectural impediment to scaling a matrix to whatever size is desired. Put another way (and, again, assuming a fixed dimensionality and interconnection topology), all sub-matrices of a given size are identical to each other, no matter what matrix they are embedded in: the structure of the cells and their interconnections is independent of the larger matrix to which they belong.

This means that two matrices can be combined into a large matrix simply by connecting the matrices to each other along an edge, i.e., connecting one matrix's edge cells' input to the other's edge cells' outputs, and vice versa. The architecture scales up without change (though of course the maximum possible latency increases). This also has interesting manufacturing implications (see Sect. 9.6).

Because configuration of cells is essentially a local operation, there is no such thing as *runtime vs. configuration time* for the matrix at large, no need to discuss *runtime reconfiguration*: the matrix is *always* running, and part of its running operation may include reconfiguration operations. Moreover, *partial configuration* (Schmit 1997) is the only type of configuration ever performed, since any single configuration operation affects only the neighbors of the cell being configured.

The Cell Matrix is completely homogeneous in structure. Cells are differentiated by their configuration information (truth table contents), but, at the underlying hardware level, all cells are identical to each other, just as are their interconnections to other cells. This has tremendously beneficial manufacturing implications. It also has positive implications for circuit reliability, since no piece of the matrix (or the circuits implemented on top of the matrix) is unique or irreplaceable.

Because of the capacity for self-modification in the Cell Matrix, high-level configuration mechanisms can be designed, tailored to the specifics of the target circuit, and then constructed out of cells. Moreover, the construction of the configuration mechanism *can itself be constructed* by using a previously-created configuration mechanism. In this way, configuring the Cell Matrix may closely resemble a traditional *bootstrap* process, wherein a simple circuit is first built using the limited control available from the edge of the matrix. This simple circuit is then used to configure a more complex circuit, which is then used to configure a more complex circuit, and so on, building more and more complex circuits until the desired configuration has been achieved. Also, note that while a single set of commands may be used repeatedly to configure multiple Cell Matrix regions, it is still possible to introduce *differentiation*, including randomness, into the configured circuits.

Finally, because cells are configured by neighboring cells, it is possible for multiple cells to be configured simultaneously, either with the same configuration as each other, or with completely different configurations.

9.2.6 Status

The Cell Matrix architecture has been fully documented (Cell Matrix Corporation 2006a, 2006b; Macias et al. 1999; Durbeck and Macias 2001c; Macias and Raju 2001). A variety of simulators and debuggers have been developed, as have various tools for developing circuitry on the matrix. Prototype tools for converting from abstract netlists to Cell Matrix configuration information have been developed (Macias 2006).

A number of fairly traditional circuits have been implemented on top of the Cell Matrix, including state machines, arithmetic units, memories, floating point processors, and cellular automata simulators. Section 9.4 will describe some of the less-traditional circuits that have been implemented, including circuits that utilize self-configuration.

Also, while the high-level behavior of the Cell Matrix is well-defined, there are multiple possible implementations of the Cell Matrix. For example, the original implementation (Macias et al. 1999) utilized a shift register for each cell's truth table. While this simplifies the design of each cell, it means that the entire truth table changes during a configuration operation. Later work produced a slightly more complicated cell implementation that utilizes a non-shifting memory that takes up a much smaller fabrication area (Durbeck and Macias 2001c). A further-modified cell incorporates bypass logic, to detect when a cell is acting as a wire and directly connect an input to an output, greatly improving signal transmission rates when cells are used as wires (Macias and Raju 2001).

9.3 Building Blocks of Self-Configuring Circuitry

This section will describe some of the *primitives* of self-configuration for the Cell Matrix, or the basic building blocks and techniques related to the implementation of self-configuring circuitry on the Cell Matrix. Section 9.4 will describe higher-level circuitry constructed from these blocks.

9.3.1 Cell-Replication

Figure 9.9 shows a simple cell-replication circuit that copies the truth table of the source cell into the target cell. The cell in the middle is the *controller* of the config-

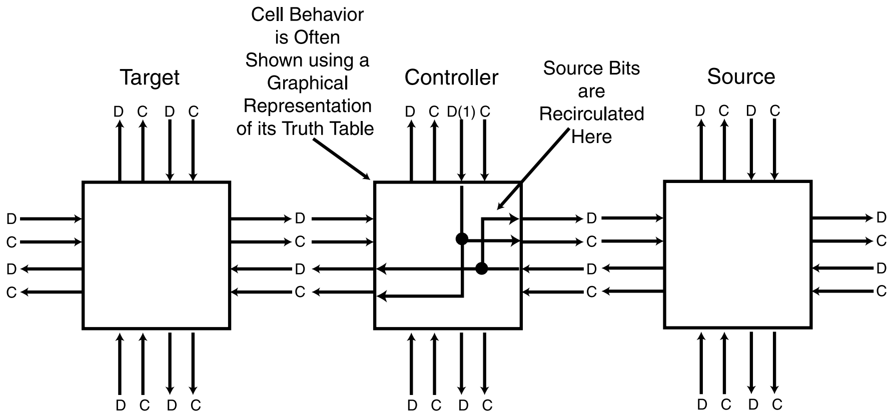

Target
Cell Behavior is Often Shown using a Graphical Representation of its Truth Table
Controller
Source Bits are Recirculated Here
Source
D C D C
D C D(1) C
D C D C
D
C
D
C
D C D C
D C D C
D C D C

Fig. 9.9 Single-Cell Replicator. When a 1 is sent into the Controller's Northern D input, it places and Source and Target Cells into C mode, reads truth table bits from the Source, copies them back to the Source, and also copies them to the Target. After 128 ticks of the system clock, the Target will be an exact copy of the source. Because the Source Cell's truth table bits are loaded back into the Source Cell's D input, the Source Cell's truth table is left unchanged by this circuit. This is thus a non-destructive read

uration operation. The cell to be replicated (called the *source cell*) is on the right. The *target cell*, which will become a copy of the source cell, is on the left. When a 1 is sent into the Northern D input of the controller, it asserts its C outputs on the left and right, thus placing the source and target cells into C mode. Each cell then begins outputting current truth table bits on one of its D outputs (on the side where C_{in} is asserted), as well as receiving new truth table bits on the same side's D input. The controller reads bits from the source cell, and sends them back into the source cell, thus rewriting the source cell's truth table while it is being read. Additionally, the controller sends the source cell's truth table bits into the target cell's D input, thus configuring the target cell's truth table as an exact copy of the source cell. After a sufficient number of clock ticks of the system clock (128 for four-sided cells, i.e., enough ticks to sample and write each of the truth table bits for a 16×8 truth table), the target cell's truth table will match the source cell's, and thus the target cell will behave exactly the same as the source cell: the source cell has effectively been replicated by the controller.

9.3.2 Remote Cell Replication

Figure 9.10 shows a circuit that is similar to Fig. 9.9, except that the source and target cells are not adjacent to the controller. Instead, the source and target are now some distance away from the controller, and cells located in between them are used to transmit C and D information between the controller and the source and target cells. The controller works the same as in Fig. 9.9, except that it cannot directly

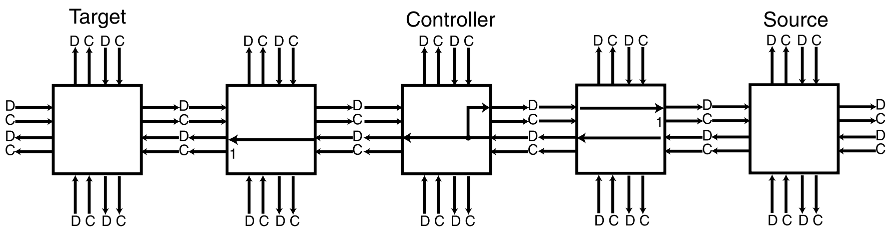


Fig. 9.10 Remote Cell Replicator. The Source and Target cells are in C-mode. The Source cell's truth table bits are read by the Controller, sent back to the Source cell, and also copied to the Target cell. Note that the Controller no longer directly controls the mode of the Source and Target cells

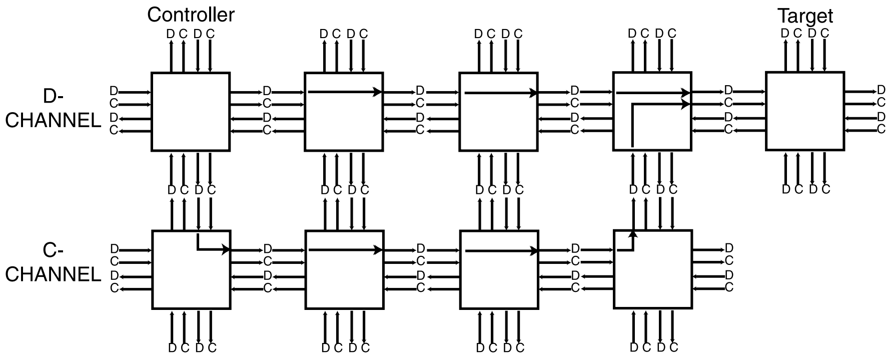


Fig. 9.11 Simple Multi-Channel Wire. The Controller sends new truth table bits into the Target via its own Eastern data output. Additionally, the Controller can now control the mode of the Target via its own Southern data output. This is a significant improvement over the circuit shown in Fig. 9.10

control the mode (C or D) of the source and target cells. This makes the circuit in Fig. 9.10 somewhat limited in its usefulness. A more useful approach involves the use of *multi-channel wires*.

9.3.3 Multi-Channel Wires

To control a cell, it is generally necessary and sufficient to control the C input, D input and D output on one of the cell's sides. The circuit shown in Fig. 9.11 is an example of a structure for controlling non-adjacent cells, by utilizing two lines of intervening cells. Such a structure is called a *multi-channel wire*. In this circuit, the controller sends information along two lines of cells (each called a *channel*).

The bottom channel (called the "C Channel") controls the C input on the source and target cells, while the top channel (called the "D Channel") access the D input and output on the source and target cells. Note that these lines are logical wires, or soft wires, rather than hard, physical wires: they are created by setting the truth tables of the cells to pass their input directly to their output. This primitive gives the

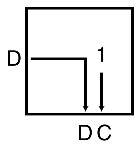


Fig. 9.12 A target cell which can be used to configure a cell *near* a wire's target cell

controller more or less complete control over the source and target cells: the ability to place them in C mode, read and write their truth tables, and then return them to D mode.

There are other types of multi-channel wires, but they all have the same basic characteristic: they allow a set of cells to interact with one or more non-adjacent cells. By using the right types of multi-channel wires, a set of controller cells can thus configure cells that are not adjacent and not directly connected to itself.

While it is evident from this example that wires allow access to non-adjacent cells, it would appear that they only allow access to the cell adjacent to the end of the wire. This is not the case, however. If the target cell is treated as *itself* being a controller, then it is possible to access cells that are near, but not adjacent to the end of the wire. For example, if the target cell (call it X) is configured as shown in Fig. 9.12, then data subsequently transmitted to cell X will, in fact, be used to configure the cell *below* cell X: the cell shown in Fig. 9.12 effectively moves the location of the wire's target cell.

Therefore, even though a wire can directly control only the cell adjacent to its end, it can *indirectly* control non-adjacent cells using intermediate cells such as that shown in Fig. 9.12.

Repeated application of this technique could, in theory, be used to gain control over a cell located *anywhere* within the matrix. However, this technique is limited in its usefulness, since accessing cells "n" locations away requires on the order of 2^n steps. Thus, wires' only practical use during configuration is to manipulate cells *near* their end. This would be a severe limitation of the Cell Matrix's nearest neighbor topology, if not for the concept of *wire building*.

9.3.4 Wire Building

Special kinds of wire building permit a cell to access any cells within a Cell Matrix. While wires can only be used to control cells adjacent to their ends, those wires themselves are built out of cells. In fact, it is possible to design a wire that allows cells at its end to be configured in order to make a new piece of the wire, i.e., to extend the wire. Beginning with a short wire, cells at its end can be configured to make the wire one cell longer. That longer wire can be used to configure the cells at the new end, thus making the wire another cell longer, and so on. This process can be repeated indefinitely (as long as there are cells available), thus allowing a set of controller cells to access cells arbitrarily far away. Moreover, it is possible to create wires that have turns, and again these turns can be created by the controller. This

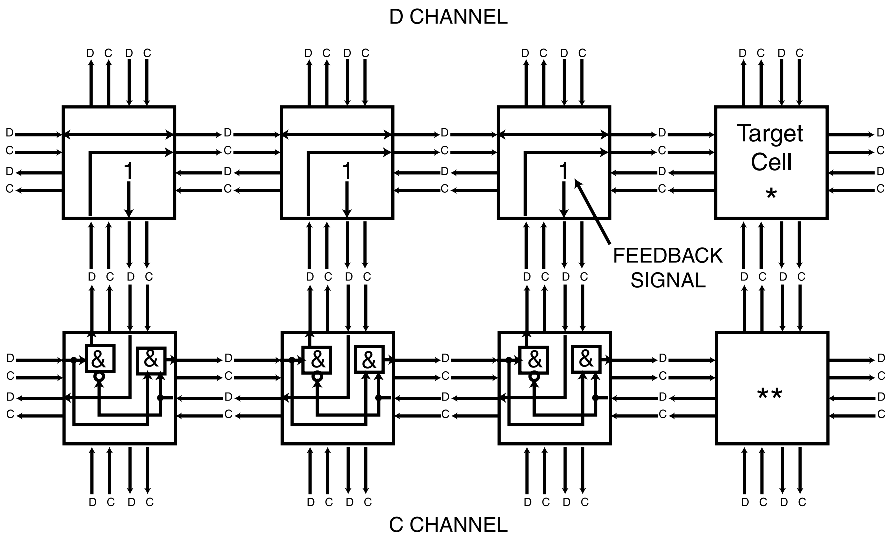


Fig. 9.13 Two Channel Extendible Wire. The D Channel transmits configuration information for the Target Cell. The C Channel controls the mode of the Target Cell. The Feedback Signal is used for autonomous determination of the location of the wire's end. If Cells (*) and (**) are configured as new channel pieces, then the wire will automatically be extended

means a set of cells can, in fact, access any cells within the matrix, through the use of proper wire-building techniques. Note that this permits remote control of cells, even though the underlying primitives used are all strictly neighbor to neighbor. Remote control permits external control of the system, but it also permits the smallest unit of a complex system to be multi-celled to an arbitrary size, which is convenient for most applications, including most work in self-organizing systems. In Sect. 9.4 we describe an application that uses a Supercell as its unit, which contains 270 × 270 cells.

Figures 9.13 shows a sample two-channel wire that is *extendible*. As in the wire of Fig. 9.11, the upper channel transmits a D signal, and the lower channel transmits a C signal. The cell labeled "*" is again called the target cell, and as in Fig. 9.12, both Cell (*) and Cell (**) can be easily configured. However, unlike the two-channel wire shown in Fig. 9.11, all of the D channel cells are identical, and all of the C channel cells are identical. This is accomplished through the use of a feedback signal: each cell within the D channel asserts a 1 to its south, which the corresponding C channel cell transmits to the previous cell of the C channel. Therefore, the end of the wire is identified not by a differently-configured cell, but rather by the lack of this feedback signal. Thus, by simply creating a new pair of D Channel and C Channel cells at the end of the wire, the wire is effectively extended, i.e., the location of the target cell is shifted one cell to the right.

This is the essence of wire building. While different sequences are needed for different types of extensions (such as turns) and for different types of wires (such as 3 channel wires), the basic mechanism is the same as in Fig. 9.13. These mechanisms are described in more detail elsewhere (Macias 2001).

Multi-channel wires and associated wire-building techniques can be used to access cells anywhere within the matrix. This raises the question, "What does one do with such access?" There are many answers to this, and in the remainder of this section, a few general examples will be presented. Section 9.4 will discuss more-specific examples.

9.3.5 Cell Testing

Given access to the C input and D inputs and outputs of a target cell, it is possible to perform a variety of tests on the target cell, to ascertain its health, i.e., to determine if it is operating as expected. For example, the cell's truth table can be loaded with 0's, and then the D output examined while the D input is toggled between 0 and 1. This would detect shorts between input and output, as well as detecting stuck-at-one faults inside the truth table memory, or along the D input or D output paths. A second example is that a set of certain bit patterns can be loaded into the cell, and then read back out and compared to the loaded pattern: different alternating bit patterns can be used to detect shorts within the truth table memory, based on the physical layout of the memory within the cell. This fault testing work was developed and successfully conducted on defective hardware for the Cell Matrix architecture using the above-described multi-channel wire building to reach each cell (Durbeck and Macias 2002).

9.3.6 Circuit Building

The question of how to *bootstrap* a Cell Matrix remains, that is, with no direct access to the vast majority of cells within the Cell Matrix, how can a Matrix be populated with the desired set of truth tables, particularly given that the Matrix is always running, and thus, truth tables are in use from the moment they are in place. However, all the necessary building blocks have already been presented. Figures 9.14a–9.14f show a sample bootstrap sequence. Note that this is not the only possible way to bootstrap a region of the Cell Matrix. It is a pedagogically interesting example because it is one of the simplest and most straightforward, but it is not used in typical practice, because it is one of the slowest ways to configure a region of cells.

In Fig. 9.14a, a two-channel wire is built from West to East, extending just one column of cells shy of the Easternmost corner of the region of interest. The target cell in the corner of the region is then configured.

In Fig. 9.14b, the wire has made a corner, and is extended one step to the South. This extended wire is then used to configure the next target cell (*).

The wire is then extended South another step, and a third target cell is configured to the East, as shown in Fig. 9.14c. This process continues, until, as in Fig. 9.14d,

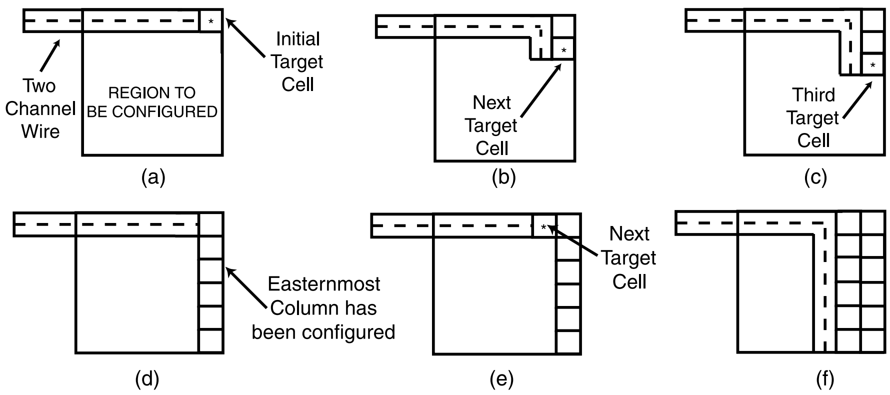


Fig. 9.14 Configuration of a Region. In (**a**), a two-channel wire is built into the region of interest, and is used to configure cell (*). In (**b**), the wire has been extended with a corner, and the next target cell is configured. In (**c**), the wire is extended further to the South, and a third target cell is configured. In (**d**), the Easternmost column has been completed. (**e**), shows the beginning of the second column's configuration: the wire has been broken and re-built, but ends one cell shy of the previous extension. In (**f**), the second column has been completely configured. This process is repeated until the entire region has been configured

an entire column of target cells has been configured, along the Easternmost edge of the region of interest.

In Fig. 9.14e, the wire has been *broken*, i.e., the end of the wire is returned to the original entry location into the region of interest. The wire is again extended to the East, but stops one cell earlier.

The wire then turns a corner, and the above steps (configure/extend) are repeated, configuring a second column of cells, as shown in Fig. 9.14f.

The above steps are repeated, until the entire region of interest has been configured. Note that this technique cannot be used exactly as described for configuring the Westernmost columns, since the wires themselves have a width to them. The easiest way to address this is to avoid this edge case by imagining that the region of interest as being one wire width wider than it really is, and leaving the Westernmost columns (which are not actually of interest) unconfigured.

There are numerous enhancements to this basic scheme, including techniques to avoid completely rebuilding the West-to-East wire after each column pass. Parallel configuration is also feasible, and will be discussed briefly in Sect. 9.4. Also, note that the configuration of cells that assert their C outputs requires special consideration, since such outputs could interfere with the configuration of the wires that are being used to configure the region's cells.

9.3.7 Circuit Reading

Using circuits and sequences similar to the bootstrap method described above, it is possible to non-destructively read a set of cell configurations from a region of

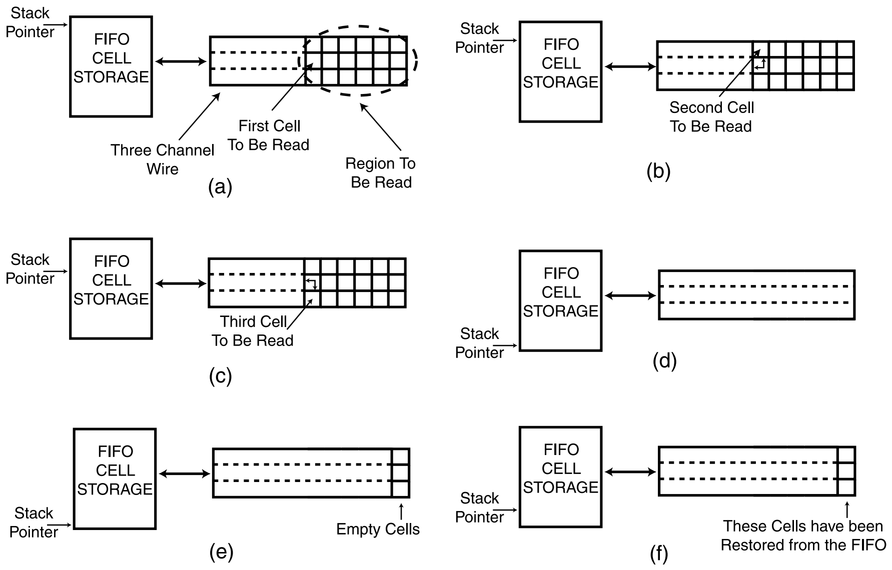


Fig. 9.15 Non-destructive read of a region of cells. In (**a**), a three-channel wire has been built to the edge of a region to be read. A first cell is read, and its configuration is stored in the FIFO. In (**b**), that first cell is used to read a second cell, which is also stored in the FIFO. In (**c**), a third cell is read and stored, after which the wire will be extended one step. In (**d**), the entire region has been read, stored and overwritten with the wire itself. In (**e**), the wire is reversed (backed up one step), and in (**g**) the Easternmost cells have been restored form the FIFO

the matrix. The technique is similar to bootstrapping, but utilizes a *reversible* wire, i.e., one that can not only be extended a single step, but can also be *shortened* a single step. The basic technique is shown in Figs. 9.15a–9.15f. For simplicity, this illustrates the reading of a single (one-dimensional) line of cells only. Note that implementation of a reversible wire requires three channels.

In Fig. 9.15a, the wire has been built to the East, to the beginning of a set of cells whose contents are to be read. The cell directly ahead of the wire (i.e., to the East of the D channel) is read, and its truth table configuration is stored in a temporary repository (a FIFO). That cell is then configured to allow reading of the cells to the North and South of it, with those cells' configurations also being stored in the temporary repository. This is shown in Figs. 9.15b and 9.15c, respectively.

The three-channel wire is then extended, and the process repeated. In Figs. 9.15d, the wire has extended all the way to the East. Again, edge cases need to be considered, but can be neglected by imagining the region of interest to be larger than it actually is. At this point, all of the cells occupied by the wire have been reconfigured from their initial configuration (in order to implement the extended wire), but their initial configurations have been read (and presumably processed by some circuitry outside that shown in these figures). Also, those configurations have been stored in a temporary storage location (which can be as simple as a set of cells arranged in a two-way shift register, i.e., a FIFO).

In Fig. 9.15e, the wire is reversed a single step (using the third channel), and in Fig. 9.15f, the previously-stored configurations are restored to the cells near the end of the wire. These two steps are repeated, until the entire row has been restored.

For a two-dimensional region, another pass would be made to the south of the original West-East wire, thus reading the next three rows of cells. At the conclusion of this, the cells within some region of interest will all have their initial configurations, but a copy of those configurations will have been sent by this circuit to some other circuitry that will perform analysis, make a new copy, vote on truth table contents among multiple copies, or conduct some other function.

Note, however, that while the above technique will read the configuration of cells without (permanently) changing them, it does not read the state of cells, i.e., the values of their inputs and outputs, and similarly does not preserve their state. State reading and preservation would require additional circuitry built into the circuit itself, since changing the configuration of a single cell can, in general, alter the state of the entire circuit.

These are a few detailed examples of techniques related to self-configuring circuitry. They form a base of primitives or building blocks that are composed to create more complex functions and circuits. The next section will describe larger-scale applications of these techniques to the implementation of circuits that exhibit distributed management and control.

9.4 Distributed Management and Control in the Cell Matrix

There are a number of examples of how the Cell Matrix can be used to manage various tasks related to its own operation and maintenance. While non-Cell Matrix systems could be designed specifically to implement any of these examples, the advantage of the Cell Matrix architecture is that *it supports all of them*: the Cell Matrix architecture does not have to be modified in any way in order to implement these systems.

9.4.1 Hardware Error Checking

The wire building and cell testing techniques described above can be used to test individual cells within the matrix, to ascertain their proper functioning. Moreover, since all that is required to perform these tests are basic logic circuits and simple state machines, these tests can be performed by circuitry within the matrix itself. This offers a number of interesting opportunities.

For example, once a small initial set of cells is known (say, via conventional validation techniques) to be functioning properly, and a state machine is built to perform subsequent cell tests, cells can be tested, verified, and then used to build longer wires, allowing testing of more-remote cells. Other than the initialization issue, this eliminates the question of "what if the test circuit itself is defective?" since only known-good cells will be used in extending the test circuitry (Macias and Durbeck 2002, 2004; Durbeck and Macias 2002).

By running multiple test circuits, cross-checking can be performed among multiple testers. Basic N-way redundancy could be used to verify the initial circuitry, after which a single copy would suffice (as far as manufacturing defects are concerned: transient errors are a different consideration).

This approach also allows parallel testing to be performed. Again, starting from a single test circuit, multiple testers can be configured from known-good cells, and these testers can operate in parallel to test multiple regions simultaneously, and then construct more parallel testers. This can reduce test time to $O(n^{1/2})$ for n cells in a two-dimensional matrix, and $O(n^{1/3})$ in a three-dimensional one (Durbeck and Macias 2001d; Macias and Durbeck 2002, 2004).

Test results can be stored in something similar to a "bad block" list (Duncan 1989), and this list used in subsequent configuration operations. If a place-and-route algorithm were implemented directly on the Cell Matrix hardware, it would simply note these defective cells as being unavailable for placement or routing, and would thereby avoid them in creating compiled circuits.

For handling run-time defects such as single event burnout (Waskiewicz et al. 1986) or single event gate rupture (Fischer 1987), these tests could be performed periodically. Multiple copies of circuitry can be maintained, with copies taken offline individually, their underlying cells re-tested, and their configuration adjusted as needed to avoid newly-defective cells.

9.4.2 Autonomous Fault Handling Through Autonomous Circuit Building

We have devised a methodology for implementing a desired target circuit on top of the Cell Matrix in a way that allows that system to configure itself in order to avoid defective regions of the matrix (Macias and Durbeck 2002, 2004; Durbeck and Macias 2001a, 2001b). If new defective regions are later found or suspected to be present (for example, because some sort of built-in self test has failed), the system can be given a single "REBUILD" command, and it will locate, isolate and avoid all defective regions, while re-implementing itself using only good cells. The goal was to have these operations performed by the system itself, with a minimal amount of external intervention required. This work combines the above techniques of hardware error checking with some bio-inspired concepts in self-organization (Mange et al. 2000).

This approach utilizes the concept of a *Supercell*. This is a general term for a collection of contiguous Cell Matrix cells configured to perform a variety of functions while still retaining the underlying self-configurability of individual cells. In our self-repairing circuit building work (Macias and Durbeck 2002, 2004) the Supercell first performs a number of *initialization* functions, including:

- testing a region of the matrix for defective Cell Matrix cells;
- configuration of new Supercells on known-good regions;
- activation of isolation circuitry within good Supercells, in order to prevent any interference from bad Supercells; and
- sharing of configuration information among a network of Supercells in order to configure new Supercells in multiple regions in parallel.

The purpose of the initialization stage is to tile a region of the Cell Matrix with known-good Supercells, while isolating defective cells. Note that this part of the system's operation requires a set of configuration strings to be sent into the empty matrix. These configuration strings depend on the high-level circuit to be implemented, but are completely independent of the location of any defects in the Matrix (since the location of such defects is assumed to be unknown). All subsequent steps are performed by the collection of Supercells themselves, without any further external intervention.

Following initialization, the system enters a *differentiation* phase. In this phase, Supercells assign themselves unique integer IDs, so that they can be differentiated from each other. Without such an assignment, all Supercells are identical to each other. This assignment is accomplished through the collective operation of the entire set of Supercells. Differentiation changes the contents of two ID registers contained within each Supercell:

- one ID contains a position-dependent integer, which is simple to assign (by incrementing an incoming neighbor's ID and passing that to other neighbors), but the set of assigned integers is not necessarily contiguous; and
- a second ID that is position-independent, and whose collection is guaranteed to form a set of contiguous integers.

When the initial configuration strings are developed, an abstract representation (called the "genome") of the final target circuit is coded inside the strings. The genome is simply a netlist, specifying the components of the final circuit, along with their input-to-output interconnections. What is not specified is the particular locations of those components in the matrix, nor the paths that will be used for their interconnections (since doing so would require knowledge of the locations of defective Cell Matrix cells).

After unique IDs have been assigned, each Supercell compares its ID with the ID of components stored in the final circuit's genome (this is why contiguous IDs are required). Using the information inside the genome, each Supercell thus knows which component it is to implement in the final circuit. Each Supercell then configures inside itself its piece of the final circuit.

Following differentiation, the collection of Supercells must be wired together in order to implement the final target circuit. This requires first determining pathways from component to component, and then creating communication channels along those pathways. Both of these steps are performed by the Supercells themselves, without any external intervention. Path-finding is done using a greedy algorithm, which picks the shortest path from component to component. Channel creation is performed by utilizing pre-existing pieces of channels inside each Supercell, and by configuring cells near the junction of these channel pieces, in order to form continuous pathways. Note that some of these channel pieces cross within the Supercell, to allow the creation of crossed communication pathways.

The final Supercell design consisted of 270 × 270 Cell Matrix cells. This was intended only as a proof-of-concept, and represents neither the smallest nor most-efficient Supercell for the given problem. There are ways to perform more traditional fault tolerance for the Cell Matrix architecture that have been developed and analyzed (Saha et al. 2004; Macias and Durbeck 2005a), but the point of the Supercell work was to demonstrate autonomous, self-organizing circuitry, and faulty hardware was simply the impetus to which the system responded in order to modify its behavior.

9.4.3 Self-Replication

Figures 9.16a–9.16e show the operation of an entirely self-replicating circuit on the Cell Matrix. The circuit is comprised of two main parts. The upper-left region of the circuit is called the "Main Grid," and is a state machine that generates bit sequences and sends them into three wires. The right half of the circuit is a copy of the left half, but with additional space (two rows of empty cells) between each row of non-empty cells. The right-hand circuit is called the "Exploded Grid," since it is a copy of the Main Grid on the left, but with the rows spaced out vertically.

The reason for using an Exploded Grid is that we do not want the circuit to actually be operating: it is intended to supply a *data* version of the circuit being replicated. Since the circuit itself is intended to modify other cells, we want to carefully control the behavior of this copy. So the cells in the Exploded Grid that can assert their C outputs are kept permanently in C mode, so that their C outputs are constantly forced to 0. This is achieved by pairing such cells with other cells, called "Guard Cells."

The bitstreams generated by the Main Grid extend the three wires as follows:

- one wire is extended into the Exploded Grid, in order to read the cells within the first row of that grid;
- another wire is extended into an empty region below the Main Grid, and will create a copy of the Exploded Grid there, but without any inter-row spaces; and
- the third wire is extended into an empty region below the Exploded Grid, and will create an exact copy of the Exploded Grid.

Figure 9.16a shows the initial circuit. Figure 9.16b shows the circuit with the three wires immediately prior to reading the first cell from the Exploded Grid. In Fig. 9.16c, the first row of the Exploded Grid has been completely read, and two copies of it have been made in the region below the original circuit. A single row has now been configured in the new Main Grid, and three rows have been configured in the new Exploded Grid (one row of Main Grid circuitry, and two rows of Guard Cell circuitry). This process is somewhat analogous to the translation and transcription steps found in the replication of DNA (Arms and Camp 1987).

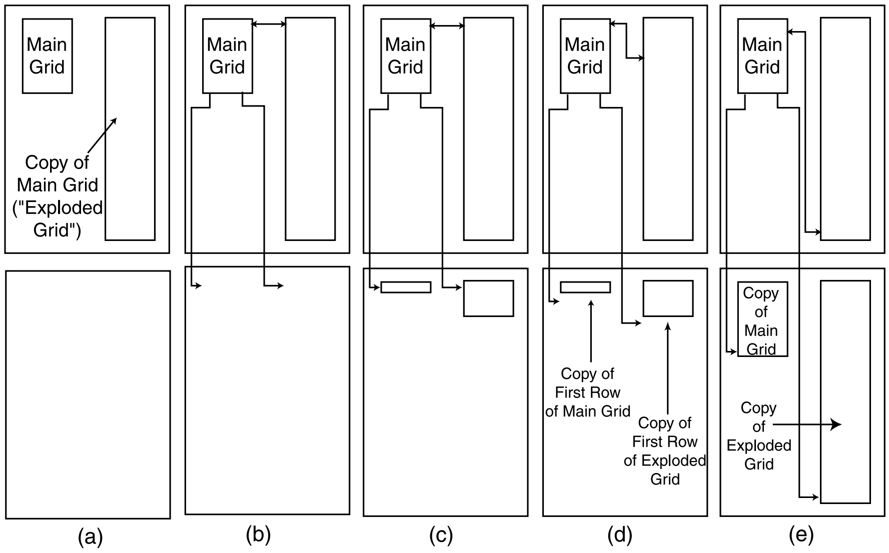

Main Grid
Copy of Main Grid ("Exploded Grid")
Copy of First Row of Main Grid
Copy of First Row of Exploded Grid
Copy of Main Grid
Copy of Exploded Grid
(a)
(b)
(c)
(d)
(e)

Fig. 9.16 Self-Replicating Circuit. (**a**) shows the initial configuration. The Main Grid is a circuit that will read the Exploded Grid, and produce a copy of it and itself. In (**b**), three wires have been built, extending into the Exploded Grid and the initially-empty regions to the South. In (**c**), the first row of the Exploded Grid has been read and used to reproduce the first row of the Main Grid and the Exploded Grid in the region to the South. In (**d**), the three wires are moved in preparation for reading the next row of the circuit. In (**e**), all rows have been read and copied, creating an exact copy of the original circuit in the region to the South

Figure 9.16d shows the circuit once the wires have been adjusted for reading the second row of the Exploded Grid. Of course, the wires extending into the original and new Exploded Grids require more extension steps than the wire in the new Main Grid.

The above steps are repeated for each row of the Exploded Grid. Figure 9.16e shows the final state of the system, where an exact copy of the original circuit has been made to the South. Typically, the lower-rightmost cell would be configured to output a "GO" signal into the circuit, which would trigger the execution of the above circuitry. Thus, as soon as a new copy of the circuit is created, it immediately begins making a new copy of itself.

This is, in itself, not necessarily useful, but is a useful building block to which a number of various enhancements can be added. For example, after being placed on a Cell Matrix, the circuit could make a copy of itself to the South. That copy could make a copy of *itself* to the South, and that copy could make a copy of itself, and so on, thus creating a single column of copies of this circuit.

The height of the column could be hardwired into the circuit, for example, by incrementing a counter within each circuit, and only replicating a fixed number of times. Alternatively, the height could be determined dynamically by the presence of a marker in the matrix indicating the desired extent, or the circuit can itself determine when it has reached the Southern edge of the matrix (by noting that cell configuration operations no longer work). Once a leftmost column has been created, the bottom circuit can signal to the other circuits in that column to begin replicating to the East, resulting in the parallel building of a new column. Note that each circuit in the column replicates in parallel with every other circuit in that column. Thus, whereas creating the initial column (containing, say, n copies of the circuit) required n replication cycles, creating the second column requires only one replication cycle.

Generation of each subsequent column will also require the same time as a single replication cycle. To create an $n \times m$ array of these circuits would thus require n replication cycles to configure the first column, plus $m - 1$ replication cycles to create each of the remaining $m - 1$ columns, for a total of $n + m - 1$ replication cycles. This is extremely better-than-linear performance. In fact, configuring n copies of this circuit requires on order of only $n^{1/2}$ steps. For a three-dimensional Cell Matrix, configuration time for n copies is a mere $n^{1/3}$ steps. This is an extremely efficient way to configure large regions of a Matrix.

Of course, we are usually interested in something more than simply filling the matrix with copies of a single circuit. The self-replicating circuit is intended to be a *carrier* of additional circuitry.

9.4.4 Fully Autonomous Self-Configuration

The above self-replicating circuit can be used to make copies of the Supercells described previously, which will then autonomously implement a desired target circuit. Collectively, this represents a fully autonomous, fault-handling, self-configuring system. This circuit can be thought of as a seed, or perhaps a biological cell. Upon placing a single copy of it inside an empty matrix, it begins to replicate, filling a region of the matrix with copies of itself. Once a sufficient number of copies have been created, they begin to differentiate and specialize, and then work together to implement some higher-order function. Moreover, this is done in a dynamic manner, with the exact configuration of the final circuit dependent on the environment (specifically, the location of defective cells within the matrix).

9.4.5 Hardware Compilation

Using the techniques described above, it is possible to perform compilation of algorithms not into software, but directly into hardware. This is already an active research area of reconfigurable logic (Page 1996). However, with the Cell Matrix as the underlying hardware substrate, it is possible to design systems that are *self-compiling*, i.e., the circuitry that produces the final compiled circuit can itself be running on the Cell Matrix.

One area in which such a setup would be useful is in implementing a Just-In-Time (JIT) (Deutsch and Schiffman 1984) compilation system. There is a huge parameter space within which algorithms could be developed. For example, the wordsize of an arithmetic unit can be adjusted based on the characteristics of data being processed. This might change over time, and circuit characteristics adjusted accordingly. Similarly, the number of registers available in a general-purpose processor (implemented on the Cell Matrix), or the character size of a hardware string processor could be adjusted over time. Sequences of operations that occur repeatedly could be analyzed, and new hardware synthesized to implement their collective function directly in hardware. Such hardware could be dismantled, and the underlying cells re-used, if the circuitry is not utilized for some period of time.

9.4.6 Hardware Operating Systems

Having self-configurable hardware, it is possible to consider hardware analogs of various software concepts, especially concepts related to operating systems. This leads to the concept of a *hardware operating system*.

For example, combining wire building techniques and bootstrap mechanisms, one can easily imagine the notion of a *hardware library*, wherein circuits consisting of Cell Matrix cells are stored, available for retrieval and replication elsewhere in the matrix, just as software libraries store code that is re-used in other programs.

As another example, the notion of virtual memory could be extended to *virtual hardware*, where a matrix appears to have more hardware than actually exists. By storing cell configuration information (and utilizing appropriate compression mechanisms), and using it to configure cells as needed, it is possible to design a system on the Cell Matrix that emulates a matrix that is larger than the physical matrix on which it resides. Such a system would, effectively, intercept accesses to non-existent cells, create them on-the-fly, and redirect requests to those newly-created cells. These cells would be located virtually in a fixed location, but *physically* might be located somewhere different.

Closely related to this notion of virtual hardware is *hardware swapping* and *hardware timesharing*, where a single Cell Matrix is shared by multiple applications, which are loaded and unloaded from the matrix's cells, so that a single set of cells are used for more than one application. Such a system would probably employ a double-buffering mechanism, so that while one circuit is executing on one region of the matrix, another region would be configured with the second circuit to be executed. Once that circuit is ready, and the desired time slice has expired, that second circuit would begin executing, while the cells of the first circuit would be re-configured to implement the third circuit, which would eventually be allowed to run while the fourth circuit was implemented, and so on. Once the last circuit was given a time slice, the first circuit would again be configured and allowed to run. Other than the need to double-buffer (since configuring a circuit takes a non-negligible amount of time), this is highly analogous to swapping and timesharing of software. Again, as described above, consideration must be given to reading, preserving and restoring not only the configuration of each cell within a running circuit, but also the state of each cell, meaning its outputs and inputs.

9.5 Extension to the Analog Domain: The Songline Processor

"*…the labyrinth of invisible pathways which meander all over Australia and are known to Europeans as 'Dreaming-tracks' or 'Songlines'; to the Aboriginals as the 'Footprints of the Ancestors' or the 'Way of the Law.' Aboriginal Creation myths tell of the legendary totemic being who wandered over the continent in the Dreamtime, singing out the name of everything that crossed their path—birds, animals, plants, rocks, waterholes—and so singing the world into existence*" (Chatwin 1986).

The previous descriptions of a self-configurable processor are all rooted in the digital domain: one where inputs and outputs are binary in nature, i.e., where each input has one of only two possible values. This can be extended by allowing inputs and outputs to have values within a continuous range. For example, instead of standard TTL-level signals, inputs and outputs can be allowed to have values anywhere between 0 and 1 volts. Such an extension not only changes the domain and range of the function, but also the basic characteristics of the mapping function. In particular, a truth table with discrete rows will no longer suffice for describing a cell's input-to-output mapping.

As will be seen in what follows, extension of the Cell Matrix architecture in this way leads to a very different type of processor whose inputs, outputs and programs are, in some sense, akin to music: time-varying signals that are copied from one element to another in C-mode by playing and recording them. In D-mode, the information stored within a song is extracted by sampling the song at a particular point in time. This is not entirely dissimilar to aspects of *Songlines*, which are passed from person to person through singing, hearing and memorizing, and whose information is extracted by singing/listening to the song at a particular point in time (corresponding to where one is geographically in their traversal of the Songline). For this reason (and with great respect), this extended version of the Cell Matrix is called a *Songline Processor*. It's internal mapping function can be called a "song," and a particular value derived from the mapping may be called a "note." For clarity in what follows though, we'll stick to the mathematical terminology of "function" and "value."

Consider a cell with a single input and a single output. For the binary version of a cell, a truth table consisting of a single entry—say an input x—will suffice. The output values corresponding to $x = 0$ and $x = 1$ must be specified, and this completely defines the characteristics of the cell. Such a truth table can be stored by simply recording two bits: basically $f(0)$ and $f(1)$, where $f(x)$ is the cell's mapping function that turns a single input bit into a single output bit.

But when the inputs and outputs are real-valued, the mapping function $f(x)$ is now a real-valued function of one real variable. To store a complete specification of this function in a truth table would require an infinite number of entries (one for each possible real-valued input value), with each entry specifying a single real-valued output. Table 9.1 shows an approximation of such a table for a sample function ($f = \sqrt{(x)}, x \in [0, 1]$).

Of course, this is only an approximation of an exact truth table, which would require an infinite number of rows (with an infinitesimal change in x from row to row).

Table 9.1 Approximation of function $f(x) = \sqrt{(x)}$

x	$f(x)$
0.000	0.000
0.001	0.032
0.002	0.045
0.003	0.055
...	...
0.998	0.999
0.999	0.999
1.000	1.000

This is complicated further in the case of a two-input cell, whose mapping function $f(x, y)$ is a function of two real-valued inputs. Table 9.2, for example, shows an approximation to the function $f(x, y) = x \times y$.

Clearly one can estimate a mapping of any number of input variables using such discretization of the input space. While this is not an ideal for physical implementation, it can be a useful model to keep in mind.

For working purposes, we consider our basic programmable cells to be four-sided, with (again) a C and D input and output on each side. These cells are arranged in a 2-D layout with each cell having four neighbors. We continue to designate the sides as N, S, W and E, but we now define a cell's "truth table" with a series of mathematical expressions describing each D and C output as a function of its four D inputs. Details related to storing such functions will be discussed below in the Implementation section. As will be seen, implementation difficulty increases significantly (for a variety of reasons) as the number of sides increases.

The tradeoff for this difficulty of implementation is a richly-powerful computing paradigm, where, for example, we can compute square roots with a single pre-programmed cell; or the product of two numbers with a differently-programmed cell. Moreover, the fact that these cells are operating on inherently-analog signals (as opposed to digitizing analog inputs and processing them discretely) has interesting implications application-wise. For example, Fig. 9.17 shows a simple amplifier circuit, comprised of a single cell.

Here, the input DN controls the degree of amplification; DW is the input signal; and DEout is the amplified output. By adjusting the variable resistor to set DN anywhere from 0 to 10 volts, the input is amplified accordingly (up to a maximum output of 1 V).

Figure 9.18 shows an implementation of a continuous-valued flip-flop. The incoming data signal is sent to the D input, and the GATE control is sent to the G input. The latched value can be read from output Q. The pair of cells operate together to create a feedback loop that traps a particular value between them. When G is high (G $>$ 0.5 V in this case), the incoming signal is sent to the feedback cell, which re-sends it to the initial cell. As the input signal changes, its value is continuously updated in the feedback loop. But when the gate closes (G $<$ 0.5 V), the

Table 9.2 Approximation of function $f(x, y) = x \times y$

x	y	$f(x, y)$
0.000	0.000	0.000
0.000	0.001	0.000
0.000	0.002	0.000
...	...	...
0.000	0.999	0.000
0.000	1.000	0.000
0.001	0.000	0.000
0.001	0.001	0.000
0.001	0.002	0.000
...	...	...
0.001	0.999	0.001
0.001	1.000	0.001
...	...	...
0.999	0.000	0.000
0.999	0.001	0.001
0.999	0.002	0.002
...	...	...
0.999	0.999	0.998
0.999	1.000	0.999
1.000	0.000	0.000
1.000	0.001	0.001
1.000	0.002	0.002
...	...	...
1.000	0.999	0.999
1.000	1.000	1.000

incoming signal D is ignored, and the value received from the feedback cell is recirculated back to that cell, creating a closed loop that traps a single value. This is effectively a sample-and-hold circuit.

Figure 9.19 shows a basic differentiation circuit. The incoming signal is sent into DW; this signal is passed to the cell on the right, which returns it to the cell on the left. The cell on the left computes DW − DE, and sends the difference to DSout. Thus, for an incoming signal $f(t)$, DSout $= f(t + \delta t) - f(t)$ where δt is the time it takes the incoming signal to enter and leave the cell on the right. Of course, a scalar multiplier can also be supplied to the cell on the left to adjust the output range based on how quickly the input signal is changing. Similar designs can be created for integrating (summing) an incoming signal; such designs also require only two cells.

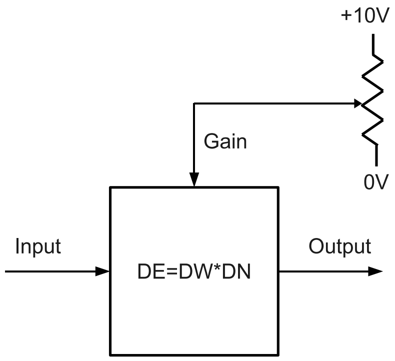


Fig. 9.17 Single-Cell Amplifier. The variable resistor sets the desired gain, which multiplies the input signal to produce an amplified output

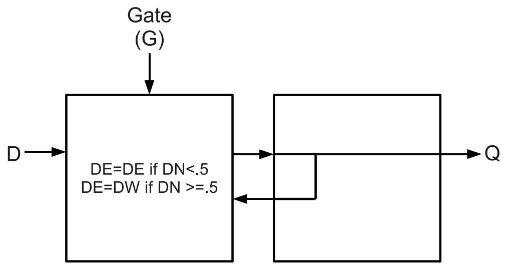


Fig. 9.18 Continuous-Valued Flip-Flop. Raising the Gate input above 0.5 V allows input D to be loaded into the device. Dropping the gate below 0.5 V traps the loaded value between the two cells. Q presents the output value

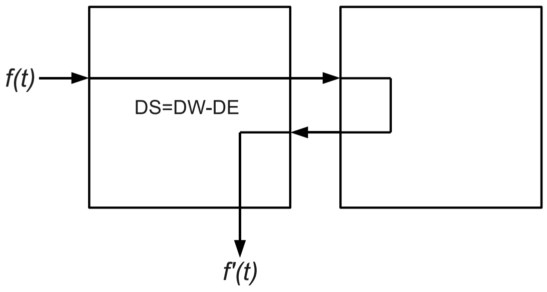


Fig. 9.19 Differentiation Circuit. $f(t)$ can be any time-varying input signal; its derivative with respect to time is produced from the bottom of the leftmost cell. The derivative is scaled by an amount related to the propagation delay of the cells

Figure 9.20 shows a ramp generator. The rightmost cell is again just a feedback path. The cell on the left receives the fed-back signal, adds an increment to it (which is supplied by the DW input), and sends the sum to the right. The value of the Increment input (in conjunction with the input-to-output time of the cells) determines how quickly the output rises. For example, to achieve an output frequency of 1 Hz, supposing the total propagation delay through the two cells is τ, we would need an increment value of 0.9τ.

The equations in the leftmost cell cause it to rollover back to 0 anytime its outputs exceeds 0.9 V. In this setup, the output Q is thus a repeating sawtooth, rising from 0.0 to 0.9 V and then returning to 0.0 V. Of course, the cell on the right could also amplify the output signal to increase the maximum output to 1.0 V. The frequency of this waveform depends on the value of the Increment input: the larger the increment,

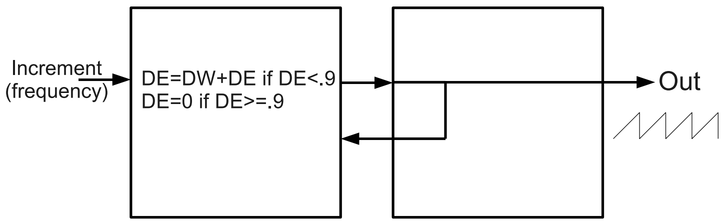


Fig. 9.20 Ramp Generator Circuit. The output rises to 90 % of a cell's maximum output value and is then reset to 0. The rate of increase depends on the Increment input, which thus adjusts the frequency of the output

the more quickly the output rises, and thus the higher the frequency. Of course, if the input-to-output time is large compared to the period of the generated waveform, the output may be badly discretized. But assuming a small propagation delay, the output will rise relatively smoothly. Figure 9.20 is thus a voltage-controlled oscillator.

9.5.1 C-Mode

The above descriptions all assume previously-programmed cells, connected to build a static (non-changing) circuit. This is analogous to using digital/binary Cell Matrix cells to implement digital circuits that do not change. While certainly useful, this is only part of the Cell Matrix story. So is the case for a Songline Processor, which incorporates its own notion of a C/D-mode state.

Recall that for a binary Cell Matrix, the mode of a cell is also a binary variable associated with a cell's C input: If $C = 1$ then the cell is in C-mode; and if $C = 0$ the cell is in D-mode. These modes are fundamentally different from each other, and the cell completely changes from one mode to the other based on transitions of its C inputs.

In a Songline system, the *mode* of a cell is a real-valued variable (say between 0 and 1), and the corresponding behavior of the cell is a mixture of its pure D and pure C mode behaviors. The pure behaviors are as follows (the use of the terms "bit" and "truth table" will be clarified below):

- In pure C-mode, incoming D bits are used to populate the cell's internal truth table, while overwritten truth table bits are output through the cell's D outputs; and
- In pure D-mode, the internal truth table of a cell is unchanged, and is used to map incoming D values to outgoing D and C values.

When C is allowed to take on values between 0 and 1, these behaviors are more complex:

- incoming D bits will be combined with the cell's pre-existing truth table bits, with the mixing ratio determined by the value of the C input;

- the combined value will replace the cell's truth table (according to some sort of pre-defined timing pattern, to be discussed below);
- the D outputs become a mix of (a) the D values specified by using the truth table as a lookup table (addressed by the incoming bits) and (b) the truth table entries themselves; and
- the C outputs become a mix of (a) the C values specified by using the truth table as a lookup table (addressed by the incoming bits) and (b) a 0.

Thus, if $C = 1$ or $C = 0$ the behavior is the same as with binary cells; but for $0 < C < 1$ the behavior is a mix of the two pure extremes.

The notion of a "truth table bit" requires clarification. Recall:

- A cell's truth table is in fact a continuous-valued *function with a continuous domain; and*
- there is no natural assignment of "bit numbers" to the function stored within a cell's memory.

In a binary cell, C-mode operation occurs bit-by-bit, with successive bits being read and written as time progresses. The goal however is simply to transfer the essence of a truth table via a narrow conduit (the D channel). In a Songline system, we serialize a cell's internal mapping function, and then transfer function values using time as a parameter for the serialized function. For a function of one variable ($y = f(x)$) with domain [0, 1], we can do so by parameterizing the function using time as an index. For example, beginning at time $t = 0$ we can transmit the value of the function at 0, i.e., $f(0)$. Over the next second, we transmit the value of $f(t)$ as t increases from 0 to 1. After one second, the entire function will have been transmitted.

For a function of more than one variable, a single parameter (time) will not suffice for sweeping the entire domain. Instead we can define a scan pattern for sweeping the entire multi-dimensional domain with a single path, as shown in Fig. 9.21. Unfortunately, this requires discretizing at least one dimension, at least for the simple solution presented here. It's is still being investigated whether something like a space-filling curve (Sagan 1994) might be used to map the higher-dimensional space to a one-dimensional parameterized space without such discretization.

9.5.2 C-Mode Applications and Benefits

Typically, the most common use of C-mode in a binary Cell Matrix is to read or write a cell's truth table, most often to copy one or more cells from one region to another. While this is also a use of C-mode in a Songline processor, there are other applications of C-mode as well. One is waveform generation: by loading a waveform into a cell's internal function memory and then placing the cell into C-mode, the pre-loaded waveform can be read out and fed to other parts of the system. This is an easy way, for example, to reproduce a fixed carrier wave for processing by an amplitude-modulation generator.

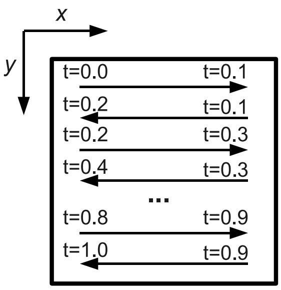
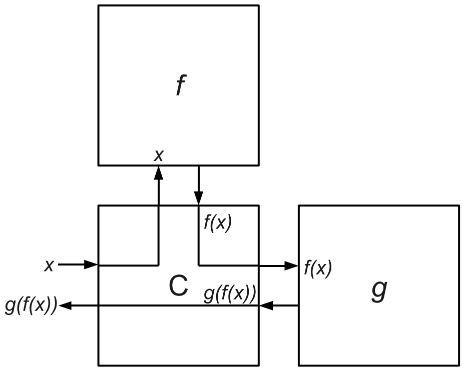


Fig. 9.21 Sample scan pattern for 2-D domain. This figure shows the domain of a function $f(x, y)$ of two variables. The values of the function can be sampled from $t = 0$ to $t = 1$ in the order shown. *Horizontal rows* faithfully reproduce $f(x, y)$ for all values of x, but from row to row there is a discrete jump in the value of y

Fig. 9.22 Data-based function composition. Cells f and g are used successively to evaluate first $f(x)$ and then $g(f(x))$



Another interesting aspect of C-mode relates to function composition, and is illustrated in Figs. 9.22 and 9.23. Each figure shows three cells: cells f and g are arbitrary "interesting" functions, while cell C is a cell whose job is to compose f with g. In Fig. 9.22, cell C receives an input value x, sends it to cell f, receives $f(x)$, sends that to cell g, and receives $g(f(x))$. This provides an evaluation of $g(f())$ at the particular point x.

In Fig. 9.23, cell C receives a "go" signal which places cell f into C-mode. Cell C reads cell f's function, sends it into cell g, reads back $g(f)$ (which is basically $g(f(t))$ at whatever time t has passed since cell f entered C-mode), and writes that new value back into cell f's function. After the entire function has been read and modified in this way, "go" is de-asserted, and cell f returns to D-mode. But now, the function stored in cell f is actually $g(f)$, i.e., the composite of functions f and g. $g(f(x))$ can subsequently be evaluated directly by sending x into cell f and reading back the value on the cell's D output.

Thus, on a Songline processor, there is a connection between C-mode and the notion of *operating on a function* (vs. operating on a function's value at a particular point). Thus we have a hardware implementation of a concept from *functional analysis* (Bachman and Narici 1966).

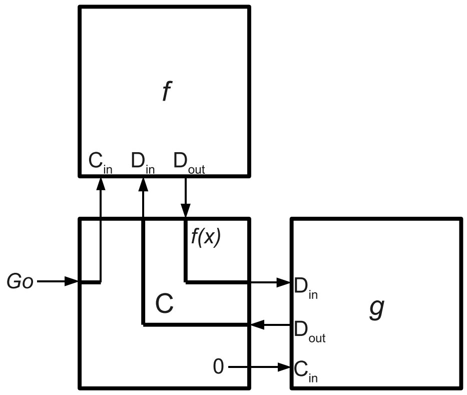


Fig. 9.23 C-Mode based function composition. Cell f is reconfigured—with help from Cell g—to implement the composite function $g(f(x))$. This is done by using Cell g to evaluate g on *the function f itself*, i.e., to evaluate $g(f)$ vs. $g(f(x))$

9.5.3 Advantages of a Songline Processor

While the notion of constructing circuits from elemental building blocks containing continuous-valued functions may seem archaic, there are a number of potential advantages to this paradigm. One advantage is speed: an arbitrary function may be evaluated, in effect, by a single memory lookup. Evaluating a transcendental function takes no longer than simple multiplication by a constant. Moreover, such function lookups may be more precise then what can be achieved in a digital domain, where functions are approximated by (for example) part of their Taylor Series polynomial, which is then evaluated at a point near to (but generally not exactly equal to) the desired point of evaluation. This of course depends on the effectiveness of the implementation and specifically the storage mechanism for a cell's mapping function.

On the other hand, being analog in nature, there is perhaps an inherent fuzziness in the behavior of these cells, whose mapping might potentially be affected by temperature, pressure, or other environmental conditions. There is speculation that a certain degree of non-deterministic behavior may be beneficial in systems that attempt to mimic intelligent behavior (Pearn 2000). Thus a Songline processor may be an interesting platform for work in machine learning and artificial intelligence.

Finally, having a basic cell that can perform an arbitrary input-to-output mapping leads to a potentially simpler way to design circuits, as suggested by Figs. 9.17, 9.18, and 9.20. Such a system is also potentially simpler to interface with in a universe whose processes appear (at least on a macro scale) to be inherently analog and continuous-valued.

9.5.4 Significance

A Songline processor maintains the essential benefits of a binary Cell Matrix, including self-configurability, inherent defect tolerance, a lack of specialized components, versatile routine, and so on. Additionally, potential significance of the Songline architecture itself includes the following:

- it represents a reconfigurable approach to implementing analog circuits, similar in spirit to Field Programmable Analog Arrays (Kluwer 1998) but in some sense more direct;
- it provides a natural mechanism for mimicking analog behaviors, by storing such analog patterns directly into the function memory of a cell for later readback; and
- by using C inputs between 0 and 1, sampled analog data can be tweaked to a desired degree, providing opportunities for adaption based upon a model signal and a desired degree of variation.

This last point has potential applications to evolvable hardware systems (Greenwood and Tyrrell 2006). In its simplest form, a desired function can be developed by applying training data and comparing its actual output a against the desired output d. The difference $|a - d|$ can be used to generate a C input to the evolving cell. For a large difference, the evolving cell's C input will be large, and its truth table will tend to mimic the training data. As the difference decreases towards 0, the C input also diminishes, causing the cell's mapping function to change less and less. Mild or short-lived errors will causes relatively small changes in the mapping function, whereas more persistent errors will more-dramatically and more-continually shift the cell's function.

More generally, the Songline architecture is a very different approach to building circuits for processing information, and as such, the potential benefits (and pitfalls) won't be fully realized until it is explored further.

9.5.5 Implementation

Implementing a binary cell is extremely simple, requiring only a small digital memory, a counter, and a bit of logic to manage the cell's two modes. For a Songline processor whose cells store a continuous function, storage of that function is a much more difficult undertaking. Initial attempts have been based primarily on digitizing the domain and range of the function (as well as the inputs and outputs), and then using a standard digital memory. This is straightforward, but has two immediate drawbacks:

(i) potential benefits unique to having a truly continuous-valued function are obviously lost, since the system is now basically again a digital circuit; and
(ii) the memory requirements for storing a function can be significant. For example, assuming 6-bit A-D/D-A converters and three-sided cells, each cell's function is stored in a truth table containing $2^{3\times6}$ rows (since each side contributes 6 bits to the table lookup); each row contains 2×3 outputs (a D and C output on each of the 3 sides); and each output is itself 6 bits wide (since each analog value is stored as a 6-bit binary value). This means a single cell's truth table requires almost 10 million bits of storage. For an 8-bit A-D/D-A, this numbers grows to

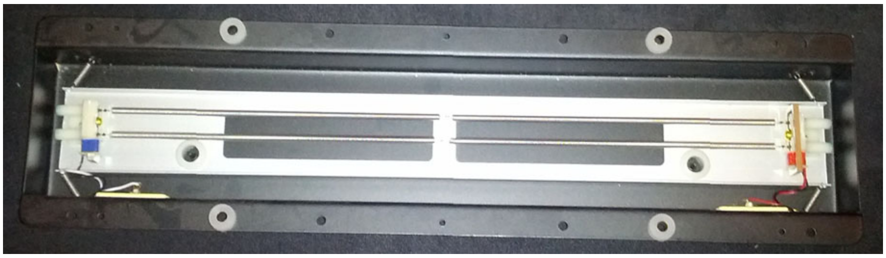


Fig. 9.24 Mechanical spring-based delay unit. An analog signal can be sent into one end of this unit, where it is transformed into mechanical motion that causes the springs to oscillate. The signal travels through the spring to the other end, where a transducer converts the mechanical motion back into an electrical signal, which can be amplified and re-sent to the starting point. This is one way to store an analog signal

8000 million bits; for 12 bit conversions it is close to 5 *trillion* bits. Thus even a modestly-accurate conversion requires a sizable memory.

It is therefore worth considering alternatives to simply digitizing the mapping function and the analog values processed in a Songline processor. One possible improvement is to continue to digitize the domain of the function, but to actually store analog values of the function at each (discrete) point in its domain. It may be feasible to implement such a system by exploiting the high density storage available on flash memory devices but to utilize the ability of each storage element's floating-gate to store an analog value (Wellekens and Van Houdt 2008).

It's worth noting that if we impose certain continuity requirements on stored functions, then the penalty for digitizing its domain may be reduced by incorporating circuitry for interpolating results. Also, note that the main difficulty in increasing the number of sides is only in the domain: for a cell with n sides, we simply implement $2n$ copies of our function storage mechanism within each cell (to drive a D and C output on each of the n sides).

The question of directly storing analog signals is not without precedent: early digital computers used analog memory modules such as mercury delay lines to store information in a series of acoustic waves that would travel the length of the tube, reach one end, be sampled, amplified, and re-transmitted to the other end (Eckert et al. 1971). A spring-based delay line (Fig. 9.24) is another example based on the same principle. There are, however, complications that arise in trying to extract information from such a storage system. Suppose a time-varying signal is stored in a spring (for example). Figure 9.25 shows a trapped sine wave $y = \sin(6\pi x)$ ($x \in [0, 1]$). To evaluate this function f at any given point x, we only need to measure the displacement of the spring from its rest position at a point corresponding to x. Sometimes, this will be the point a distance x from the left of the spring (assuming the length of the spring is normalized to 1), as shown in Fig. 9.26; but because the wave travels, the position of x changes, as shown, for example, in Fig. 9.27.

So how do we measure this displacement? In a typical spring reverb system (such as is used for creating a reverberation effect with a guitar (Amplified Parts 2012),

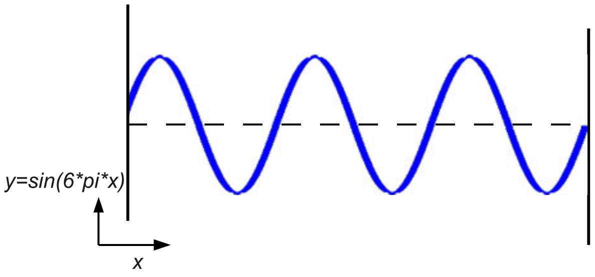


Fig. 9.25 Example of a trapped waveform. Here, the function $y = \sin(6\pi x)$ has setup a displacement in the spring

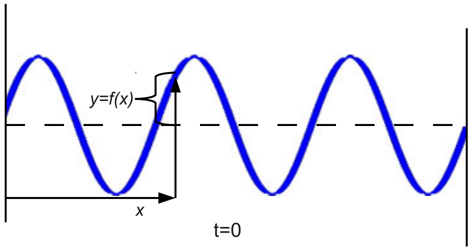


Fig. 9.26 Reading a saved value. To read the value of $f(x)$ at time $t = 0$, one needs to measure the displacement of the spring from its rest position at a distance x from the left side (where the signal is assumed to originate)

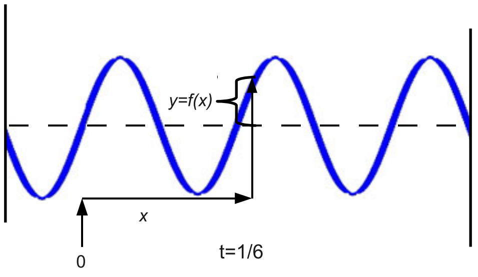


Fig. 9.27 Reading a saved value 1/6 second later. Since the trapped waveform travels down the spring, the position of "x" (and hence $f(x)$) will vary over time

the end of the spring is connected to a transducer that transforms the spring's displacement into an electrical signal. It's thus possible to directly read the value of $f(x)$ for whatever value of x currently corresponds to the rightmost position on the spring. Ergo, assuming it takes 1 second for the wave to travel the entire length of the spring, we could read the value of $f(x)$ by simple waiting for time $t = 1 - x$ and then reading the displacement at the rightmost edge of the spring. This would give us an exact value, but with a time penalty of up to 1 second.

If a 1 second delay is unacceptable, we can instead decide to wait a maximum of $\frac{1}{2}$ second, and read the closest point available to the sensor anywhere during that $\frac{1}{2}$ second. Now we have reduced the time delay, but introduced a potential error in the value we read. More precisely, we will read the exact value of $f(x + \Delta x)$ where $|\Delta x| <= 0.5$.

An alternative is to add a second sensor in the middle of the spring (Fig. 9.28). Now we can be assured that the point x will pass *some* sensor in no more than $\frac{1}{2}$ second. In this setup though we have another option: we can immediately read the value at whichever sensor is closest to the point x, and thus obtain *without delay*

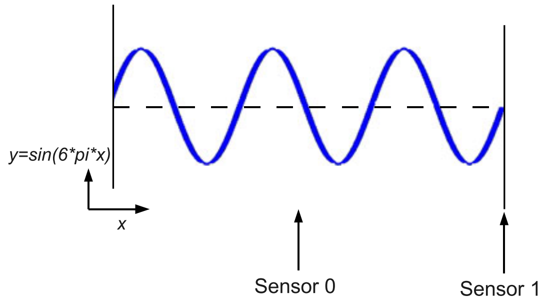


Fig. 9.28 Spring-based storage system with a second sensor. Adding this second sensor (in the middle of the unit) allows $f(x)$ to be sampled with a delay of less than 0.5 s

an *approximation* to $f(x)$. Or we can accept a delay somewhere between 0 and $\frac{1}{2}$ an a possibly smaller error in where we evaluate the function. More-generally, for n evenly-spaced sensors, we can read with a delay of 0 but an error of $\frac{1}{n}$, or we can wait up to $\frac{1}{n}$ seconds and read an exact value. The more sensors, the better our situation. But for any finite number of sensors, we do not have the option of eliminating both the delay in reading a given function value and the potential error in that reading. Sampling at a precisely desired point in time and space is not possible: we can decrease the error in one, but only at the expense of increasing the error in the other. This is perhaps reminiscent of other natural phenomena (Heisenberg 1927), though it's unclear what the actual connection might be.

All of the above is only for a function of one variable $f(x)$. The question of two variables is more complex. Visually, an image of a deformed rubber sheet may be useful: at any point (x, y) the displacement of the sheet corresponds to the function's value $f(x, y)$. If this deformation is setup as a traveling wave, then a situation similar to the one-dimensional case arises, and we could populate the domain with an array of sensors capable of reading the sheet's displacement at certain predefined points. Alternatively, if the sheet is rigidly deformed, a series of movable sensors might be used to take a height reading at a desired point. Here again, we can read without delay by approximating the point (x, y); or we can introduce a delay as a sensor is repositioned and then used to read the value exactly.

A different model for storing a function $f(x, y, z)$ might be to use the temperature at a point (x, y, z) within a three-dimensional region of space as a coding for the function value $f(x, y, z)$. Of course, creating a system to do this (and to maintain the temperature values) is likely impossible, but it suggests the general idea of mapping the domain to 3-space and reading some physical property of each point (x, y, z) in space to discern the value of $f(x, y, z)$ at that point. Another idea that comes to mind is, perhaps, to code information in the phase of light at each point in a 3-D region: perhaps something along the lines of creating and reading a hologram. It's not currently clear how a function $f(x, y, z, w)$ of four variables might be stored physically. To date, most work on Songline processors has been done with three-sided cells.

Yet another alternative for storing functions of more than one variable is (again) to consider a space-filling curve (or an approximation to it) to parameterize a 2-, 3- or 4-dimensional region, mapping it to a one-dimensional region (in a way similar to Fig. 9.21) and using that mapping to locate desired points. Of course, this

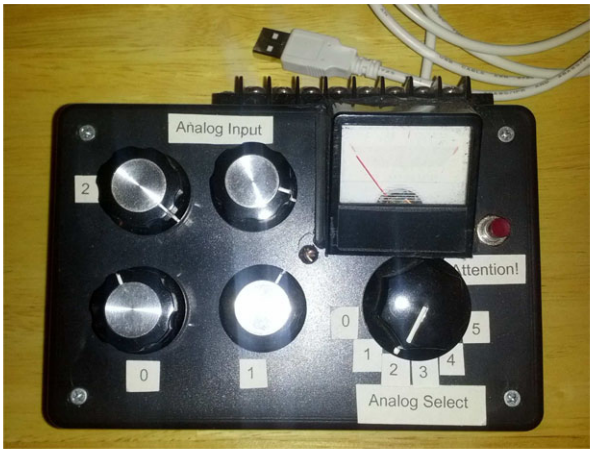


Fig. 9.29 Songline processor prototype. Analog inputs are specified with the knobs, while analog outputs can be read from the meter (the selector below the meter chooses which output channel is displayed). Software running on a USB-connected host machine emulates an 8×8 array of 3-sided cells. 6-bit digitizing is used, resulting in a total function memory of 128 MB

requires certain continuity assumptions on the function being stored, and we are in effect digitizing the domain. But as an approximation, it does provide a way to store functions of higher-dimensional domains using a one-dimensional storage system.

Finally, it should be noted that the C-mode component of a cell's operation—in which the function itself is interrogated and read/written—such parameterization is necessary (as described previously), in which case using this approach to store the function may have its own benefits in terms of consistency of the cell's function.

9.5.6 Songline Processor Prototype

Figure 9.29 shows a prototype of a Songline processor. The dials are used to generate analog inputs, while the meter reads the analog output of one of several channels. Software on a host machine acts as the actual processor (an 8×8 array of 3-sided cells). A compiler converts desired cell behavior (expressed in a C-like syntax) into the internal structures necessary, based on 6-bit digitizing of analog values. A configuration file designates certain edge cells' inputs and outputs as corresponding to particular channels, which the I/O box (shown in the figure) can write and read. Additional connection points are available on the back of the unit, for connection of channels to function generators, oscilloscopes, and so on.

Using this system, initial designs have been developed and tested, including basic amplifiers, differentiators, flip flops, and so on. Cell replication from a source to a target using C-mode has also been successfully demonstrated. The system works well, though the differences from a typical digital system are often surprising (such as the wide range of effects race conditions can exhibit).

Next steps include:

- continuing to develop circuits on this test bed;
- experimenting with single-variable function storage using a spring-based delay system;

- looking into the feasibility of exploiting floating-gate technology as a means of storing and retrieving real-valued quantities;
- trying to better understand the time vs. error tradeoffs of incorporating multiple sensors in the function readback system; and
- looking further into holographic techniques to see if there's some way to use holography to store functions of 2 or 3 variables.

9.6 Status and Future Work

9.6.1 Current Status

Several specifications for the Cell Matrix architecture have been designed (Macias et al. 1999; Durbeck and Macias 2001c; Macias and Raju 2001), and these different implementations have been used in a number of software simulators (Cell Matrix Corporation 2006b). Most work to date has been done using software simulators.

Hardware implementations have also been developed, including a small custom ASIC implementation. More recent implementations have used traditional FPGAs to implement the Cell Matrix architecture (Macias and Durbeck 2004, 2005b; Durbeck and Macias 2001a, 2002), including a self-contained 8×8 Cell Matrix board called the Mod-88 (Macias and Durbeck 2005b), a 4×4 array of which has been placed on the web for use (Cell Matrix Corporation 2006c).

A complete set of tools has been constructed and placed on the web to permit anyone to construct Cell Matrix circuits and debug them using a graphical layout editor and libraries of already-built components (Cell Matrix Corporation 2006b). A place and route tool was recently developed to automatically generate layouts from circuit descriptions (Macias 2006).

9.6.2 Some Applications of Self-Configurability

There are a number of areas where this shift from external- to internal-control appears especially beneficial to the design of computational and processing systems. There has been and continues to be impetus to scale systems to greater numbers of components and greater levels of complexity in the tasks they undertake. As the system's size and complexity are scaled up, the difficulties associated with managing and maintaining the system also increase. When rapid increases in scaling eventually occur (Vinge 1993), it may no longer be practical to continue scaling up current strategies. Instead, new approaches to managing extreme complexity may be required.

Utilizing internal control mechanisms is a better strategy than today's external system control as computers become more complex. A key difficulty in managing extremely complex systems is the dependence on a single, centralized unit for all management tasks. In contrast, one can distribute the management and control of the system among a large number of separate management units, thereby reducing the load on each management unit, while also improving the proximity between each management unit and the circuits that it is managing. Care must be taken to avoid introducing new complexities as the number of management units is itself scaled up.

A possible solution is to use the massive resources of such large-scale systems to solve the very problem being caused by their size, i.e., tackle the problems of large-scale system design by using a large-scale system. Some example areas where this may be applied are:

- Manufacturing Defects: In order to utilize many orders of magnitude more devices, computational systems such as CPUs and FPGAs will need to be able to use imperfect hardware, because it will be too difficult and expensive to build perfect hardware. It thus seems there will need to be a design shift, from systems that are completely free of defects to systems that can handle such defects. One way to approach this challenge is to have the system itself perform initial checks on its own hardware, testing its subsystems in an efficient (parallel) manner, and noting defective regions in a way that subsequent processing can use to avoid those defects.
- Run-time Defects: Even in systems that are manufactured perfectly, or whose defects have been worked-around, there are still run-time defects, i.e., temporary or permanent errors that occur while the system is operating. Given the large number of components expected in future systems, the job of monitoring them for proper functioning cannot reasonably be handled from a single, centralized (i.e., external) location. Instead, the job of defect detection and mitigation may better be handled from within the system, using the large number of system components as a resource for handling the complexity of this task.
- System Design: With a jump to Avogadro-scale systems, one could expect system design times to become prohibitively long. One example of applying our thesis to this problem would be to use a massive FPGA to implement and run massively-parallel CAD tools, which are then used for subsequent design work. Another is to decouple system design from system construction, so that system construction happily continues to march down Moore's Law curve, while the increasingly complex task of systems design has time and opportunity to develop as resources are made available.
- Initial System Configuration/Bootstrap: As systems scale to use many orders of magnitude more devices, the problem of configuring or bootstrapping them (the process of specifying their initial setup in the case of such soft hardware as FPGAs, and the bootstrapping process of invoking and initializing all processes that constitute the running system in the case of all computation systems including FPGAs and computers) rapidly becomes worse, until configuration and initialization times may be so long that systems cannot even complete initialization. Again, a possible solution is to use the massively-parallel system itself as the control system used to implement a very powerful, parallel bootstrap system.

9.6.3 Possible Manufacturing Options

Because even simple circuitry implemented on the Cell Matrix tends to consume a large number of cells, practical hardware Cell Matrices are difficult to create using conventional manufacturing techniques. There are, however, a number of conceivable approaches to manufacturing large Cell Matrices ("large" means a Cell Matrix containing a large number of cells).

9.6.3.1 Aggressive Silicon Techniques

One approach is to use aggressive techniques in silicon, such as deep sub-micron technology, while taking advantage of the fault-handling capabilities of the Cell Matrix to manage the inevitably-high defect count. This, of course, requires a mitigation technique that costs less (in terms of area/cell count) than what is gained by using a very aggressive fabrication technology.

An added benefit to using cutting-edge technologies to manufacture a Cell Matrix is the possibility of using the Cell Matrix itself as a *process driver*, i.e., as a means of debugging the fabrication process itself (Durbeck and Macias 2002). Because a Cell Matrix has an inherent introspection capability, it can be used to analyze the characteristics of the manufactured cells within itself, including things such as their logical behavior, the speed of their operation, the pattern of defective cells' locations, and so on. Because pathways from cell to cell are built out of cells, there are generally multiple pathways from any cell to any other cell. This means that even regions containing a large number of defects might be thoroughly analyzed (Durbeck and Macias 2002).

9.6.3.2 Wafer-Scale Integration

Another potential approach to manufacturing large Cell Matrices is to employ wafer-scale integration (WSI) (Saucier and Trilhe 1986; Wyatt and Raffel 1989; Fuchs and Swartzlander 1992; IEEE 1989–1995; Zeng et al. 2005), where, instead of dicing a wafer into a number of individual chips, the entire wafer is used to implement a single circuit. WSI has been explored as a means to overcome die-level defects, and in general has to address the problem of a wafer typically containing some defective die (e.g., Boubekeur et al. 1992; Saucier et al. 1988). Various special-purpose architectures have been developed for WSI to allow operation on top of imperfect wafers (e.g., Boubekeur et al. 1992; Saucier et al. 1988), as well as a general-purpose methodology for using wafer-scale fabrication (Alam et al. 2002).

Of course, since a Cell Matrix is inherently a fault-isolating architecture, and since faults can be detected and managed efficiently using some of the techniques described above, it is an ideal architecture for implementing using WSI.

9.6.3.3 Three-Dimensional Fabrication

While most discussion of the Cell Matrix architecture in this chapter has focused on two-dimensional matrices, it is perfectly feasible to create the three-dimensional Cell Matrix. A three-dimensional Cell Matrix is actually much more powerful than a two-dimensional one, for a number of reasons:

- each cell is more powerful, being a 6-input 6-output device (assuming a cube-based structure/topology);
- circuit routing is easier, at least for two-dimensional circuits, since, being embedded in a higher-dimensional space, non-adjacent components can be connected via the third dimension;
- for a given number of components, the maximum path length, and hence maximum delay, can be greatly decreased—for example, a square circuit containing a trillion cells would be 1,000,000 $\times$ 1,000,000 cells, giving a maximum corner-to-corner path length of 2,000,000 cells; while a cubic circuit containing a trillion cells would be 10,000 $\times$ 10,000 $\times$ 10,000, giving a maximum corner-to-corner path length of only 30,000 cells;
- configuration of a three-dimensional circuit can be much faster than configuration of a two-dimensional circuit with the same cell count; in the above example, the two-dimensional case would require 2,000,000 operations (1,000,000 to construct one row of Supercells, and another 1,000,000 for each Supercell to create a column of Supercells), while the three-dimensional case would require only 30,000 operations (10,000 to make a row of Supercells; another 10,000 to create a column of 10,000 Supercells below each of those, thus giving a two-dimension plane of Supercells; and a final 10,000 operations for each of *those* 100,000,000 Supercells to create a line of Supercells in the Z-axis);
- a three-dimensional matrix can be treated as a series of two-dimensional matrices with inter-matrix access in the Z dimension, thus enabling techniques such as rapid context switching, parallel reading of cells, parallel writing of cells, plane-to-plane voting, and so on; and
- a three-dimensional matrix has a number of two-dimensional surfaces available to the outside world, thus affording a high-bandwidth mechanism for parallel input and output of data to and from the matrix.

There have been various attempts by researchers to create three-dimensional circuitry (Seeman 1982; DePreitere and a 1994; Alexander et al. 1995; Borriello et al. 1995; Meleis et al. 1997; Leeser et al. 1997). One problem often encountered is heat buildup, but this need not be an issue with a massively parallel architecture such as the Cell Matrix, since the idea is to get algorithm speedup through the use of massively-parallel algorithms, rather than through raw uni-processor speed. Another significant impediment for three-dimensional fabrication is, again, the manufacturing defect rate, but this can be (to some degree) mitigated using Cell Matrix fault handling techniques.

9.6.4 Nanotechnology

Any discussion of high-density, three-dimensional fabrication inevitably leads to the topic of Nanotechnology (Montemerlo et al. 1996; Kamins and Williams 2001; Stan et al. 2003). Roughly speaking, nanotechnology is the science of atomic-scale manufacturing. In the context of using nanotechnology to manufacture Cell Matrices, our primary interest is not so much in the size of the manufactured cells per se, but rather in the extremely high cell count that can be achieved because of that size. That is, our interest is not in making small Cell Matrices, but rather in creating Cell Matrices containing a huge number of cells. This might involve manufacturing cells out of logic gates that themselves are comprised of a small number (say 10–1,000) of atoms, single electrons, etc., with each resulting cell containing fewer than a million atoms. At such a scale, we can talk about quantities such as one *mole* of cells, i.e., a number of cells equal to Avogadro's numbers, or roughly 10^{23} cells. Note that a three-dimensional Cell Matrix containing 10^{23} cells would have only 100,000,000 cells along each edge.

9.6.5 Cell Matrix Support for Nanotechnology

While the Cell Matrix architecture stands to benefit a great deal from a true atomic-scale fabrication technologies, it is also possible that such technologies may also benefit from the Cell Matrix. For example, in trying to manufacture three-dimensional circuitry, one often manufactures a series of two-dimensional layers using conventional techniques (e.g., lithographic techniques), and then stacks these layers in the third dimension. While the former process is usually quite precise, the latter is not: it involves the manipulation of macro-scale objects, such as individual silicon die or silicon wafers (Fraunhofer Institute for Reliability and Microintegration 2006; Misc 2006; IEEE 1989–1995; Ababei et al. 2004).

However, if the layers being stacked are actually Cell Matrices, the cells themselves can be used to create circuits on each layer that investigate their own placement relative to the layers above and below them. By staggering the placement of the cells within each layer, that is, using non-uniform spacing between cells, it may be possible to guarantee that some of the cells will align between layers (while also guaranteeing that some cells will not). By voluntarily sacrificing some of the cells, we can insure that some of the cells will create inter-layer connections. Circuits can then be constructed to discover where these connections have been made, and that information used in the configuration of subsequent circuits.

9.6.6 Other Approaches to Manufacturing

When thinking about manufacturing a Cell Matrix, it is worthwhile to consider approaches that may lack characteristics typically required for conventional circuit manufacture. Some of the characteristics that differ for a Cell Matrix manufacturing process vs. conventional circuits include:

- speed—a Cell Matrix is not required to have extremely fast cells, since one may hope to obtain algorithm speedup via massive parallelism rather than raw component speed;
- power dissipation—because, again, the individual cells can be run slowly, with overall speedup achieved via parallelism;
- reliability—as we have seen, perfect manufacture is not strictly necessary for the creation of a viable Cell Matrix: a certain degree of defects is acceptable; and
- physical size—the primary requirement for a useful Cell Matrix is not that it be physically small, but rather that it contain a huge number of cells.

Taking these considerations into account, there are some unconventional possibilities for manufacturing a Cell Matrix. One possibility is to use printable circuit technology (Plastic Logic 2006; Burns et al. 2004; Sirringhaus et al. 2006; MacDonald 2006; Wong et al. 2006) to create two-dimensional sheets of cells. These sheets could be folded and stacked, to create narrow, but arbitrarily-long matrices. Alternatively, single sheets could be stacked, and connectors pierced through the sheets to create inter-sheet connections. Even if only some cells are connected from sheet-to-sheet, this would still offer some of the benefits of a fully three-dimensional matrix.

This also leads to the notion of trying to weave a matrix, utilizing some of the ideas being used for the design of smart clothing (Cakmakci and Koyuncu 2000; Cakmakci et al. 2001; Martin 2006; Edmison et al. 2002; Martin et al. 2003; Marculescu et al. 2003). If cells can be constructed simply by controlling the pattern of a weaving, then, again, arbitrarily-long two-dimensional sheets could be manufactured. This approach is worth considering if only because the manufacture of textiles is one of the oldest manufacturing processes in human history, and is estimated to date back 12,000 years (Sabalan Group 2006).

Because of the regular structure of the Cell Matrix, it may be possible to exploit natural processes in its manufacture: for example, the process of crystal growth. Of course, this would require a means for associating logic circuits with certain types of crystals, arranging things so that as the crystal grows, so does the matrix. A more-controlled approach is being taken in the use of DNA as a scaffolding for the placement of carbon nanotubes (Seeman 1982, 2003; Winfree 1998, 2003; Winfree et al. 1998; Dwyer et al. 2004a, 2004b; Patwardhan et al. 2004, 2006; Park et al. 2006; Pistol et al. 2006; Kim et al. 2004; Winfree and Bekbolatov 2004; Rothemund et al. 2004; Robinson and Seeman 1987), which could be applied to the construction of Cell Matrix cells.

9.6.7 CAD Issues—Magic Polygons

It remains unanswered how best to represent self-modifying circuitry such as that which can be implemented on the Cell Matrix. One idea is to use the notion of

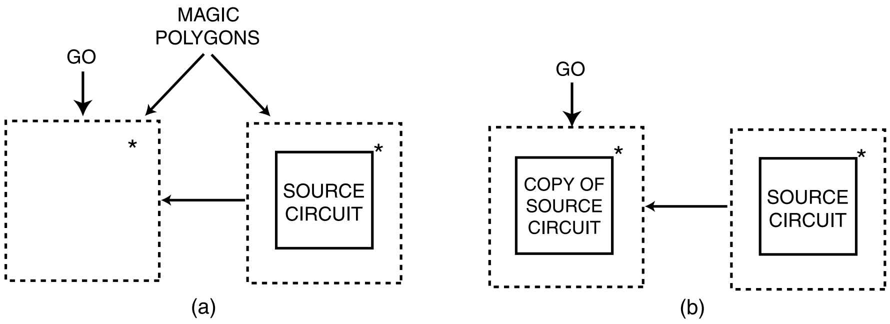


Fig. 9.30 Serialization of a source circuit. In (**a**), the truth tables comprising the source circuit are transmitted along the wire to the de-serializer on the left. "*" is an anchor point for positioning the new circuit. In (**b**), the de-serializer has synthesized a copy of the source circuit using the serial bit stream it receives

serialization, wherein a collection of cells is used to generate a stream of binary data corresponding to it. Once a circuit has been serialized, it can be processed using standard digital circuits: it can be stored, retrieved, steered through circuitry via mux/demux logic, compared with other streams, and so on. Along with serialization is the process of *de-serialization*, wherein a stream corresponding to a circuit is used to configure a set of cells to implement that circuit.

Figure 9.30a shows an example of this serialization/de-serialization process. A sample source circuit is shown on the right, surrounded by a dashed box called a "Magic Polygon," which indicates a region of the circuit that is to be serialized. The line coming from the box transfers this stream of ordinary binary data to the de-serializer on the left, where a new copy of the circuit is created in Fig. 9.30b. The "*" inside each polygon is used to position de-serialized circuitry relative to the original serialized circuit. The GO signal indicates initiation of the de-serialization operation. Figure 9.30 shows a simple circuit replication.

Figure 9.31 shows an example of a simple Hardware Library, where one of four circuits can be selected for synthesis. The bitstreams for each circuit are sent into the 4-1 selector, which chooses one of them for synthesis by the Magic Polygon to the South.

Figure 9.32 shows a more-complex example. In this case, the Magic Polygon on the right itself contains a Magic Polygon. This means the synthesized circuit will also contain a Magic Polygon. When the circuit in Fig. 9.32a is serialized and then de-serialized, the circuit of Fig. 9.32b results. This looks exactly like Fig. 9.32a, except that the East-to-West wire has been extended. Each time the circuit's bitstream is de-serialized, the wire is extended another step. This can be used as a mechanism for synthesizing wires, as described above in Sect. 9.3. Note that in this example, the GO line has not been shown. The Magic Polygon on the left needs a GO signal in order to initiate its de-serialization process, but since the location of this Magic Polygon changes, the line driving the GO input also needs to be extended at each step. For simplicity, this and other details have been excluded from Fig. 9.32.

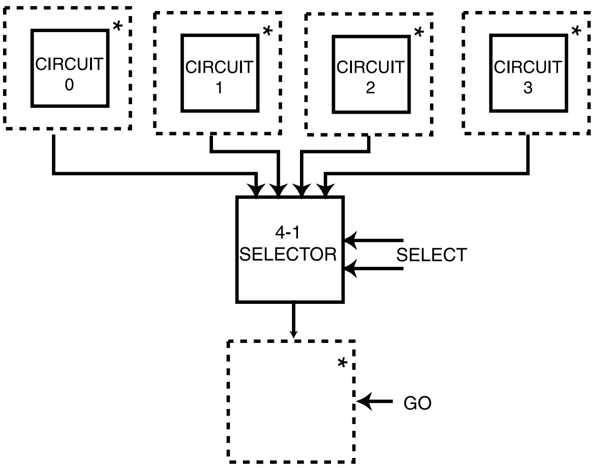


Fig. 9.31 Example of a Hardware Library. The 4-1 Selector can choose the bitstream corresponding to one of four circuits. The selected circuit will be synthesized to the South

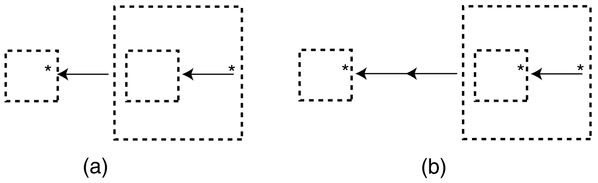


Fig. 9.32 Magic Polygon Representation of an Extendible Wire. (**a**) shows the initial setup. (**b**) shows the result after deserialization and reserialization: the wire has been extended

While Magic Polygons are an easy way to visualize the serialization/de-serialization process, they are more applicable to circuit schematics than to, say, a Hardware Definition Language (HDL) description of a circuit. Ongoing research efforts seek to elaborate on the details of Magic Polygons, to extend them to the realm of HDLs, and to develop tools for implementing their functional behavior.

References

Ababei, C., Maidee, P., & Bazargan, K. (2004). Exploring potential benefits of 3D FPGA integration. In *Field-programmable logic and its applications* (pp. 874–880). Heidelberg: Springer.

Abdi, H. (1994). A neural network primer. *Journal of Biological Systems*, *2*(3), 247–283.

Alam, S., Troxel, D., & Thompson, C. (2002). A comprehensive layout methodology and layout-specific circuit analyses for three-dimensional integrated circuits. In *ISQED international symposium on quality electronic design, 2000* (p. 246). Washington: IEEE Computer Society.

Alexander, M., Cohoon, J., Colflesh, J., Karro, J., & Robins, G. (1995). Three-dimensional field-programmable gate arrays. In *Proceedings of the eighth annual IEEE international ASIC conference and exhibit, 1995* (pp. 253–256).

Amplified Parts (2012). Spring reverb tanks explained and compared. http://www.amplifiedparts.com/tech_corner/spring_reverb_tanks_explained_and_compared. Retrieved November 2012.

Arms, K., & Camp, P. (1987). *Biology* (3rd ed.). Philadelphia: Saunders.

Aspray, W., & Burks, A. (1987). *Charles Babbage Institute reprint series for the history of computing: Vol. 12. Papers of John von Neumann on computing and computer theory*.

Bachman, G., & Narici, L. (1966). *Functional analysis*. San Diego: Academic Press.

Borriello, G., Ebeling, C., Hauck, S., & Burns, S. (1995). The Triptych FPGA architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *3*(4), 491–501.

Boubekeur, A., Patry, J., Saucier, G., & Trilhe, J. (1992). Configuring a wafer-scale two-dimensional array of single-bit processors. *Computer*, *25*(4), 29–39.

Burns, S., Kuhn, C., Jacobs, K., MacKenzie, J., Ramsdale, C., Arias, A., Watts, J., Etchells, M., Chalmers, K., Devine, P., et al. (2004). Printing of polymer thin-film transistors for active-matrix-display applications. *Journal of the Society for Information Display*, *11*, 599.

Cakmakci, O., & Koyuncu, M. (2000). Integrated electronic systems in flexible and washable fibers. Filed with the United States Patent Office and the European Patent Office.

Cakmakci, O., Koyuncu, M., & Eber-Koyuncu, M. (2001). Fiber computing. In *Proc. of the workshop on distributed and disappearing user interfaces in ubiquitous computing, CHI*.

Cell Matrix Corporation (2006a). Bibliography for cell matrix-related research. http://www.cellmatrix.com/entryway/products/pub/bibliography.html.

Cell Matrix Corporation (2006b). Cell matrix software. http://www.cellmatrix.com/entryway/products/software/software.html.

Cell Matrix Corporation (2006c). MOD 88 online viewer. http://cellmatrix.dyndns.org:12001/cgi-bin/mod88/obs2.cgi?.

Chatwin, B. (1986). *The songlines*. Baltimore: Penguin Books.

Darwin, C. (1859). *The origin of species by means of natural selection: or, the preservation of favoured races in the struggle for life*. London: Murray.

DePreitere, J. et al. (1994). An optoelectronic 3D field programmable gate array. In W. Hartenstein, M. Servit, & (Eds.), *Lecture notes in computer science: Vol. 849. Field-programmable logic: architectures, synthesis, and applications*. Berlin: Springer.

Deutsch, L., & Schiffman, A. (1984). Efficient implementation of the smalltalk-80 system. In *Proceedings of the 11th ACM SIGACT-SIGPLAN symposium on principles of programming languages* (pp. 297–302). Salt Lake City: ACM.

Duncan, R. (1989). Design goals and implementation of the new high performance file system. *Microsoft Systems Journal*, *4*(5), 1–14.

Durbeck, L., & Macias, N. (2001a). *Autonomously self-repairing circuits* (NASA SBIR Phase II Proposal).

Durbeck, L., & Macias, N. (2001b). *Autonomously self-repairing circuits* (NASA SBIR Phase I Final Report).

Durbeck, L., & Macias, N. (2001c). Self-configurable parallel processing system made from self-dual code/data processing cells utilizing a non-shifting memory. US Patent 6,222,381.

Durbeck, L., & Macias, N. (2001d). The cell matrix—an architecture for nanocomputing. *Nanotechnology*, *12*(3), 217–230.

Durbeck, L., & Macias, N. (2002). Defect-tolerant, fine-grained parallel testing of a cell matrix. In *Proceedings of SPIE ITCom* (Vol. *4867*). Boston: SPIE.

Dwyer, C., Johri, V., Patwardhan, J., Lebeck, A., & Sorin, D. (2004a). Design tools for self-assembling nanoscale technology. *Nanotechnology*, *15*(9), 1240–1245.

Dwyer, C., Poulton, J., Taylor, R., & Vicci, L. (2004b). DNA self-assembled parallel computer architectures. *Nanotechnology*, *15*(11), 1688–1694.

Eckert, J. P., et al. (1971). *The UNIVAC system*. New York: McGraw-Hill, reprinted in *Computer structures: readings and examples*.

Edmison, J., Jones, M., Nakad, Z., & Martin, T. (2002). Using piezoelectric materials for wearable electronic textiles. In *Proceedings of sixth international symposium on wearable computers (ISWC 2002)* (pp. 41–48). Berlin: Springer.

Fischer, T. (1987). Heavy-ion-induced, gate-rupture in power MOSFETs. *IEEE Transactions on Nuclear Science*, *34*(6), 1786–1791.

Fraunhofer Institute for Reliability and Microintegration, Munich (2006). Department of Si Technology and Vertical System Integration. http://www.izm-m.fraunhofer.de/files/fraunhofer2/si-technology__vsi.pdf. Accessed 10/31/2006.

Fuchs, W., & Swartzlander, E. Jr (1992). Wafer-scale integration: architectures and algorithms. *Computer*, *25*(4), 6–8.

Greenwood, G., & Tyrrell, A. (2006). *Introduction to evolvable hardware*. New York: Wiley/IEEE Press.

Haldane, J. (1931). *The philosophical basis of life*.

Heisenberg, W. (1927). Werner Heisenberg, in a letter to Wolfgang Pauli. February 1927.

IEEE (1989–1995). *Proceedings of the international conference on wafer scale integration*.

Kamins, T., & Williams, R. (2001). Trends in nanotechnology: self-assembly and defect tolerance. In *Proc. NSF partnership in nanotechnology conf.*

Kauffman, S. (1993). *The origins of order: self-organization and selection in evolution*. London: Oxford University Press.

Kim, J., Hopfield, J., & Winfree, E. (2004). Neural network computation by in vitro transcriptional circuits. *Advances in Neural Information Processing Systems*, *17*, 681–688.

Kluwer (1998). *Analog Integrated Circuits and Signal Processing*, *17*(1–2). Special issue on Field-Programmable Analog Arrays.

Koza, J. (1992). *Genetic programming: on the programming of computers by means of natural selection*. Cambridge: Bradford Book.

Leeser, M., Meleis, W., Vai, M., & Zavracky, P. (1997). Rothko: a three dimensional FPGA architecture, its fabrication, and design tools. In *Seventh international workshop on field programmable logic and applications*. London: Springer.

Lennox, J. (2001). *Aristotle's philosophy of biology: studies in the origins of life science*. Cambridge: Cambridge University Press.

MacDonald, W. A. (2006). Advanced flexible polymeric substrates. In H. Klauk (Ed.), *Organic electronics: materials, manufacturing & its applications*. New York: Wiley.

Macias, N. (1999). The PIG paradigm: the design and use of a massively parallel fine grained self-reconfigurable infinitely scalable architecture. In *Proceedings of the first NASA/DOD workshop on evolvable hardware (EH'99)*. Pasadena: IEEE.

Macias, N. (2001). Circuits and sequences for enabling remote access to and control of non-adjacent cells in a locally self-reconfigurable processing system composed of self-dual processing cells. US Patent 6,297,667.

Macias, N. (2006). *Cell matrix place and route tool: changes and improvements*. White Paper delivered to Los Alamos National Laboratory under sub-contract #90843-001-04 4x.

Macias, N., & Durbeck, L. (2002). Self-assembling circuits with autonomous fault handling. In *Proceedings of NASA/DoD conference on evolvable hardware, 2000* (pp. 46–55). Washington: IEEE.

Macias, N., & Durbeck, L. (2004). Adaptive methods for growing electronic circuits on an imperfect synthetic matrix. *Biosystems*, *73*(3), 172–204.

Macias, N., & Durbeck, L. (2005a). Unpublished white papers and talks delivered to Los Alamos National Laboratory under sub-contract #90843-001-04 4x.

Macias, N., & Durbeck, L. (2005b). A hardware implementation of the cell matrix self-configurable architecture: the cell matrix MOD 88. In *Proceedings of 2005 NASA/DoD conference on evolvable hardware* (pp. 103–106). Washington: IEEE.

Macias, N., & Raju, M. D. (2001). Method and apparatus for automatic high-speed bypass routing in a cell matrix self-configurable hardware system. US Patent 6,577,159.

Macias, N., Henry, L. III, & Raju, M. (1999). Self-reconfigurable parallel processor made from regularly-connected self-dual code/data processing cells. US Patent 5,886,537.

Mange, D., Sipper, M., Stauffer, A., & Tempesti, G. (2000). Toward self-repairing and self-replicating hardware: the embryonics approach. In *Proceedings of the second NASA/DoD workshop on evolvable hardware, 2000* (pp. 205–214). Palo Alto: IEEE.

Marculescu, D., Marculescu, R., Zamora, N., Stanley-Marbell, P., Khosla, P., Park, S., Jayaraman, S., Jung, S., Lauterbach, C., & Weber, W. (2003). Electronic textiles: a platform for pervasive computing. *Proceedings of the IEEE*, *91*(12), 1995–2018.

Martin, T. (2006). Tom Martin's Wearable Electronic Textiles research group at Virginia Tech. http://www.ccm.ece.vt.edu/etextiles/, http://www.ccm.ece.vt.edu/etextiles/publications/. Accessed 10/31/2006.

Martin, T., Jones, M., Edmison, J., & Shenoy, R. (2003). Towards a design framework for wearable electronic textiles. In *Proceedings of seventh IEEE international symposium on wearable computers, 2003* (pp. 190–199).

Meleis, W., Leeser, M., Zavracky, P., & Vai, M. (1997). Architectural design of a three dimensional FPGA. In *Proceedings of seventeenth conference on advanced research in VLSI, 1997* (pp. 256–268). Ann Arbor: IEEE.

Misc (2006). International Journal of Chip-Scale Electronics, Flip-Chip Technology, Optoelectronic Interconnection and Wafer-Level Packaging. http://www.chipscalereview.com. Accessed 10/31/2006.

Montemerlo, M., Love, J., Opiteck, G., Goldhaber-Gordon, D., & Ellenbogen, J. (1996). *Technologies and designs for electronic nanocomputers* (MITRE Tech. Rep. MTR 96W0000044). The MITRE Corporation, McLean, VA. July.

Ortega-Sanchez, C., Mange, D., Smith, S., & Tyrrell, A. (2000). Embryonics: a bio-inspired cellular architecture with fault-tolerant properties. *Genetic Programming and Evolvable Machines*, *1*(3), 187–215.

Page, I. (1996). Constructing hardware-software systems from a single description. *Journal of VLSI Signal Processing*, *12*(1), 87–107.

Park, S., Pistol, C., Ahn, S., Reif, J., Lebeck, A., Dwyer, C., & LaBean, T. (2006). Finite-size, fully-addressable DNA tile lattices formed by hierarchical assembly procedures. *Angewandte Chemie*, *45*, 735–739.

Patwardhan, J., Dwyer, C., Lebeck, A., & Sorin, D. (2004). Circuit and system architecture for DNA-guided self-assembly of nanoelectronics. In *Proceedings of 2004 conference on foundations of nanoscience: self-assembled architectures and devices* (pp. 344–358). Snowbird: Science Technica.

Patwardhan, J., Dwyer, C., Lebeck, A., & Sorin, D. (2006). NANA: a nano-scale active network architecture. *ACM Journal on Emerging Technologies in Computing Systems*, *2*(1), 1–30.

Pearn, J. (2000). Email conversation with. N. Macias, May 2000. http://www.artificialbrains.com.

Pistol, C., Lebeck, A., & Dwyer, C. (2006). Design automation for DNA self-assembled nanostructures. In *Proceedings of the 43rd annual conference on design automation* (pp. 919–924). New York: ACM.

Plastic Logic (2006). Plastic Logic, developer of printed flexible thin film transistor (TFT) arrays. http://www.plasticlogic.com/technology.php. Accessed 10/31/2006.

Prodan, L., Tempesti, G., Mange, D., & Stauffer, A. (2003). Embryonics: electronic stem cells. In H. Abbass, R. Standish, & M. Bedau (Eds.), *Artificial Life VIII: proceedings of the eighth international conference on artificial life* (pp. 101–105). Cambridge: MIT Press.

Robinson, B., & Seeman, N. (1987). The design of a biochip: a self-assembling molecular-scale memory device. *Protein Engineering Design and Selection*, *1*, 295–300.

Rothemund, P., Papadakis, N., & Winfree, E. (2004). Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, *2*(12), 2041–2053.

Sabalan Group (2006). Textile history. http://www.sabalangroup.com/aboutus-history-textilehist-en.html.

Sagan, H. (1994). *Space-filling curves*. Berlin: Springer.

Saha, C., Bellis, S., Mathewson, A., & Popovici, E. (2004). Performance enhancement defect tolerance in the cell matrix architecture. In *Proceedings of MIEL* (Vol. 2, pp. 777–780).

Saucier, G., & Trilhe, J. (1986). *Wafer scale integration*. Amsterdam: North-Holland.

Saucier, G., Patry, J., & Kouka, E. (1988). Defect tolerance in a wafer scale array for image processing. In *Proc. int'l workshop on defect and fault tolerance in VLSI systems*, Univ. of Massachusetts, Amherst, October (Vol. 8, pp. 8.2-1–8.2-13).

Schmit, H. (1997). Incremental reconfiguration for pipelined applications. In *IEEE symposium on FPGAs for custom computing machines* (pp. 47–55). Napa: IEEE.

Seeman, N. (1982). Nucleic acid junctions and lattices. *Journal of Theoretical Biology*, *99*(2), 237–247.

Seeman, N. (2003). Biochemistry and structural DNA nanotechnology: an evolving symbiotic relationship. *Biochemistry*, *42*(24), 7259–7269.

Sirringhaus, H., Sele, C. W., von Werne, T., & Ramsdale, C. (2006). *Manufacturing of organic transistor circuits by solution-based printing*. New York: Wiley.

Stan, M., Franzon, P., Goldstein, S., Lach, J., & Ziegler, M. (2003). Molecular electronics: from devices and interconnect to circuits and architecture. *Proceedings of the IEEE*, *91*(11), 1940–1957.

Thompson, A. (1996). An evolved circuit, intrinsic in silicon, entwined with physics. In *Proceedings of the first international conference on evolvable systems: from biology to hardware* (pp. 390–405). Berlin: Springer.

Trimberger, S. (1998). Scheduling designs into a time-multiplexed FPGA. In *Proceedings of the 1998 ACM/SIGDA sixth international symposium on field programmable gate arrays* (pp. 153–160). New York: ACM.

Vinge, V. (1993). Technological singularity In *VISION-21 symposium sponsored by NASA Lewis Research Center and the Ohio Aerospace Institute*, March.

Waskiewicz, A., Groninger, J., Strahan, V., & Long, D. (1986). Burnout of power MOS transistors with heavy ions of Californium-252. In *IEEE, DNA, Sandia National Laboratories, and NASA, 1986, 23rd annual conference on nuclear and space radiation effects*, Providence, RI, 21–23 July 1986. IEEE Transactions on Nuclear Science (ISSN 0018-9499), 33(pt 1):1710–1713.

Wellekens, D., & Van Houdt, J. (2008). The future of flash memory: is floating gate technology doomed to lose the race? In *2008 international conference on integrated circuit design and technology* (pp. 189–194).

Winfree, E. (1998). *Simulations of computing by self-assembly* (Caltech CS Technical Report 1998.22).

Winfree, E. (2003). DNA computing by self-assembly. *The Bridge*, *33*(4), 31–38.

Winfree, E., & Bekbolatov, R. (2004). Proofreading tile sets: error-correction for algorithmic self-assembly. *DNA Computing*, *9*, 126–144.

Winfree, E., Liu, F., Wenzler, L., & Seeman, N. (1998). Design and self-assembly of two-dimensional DNA crystals. *Nature*, *394*(6693), 539–544.

Wong, W. S., Daniel, J. H., Chabinyc, M. L., Arias, A. C., Ready, S. E., & Lujan, R. (2006). *Thin-film transistor fabrication by digital lithography*.

Wyatt, P., & Raffel, J. (1989). Restructurable VLSI—a demonstrated wafer-scale technology. In *Proceedings of 1st international conference on wafer scale integration, 1989* (pp. 13–20).

Xilinx, I. (2006). Xilinx, Inc. http://www.xilinx.com. Accessed 10/31/2006.

Zeng, A., Lu, J., Rose, K., & Gutmann, R. (2005). First-order performance prediction of cache memory with wafer—level 3D integration. *IEEE Design & Test of Computers*, *22*(6), 548–555.

Chapter 10
Self-Organizing Nomadic Services in Grids

Tino Schlegel and Ryszard Kowalczyk

10.1 Introduction

The Grid has emerged as a global platform to support on-demand computing and on-demand virtual organizations for coordinated sharing of distributed data, applications and processes. The service orientation of the Grid also makes it a promising platform for seamless and dynamic provision of service oriented applications across organizations and computing platforms. Service oriented computing enable new kinds of flexible business applications in open systems, which has changed the way of thinking about building, delivering and consuming software over recent years. Services are made available via standard interfaces independent from their underlying platform implementations. The loose coupling, implementation neutrality and flexible configuration of services allow the creation of large networks of collaborating applications. This computing paradigm allows companies to focus on their core competencies by providing complex business applications that are composed of their own services plus services provided by external partners. These new kinds of distributed systems are not only connected clusters of computers in a local-area network. They are loosely coupled components collaborating in a world-wide Service Grid within and across organizational boundaries. The traditional intra-organizational view is shifting to a global perspective.

The OASIS SOA Reference Model group defines Service Oriented Architecture (SOA) as "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations." (OASIS 2006)

T. Schlegel (✉) · R. Kowalczyk
Swinburne Centre for Information Technology Research, Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia
e-mail: tschlegel@ict.swin.edu.au

R. Kowalczyk
e-mail: rkowalczyk@ict.swin.edu.au

The underlying fundamental principle of service oriented computing is the exchange of standardized documents (e.g. XML documents) via standard interfaces and protocols (SOAP, HTTP, etc.) between services located on different hosts. This simple mechanism provides a great flexibility with clear advantages for service grids. However, the advantages of flexibility and service distribution are often in conflict with the increased network traffic produced by highly communicative services in a distributed application. If services are located on distant servers, exchanging documents between them generates network traffic between the corresponding network nodes. Every increase in network traffic causes higher execution time because of a longer time for network transmission and latency compared with a centralized solution.

Besides the communication overhead in a service oriented environment, the organic nature of such a system demands self-managing capabilities for a seamless-link operation. Each service can be part in a vast number of applications. Over the lifetime of a service, interaction partners and quality of service parameters may change over lifetime, which creates an unpredictable environment. Therefore, each individual service must be responsible for managing its own behavior in accordance with agreements established with other services. Self-management and self-organization of services includes self-configuration, self-protection, self-healing, and self-optimization which arise from more than just individual self-organizing capabilities. It is an emergent property of the whole system that can only be achieved by the mutual interactions between services.

The communication overhead in a service oriented computing environment requires a trade-off between the advantages of service distribution and the drawbacks regarding their execution and communication time and costs. Not only will application data becomes larger because of its human readable and interoperable XML representation and increasing multimedia content (Fontana 2004). New user requirements also introduce additional network load for meta-communication, such as semantic information about services, negotiations plus monitoring of quality-of-service parameters between services and other management and configuration information. This communication overhead has to be considered as one of the main practical downsides of the service grids despite the significant increase in network bandwidth observed over recent years (Fontana 2004).

Different communication paradigms for distributed computing have been developed over the past years to reduce the increasing network load (Braun and Rossak 2005; Bruneo et al. 2003). However, none of them can eliminate the problem as all paradigms have advantages and disadvantages depending on a concrete application scenario. The two main communication paradigms are called *remote communication* and *mobile code* whereas the first one is the currently dominant paradigm because of its simplicity, universality regarding execution platforms and its high security. The interacting services located at different locations communicate by exchanging all information via remote messages. This can cause a huge amount of unnecessary network traffic if only small pieces of the transferred information are required. The mobile code paradigm describes the reallocation of code or parts of code to another platform for remote execution. The relocated service can communicate with

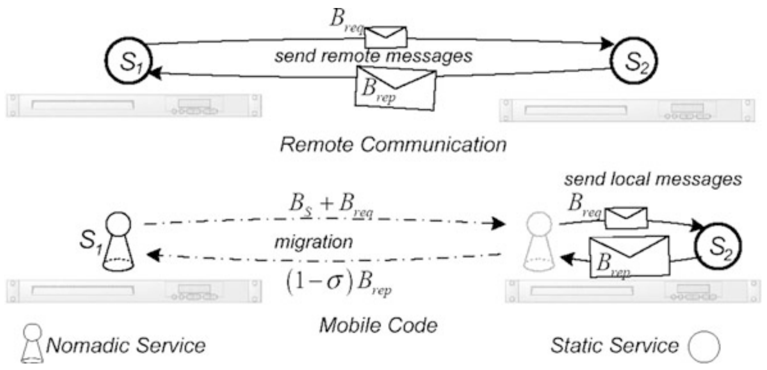


Fig. 10.1 Communication paradigms

its communication partner that is located on the same server locally and returns only the results. Figure 10.1 illustrates the two paradigms and shows the differences between them.

It is obvious that mobile code can reduce network traffic only if the code of the service that has to be transmitted over the network is smaller than the amount of data that can be saved using this paradigm. The problem of deciding between the two communication paradigms is known as the *migration decision problem* (Braun and Rossak 2005).

The software agent community developed mobile agents as an implementation of the mobile code paradigm for the optimization of network and data management in multi-agent systems (Wooldridge 2002). A mobile agent has the capability to migrate with its code and execution state from host to host to fulfill its task. Mobile agents could decide to meet at an agent server and then communicate only locally without generating any network traffic except that of the migration itself (Braun and Rossak 2005). Mobile agents can not only reduce network load and increase application response time but can also improve reliability by making the application more robust against network failures and reduce power consumption in case of mobile devices. Many research groups have investigated and compared network communication costs of different paradigms in terms of network load and processing time in various application scenarios (Outtagarts et al. 1999; Puliafito et al. 2001; Samaras et al. 1999). The main expectation of mobile agents to reduce the network load was not satisfied and interest in the research area of mobile agents has dwindled. Vigna (2004) said that mobile agents are very expensive, cause security problems and in general produce worse performance than other communication paradigms. In fact, no single communication paradigm produces optimal network traffic under all conditions. Optimal performance can only be achieved with a combination of different paradigms depending on the current environment and application parameters (Strasser and Schwehm 1997).

Developments in grid computing supply excellent distributed infrastructures for the realization of service oriented computing, which are able to execute services at distributed heterogeneous resources connected by a network (Buyya et al. 2000;

Foster and Kesselman 1997; Frey et al. 2002). One of the major problems in grid computing is the efficient resource allocation of tasks to available resources. Resource allocation in grids is usually done in accordance with Service Level Agreements (SLA).

Most existing grid toolkits allow the reallocation of a service during execution for runtime optimization. Services that can change their execution platform are called *nomadic services*. Nomadic services can migrate at runtime to a server with more available resources for faster processing or one closer to other services to speed up the network communication. A practical example for service reallocation is the negotiation of quality of service parameters before the actual service execution, by exchanging offers with a number of candidates until an agreement with one of them is reached. The group of interacting services could decide to meet at the same server or a cluster of servers with enough free resources for faster negotiation and only local communication. After the negotiation has finished, the services who reach an agreement can send back a message with the result only. Many distributed service grid applications could benefit from using this paradigm. However, current grid computing toolkits focus on the processing of computational expensive jobs. They usually do not take communication costs into account and the jobs requesting resources cannot influence the resource allocation decision.

In this chapter, we propose a distributed, innovative self-organizing approach to provide intelligent communication and resource management for service grids that increases the services' self-managing capabilities. We eliminate the need for a central facilitator or resource broker. In our distributed resource allocation, solely the nomadic services in the system are responsible for all resource allocation decisions. They consider the amount of available resources as well as the network transmission costs during service execution in their resource allocation decisions.

This approach is based on self-organization of nomadic services. Each nomadic service can decide between remote communication and allocation at the server of the communication partner followed by local communication, provided that enough resources for execution are available. The services learn from past allocation decisions and adapt to new environments in order to minimize network traffic, mitigate network latency and balance resource load between the grid nodes.

Our solution uses short term histories of the last communication acts with other services and resource load information of potential servers for an allocation of resources. Based on this purely local information, a nomadic service forecasts all environment parameters that have an impact on the communication costs of the following communication act and decides on the most beneficial communication paradigm. In addition, the resource loads of potential remote servers that provide resources are predicted. Even if a service would opt for migration when considering only communication costs, a server that is not overloaded also has to be selected. In some cases, a remote communication could be the better alternative if none of the available servers have free resources. Thus an open system needs self-healing and self-organization mechanisms for continuous optimization of the resource allocation and communication management. It takes into account service oriented computing tendencies toward autonomy, heterogeneity and unreliability of resources and services. The solution we present is inspired by the concept of inductive reasoning and bounded rationality introduced by Arthur (1994).

Our self-organizing approach does not require a central authority for controlling, decision support or information sharing between services. The resource allocation in the system is created by the effective competition of services for available resources and is a purely emergent effect. We will demonstrate in various simulation experiments that the services can self-organize in a dynamic service grid environment without the need of active monitoring or a central controlling authority.

10.2 Related Work

The Grid community has developed grid toolkits like Globus (Foster and Kesselman 1997) or Condor-G (Frey et al. 2002) which provide middleware infrastructures for service oriented computing. These toolkits mainly focus on effective resource allocation of computationally very expensive jobs without addressing communication costs. In contrast, service grids are transaction intensive with a large number of services that are not computational expensive.

Existing distributed approaches for resource management in Multi-Agent Systems can avoid server overloading by refusing or queuing incoming mobile agents (Fluess 2005). From the system perspective, the problem of load balancing between different agent servers is important and existing solutions to this problem provide a good supplement to our self-organizing resource allocation.

Most of today's techniques for resource scheduling that can be found in grid computing toolkits like Globus (Foster and Kesselman 1997) or Condor-G (Frey et al. 2002) use a coordinator instance such as an auctioneer, arbitrator, dispatcher, scheduler or manager that needs to have global knowledge about the state of all resources. These resource allocation mechanisms are appropriate for building self contained software systems, but they are not designed to face the challenges of open and dynamic environments, where resources can be added or removed at any time. The Cactus project (Allen et al. 2001) focuses on dynamic resource allocation in grids using a central resource directory to discover and assign available resources, which causes performance problems in large-scale grids.

Recent research in Grid computing has recognized the value of decentralized resource allocation mechanisms and has investigated a number of approaches based on economic market models for trading resources and services for the regulation of supply and demand inspired by principles of real stock markets (Buyya et al. 2002; Clearwater 1996). These approaches use different pricing strategies such as a commodity market model, posted price models or different auction methods. Users try to purchase resources that are required to run the job cheaply, while providers try to make as much profit as possible and operate at full capacity. Buyya et al. (2002) presented a market based economy framework for resource allocation based on the regulation of supply and demand (Buyya 2002) for the Grid Toolkit Nimrod-G (Buyya et al. 2000), having a main focus on job deadlines and budget constraints.

Even if the decision making process in those approaches is distributed, these kind of approaches use a central facilitator during the resource allocation process (Buyya et al. 2000). Another mainly unsolved problem of these approaches is the fine-tuning of price-, time- and budget constraints to enable an efficient resource allocation in large systems (Wolski et al. 2001).

Some approaches for resource allocation in grids, like the Agent based Resource Allocation Model (ARAM) are designed to schedule jobs using agent technology. Agents located on each resource cooperate in order to allocate jobs for an efficient execution. Main drawback of this model is the extensive use of messages for periodic monitoring and information exchange within the hierarchical structure. Subtasks of a job migrate through the network until they find an available resource that meets the price constraints. This migration itinerary is determined by the resource connection topology (Manvi et al. 2005).

There has been considerable work on decentralized resource allocation techniques using game theorya published over recent years. Most of them are formulated as repetitive games in an idealistic and simplified environment (Arthur 1994; Challet and Zhang 1997; Galstyan et al. 2003; Grosu et al. 2002). A self-organizing resource allocation approach for sensor networks based on reinforcement learning techniques is presented in Mainland et al. (2005) with the focus on optimizing energy consumption for the network nodes.

The problem of network communication cost optimization has been addressed in isolation by other research groups. Empirical evaluations showed which communication paradigm performed better regarding network load and processing time in various application scenarios (Outtagarts et al. 1999; Puliafito et al. 2001; Samaras et al. 1999). Depending on the sizes of the exchanged documents, the code sizes, and the network environment it can be decided whether the mobile code paradigm or the remote communication paradigm performs better. Most current solutions of the migration decision problem are based on mathematical models of the application's network load using parameter estimations (number of communication steps, average document sizes, etc.) and suggest a decision at design time. We believe that this type of solution is not sufficient in an open and distributed application scenario. It will be beneficial to address the migration decision problem at run-time, because only then will all influencing parameters (network throughput, latency, document and code sizes) actually be known or can best be approximated.

To illustrate this problem, we use the simple model presented in Braun and Rossak (2005) (compare Fig. 10.1). In the case of remote communication, a request message of size B_{req} is sent to a remote service, which answers with a result message of size B_{res}. The total network load is $B_{RC} = B_{req} + B_{res}$. In the case of migration, nomadic service S_1, which contains code of size B_C and state information of size $B_S + B_{req}$ migrates to the remote server followed by local communication with service S_2. Service S_1 can process the result and extract only necessary information or compress the reply message by a factor of σ $(0 \leq \sigma < 1)$, so that only $(1 - \sigma) \cdot B_{res}$ must be carried back to the home platform. The service does not carry its code or the request message back, because the code is already available at the home server and the request message is obsolete. However,

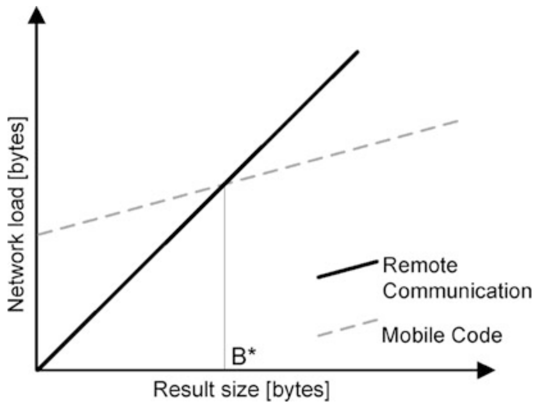


Fig. 10.2 Evaluation of the model for both communication paradigms

new state information must be carried back which accumulates this network load to $B_{MC} = B_C + 2 \cdot B_S + B_{req} + (1 - \sigma) \cdot B_{res}$. An evaluation of the model using an artificial parameter setting shows that there is a break-even point B^*, at which the migration overhead produced by the code and state relocation is exactly compensated for by the amount of data reduction (compare Fig. 10.2).

Picco (1998) advocated estimating all influencing application parameters at design time then deciding between the two paradigms. This approach is suitable for simple scenarios in which all the parameters are known to have constant size. Also in case that the size of parameters (e.g. the result size) is randomly Poisson distributed with parameter λ, and $B^* \ll \lambda$ or $B^* \gg \lambda$ which indicates that the probability for a wrong decision is very low, a static decision is sufficient. In more complicated and dynamic situations a static decision on communication paradigms is error-prone.

Strasser and Schwehm (1997) have improved this static analysis by providing an algorithm to determine the optimal itinerary for a mobile agent for n servers by allowing a combination of both paradigms. In their approach, the agent only migrates to a subset of all servers, whereas others are contacted using remote communication under the assumption of full knowledge. This approach was further improved by collecting all necessary information about network quality using distance maps before planning the optimal migration itinerary (Theilmann and Rothermel 2000), which increased the network load of the system significantly due to active monitoring data.

10.3 Communication Cost Optimization and Resource Allocation

The basic mechanism of our solution is inspired by techniques from adaptive and complex systems using inductive reasoning and bounded rationality principles. A complex adaptive system is "a dynamic network of many agents acting in parallel, constantly acting and reacting to what the other agents are doing. The control of a system tends to be highly dispersed and decentralized. If there is to be any coherent behavior in the system, it has to arise from competition and cooperation among the agents themselves. The overall behavior of the system is the result of a huge number of decisions made every moment by many individual agents" (Waldrop 1992).

10.3.1 The El Farol Bar Problem

Arthur (1994) introduced an ill-defined decision problem, called El Farol bar problem, which assumes and models inductive reasoning. It is probably one of the most studied examples of complex adaptive systems. In the bar problem, 100 people decide independently each week whether to go to a bar on a certain night. Space is limited and people can enjoy the evening only if the bar is not too crowded, defined as less than 60 people in attendance. There is no way to be sure of the number of people in advance. All people have the same preferences and a person goes if he or she thinks they will enjoy themselves. There is no communication allowed between patrons and choices are unaffected by their individual experiences. The only information available to all people is the attendance figure for the past weeks. This problem has no perfect, logical and rational solution. If all believe that few people will go, all will go and if all believe that most people will go, nobody attends the bar invalidating the beliefs in both situations. The only way it can work is if people's expectations differ.

The solution is derived from the human way of deciding ill-defined problems. Humans tend to keep in mind many hypotheses and act on the most plausible one. Therefore, each agent keeps track of the performance of a private collection of its belief models (predictors) and selects the one that is currently most promising for decision making. At the beginning of the simulation, each person is assigned a number of predictors randomly from some predefined set. One of these predictors, called the active predictor is used to predict the next week's attendance, which is the basis for the decision to go or stay at home. After all agents made the allocation decision, the accuracy of every predictor is calculated. The predictor with the best accuracy for the last n weeks becomes the new active predictor for each person. For initialization, reasonable accuracies and past attendance information are chosen. The result of Arthur's simulation is that the attendance figures show a fast stabilizing number of people attending at around the optimal number of 60. Challet et al. (2004) have analyzed this problem and presented a rigorous mathematical model.

Distributed resource allocation in an open and dynamic environment is a similar problem that cannot assume or guarantee a perfect rationality. Services cannot rely on other services they are dealing with and they are forced to guess their behavior. Methods that are needed in such a scenario are not deductive, but inductive.

10.3.2 Adaptation to a Service-Oriented Computing Scenario

Assumptions and constraints of the service grid scenario cannot be directly mapped to the model presented by Arthur (1994). Our assumptions differ as follows:

(i) We do not allow free information dissemination about past server utilization. Nomadic services learn from their own past experience, which means that each service only has information about server utilization for the times when it was at located at the server. The service has no information about resource utilization at other times; in particular the service cannot validate allocation decisions not to go because it has no information for assessment.

(ii) We neither know the number of available system resources and its capacities nor the number of services competing for resources. This means the system must adapt to situations with different numbers of available resources and nomadic services.

(iii) We do not assume that all services make synchronized decisions at the same time.

(iv) We do not have any information on which to base the initialization of the history at the beginning of the life-cycles.

10.3.3 Detection of Service Providers in a Service-Oriented Environment

Services have to know the locations of other services that are needed to accomplish their goals. We assume that each nomadic service has a list with locations of services that can be used. The problem of service discovery is a separate one and out of the scope of our research. However, there are different ways in which services can be retrieved, such as a central service directory (yellow page service) or more sophisticated distributed solutions dependent on the environment and requirements. Services can exchange such location information about other services with their communication partners or browse the service directory when they migrate to another server.

10.3.4 Model

We model a distributed service grid environment as a set of servers $L = \{l_1, \ldots, l_n\}$ and nomadic services $S = \{s_1, \ldots, s_n\}$, each located on its home server $hs(s_i)$ at the beginning of its life-cycle. The map $\mathcal{L} : (S, t) \rightarrow L$ defines the location of each service at time t in general. Services bound to large databases or otherwise immobile legacy systems are called stationary and denoted S^s. Each service has to process a list of communication steps $CS = \langle c_i \mid i : 1 \ldots p \rangle$ as part of its task. A communication step $c_i := \langle s_a, s_b, m_k, m_j \rangle_i$ defines that a request message m_k is sent from service s_a (source) to service s_b (destination), which responds with a reply message m_j. Each message m can be seen as an arbitrary sequence of bytes of length $B_m(m_k)$. The network cost for remote communication comprised of a request message and a reply message is calculated using Eq. (10.1). The cost for the mobile code paradigm with the migration of two nomadic services s_a and s_b to a remote server is calculated by Eq. (10.2). Both services have to carry the codes of size B_C. The requesting service additionally has to carry the request message m_i, and the results of size $(1-\sigma) \cdot B_m(m_j)$ on the way back. If a service migrates to the server of its communication partner, the code size of the not migrating service has to be disregarded.

$$B_{RC}(c_i) = \begin{cases} 0 & \text{if } \mathcal{L}(s_a) = \mathcal{L}(s_b) \\ B_m(m_k) + B_m(m_j) & \text{otherwise} \end{cases} \tag{10.1}$$

$$B_{MC}(c_i) = \begin{cases} 0 & \text{if } \mathcal{L}(s_a) = \mathcal{L}(s_b) \\ B_C(s_a) + B_m(m_k) + B_C(s_b) + (1-\sigma)B_m(m_j) & \text{otherwise} \end{cases} \tag{10.2}$$

Each server l_i has a capacity $\mathcal{C}(l_i, t)$ which may vary over time. Each nomadic service consumes $U(s_i, t)$ server resource for its execution independent of the executing server. The resource load $\mathcal{U}$ of server l_i at time t is calculated using Eq. (10.3).

$$\mathcal{U}(l_i, t) = \sum_{m=1}^{n} U(s_m, t) \mid \mathcal{L}(s_m, t) = l_i \tag{10.3}$$

10.3.5 Algorithm Description

This section describes our self-organizing, distributed resource allocation algorithm for nomadic services that also optimizes the network communication costs of the system. The algorithm is implemented in each nomadic service and is based on beliefs about their environment. The beliefs are modeled using forecasts of environment parameters based on individually observed data. A visualization of the decision making of the algorithm is illustrated in Fig. 10.3. The forecast of the necessary environmental parameters requires short-term histories H of these parameters (Eq. (10.4)). Each history item $h_i = (x_i, y_i)$ is a pair comprising the date x and the value y. The most recent history value is h_0.

$$H_m(l) = (h_0, \ldots, h_i) \mid 0 \leq i < m \tag{10.4}$$

All nomadic services use a set of different predictors for forecasting each parameter. Each predictor in such a predictor set is a function $p : H \to \mathrm{N}^+ \cup \{0\}$ from the history space to a positive integer that is the predicted value. An example predictor for the prediction of the resource utilization is, for instance, the average resource utilization that the nomadic server observed over the last 5 executions at this server.

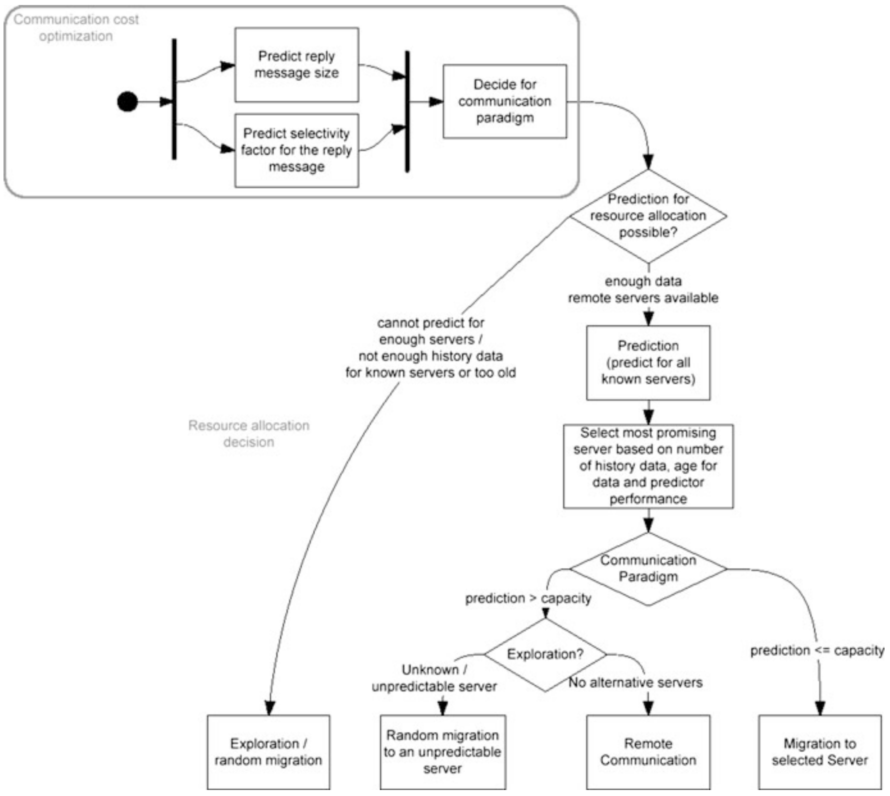

Communication cost optimization
Predict reply message size
Predict selectivity factor for the reply message
Decide for communication paradigm
Prediction for resource allocation possible?
enough data remote servers available
cannot predict for enough servers / not enough history data for known servers or too old
Resource allocation decision
Prediction (predict for all known servers)
Select most promising server based on number of history data, age for data and predictor performance
Communication Paradigm
prediction > capacity
prediction <= capacity
Exploration?
Unknown / unpredictable server
No alternative servers
Exploration / random migration
Random migration to an unpredictable server
Remote Communication
Migration to selected Server

Fig. 10.3 Graphical visualization of the distributed, self-organizing algorithm of a service

All predictors of a set $\mathcal{P} := \{p_i \mid p \in P \wedge i = 1 \ldots k\}$ are randomly chosen from some predefined set of predictors. Each set has one active predictor $p^A \in \mathcal{P}$ that makes the next step's prediction. After each decision, all predictors in the set are evaluated based on their predicted values and a new active predictor is chosen. The decision nomadic service's making process and the selection mechanism of the new active predictors will be described in more detail in the following section.

10.3.6 Communication Costs

The optimization of network communication costs considers only predicted costs for the next communication act. Before the next communication act c_{i+1}, a nomadic service decides which communication paradigm produces lower network load. This decision requires predictions for two unknown parameters—the reply message size and the message selectivity (Fig. 10.3: Communication cost optimization). The active predictors in the set for these parameters will predict the sizes based on the current historical information. Based on the predictions, the nomadic service calculates expected network load for both paradigms with Eq. (10.1) and Eq. (10.2) and selects the better communication paradigm. After the communication act, all predictors of both sets are evaluated based on the accuracies of the predictions, new active predictors for both sets are chosen and history information of both parameters are updated.

The predictor with the smallest accumulated absolute error R over the last l predictions of each set becomes the new active predictor. The absolute error of a prediction is calculated using Eq. (10.5).

$$R = \sum_{i=0}^{l-1} \delta_i; \qquad \delta\left(p^C\right) = |y - \tilde{y}| \tag{10.5}$$

where y is an actual value, and $\tilde{y}$ is the predicted value.

In our first experiments we were keen to learn about the quality of very simple predictors. Actually, it is not as important to achieve a precise prediction of the environment parameters as to make the correct decision based on the predicted values.

All experiments for predictions of environment parameters for the communication cost optimization use the following types of predictors:

- Same value as nth last communication act: $p^C(n) = y_n$.
- Mean value of last n communication acts: $p^A(n) = \frac{1}{n} \cdot \sum_{i=0}^{n-1} y_i$.
- Linear regression over the last n communication acts: $p^L(n,t) = a \cdot t + b$, where a, b are calculated using linear regression with least squares fitting of the last n history values against their observation date.
- Gaussian distributed random value of last n communication acts:

$$p^G(n) = \mathrm{N}\left(\mu, \sigma^2\right); \quad \mu = \frac{1}{n} \cdot \sum_{i=0}^{n} y_i; \sigma^2 = \frac{1}{n} \cdot \sum_{i=0}^{n} (y_i - \mu)^2$$

The technique of using a set of predictors provides good results as the best predictor will be chosen automatically while the less accurate predictors are not considered in the decision process. Specialized predictors can easily be implemented to recognize application specific patterns to increase the accuracy and improve the decision making process. To reduce the computational overhead required for evaluation of all predictors, a user-defined threshold d can be nominated, which only evaluates the active predictor's performance until the relative prediction error ε exceeds d.

10.3.7 *Self-Organizing Resource Allocation*

Before a nomadic service considers a migration to a remote server, it evaluates if the mobile code paradigm is beneficial in terms of network communication costs. Only if the mobile code paradigm is beneficial or the network communication costs are not considered, migration and allocation of resource at other servers is an option. In this section, we focus on the resource allocation algorithm and assume that a nomadic service is willing to allocate resources at a remote server.

The basic prediction mechanism for the distributed resource allocation is similar to the communication cost optimization mechanism. The main difference lies in the selection of the active predictor, which is non-deterministic in the algorithm for the resource allocation. This is to prevent the invalidation of the nomadic services beliefs, which can occur because the allocation decisions of nomadic services competing for a resource are not independent of each other. Competing nomadic services must have different beliefs of the future resource utilization to prevent this previously described invalidation of beliefs and allow a successful adaptation of their strategies to changing environments. A probability distribution over all predictors in a set is created based on the predictor efficiency values $E(p^R)$ over the last l predictions. The selection of the new active predictor is implemented as a roulette-wheel selection according to this distribution (Fig. 10.4). The predictor efficiency is a measure for the correctness of the decisions that a nomadic service made based on the predictions. The probability of selecting a predictor as new active predictor is zero if the predictor cannot predict a value based on the current historical information. This happens in the case that not enough historical information is available. Equation (10.6) is used to calculate the predictor efficiency of a resource utilization predictor, which is the sum of the predictor efficiency ratings of the last l predictions. A positive efficiency rating is given if the prediction led to a correct decision, a negative rating for each prediction that led to a wrong decision and a neutral rating in the case that the decision cannot be assessed due to a lack of information. The prediction accuracy is not considered for the resource allocation as this will not improve the resource allocation decisions. Imagine a server with a capacity of 100 resource units and 90 of them are occupied. Suppose a nomadic service with a resource requirement of 2 units for its execution predicts 103 occupied resource units. Such a prediction indicates over-utilization and an allocation of resources at this server would be dismissed. This is worse than a prediction of 15 occupied resource units which is less accurate but leads to a correct allocation decision. Therefore, the efficiency of predictors is used for the selection of the active predictor.

$$E\left(p^R\right) = \sum_{i=0}^{l-1} e_i \tag{10.6}$$

where

$$e_i = \begin{cases} 1 & \text{if } i\text{th decision was correct} \\ 0 & \text{if } i\text{th decision had unknown outcome} \\ -1 & \text{if } i\text{th decision was wrong} \end{cases}$$

Figure 10.4 shows an example of a probability distribution of a set of resource utilization predictors, which was created using efficiency ratings. Although predictor $P9$ has the highest efficiency in the set, predictor $P5$ was chosen as the active

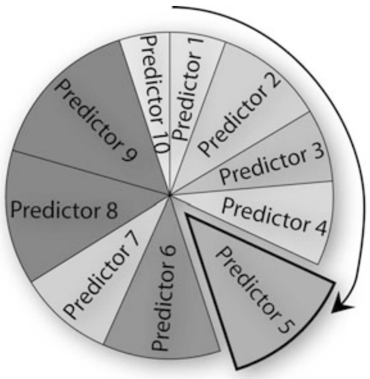


Fig. 10.4 Example probability distribution of a predictor set

predictor. This non-deterministic selection mechanism enables the selection of different active predictors, which leads to different resource utilization predictions for nomadic services even though the set of predictors is the same.

For faster adaptation of the system to a changing environment (different resource capacities, number of servers or nomadic services) and to reduce unnecessary data overhead, only the last l predictions are considered for the calculation of efficiency.

Each nomadic service uses 10 simple predictors per set. All predictors are chosen randomly from a pool of 34 predictors in total. These predictors are chosen from a predefined set which includes all predictor types that are used for the communication cost prediction with different cycles or window sizes, as well as the following two predictors:

- n-frequency distribution predictor: $p^F(n)$ uses a random value from the frequency distribution of the n last history values
- n-mirror predictor: $p^F(n) = 2 \cdot \overline{H} - y_i$ uses the mirror image around the mean value of all history values of the nth last history value.

There is no resource utilization information about other servers available at the beginning of the simulation. Therefore, the algorithm has an initialization phase in which the resource utilization information about other servers is collected. A nomadic service migrates randomly to one server and allocates resources. If a nomadic service never migrates to the server, the resource utilization prediction mechanism and especially the self-organization would never start working due to a lack of historical information. If resource utilization predictions are available, a nomadic service will select a server with expected free resources or chooses remote communication with the partner from its current location. The resource utilization prediction can only be evaluated if the nomadic service migrates to the servers and has the actual resource utilization information. In the case of remote communication, a majority voting of all predictors is used for the evaluation of the all predictors. In addition, old historical information of all servers is deleted using a decay rate dependent from the age of the data. The decay rate increases linearly with the age of the historical data. Recent history information has zero probability below the lower bound and

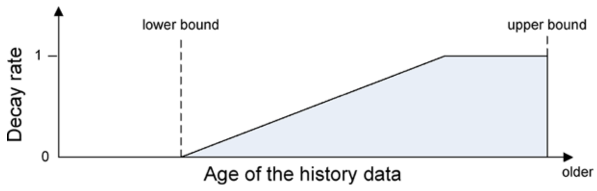


Fig. 10.5 Evaporation of old historical information

probability equal to one for the information older than the upper bound (Fig. 10.5). The decay of old historical information is necessary to allow an adaptation to a dynamic environment. A nomadic service that predicts a server to be overloaded may predict overload in the future because it uses the same historical information, which can only be prevented by updating historical information. Predictions are only as good as the provided historic information. Very old historical information may not allow a reflection of the current system state and is removed.

In most cases, a nomadic service can choose from a set of servers that it could migrate to. In a service oriented environment one service is usually provided by a number of different service providers. A nomadic service can choose between those providers based on their available resources. The service predicts the resource utilization of all potential servers and selects one server from the set of servers that are predicted to have free resources. The selection depends on the predictor set efficiency that calculated using Eq. (10.7). It considers the amount of historical information about the server, the average age of the historical information and the efficiency of the active predictor. These values are transformed into a probability distribution and one server is selected using a roulette wheel selection algorithm.

$$C(\mathrm{P}) = w_1 \cdot \frac{size(H)}{m} + w_2 \cdot \frac{\overline{Age(H)}}{\max(\overline{Age(H)})} + w_3 \cdot \frac{C(p)}{\max(C(p))} \tag{10.7}$$

where:

w_i weights; $w_i \geq 0 \wedge \sum w_i = 1$;
$size(H)$ the number of data in history;
m maximal number of history values;
$\overline{Age(H)}$ average age of historical data.

10.4 Performance Evaluation

The performance of the proposed algorithm was evaluated in various simulation experiments. A number of results presented in the next section show the behavior of the distributed self-organizing resource allocation in different simulated environmental conditions with and without the optimization of communication costs of the system. All experiments were conducted in a special test-bed developed in the Java programming language, independent from concrete grid toolkits. The simulation environment enabled different configurations for all model parameters that influence communication costs and resource allocation such as the number of servers, resource capacities of the servers, the number of static and nomadic services, the number of communications steps, etc. A discrete event simulation model was used to trigger all events that change the state of the system. The conducted experiments were divided into sections that focus on different aspects of our model and its behavior in different environmental situations.

The first set of experiments show the performance of the communication cost optimization algorithm. It focused on the prediction accuracy of the unknown parameters that were necessary to select the communication paradigm that produces less network traffic. Network protocol overhead for TCP or IP was not considered.

The other experiments show the behavior of the resource allocation algorithm in isolation as well as in combination of the communication cost optimization. We demonstrate the effective self-organization of nomadic services when they compete for a single server with constant and limited resources. Experiments in a multi-server environment with different constant resource capacities follow. Experiments in a dynamic environment report on the adaptability of the model to changing resources capacities and varying numbers of nomadic services.

The presented results show the behavior for one representative experiment and also mean values and standard derivation from 100 repeated experiments for the average case. The resource utilization of each server was calculated with Eq. (10.3). Even if resources were limited, we did not limit the number of services that could migrate to a server in our simulation experiments. Therefore, in some cases the resource utilization was higher than the actual server capacity. This can be easily avoided by queuing incoming services or specify the server capacity slightly below the actual capacity to avoid this situation.

The following attributes were used to assess the performance of our approach:

- Server resource load—The development of the resource utilization of all servers was measured in resource units over the simulation time.
- Communication paradigm selection (system view)—the absolute number of services using remote communication or migration to a remote server over the simulation time.
- Communication paradigm selection (service view)—the accumulated number of migrations and remote communications per agent over the whole experiment.

The following parameters influenced the simulation experiments model and were used for the performance assessment:

- Number of nomadic services—Experiments have been conducted with different number of static and nomadic services between 100 and 2000.
- Resource consumption—Nomadic services consumed resources during execution at a server. To measure server resource utilization, we assigned a value for the resource consumption during execution to every nomadic service. This value corresponds to real world metrics like memory or processor cycles. All experiments used variable values for the resource consumption that were assigned from a specified interval before each execution.
- Server—Servers were resource providers and accommodated services. Nomadic services could migrate there and allocate resources for a limited amount of time.

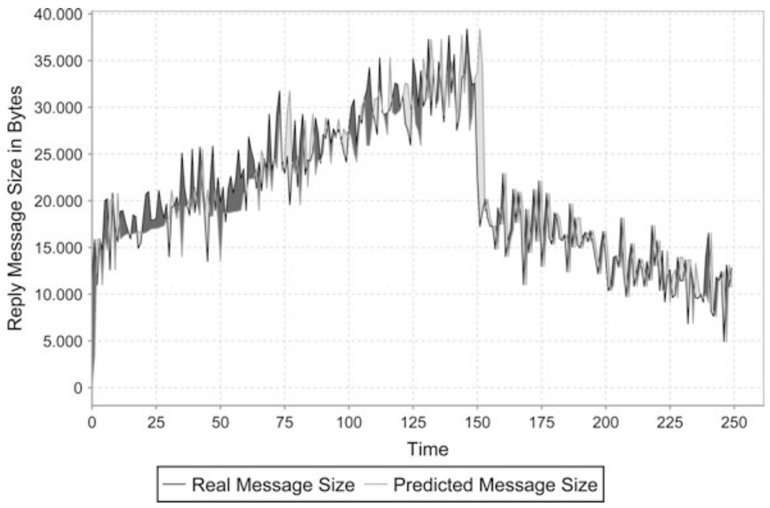


Fig. 10.6 Prediction of a variable reply message size

All servers host a static service which was the communication partner of all nomadic services. The resource consumptions of these static services were not considered as they did not influence the resource allocation itself. The home servers of nomadic services were also not incorporated into the resource allocation process.

- Execution time—The execution time that was needed for the execution of the nomadic service, independent from the execution platform.
- Time between executions—The amount of time that lied between two executions of a nomadic service. The default value was zero, which means that a nomadic service restarts itself immediately after it has finished execution. The default value was used in most experiments and had a major influence on the age and amount of the historical data about servers.

10.5 Simulation Results

10.5.1 Communication Cost

This section reports on results of the communication costs optimization algorithm. The prediction technique for the two unknown environment parameters performed well especially in a dynamic environment as we used a set of predictors instead of only one. In fact, it could keep pace with every static analytical approach. Figure 10.6 shows the development of the reply message size that varies over time and the corresponding predicted values. The reply message was generated by a Gaussian distribution with increasing mean μ over the first 150 time units, beginning

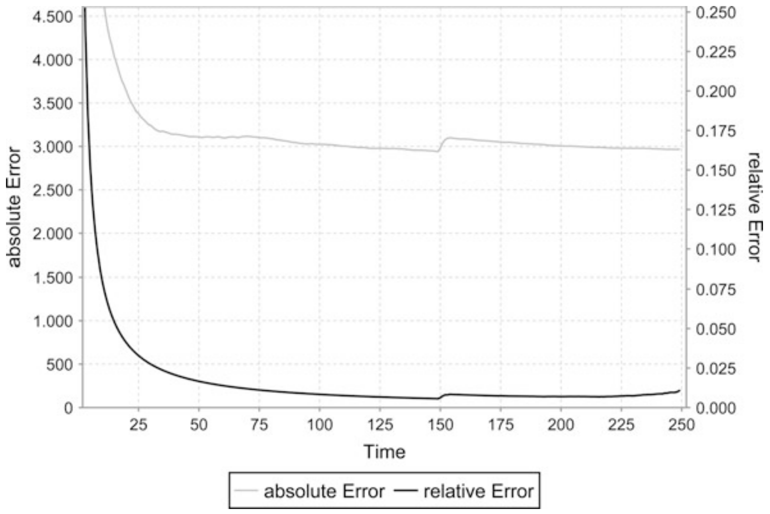


Fig. 10.7 Average absolute and relative error of the reply message size prediction

with 15 kbyte and later dropping. The variance σ^2 of the Gaussian distribution was kept constant at 2450 byte over the whole experiment. We were keen to learn about the quality of simple predictors. All services had a set of 15 randomly assigned predictors. We observed the most predictions were made by mean and $(1,2)$-cycle predictors as they provided the most accurate results for this distribution. Figure 10.7 shows average relative and absolute errors of the predictions. It was expected that the error was close to the standard derivation of the distribution, which can be seen in Fig. 10.7. Application scenarios with communication costs that are not close to the break-even point of both paradigms lead to correct decisions using such simple predictors. However, the quality of the predictors influences the prediction and some application scenarios require specialized predictors which are easy to integrate in the corresponding nomadic services. The prediction mechanism leads to an effective optimization of the network communication costs due to the automatic selection of the beneficial communication paradigm.

10.5.2 Resource Allocation for a Single Server

The following experiments focus on the decentralized, self-organizing resource allocation algorithm. All nomadic services opt for a migration to the remote server and allocate its resources if they expected free resources to be available. Communication costs were not considered in these experiments. The first experiment shows results in a single server environment with a limited constant resource capacity of $C(l_1) = 1000$ resource units. The number of nomadic services was 100 with one static service s_1^s located on server l_1. The home servers of the nomadic services

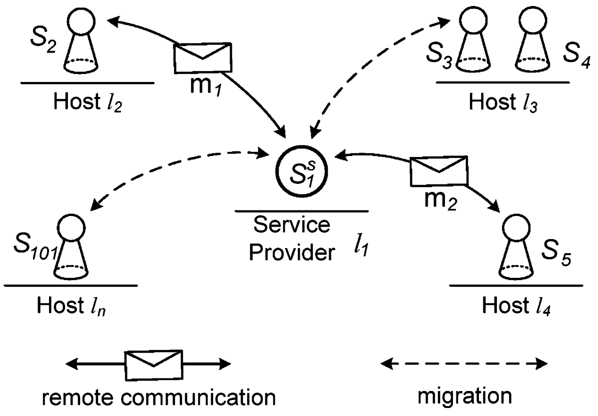


Fig. 10.8 Communication model with multiple nomadic agents and a single remote server with limited capacity

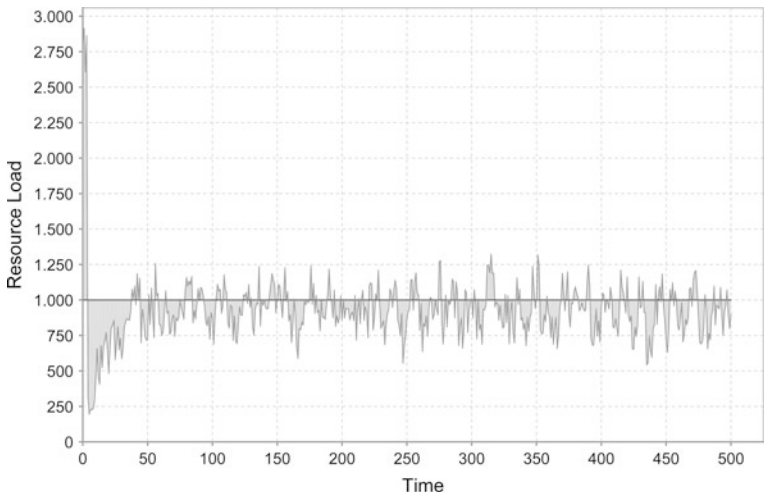


Fig. 10.9 Resource load development of a single server over 500 time units from one representative experiment

were not considered. Figure 10.8 illustrates the experimental setup of the simulation environment. The resource consumptions $U(s_i, t)$ of all nomadic services were randomly assigned from the interval [5, 50] resource units. Execution time of all nomadic services was 1 time unit. Figures 10.9 and 10.10 illustrate the development of the server utilization for one representative experiment and the average resource load development in all experiments, respectively. The provided capacity of the server was utilized to a satisfying level. Not many resources were wasted and only a few situations of overestimation occurred. The available resources in this experiment were very limited. As indicated in Fig. 10.11, only approximately 30 out of 100 nomadic services could be executed simultaneously on the server. The services could self-organize well even if resources were very limited. Figure 10.13 shows the selection of the server from the nomadic services' view. The self-organization

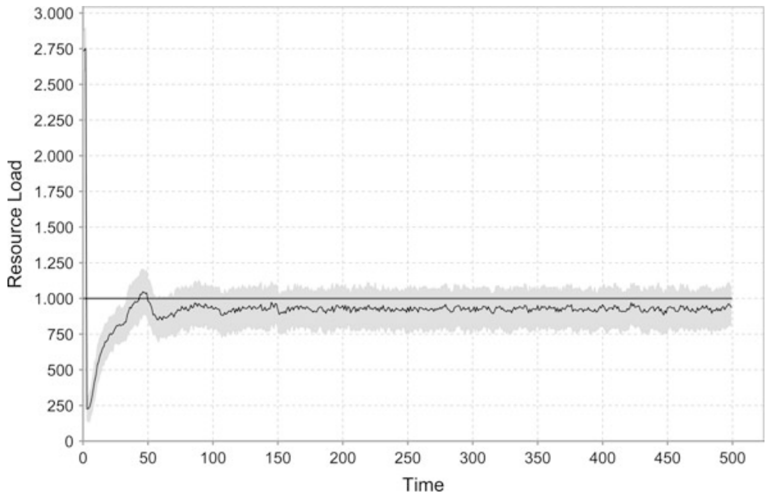


Fig. 10.10 Average resource load development (mean and standard derivation) over 500 time units

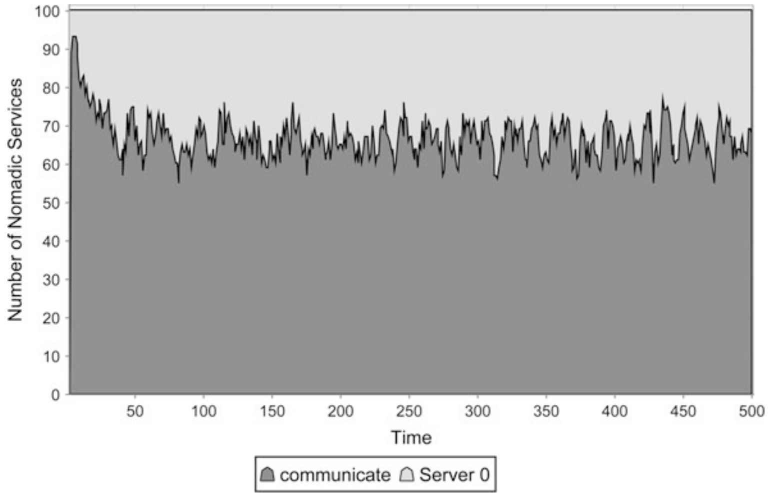


Fig. 10.11 Development of migration vs. communication of the nomadic services over time in one experiment

was fair as there was no group of services that always migrate to the server while others had no chance as to use remote communication due to no free resources. The utilization chart in Fig. 10.10 over 100 repetitions of this experiment shows that the average server utilization was close to the optimal value of 100 per cent with only small variance. The average numbers of migrations versus remote communications is shown in Fig. 10.12 and the accumulated numbers of migration and remote communication are shown in Fig. 10.13. Figure 10.14 shows the active predictor usage

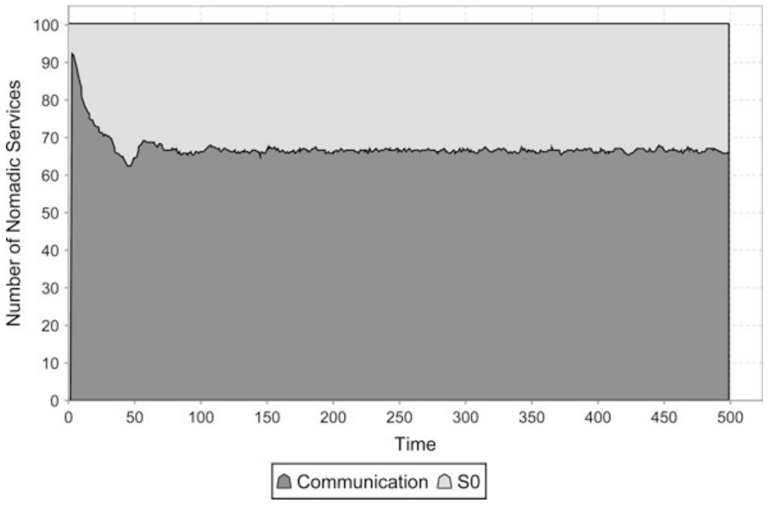


Fig. 10.12 Average development of migration vs. communicating nomadic services over time and 100 experiments

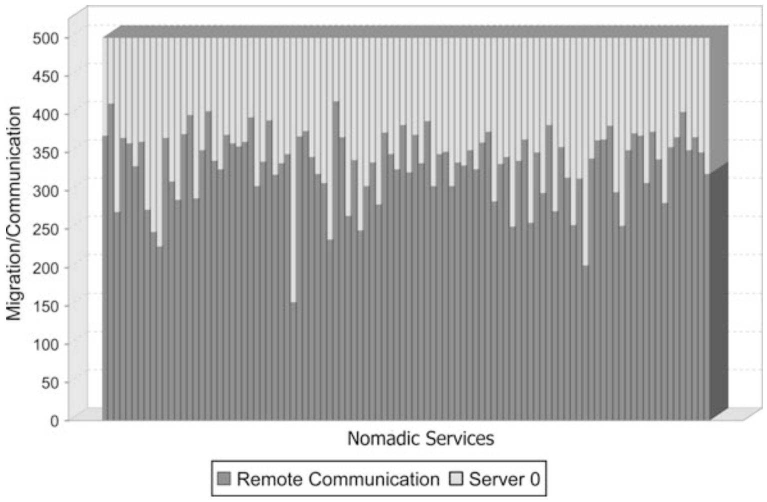


Fig. 10.13 Accumulated number of migration and remote communication per agent over 500 time units

statistics, which show how often a predictor was selected as the active predictor. It is obvious to see the predictors based on mean values were favored while others types of predictors were not considered (mirror predictors) for decision making.

We also conducted experiments with different server capacities ranging from 500 resource units to 5000 resource units. All experiments showed that this simple case with one server can be handled very well if the provided resource capacity allowed

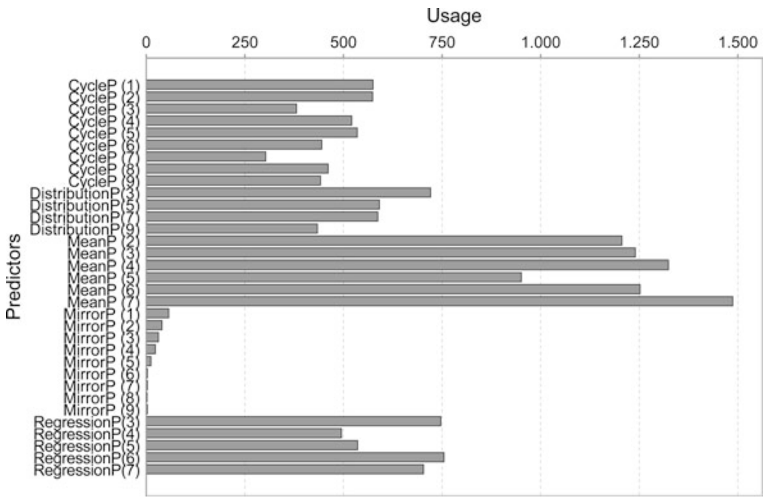


Fig. 10.14 Active predictor usage statistics accumulated over all nomadic services for 500 time units

the simultaneous execution of 20 per cent of nomadic services or more at the server in such a stable environment. Below this threshold, not enough up-to-date historical information was available which led to frequent random migrations and over utilization of the server resources. Decreasing the decay rate of historical data improved this situation in the stable environment but led to problems in more dynamic scenarios.

10.5.3 Distributed Resource Allocation for Multiple Servers

For the experiments in a multiple server environment we used a similar setup as before with the difference that the static service s_1^s was provided by six different service providers. All nomadic services knew those service providers. The only difference between all providers was a different resource capacity. Figure 10.15 illustrates the experimental setup used for the following experiments. Communication costs were not considered in these experiments.

The first set of experiments reports on results in a stable multiple server scenario with constant resource capacities $\mathcal{C}(l_i)$ of 3500, 1350, 2500, 2500, 1500 and 10000 resource units respectively and a number of 800 nomadic services. The resource consumptions $U(s_i, t)$ for all nomadic services were randomly assigned from the interval of $[5, 45]$ resource units. The execution time was assigned from the interval $[1, 10]$ time units. After completing the execution, all nomadic services were restarted immediately. The total capacity of all servers was slightly higher than the average amount of simultaneously requested resources by all nomadic services. The development of the average resource utilization of all six servers is shown in

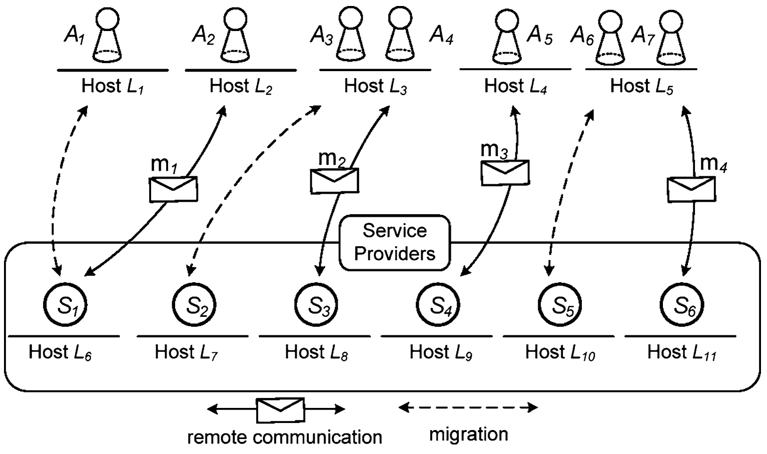


Fig. 10.15 Communication model in a multiple server environment

Fig. 10.16. After around 100 time units all nomadic services self-organized their resource allocation requests in this environment and the resource utilization of each server was stable with only slight variations. It can be observed that servers with low capacity operated at full capacity while other servers had free resources available. This happened because the objective for nomadic services was a resource allocation at a not overloaded server, which was achieved and not to balance the load of all servers equally. Figure 10.17 shows the number of services using the mobile code paradigm versus remote communication. Almost all nomadic services migrated to one of the servers as enough resources were provided by all service providers. In average, nomadic services made 160 resource allocation decisions over the duration of 600 time units during the experiment. Interesting result of the self-organization was the migration frequency to each server. It can be observed that nomadic services migrated to each server according to their capacity as shown in Fig. 10.18. Some might expect that services had "favorite" servers that were used most as they were best predictable based on the most up-to-date historical information. However, nomadic services migrated to all servers according to the resource capacity distribution of the servers, which is represented in the layered structure shown in Fig. 10.18.

10.5.4 Distributed Resource Allocation in a Dynamic Server Environment

This experiment demonstrates the adaptability of our resource allocation algorithm to varying server capacities in a multiple server environment. The servers had different initial resource capacities of 2500, 1500, 2500, 2500, 3000 and 10000 resource units respectively. After an initialization period of 150 time units, the capacities of 4 servers began to change over time. Server 2 (Fig. 10.19(b)) and server

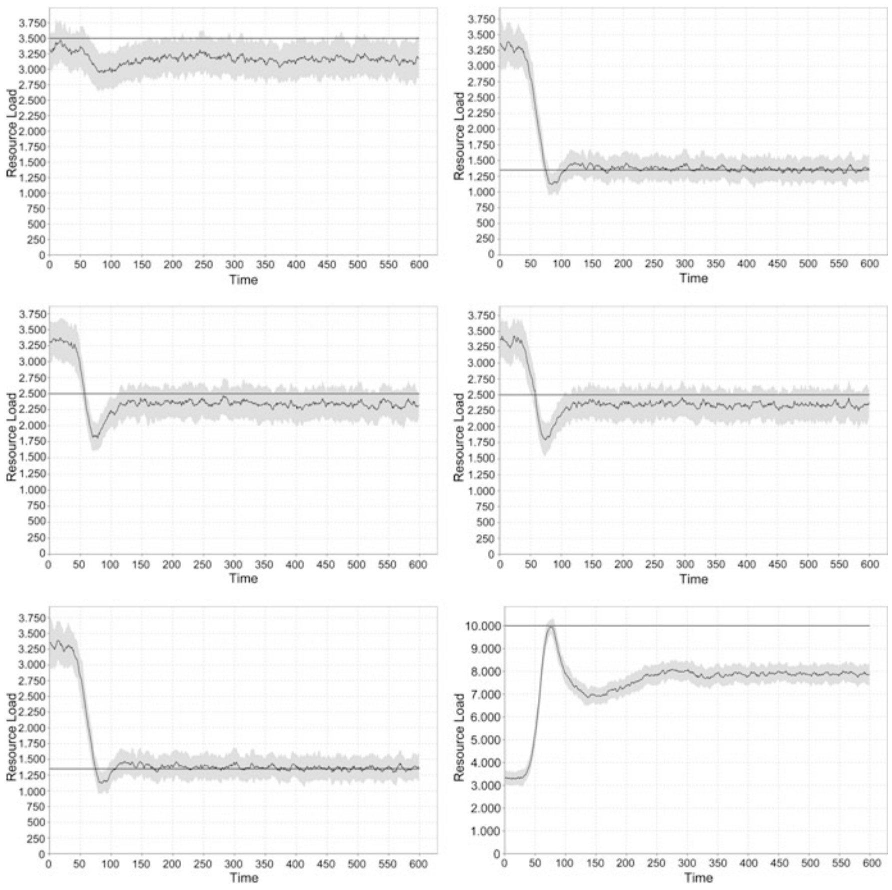


Fig. 10.16 Average resource load development of six servers over 600 time units (mean values and standard derivation)

6 (Fig. 10.19(f)) changed their capacities periodically over time. The capacities of server 3 and server 5 remained constant. Server 1 decreased its available amount of resource to 2000 resource units (Fig. 10.19(a)), server 4 (Fig. 10.19(d)) increased the initial resource capacity to 4000 resource units. A number of 2000 nomadic services competed for the available resources in this experiment. Values for the nomadic services' resource consumptions were unchanged in the interval $[5, 45]$ units. The execution time of each service was in the interval of $[1, 10]$ time units. Nomadic services had a break between two executions randomly assigned from the interval of $[1, 15]$ time units.

It can be seen that this dynamic environment was more challenging for our distributed resource allocation algorithm. The resource allocation mechanism required a constant adaptation to new environmental conditions. Figure 10.19 also shows that the initial adaptation period for all servers increased compared to the previous exper-

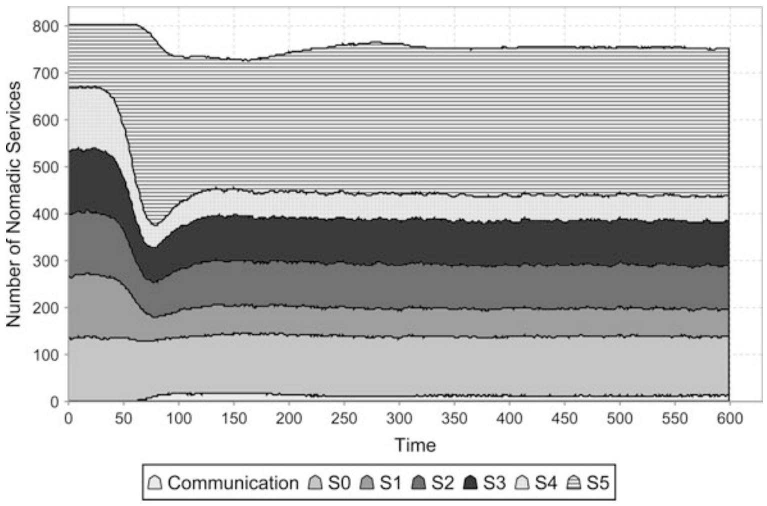


Fig. 10.17 Communication paradigm selection (system view)

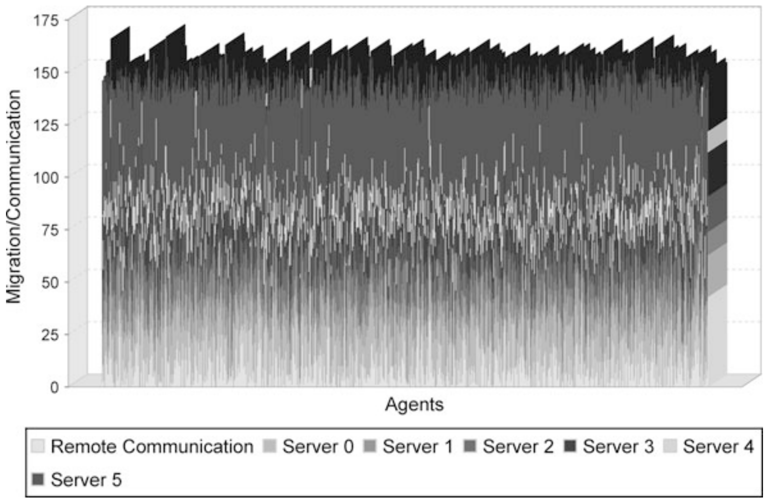


Fig. 10.18 Communication paradigm selection (service view)

iment as this amount of time was required for collecting historical information about other resources. After around 200 time units, almost all nomadic services adapted to the multiple-server environment and utilize the available server resources optimal. Figure 10.20 depicts the number of active services, showing that on average only around 900 of 2000 nomadic services were simultaneously active. It also shows that nomadic services migrated to other servers when their amount of available resources increased. However, some nomadic services used remote communication even free resources were available. The resource load development in Fig. 10.19 shows that

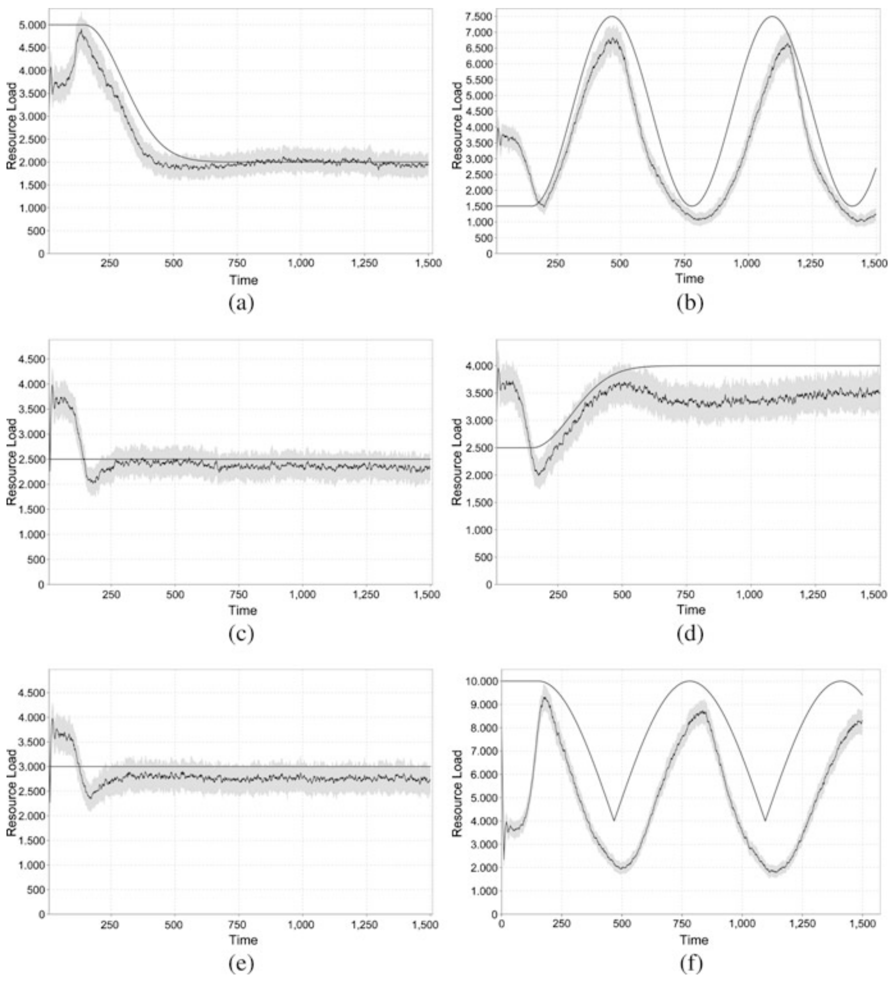


Fig. 10.19 Resource load development in a dynamic environment

the adaptation to different capacities was good and very fast. The adaptation was excellent especially in cases of a periodic and smooth alteration. We ran experiments with sudden changes in the server capacity, which showed worse adaptation as this sudden change can be compared to the initialization period in a new environment. The most difficult case for nomadic services is the exploration of additional resources of already visited servers. This is because they only learn about additional resources when they visit the server the next time. Therefore, each nomadic server should decide between exploration of new resources by random migration to such servers and an exploitation of reliable servers with free resources.

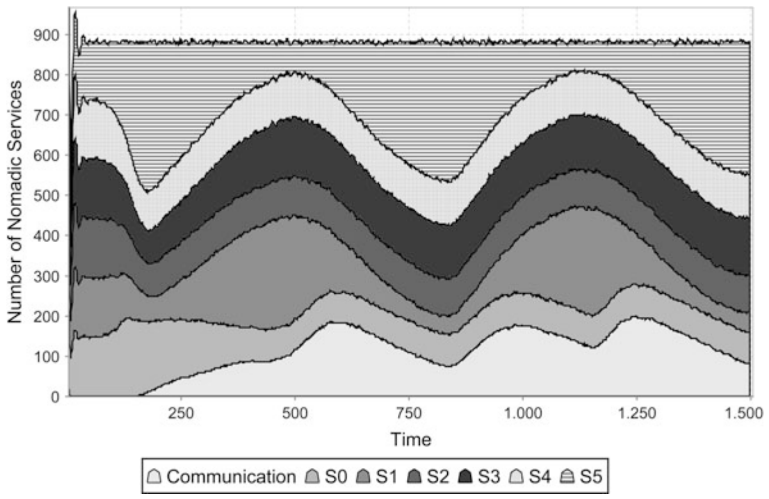


Fig. 10.20 Communication paradigm selection (system view)

10.5.5 Resource Allocation with Communication Cost Optimization

The last experiment reports on results of our decentralized self-organizing resource allocation algorithm in combination with the optimization of the communication costs. The experiment was conducted with two servers of constant capacity of 1000 units and 300 nomadic services. The initialization period was different to resource allocation experiments before as all services used remote communication as the default communication strategy until the communication cost were predictable. The execution time and time between two executions was randomly assigned from the interval [1, 15] units, the resource consumption was between [5, 45] units. The communication cost parameters were randomly initialized as Gaussian distributed random values with different values for μ and σ^2.

Both servers were not fully occupied as the number of services that prefer mobile code was below the provided capacities. Figure 10.21 shows the average number of remote communications versus migrations per services. It can be observed that some nomadic services used remote communication in most cases as this paradigm was the best trade-off between the network load and available server resources. The number of migrations was equally distributed between both servers.

Figure 10.22 shows the comparison between selecting the communication paradigm for all nomadic services and our adaptive strategy. Choosing either remote communication or mobile code was not the best solution as expected. Even if the difference was small, applying our adaptive strategy could reduce the network load. A scenario with a bigger difference between both paradigms would be more beneficial regarding the network traffic reduction with our proposed adaptive strategy.

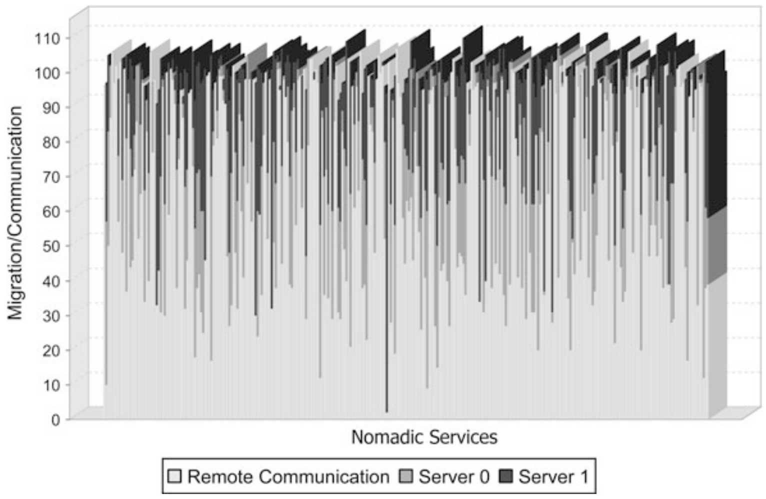
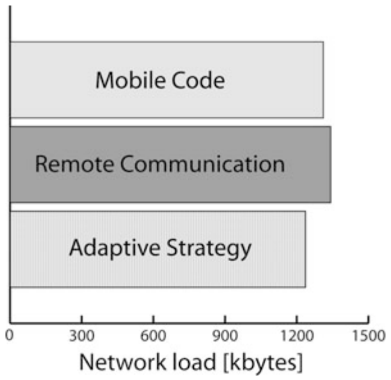


Fig. 10.21 Communication paradigm selection (service perspective)

Fig. 10.22 Comparison between communication paradigms



10.6 Conclusions

This chapter described a distributed, self-organizing resource allocation mechanism for a service grid environment in combination with network communication costs optimization. Nomadic services act purely on local information that is learnt from previous communication acts and resource allocations at remote servers. In order to optimize the communication costs, the communication paradigm that is expected to produce less network traffic is used for the next communication act. If the mobile code paradigm is beneficial in term of the network load, the nomadic services balance the available resource capacities by choosing the server that has the most likely free resources available. The prediction mechanism stores short-term histories of environment parameters and forecast values for the decision making in the next step. This mechanism was inspired by inductive reasoning and bounded rationality principles. All control is distributed among nomadic services in the system. No additional control mechanism or management layer is required. The resource allocation is a purely emergent effect created by the effective competition of nomadic services for the system resources. We demonstrated in various simulation experiments that the proposed system can adapt to changes in the environment by adjusting the services' behavior at run-time.

In our solution, nomadic services adapt the usage of different preassigned strategies but they never change or update them. In biological self-organized systems, interactions between individuals or the behavior of individuals may change slowly over generations. In our system the services' behaviors stay always the same. An update of the nomadic services' behaviors over their lifetime might be beneficial and needs further investigation.

The proposed solution has no central or hierarchical structure and therefore no classical scalability problems if more services and servers join the system. We discovered problems of a different nature. The initialization period increases linear with the number of potential servers for the allocation of resources in the case that the total amount of provided resources is lower than the average amount of requested resources. Imagine an environment with a large number of potential resource providers and all nomadic services can choose between all of them. In the initialization period, services will randomly migrate to any of the available servers. Because of a lack of resource capacity, nomadic services cannot find any server that is not crowded and explore all other servers to gather more information. However, they cannot find any if all nomadic services continue to migrate randomly. Before all services find out that there are no free available resources, historical information is already outdated. In fact, there is no way to gather resource utilization information of many servers. Additionally, the decay rate for historical information must be decreased manually in our current implementation to prevent the too frequently random migration to the potential servers. Another straight forward solution is the limitation of the number of potential servers for resource allocation.

The decay rate for old historical information is in our current implementation a static predefined function. A dynamic environment requires up-to-date information while a stable environment can be predicted with old historical data. Depending on the environmental conditions and execution frequency of a nomadic service the decay rate should be altered to improve for a better resource allocation performance.

We tried different types and different numbers of predictors per nomadic service and learnt that there are no optimal set of predictors for the resource allocation. It is important that services have a selection of different types of predictors that can predict a variety of values above and below the resource capacity to prevent the invalidation of their beliefs.

This kind of distributed resource allocation based on self-organization requires many nomadic services with a numerous and repeated interactions. Interactions in our algorithm are indirect when nomadic services compete for a resource at the same time. Science Grids that usually run few computational very expensive tasks are not suitable for this kind of self-organizing resource allocation algorithm.

References

Allen, G., Angulo, D., Foster, I., Lanfermann, G., Lui, C., Radke, T., Seidel, E., & Shalf, J. (2001). The Cactus Worm: experiments with dynamic resource selection and allocation in a grid environment. *The International Journal of High Performance Computing Applications*, *15*(4), 345–358.

Arthur, W. B. (1994). Inductive reasoning and bounded rationality. *The American Economic Review*, *84*(2), 406–411.

Braun, P., & Rossak, W. R. (2005). *Mobile agents—basic concept, mobility models, and the Tracy toolkit*. San Francisco: Morgan Kaufmann.

Bruneo, D., Scarpa, M., Zaia, A., & Puliafito, A. (2003). Communication paradigms for mobile grid users. In *Proceedings of the 3rd IEEE/ACM international symposium on cluster computing and the grid (CCGrid 2003)*, Tokyo, Japan (p. 669).

Buyya, R. (2002). *Economic-based distributed resource management and scheduling for grid computing*. Melbourne: Monash University.

Buyya, R., Chapin, S., & DiNucci, D. (2000). Architectural models for resource management in the grid. In *Proceedings of the first international workshop on grid computing* (pp. 18–35). Bangalore, India. Berlin: Springer.

Buyya, R., Abramson, D., Giddy, J., & Stockinger, H. (2002). Economic models for resource management and scheduling in grid computing. *Concurrency and Computation*, *13–15*(14), 1507–1542 (Special Issue on Grid Computing Environments).

Challet, D., Marsili, M., & Ottino, G. (2004). Shedding light on El Farol. *Physica. A*, *332*, 469–482.

Challet, D., & Zhang, Y. C. (1997). Emergence of cooperation and organization in an evolutionary game. *Physica A*, *407*(246).

Clearwater, S. H. (1996). *Market-based control. a paradigm for distributed resource allocation*. Singapore: World Scientific.

Fluess, C. (2005). *Capacity planning of mobile agent systems designing efficient Intranet applications*. PhD Thesis. Universiteat Duisburg-Essen (Germany).

Fontana, J. (2004). Service-oriented hype to meet hard realities. *Network World.*

Foster, I., & Kesselman, C. (1997). Globus: a metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications*, *11*(2), 115–129.

Frey, J., Tannenbaum, T., Foster, I., Livny, M., & Tuecke, S. (2002). Condor-G: a computation management agent for multi-institutional grids. *Cluster Computing*, *5*(3), 237–246.

Galstyan, A., Kolar, S., & Lerman, K. (2003). Resource allocation games with changing resource capacities. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, & M. Yokoo (Eds.), *Proceedings of the second international joint conference on autonomous agents and multi-agent systems*, 2003, Melbourne, Australia, 14–18 July 2003 (pp. 145–152). New York: ACM Press.

Grosu, D., Chronopoulos, A. T., & Leung, M.-Y. (2002). Load balancing in distributed systems: an approach using cooperative games. In *Proceedings of the international parallel and distributed processing symposium*, Ft. Lauderdale, FL, USA (p. 196). Washington: IEEE.

Mainland, G., Parkes, D. C., & Welsh, M. (2005). Decentralized adaptive resource allocation for sensor networks. In *Proceedings of the 2nd USENIX symposium on network systems design and implementation (NSDI '05)*, Boston, MA.

Manvi, S. S., Birje, M. N., & Prasad, B. (2005). An agent-based resource allocation model for computational grids. *Multiagent and Grid Systems*, *1*(1), 17–27.

OASIS (2006). Software Oriented Architecture (SOA) reference model. Available at http://www.oasis-open.org/committees/download.php/19361/soa-rm-cs.pdf. 15.09.2006.

Outtagarts, A., Kadoch, M., & Soulhi, S. (1999). Client-server and mobile agent: performances comparative study in the management of MIBs. In *Proceedings of the first international workshop on mobile agents for telecommunication applications (MATA 1999)*, Ottawa (Canada), October 1999. Singapore: World Scientific.

Picco, G. P. (1998). *Understanding, evaluating, formalizing, and exploiting code mobility*. Politecnico di Torino (Italy).

Puliafito, A., Riccobene, S., & Scarpa, M. (2001). Which paradigm should I use? An analytical comparison of the client-server, remote evaluation and mobile agent paradigms. *Concurrency and Computation*, *13*(1), 71–94.

Samaras, G., Dikaiakos, M. D., Spyrou, C., & Liverdos, A. (1999). Mobile agent platforms for Web-databases: a qualitative and quantitative assessment. In *Proceedings of the first international symposium on agent systems and applications (ASA'99)/Third international symposium on mobile agents (MA'99)*, Palm Springs (USA), October 1999. Washington: IEEE.

Strasser, M., & Schwehm, M. (1997). A performance model for mobile agent systems. In *Proceedings of the international conference on parallel and distributed processing techniques and applications (PDPTA'97)*, Las Vegas (USA), 30 June 1997 (pp. 1132–1140). Las Vegas: CSREA Press.

Theilmann, W., & Rothermel, K. (2000). Dynamic distance maps of the Internet. In *Proceedings of the 2000 IEEE INFOCOM conference*, Tel Aviv, Israel, March 2000 (pp. 275–284). Washington: IEEE.

Vigna, G. (2004). Mobile agents: ten reasons for failure. In *Proceedings of the 2004 IEEE international conference on mobile data management (MDM'04)* (pp. 298–299). Los Alamitos: IEEE.

Waldrop, M. M. (1992). *Complexity: the emerging science at the edge of order and chaos* (1st ed.). New York: Simon and Schuster.

Wolski, R., Plank, J. S., Brevik, J., & Bryan, T. (2001). Analyzing market-based resource allocation strategies for the computational grid. *The International Journal of High Performance Computing Applications*, *15*, 258–281.

Wooldridge, M. (2002). *An introduction to multi-agent systems*. Chichester: Wiley.

Chapter 11
Immune System Support for Scheduling

Young Choon Lee and Albert Y. Zomaya

11.1 Introduction

Haven't there been enough approaches to scheduling problems? In terms of variety the answer might be 'yes'. However, the answer is not as straightforward in terms of effectiveness. In many scheduling problems, it is highly improbable, if not impossible, to obtain optimal schedules within a reasonable amount of time in spite of adopting a wide range of approaches, including evolutionary computation (EC), artificial neural networks (ANN), fuzzy systems (FS), simulated annealing (SA) and Tabu search (TS).

In recent years attention has been drawn to another biologically-inspired computing paradigm called artificial immune systems (AIS). An AIS abstracts and models the principles and processes of the biological immune system in order to effectively tackle challenging problems in dynamic environments. Major AIS models include negative selection, clonal selection, immune networks and more recently danger theory (Garrett 2005). There are some similarities between these principles and processes in the immune system and those found in other nature-inspired computing approaches, EC and ANN in particular. However, there are also substantial differences. In particular, adaptive cloning and mutation processes make AIS distinctive and useful.

Two fundamental immune activities are the recognition and the elimination of foreign agents irrespective of their previous exposure to the host. One can easily notice the inherent applicability of these robust and adaptive immune functionalities to many hard problems, such as pattern recognition, anomaly and fault detection, and machine learning. Although not as directly applicable, scheduling can also benefit

Y.C. Lee (✉) · A.Y. Zomaya
School of Information Technologies, The University of Sydney, Sydney, NSW, Australia
e-mail: yclee@it.usyd.edu.au

A.Y. Zomaya
e-mail: zomaya@it.usyd.edu.au

from AIS as illustrated in recent scheduling strategies (Swiecicka et al. 2006; King et al. 2001; Costa et al. 2002; Coello Coello et al. 2003; Engin and Doyen 2004).

Due to the NP-hard nature of scheduling problems, in most cases heuristic approaches have been extensively studied. In general, these heuristics are susceptible to dynamic environments. This motivates the search for more robust alternatives, and some effective strategies have already been proposed in the literature. Since environmental flux is ubiquitous in natural environments, many of the proposed approaches are inspired by nature; hence the name nature-inspired computing. AIS as a new breed of this soft computing paradigm has been showing potential in scheduling as well as many other application areas.

It is noted that the AIS community is actively striving to broaden the application areas of AIS. In addition, an increasing amount of research attempting to better exploit the rich set of immune components, principles and processes have been made. Some outcomes from theses attempts (Hajela and Yoo 1999; Dasgupta et al. 1999; Krishna Kumar et al. 1995; Feng and Feng 2004) have been integrated with other approaches, such as genetic algorithms (GA) and FS to further improve their performance.

11.2 Scheduling Problem

Scheduling is the process of allocating a set of resources to tasks or jobs in order to achieve certain performance objectives and satisfying certain constraints. These scheduling objectives include minimizing schedule length (SL), also called *makespan*, minimizing response time and maximizing resource utilization. Temporal and resource constraints are two primary conditions imposed on scheduling. For example, there may be a specific execution order that the tasks must follow, and a resource can be used for no greater than one task at a time.

Due to the importance of the scheduling problem it has been extensively studied in many disciplines, such as operations research, manufacturing, computer science and economics. There are a number of different scheduling problems including multiprocessor scheduling, job shop scheduling and flow shop scheduling. Although these various scheduling problems are taken into consideration throughout the chapter the main focus is on the multiprocessor scheduling problem. Hereafter, scheduling denotes multiprocessor scheduling unless stated otherwise.

Broadly, scheduling problems are classified as static or dynamic (Grama et al. 2003). They are distinguished by the time at which the scheduling decisions are made. With static scheduling, the necessary information, such as the processing requirements of tasks and the processing capacities of resources are identified and schedules are determined *a priori*. Conversely, scheduling information in a dynamic scheduling scheme is obtained on-the-fly. Dynamic scheduling attempts to reduce scheduling overheads as well as job completion time. Both of these scheduling models have been studied widely and intensively. The two most common scheduling strategies are heuristics and sub-optimal approximation techniques, such as randomized search methods.

11.2.1 Multiprocessor Scheduling

Ideally, it would be expected that the execution time of a job, consisting of a set of tasks, in a computer system with m processors would be m times faster than a single processor computer system. However, this is not quite true in practice, because generating an optimal schedule of the partitioned tasks is NP-hard; and the partitioned tasks of a job may not be completely independent (Grama et al. 2003). There may be additional challenges, such as resource and task heterogeneity; and the uncertainty of resource availability and capability, which further complicate scheduling.

Scheduling approaches include list scheduling, approximation and random guided search. Because the scheduling problem is NP-complete, algorithms that generate near optimal schedules have a high time complexity. Conversely, for any upper limit on time complexity, the quality of the schedule in general will also be limited. Together, this suggests a trade-off between performance and time complexity over the class of all scheduling problems.

11.2.2 Scheduling Heuristics

Heuristics are popular because in most cases they deliver good solutions in less than polynomial time. A primary intention of heuristics is to find a solution as fast as possible potentially at the cost of quality. Heuristics are characterized by their essentially deterministic operation: the choice of solutions to a scheduling problem is not stochastic.

List scheduling heuristics are the single dominant heuristic model. This is because empirically, list scheduling algorithms tend to produce competitive solutions with lower time complexity compared to algorithms in the other categories (Kwok and Ahmad 1998). The two phases commonly found in list scheduling are task prioritization and processor selection. The tasks in the task graph are first assigned priorities using some prioritization method and are kept in a list in decreasing order of priorities. In the processor selection phase, each task is assigned to an appropriate processor based on the processor selection policy of the scheduling algorithm. List scheduling heuristics can be further improved by incorporating other techniques, such as task insertion and task duplication. With the task insertion method a task is allocated to the earliest start time slot of the processor as long as it does not violate the precedence constraints. Here, the allocated slot might be between two already assigned tasks. The rationale behind duplication based scheduling algorithms is to increase the processor utilization and to decrease the communication overhead by duplicating tasks on different processors.

An alternative scheduling heuristic is clustering, where each task is initially regarded as a cluster. If the merging of two clusters produces a shorter finish time they are merged. This process repeats until no further merging can happen. After the clustering is done the tasks in each cluster are assigned to the same processor in order to reduce the communication overhead.

11.2.3 Randomized Search Techniques

Many available randomized search algorithms do not generate completely random schedules—they utilize information from previous search paths—but they do make decisions stochastically which shape the search paths. Randomized search techniques can usually be interpreted as a biased random walk over the state space of all possible schedules. Typical examples are GA, SA and TS. Despite their high time complexity they are robust and adaptive, because they do not make assumptions about the structure of the problem.

GAs are inspired by the process of biological evolution, where natural selection operates on a population of phenotypes, resulting in differential reproduction that transfers the essential structures of the most successful genotypes to the subsequent generation. The main steps involved in a GA are: (1) An initial population is randomly generated. (2) The entire population is evaluated by a fitness function and the best solutions are selected based on their fitness values. (3) The selected solutions are then mutated and/or recombined, which gives a new generation of individuals. (4) Steps 2 and 3 are repeated until a termination condition (e.g., a fixed number of generations has elapsed with no improvements on the best solution) has been reached. The mutation operator is the source of the randomized search decisions. The performance of a GA is sensitive to the value of its control parameters, such as population size, crossover frequency and mutation frequency.

Like GAs, SA repeats a series of processes until the termination criteria is satisfied. At each iteration step, the algorithm randomly chooses a neighbor of the current state and always moves to the neighbor if it is an improvement on the value of the current state. If it is worse than the current state, the algorithm may still move to the new state with some probability. Initially this probability is high, allowing free movement across the state space, but over time the probability diminishes according to a "cooling" schedule. The performance of SA is affected significantly by the neighbor selection method and the cooling schedule. Unlike GAs, SA has been shown to converge to the optimal solution given infinite time.

TS searches neighbors of the current solution like SA at each step. Because it prevents cycles in search paths, it may produce an approximation to the optimal solution more efficiently. TS works by forbidding moves to states that have been visited within a fixed number of previous steps.

11.3 The Immune System

The immune system is a biological defence mechanism designed to protect an organism primarily from microbes, such as bacteria, archaea, fungi, protists and viruses. An allied force of cells, tissues and organs battles these foreign invaders. Although at first glance the immune system easily performs its core functions of detecting and killing infections, it is made possible only by the careful coordination of various immune entities incorporated with immune principles and processes.

Some examples of these immune functionalities are pattern recognition, memory, learning, negative selection and clonal selection.

At the highest level, two defence lines (the innate and the adaptive immune systems) are embodied. The core forces of both systems are different types of white blood cells.

11.3.1 Innate Immune System

The innate or non-specific immune system is the first line of defence that uniformly combats invaders directly and immediately with chemical substances and specific types of white blood cells. As its name implies this innate immunity already exists at the time of birth and is triggered to respond against known invading entities. Typical examples of the former chemical substances are skin, saliva, tears, sweat and gastric acid. The four cell types in the latter are neutrophil granulocytes, macrophages, eosinophil granulocytes and basophil granulocytes. In addition to physical barriers and phagocytic cells mentioned above anti-microbial proteins, such as complement proteins, acute phase proteins and interferons, play an important role to further protect the host.

While the adaptive immune system can respond to a diverse set of attackers (antigens) the innate immune system is limited to recognize several common structures present in many different microorganisms. These innately recognizable structures are called pathogen-associated molecular patterns. Note that, since there are relatively a small number of these patterns and they can be easily recognized by pattern-recognition receptors of most body defence cells, a response against foreign invaders is immediate.

11.3.2 Adaptive Immune System

During the lifetime of an organism it encounters numerous different antigens that the innate immune system is not able handle effectively. The adaptive or specific immune system comes into play in such a circumstance. Note that some of those antigens might have been previously exposed to the organism. In case of a subsequent invasion of the same antigen the adaptive immunity initiates a swifter and more effective response compared to the response to the first exposure of the invader. This is enabled by the memory function of the adaptive immune system. In addition to this memory property it functions with a series of sophisticated features, such as differentiation and self-organization. The adaptive immune system is often referred to simply as the immune system.

The two key components in the adaptive immune system are B lymphocytes (B-cells) and T lymphocytes (T-cells) of white blood cells that are produced by stem cells in the bone marrow. They are named after the lymphoid organs in which they

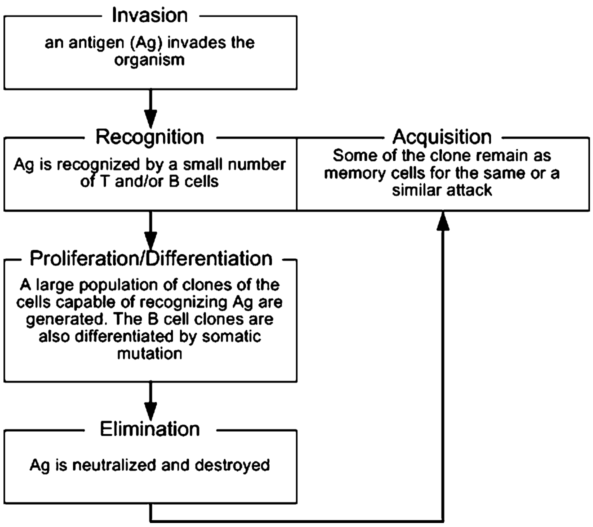

Invasion
an antigen (Ag) invades the organism
Recognition
Ag is recognized by a small number of T and/or B cells
Acquisition
Some of the clone remain as memory cells for the same or a similar attack
Proliferation/Differentiation
A large population of clones of the cells capable of recognizing Ag are generated. The B cell clones are also differentiated by somatic mutation
Elimination
Ag is neutralized and destroyed

Fig. 11.1 The primary steps involved in the adaptive immune system

are produced and developed, namely in the **B**one marrow and the **T**hymus. Each of these two cell types plays a primary role in one of the two defence mechanisms (the humoral immunity and the cellular immunity).

The humoral immunity is overseen by B-cells. More specifically, immunoglobulins or antibodies produced by plasma cells matured from B-cells are ones that actually take actions. The humoral immune response takes place against invading microbes by antibody-antigen binding within a matter of minutes. As its name implies, the humoral immunity responses are activated in the body liquids, e.g., blood against bacteria and viruses.

T-cells, matured in the thymus as the primary mediator, take charge of the cellular immunity that responds to other cells that are infected by viruses.

Now the question is how the adaptive immune system can respond against a virtually unlimited and diverse set of antigens. A sequence of phases for battling against these immunological enemies shown in Fig. 11.1 can answer this question.

11.3.3 Applicable Potentials of the Immune System: A Computational Science Perspective

The immune system consists of a complex, sophisticated and effective set of properties, principles and processes. There are a number of important immune features that should be noted, such as pattern recognition, adaptability, learning and memory, to name a few. The list of key immune characteristics can be categorized into three as follows:

Self-organization and Self-maintenance: the immune agents, primarily cells are autonomous in nature. Their activities, such as reaction to microbial attack, proliferation, differentiation and memory take place without any central guidance or control. In essence, the immune system attempts to maintain superior cells while eliminating inferior and foreign cells. The key immune principle that enables and facilitates this feature is clonal selection with affinity maturation.

Cooperativeness: there are two major sets of allies in the immune system. They are the innate and the adaptive immune systems, and T and B cells (or humoral and cellular immunity) in the adaptive immune system. While the latter allies are well known the former are rarely recognized with fairly recent discoveries, such as dendritic cells and toll-like receptors as a critical adjuvant in the activation of adaptive immunity. The cooperation and coordination of the two parties in each coalition exist for effective immune responses that no agent could achieve in isolation.

Robustness: the immune system maintains a diverse set of lymphocytes distributed all over the body. These lymphocytes are constantly updated and/or upgraded. In this way immune responses can very quickly and effectively take place regardless of where the attack occurs and what the attacking agent's pattern is. In addition, the immune system intensifies some lymphocytes, upon the recognition of the attacker, in order to win the battle as soon and easily as possible.

Note that the immune system does not rely on just one or two of these features. Rather, it works as an integrated system of all these powerful features. In other words, these characteristics should be carefully orchestrated when applied to computational problems.

11.4 Artificial Immune Systems

As the biological immune system is an effective defence against constant threats in dynamically changing environments it has inspired models in an increasing number of areas, such as computer and network security, data mining and pattern recognition. An artificial immune system can be defined as a real-world problem solving methodology designed as a result of abstraction and modeling of immune features. For alternative definitions see de Castro and Timmis (2002).

To date the majority of AIS are based primarily upon the adaptive immune system mainly because it can deal with unforeseen circumstances. The two commonly modeled immune entities are antibodies and antigens, since they are the key players in the adaptive immune system. In conjunction with these immune entities a few immunological theories, such as negative/positive selection, clonal selection and immune networks have been actively modeled. Note that these are merely some popularly modeled instances among a rich set of immune characteristics.

A recent analytical study (Garrett 2005) discusses how AIS differ from other biologically-inspired computing approaches, such as EC, ANN and FS. With its comparison study it evaluates the usefulness of AIS based on distinctiveness and effectiveness, and concludes AIS have promising potential and have successfully demonstrated their applicability.

11.4.1 Negative Selection

The immune system's competence for recognizing foreign intruders is one very appealing function due to its direct applicability in many areas, such as anomaly detection and pattern recognition. Although there is not complete consensus on how this recognition—so-called *self-nonself discrimination*—is accomplished, a single dominant mechanism of this immune activity is the negative selection principle.

It is crucial that the immune system does not become aggressive against its host. A general view of how the immune system distinguishes between self and nonself antigens is that there are only nonself recognizable T and B cells, as a result of the elimination of self-specific lymphocytes, circulating the body of a host. This selection takes place in the primary lymphoid organs of these lymphocytes before they are released.

Based upon this biological negative selection a well-known artificial negative selection scheme was proposed in Forrest et al. (1994). It modeled the negative selection of T cells in order to be applied for detecting changes in computer systems. The three principles of the algorithm presented in Forrest et al. (1994) take advantage of having a diverse set of change detectors that lead to the improvement of system robustness. Both self and nonself entities are represented as bit strings; namely self strings and nonself strings.

The negative selection algorithm consists of two major phases, detector generation and anomaly monitoring. In the detector generation phase, a set of detectors are randomly generated and matched against a predefined set of self strings. The detectors are filtered out based on how closely they match the self strings. Specifically, a pre-selected integer, r, is used as a censoring parameter that represents the number of contiguous matching bits. For example, if two strings have r or more contiguously identical bits they are called a closely matched pair. The detector of this pair then gets discarded due to its self-detecting ability. The anomaly monitoring phase continually performs matching between the detectors and the self strings in order to detect any changes in the self ones.

This flagship negative selection algorithm has spawned a series of subsequent studies mainly in order to streamline the detector generation strategy and to widen its applicability.

There have been a number of improvements suggested to the random detector generation process. They include linear and greedy algorithms (D'haeseleer et al. 1996), a deterministic analysis of negative selection (Wierzchon 2000), the permutation mask approach (Hofmeyr and Forrest 2000) and a detector generation scheme by random means with clonal selection support (Ayara et al. 2002).

Most applications of negative selection consider two opposing parties—good and evil. Here, the sole objective is to protect good from evil as in nearly every story and movie. Typical application areas include anomaly detection (Gonzalez et al. 2002; Gonzales and Cannady 2004), fault detection and diagnosis (Dasgupta et al. 2004; Gao et al. 2004), intrusion detection (Kim and Bentley 2001; Stibor et al. 2005), and more recently negative database (Esponda et al. 2005).

11.4.2 Danger Theory

For many years self-nonself discrimination has been widely regarded to be the most compelling mechanism for immune responses. Traditionally, it is believed that an immune response is initiated upon the recognition of nonself antigens as in negative selection. However, this viewpoint was questioned especially by Matzinger proposing a new immunological model, the danger theory (Matzinger 2002). This model is grounded on the discrimination between dangerous and non-dangerous instead of the conventional self-nonself classification. Burgess (1998) identified problems with the traditional model by posing the following questions:

Why does self changing in the course of numerous lifetime events, such as puberty, pregnancy and aging not trigger an immune response?

How are vaccines composed of inert foreign proteins harmoniously incorporated with the immune system?

What prevents the immune system from responding against those believed to be mutated proteins (e.g., tumors) or against autoreactive lymphocytes that might cause autoimmune diseases, such as Hashimoto's thyroiditis, Graves' disease and Rheumatoid arthritis?

These questions were answered by the danger model that the immune system responds not against any nonself, but against some that are recognized as a danger to the host. Then, how do we define danger? In the biological immune system a cell can die in one of two ways: apoptosis or necrosis. The former is a programmed death as a result of homeostatic regulation, whereas the latter is a non-programmed death due to a number of different causes, such as injury, infection and inflammation. This cellular necrosis is believed to signal danger.

As well as the negative selection principle, danger theory has been applied to several similar areas (Burgess 1998; Aickelin and Cayzer 2002; Aickelin et al. 2003). The danger model is, however, still in its early stage of development. Aickelin and Cayzer (2002) and Aickelin et al. (2003) have introduced and described rather abstract models of its applications in anomaly detection and data mining; and intrusion detection respectively.

An appealing strength of the danger theory for intrusion detection systems is its scalability. The most fundamental requirement for intrusion detection systems is to define and model normal activities. Although negative selection is capable of this process based solely on the information of normal behaviors it may suffer from a serious problem when the system is large and dynamic. In the meantime, the danger model is highly scalable since it only deals with activities known to be dangerous. Note that, in this regard dangerous behaviors have to be defined. This gives rise to an adaptability issue.

11.4.3 Clonal Selection

As mentioned earlier, one of the most powerful features of the immune system is its adaptability. The clonal selection principle (Burnet 1959) in the adaptive immune

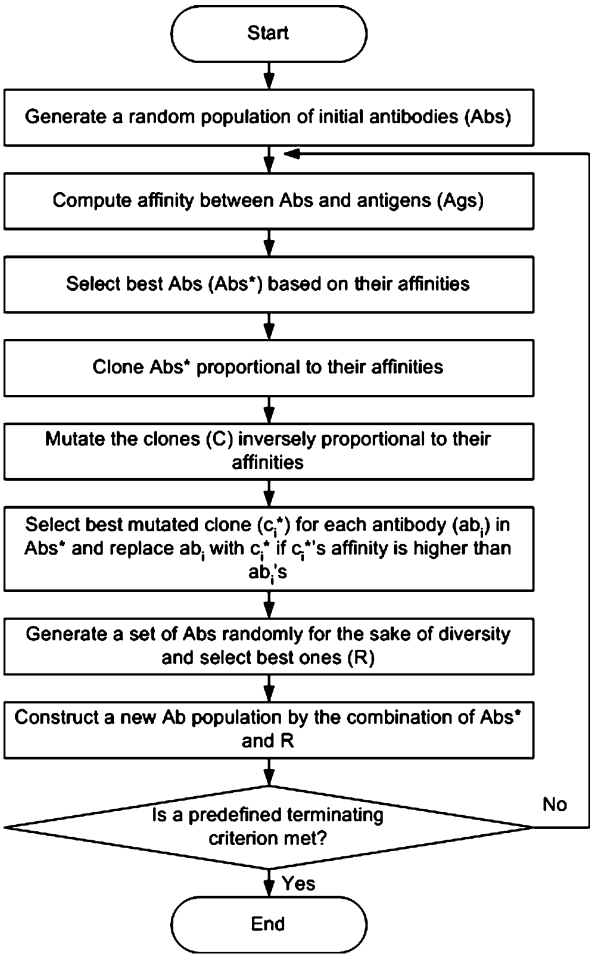

Start
Generate a random population of initial antibodies (Abs)
Compute affinity between Abs and antigens (Ags)
Select best Abs (Abs*) based on their affinities
Clone Abs* proportional to their affinities
Mutate the clones (C) inversely proportional to their affinities
Select best mutated clone (c_i*) for each antibody (ab_i) in Abs* and replace ab_i with c_i* if c_i*'s affinity is higher than ab_i's
Generate a set of Abs randomly for the sake of diversity and select best ones (R)
Construct a new Ab population by the combination of Abs* and R
Is a predefined terminating criterion met?
No
Yes
End

Fig. 11.2 The CLONALG algorithm

system plays an important role to this property. Although clonal selection occurs on both T and B cells, the focus in the field of AIS is often aimed at B cells. This is primarily due to the fact that B cell clonal selection involves mutation that further enhances the adaptability of B cells. Hereafter, clonal selection simply refers to that of B cells.

The rationale behind the clonal selection theory is that superior B cells are preserved with a minor degree of mutation and become prevalent, and inferior ones are mutated at a high rate hoping to be improved, and become rare. More specifically, when a foreign intruder (antigen) attacks the host, B cells matching the antigen will be cloned (i.e., clonal expansion) and mutated (i.e., affinity maturation) at rates directly proportional to and inversely proportional to the degree of the match (or affinity), respectively.

The clonal selection principle is the most popular model applied to AIS. In the past couple of decades a growing number of AIS (de Castro and Von Zuben 2002;

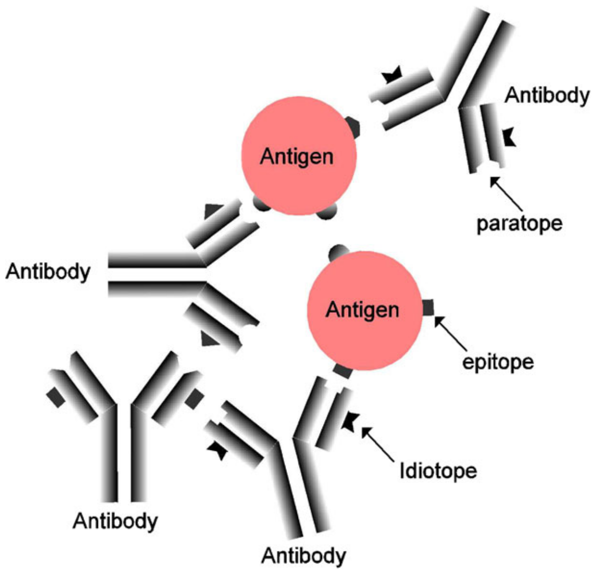


Fig. 11.3 The immune network theory

Cutello and Nicosia 2002) based on the clonal selection theory have been developed. Some example applications can be found in areas of pattern recognition, scheduling and graph coloring.

Among many existing AIS using this principle, CLONALG presented in de Castro and Von Zuben (2002) is most well known largely due to its algorithmic simplicity. The algorithm CLONALG is described in Fig. 11.2.

11.4.4 Artificial Immune Networks

Another distinct immunological hypothesis attracting serious attention is Jerne's immune network theory (Jerne 1974). As its name implies the immune system is a network of self entities or antibodies that interact (i.e., stimulate and suppress) with each other as well as with foreign entities or antigens. As shown in Fig. 11.3 *paratopes* of an antibody bind to not only *epitopes* of an antigen, but also to *idiotopes* of other antibodies. Idiotopes—parts of antibodies—are the distinct feature introduced in the immune network theory. The existence of idiotopes contributes to establishing a network of antibodies; hence the name immune network theory or idiotypic network hypothesis. This leads to the assumption that the immune system operates dynamically even in the absence of antigen.

Like the clonal selection principle, superior antibodies proliferate and inferior antibodies are suppressed. That is, antibody clones with strong antigen and/or antibody recognition capability remain at high population levels whereas less effective antibody clones are kept at lower population levels. This way, the immune system maintains the good quality antibodies resulting in effective immune responses. Once again the superiority measure used in the immune network theory is affinity, i.e., how strong an antibody-antigen or antibody-antibody match is.

Jerne's immune network theory has initially populated several serious theoretical models (Farmer et al. 1986; Varela and Coutinho 1991) developed by immunologists. These models served as inspiration for computational network models for practical use. Two recent and well-known computational immune network models are RAIN (Timmis and Neal 2000) and AINET (de Castro and Von Zuben 2001). They both work similarly to the clonal selection principle, with one significant difference: the reactivity between antibodies. This is the most fundamental feature of the immune network theory. More specifically, stimulatory and suppressive interactions between antibodies are incorporated in the process of regulating antibody clones based on affinities between them.

11.5 Abstraction and Modeling of the Immune System for Scheduling

A rich set of entities, principles and processes in the immune system has shown great potential in many problem domains. It should be noted that successful exploitation of these immune features is heavily based on careful abstraction and modeling as in most, if not all, other nature inspired approaches. Some examples of this abstraction and modeling in other approaches include artificial neurons in artificial neural networks, pheromones in ant colony optimization algorithms, crossover and mutation processes in genetic algorithms.

Although different scheduling problem instances exhibit their own problem specific properties they have a series of characteristics in common. For example, each task has processing amount and each resource is associated with processing capacity. The main objective of the scheduling problem, regardless of different problem instances, is to find optimal task/resource allocation combinations taking into account certain performance metrics.

It should be noticed that most AIS consist of a common set of steps: (1) initial population generation, (2) antibody-antigen binding, (3) population refinement, and (4) learning and adaptation. The last three steps iterate until a satisfactory solution is found.

11.5.1 Immune Entities

Both the scheduling problem and the immune system have two main entities being tasks or jobs and resources in the former, and antibodies and antigens in the latter. Unlike the obvious mapping between the two immune entities and components in many problem domains, such as anomaly detection and pattern recognition, such mapping in scheduling is not obvious. Typically, most scheduling schemes based on the immune system use antibodies alone to represent schedules. However, other immune entities, such as antigens and lymphoid organs may also come into play in many scheduling problem instances.

11.5.1.1 Antibodies

As stated above, the most obvious immune component that can be modeled is the antibody. A typical representation of an antibody is a string of resource identifiers or simply resources with each of which a task is associated. Here, the arrangement of the resources determines the quality of schedule.

11.5.1.2 Lymphoid Organs

As in the biological immune system the production of antibodies, or more precisely lymphocytes, in an AIS can be modeled using abstract lymphoid organs. The two primary lymphoid organs are the bone marrow and the thymus. They can be abstracted and modeled for initial population generation and refinement, respectively.

In most AIS, the initial population is generated randomly. However, the use of a good quality initial population might result in faster convergence with a better solution compared to that delivered using a randomly generated initial population.

The immunological crossover process (Bersini 2002) with a simple, yet efficient heuristic can be an option to generate decent quality initial populations. More specifically, a heuristic is used to generate several solutions that are expected to be at least better than those generated randomly. The immunological crossover process further populates more initial solutions based on those generated using the heuristic.

In some particular scheduling problems in which similar or even the same sequences of tasks are regularly fed to a static set of resources, a schedule repository can be set up in order to store previous schedules. The repository maintains not only complete schedules, but also partial schedules commonly found in many complete schedules. Initial antibodies can be then composed via a process of schedule assortment using the schedule repository. This approach resembles the actual antibody generation process of the immune system.

The initial population generated in an AIS may undergo a refinement process similar to the negative selection of lymphocytes (T and B cells). The process is used as an attempt to ensure that the antibodies or solutions in the initial population cover a wide range of search space. A simple way to refine the initial population is to remove an antibody that overlaps its certain portion with another one.

11.5.1.3 Antigens

Due to the fact that antigens tend not to present direct applicability to scheduling their usefulness has been somewhat neglected. However, two possible applications of antigens are different task ordering and resource selection.

Typically, the order of tasks to be scheduled may be determined based on arrival time or priority, and it tends not to change in the course of scheduling. What's more, a single arrangement of the tasks is usually used. For example, tasks with precedence constraints in multiprocessor scheduling (e.g., list scheduling) are prioritized based on task dependency. There are a series of different prioritization methods, such as b-level, t-level and s-level. Their priorities are used to determine the scheduling order. The tasks are then scheduled by using a task allocation scheme. Note that the scheduling order has an effect on the quality of the final schedule. Here, a number of different orders can play as antigens. An antibody that has a strong match (i.e., good schedule) with one or more of these antigens can then be selected.

Another possible use of antigens is the effective selection of resources. It is normally the case that the numbers of tasks and resources do not match perfectly. This implies resources for the given tasks should be appropriately selected in order to generate a good, if not the optimal, schedule. In this regard an antigen can represent resource requirements, whereas an antibody represents available resources (e.g., memory, processing speed, etc.). An example of this usage (i.e., as a binary matching scheme) is introduced in King et al. (2001).

11.5.2 Immune Principles and Processes

An immune response is a result of a series of complex and sophisticated dynamics of immune entities. Over the lifetime of an organism these immune cells and molecules increasingly become more effective at responding to foreign intruders. This effectiveness is achieved by reinforcing the immune entities with various immune principles and processes. With an immune system based scheduling approach, one or more of these principles and processes typically are incorporated into it and repeatedly carried out as in the biological immune system.

11.5.2.1 Negative Selection

The negative selection of lymphocytes (antibodies) can be modeled and adopted in a few processes of scheduling. The first two are, as described earlier, at the resource selection and initial population generation stages. Another scheduling step in which the use of regulating self-reactive antibodies (similar to the negative selection process) can be found is at a late stage of scheduling schemes based on the immune network model. More specifically, in an immune network based scheduling approach the antibodies selected via a process of clonal selection are filtered out if they are bound by other antibodies, i.e., some schedules are nearly or exactly identical to others.

11.5.2.2 Danger Theory

The most fundamental issue in the danger theory is how to define and model a danger signal. The definition of a danger signal probably varies from one application to another. In dynamic scheduling the fluctuation of resource availability and capability can be modeled using danger theory. A danger signal in this case might be defined to be a certain degree of deviation from the expected performance of a resource. It is most appropriate when the resource plays a critical role in the schedule.

Assuming that in a given system (e.g., a grid) schedules are first generated based on the static information of the resources and adapted during the actual scheduling process as the environment changes, then the failure or sudden overload of a resource can be considered as dangerous to the quality of the statically generated schedule resulting in the modification of the schedule.

11.5.2.3 Clonal Selection

The clonal selection principle along with affinity maturation process is the most popularly adopted and modeled immune feature for two reasons. The first reason is the selection and mutation schemes are distinct and effective: they are not performed uniformly as in many evolutionary techniques, but directly and inversely proportional to the quality of a modeled immune entity, e.g., an antibody. The other reason is that most major steps involved in an AIS based on the clonal selection principle are analogous to those found in other well-developed evolutionary techniques; hence easy to model and implement.

There have been a noticeable number of studies (King et al. 2001; Costa et al. 2002; Coello Coello et al. 2003; Ong et al. 2005) conducted on the clonal selection principle for various scheduling problems. The approaches proposed in these studies differ from one another mostly in terms of functions that determine the clonal selection rate and the hypermutation rate.

A clonal selection based scheduling approach typically models one immune entity (antibody) to represent schedules and two immune processes (clonal selection and affinity maturation) to enhance schedules ultimately aiming at finding the optimal schedule. The same principle as in the immune system, applies: good schedules proliferate, whereas poor ones become extinct. These immune processes are the two primary sources of performance gain that AIS based on clonal selection principle can exploit. The goodness of the former has a significant impact on narrowing down search space leading towards the optimal schedule. One may model it in one of the following two approaches, both based on antibody affinity (the quality of schedule):

Proportional clonal generation. The clone size of an antibody is directly proportional to its affinity. The best clone among the clone set is compared with the original antibody and the better one is finally selected for the next generation.

Proportional clonal selection. The number of clones for every antibody is fixed, whereas the number of clones for each antibody to be selected is directly proportional to the affinity of the antibody.

It is obvious that the higher the affinity of an antibody the more its clones are generated or selected.

Note that the final clonal selection process in both approaches takes place after the affinity maturation process to clone. More specifically, each clone of an antibody undergoes hypermutations and/or receptor editing as a process of differentiation hoping to improve its affinity. It is then evaluated based on its affinity.

An antibody with a high affinity value can be interpreted as a solution similar to the optimal one; hence fewer modifications compared to that with a low affinity value. There are a series of ways to mutate antibodies. A simple mutation method might be a random one that arbitrary selects different points in an antibody and mutate them with randomly selected values. One can also adopt a mutation method that uses a set of systematic approaches to select those points to mutate and to generate new values. Here, the values of those mutating points might represent resource identifiers. A possible function that guides the hypermutation rate is the inverse of an exponential function (de Castro and Timmis 2002).

The receptor editing process in the immune system is very similar to the negative selection process in the primary lymphoid organs except that entirely new antibodies are generated in place of those eliminated.

11.5.2.4 Immune Networks

One may see the immune network model as a superset of the clonal selection principle. In addition to those processes carried out with clones, artificial immune network (AIN) models involve antibody stimulation and suppression as given by the immune network theory. These dynamics apply to the selected antibodies after the clonal selection process in order to further improve the superiority of the antibody population.

A simple application that models them with a slight modification in scheduling can be used to maintain the diversity as well as quality of schedules. For example, some previously discarded schedules may be kept for a certain number of evolutions and compared with the current schedules. If any schedule in the current schedule repertoire has been previously considered and discarded, it would be modified or replaced with a new one. A similar strategy to this can be found in TS.

11.5.3 Fitness Functions

As most immune system based scheduling approaches tend to be a single player (antibody) game, commonly used distance metrics (e.g., the Euclidean distance and the Manhattan distance) in many AIS are less frequently adopted. Rather, the Hamming distance and schedule length (the most typical metric in scheduling) has wide acceptance.

The Hamming distance between two strings is the number of symbols that are different. This metric can be used to measure the degree of matching in the resource selection process and the degree of schedule identity in negative selection and antibody stimulation and suppression in immune networks. Let S_1 and S_2 be two sets of schedules, each with a set of resources that are assigned to tasks. More formally,

$$S_1 = \{S_{1,1}, S_{1,2}, \ldots, S_{1,n}\} \quad \text{and} \quad S_2 = \{S_{2,1}, S_{2,2}, \ldots, S_{2,n}\}.$$

The Hamming distance $\mathrm{HD}(S_1, S_2)$ between two schedules, S_1 and S_2 is

$$\mathrm{HD}(S_1, S_2) = \sum_{i=1}^{n} \delta_i, \quad \text{where } \delta_i = \begin{cases} 0 & \text{if } S_{1,i} = S_{2,i} \\ 1 & \text{otherwise.} \end{cases} \tag{11.1}$$

The schedule length (the completion time) is defined to be the amount of time from the time the first task starts to the time the last task finishes. The time complexity of a scheduling algorithm heavily depends on the computational cost of its fitness function. Thus, when modeling an immune system based scheduling algorithm one should carefully choose its parameters, such as the population size, the number of generations and so on. This has a significant impact particularly on traditional multiprocessor scheduling in that the amount of scheduling time one can afford is much less than that in other scheduling problems.

One use of the Euclidean distance is for identifying the similarity between a pair of antibodies in an immune network model (Zuo and Fan 2005). The Euclidean distance is given by

$$\mathrm{ED}(S_1, S_2) = \sqrt{\sum_{i=1}^{n} (S_{1,i} - S_{2,i})}. \tag{11.2}$$

11.6 Scheduling Algorithms with Immune System Support: A Survey

The AIS approach has only recently drawn attention from researchers in the scheduling area. Although there are a growing number of AIS implementations for scheduling, many of them are not significantly different from each other. A selection of the unique AIS implementations are presented in this section.

11.6.1 Multiprocessor Scheduling

King et al. (2001) investigated functionalities of the immune system to design intelligent agents for task allocation in a heterogeneous computing environment. The main immune functionalities that inspired their AIS include the recognition process, learning and memory mechanisms.

The AIS was designed with two intelligent agents, H-cells and S-cells. They control hardware resources, and software properties and scheduling respectively.

The H-cells behave like typical resource managers in multiprocessor systems. However, they also use adaptive resonance theory (ART) as an adaptive method as well as the immunological functionalities mentioned above. Antigens are defined as adverse performance conditions and maintained by the H-cells clustering them based on their similarity. This clustering facilitates antibody adaptation in that an antigen similar to those in a particular antigen cluster can be quickly identified.

The primary functionalities of the S-cells are identifying the characteristics and resource requirements of a parallel program code, and making scheduling decisions. In addition, they monitor the progress of program execution and perform rescheduling if any abnormal behavior of the resources in the schedule is detected.

Costa et al. (2002) attempted to tackle an instance of the multiprocessor scheduling problem with the support of clonal selection and affinity maturation. Their AIS schedules a set of jobs to a set of identical parallel processors such that the completion time of the last processor to finish execution is the lowest possible. The AIS uses a lower bound solution calculated by the sum of all job processing times divided by the number of processors with the use of preemption. Antibodies representing schedules (strings of processor IDs) are compared against this lower bound solution to compute their affinity values. As usual the numbers of clones and mutations for each of the antibodies are determined by its affinity. Note that the number of mutations per antibody is empirically set. One interesting approach they used includes 5 types of mutation, and when a mutation is required, a mutation type is randomly selected.

11.6.2 Job Shop Scheduling

Coello Coello et al. (2003) applied the CLONALG algorithm with some modifications to approach the Job Shop Scheduling Problem (JSSP). In the JSSP, there are a set of jobs and a set of machines. Each job contains a series of a potentially different number of operations associated with precedence constraints and processing times. It is an NP-hard problem to assign the jobs onto the machines in such a way that the overall completion time of the jobs is minimal.

In each generation the AIS maintains an antigen and antibody, where both are possible schedules. The antigen is initially a randomly generated schedule and is constantly updated with the best schedule in each of the following generations. While most AIS generate antibodies using a uniform random distribution, the antibody generation method in this AIS implementation is derived from the actual mechanism (concatenating gene segments) in the immune system.

The major steps of the AIS involved in each generation are as follows: (1) an antibody is generated and improved by a local search. (2) The antibody is compared with the antigen and it replaces the antigen if it is better than the antigen. (3) The antibody then gets cloned and mutated. (4) The best segments (i.e., the best job sequence for each machine) of the clone are stored in the gene libraries. Most parameters, such as the number of clones and the number of gene libraries used are empirically determined.

Another AIS for the JSSP proposed is the adaptive scheduling system (Mori et al. 1998). It is mainly based on the immune network model. In addition to the minimization of makespan, the AIS attempts to achieve the minimization of setup and waiting times.

Antibodies are encoded in two different representations; hence two types. An antibody of type I is composed of a sequence of batch sizes, whereas one of type II represents a sequence of job priorities. The proliferation and suppression of antibodies is determined by the variety and the concentration of antibodies, respectively.

11.6.3 Flow Shop Scheduling

Engin and Doyen (2004) proposed an AIS, based on the clonal selection principle of the immune system with some effort to optimize parameters, to tackle hybrid flow shop (HFS) problems. The HFS problem in their study consists of a set J of jobs and a set P of machines. Each job in J is processed, undergoing a series of stages. At each stage it has to be processed on any one machine in P.

The proposed AIS modeled antibodies to represent sequences of jobs and the clonal selection principle with two different mutation methods, inverse and pairwise interchange. The size of antibody population and the elimination ratio of antibodies in their AIS are the two parameters attempted to be optimized. The rate of cloning for an antibody ab_i is calculated by a selection probability function defined by

$$CR(ab_i) = \frac{\max_{ab_j \in AB}\{SL(ab_j)\} + 1 - SL(ab_i)}{\sum_{j=1}^{|AB|} SL(ab_j)} \tag{11.3}$$

where AB is the antibody population and $SL(ab_i)$ is the makespan of the antibody ab_i. Apart from the adoption of a receptor editing process the AIS basically works similar to the CLONALG algorithm introduced earlier.

11.7 DAG Scheduling on Heterogeneous Computing Systems with Clonal Selection

This section presents an AIS for directed acyclic graph (DAG) scheduling in heterogeneous computing systems. The core immune component adopted for the AIS is the clonal selection principle. It is compared with a genetic algorithm (GA) and a well-known heuristic (HEFT) (Topcuoglu et al. 2002).

11.7.1 Problem Definition

Parallel programs, in general, can be represented by a directed acyclic graph. A DAG, $G = (V, E)$, consists of a set V of v nodes and a set E of e edges. A DAG is also called a task graph or macro-dataflow graph. In general, the nodes represent tasks partitioned from an application and the edges represent precedence constraints. An edge $(i, j) \in E$ between task n_i and task n_j also represents the inter-task communication. In other words, the output of task n_i has to be transmitted to task n_j in order for task n_j to start its execution. A task with no predecessors is called an *entry* task, n_{entry}, whereas an *exit* task, n_{exit}, is one that does not have any successors.

The target system used in this work consists of a set P of p heterogeneous processors/machines that are fully interconnected. The inter-processor communications are assumed to perform with the same speed on all links without contentions. It is also assumed that a message can be transmitted from one processor to another while a task is being executed on the recipient processor which is possible in many systems.

The communication to computation ratio (CCR) is a measure that indicates whether a task graph is communication intensive or computation intensive. For a given task graph, it is computed by the average communication cost divided by the average computation cost on a target system.

The task scheduling problem in this study is the process of allocating a set P of p processors to a set V of v tasks aiming to minimize SL without violating precedence constraints. The schedule length is defined to be the maximum completion time of the exit task in a task graph.

11.7.2 The Proposed Artificial Immune System

The AIS presented in this section adopts antibodies and clonal selection for representing schedules and refining the quality of schedules, respectively. Its mutation method is distinct from those found in many other AIS in that mutations take place at idling slots of processors in a schedule. A set of randomly generated schedules are fed into this affinity maturation process of the AIS. Here, the incorporation of a randomly generated schedule into the original schedule (antibody) is regarded as a mutation. This can be viewed as 'schedule overlapping' resulting in duplicating tasks. The major benefit of this mutation scheme is the reduction of communication overhead. The AIS algorithm is presented in Algorithm 11.1.

Note that the AIS removes the redundant tasks (step 9) after each mutation. This is because some tasks, after the mutation, have no effect for the schedule.

The number of clones for the antibody ab_i and that of mutations for the clone c_i^j are determined by the following equations:

$$NC = \max\left\{0, 2|AB| - |AB|\big(\big(SL(ab_i) - SSL\big)/SSL\big)\right\}, \tag{11.4}$$

Algorithm 11.1: The proposed AIS

```
1:  Generate initial antibody population at random
2:  for each generation do
3:      for each antibody $ab_i$ in the antibody population $AB$ do
4:          Clone $ab_i$ proportional to its affinity (schedule length)
5:          for each clone $c_i^j$ in the clone set $C_i$ do
6:              Generate a set $M$ of random schedules inversely proportional to
                $ab_i$'s affinity
7:              for each random schedule $m_k$ in $M$ do
8:                  Mutate $c_i^j$ with $m_k$
9:                  Remove redundant tasks in $c_i^j$
10:                 Compute the affinity of $c_i^j$
11:             end
12:         end
13:         Set $ab_i$ to its best clone whose affinity is higher than $ab_i$'s
14:     end
15:     Replace worst $b$ % of antibodies in $AB$ by randomly generated ones
16: end
```

$$NM = \max\big\{2, 2|AB|\big(\big(SL(ab_i) - SSL\big)/SSL\big)\big\}, \tag{11.5}$$

where AB is the antibody population, $SL(ab_i)$ is the schedule length of the antibody ab_i, and SSL is the shortest schedule length among schedule lengths of all antibodies in AB.

11.7.3 Experiments and Results

Typically, the schedule length of a task graph generated by a scheduling algorithm is used as the main performance measure of the algorithm. The performance metric used for the comparison is the *normalized schedule length* (NSL). The normalized schedule length is defined to be schedule length obtained by a particular algorithm over schedule length obtained by the HEFT algorithm.

The actual values of the parameters used for the AIS and GA are: (1) The number of generations = 10, (2) The numbers of antibodies/chromosomes = 10 and 55, (3) The number of mutations for GA = 10, and (4) The antibody elimination rate, $b = 20$ %.

The parameters used in the experiments are summarized in Table 11.1. The total number of experiments conducted with various randomly generated task graphs on the five different numbers of processors is 7,200. More specifically, the random task

Table 11.1 Experimental parameters

Parameter	Value
The number of tasks	U(10, 600)
CCR	{0.1, 1, 10}
The number of processors	{4, 8, 16, 32}
Processor heterogeneity	{100, 200, random}

graph set consists of 90 base task graphs generated with combinations of 10 graph sizes, 3 CCRs and 3 processor heterogeneity settings. For each combination 20 task graphs are randomly generated retaining the base one's characteristics. These 1,800 graphs are then experimented on with 4 different numbers of processors.

The computation and communication costs of the tasks in each task graph were randomly selected from a uniform distribution with the mean equal to the chosen average computation and communication costs. The processor heterogeneity value of 100 is defined to be the percentage of the speed difference between the fastest processor and the slowest processor in a given system.

It is clearly shown in Fig. 11.4 that the AIS delivers quite competitive schedule lengths irrespective of different application and system characteristics, e.g., graph sizes and the number of processors. The schedule lengths obtained from communication intensive task graphs indicate that the AIS best suits task graphs consisting of fine-grain tasks with large communication costs.

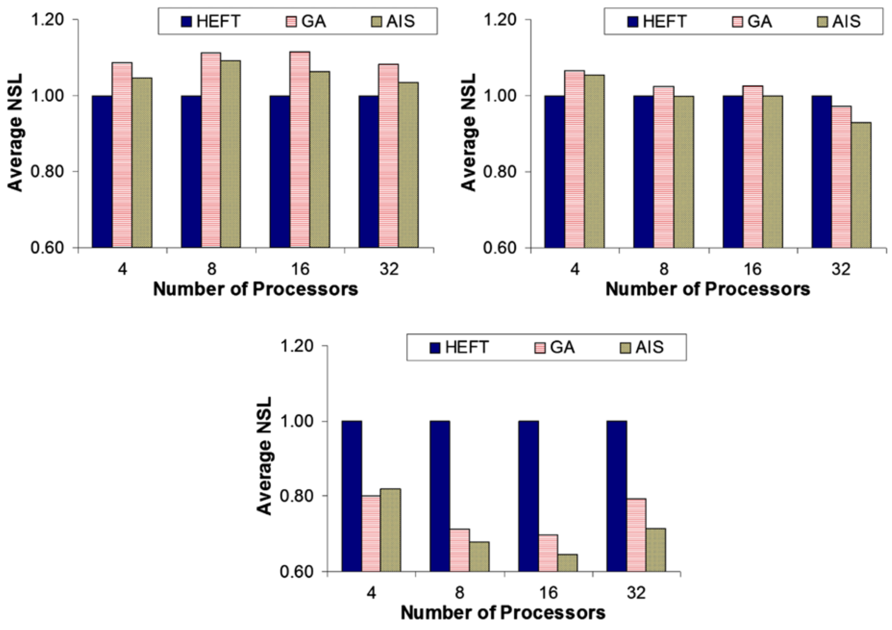


Fig. 11.4 Experimental results. *Upper left*: CCR = 0.1; *upper right*: CCR = 1; *bottom*: CCR = 10

11.8 Conclusion

As being still in its infant stage, the field of AIS has confidently exhibited its potential in several areas including scheduling. The noticeable performance of AIS mostly benefit from the distinct adaptability and effectiveness of the adaptive immune system. A simple AIS for multiprocessor scheduling, presented in the previous section, once again demonstrates that AIS can be good alternatives for tackling many problems that are computationally hard and/or in dynamic environments.

Although an increasing number of researchers have been putting a lot of efforts in applying immune features to various areas, many proposed AIS are limited to certain types of applications and to being developed based on a few popular principles and processes of the adaptive immune system. Therefore, the applicability of various other immune characteristics should be further investigated and exploited.

References

Aickelin, U., & Cayzer, S. (2002). The danger theory and its application to artificial immune systems. In J. Timmis & P. J. Bentley (Eds.), *Proceedings of the first international conference on artificial immune systems (ICARIS)* (pp. 141–148). University of Kent at Canterbury, September 2002. University of Kent at Canterbury Printing Unit.

Aickelin, U., Bentley, P., Cayzer, S., Kim, J., & McLeod, J. (2003). Danger theory: the link between AIS and IDS? In *Lecture notes in computer science: Vol. 2787. Proceedings of the second international conference on artificial immune systems (ICARIS)* (pp. 147–155). Berlin: Springer.

Ayara, M., Timmis, J., de Lemos, R., de Castro, L., & Duncan, R. (2002). Negative selection: how to generate detectors. In J. Timmis & P. J. Bentley (Eds.), *Proceedings of the first international conference on artificial immune systems (ICARIS)* (pp. 89–98). University of Kent at Canterbury, September 2002. University of Kent at Canterbury Printing Unit.

Bersini, H. (2002). The immune and the chemical crossover. *IEEE Transactions on Evolutionary Computation*, *6*(3), 306–313.

Burgess, M. (1998). Computer immunology. In *Proceedings of the 12th USENIX conference on system administration* (pp. 283–298). Boston: USENIX Association.

Burnet, F. M. (1959). *The clonal selection theory of acquired immunity*. Cambridge: Cambridge University Press.

Coello Coello, C. A., Rivera, D. C., & Cortés, N. C. (2003). Use of an artificial immune system for job shop scheduling. In *Lecture notes in computer science: Vol. 2787/2003. Proceedings of the second international conference on artificial immune systems (ICARIS)* (pp. 1–10). Berlin: Springer.

Costa, A. M., Vargas, P. A., Von Zuben, F. J., & Franca, P. M. (2002). Makespan minimization on parallel processors: an immune-based approach. In *Proceedings of the 2002 congress on evolutionary computation (CEC'02)* (Vol. 1, pp. 920–925). Washington: IEEE.

Cutello, V., & Nicosia, G. (2002). Multiple learning using immune algorithms. In *Fourth international conference on recent advances in soft computing (RASC-2002)* (pp. 102–107). Nottingham, UK. Berlin: Springer.

D'haeseleer, P., Forrest, S., & Helman, P. (1996). An immunological approach to change detection: algorithms, analysis and implications. In *Proceedings of IEEE symposium on security and privacy* (pp. 132–143). Oakland: IEEE.

Dasgupta, D., Cao, Y., & Yang, C. (1999). An immunogenetic approach to spectra recognition. In W. Banzhaf, J. M. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, & R. E. Smith (Eds.), *Proceedings of the genetic and evolutionary computation conference (GECCO)* (pp. 149–155). San Francisco: Morgan Kaufmann.

Dasgupta, D., Krishna Kumar, K., Wong, D., & Berry, M. (2004). Negative selection algorithm for aircraft fault detection. In *Lecture notes in computer science: Vol. 3239. Proceedings of the third international conference on artificial immune systems (ICARIS)* (pp. 1–13). Berlin: Springer.

de Castro, L. N., & Timmis, J. (2002). *Artificial immune systems: a new computational intelligence approach*. London: Springer.

de Castro, L. N., & Von Zuben, F. J. (2001). AINET: an artificial immune network for data analysis. In H. A. Abbass, R. A. Sarker, & C. S. Newton (Eds.), *Data mining: a heuristic approach* (pp. 231–259). Hershey: Idea Group. Chap. XII.

de Castro, L. N., & Von Zuben, F. J. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, *6*(3), 239–251.

Engin, O., & Doyen, A. (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system. *Future Generations Computer Systems*, *20*(6), 1083–1095.

Esponda, F., Ackley, E. S., Forrest, S., & Helman, P. (2005). On-line negative databases. *International Journal of Unconventional Computing*, *1*(3), 201–220.

Farmer, J., Packard, N., & Perelson, A. (1986). The immune system, adaptation and machine learning. *Physica. D*, *22*, 187–204.

Feng, Y.-J., & Feng, Z.-R. (2004). An immunity-based ant system for continuous space multi-modal function optimization. In *Proceedings of international conference on machine learning and cybernetics* (Vol. 2, pp. 1050–1054). Washington: IEEE.

Forrest, S., Perelson, A. S., Allen, L., & Cherukuri, R. (1994). Self-nonself discrimination in a computer. In *Proceedings of IEEE symposium research in security and privacy* (pp. 202–212). Washington: IEEE.

Gao, X. Z., Ovaska, S. J., Wang, X., & Chow, M.-Y. (2004). Neural networks-based negative selection algorithm with applications in fault diagnosis. In *Proceedings of international conference on systems, man and cybernetics* (Vol. 4, pp. 3408–3414).

Garrett, S. (2005). How do we evaluate artificial immune systems? *Evolutionary Computation*, *13*(2), 145–177.

Gonzales, L. J., & Cannady, J. (2004). A self-adaptive negative selection approach for anomaly detection. In *Proceedings of congress on evolutionary computation (CEC 04)* (Vol. 2, pp. 1561–1568).

Gonzalez, F., Dasgupta, D., & Kozma, R. (2002). Combining negative selection and classification techniques for anomaly detection. In *Proceedings of congress on evolutionary computation (CEC'02)* (Vol. 1, pp. 705–710).

Grama, A., Gupta, A., Karypis, G., & Kumar, V. (2003). *Introduction to parallel computing* (2nd ed.). Boston: Addison Wesley.

Hajela, P., & Yoo, J. S. (1999). Immune network modelling in design optimization. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization* (pp. 203–215). London: McGraw-Hill.

Hofmeyr, S., & Forrest, S. (2000). Architecture for the artificial immune system. *Evolutionary Computation*, *8*(4), 443–473.

Jerne, N. (1974). Towards a network theory of the immune system. *Annals of Immunology*, *125*, 373–389.

Kim, J., & Bentley, P. J. (2001). Towards an artificial immune system for network intrusion detection: an investigation of clonal selection with a negative selection operator. In *Proceedings of congress on evolutionary computation (CEC'01)* (Vol. 2, pp. 1244–1252). Washington: IEEE.

King, R. L., Russ, S. H., Lambert, A. B., & Reese, D. S. (2001). An artificial immune system model for intelligent agents. *Future Generations Computer Systems*, *17*(4), 335–343.

Krishna Kumar, K., Satyadas, A., & Neidhoefer, J. (1995). An immune system framework for integrating computational intelligence paradigms with applications to adaptive control. In M. Palaniswami, Y. Attikiouzel, R. J. Marks II, D. Fogel, & T. Fukuda (Eds.), *Computational intelligence a dynamic system perspective* (pp. 32–45). New York: IEEE Press.

Kwok, Y. K., & Ahmad, I. (1998). Benchmarking the task graph scheduling algorithms. In *Proceedings of first merged international parallel symposium/Symposium on parallel and distributed processing (IPPS/SPDP '98)* (pp. 531–537). Washington: IEEE.

Matzinger, P. (2002). The danger model: a renewed sense of self. *Science*, *296*, 301–305.

Mori, K., Tsukiyama, M., & Fukuda, T. (1998). Adaptive scheduling system inspired by immune system. In *Proceedings of international conference on systems, man, and cybernetics* (Vol. 4, pp. 3833–3837). Washington: IEEE.

Ong, Z. X., Tay, J. C., & Kwoh, C. K. (2005). Applying the clonal selection principle to find flexible job-shop schedules. In *Proceedings of international conference on artificial immune systems (ICARIS)* (pp. 442–455). Berlin: Springer.

Stibor, T., Timmis, J., & Eckert, C. (2005). On the appropriateness of negative selection defined over Hamming shape-space as a network intrusion detection system. In *Proceedings of congress on evolutionary computation* (Vol. 2, pp. 995–1002). Washington: IEEE.

Swiecicka, A., Seredynski, F., & Zomaya, A. Y. (2006). Multiprocessor scheduling and rescheduling with use of cellular automata and artificial immune system support. *IEEE Transactions on Parallel and Distributed Systems*, *17*(3), 253–262.

Timmis, J., & Neal, M. J. (2000). A resource limited artificial immune system for data analysis. In *Proceedings of ES 2000* (pp. 19–32). Berlin: Springer.

Topcuoglu, H., Hariri, S., & Wu, M. (2002). Performance-effective and low-complexity TaskScheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, *13*(3), 260–274.

Varela, F. J., & Coutinho, A. (1991). Second generation immune networks. *Immunology Today*, *12*(55), 159–166.

Wierzchon, S. T. (2000). Discriminative power of the receptors activated by k-contiguous bits rule. *Journal of Computer Science and Technology*, *1*(3), 1–13. Special Issue on Research Computer Science.

Zuo, X.-Q., & Fan, Y.-S. (2005). Solving the job shop scheduling problem by an immune algorithm. In *Proceedings of international conference on machine learning and cybernetics* (Vol. 6, pp. 3282–3287). Washington: IEEE.

Chapter 12
Formal Immune Networks: Self-Organization and Real-World Applications

Alexander O. Tarakanov and Alla V. Borisova

12.1 Introduction

Two types of self-organizing biological systems, the neural system and the immune system of the vertebrates possess the capabilities of "intelligent" information processing, which include memory, the ability to learn, to recognize, and to make decisions with respect to unknown situations. The potential of the natural neural system as a biological prototype of a computing scheme has already been well-established as a field of artificial neural networks, or neural computing (Cloete and Zurada 2000). However, the computing capabilities of the natural immune system (Jerne 1973, 1974; de Boer et al. 1992) have only recently been appreciated as a field of artificial immune systems (AIS) (Dasgupta 1999; de Castro and Timmis 2002; NASA 2004). The mathematical formalization of these capabilities (Tarakanov and Dasgupta 2000) forms the basis of immunocomputing (IC) as a new computing approach that replicates the principles of information processing by proteins and immune networks (Tarakanov et al. 2003).

This IC approach looks rather constructive as a basis for a new kind of computing. It has been reported a row of successful applications of IC to real-world tasks, including detection of dangerous ballistic situations in near Earth space (Tarakanov and Dasgupta 2002), computing of ecological map and optical response of laser diode (Tarakanov and Tarakanov 2004, 2005), and reconstruction of hydro-acoustic fields (Tarakanov et al. 2007). It is also worth noting that a connection between IC and cellular automata has led to encouraging results in three-dimensional (3D) computer graphics (Tarakanov and Adamatzky 2002).

As for biological applications of IC, a concept of "biomolecular immunocomputer" as a computer controlled fragment of the natural immune system has recently been reported (Goncharova et al. 2005). A connection of IC with brain research has

A.O. Tarakanov (✉) · A.V. Borisova
St. Petersburg Institute for Informatics and Automation, Russian Academy of Sciences,
14-line 39, St. Petersburg 199178, Russia
e-mail: tar@iias.spb.su

also led to promising results in understanding of basic principles of organization of receptor mosaics and molecular networks (Agnati et al. 2005a, 2005b).

In such background, this chapter proposes a generalized model of formal immune network (FIN) based on self-organizing features of apoptosis (programmed cell death) and autoimmunization both induced by cytokines (messenger proteins). The chapter also describes real-world applications of such FIN to intrusion detection in computer networks and forecast of hydro-physical fields in the ocean. The obtained results demonstrate that FIN outperforms (by training time and accuracy) other approaches of computational intelligence as well as more conventional methods of interpolation.

12.2 Biomolecular Background

Cytokines (messenger proteins) are a group of biologically active mediator molecules that provide the intercellular interactions within the immune system. They are the central regulators of leukocyte growth and differentiation, being produced by a wide variety of cell types, targeting various cell subsets and exhibiting numerous biological activities.

Up to now more than 100 different human cytokines are identified. An increasing volume of experimental data suggests that cytokines play one of the central roles in the immune regulation as well as in the neuro-immune-endocrine modulation (Ader et al. 2001). Such concept of cytokines as a network modulating and switching several cascades of immune reactions (Balkwill 2000) adjoins with the concept considering such molecules as a field or a milieu, which local properties mediate immune response (Kourilsky and Truffa-Bachi 2001).

There exists a relationship between cytokine levels in human body fluids and disease pathogenesis, including the inflammation and even depression (Bunk 2003). Many types of cancers have taken advantage of the regulatory role of cytokines to down-regulate appropriate immune responses targeted at destroying cancer cells. They do this by secreting immunosuppressive cytokines that induce generalized and specific inhibition of immune responses (Kurzrock 2001; Igney and Krammer 2002). So, the use of immunostimulatory cytokines as tumor vaccines has become a promising strategy in cancer immunotherapy (Vilcek and Feldman 2004).

Recent developments show that cytokines induce apoptosis (programmed cell death) in cancer cells (Wall et al. 2003). The induction of apoptosis is associated with a dose-dependent inhibition of cancer cell division, and this activity has been demonstrated for a wide range of cancer types including bladder, breast, leukemia, melanoma, ovarian and prostate.

Apoptosis is a natural mechanism by which cells "commit suicide" when they have outlived their purpose, become defective, or have aged. Apoptosis prevents cells from accumulating and forming tumors. Understanding of the control of apoptosis in normal and malignant cells will help to improve the diagnosis and treatment of malignancies. The goal of many treatments, including chemotherapies is to induce malignant cells to undergo apoptosis. Current data also suggests that a cytokine may function as a dual-acting cytokine in which its normal physiological functions may be related to specific aspects of the immune system and over-expression culminates in cancer-specific apoptosis (Fisher et al. 2003).

Based on such biomolecular background, a notion of cytokine FIN (cFIN) has been proposed in our previous work (Tarakanov et al. 2005a). Below, this mathematical notion is generalized and applied to two tasks where self-organizing features of FIN seem to play a key role.

12.3 General Mathematical Model

Vector-matrix transposing will be designated by upper stroke $[\,]'$. For example, if X is a column vector then X' is a row vector. End of proof will be designated by the symbol $\diamondsuit$ ("rhomb").

Definition 1 Cell is a pair $V = (f, P)$, where f is real value $f \in R$, whereas $P = (p_1, \dots, p_q)$ is a point of q-dimensional space: $P \in R^q$, and P lies within unit cube: $\max\{|p_1|, \dots, |p_q|\} \le 1$.

Let distance ("affinity") $d_{ij} = d(V_i, V_j)$ between cells V_i and V_j be defined by a norm:

$$d_{ij} = \|P_i - P_j\|.$$

For example, it can be Euclidean $\|P\|_E$, Manhattan $\|P\|_M$, Tchebyshev $\|P\|_T$, or any appropriate norm:

$$\|P\|_E = \sqrt{p_1^2 + \cdots + p_q^2},$$

$$\|P\|_M = |p_1| + \cdots + |p_q|,$$

$$\|P\|_T = \max\{|p_1|, \dots, |p_q|\}.$$

Fix some finite non-empty set of cells ("innate immunity") $W_0 = (V_1, \dots, V_m)$ with non-zero distance between cells: $d_{ij} \ne 0, \forall i, j \colon i \ne j$.

Definition 2 FIN is a set of cells: $W \subseteq W_0$.

Definition 3 Cell V_i recognizes cell V_k if the following conditions are satisfied:

$$|f_i - f_k| < \rho, \qquad d_{ik} < h, \qquad d_{ik} < d_{ij}, \qquad \forall V_j \in W, \quad j \ne i,\ k \ne j,$$

where $\rho \ge 0$ and $h \ge 0$ are non-negative real values ("recognition threshold" and "affinity threshold").

Let us define the behavior ("self-organization") of FIN by the following two rules.

Rule 1 (Apoptosis) If cell $V_i \in W$ recognizes cell $V_k \in W$ then remove V_i from FIN.

Rule 2 (Autoimmunization) If cell $V_k \in W$ is nearest to cell $V_i \in W_0 \backslash W$ among all the cells of FIN: $d_{ik} < d_{ij}$, $\forall V_j \in W$, whereas $|f_i - f_k| \geq \rho$, then add V_i to FIN.

Let W_A be FIN as a consequent of application of apoptosis to all cells of W_0. Let W_I be FIN as a consequence of autoimmunization of all cells of W_A by all cells of W_0. Note that the resulting sets W_A and W_I depend on the ordering of cells in W_0. Further it will be assumed that the ordering is given.

Let us consider general mathematical properties of FIN. It is obvious that neither the result of apoptosis W_A nor the result of autoimmunization W_I can overcome W_0 for any innate immunity or threshold:

$$W_A \subseteq W_0, \qquad W_I \subseteq W_0, \quad \forall W_0, h, \rho.$$

The following Propositions give more important and less evident properties of FIN.

Proposition 1 *For any innate immunity W_0 and recognition threshold ρ there exists affinity threshold h_0 such that apoptosis does not change W_0 for any h less than h_0: $W_A = W_0$, $\forall h < h_0$.*

Proof Let h_0 be the minimal distance for any pair of FIN cells that satisfy the recognition threshold:

$$h_0 = \min_{i,j}\{d_{ij}\}: \quad |f_i - f_j| < \rho, \quad i \neq j.$$

Then, according to Definition 3, none of the cells of FIN can recognize other cells, because $d_{ij} > h_0$ for any pair of cells V_i and V_j. According to Rule 1, none of the cells can be removed from FIN for any h less than h_0, because $d_{ij} > h$, $\forall h < h_0$, $\forall V_i, V_j \in W_0$. Thus, $W_A = W_0$, $\forall h < h_0$. □

Proposition 2 *For any innate immunity W_0 and recognition threshold ρ there exists affinity threshold h_1 such that the consequence of apoptosis and autoimmunization $W_1 = W_I(h_1)$ provides the minimal number of cells $|W_1|$ for given W_0 and ρ, and any h: $|W_1| \leq |W_I(h)|$, $\forall h$, $\forall W_I \subseteq W_0$.*

Proof Let h_1 be maximal distance for any pair of cells of FIN, which satisfy to recognition threshold:

$$h_1 = \max_{i,j}\{d_{ij}\}: \quad |f_i - f_j| < \rho, \quad i \neq j.$$

Then, according to Definition 3, any cell V_i can recognize the nearest cell V_j if the last one satisfies the recognition threshold: $|f_i - f_j| < \rho$. Let W_- be the set of all such cells V_i. Then, according to Rule 1, $|W_A(h_1)| = |W_0| - |W_-|$, and the number of such cells after apoptosis is minimal among any h: $|W_A(h_1)| \leq |W_A(h)|$, $\forall h$.

Let W_+ be set of cells, which is added to $W_A(h_1)$ as a consequence of autoimmunization: $W_1 = W_A(h_1) \cup W_+$. It is also evident that W_+ is a subset of W_-: $W_+ \subseteq W_-$, and $|W_+|$ represents a number of "mistakes" of apoptosis when FIN "kills" some cells, which lead to further recognition errors. Such cells are then "restored" by autoimmunization (Rule 2).

Let $W_* = W_- \backslash W_+$ be cells which yield apoptosis without further recognition errors. Then $|W_+| = |W_-| - |W_*|$. On the other hand: $|W_1| = |W_A(h_1)| + |W_+|$. Substitutions of $|W_A(h_1)|$ and $|W_+|$ lead to the following result: $|W_1| = |W_0| - |W_*|$. Thus, $|W_1| \le |W_I(h)|$, which proves Proposition 2. □

Actually, Proposition 2 states that the minimal number of cells after apoptosis and autoimmunization is a kind of "inner invariant" of any FIN, which depends on the innate immunity and the recognition threshold but does not depend on the affinity threshold. Practically, it means that such invariant can be found for any FIN by apoptosis and autoimmunization without considering any affinity threshold (in Definition 3) at all.

Now we can define a general model of molecular recognition in terms of FIN. Let "epitope" ("antigenic determinant") be any point $P = (p_1, \ldots, p_q)$ of q-dimensional space: $P \in R^q$. Note that any cell of FIN also contains an epitope, according to Definition 1.

Definition 4 Cell V_i recognizes epitope P by assigning him value f_i if the distance $d(V_i, P)$ between the cell and the epitope is minimal among all cells of FIN: $d(V_i, P) = \min\{d(V_j, P)\}, \forall V_j \in W$.

Let pattern ("molecule") be any n-dimensional column-vector $Z = [z_1, \ldots, z_n]'$, where $z_1, \ldots, z_n$ are real values. Let pattern recognition be mapping of the pattern to an epitope: $Z \to P$, and recognition of the epitope by the value f of the nearest cell of FIN.

Let $X_1, \ldots, X_m$ be a set of n-dimensional patterns ("cells") with known values $f_1, \ldots, f_m$. Let $A = [X_1 \ldots X_m]'$ be matrix of dimension $m \times n$. Consider singular value decomposition (SVD) of this matrix (Horn and Johnson 1986):

$$A = s_1 L_1 R_1' + \cdots + s_q L_q R_q' + \cdots + s_r L_r R_r', \tag{12.1}$$

where r is the rank of matrix A, s_k are singular values and L_k, R_k are left and right singular vectors with the following properties:

$$L_k' L_k = 1, \quad R_k' R_k = 1, \quad L_k' L_i = 0, \quad R_k' R_i = 0, \quad i \ne k, \ k = 1, \ldots, r,$$
$$s_{k-1} \ge s_k, \quad k > 1.$$

Consider the following mapping of any n-dimensional pattern Z to epitope P:

$$p_k = \frac{1}{s_k} Z' R_k, \quad k = 1, \ldots, q, q \le r. \tag{12.2}$$

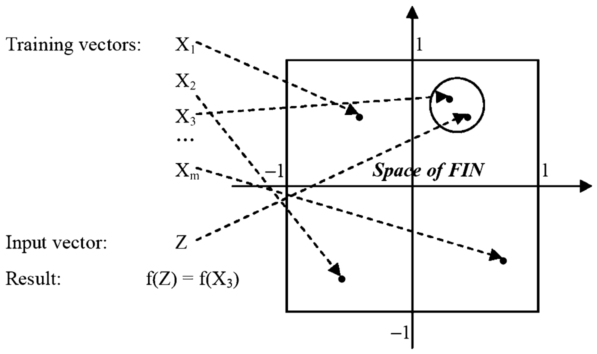


Fig. 12.1 Example of pattern recognition in 2D space of FIN

Note that any epitope obtained by application of formulas (12.2) to any training pattern lies within unit cube (see Definition 1), according to the above properties of singular vectors.

If value f (in Definition 1) is a natural number $f = c$, where $c \in N$ (i.e. "cytokine" or "class"), whereas recognition threshold (in Definition 3) $\rho < 1$ and distance between cells is determined by Tchebyshev norm, then we obtain a special case of cFIN (see Sect. 12.2) proposed in Tarakanov et al. (2005a) and applied for discrete pattern recognition.

12.4 General Pattern Recognition Algorithm

The IC approach to pattern recognition is inspired by a principle of molecular recognition between proteins, including antibody (also called immunoglobulin) of natural immune system and any other antigen (including another antibody). Consider the following informal example to shed light upon this inspiration.

Let *Ig1* and *Ig2* be two antibodies, while *Ag* be antigen. The strength of biophysical interaction between any pair of proteins can be measured by their binding energy. Let $e1$, $e2$ be values of binding energy between *Ag* and *Ig1, Ig2*, correspondingly. Then any protein (including antibody) can be presented and recognized by corresponding couple of numbers $e1$ and $e2$ in such two-dimensional immune network of interactions formed by two antibodies *Ig1* and *Ig2*. Consider more formal description of this idea.

In terms of general model (see previous section), any n-dimensional input vector Z ("antigen") is projected to q-dimensional space of FIN and recognized by class (value f, in general) of the nearest point ("cell") of FIN (e.g. see Fig. 12.1, where $q = 2$). Coordinate axes of such space of FIN are determined by right singular vectors ("antibodies") of SVD of the training matrix $A = [X_1 \dots X_m]'$, where $X_1, \dots, X_m$ are n-dimensional training vectors.

Such using of SVD to construct "antibodies" of FIN has some theoretical and practical advantages over other methods of extracting features from training data. For example, SVD guarantees mathematically optimal reconstruction of the training matrix by reduced set of components so that least square error is minimal among all other methods. As a method of linear algebra, SVD can be implemented by relatively simple and robust algorithm.

According to the general model of FIN, consider the following description (in pseudo-code) of generalized IC algorithm of pattern recognition, which provides real-valued (continuous) output f for any input vector Z. Main idea of this generalization is to find more than one nearest cell (point) in the space of FIN and interpolate the output of FIN using the known values of the function in these nearest training points.

Algorithm 12.1: Generalized IC algorithm of pattern recognition

```
1:  while Training do
2:      begin 1st stage training          // map training data to FIN
3:          Get training patterns
4:          Form training matrix
5:          Compute SVD of the training matrix
6:          Store $q$ singular values      // "binding energies"
7:          Store $q$ right singular vectors   // "antibodies"
8:          Store left singular vectors    // cells (points) of FIN
9:      end
10:     begin 2nd stage training
11:         // compress data by "self-organization" of FIN
12:         // compute consecutively for all cells of FIN:
13:         begin Apoptosis
14:             if $cell[i1]$ is the nearest to $cell[i2]$ and
                abs($f.cell[i1] - f.cell[i2]$) $<$ recognition_threshold then
15:                 kill $cell$[i1]
16:             end
17:         end
18:         begin Autoimmunization      // correct mistakes of Apoptosis
19:             if $cell[i1]$ is the nearest to $cell[i2]$ and
                abs($f.cell[i1] - f.cell[i2]$) $>=$ recognition_threshold then
20:                 restore $cell[i1]$
21:             end
22:         end
23:     end
24: end
25: while Recognition do
26:     Get pattern                    // "antigen"
27:     Map the pattern to FIN
28:     Find $p$ nearest cells of FIN
29:     Interpolate value $f$ by the values of $p$ nearest cells
30:     Assign the interpolated value to the pattern
31: end
```

Steps 1–7 below describe the algorithm in more rigorous mathematical terms. Note that Steps 1–5 are usual also for the IC algorithm of (discrete) pattern recognition, whereas Steps 6 and 7 provide a real-valued (continuous) output of the generalized FIN.

1. Form training matrix $A = [X_1 \dots X_m]'$ of dimension $m \times n$.
2. Compute first q singular values $s_1, \dots, s_q$ and corresponding left and right singular vectors $L_1, \dots, L_q$ and $R_1, \dots, R_q$ by SVD of training matrix (12.1), where $q \le r$ and r is rank of the matrix.

   According to Tarakanov et al. (2003), such SVD can be computed by the following iterative scheme (Steps 2.1–2.3).

   2.1. Compute maximal singular value s_1 and corresponding singular vectors L_1 and R_1 of the training matrix:

   $$L_{(0)} = [1 \dots 1]',$$

   $$R' = L'_{(k-1)} A, \qquad R_{(k)} = \frac{R}{|R|},$$

   $$L = A R_{(k)}, \qquad L_{(k)} = \frac{L}{|L|},$$

   $$s_{(k)} = L'_{(k)} A R_{(k)},$$

   where $|X| = \|X\|_E$ and $k = 1, 2, \dots$ until the following condition is satisfied for given constant ε:

   $$|s_{(k)} - s_{(k-1)}| < \varepsilon.$$

   Then

   $$s_1 = s_{(k)}, \qquad L_1 = L_{(k)}, \qquad R_1 = R_{(k)}.$$

   2.2. Form matrices

   $$A_2 = A - s_1 L_1 R_1', \quad A_3 = A_2 - s_2 L_2 R_2', \quad \dots,$$
   $$A_q = A_{q-1} - s_{q-1} L_{q-1} R'_{q-1},$$

   and compute their maximal singular values and corresponding singular vectors by Step 2.1.

   2.3. Store q singular values $s_1, \dots, s_q$ and right and left singular vectors $R_1, \dots, R_q$ and $L_1, \dots, L_q$.
3. For any training vector X_i, compute its mapping $Y(X_i) = [y_{i1} \dots y_{iq}]'$ to q-dimensional space of FIN:

   $$y_{i1} = \frac{1}{s_1} X_i' R_1, \dots, \qquad y_{iq} = \frac{1}{s_q} X_i' R_q.$$

4. Using apoptosis and autoimmunization, reduce m training points of FIN to $k \le m$ points, where the points number k is "self-defined" by the inner invariant of FIN (see Proposition 2).

5. For any n-dimensional vector Z, compute its mapping $Y(Z) = [y_1 \dots y_q]'$ to q-dimensional space of FIN:

$$y_1 = \frac{1}{s_1} Z' R_1, \dots, \qquad y_q = \frac{1}{s_q} Z' R_q.$$

6. Among the reduced training points of FIN $Y_1, \dots, Y_k$, determine p nearest to $Y(Z)$ points $Y_1, \dots, Y_p$ and their distances:

$$d_1 = \|Y_1 - Y\|, \dots, \qquad d_p = \|Y_p - Y\|.$$

7. Interpolate $f(Z)$ by the following sum:

$$f = \sum_{i=1}^{p} a_i f_i,$$

where $f_i = f(Y_i)$ are training values, which correspond to the nearest points of FIN, whereas coefficients a_i are determined by the distances:

$$a_i = \frac{1}{1 + d_i \sum_{j \neq i}^{p} \frac{1}{d_j}}.$$

Note the following useful features of FIN. It can be shown that

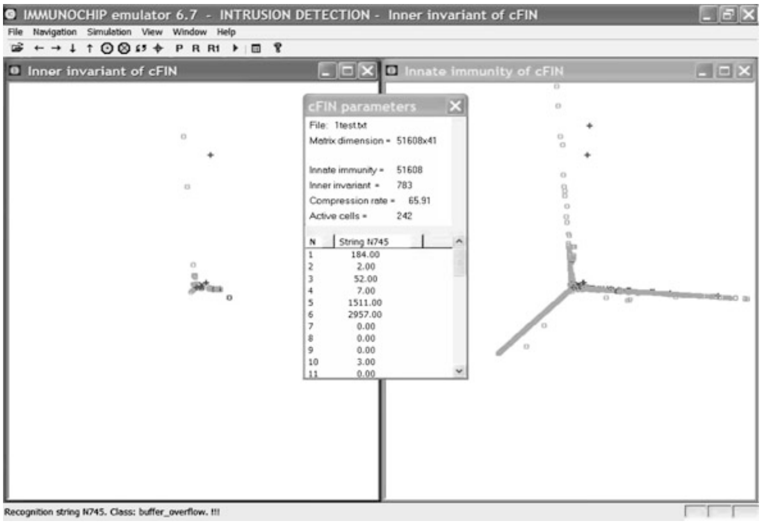
$$\sum_{i=1}^{p} a_i = 1.$$

It can be also shown that $f = f_i$ if $d_i = 0$ for any i (then $d_j \neq 0$ for any $j \neq i$). To prove this, consider a special case when input antigen represents a row of the training matrix and, thus, it is equal to a training vector: $Z = X_i, i = 1, \dots, m$. According to SVD properties utilized by Steps 1–3, the projection of such antigen to the space of FIN coincides with corresponding training point of FIN: $Y(Z) = Y(X_i)$. In such case, Step 4 calculates the following distances of the nearest points of FIN: $d_1 = 0, d_2 \neq 0, \dots, d_p \neq 0$. Then, according to Step 7: $a_1 = 1, a_2 = 0, \dots a_p = 0$, and the output of FIN is equal to the value of the function $f(X_i)$ for corresponding training vector X_i: $f = f_i$.

Thus, IC does not make mistakes on any training set.

Note that Step 2 can be also considered as an example of self-organization of FIN (together with apoptosis and autoimmunization of Step 4). It can be said that iterations of Step 2 for any training set "self-converge" to "antibodies". More rigorously, such convergence can be derived from Rayleigh-Ritz theorem (Horn and Johnson 1986) due to the fact that maximal singular value of any matrix A is actually the maximum of the bilinear form $L'AR$ over unit vectors $L'L = 1$, $R'R = 1$.

It is also worth noting that this algorithm can be supplied by online learning capabilities. In case of a change in any training vector X_i, just compute its mapping to the space of FIN (by Step 3) and add this point to the reduced training points of


IMMUNOCHIP emulator 6.7 - INTRUSION DETECTION - Inner invariant of cFIN
File Navigation Simulation View Window Help
Inner invariant of cFIN
Innate immunity of cFIN
cFIN parameters

File: 1test.txt
Matrix dimension = 51608x41

Innate immunity = 51608
Inner invariant = 783
Compression rate = 65.91
Active cells = 242

N	String N745
1	184.00
2	2.00
3	52.00
4	7.00
5	1511.00
6	2957.00
7	0.00
8	0.00
9	0.00
10	3.00
11	0.00

Recognition string N745. Class: buffer_overflow. !!!

Fig. 12.2 Intrusion detection by cFIN: "Antigen" (String 745 of File 1.1) is mapped to 3D cFIN (bold skew cross in the centers of both screens) and recognized by the "cytokine" of the nearest cell of cFIN ("Class: buffer_overflow !!!" in the bottom status bar). Cells of cFIN related to the attacks are designated by bold "+"; normal class cells are designated by "o". Note that three clumps of "normal" cells of cFIN ("Innate immunity" in *right-hand screen*) look like "tumors" eliminated by apoptosis and autoimmunization ("Inner invariant" in *left-hand screen*)

FIN (to use in Step 6). Therefore, in case of a change in the training patterns, the training phase does not necessarily have to be repeated with the new training set.

12.5 Intrusion Detection in Computer Networks

A special case of cFIN (see Sects. 12.2, 12.3) has been implemented as a version of the "immunochip emulator" (Tarakanov et al. 2005b) using Visual C++ with build in assembler code of the affinity function (Tchebyshev norm) in 3D space ($q = 3$) and OpenGL tools for user-friendly 3D visualization. A screen-shot of the emulator is shown in Fig. 12.2.

The known UCI KDD archive (Bay 1999) has been used for testing the emulator. Lincoln Labs set up an environment to acquire nine weeks of raw TCP (transmission control protocol) dump data simulating a typical US Air Force local area network (LAN). They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP (Internet protocol) address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes.

Two data files from KDD archive has been used to test the emulator:

– File 1: kddcup_data_10_percent_gz.htm (7.7 MB);
– File 2: kddcup_newtestdata_10_percent_ unlabeled_gz.htm (44 MB).

File 1 is the training data file. It contains 51608 network connection records. Any record (file string) has the following format, where parameters 2, 3, 4, 42 are symbolic, while other 38 parameters are numerical (real values):

```
1) duration, 2) protocol_type, 3) service, 4) flag,
5) src_bytes, 6) dst_bytes, 7) land, 8) wrong_fragment,
9) urgent, 10) hot, 11) num_failed_logins, 12) logged_in,
13) num_compromised, 14) root_shell, 15) su_attempted,
16) num_root, 17) num_file_creations, 18) num_shells,
19) num_access_files, 20) num_outbound_cmds, 21) is_host_login,
22) is_guest_login, 23) count, 24) srv_count, 25) serror_rate,
26) srv_serror_rate, 27) rerror_rate, 28) srv_rerror_rate,
29) same_srv_rate, 30) diff_srv_rate, 31) srv_diff_host_rate,
32) dst_host_count, 33) dst_host_srv_count,
34) dst_host_same_srv_rate, 35) dst_host_diff_srv_rate,
36) dst_host_same_src_port_rate,
37) dst_host_srv_diff_host_rate, 38) dst_host_serror_rate,
39) dst_host_srv_serror_rate, 40) dst_host_rerror_rate,
41) dst_host_srv_rerror_rate, 42) attack_type.
```

For example, two records (# 1 and # 745) of File 1 are as follows:

```
0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,
0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,
0.00,0.00, normal.

184,tcp,telnet,SF,1511,2957,0,0,0,3,0,1,2,1,0,0,1,0,0,0,0,0,1,1,
0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,3,1.00,0.00,1.00,0.67,0.00,
0.00,0.00,0.00, buffer_overflow.
```

File 1.1 has also been prepared with the same 51608 records of the same format just without the last parameter 42) attack_type.

File 2 contains 311079 records of the same format as in File 1.1.

File 1.1 and File 2 are the test data files.

Note that KDD archive does not indicate the correct types of attack for none of the records of File 2. The only available information on possible attacks is gathered in Table 12.1 (column 'Code' is the emulator's code of attack). Nevertheless, File 2 has been used to test whether the emulator is able to detect unknown intrusions, which had not been presented in the training data of File 1.

The results of training the emulator by File 1 are shown in Fig. 12.2, where right-hand screen represents the initial population of cFIN after SVD ("Innate immunity

Table 12.1 Attack types

Code	Attack type	File 1	File 2	Code	Attack type	File 1	File 2
0	normal	+	+				
1	apache2		+	16	pod	+	+
2	back	+		17	portsweep	+	+
3	buffer_overflow	+	+	18	rootkit	+	
4	ftp_write			19	saint		+
5	guess_passwd		+	20	satan	+	
6	imap			21	sendmail		+
7	ipsweep	+	+	22	smurf	+	
8	land	+		23	snmpgetattack		+
9	loadmodule			24	spy		
10	multihop		+	25	teardrop	+	
11	named		+	26	udpstorm		+
12	Neptune	+		27	warezclient		
13	nmap			28	warezmaster		
14	perl			29	xlock		+
15	phf	+	+	30	xsnoop		+

of cFIN": $|W_0| = 51608$), while left-hand screen shows cFIN after apoptosis and autoimmunization ("Inner invariant of cFIN": $|W_1| = 783$). Total training time (for AMD 1.5 GHz) is 62 seconds, including 8 s for the 1st stage (SVD) and 54 s for the 2nd stage (apoptosis and autoimmunization).

During the recognition of the records of File 1.1 and File 2, the emulator writes test results into the output file in the format: Record # – attack_type. For example, four records (## 744–747) with test results for File 1.1 are as follows (see also Table 12.2):

```
744 - normal.
745 - buffer_overflow. !!!
746 - buffer_overflow. !!!
747 - normal.
```

The emulator also shows on-line projection of any pattern to 3D space of cFIN (see bold skew cross in both screens) and write the recognition result on the bottom panel (see "Class: back !!!").

Test results in Table 12.2 correspond completely to the correct attack types (parameter 42) of File 1.

Another test has been performed over File 2 to check whether the emulator is able to detect unknown intrusions, which had not been presented in the training data of File 1. The intrusion is treated as unknown if the projection of corresponding pattern to cFIN lies outside of the unit cube (according to Definition 1). The emulator has recognized 13 unknown intrusions as the following records ## of File 2:

Table 12.2 Test results for File 1.1

Records ##	attack_type	Records ##	attack_type
745–746	Buffer_overflow	38036–38051	ipsweep
3095–7373	Smurf	38052–38151	back
9520–9523	Buffer_overflow	38302–38311	ipsweep
9590–9591	rootkit	42498–42519	ipsweep
9928–10007	neptune	42548–42567	ipsweep
10072	satan	42593–42594	ipsweep
10320	phf	42706–42708	ipsweep
13340–13519	portsweep	42730–42761	ipsweep
13569	land	42762–42770	buffer_overflow
13845–13864	pod	42771–42772	land
16326–16327	pod	42773–43385	neptune
17446–37902	neptune	44451–44470	neptune
37929–37939	ipsweep	44800–48452	smurf
37959–37963	ipsweep	48453–48552	teadrop
38005–38012	ipsweep	All other	normal

```
417, 12674, 97891, 139795, 170498, 176201, 177958, 232570,
236975, 296561, 296657, 96796, 297658.
```

According to Table 12.1, any unknown intrusion can correspond to one of the following types of attack that had not been presented in the training data:

```
apache2, guess_passwd, multihop, named, saint, sendmail,
snmpgetattack, udpstorm, xlock, xsnoop.
```

The recognition time per record is 15.7 ms for both tests of File 1.1 and File 2. This time includes not only computations but mainly reading the record from test file, visualization of the recognition result (projection of the pattern to cFIN) in both screens of the emulator and writing the result into output file.

12.6 Forecast of Hydro-Physical Fields in the Ocean

Actually, this section proposes a new method of identification of cellular automata (CA), using the IC approach to pattern recognition (described in Sect. 12.4). The essence of the method is the representation of states and transitions of CA by using FIN and the faultless reducing of number of the transitions by apoptosis and autoimmunization. The task is formulated using an analogy to the computation of the ecological atlas (Tarakanov and Tarakanov 2004) as well as the reconstruction of the hydro-physical field (Tarakanov et al. 2007). This approach is compared with the existing methods of neurocomputing in computational intelligence and the more conventional interpolation by the least square method (LSM). Numerical example utilizes real-world atlas of the sea surface temperature (SST) from NOAA (1998).

According to Adamatzky (1994), identification of CA solves the problem how to learn the local behavior of cells from temporal slices of global evolution.

Let $c_{i,j,k} \in C$ be set of cells of CA with coordinates determined by indices i, j, k. Let $u(c) \subset C$ be set of the nearest neighbors defined for any cell. Let c^t be state of cell at discrete time step $t \in N$. Let us form set of parameters (state vector) for any cell $X = [x_1 \ldots x_n]'$ which can include current and/or previous states of the cell $c^t, c^{t-1}, \ldots, c^{t-p}$ and/or states of its nearest neighbors: $u^t, u^{t-1}, \ldots, u^{t-p}$. Note that values of some parameters can be unknown for some cells and/or time steps.

Let values of transition function $c^{t+1} = f(c^t, u^t)$ be given for some subset of cells $C_0 \subset C$ and time steps $N_0 \subset N$. The task is to determine the state of any cell at any time. Note that in real-world applications, the parameters $x_1, \ldots, x_n$ and the function f can be real-valued, whereas functional dependence $f(x_1, \ldots, x_n)$ can be too complicated or even unknown.

According to Sect. 12.4, consider a special case of the general IC algorithm, which solves the task of identification of CA by given training vectors $X_1, \ldots, X_m$ and corresponding values of the transition function $f_1, \ldots, f_m$.

1. Form training matrix $A = [X_1 \ldots X_m]'$ of dimension $m \times n$.
2. Compute first q singular values and corresponding left and right singular vectors by SVD of the training matrix.
3. For any training vector X_i, compute its mapping $Y(X_i) = [y_{i1} \ldots y_{iq}]'$ to q-dimensional space of FIN.
4. Using apoptosis and autoimmunization, reduce m training points of FIN to $k \leq m$ points.
5. Consider k points of FIN $Y_1, \ldots, Y_k$ together with their classes $f(Y_1), \ldots, f(Y_k)$ as the identified CA.
6. For any n-dimensional vector Z, compute its mapping $Y(Z) = [y_1 \ldots y_q]'$ to q-dimensional space of FIN.
7. Among the reduced training points $Y_1, \ldots, Y_k$, determine the nearest one to $Y(Z)$:

$$Y_{i^*} = \min_{i=1}^{k} \left\| Y_i - Y(Z) \right\|.$$

8. Assign class of the nearest point Y_{i^*} to the vector Z:

$$f(Z) = f(Y_{i^*}).$$

Therefore, Steps 1–5 identify CA, whereas Steps 6–8 compute the value of the transition function $c^{t+1} = f(c^t, u^t)$ for any cell of CA at any time step. Note also that the existence, invariance and faultlessness of such identification are guaranteed by the propositions and their proofs in Sect. 12.3.

Real-world data for the following numerical example have been obtained from the computer atlas of the Barents Sea (NOAA 1998).

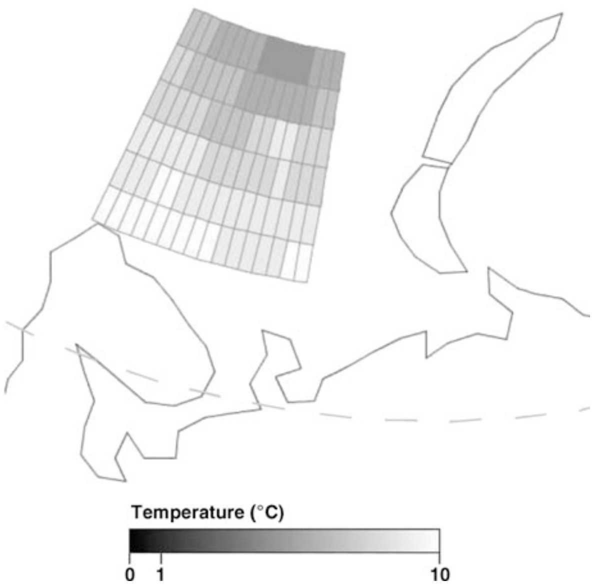


Fig. 12.3 A fragment of CA for the forecast of SST of the Barents Sea

Consider 2D lattice $c_{ij}, i = 1, \dots 6, j = 1, \dots, 16$, which is determined by the northern latitude $B_i = 75° - (i-1)$ and the eastern longitude $L_j = 30° + (j-1)$, where c^t is the monthly average SST in centigrade degree (T °C) which is given for all months: $t = 1, \dots, 12$. Let us identify such CA (CA-SST) that forecasts the field of SST c^t_{ij} for any $t = n \mod 12$, $n \in N$. For example, a screen-shot of the SST field for August ($n = 8, 20, 32, \dots$) is shown in Fig. 12.3.

Let us define state vector $X = [x_1 \dots x_n]'$ and transition function f for any cell so that $f = c^{t+1} - c^t$, $x_1 = c^t - c^{t-1}, \dots, x_n = c^{t-(n-1)} - c^{t-n}$. Thus, the behavior of CA-SST can be completely described by $6 \times 16 \times 12$ training vectors $X_1, \dots, X_{1152}$ and corresponding values of the transition function $f_1, \dots, f_{1152}$. However, such CA may be non-deterministic since different values of the transition function $f_{i1} \neq f_{i2}$ may correspond to identical vectors $X_{i1} = X_{i2}$. The IC algorithm can identify such conflicting transitions of CA by using only the 1st stage training Steps 1–3 (without apoptosis and autoimmunization) and the testing of the identified CA by Steps 6–8. Then any mistake reveals the conflicting rules of CA. The numbers of such conflicts against the memory size of the CA-SST are shown in Table 12.3.

Thus, the deterministic CA-SST is possible only for $n \geq 11$ and the IC algorithm with apoptosis and autoimmunization (Steps 1–5) identifies such CA-SST by reducing 1152 transition rules in 11-dimensional space X to 646 points of 3D FIN ($q = 3$).

Table 12.3 Number of conflicts in CA-SST identified by FIN

Memory size of CA (n)	2	3	4	5	6	7	8	9	10	11
Number of conflicts	761	569	405	323	238	160	91	61	33	0

After the identification (of the minimal memory size) of the deterministic CA-SST, artificial neural network (ANN) and LSM can be utilized for the comparison with FIN.

ANN has been taken from Tarakanov and Tarakanov (2004). Its output f is computed by the following formulas (so called feedforward run):

$$Y = \sigma(WX_b), \qquad f = c_f \sigma\big(V'Y_b\big),$$

where σ is the function of activation of neurons (so called sigmoid):

$$\sigma(x) = \frac{2}{1+\exp(-x)} - 1;$$

X_b and Y_b are vectors of input and hidden neurons with the bias $b = -1$:

$$X_b = \frac{1}{c_X}[x_1 \quad \dots \quad x_n \quad -c_X]', \qquad Y_b = [Y \quad -1]';$$

$c_f = |f_{\max}|$ and $c_X = |x_{\max}|$ are scaling coefficients which provide compatibility with the diapason of the sigmoid; W is weight matrix of input neurons and V is weight vector of hidden neurons. Both weight matrix and weight vector are trained by error back propagation (EBP) method. Training cycle of ANN consists of m runs to consider all training vectors $X_1, \dots, X_m$, whereas any run for training vector X_i consists of a) the feedforward run and b) the EBP correction of weights. Error of the output neuron after the feedforward run is calculated by the following formula:

$$d_0 = \big(f_i^* - f_i\big)\big(1 - f_i^2\big),$$

where i is number of the training vector, f_i is value of training function computed by ANN and f_i^* is given value of training function. Error of k-th neuron of hidden layer is calculated by the following formula:

$$d_k = d_0\big(1 - y_k^2\big)v_k,$$

where y_k is output of the neuron, v_k is value of k-th component of weight vector V, $k = 1, \dots, n_k$, and n_k is number of hidden neurons. Weight vector and weight matrix are corrected by the following (gradient) formulas:

$$\Delta V = \eta d_0 Y_b, \qquad \Delta w_{ji} = \eta d_k x_{ij},$$

where η is so called learning constant and x_{ij} is value of j-th component of input vector X_i. Training cycles are repeated until total (training) error becomes lower than some given value:

$$\frac{1}{m}\sum_{i=1}^{m}\big(f_i^* - f_i\big)^2 < e_0.$$

Table 12.4 MSE of identification of CA

Type of CA	FIN	ANN	LSM
Chemical waves	0.00	0.69	0.82
Solitons	0.00	0.26	0.38
SST of the Barents Sea	0.00	0.05	0.90

LSM calculates output $f = CX$ by the following vector of coefficients:

$$C = A^{+}F,$$

where $F = [f_1 \dots f_m]'$ is vector of training values of function f, whereas A^{+} is the so called pseudoinverse of training matrix:

$$A^{+} = \left(A'A\right)^{-1}A'.$$

Comparative accuracy of three different methods (FIN, ANN, LSM) is presented in Table 12.4 based on the mean square error (MSE):

$$e = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(f_i - f_i^{*}\right)^{2}},$$

where $k = 1152$ for CA-SST.

For more representatives, a couple of other CA (so called "chemical waves" and "solitons") has been identified by FIN using the data from Adamatzky (2001). The comparison of accuracy of FIN with ANN and LSM is provided in Table 12.4.

Note that neither ANN nor LSM is able to identify the conflicting states of CA.

12.7 Discussion

According to the obtained results for intrusion detection (Sect. 12.5), FIN reduces the storing patterns by 65.9 times using apoptosis and autoimmunization without any loss of accuracy of recognition. Although this increases the training time (from 8 seconds to 1 minute for AMD 1.5 GHz), nevertheless, more important is the decrease of the recognition time at least by 60 times per pattern by decreasing number of the stored cells of FIN to be compared with recognizing pattern.

It is worth noting that such a good performance of FIN (error-free recognition with rather low training time) on the data of real-life dimension looks unobtainable for main competitors in the field of computational intelligence like ANN and genetic algorithms (GA). According to our previous comparisons in Tarakanov and Tarakanov (2004, 2005), FIN trains at least 40 times faster and recognizes correctly more than twice as often as ANN and GA in the tasks of environmental monitoring and laser physics. These applications demonstrates not only the error-free recognition of the training set (like in Sects. 12.5, 12.6) but mainly the advanced accuracy of FIN on the test set, which may differ strongly from the training one. These tasks have rather low dimensions: $17\times23\times6$ for ecological atlas and 19×5 for laser diode. The drawbacks of ANN and GA become especially inadmissible for the task of intrusion detection with rather high dimension 51608×41 and more.

It is also worth noting that FIN differs essentially from the negative selection algorithm (NSA) widely used in the field of AIS (Dasgupta 1999; de Castro and Timmis 2002). Actually, NSA aims to provide a set of detectors for self-nonself discrimination, whereas FIN guarantees a minimal set of "cells" for the correct recognition of any number of classes based on "cytokines". Apparently, this makes FIN advantageous not only for the intrusion detection on-line (Tarakanov et al. 2005b) but also for medical oriented applications to simulate cancer specific apoptosis (Goncharova et al. 2005). Moreover, cytokines modulate proliferation and apoptosis of thymic cells as well as intrathymic T cell differentiation that includes not only negative but also positive selection (Savino and Dardenne 2000). Therefore, FIN also seems to be better suited for such kind of simulations.

Obtained results (in Sect. 12.6) also show a clear advantage of the use of FIN for identification of CA over both neurocomputing and the more conventional method of interpolation. These results confirm the advantages of an IC approach over other methods revealed by its application to the reconstruction of hydro-physical fields (Tarakanov et al. 2007). These advantages are expected to rise drastically with the rise of the dimension of training data.

We also point out that any run of IC with fixed dimension of FIN q and fixed number of nearest points of FIN for interpolation p gives the same MSE. Thus, FIN needs only $q \times p$ runs to determine the optimal parameters q^*, p^*, which provide the minimal error for any specific application, where $q \le r$, $p \le r$, while r is rank of the training matrix. Contrary to this fact, different training runs of ANN with fixed number of hidden neurons n_k, learning constant η and training error e_0 usually give different MSE. This makes ANN somewhat unpredictable and dictates its statistical characterization.

Moreover, ANN is too slow in comparison with FIN and conventional interpolation. For example (Tarakanov et al. 2007), IC needs just 21 runs (about 20 seconds) to obtain optimal parameters of FIN ($q^* = 3$, $p^* = 5$), whereas ANN needs 1750 runs (about 24 hours!) for the same purpose ($n_k^* = 3$, $\eta^* = 0.01$) still without any guarantee that, say, 20 or 100 hidden neurons may not minimize total error.

Table 12.4 shows better accuracy of FIN over ANN and LSM. Similar results in Tarakanov et al. (2007) also demonstrate the theoretically rigorous feature of training FIN with zero error rate. Just the opposite, training errors of ANN and LSM are usually too high. In addition, attempts to reduce training errors of ANN may lead to the so called overtraining effect when total error increases drastically.

The memory constraints of FIN and ANN look more comparable. For FIN, they are determined mainly by the dimensions of training matrix ($m \times n$) and FIN ($m \times q$), i.e. by the number and the dimension of training vectors and the dimension of space of FIN. The memory constrains of ANN are not much lower and determined mainly by the dimension of weight matrix ($n_k \times n$), i.e. by the number of hidden neurons and also by the dimension of training vectors. However, it is no need to store the training matrix after FIN has been trained. So, the memory constraints of IC and ANN can be compared by the dimensions of FIN and weight matrix, correspondingly.

As a conclusion, the obtained results confirm similar advantages of FIN over other methods of computational intelligence and more conventional methods of interpolation revealed by their applications to real world data of information assurance, ecology, laser physics, and hydro-acoustics. Possible ways to reinforce these advantages may be norms other than Euclidean together with more delicate methods of interpolation by nearest points of FIN. The advantages of the proposed approach coupled with its advantages for 3D modeling (Tarakanov and Adamatzky 2002) make FIN rather promising for on-line simulation of real-world 3D fields.

12.8 Epilogue

After the 1st edition of this Chapter (Tarakanov 2008), the described approach has been generalized as a new way to intelligent signal processing. The results of numerical experiments reported in Tarakanov (2008) suggest that the speed and accuracy of the approach is probably unobtainable for other robust methods of computational intelligence (in particular, neurocomputing and evolutionary algorithms). These advances of the approach together with its biological nature probably indicate a further step toward placing more of the intelligent functions on the chip.

It is also worth highlighting that the approach appears to be useful in brain research, especially for discovering deep biomolecular similarities between marine sponges, human brain, and immune system. Namely, a set of triplet homologies of amino acid residues has been deduced that may be responsible for receptor-receptor interactions (Tarakanov and Fuxe 2010). Based on the same mathematical approach, main triplets in cell-adhesion receptors of marine sponges have been discovered, which appear also as homologies in several receptor heteromers of human brain (Tarakanov et al. 2012). It has been also demonstrated their relevance to protein-protein interactions and mentioned possible implications for novel pharmacological targets and strategies for treatment of diseases, e.g. neuroinflammatory diseases.

Last but not least, the approach to forecast of hydrophysical fields (Tarakanov 2009) "might not seem to harbor an artistic statement, but a novel application of data simulation has led to a unique merger of science and art" (NASA 2011). Namely, a mathematical model of global dynamics of sea surface temperature (SST) has been developed utilizing data of NASA. The model provides fast computing and visualization of daily SST of any area in the World Ocean (sea, lake). The special models of the Caspian, Black, Barents, Mediterranean, and Baltic Seas as well as the Gulf of Mexico and Ladoga Lake have been created. The animated results can be viewed in corresponding YouTube videos (e.g., the YouTube video address of the Baltic Sea SST is http://www.youtube.com/watch?v=JIng8MAXTsQ).

References

Adamatzky, A. (1994). *Identification of cellular automata*. London: Taylor & Francis.

Adamatzky, A. (2001). *Computing in nonlinear media and automata collectives*. Bristol: IOP

Ader, R., Felten, D. L., & Cohen, N. (Eds.) (2001). *Psychoneuroimmunology*. New York: Academic Press.

Agnati, L. F., Tarakanov, A. O., Ferre, S., Fuxe, K., & Guidolin, D. (2005a). Receptor-receptor interactions, receptor mosaics, and basic principles of molecular network organization: possible implication for drug development. *Journal of Molecular Neuroscience*, *26*(2–3), 193–208.

Agnati, L. F., Tarakanov, A. O., & Guidolin, D. (2005b). A simple mathematical model of cooperativity in receptor mosaics based on the "symmetry rule". *Biosystems*, *80*(2), 165–173.

Balkwill, F. (Ed.) (2000). *The cytokine network*. New York: Oxford University Press.

Bay, S. D. (1999). *The UCI KDD archive*. Irvine: University of California, Dept. of Information and Computer Science. Available at http://kdd.ics.uci.edu.

Bunk, S. (2003). Signal blues: stress and cytokine levels underpin a provocative theory of depression. *The Scientist*, *25*, 24–28.

Cloete, I. & Zurada, J. M. (Eds.) (2000). *Knowledge-based neurocomputing*. Cambridge: MIT Press.

Dasgupta, D. (Ed.) (1999). *Artificial immune systems and their applications*. Berlin: Springer.

de Boer, R. J., Segel, L. A., & Perelson, A. S. (1992). Pattern formation in one and two-dimensional shape space models of the immune system. *Journal of Theoretical Biology*, *155*, 295–333.

de Castro, L. N., & Timmis, J. (2002). *Artificial immune systems: a new computational intelligence approach*. London: Springer.

Fisher, P. B., et al. (2003). mda-7/IL-24, a novel cancer selective apoptosis inducing cytokine gene: from the laboratory into the clinic. *Cancer Biology & Therapy*, *2*, S023–S037.

Goncharova, L. B., Jacques, Y., Martin-Vide, C., Tarakanov, A. O., & Timmis, J. I. (2005). Biomolecular immune-computer: theoretical basis and experimental simulator. In *Lecture notes in computer science* (Vol. 3627, pp. 72–85). Berlin: Springer.

Horn, R., & Johnson, Ch. (1986). *Matrix analysis*. Cambridge: Cambridge University Press.

Igney, F. H., & Krammer, P. H. (2002). Immune escape of tumors: apoptosis resistance and tumor counterattack. *Journal of Leukocyte Biology*, *71*(6), 907–920.

Jerne, N. K. (1973). The immune system. *Scientific American*, *229*(1), 52–60.

Jerne, N. K. (1974). Toward a network theory of the immune system. *Annals of Immunology*, *125C*, 373–389.

Kourilsky, P., & Truffa-Bachi, P. (2001). Cytokine fields and the polarization of the immune response. *Trends in Immunology*, *22*, 502–509.

Kurzrock, R. (2001). Cytokine deregulation in cancer. *Biomedicine & Pharmacotherapy*, *55*(9–10), 543–547.

NASA (2004). Human immune system inspires NASA machine-software fault detector. *NASA Bulletin*, 26 October.

NASA (2011). Russian scientist creates simulation of daily sea surface temperatures in the Caspian Sea. *NASA Goddard Earth Sciences Data and Information Services Center News*, 14 January 2011. Available at http://disc.sci.gsfc.nasa.gov/giovanni/gesNews/caspian_sea_sst_animation.

NOAA-NESDIS-National Oceanographic Data Center (1998). *Climatic atlas of the Barents Sea*. Available at http://www.nodc.noaa.gov/OC5/barsea/barindex.html.

Savino, W., & Dardenne, M. (2000). Neuroendocrine control of thymus physiology. *Endocrine Reviews*, *21*(4), 412–443.

Tarakanov, A. O. (2008). Formal immune networks: self-organization and real-world applications. In M. Prokopenko (Ed.), *Advances in applied self-organizing systems* (1st ed.). London: Springer.

Tarakanov, A. O. (2008). Immunocomputing for intelligent intrusion detection. *IEEE Computational Intelligence Magazine*, *3*(2), 22–30.

Tarakanov, A. O. (2009). Immunocomputing for geoinformation fusion and forecast. In *Lecture notes in geoinformation and cartography* (Vol. XIII, pp. 125–134). Berlin: Springer.

Tarakanov, A., & Adamatzky, A. (2002). Virtual clothing in hybrid cellular automata. *Kybernetes*, *31*(7–8), 394–405.

Tarakanov, A., & Dasgupta, D. (2000). A formal model of an artificial immune system. *Biosystems*, *55*(1–3), 151–158.

Tarakanov, A., & Dasgupta, D. (2002). An immunochip architecture and its emulation. In *NASA/DoD conference on evolvable hardware (EH'02)* (pp. 261–265). Los Alamitos: IEEE.

Tarakanov, A. O., & Fuxe, K. G. (2010). Triplet puzzle: homologies of receptor heteromers. *Journal of Molecular Neuroscience*, *41*(2), 294–303.

Tarakanov, A. O., & Tarakanov, Y. A. (2004). A comparison of immune and neural computing for two real-life tasks of pattern recognition. In *Lecture notes in computer science* (Vol. 3239, pp. 236–249). Berlin: Springer.

Tarakanov, A. O., & Tarakanov, Y. A. (2005). A comparison of immune and genetic algorithms for two real-life tasks of pattern recognition. *International Journal of Unconventional Computing*, *1*(4), 357–374.

Tarakanov, A. O., Skormin, V. A., & Sokolova, S. P. (2003). *Immunocomputing: principles and applications*. New York: Springer.

Tarakanov, A. O., Goncharova, L. B., & Tarakanov, O. A. (2005a). A cytokine formal immune network. In *Lecture notes in artificial intelligence* (Vol. 3630, pp. 510–519). Berlin: Springer.

Tarakanov, A. O., Kvachev, S. V., & Sukhorukov, A. V. (2005b). A formal immune network and its implementation for on-line intrusion detection. In *Lecture notes in computer science* (Vol. 3685, pp. 394–405). Berlin: Springer.

Tarakanov, A., Prokaev, A., & Varnavskikh, E. (2007). Immunocomputing of hydroacoustic fields. *International Journal of Unconventional Computing*, *3*(2), 123–133.

Tarakanov, A. O., Fuxe, K. G., & Borroto-Escuela, D. O. (2012). Integrin triplets of marine sponges in human brain receptor heteromers. *Journal of Molecular Neuroscience*, *48*(1), 154–160.

Vilcek, J., & Feldman, M. (2004). Historical review: cytokines as therapeutics and targets of therapeutics. *Trends in Pharmacological Sciences*, *25*, 201–209.

Wall, L., Burke, F., Caroline, B., Smyth, J., & Balkwill, F. (2003). IFN-gamma induces apoptosis in ovarian cancer cells in vivo and in vitro. *Clinical Cancer Research*, *9*, 2487–2496.

Chapter 13
A Model for Self-Organizing Data Visualization Using Decentralized Multi-Agent Systems

Andrew Vande Moere

13.1 Introduction

In the information society of today, corporations, government agencies and various scientific fields are continuously accumulating data. The complexity of this data is staggering, and our ability to collect data is increasing faster than our ability to analyze it. Although current database technology has made it possible to store and manage huge amounts of data in a comprehensive manner, the exploration of this data is still bound to relatively rigid interfacing methods. *Visualization*, the representation of data graphically rather than textually, aims to exploit the high-bandwidth human perceptual and cognitive capabilities to draw interferences from visual form. Its real purpose goes beyond that of generating beautiful pictures, as visualization aims to induce useful insights where there was none before. Such insights can take different forms, such as through the discovery of unforeseen data phenomena, the ability to derive decisions or to visually communicate knowledge and insights based on discoveries made within the dataset. More particularly, the field of *data visualization* faces the need to represent the structure of and the relationships within large, complex datasets that contain so-called 'abstract' concepts, which lack a natural notion of position in space (Card et al. 1999; Chi 2000; Tory and Möller 2004). Data visualization therefore differs from *scientific visualization*, which generally represents datasets that have a physical form in nature and generally can be represented through a process of graphical reproduction, such as geographical layouts, wind simulations or medical imaging. Because abstract data is non-spatial and lacks any natural representation, the fundamental challenge for data visualization is thus "… how to invent new visual metaphors for presenting information and developing ways to manipulate these metaphors to make sense of the information" (Eick 2001). Data visualization is different from *data mining*, which deals with the analysis of datasets to establish relationships and identify patterns, so that data items are

A. Vande Moere (✉)
Faculty of Architecture, Design and Planning, The University of Sydney, Sydney, NSW, Australia
e-mail: andrew@arch.usyd.edu.au

clustered into groups according to apparent data similarity. Although data visualization predominantly aims to convey meaning and insights, it often uses data mining techniques to analyze or order the dataset first. As datasets continuously become more complex in terms of size, time-variance and data dimensionality, data visualization techniques need to evolve to accommodate for more sophisticated ways of data analysis and visual representation.

Self-organization is a process in which the internal structure of a complex system automatically increases without being guided by an outside source. Self-organizing processes are often coupled to the occurrence of *emergent* phenomena. An emergent property is generally characterized when perceived complex behavior arises from a collective of entities that individually where not 'explicitly' programmed to do so. Emergent behavior is generally generated by the continuous and recursive interaction of individual entities with similar entities in their immediate environments. When well-considered interaction mechanisms are used, these interactions can, on a holistic level, lead to seemingly rational behavior and the observable proliferation of order. The probably best known examples of such emergent behaviors are the flocking of birds and schooling of fish, or the amazing organizational capabilities (e.g. shortest way finding, waste disposal) of ant colonies. Principles of emergence generation have been successfully applied to a wide set of disciplines, including problem solving, learning and optimization problems in the fields of system design, pattern recognition, biological system modeling, signal processing, decision making, simulation and identification (Kennedy and Eberhart 2001). Such systems are able to successfully solve complex problems that contain multiple unpredictable or time-varying parameters. Self-organization intelligence is basically divided and distributed to simple and comprehensible units, which holistically are tolerant to local failures or changing environments. Self-organization seems thus to be an ideal method to be applied to data visualization, as this field often involves multifaceted and inherently unpredictable datasets that need to be 'ordered' into apparently well-ordered diagrams. By applying self-organization to the problem of data visualization, data mapping metaphors might emerge that suit particular dataset characteristics, hereby potentially revealing data patterns that were unknown before.

Self-organizing data visualization is based on augmenting direct data mapping techniques with simple forms of artificial intelligence, assuming that datasets inherently contain sufficient characteristics to organize themselves. It aims to instigate visual emergent phenomena from meaningful data relationships inherently present in the dataset. It is based on the conception that each single data item within a dataset can be mapped onto a unique *data visualization agent*. Each agent then determines its own visual properties through a process of continuous and recursive interactions with other agents. The iterative organization of the agents creates emergent visual effects that are based on data properties that are autonomously detected by agents, and can be perceived and interpreted by users. Ultimately, this approach might lead to new data visualization techniques that are more 'aware' of the data characteristics they represent, and can optimize the overall representation according to good visualization guidelines and human visual cognitive knowledge.

This chapter aims to demonstrate the potential of self-organization for data visualization purposes. At the same time, it also illustrates how self-organization probably is not the most efficient existing method to visually represent data, due to the required computation power and the considerable amount of efforts needed in managing the various parameters that control the emergent phenomena. As will be discussed in Sect. 13.5, self-organizing visualization suffers from inadequate calculation performance, comprehensibility difficulties, and the influence of multiple parameters on the emergent effects. However, self-organizing data visualization still forms a valuable alternative approach to the predetermined and fixed data mapping techniques that currently exist, as it specifically allows unexpected visual organizations to occur.

In this chapter, the principles of the self-organizing data visualization model are described and illustrated with three different case studies. The 'information particle', 'information flocking' and 'cellular ant' approaches each use a different visual metaphor, respectively based on particle animation, swarming and cellular automata principles. Each system demonstrates how meaningful properties contained within datasets can be made apparent by the simulation of visually emergent effects, using relatively simple self-organizing behavior rules. Based on these findings, Sect. 13.5 discusses the potential benefits and shortcomings of this concept.

13.2 Background

Self-organizing data visualization is fundamentally different from most other agent-based approaches known in the fields of visualization or data mining, which tend to focus on using agent intelligence for data analysis, visualization dataflow or software development optimization.

13.2.1 Decentralized Multi-Agent Systems

An *agent* is a system situated within an environment. It senses that environment and acts on it autonomously, over time, to realize a set of design objectives (Franklin and Graesser 1996). An *intelligent agent* typically can be characterized as social, reactive and proactive, as it operates in a changing, unpredictable environment where there is a possibility that actions can fail (Woolridge 2001). Such agents can collaborate with other agents, can perceive and respond to changes, and can exhibit goal-directed behavior. A *multi-agent system* is populated with multiple equal agents, generally because they pursue different goals, or because the environment is too complex for a single agent to observe completely. The agents within such system interact and negotiate with each other, a process which is generally determined by concepts of cooperation, coordination and negotiation. An agent's behavior can be determined by a *rule-based system* that interprets communications and interactions

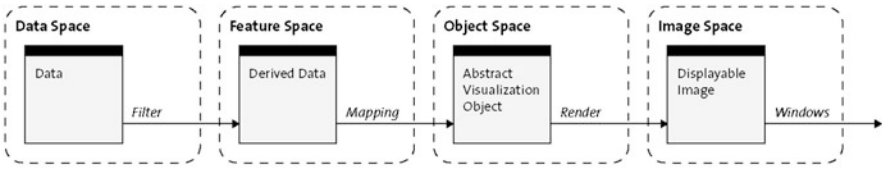


Fig. 13.1 Typical information visualization dataflow model (after Upson et al. 1989)

with other agents, the perceived environmental changes and the agent's goals. A *decentralized multi-agent system* contains numerous equal agents that have communication links with those in their neighborhood, either directly or through the environment, but always in absence of a centralized coordinator. Such systems are designed to facilitate *self-organization*, the spontaneous increase in complexity of internally organized structures.

Accordingly, self-organizing data visualization is based on decentralized multi-agent systems that are inherently capable of simulating collective behavior, such as swarming, cellular automata and particle animations. By controlling these agents with data values, the resulting behaviors are fully controlled by dataset properties. The resulting visual 'effects' are presented in an organized and consistent manner, so they aim to be perceived and interpreted efficiently.

13.2.2 Data Visualization

Data visualization is based on the principle that meaningful data properties, such as tendencies, trends, outliers and similarities between data items, can be made apparent by translating raw textual or numerical data values into a holistically interpretable visual representation. As shown in Fig. 13.1, the typical data flow process for translating data into visual form is considered to be an iterative analysis cycle that passes through four distinct phases (Upson et al. 1989). First, a large amount of 'raw' data is filtered into manageable and interpretable data subsets in the Data Space domain. The filtering of these subsets is determined by dataset quality and user interests. First, corrupt or otherwise invalid data items are removed from the dataset, which might have resulted from faulty parsing or querying procedures. Then, the system caches only those data items that are relevant for a particular user, for instance those particular data items that are visible, or are selected by settings in the user interface. The derived data subsets in Feature Space are then translated into visual distinguishable forms in Object Space, as specifically designed 'data mapping rules' generate visual artifacts that are consistent with the values and attributes within the dataset. In general, specific data attributes relate to visual objects (e.g. points, lines, shapes), whereas their values determine the transformations applied to those forms (e.g. position, color, direction, size). As each visual object stands for a unique data item, the resulting visual constellation of objects is representative for the whole dataset. Finally, the resulting collection of visual objects, labels and legends needs to be rendered into a coherent perceivable form in Image Space, as different forms of media require specific treatments and formats, depending on aspects such as the display size, distributed computation resources, user context or hardware capabilities. As will be described in the next section, agent-based systems have been successfully used in each of these visualization dataflow stages.

13.2.3 Agent-Based Visualization Approaches

Applied artificial intelligence and the visualization domain have been successfully combined for various purposes. However, most past work has been focused on either the filtering (Feature Space) or rendering (Image Space) phases of the typical visualization data flow. In contrast, the proposed model in this chapter uses agents for the mapping of derived data to visualization objects.

Agent-based data mining addresses the retrieval, management and extraction of information out of large, distributed databases. Data mining agents are capable of accessing distributed datasets from which useful, higher-level information is extracted. Such agents keep track of and organize information spaces by understanding meaningful relationships within datasets, and are able to present these proactively to users. Some agent-based data mining algorithms are based upon a centralized system component, such as middle agents or directory services, which know the location and capabilities of each agent, enabling each agent to communicate with all other agents. Complex problems, such as clustering, are optimized by reducing the solution space that the algorithm has to search, or by partitioning them into a series of smaller problems. In contrast, a decentralized multi-agent system contains numerous equal agents that have some communication links to other agents. The goal of such agents is to iteratively select and rearrange these links to form interactive connections, so that data clustering can be achieved in a collective effort (Ogston et al. 2003).

Most so-called *agent-based visualizations* consist of traditional representation techniques that utilize the agents as their dataset to be visualized. These agents are thus not designed to interfere with the visualization, and the resulting representations convey typical agent properties instead of characteristics of complex, abstract datasets. For instance, so-called *multi-agent visualizations* have been used to display intrinsic relations (e.g. number of messages, shared interests) between agents for monitoring and engineering purposes (Schroeder and Noy 2001). Visualization in data mining tends to be used to present final results, rather than playing an important part throughout the entire data exploration process (Robinson and Shapcott 2002). Multi-agent systems have been developed to organize the visualization dataflow, for instance for representing complex fuzzy systems (Pham and Brown 2003). Some visualization systems utilize agents to determine the most efficient display parameters within the Image Space. For instance, an advanced graphic visualization rendering pipeline can consist of several remotely dispersed agents to allow for the abstraction and reuse of rendering strategies, including distributed or progressive rendering (Roard and Jones 2006). Similarly, the e-Viz system is based on agents, so-called active visualization components, that are designed to self-heal software failures or network problems, and can self-optimize the rendering performance (Brodlie et al. 2006). Multi-agent systems are then used to support flexibility regarding the interaction possibilities or the rendering quality of the visualization (Ebert et al. 2001). Agents and visualizations have been combined to organize, analyze or mine abstract datasets, with agents embedded in both Data and Feature Space. Data can be ordered and filtered by so-called "visualization agents", which then becomes represented using conventional techniques, such as for discovering the most desired pages from a large web site (Hiraishi et al. 2001). Agents can help users to decide the most suitable visualization technique depending on the dataset characteristics (Marefat et al. 1997), guide users towards the most effective visualization approach according to dataset characteristics and visual cognitive guidelines (Senay and Ignatius 1994), or act as visualization assistants to help choose methods for effectively visualizing e-commerce data (Healey et al. 2001). Similarly, some visualizations approaches support developers to choose the most optimal association rules that allow for efficient data exploration (Sadok and Engelbert 2004).

Only few approaches exist that use multi-agent systems as a way to map data into visual form, or, with other words, that contain agents in Object Space that act as visual elements themselves. In 1996, Ishizaki proposed the idea of deriving design solutions by the emergent behavior of a collection of agents, so called *performers*, which are individually responsible for presenting a particular segment of information (Ishizaki 1996). Although this approach primarily focused on interactive and information-rich interfaces consisting of agents that represent news headlines, several conceptual similarities with our proposed data visualization model exist: the agents represent data by themselves, are not bound to particular types of visual expression, are governed by a set of behavior rules that are based on the detection of data characteristics, and even collaborate together to reach a common strategic goal. More recently, a similar collaborative multi-agent system has been developed that produces artistic Mondriaan-like paintings emergently (Mason et al. 2004). Other research approaches that are more specifically relevant to each of the case study approaches are mentioned in their respective sections.

13.3 Emergence in Data Visualization

The self-organizing data visualization model is based on the assumption that data items themselves can be treated as agents that, depending on their inherent relationships, are capable of autonomously determining their own visual representation. Accordingly, this section investigates the specific requirements to derive emergent visual effects out of data characteristics.

13.3.1 Visual Emergence Versus Data Pattern Emergence

Each data visualization technique is uniquely determined by its *data mapping rules*, a set of simple conditions that 'translate' data values into visual form from Feature Space to Object Space. Typically, each single data item within a dataset is mapped onto a separate *visual object*, such as a point, line, shape or three-dimensional object. The data variables of a data item then determine the *visual transformations* of this visual object, such as its position, color, shape, length, or orientation. Generally, these data mapping rules are implemented by the developer, who, at the very least, takes into consideration following aspects.

(i) **Anticipated dataset characteristics**, in order to specify the visual objects and their properties, and design the according data mapping rules. Several empirically derived guidelines exist that correlate effective visualization techniques according to dataset typology (Mackinlay 1986) (e.g. numerical, categorical, etc.). However, the data mapping choice is typically made by its developer. Even when a specific technique has been chosen, a visualization developer needs to fine-tune the data mapping rules according to the expected size, time-dependency, data dimensionality, time-variance, data value range and data pattern semantics of the dataset. For instance, anticipated maximum and minimum data values have to correspond to the specific axis scales or color scale, while the dataset update frequency determines to how fast the visualization should adapt to any dynamic changes.

(ii) **User Interests**. Similarly to the dataset characteristics, the data mapping rules should assign the most dominant and pre-attentive visual properties, such as color and position, to the data patterns a user is most interested in.

(iii) Aspects of human **visual perception and cognition**, so that data patterns are represented in an easily perceived and intuitively understandable way. For a data visualization to be effective, it has to translate data patterns into artifacts that visually stand out, while allowing the user to comprehend their meaning. In other words, the data mapping needs to allow for an 'inverse' mapping to occur rapidly and faultlessly by its users. Therefore, the data mapping design process needs to incorporate insights from different adjacent disciplines, such as perception in visualization (Ware 2000), visualization guidelines (Card et al. 1999), user interaction (Shneiderman 1998), data exploration (Jankun-Kelly et al. 2002), task-related and human factors (Tory and Möller 2004) and graphic design principles (Tufte 2001). As shown by insights from the Gestalt School of Psychology, humans attempt to perceive and understand any graphical representation as one, single, coherent form or Gestalt, instead of a collection of individual components. The Gestalt research aimed to describe the principles of perceptual visual processes that result in perceptual coherence, which was synthesized in a set of so-called Gestalt laws. Figure 13.2 illustrates how groups of nearby objects, or objects similar in color or shape, are perceived as belonging together. Conceptually, this principle implies a level of *emergence* occurring on a perceptual level, as visually complex patterns can be

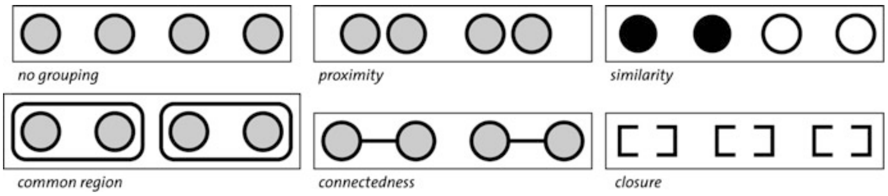


Fig. 13.2 Visual emergence as gestalt rules. Separate objects are perceived as belonging together due to specific common characteristics

recognized from a collective of entities that individually where not explicitly informed to do so.

The traditional design of data mapping rules is specifically motivated by the wish to highlight meaningful data phenomena by purposively simulating the occurrence of "visually emergent" effects. Traditional visualization approaches rely on the direct translation of similar data values to similar visual properties, so that, for instance, data items that are 'similar' tend to be represented nearby each other or are highlighted by an identical color. In contrast, a self-organizing system aims to achieve a visually similar effect by a decentralized approach, so that similar data items should first 'find' each other, and then 'stay close' together, or 'determine' a 'common' color. The detection of similar items, their grouping or their cooperative color choice can be considered emergent, as self-organization requires simple pairwise comparisons between data items instead of a centralized data analysis. Consequently, self-organizing data mapping is a process that aims to magnify 'dataset emergent' phenomena into 'visual emergent' effects. This dual reliance on emergence, first in the detection of data patterns and then in the deliberate generation of visual forms, is the main driving force behind the proposed self-organizing visualization model.

Technically, self-organizing data visualization is based on mapping meaningful data values with the numerical parameters that influence emergent behaviors of applied artificial intelligence simulations. As those weighting values are directly derived from the dataset, the resulting emergent behavior 'represents' that dataset. In contrast, traditional data visualization approaches are determined by rigid data mapping rules, predefined by the application developer. Any alterations to such data mapping rules tend to be strictly limited to the configuration of the visual object transformations, such as the application's color palette or the axes scales. Although most data visualization applications offer a set of interactive features facilitating typical "human-computer interaction mantra" operations (i.e. overview, zoom and select) (Shneiderman 1998), data mapping rules are generally considered to be a fixed part inherent to the visualization technique. In contrast, by merging the concept of emergence with data mapping, unforeseen behaviors might become apparent that convey useful data patterns in unexpected ways.

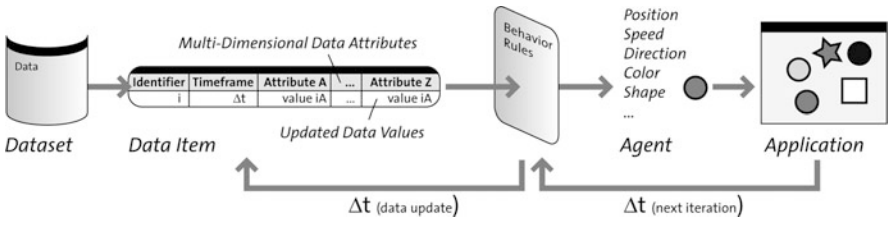


Fig. 13.3 A data visualization agent and its behavior as determined by data-driven behavior rules

13.3.2 Data Visualization Agent

The self-organizing data visualization concept is based on mapping data items directly onto agents. Each single data item (e.g. *data object, data tuple* or *database row,* retrieved from a database or dataset) is mapped onto a unique data visualization agent. The aim of this particular approach is to let agents "behave" according to their individual data values or according to any differences with the data values of their neighbors. As illustrated in Fig. 13.3, each agent is visually represented by a visual object, and its dynamic behavior (e.g. position, color, shape, speed) is determined by a set of behavior rules that in turn is controlled by its data item's data values. As time progresses, its data values change and the agent's dynamic behavior with adapt accordingly.

Traditional data visualization techniques are generally based on pre-analyzed datasets. For instance, some approaches are based on *data similarity tables*, which contain quantitative measurements of how pair-wise data items relate to each other. In contrast, self-organizing data visualization performs the dataset analysis during the visualization itself, where it is executed by a collection of agents instead of a single, central process. All agents exist in a shared visualization space, which is ruled by a common *application timeline*. All agents are 'situated', viewing the world from their own perspective rather than from a global one. Their actions are determined by internal states, as determined by a set of common behavior rules. These rules represent the strategy of the agent, and how it should behave according to its own data characteristics, those of neighboring agents, and to any external influences, such as real-time changes in the dataset, or user interactions (e.g. selection, filters).

For *static* datasets, each agent continuously represents the same set of data values. For *time-varying* (also called *dynamic, time-dependent, time-variant, time-based* or *temporal*) datasets, all agents are synced to the sequential progress of the application timeline, which moves 'forward' or 'backwards' in an iterative way. Each agent then is subject to a 'data update', which either corresponds to a 'data alteration' or no change at all. The application timeline progresses according to a specific rhythm, so that 'newly updated' data items are fetched from the database, and then assigned to the according agent. An updated agent will reconsider its visual transformations, and might change its dynamic behavior or representation cues accordingly. Because of the presence of multiple, equal agents that are not centrally controlled, the data visualization is essentially a decentralized multi-agent system.

In short, a typical data visualization agent needs to include at least following characteristics:

- **Data Interpretation.** Each agent is aware of all the data attributes and values of the data item it represents. It can also detect any changes that occur to it over time, caused by the application timeline simulation or by user interaction.
- **Local Perception.** The agent is capable of perceiving the environment it is present in, including any other agents in its vicinity and any external objects or space boundaries.
- **Local Communication.** Each agent can communicate with other agents nearby, and can compare all the attributes of those agents, including their data item or their actual visual state.
- **Negotiation.** More 'intelligent' agents can perform complex negotiations with neighboring agents, such as simple tit-for-that strategies (e.g. the shape of agent A grows while taking away the space required from the shape of its neighboring agent B) or positional swapping (e.g. agent A and B swap their position).
- **Visual Presence Autonomy.** Each agent is capable of autonomously determining its own visual presence, in form of altering the visual attributes it inherently possesses, including its spatial position, size, color, orientation, speed, direction, shape and so on. According to the 'negotiation' characteristic, agents can also change the visual properties of its neighboring agents to some degree.
- **Historical Memory.** Each agent can store and historical events and access them, such as the previous values that its data item has contained, the coordinates it has passed through, the visual attributes it has adapted to, and the other agents it has encountered over time.

More complex agent characteristics that would be useful for data visualization purposes can be easily imagined, such as learning, reasoning, motivation, curiousness and so on. This chapter instead will focus on the self-organizing and emergent capacities that can be achieved by using reasonably simple agent principles.

13.3.3 Behavior Rules

At initialization, all agents are positioned randomly on the visualization canvas. All agents are governed by the same set of behavior rules. These rules indirectly 'map' the agent's data values into visual properties. For instance, one ore more behavior rules could define an agent's color according to the relative difference in data values with one of its neighbors, or let it move towards the most similar agent. As mentioned in Sect. 13.3.1, these behavior rules need to be specifically designed to simulate visually emergent effects, so that similar data items can be effectively perceived as 'belonging together'. For instance, the Gestalt Rule of Proximity states that objects that are located nearby each other are emergently perceived as one. In traditional data visualization, this particular characteristic is exploited by specifically choosing the data attributes so that data items with similar values are positioned in the vicinity of each other. For the agent-based approach, a similar visual effect is achieved by a well-considered sequence of individual agent actions. Because of the multitude of such pair-wise agent interactions, multiple similar agents will group, and large spatial clusters will form. Although the end result of these different methods might seem similar at first sight, self-organizing data visualization is intrinsically dynamic, completely decentralized and determined in real time.

13.4 Case Studies

The following section describes three different approaches of the self-organizing data visualization model. From a simple set of cause-and-effect behavior rules to more elaborate and complex inter-agent negotiation strategies, the proposed techniques demonstrate how the mapping of data onto visual properties can be accomplished without the need for central control. The simulation of self-organizing behavior in each technique is based on insights borrowed from the fields of applied artificial intelligence and artificial life. The resulting emergent behavior can be interpreted as meaningful data trends and patterns, as the apparent effects are fully determined by, and thus inherently reflect, inherent dataset characteristics.

13.4.1 Metaphor 1: Infoticle Method

The *infoticle* system is based on a simple set of sequential behavior rules that determine the speed and direction of moving particles in a three-dimensional, virtual space. This particular approach demonstrates how the traditional vocabulary used by data visualization, consisting of position, color, shape, size, etc., can be effectively broadened with a novel visual property, that of *dynamic animation*. The power of animation to create interpretatively rich behaviors allows for the generation of distinctive motion typologies that convey time-varying data trends.

13.4.1.1 Self-Organizing Method: Particle Animation

A *particle* can be imagined as a point-like mathematical object in three-dimensional space. It is generally determined by a fixed set of attributes, such as position, velocity (speed and direction), acceleration, mass, color, lifespan, age, shape, size and transparency. A *particle system* consists of a collection of particles, each of which is influenced independently through time by a set of predefined conditions (Martin 1999). Particle systems were first formally proposed by Reeves (1983) as a rendering technique that is able to realistically simulate dynamic, natural phenomena such as rain, explosion, waterfalls, fire or smoke. Currently, particle systems are a widely used computer graphics technique, as the essential programming logic required to efficiently implement real-time particle systems have been described in detail (see

Lander 1998; van der Burg 2000 for applicable software programming approaches). Particle systems can be combined with so-called '*forces*', which are abstract, point-like elements in virtual space that influence the movement and speed of each single particle by attracting or repulsing it according to the laws of Newtonian mechanics. Given correct initial conditions, and combined with internal relationships or external forces, particle systems can be animated over time to convey seemingly complex and intriguing behaviors (Tonnesen 2001). For instance, the combination of particles and forces enable computer graphic designers to convey realistic visual effects, from simulating gravity in space galaxies over the bouncing of balls on surfaces to the fading of flames.

The ultimate goal of using dynamic animation as an independent visual property in data visualization is the creation of interpretatively rich behaviors that seem to be intentional, possibly even provoking causality, animacy and initiative. *Behavioral animation* techniques typically employ rule-based systems to specify the dynamic motions, so that a set of cause-and-effect rules is listed which the elements that are being animated must follow. For instance, Lethbridge and Ware (1990) used simple behavior functions based on distance, velocity and direction to model complex causal relationships. Empirical cognitive research suggests a huge potential of using motion for data visualization purposes (Ware et al. 1999). Bartram and Ware (2002) proved that motion typology has a strong perceptual grouping effect that can be effectively used for information display. For instance, similar motion typologies are perceived as grouped over time, a phenomenon also called *temporal grouping*, which relates to the appearance and locations of the elements, the proximity in time and the similarity of motions (Kramer and Yantis 1997). The use of motion for representing information might seem to contradict with traditional mapping techniques that position data items on 'fixed' Euclidian coordinates. Instead, motion should be considered as a property that is independent of position, and instead relies on the perceived dynamic relationships between the various elements that move similarly or differently. Instead of 'morphing' visual elements from one fixed Euclidian position to another, behaviorally moving particles have the inherent capability to be updated on-the-fly.

13.4.1.2 Particle Animation as Data Mapping

In the infoticle system, each data visualization agent is represented as a unique particle within a three-dimensional virtual space. This particle is coined 'infoticle' (short for '*information-particle*'), and will be further referred to as 'agent'. The virtual world also contains a set of 'forces', which are fixed points in space that represent specific static data attribute values that an agent's data item potentially can contain. The dynamic behavior of each agent is determined by a set of behavior rules, which in turn are triggered by the alterations of its data values over time. In the infoticle system, these behavior rules only specify (1) to which particular force the agent is 'attracted' to, and (2) whether it should speed up or slow down. In practice, the application simulates the linear progression of a timeline that corresponds to the time

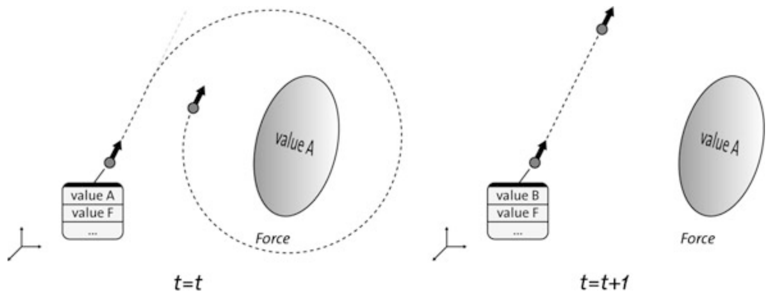


Fig. 13.4 A force attracts agents that contain data items with equal data values (*left*). Agents with dissimilar data items are not influenced by such force (*right*)

stored in the datasets. As the timeline progresses, data objects are updated with new data values. As a result, some data items, and thus agents, are updated and some are not. For those that are updated, their data items may either contain different data values, or equal ones. The infoticle behavior rules specify that:

- **Data Update and Alteration**. Each agent of which its data item is updated over time, speeds up, and is 'attracted' to a force that represents the new 'updated' data value (see Fig. 13.4). Once it reaches a particular distance of the force, it starts orbiting around it. This rule causes agents with similar data items to congregate in a circular motion around the same data attribute (force), similar to space satellites that orbit planets.
- **Data Update Only.** Agents that are updated but encounter no change in data value only speed up while being attracted to the same force. This rule causes these agents to follow a more elliptic path around the same data attribute (force), similar to space comets around our sun.
- **No Data Update.** Agents that are not updated, are subjected to a virtual 'friction' and slow down, while being attracted to the same force. Such agents tend to 'fall back' and eventually 'crash' to the center of the force.

As shown in Fig. 13.5, flat ribbon-like visual elements graphically traced the spatial trajectories of the agents, so their dynamic behavior could be investigated in more detail, even in a static state. The width of the ribbon corresponds to a specific data attribute value, while time stamps clarify the historical trajectory of the agent.

13.4.1.3 Emergent Data Visualization Patterns

The infoticle method has been applied for a large dataset containing the Intranet file usage of a global, medium-sized company with about 7,000 employees over a timeframe of a year. Each agent represented a document that was stored on the Intranet file servers. The global offices were divided in seven different geographical categories (e.g. Asia, Europe, USA, etc.), each of which was assigned a unique 'force'. The application timeline simulated the daily file usage according to the data

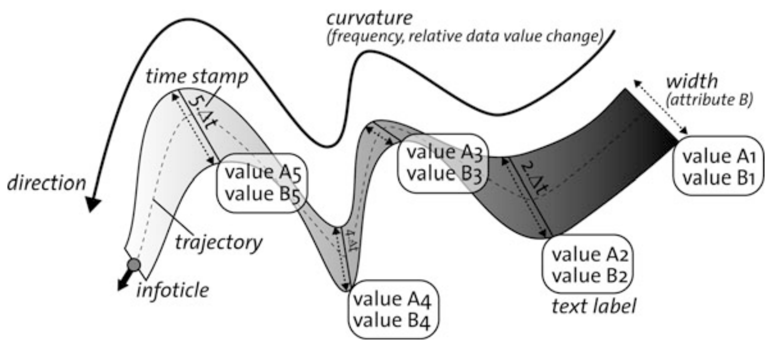


Fig. 13.5 Ribbon representation tracing the trajectory of the agent

stored inside the historical Intranet log files. During the visualization, Intranet files (or agents) traveled to those global offices in which they were downloaded for a specific timeframe. As a result, the visualization conveyed the global knowledge distribution over time, so that particular data patterns could be detected and compared to the traditional means of Intranet log file reports and file usage rankings that the company was purchasing to assess the effectiveness of their Intranet. Several additional technical features to deal with the nature of continuous, widely ranging, unpredictable data streams of different time zones (Vande Moere et al. 2004).

Users could explore the data visualization at any moment, by pausing, rewinding and forwarding the application. Accordingly, agents smoothly moved back and forward along their historical trajectories. Because of the specifically designed interactions between both speed and direction alterations, emergent visual effects appeared that conveyed meaningful data alteration trends over time. As agents could freely move in three-dimensional space depending only to their attraction to a specific force, agents were initially randomly distributed so that the initial point of origin of a data pattern plays no role. As illustrated in Fig. 13.6, the most meaningful emergent patterns included:

- **Star Pattern**. Documents that were only downloaded 'once' during a relatively long timeframe 'circled' around their representative geographical forces. The smaller the agent-force distance, the longer ago they had been downloaded. These documents were old and not effectively shared within the company, and possibly should be reclassified in the Intranet system.
- **Comet Pattern**. Documents that were downloaded often, but by a same geography, moved further away from that representative force in elliptical formed tracks. These were documents that were only shared within the same geographical unit, probably containing only locally valid knowledge.
- **Quark Pattern**. Documents that were downloaded (1) relatively often and (2) by several different regions were characterized by erratic movements in-between the respective forces. Obviously, these particles represented the most shared documents that contained information relevant to the whole company during that particular timeframe.

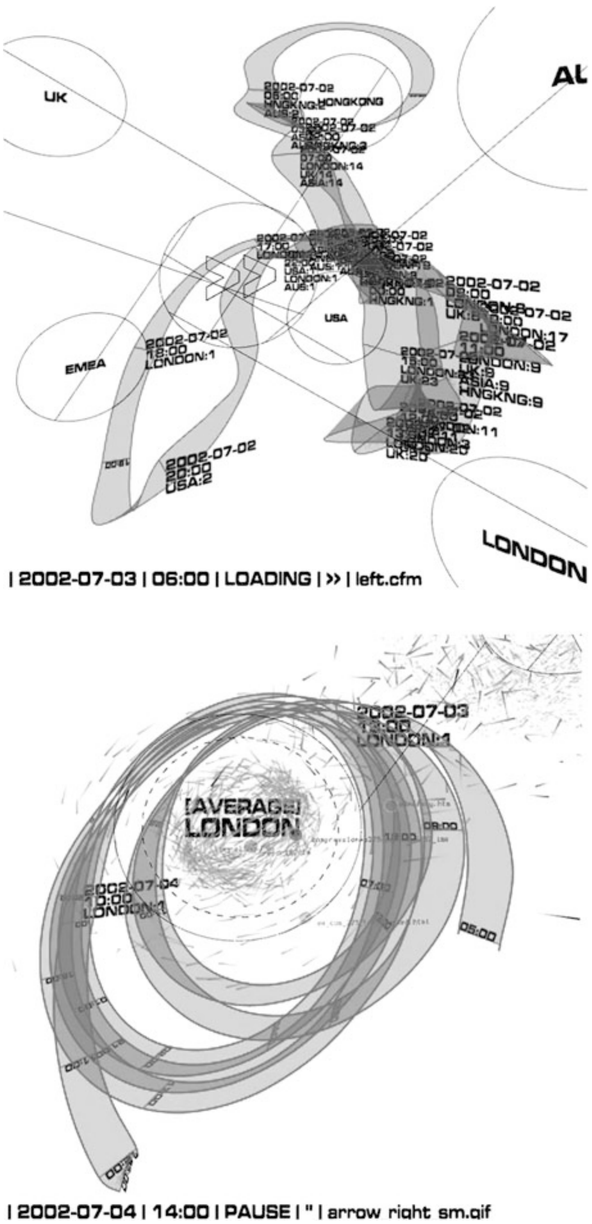

UK
AU
EMEA
USA
LONDON
2002-07-02 18:00 LONDON:1
2002-07-02 20:00 USA:2
| 2002-07-03 | 06:00 | LOADING | >> | left.cfm
2002-07-03 13:00 LONDON:1
[AVERAGE] LONDON
2002-07-04 10:00 LONDON:1
| 2002-07-04 | 14:00 | PAUSE | " | arrow right sm.gif

Fig. 13.6 Visually emergent Infoticle patterns, resulting from data-driven self-organizing behavior. The *circles* represent the different geographical units, the ribbon and time stamps trace the three-dimensional trajectory of a particle. Patterns visible: erratic Quark Pattern (*top*) versus elliptical Comet Patterns (*bottom*)

Users could investigate the resulting visual patterns in a dynamic as well as static state. Dynamically, users specifically looked for erratic, circular, elliptical or suddenly changing movements of agents, following the effect of specific document advertisements, or the passing-by of time-zone patterns. As illustrated in Fig. 13.7, users could search within spatial zones for regional trends, or followed the per-

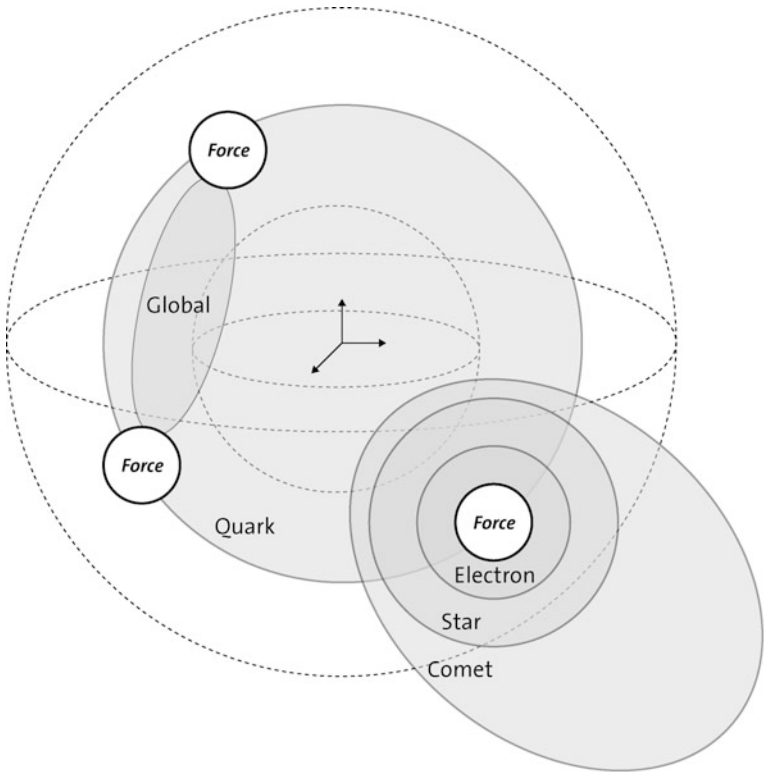


Fig. 13.7 Static zoning of agents according to the data pattern

formance of individual documents over time. In a static state, the formality of the ribbons tracing the agents' paths intuitively conveyed the dynamic characteristics of file usage. Globally, specific constellations of multiple agents appeared that conveyed so-called 'axes of knowledge' between two or more dominant global offices.

Several data patterns were discovered of which the company was not aware before. For instance, previously assumed highly 'popularity ranked' PDF documents tended to show up as comet patterns within very short timeframes. Although those documents were originally reported to be 'widely and popularly shared', their particular pattern proved that their relatively high download frequency was rather caused by a single user rapidly browsing through pages, causing the PDF software reader to cache separate pages and successively 'hitting' the Intranet server.

13.4.2 Metaphor 2: Information Flocking Method

The information flocking method augments the previously described infoticle technique and its capability to simulate interpretable motion typologies, with the concept of *swarming*. Individual agents are enhanced with limited vision and communication capabilities, so they can identify other agents in their neighborhood, read their data items, and move towards or away from them without the need of external forces.

13.4.2.1 Self-Organization Method: Swarming

Swarming is based on the mathematical simulation of flocking birds or schooling fish. Models of flocking reveal that overall group structures in animals are directly affected by transformations at local levels (Couzin et al. 2002). More specifically, swarming is believed to be reflective of the underlying internal relationships and energy considerations within the social hierarchy of a typical flock (Davies 2004). A possible functional explanation for emergent flocking behavior perceived in nature describes how animals at the edge of the herd are more likely to be selected by predators (Hamilton 1971). Accordingly, the flocking members would 'selfishly' attempt to move as close to the center of the herd as possible, and thus, in data visualization terminology, 'cluster' together. Reynolds (1987) successfully modeled the movements of so-called *boids* (or 'bird-objects') within a flock by designing a set of simple behavior rules that each member of the flock has to follow. This algorithm has been extensively used in the field of computer graphics, for instance to simulate swarming in popular cartoon movies (Stanton and Unkrich 2003). The boid concept has also been used in a technical context, offering a mathematical model of dynamic formations with respect to parameters of wireless, ad-hoc network communications systems (Kadrovach and Lamont 2002), or effective search strategies for performing exploratory geographical analyses (Macgill and Openshaw 1998). Swarming is an example of the so-called Particle-Swarm Optimization (PSO) research field that, like other evolutionary computation algorithms, can be applied to solve complex optimization problems from cross-disciplinary fields (Kennedy and Eberhart 2001).

Reynolds used the observation of real flocks to derive the three primary behavior rules that each member of a flock needs to follow. Each of these rules is applied in parallel between all pairs of boids, when their distance is smaller than predefined threshold values. Instead of simply averaging all behavior rules, a *prioritized acceleration allocation* strategy has to be used. This approach allocates priority according to the importance of the rule and normalizes the resulting values when a rule receives less than what was requested. This weighted average is used to compute the final velocity vector. The three behavior rules that govern each boid include (see Fig. 13.8):

- **Collision Avoidance.** Each boid needs to move away from boids nearby, in order to avoid crashing into one another.
- **Velocity Matching.** Each boid should move with about the same speed as the agents in its neighborhood. As a result, multiple boids, as a group, seem to follow a common goal.
- **Flock Centering.** Boids should attempt to stay close to other boids nearby. Boids in the center will feel no pull outwards, whereas boids in the periphery will be deflected inwards. As a result, each boid stays relatively near the center of the flock.

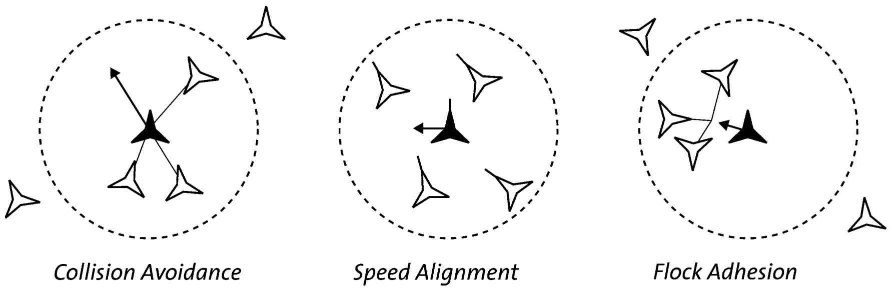


Fig. 13.8 Standard boid behavior rules (after Reynolds 1987)

13.4.2.2 Swarming Simulation as Data Mapping

Flocking behavior has been explored for the purpose of information display by Proctor and Winter (1998). They showed how the clustering tendencies of swarming fish can be exploited to represent a pair-wise similarity weight data matrix, containing the relationships of interest of employees. We have extended this method with several features, such as the inclusion of time-varying data, a more stable flocking algorithm and several technical features that are required to analyze the continuous stream of dynamic, unpredictable data in real time (Vande Moere 2004). Similar to the infoticle method, each agent is represented as a particle within three-dimensional virtual space. However, the boid agents are able to sense the presence of other agents in their vicinity. Each agent is visually represented by a colored, curved line connecting the three-dimensional points it has passed through, and show a gradually decreasing transparency to convey directionality. The agent color depicts positive (green) or negative (red) data value changes for a predefined data attribute. The agents' data items are sequentially updated according to the application timeline. In addition to the three traditional swarming rules mentioned before, each agent is governed by additional two behavior rules.

- **Data Similarity.** Each agent attempts to stay close to those agents with similar data values. This attraction rule results in the spatial clustering of agents that have similar data items. The similarity between agents is determined by testing whether the difference in data values of a single data attribute is lower than a predefined *data similarity threshold.*
- **Data Dissimilarity.** Each agent attempts to move away from those agents with dissimilar data values. This repulsion rule significantly accelerates the clustering effect of the first rule. Agents clustering too slowly would eventually cause the visualization to be confusing and non-representative. Accordingly, the continuous progression of the timeline requires all boids to cluster, 'un-cluster' and 're-cluster' efficiently.

It should be noted that too rapid data updates causes a flock to never attain a 'true equilibrium' with boids finding their ideal position, and thus globally reach a fully representative data representation. Therefore, it is necessary to stabilize the

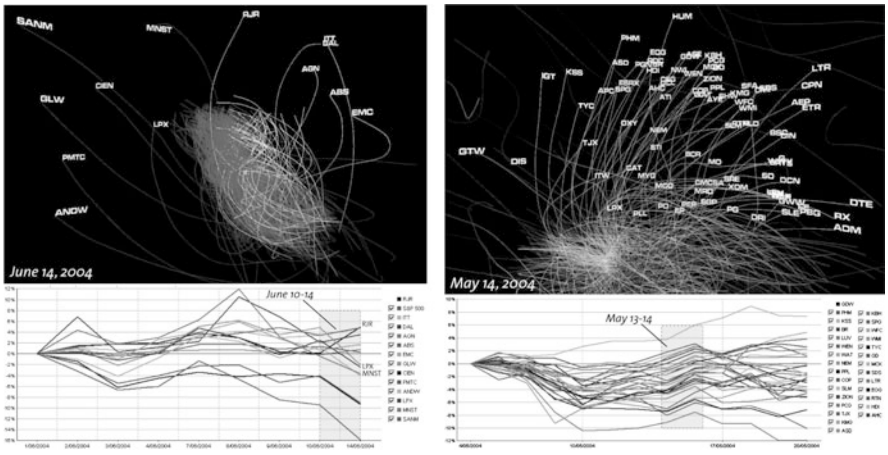


Fig. 13.9 Short-term zoning patterns: individual boids being expulsed by the main flock (*left*), and sub-flocks with similar data value changes appearing from the main flock (*right*)

boid behavior as much as possible by fine-tuning the swarming weighting factor values and the pair-wise boid threshold distances. The exact numerical values of these variables are determined through a trial-and-error process, as the application designer is unable to foresee the exact outcomes of the simultaneously applied local interactions.

13.4.2.3 Emergent Data Visualization Patterns

The information flocking method was tested with a dataset consisting of historical stock market quotes of one year totaling about (±500 companies × ± 200 working days) 12.631 data entries. The dataset was acquired from a public website (SWCP 2003) that accumulates the historical stock market prices of the 500 Standard & Poor's Index Directory, an index that represents a sample of 500 leading companies in the most important industries of the U.S. economy. Each data visualization agent represented a unique company. Agents calculated and compared the relative difference in price with their previous data update. As a result, the swarming behavior reflected the similarities in how their stock market quotes changed over time, rather than their exact stock market quote price. This method was capable of generating several interpretable emergent phenomena, as illustrated by Figs. 13.9 and 13.10.

- **Short-Term Clustering Patterns.** Boids that were nearby each other and moving in parallel directions represented similarities in stock quote change. Over time, several sub-clusters containing 'winning' and 'losing' companies formed within the main flock. Some individual boids and larger sub-flocks, consisting of 'outlying' agents that were affected by significantly different short-term stock market changes, split away from the main flock.

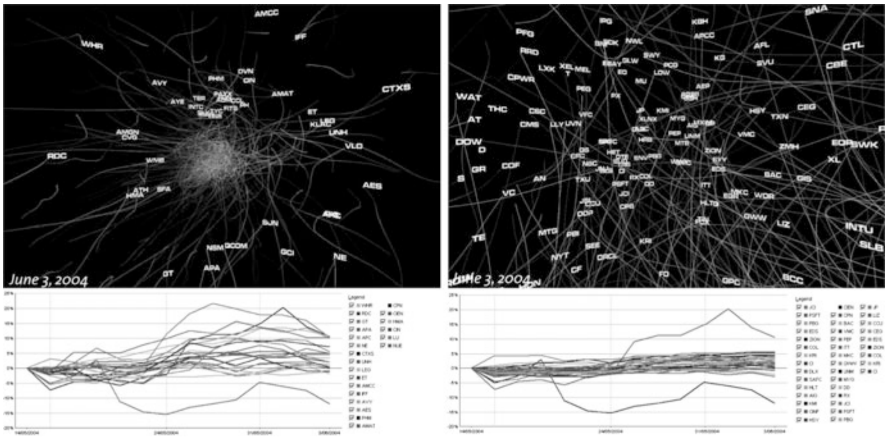


Fig. 13.10 Long-term zoning patterns: the flock core (*left*), versus the flock periphery (*right*). The according line graphs convey the relative volatility of the according stock quotes

- **Long-Term Zoning Patterns.** A visual distinction could be made between boids in the perceived flock center, and those positioned in the flock periphery. Subsequent analysis of the corresponding stock market price evolutions showed that data items with long-term and volatile data value changes were loosely positioned at the outside of the flock, whereas relatively long-term stable conglomerate entities remained closely together in the flock center.
- **Collective Motion Behavior.** The main flock often suddenly changed its global direction, or even seemed to 'implode' or 'explode' at specific moments in time. These erratic behaviors were traced back to significant global instabilities in the stock market. The emergent behavior can be explained by the 'guiding' influence of the more volatile boids in the flock periphery, which join their position with boids in the flock center when their stock market quotes have stabilized, causing an apparent implosion or explosion.
- **Visual Swarming Formality.** It was expected that the formality of the boid swarm as a whole would correspond with specific data characteristics, so that, for instance, irregular formations denoted chaotic data alterations, while sphere-like swarms formed out of stably altered stock market quote prices. However, it was concluded that such interpretations could only be made by a continuous observation of the changing shapes, and not from the perception of resulting static formations.

The information flocking method is able to represent short-term (e.g. clustering) as well as long-term (e.g. zoning) data tendencies in datasets with complex and potentially randomly changing data values. It is capable of displaying hundreds of time-varying data items simultaneously, even when the data changes in real-time, as it performs the data value comparisons within the visualization canvas itself. Displaying or even visually analyzing such amount of stock market companies simultaneously by line graphs would be impracticable. Whereas this approach certainly is not ideal for real-world stock exchange applications, it is proposed as a prototype towards more usable visualizations that are able to perform data analysis as well as data representation simultaneously and in real-time. It also illustrates how motion typology and spatial clustering can be used to convey complex data tendencies.

13.4.3 Metaphor 3: Cellular Ant Method

The *cellular ant* method augments the previously applied principles of self-organization. Agents will also be able to determine their visual attributes, such as its position, color and shape size. The resulting visual diagrams appear similar to those generated by the *multi-dimensional scaling* (MDS) approach, which displays the structure of distance-like datasets as simple geometrical pictures (Torgerson 1952) arranged in two-dimensional space. In MDS diagrams, the distance between pairs of data items denotes the degree of data similarity. Several MDS-like data visualization techniques exist, for instance in combination with animation (Bentley and Ward 1996) or recursive pattern arrangements (Ankerst et al. 1998). One should note that multi-dimensional scaling differs from *clustering* in that clustering partitions data into classes, while MDS computes positions, without providing an explicit decomposition into groups.

13.4.3.1 Self-Organization Method: Cellular Automata and Ant Foraging

The cellular ant method merges two established approaches: *ant-based foraging* from applied artificial intelligence and *cellular automata* from artificial life. More particularly, agent-based foraging is derived from the nest-cleaning characteristics of ant colonies in nature, and has been successfully applied to data clustering in the field of data mining.

Cellular automata (CA) is a well-known computational method originally proposed by Von Neumann (1966). It consists of a number of cells in a grid that each represents a discrete state (i.e. ‘alive’ or ‘dead’). All cells are governed by behavior rules that are iteratively applied, and generally only consider the states of the neighboring cells. When a specifically designed rule set is applied on well-chosen initialization constellations, visually intriguing cell patterns can emergently occur, such as demonstrated by the Game of Life application invented by Conway (Gardner 1970). Cellular automata can also be used for a case-study to investigate the controlling mechanisms of large multi-agent systems (Zambonelli et al. 2002).

Ant foraging is an example of *ant-based data mining*, which in turn combines the nest-cleaning characteristics of ant colonies with the task of data clustering. The typical ant clustering algorithm starts from a toroidal grid on which data objects are randomly scattered. Data objects are thus considered ‘lifeless’ entities that are moved around by foraging ants (Deneubourg et al. 1990; Kuntz et al. 1997; Lumer and Faieta 1994). Ants pick up data items and move around in randomly chosen directions. Ants then probabilistically decide whether to drop the data item, preferably in the vicinity of similar data items. A specific *object distance measure* variable α determines the degree of similarity between pairs of data objects, so that dissimilar items will not be placed together and similar items will be clustered. The optimal value for α cannot be determined without prior knowledge of the dataset, unless its value is adaptable (Handl and Meyer 2002). Different ant-based clustering approaches exist, which incorporate fuzzy-set theory (Pham and Brown 2003), topographic maps (Handl et al. 2005), or even biologically-inspired genetic algorithms (Ramos and Abraham 2004). These ant-based data mining techniques typically result in graphical diagrams with spatially separate clusters that, internally as well as relatively to each other, are unordered in term of data value. In effect, the usefulness and effectiveness of these methods for visual information display is low.

While CA simulations are typically determined by environmental states, ants can 'act' upon the environment and even change it to some degree, for instance by removing and dropping objects. The cellular ant approach applies typical CA rules, traditionally used for grid cell states, to the perception and reasoning of ants: cellular ant agents decide their actions depending on the discrete amount of 'similar' agents in their neighborhood, rather than using probabilistic mathematical functions. The essence of the cellular ant methodology is thus that agents will roam around, a technique similar to the ant-based foraging technique, and take actions depending on neighborhood densities, similar to the cellular automata technique. In addition, instead of simply picking up data objects, the ants move, as they 'are' the data items themselves. The idea of mapping data objects directly onto ants is relatively new. Labroche et al. (2002) associate data objects to ant genomes and simulates meetings between them to dynamically build partitions: ants detect the label that best fits their genome, which corresponds to the best cluster. A recent data mining clustering method has shown that data objects can be mapped onto ants, of which the apparent behaviors resemble that of cellular automata (Chen et al. 2004). It differs from the proposed agent data visualization methodology as it does not spatially order clusters, and is based on probability functions and predefined internal ant states.

13.4.3.2 Cellular Automata Ants as Data Mapping

Each data visualization agent is represented as a unique colored square cell, positioned within a toroidal rectangular grid. Each agent has a limited perception of its surrounding neighborhood, which consists of eight neighboring cells (see Fig. 13.11). Similarly to the swarming approach, an agent's behavior is based on a set of behavior rules that takes into account the presence of all other agents in its local neighborhood, and their corresponding data values. However, agents are in control of their dynamic behavior as well as their visual properties: each agent can move around or stay put, swap its position with a neighbor, and adapt its own color or negotiate its shape size.

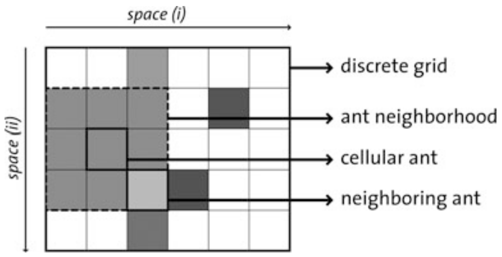


Fig. 13.11 Data visualization agent (or Cellular Ant) within its grid cell neighborhood

At initialization, all agents are randomly positioned within a toroidal grid. Similar to classical MDS (or CMDS) method, each agent calculates the Euclidian distance between its own normalized data item and that of each of its eight neighbors. This data distance measure represents an approximation of the similarity between pairs of data items, even when they contain multidimensional data values. Next, an agent will only consider and summate those agents of which the pair-wise similarity distance is below a specific, predefined *data similarity tolerance threshold value* t. Value t is conceptually similar to the *object distance measure* α in common ant-based clustering approaches. However, t originates from a cellular automata approach in that it is a fixed and discrete value, which generates a Boolean result (i.e. either a pair of data objects is "similar" or not) instead of a continuous similarity value (e.g. representing a numerical degree of similarity between pairs of data objects). Depending on the amount of agents in its neighborhood a particular agent considers as 'similar', an agent will then decide either to stay put, or to move. For each iteration, each agent is governed by five different behavior rules: edge repulsion and surface tension (together causing clustering), positional swapping, color determination and shape size adaptation (Vande Moere and Clayden 2005; Vande Moere et al. 2006).

- **Surface Tension.** Each agent has the freedom to roam around and meet other agents. If an agent has less than four similar neighbors, it will attempt to move to an empty cell next to the most similar ant in its neighborhood. Once an agent has four or more similar neighbors, it will stay put. As a result, agents that represent similar data items will stay close to each other, and small clusters form that act as seeds for other agents to connect to. Over time, these initial clusters can grow or merge with each other.
- **Edge Repulsion.** An agent that has one or more dissimilar agents in its neighborhood moves away from its current position, towards an empty cell closest to its most similar neighbor. This part of the algorithm is meant to create series of 'empty cells' as 'borders' around the separate clusters, so geometric clusters become better visually distinguishable. This means that two or more individual clusters that eventually 'share' a border with each other contain similar data elements between each other, thus creating a more meaningful data representation.
- **Positional Swapping.** At each iteration, each agent picks a random direction (i.e. horizontal, vertical, or one of both diagonals) in its neighborhood, with itself as the middle agent. It then reads the normalized data values of the corresponding neighbors, and calculates the (one-dimensional) distances between all these

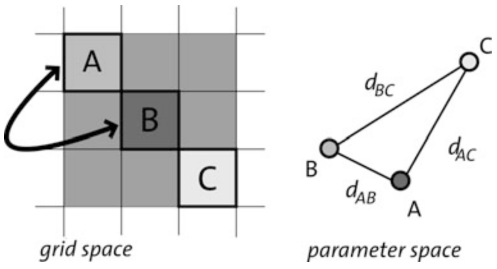


Fig. 13.12 Agents swapping grid position depending on data value. The largest data value distance in parameter space is d_{BC}, so that agent A should be positioned between B and C

agents in the multi-dimensional parameter space. Based on these three numerical pair-wise values, an agent is able to determine if it needs to 'swap' its position with one of both of its outer neighbors, or whether the current constellation is ideal. In practice, this means that if the Euclidian distance in parameter space between the middle agent and one of the outer agents is larger than the distance between the outer agents, the middle agent has to swap with the most similar outer agent (see Fig. 13.12). By swapping agents, the data value gradient between the three agents becomes continuous and monotonic. Subsequently, the swapping rule linearly orders agents in any chosen grid direction by multi-dimensional data similarity, so that 'more similar' agents are positioned closer to each other, and dissimilar ones are put further apart in the grid. Although this rule is applied in randomly chosen directions, a globally ordered structure emerges due to the multitude of iterations.

- **Color Determination.** At initialization, all agents are assigned a neutral color (i.e. white). Each ant that has not been swapped (and thus probably is well placed within its neighborhood) and is fully surrounded by eight similar neighbors, considers the degree of data similarity with all of its neighbors. If this degree is below a predefined, discrete *color seed similarity threshold* c, it will request a unique color from the application system. An agent with a neighborhood containing four or more data items of which the Euclidian distance is smaller than t but larger than c is relatively 'satisfied' with its current position and adopts the color of the most similar agent in its neighborhood. As a result, once the collection of ants is sufficiently ordered, colors become introduced where there are stable clusters of very similar ants. The individual agents that have received a unique color act as initial 'color seeds', which spread throughout the whole agent population. Because of the multitude of pair-wise interactions, any surplus of colors (in respect to data clusters) will disappear, while any shortfall of colors will reemerge once another potential ant is surrounded by eight very similar neighbors. One should note that the color of an agent is thus identical to its assumed *data class* or *data label.* Consequently, the resulting diagrams resemble that of (ant-based) data clustering in the domain of unsupervised data mining.
- **Shape Size Determination.** For each iteration step, the visual shape size of an ant is determined by following inducements. First, an agent chooses a random neighboring agent and reads the numerical value of a predefined data attribute, and its circular radius size, measured in screen pixels. It then considers whether

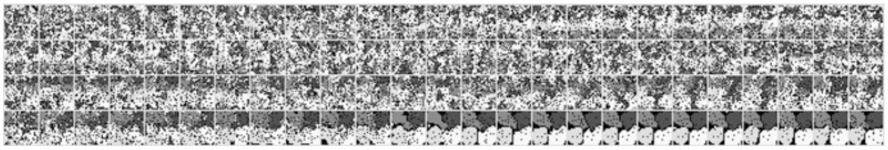


Fig. 13.13 Timeline snapshots of a random constellation of agents clustering, one snapshot per 100 iterations

its own radius versus data value ratio is similar to that of the neighboring agent, and adapts its own as well as its neighbor's shape size accordingly. For instance, if its size is too large in relation to its data value, it 'shrinks' by decreasing its amount of available pixels with a specific amount of pixels, and then offers these pixels to the other agent. When an agent becomes too large, it will 'punish' and decrease the size of its neighbor, so that this step will not longer be required in the next iteration. These constraints will emergently 'detect' the upper and lower shape size boundaries according to the data value scale, which emergently spread throughout all ants. Accordingly, instead of mapping data values directly to specific shape sizes, each agent is able to map one of its data attributes onto its size by negotiating with its local neighbors. However, the size of an agent does not necessarily correspond to the 'exact' value of that data attribute, but rather to how a data value locally relates to its neighborhood, and therefore whether clusters are locally homogeneous in respect of a specific data attribute. The shape size scale is able to autonomously adapt to the data scale, without the need for a separate data analysis or a predefined mapping rule between assumed data value and visual transformation (see also Sect. 13.3.1).

13.4.3.3 Emergent Data Visualization Patterns

The cellular ant method was applied with various artificially created and standard machine learning benchmarking datasets, ranging from two to up to nine data dimensions, and from 150 to 768 data items. Several emergent effects can be perceived:

- **Clustering.** As shown in Fig. 13.13, the initially randomly scattered data items self-organize in clearly perceivable clusters of similar data type over a process of several hundred iterations. The amount of required iterations varies with the random initialization distribution and with the grid size for an equal dataset. On an emergent level, a relatively long period of iterations in apparent chaos suddenly transforms in an orderly and stable constellation.
- **Internal Cluster Order.** As shown in Fig. 13.14, the resulting clusters themselves are ordered internally: data items that are similar in parameter space, are also positioned nearby each other in visualization space. For instance, the highlighted ants share a border between different adjoining clusters, but also share this border in the 2D parameter space. This particular emergent phenomenon is ideal

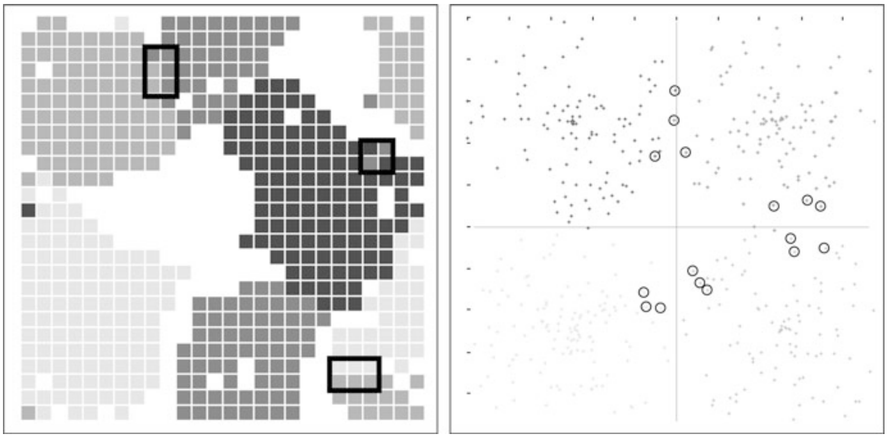


Fig. 13.14 Internal cluster order according to data similarity (500 ants, toroidal 26 × 26 grid, 2D synthetic dataset, 4 classes, $t = 0.27$). Agents positioned in grid space (*left*) versus those the data items in 2D parameter space (*right*). The *highlighted grid cells* and parameter points demonstrate that agents on 'shared' borders of two clusters are also shared in parameter space. Respective diagonal clusters in parameter space do not share any borders in the visualization

for the purpose of information display, as the resulting clusters have meaning and can be easily interpreted by users.

- **Relative Cluster Order.** The clusters themselves, seen as global phenomena, are positioned in an ordered way within the visualization grid. Cluster that are dissimilar in parameter space, have no 'common' borders in visualization space, and vice versa. For instance, the diagonal clusters in parameter space do not share any border in visualization space, as they are more dissimilar than the horizontally and vertically adjoining clusters. Accordingly, the information display becomes is more accurately representative and more easily comprehensible in comparison to other unsupervised clustering methods.
- **Data Class Determination.** The color determination rule results in a few different colors to emergently appear. Each color corresponds to a separate data class or data category, consisting of multiple data items that are significantly similar. At start, many colors appear, of which most disappear by merging with each other. Figure 13.15 shows how a car specification dataset (containing 38 items and 7 data dimensions, as taken from Wojciech 2001) results in a single spatial cluster, with three different colors. Whereas the spatial clustering was not able to separate the different types of cars because of similarity links between them, the color negotiation recognized three different types, roughly corresponding to the amount of car cylinders. The color negotiation is another feature that increases the legibility of the resulting information display.
- **Relative Data Attribute Distribution.** Figure 13.16 illustrates how shape size negotiation can be used to clarify high-dimensional data dependencies, without prior knowledge of the data scale and without using any predefined data mapping rules. Figure 13.16 uses the same dataset as Fig. 13.15, but maps a single

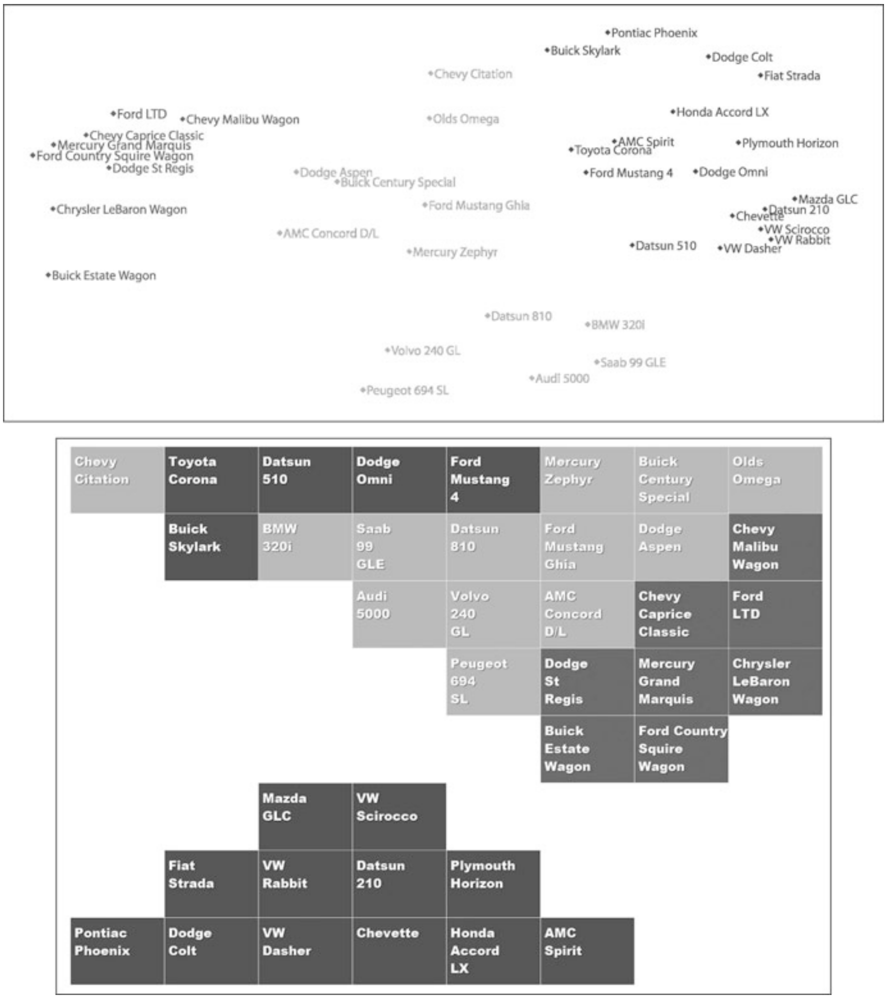


Fig. 13.15 Two different data visualizations of a car dataset. *Top*: Represented by traditional MDS (diagram based on Wojciech 2001, any grayscaling added by hand); *Bottom*: represented by the Cellular Ants representation with unsupervised color negotiation, here represented in grayscale

data attribute to circle size, showing the relative dominance of the cylinder count and MPG within clearly distinguishable clusters. The shape negotiation allows users to investigate how different singular data attributes are relatively distributed within multidimensional clusters.

One should note that the effectiveness of the cellular ant method is influenced by several variables that need to be configured beforehand, and mostly depend on the dataset characteristics. Such influential parameters include specific threshold values and overall grid size. Their specification requires a process of trial-by-error itera-

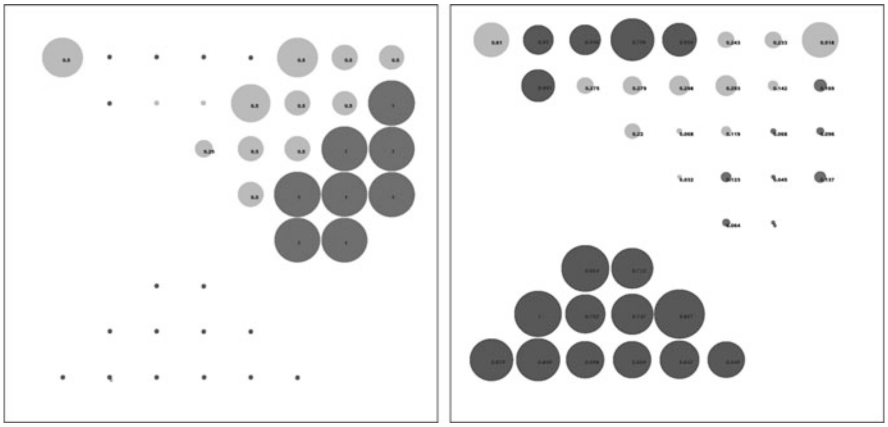


Fig. 13.16 A Cellular Ant representation in a toroidal grid with color and shape negotiation. Singular data attribute represented by the shape size: cylinder count (*left*) and Miles per Gallon (MPG) (*right*)

tions to assure that all variables are optimally chosen. Also, it cannot be assured that this method always finishes with a stable emergent result: a small number of ants might be swapping indefinitely, or individual ants (e.g. data outliers) might continuously roam around. Therefore, a simple performance graph was implemented that summated the number of similar ants in each ant's neighborhood. The visualization's efficiency corresponds to the slope of the according graph: once a plateau value has been reached over a number of iterations, the visualization has reached a stable state and can be halted. Quantitative benchmarking has shown similar performance with comparable data mining techniques (Vande Moere and Clayden 2005; Vande Moere et al. 2006). This simple prototype was developed to demonstrate that even simple self-organizing principles can lead to useful and meaningful results. Future work should focus on conceptual optimizations that should allow the method to be less dependent from external variables, and increase the overall performance in terms of required iterations, calculation performance and configurability.

13.5 Discussion

Data visualization supports users in creating a mental model by artificially constructing a visual representation from otherwise invisible data characteristics. An effective visualization facilitates rapid understanding of this model, generating intuitive connotations between visually emergent patterns and meaningful data patterns. This section will discuss some of the most important shortcomings of the self-organizing method, and some of potential solutions in the future.

13.5.1 Analysis

The unique characteristics of the self-organizing data visualization model is related with a set of specific issues, including:

- **Real-Time Data Analysis.** The self-organizing approach calculates data similarities in real-time. This means that a user does not necessarily need to conduct a pre-analysis of the datasets to determine the upper and lower bounds of the data mapping, and that datasets even can be updated by external sources in real-time. However, the continuous streaming of time-varying data poses specific issues as data items tend to be updated in irregular sequences so that some degree of data time-averaging is required. In addition, the continuous dataset querying, networking and analysis computations can negatively influence the calculation performance.
- **Calculation Performance.** As with any *parallel algorithm*, the simultaneous execution of behavior rules for each single agent is relatively calculation intensive. This effort increases exponentially with the number of agents, and thus the dataset size. Especially for the information flocking method, which requires one-to-one comparisons, and the cellular ant method, which requires one-to-eight comparisons, the dataset size is an important delimiting factor. Possible solutions for optimizing this issue include (1) predefined data similarity lookup tables, so that data item comparisons do not need to be calculated on the fly, (2) skipping behavior rules for specific iteration cycles, so that agents are progressively calculated, or (3) giving priority to specific agents, for instance those that are recognized to be guiding the emergent process. More technical solutions for performance optimization consist of (4) real-time data optimization, including data approximation and gradual data streaming (Hellerstein et al. 1999), agent adaptation, (5) the distribution or balancing of loads between agents (Decker and Sycara 1997), and (6) agent cooperation, by creating coalition formations of 'super agents' that adapt together as they have similar objectives, experience or goals (Ogston et al. 2003).
- **Non-Deterministic Results.** Using emergence as the core organizing principle inevitably ports aspects of uncertainty in the final results. This means that most generated representations are *non-deterministic*: different graphical diagrams can result from equal datasets result due to different initialization conditions. However, reoccurring data trends are consistently represented. In the case of the cellular ants, different initialization constellations inevitably result in different amounts of clusters, colors and sizes, so that multiple program executions and averaging is required to ensure results that can be confidently generalized.
- **Equilibrium.** Because of the continuous nature of the iterative cycles, it is often not intuitively determinable when the representation is 'finished', that is when the agents have found a suitable equilibrium for the current dataset situation. Therefore, self-organizing methods require some externally measurable degree of agent 'satisfaction', determined holistically over the whole agent population. Once this measurement reaches a 'plateau' over time, one can assume the most optimal solution for a specific configuration has been reached.

- **Comprehensibility**. The three different case studies have demonstrated how motion typology or visual properties of particles and colored grid cells are able to convey useful data patterns. However, users are still required to correlate and interpret these behavioral effects to particular meanings in relation to the dataset. Users might initially be confused because of the reliance on fully dynamic features, which are typically under-used in most software applications. There is also a latent danger of allowing users to interpret some emergent effects as meaningful, whereas no cause-and-effect in the context of the dataset could be detected.
- **Parameter Influence.** As with any algorithm that applies principles of self-organization, the occurrence of emergence is highly reliant on fine-tuning the variables involved. Altering these variables tend to result in different emergent visual effects, hereby affecting the effectiveness of depicting specific data patterns. Some variables control the visual effect of the emergent phenomena, so that they become easier to perceive and understand. As shown in Table 13.1, in the case of information flocking, at least 10 different variables are required to drive the traditional flocking behavior alone: 3 traditional plus 2 data similarity variables determine the Euclidian distances at which the behavior rules need to be invoked, and another 3 traditional plus 2 data similarity variables specify the relative importance of each behavior rule to one another (Reynolds 1987). The exact, ideal specification of such emergent-behavior influencing variables generally requires a relatively large amount of *trial-and-error* iterations, as one is unable to predict the emergent behavior of hundreds or thousands of internally interacting elements. Additional variables fine-tune the data analysis process: for instance, for each dataset separately, the exact threshold value that determines whether two data items are ‘similar’ or not needs to be defined. This value can be found by measuring the performance of agents towards a global equilibrium over several iterations. Several solutions for this issue exist, such as the implementation of an easily configurable software framework that allows the developer to fine-tune the configurable variables on-the-fly, without the need to change any programming code. Alternatively, the software itself could determine and then ‘optimize’ the possible solution space by brute force experimentation, and present the results to the user. Lastly, most of the variables could gradually change over time, towards their assumed most optimal value, allowing the emergent processes to be influenced by alternative values in a dynamic way.
- **Dataset Characteristics.** The different case studies have proven how self-organization for data visualization purposes can be accomplished with relatively small and low dimensional datasets, as the research focused on determining the feasibility of the proposed methodology. In particular, this technique seems to be ideally suited for relatively small, mid-dimensional and time-varying datasets. Further research will need to be conducted to determine whether this approach maintains its usefulness for more complex datasets, in terms of size, dimensionality, or time-variance.

Table 13.1 summarizes the qualitative differences between the three case studies. Because these different applications targeted fundamentally different datasets, purposes and usages, no benchmarking comparisons have been made.

Table 13.1 Qualitative differences between the 3 case studies demonstrating the self-organizing data visualization model

	Infoticle	**Information flocking**	**Cellular ants**
Agent concept	*– particle*	*– boid*	*– ant-like grid cell state*
Agent description	*– a point in 3D space that is attracted by a set of point forces following the rules of Newtonian mechanics*	*– a point in 3D space that swarms or flocks with similar flock members*	*– a grid cell that can move to neighboring grid cells, similar to the movement of an ant*
Dataset features	*– time-varying* *– 2-dimensional*	*– time-varying* *– 2-dimensional*	*– static* *– multi-dimensional*
Behavior algorithm	*– rule-based* *– cause and effect*	*– rule-based* *– decentralized inter-agent communication*	*– rule-based* *– decentralized inter-agent communication*
Behavior influence	*– data alterations over time* *– position of force in space*	*– data alterations over time* *– pair-wise data item comparisons*	*– data item comparisons between up to 8 neighbors*
Visual data mapping cues	*– 3D dynamic motion, determined by speed and direction*	*– 3D dynamic motion, determined by speed and direction* *– motion typology*	*– 2D position by 1-cell movement and pair-wise swapping* *– color* *– size*
Emergent phenomena	*– individual motion typology* *– grouping by collective motion typology* *– spatial grouping* *– motion path formality*	*– spatial clustering (position, speed & direction)* *– spatial zoning* *– individual and collective motion typology*	*– clustering* *– size determination* *– color determination*
Emergence variables	*– speed alteration (2)* *– force data attributes (1)*	*– flocking rules neighborhood distances (5)* *– prioritized acceleration allocation parameters (5)* *– data similarity threshold t*	*– data similarity threshold t* *– color seed threshold c* *– square grid size (1)*

13.5.2 Future

Traditional data visualization applications rely on the discretion and expertise of users to detect data patterns. Pattern detection is complex, and therefore, a future potential exists for using the results of data analysis to 'inform' the data mapping process. It is in this aspect that self-organizing data visualization might have the most useful potential: to construct a more effective, and more informed, visual representation based on the unsupervised analysis of dataset properties, visualization guidelines and visual cognitive knowledge. Such approach would be the first step towards more intelligent data visualization applications that become "aware" of the data they represent, the useful patterns they contain, and the most effective techniques to translate these into visual form. By using principles of self-organization, the intelligence required to detect data patterns is distributed: by spreading the required computations over multiple equal elements. As a result, the application can analyze the data in real time, becomes tolerant to local failures, allows for unexpected data alterations and potentially creates emergent solutions that are optimized to the specific patterns in and characteristics of the dataset. A possible future for self-organizing data visualization could lead to data mappings that are themselves emergent, creating unique, unforeseen, but ordered and interpretable representations that inherently reflect the relationships and complexities hidden within the dataset.

Self-organizing data visualization aims to provoke visual emergence by behavioral emergence. It is able to determine the position, color, shape and dynamic behavior of a data item according to its inherent relationships with other data items in the dataset. Each data item itself is a member of a decentralized multi-agent system, based on existing emergent simulations such as particle animation, flocking and swarming, ant-foraging or cellular automata. Self-organizing data visualization exploits already known emergent simulation algorithms to generate data-driven visual patterns. It is based on the assumption that a dataset is a collaborative agent system, which inherently contains all the knowledge required to derive a useful visual representation. Such dataset is capable of determining its own visual presence, and conveying the most meaningful data trends that it contains. Therefore, the self-organizing data visualization model points to a future in which data visualization becomes a data analysis tool that is aware of the data it represents, and how it should adapt its representation accordingly. As discussed before, several technical and conceptual hurdles still need to be overcome before principles of self-organizing data visualization can be integrated in useful applications. However, such applications will be able to proactively support users in their continuous urge to understand the ever more complex data collections that are continuously generated and accumulated in our society.

References

Ankerst, M., Berchtold, S., & Keim, D. A. (1998). Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *Proceedings of symposium on information visualization (Infovis'98)* (pp. 52–60). Washington: IEEE Computer Society.

Bartram, L., & Ware, C. (2002). Filtering and brushing with motion. *Information Visualization*, *1*(1), 66–79.

Bentley, C. L., & Ward, M. O. (1996). Animating multidimensional scaling to visualize *n*-dimensional data sets. In *Proceedings of symposium on information visualization (Infovis'96)* (pp. 72–73). Washington: IEEE Computer Society.

Brodlie, K. W., Brooke, J., Chen, M., Chisnall, D., Hughes, C. J., & John, N. W. (2006). A framework for adaptive visualization. In *IEEE visualization 2006*, Baltimore, Maryland, USA. Washington: IEEE Computer Society.

Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). *Readings in information visualization: using vision to think*. San Francisco: Morgan Kaufmann.

Chen, L., Xu, X., Chen, Y., & He, P. (2004). A novel ant clustering algorithm based on cellular automata. In *Proceedings of international conference of the intelligent agent technology (IAT'04)* (pp. 148–154). Washington: IEEE Computer Society.

Chi, E. H. (2000). A taxonomy of visualization techniques using the data state reference model. In *Proceedings of IEEE symposium on information visualization (Infovis)* (pp. 69–75).

Couzin, I. D., Krause, J., James, R., Ruxton, G. D., & Franks, N. R. (2002). Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, *218*, 1–11.

Davies, J. (2004). Why birds fly in formation: a new interpretation. *Interpretive Birding*, *5*(2).

Decker, K. S., & Sycara, K. (1997). Intelligent adaptive information agents. *Journal of Intelligent Information Systems*, *9*(3), 239–260.

Deneubourg, J., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., & Chrétien, L. (1990). The dynamics of collective sorting: robot-like ants and ant-like robots. In *From animals to animats: 1st international conference on simulation of adaptative behaviour* (pp. 356–363).

Ebert, A., Bender, M., Barthel, H., & Divivier, A. (2001). Tuning a component-based visualization system architecture by agents. In *Proceedings of international symposium on smart graphics*, IBM T.J. Watson Research Center.

Eick, S. G. (2001). Visualizing online activity. *Communications of the ACM*, *44*(8), 45–50.

Franklin, S., & Graesser, A. (1996). Is it an agent, or just a program? A taxonomy for autonomous agents. In *Proceedings of third international workshop on agent theories, architectures, and languages (ATAL'96)* (pp. 21–35). Heidelberg: Springer.

Gardner, M. (1970). Mathematical games: the fantastic combinations of John Conway's new solitaire game 'Life'. *Scientific American*, *223*, 120–123.

Hamilton, W. D. (1971). Geometry for the selfish herd. *Journal of Theoretical Biology*, *31*, 295–311.

Handl, J., & Meyer, B. (2002). Improved ant-based clustering and sorting in a document retrieval interface. In *Lecture notes of computer science: Vol. 2439. Proceedings of international conference on parallel problem solving from nature (PPSN VII)* (pp. 913–923). Heidelberg: Springer.

Handl, J., Knowles, J., & Dorigo, M. (2005). Ant-based clustering and topographic mapping. *Artificial Life*, *12*(1), 35–61.

Healey, C. G., Amant, R. S., & Chang, J. (2001). Assisted visualization of e-commerce auction agents. In *Proceedings of graphics interface 2001*, Ottawa, Canada (pp. 201–208). New Jersey: Lawrence Erlbaum.

Hellerstein, J. M., Chou, A., Hidber, C., Olston, C., Raman, V., Roth, T., et al. (1999). Interactive data analysis: the control project. *IEEE Computer*, *32*(8), 51.

Hiraishi, H., Sawai, H., & Mizoguchi, F. (2001). Design of a visualization agent for WWW information. In *Lecture notes in computer science* (Vol. 2112, pp. 249–258). Heidelberg: Springer.

Ishizaki, S. (1996). Multiagent model of dynamic design: visualization as an emergent behavior of active design agents. In *Proceedings of SIGCHI conference on human factors in computing systems (CHI'96)*, Vancouver, British Columbia, Canada (pp. 347–354).

Jankun-Kelly, T. J., Ma, K.-L., & Gertz, M. (2002). A model for the visualization exploration process. In *Proceedings of IEEE visualization*, Boston, MA, USA. Washington: IEEE Computer Society.

Kadrovach, B. A., & Lamont, G. B. (2002). A particle swarm model for swarm-based networked sensor systems. In *Proceedings of ACM symposium on applied computing*, Madrid, Spain (pp. 918–924). New York: ACM.

Kennedy, J., & Eberhart, R. C. (2001). *Swarm intelligence*. San Mateo: Morgan Kaufmann.

Kramer, P., & Yantis, S. (1997). Perceptual grouping in space and time: evidence from the Ternus display. *Perception & Psychophysics*, *59*(1), 87–99.

Kuntz, A., Layzell, P., & Snyers, D. (1997). A colony of ant-like agents for partitioning in VLSI technology. In *Proceedings of European conference on artificial life* (pp. 417–424). Cambridge: The MIT Press.

Labroche, N., Monmarché, N., & Venturini, G. (2002). A new clustering algorithm based on the chemical recognition system of ants. In *Proceedings of European conference on artificial intelligence*, Lyon, France (pp. 345–349). Amsterdam: IOS Press.

Lander, J. (1998). Ocean spray in your face. *Game Developer*, 9–13.

Lethbridge, T. C., & Ware, C. (1990). Animation using behavior functions. In T. Ichikawa, E. Jungert, & R. R. Korfhage (Eds.), *Visual languages and applications* (pp. 237–252). New York: Plenum Press.

Lumer, E. D., & Faieta, B. (1994). Diversity and adaptation in populations of clustering ants. In *From animals to animats: conference on simulation of adaptative behaviour* (pp. 501–508). Cambridge: The MIT Press.

Macgill, J., & Openshaw, S. (1998). The use of flocks to drive a geographic analysis machine. In *Proceedings of international conference on GeoComputation*, Bristol, United Kingdom. Manchester: GeoComputation. CD-ROM.

Mackinlay, J. D. (1986). Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, *5*(2), 110–141.

Marefat, M. M., Varecka, A. F., & Yost, J. (1997). An intelligent visualization agent for simulation-based decision support. *Computing in Science & Engineering*, *4*(3), 72–82.

Martin, A. (1999). Particle systems. http://www.cs.wpi.edu/~matt/courses/cs563/talks/psys.html. Retrieved August 2006.

Mason, K., Denzinger, J., & Carpendale, S. (2004). Negotiating gestalt: artistic expression and coalition formation in multiagent systems. In *Proceedings of international symposium on smart graphics* (pp. 103–115). Berlin: Springer.

Ogston, E., Overeinder, B., van Steen, M., & Brazier, F. (2003). A method for decentralized clustering in large multi-agent systems. In *Proceedings of international conference on autonomous agents and multi-agent systems*, Melbourne, Australia (pp. 789–796). New York: ACM.

Pham, B., & Brown, R. (2003). Multi-agent approach for visualisation of fuzzy systems. In *Lecture notes in computer science* (Vol. 2659, pp. 995–1004). Berlin: Springer.

Proctor, G., & Winter, C. (1998). Information flocking: data visualisation in virtual worlds using emergent behaviours. In *Proceedings of virtual worlds 98*, Paris, France (pp. 168–176). Berlin: Springer.

Ramos, V., & Abraham, A. (2004). Evolving a stigmergic self-organized data-mining. In *Proceedings of international conference on intelligent systems, design and applications (ISDA-04)*, Budapest, Hungary (pp. 725–730). Washington: IEEE Computer Society.

Reeves, W. T. (1983). Particle systems: a technique for modeling a class of fuzzy objects. *Computer Graphics*, *17*(3), 359–376.

Reynolds, C. W. (1987). Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, *21*(4), 25–34.

Roard, N., & Jones, M. W. (2006). Agent based visualization and strategies. In *Proceedings of conference in central Europe on computer graphics, visualization and computer vision (WSCG)*, Pilzen, Czech Republic. Pilzen: University of West Bohemia.

Robinson, N., & Shapcott, M. (2002). Data mining information visualisation—beyond charts and graphs. In *Proceedings of international conference on information visualisation*, London (pp. 577–583).

Sadok, B. Y., & Engelbert, M. N. (2004). Emulating a cooperative behavior in a generic association rule visualization tool. In *Proceedings of IEEE international conference on tools with artificial intelligence (ICTAI'04)*, Washington, DC, USA (pp. 148–155). Washington: IEEE Computer Society.

Schroeder, M., & Noy, P. (2001). Multi-agent visualisation based on multivariate data. In *Proceedings of international conference on autonomous agents*, Montreal, Quebec, Canada (pp. 85–91). New York: ACM.

Senay, H., & Ignatius, E. (1994). A knowledge-based system for visualization design. *IEEE Computer Graphics and Applications*, *14*(6), 36–47.

Shneiderman, B. (1998). *Designing the user interface: strategies for effective human-computer interaction*. Reading: Addison-Wesley.

Stanton, A., Unkrich, L. (Writers) (2003). Finding Nemo. In W.D.P.P.A. Studios (Producer), USA. Buena Vista Pictures/Walt Disney Pictures.

SWCP (2003). Historical data for S&P 500 stocks. http://kumo.swcp.com/stocks. Retrieved October 2006.

Tonnesen, D. (2001). Particle systems for artistic expression. In *Proceedings of subtle technologies conference*, Toronto, Canada (pp. 17–20) Toronto: University of Toronto.

Torgerson, W. S. (1952). Multidimensional scaling. *Psychometrika*, *17*, 401–419.

Tory, M., & Möller, T. (2004). Rethinking visualization: a high-level taxonomy. In *Proceedings of IEEE symposium on information visualization (Infovis'04)*, Austin, Texas (pp. 151–158).

Tufte, E. R. (2001). *The visual display of quantitative information*. Cheshire: Graphics Press.

Upson, C., Faulhaber, T. A. Jr., Kamins, D., Laidlaw, D., Schlegel, D., & Vroom, J. (1989). The application visualization system: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, *9*, 30–42.

van der Burg, J. (2000). Building an advanced particle system. *Game Developer*, 44–50.

Vande Moere, A. (2004). Time-varying data visualization using information flocking boids. In *Proceedings of symposium on information visualization (Infovis'04)*, Austin, USA (pp. 97–104).

Vande Moere, A., & Clayden, J. J. (2005). Cellular ants: combining ant-based clustering with cellular automata. In *Proceedings of IEEE international conference on tools with artificial intelligence (ICTAI'05)* (pp. 177–184).

Vande Moere, A., Mieusset, K. H., & Gross, M. (2004). Visualizing abstract information using motion properties of data-driven infoticles. In *Proceedings of conference on visualization and data analysis 2004 (IS&T/SPIE symposium on electronic imaging)*, San Jose, CA (pp. 33–44).

Vande Moere, A., Clayden, J. J., & Dong, A. (2006). Data clustering and visualization using cellular automata ants. In *Proceedings of ACS Australian joint conference on artificial intelligence (AI'06)*, Hobart, Australia (pp. 826–836). Berlin: Springer.

Von Neumann, J. (1966). *Theory of self-reproducing automata*. Illinois: University of Illinois Press.

Ware, C. (2000). *Information visualization perception for design*. San Francisco: Morgan Kaufmann.

Ware, C., Neufeld, E., & Bartram, L. (1999). Visualizing causal relations. In *Proceedings of IEEE symposium on information visualization (Infovis'99)*, San Francisco, CA (pp. 39–42). Washington: IEEE Computer Society.

Wojciech, B. (2001). Multivariate visualization techniques. http://www.pavis.org/essay/multivariate_visualization_techniques.html. Retrieved June 2006.

Woolridge, M. (2001). *Introduction to multiagent systems*. New York: Wiley.

Zambonelli, F., Mamei, M., & Roli, A. (2002). What can cellular automata tell us about the behavior of large multi-agent systems? In A. Omicini & J. Castro (Eds.), *Lecture notes in computer science* (Vol. 2603, pp. 216–231). Berlin: Springer.

Chapter 14
Memristive Excitable Automata: Structural Dynamics, Phenomenology, Localizations and Conductive Pathways

Andrew Adamatzky and Leon Chua

14.1 Introduction

The memristor (a passive resistor with memory) is a device whose resistance changes depending on the polarity and magnitude of a voltage applied to the device's terminals and the duration of this voltage's application. Its existence was theoretically postulated by Leon Chua in 1971 based on symmetry in integral variations of Ohm's laws (Chua 1971, 1980; Chua and Kang 1976). The memristor is characterised by a non-linear relationship between the charge and the flux; this relationship can be generalised to any two-terminal device in which resistance depends on the internal state of the system (Chua and Kang 1976). The memristor cannot be implemented using the three other passive circuit elements—resistor, capacitor and inductor—therefore the memristor is an atomic element of electronic circuitry (Chua 1971, 1980; Chua and Kang 1976). Using memristors one can achieve circuit functionalities that it is not possible to establish with resistors, capacitors and inductors, therefore the memristor is of great pragmatic usefulness. The first experimental prototypes of memristors are reported in Williams (2008), Erokhin and Fontana (2008), and Yang et al. (2008). Potential unique applications of memristors are in spintronic devices, ultra-dense information storage, neuromorphic circuits, and programmable electronics (Strukov et al. 2008).

Despite explosive growth of results in memristor studies there is still a few (if any) findings on phenomenology of spatially extended non-linear media with hundreds of thousands of locally connected memristors. We attempt to fill the gap

A. Adamatzky (✉)
University of the West of England, Bristol BS16 1QY, UK
e-mail: andrew.adamatzky@uwe.ac.uk

L. Chua
EECS Department, University of California, Berkeley, 253 Cory Hall 1770, Berkeley, CA 94720-1770, USA
e-mail: chua@eecs.berkeley.edu

and develop a minimalistic model of a discrete memristive medium. Structurally-dynamic (also called topological) cellular automata (Ilachinsky and Halpern 1987; Halpern and Caltagirone 1990) seem to be an ideal substrate to imitate discrete memristive medium. A cellular automaton is structurally-dynamic when links between cells can be removed and reinstated depending on states of cells these links connect. Structurally-dynamic automata are now proven tools to simulate physical and chemical discrete spaces (Rosé et al. 1994; Hasslacher and Meyer 1998; Hillman 1998; Requardt 2003; Alonso-Sanz 2006) and graph-rewriting media (Tomita et al. 2009); see overview in Ilachinsky (2009).

We must highlight that simulation of cellular automata in networks of memristors is discussed in full details in Itoh and Chua (2009). Itoh-Chua memristor cellular automata are automata made of memristors. Memristive cellular automata studied in present chapter are cellular automata which exhibit, or rather roughly imitate, certain memristive properties but otherwise are classical excitable structurally-dynamic cellular automata.

14.2 Memristive Automaton

A memristive automaton is a structurally-dynamic excitable cellular automaton where a link connecting two cells is removed or added if one of the cells is in excited state and another cell is in refractory state.

A cellular automaton $\mathcal{A}$ is an orthogonal array of uniform finite-state machines, or cells. Each cell takes finite number of states and updates its states in discrete time depending on states of its closest neighbours. All cells update their states simultaneously by the same rule. We consider eight-cell neighbourhood and three cell-states: resting $\circ$, excited $+$, and refractory $-$.

Let $u(x) = \{y : |x - y|_{L\infty} = 1\}$ be a neighbourhood of cell x. A cell x has a set of incoming links $\{l_{xy} : y \in u(x)\}$ which take states 0 and 1. A link l_{xy} is a link of excitation transfer from cell y to cell x. A link in state 0 is considered to be high-resistant, or non-conductive, and link in state 1 low resistant, or conductive. A link-state l^t_{xy} is updated depending on states of cells x and y at time step t: $l^t_{xy} = f(x^t, y^t)$. Resting state gives little indication of cell's previous history, therefore we will consider not resting cells contributing to a link state updates. When cells x and y are in the same state (bother cells are in state $+$ or both are in state $-$) no 'current' can flow between the cells, therefore scenarios $x^t = y^t$ are not taken into account.

Thus we assume that the only situations when $x^t, y^t \in \{+, -\}$ and $x^t \neq y^t$ may lead to changes in links conductivity:

$$l^{t+1}_{xy} = \begin{cases} a, & x^t = + \text{ and } y^t = - \\ b, & x^t = - \text{ and } y^t = + \\ l^t_{xy}, & \text{otherwise} \end{cases} \tag{14.1}$$

where $a \neq b$ and $a, b \in \{0, 1\}$. Thus we consider two types of automata $\mathcal{A}^{ab}$: $\mathcal{A}^{01}$ and $\mathcal{A}^{10}$.

A resting cell excites ($x^t = \circ \rightarrow x^t = +$ transition) depending on number of excited neighbours.

There are two ways to calculate a weighted sum of number of excited neighbours:

(i) $\Sigma_+ = \sum_{y \in u(x)} \chi(y^t, +)$
(ii) $\sigma_+ = \sum_{y \in u(x)} l_{xy} \chi(y^t, +)$,

where $\chi(y^t, +) = 1$ if $y^t = +$ and $\chi(y^t, +) = 0$ otherwise.

Thus, we consider two types of memristive automata. In automaton $\mathcal{A}_1$ a resting cell excites if $\sigma_+ > 0$. In automaton $\mathcal{A}_2$ a resting cell excites if $\Sigma_+ > 1$ or $\sigma_+ > 0$.

Note 1 Automaton $\mathcal{A}_1$ is a pre-memristive cellular automaton because it imitates only polarity (links update (14.1)) not voltage; automaton $\mathcal{A}_2$ is a memristive cellular automaton because it imitates both polarity (links update (14.1)) and current intensity (use of Σ_+ and σ_+).

A polarity of a current is imitated by excitable cellular automaton using excited $+$ and $-$ refractory states. If a cell x is in state $-$ and a cell y is in state $+$ then cell y symbolises an anode, and cell x a cathode. And we can say that a current flows from y to x. Indeed, such an abstraction is at the edge of physical reality, however this is the only way to develop a *minimal* discrete model of a memristive network. In automaton $\mathcal{A}_2$ the condition $\Sigma_+ > 1$ symbolises propagation of a high intensity current along all links, including links non-conductive for a low intensity current. This high intensity current in $\mathcal{A}_2$ resets conductivity of the links and also states of cells. The condition $\sigma_+ > 0$ reflects propagation of a low intensity current along conductive links. The current of low intensity does not affect states of links but only states of cells.

14.2.1 Experiments

We experiment with 300×300 cell automaton arrays, with non-periodic absorbing boundaries. We conduct experiments for two initial conditions on links' 'conductivity'

- L_1-condition: all links are conductive (for every cell x and its neighbour y $l^0_{xy} = 1$), and
- L_0-condition: all links are non-conductive (for every cell x and its neighbour y $l^0_{xy} = 0$).

While testing automata's response to external excitation we use point-wise and spatially extended stimulations. By point-wise stimulation we mean excitation of a single cell ($\mathcal{A}_1$) or a couple of cells ($\mathcal{A}_2$) of resting automata. These are minimal excitations to start propagating activity.

Let D-disc be a set of cells which lie at distance not more than r from the array centre $(n/2, n/2)$, $n = 300$, $r = n/3$). When undertaking D-stimulation we assign excited state $+$ to a cell of D with probability 0.05. In some cases we apply D-stimulation twice as follows. An automaton starts in L_1- or L_2-condition, we apply D-stimulation first time (we call it E_1-excitation) and wait till excitation waves propagate beyond boundaries of the array or a quasi-stationary structure is formed. After this transient period we apply D-stimulation again (E_2-excitation) without resetting links states.

Space-time dynamics of automata is illustrated by configurations of excitations and dynamics of link conductivity is shown either explicitly by arrows (in small configuration) or via grey-scale representation of cells' in-degrees: cell x is represented by pixel with grey-value proportional to the cell x's degree. Despite not representing exact configuration of local links, in-degrees give us a rough indicator of spatial distribution of conductivity in the medium. The higher is the in-degree at a given point, the higher is the conductivity at this point.

14.3 Phenomenology

A point-wise stimulation of automaton $\mathcal{A}_2$ leads to a persistent excitation, while automaton $\mathcal{A}_1$ returns to a resting state.

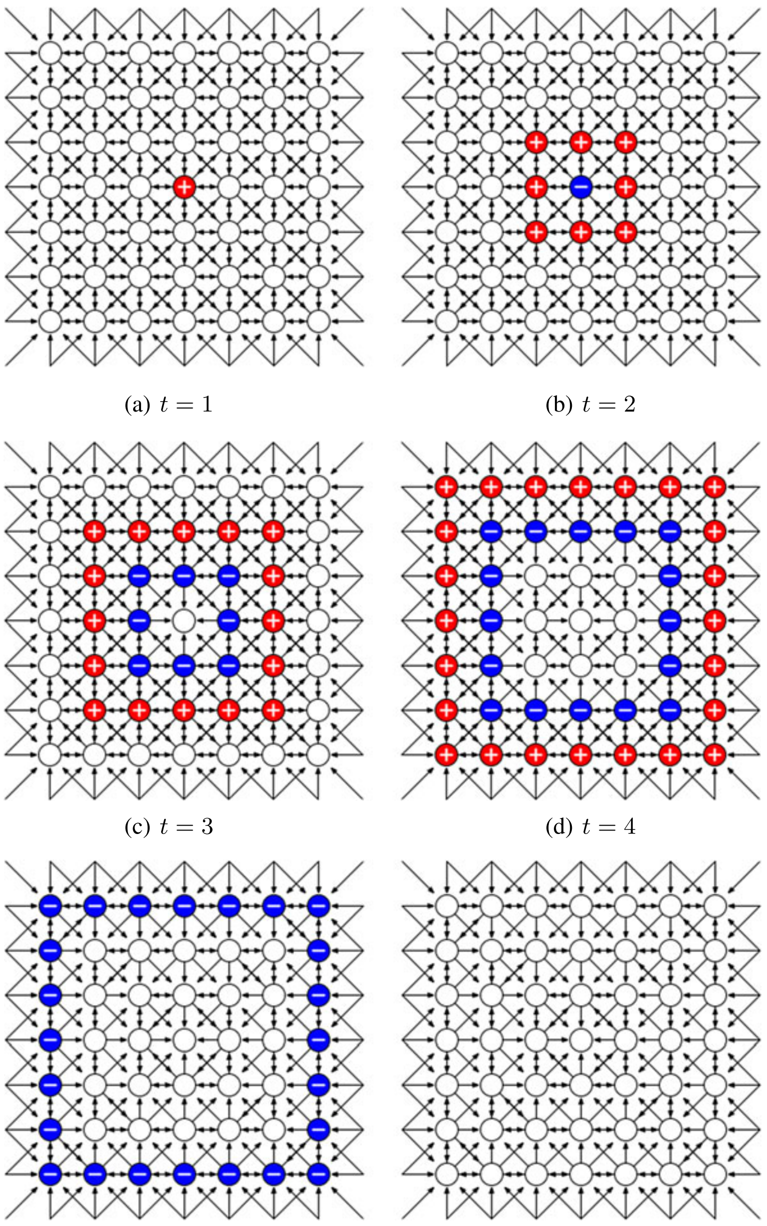
A single-cell excitation of resting automaton $\mathcal{A}_1$ (Fig. 14.1) or two-cell excitation of resting automaton $\mathcal{A}_2$ (Fig. 14.2) in L_1 initial conditions lead to formation of a 'classical' excitation wave-front. The wave-front propagates omni-directionally away from the initial perturbation site and updates states of links it is passing through.

Links leading from cells to the neighbours they excited are made non-conductive in development of $\mathcal{A}_i^{01}$ (Figs. 14.1 and 14.2); or we can say that links corresponding to normal vectors of propagating wave-front are made non-conductive. Cell excited at time step $t = 1$ becomes isolated. The situation is similar in development of automata $\mathcal{A}_1^{10}$ and $\mathcal{A}_2^{10}$ with the only difference that links connecting cells which are excited at time step t to cells they have been excited by are removed.

In summary, in automata $\mathcal{A}_i^{01}$ links associated with forward propagation of perturbation are made non-conductive, and in $\mathcal{A}_i^{10}$ links associated with backward propagation are made non-conductive. Excitation wave-front travelling from a single stimulation site forms a domain of co-aligned links. Excitation waves initiated in different cells collide and merge. Boundaries between domains formed by different fronts are represented by distinctive configurations of links (Fig. 14.3).

Automaton $\mathcal{A}_1$ is non-excitable in L_0-conditions because no excitation can propagate along non-conductive links. An outcome of two-site excitation of resting automaton $\mathcal{A}_2$ in L_0-condition depends on configuration of the initial excitation.

Let two diagonal neighbours (north-west and south-east) be excited (Fig. 14.4a) at time step t. These two cells transfer excitation to their two neighbours (north-east and south-west) as shown in Fig. 14.4b. Only links from north-west cell to north-east


(a) $t = 1$ (b) $t = 2$

(c) $t = 3$ (d) $t = 4$

(e) $t = 5$ (f) $t = 6$

Fig. 14.1 Snapshots of excitation and links dynamics in automaton $\mathcal{A}_1^{01}$. In initial configuration ($t = 1$) one cell is excited others are resting. *Arrow* from cell x to cell y means $l_{xy} = 1$. Excited cells are shown by *red discs* with sign '+', refractory cells by *blue discs* with sign '$-$', resting cells are blank. Links at the edges of cellular array corresponds to an absorbing boundary cells, which are always in resting state independently on states of their neighbours

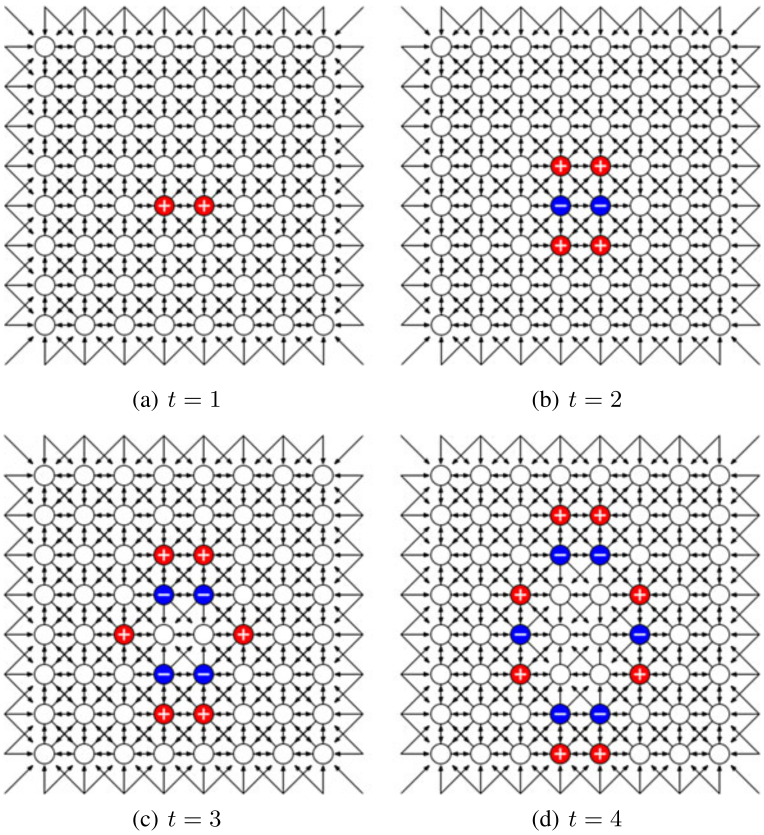


Fig. 14.2 Snapshots of excitation and links dynamics in automaton $\mathcal{A}_2^{01}$. In initial configuration two cells (minimum size of non-dying excitation for $\mathcal{A}_2$) are excited others are resting. *Arrow* from cell x to cell y means $l_{xy} = 1$. Excited cells are shown by *red discs* with sign '$+$', refractory cells by *blue discs* with sign '$-$', resting cells are blank (Color figure online)

and south-west, and south-east to south-west and north-east are formed (Fig. 14.4c) and excitation becomes extinguished (Fig. 14.4d).

Let excited cells be neighbours at the same row of cells (Fig. 14.5a). If they are excited at time step t then their north and south neighbours are excited due to condition $\Sigma_+ > 1$ taking place (Fig. 14.5b). The localised (two-cell size) excitations propagate north and south (Fig. 14.5c) and make links pointing backwards (towards source of initial excitation) conductive. At the second iteration cell lying east and west of initially perturbed cells becomes excited (Fig. 14.5c). By that initially perturbed cells return to resting state (Fig. 14.5d) and thus they become excited again. A growing pattern of recurrent excitation fills the lattice (Fig. 14.5d,e).

Repeated stimulation of memristive automata in a spatially-extended domain leads to formation of either disorganised activity domain emitting target waves of excitation or a sparse configuration of stationary oscillating localizations.

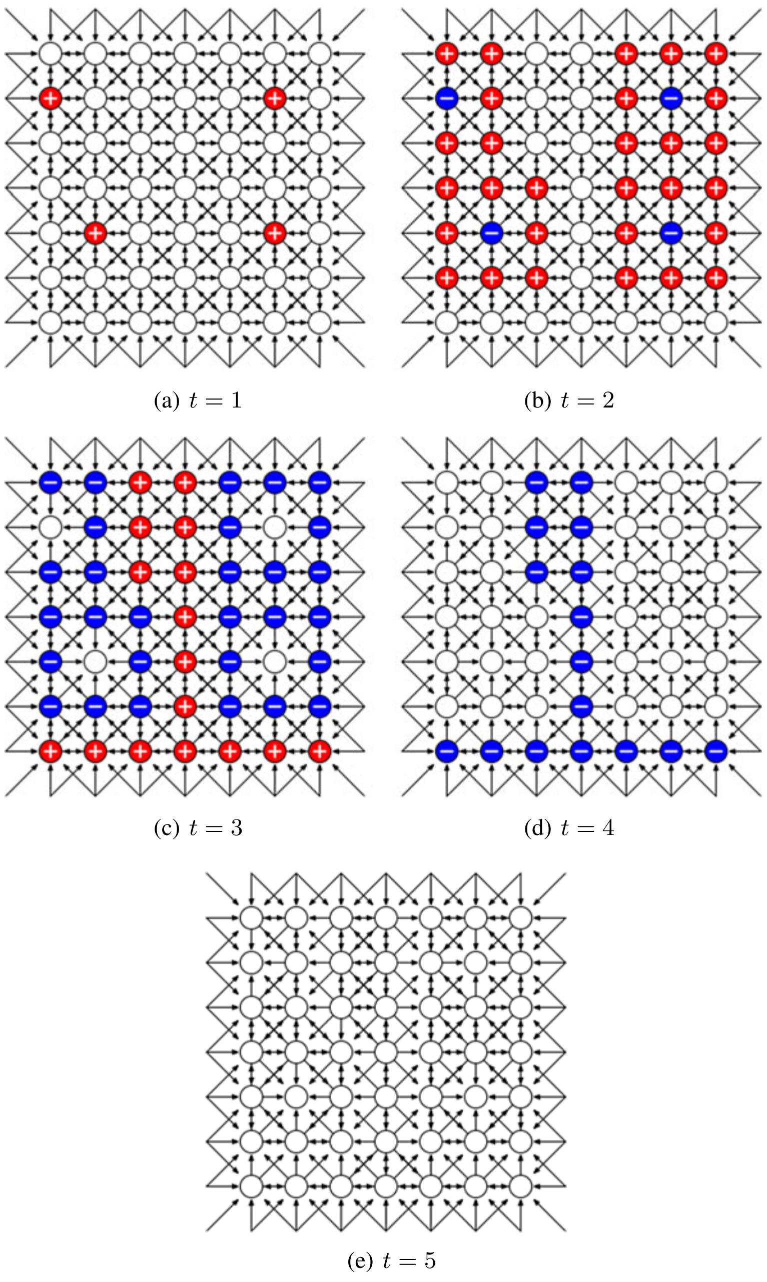


Fig. 14.3 Snapshots of excitation and links dynamics in automaton $\mathcal{A}_1^{01}$. In initial configuration several cells (those in '+'-state at $t = 1$) are excited others are resting. *Arrow* from cell x to cell y means $l_{xy} = 1$

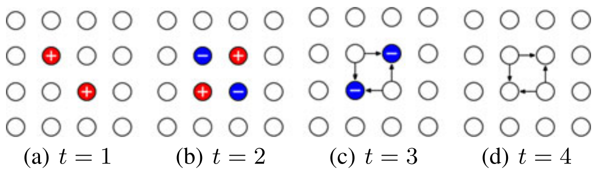


Fig. 14.4 Snapshots of excitation and links dynamics in automaton $\mathcal{A}_2^{01}$, L_0-start. In initial configuration two cells (minimum size of non-dying excitation in $\mathcal{A}$) are excited others are resting. *Arrow* from cell x to cell y means $l_{xy} = 1$

Examples of excitation dynamics in response to E_1- and E_2-excitations are shown in Fig. 14.6. Each singular perturbation in the first stimulation of automata being in L_1-condition leads to propagation of excitation. The waves merge into a single wave and disappear beyond lattice boundary.

Automata' behaviour become different after second D-stimulation (E_2-excitation). They all exhibit quasi-chaotic dynamic inside boundaries of disc D (shown by circle in Fig. 14.6). However the excitation dynamics in automaton $\mathcal{A}_1^{01}$ is reduced (after long transient period) to stationary oscillating localizations while in automata $\mathcal{A}_1^{10}$, $\mathcal{A}_2^{01}$, and $\mathcal{A}_2^{10}$ excitations outside boundaries of initial stimulation merge into target waves (Fig. 14.6). Oscillating localizations developed in $\mathcal{A}_1^{01}$ stay inside the boundaries of D.

Random excitation is extinguished immediately in automata $\mathcal{A}_1$ being in L_0-condition of total non-conductivity. When automaton $\mathcal{A}_2$ in L_0-condition is excited, the quasi-random excitation activity persists inside boundaries of D while omni-directional waves are formed outside D. Patterns of activity are not changed significantly after second random perturbation, E_1-excitation (Fig. 14.6).

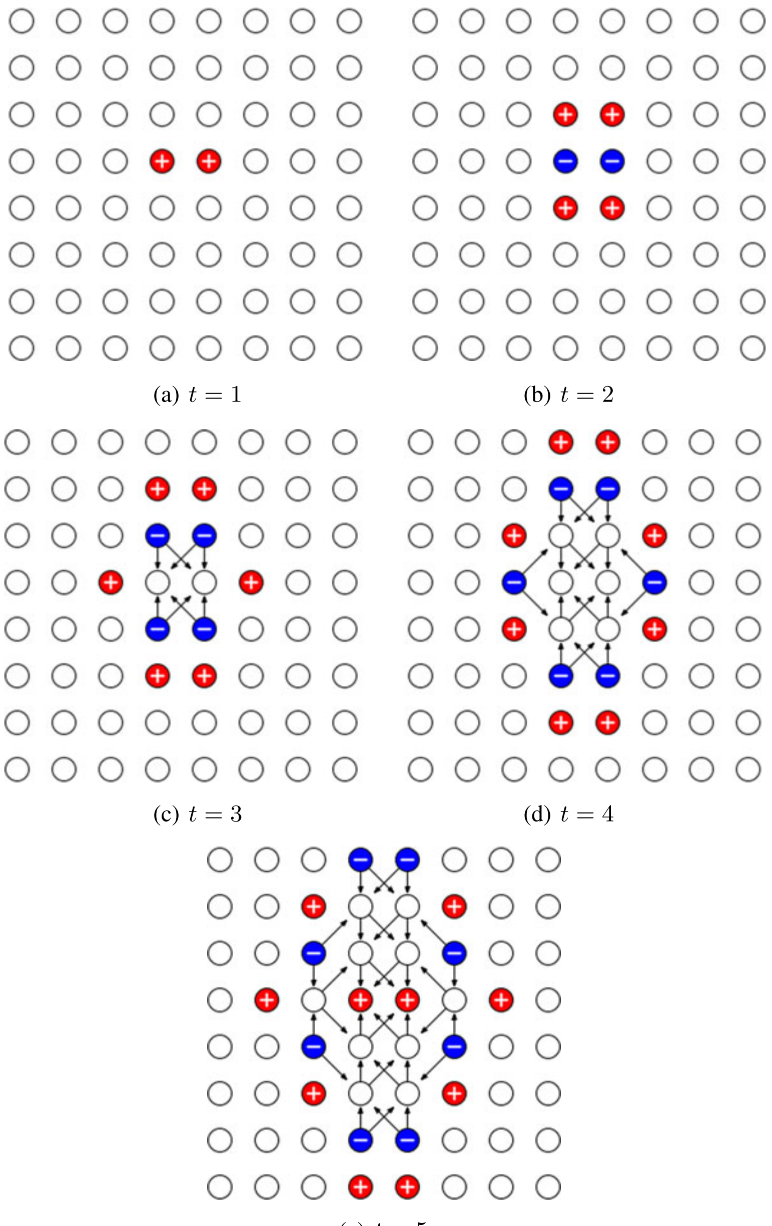
If there are both excited and refractory states in the external stimulation domain the developments are almost the same but $\mathcal{A}_1^{10}$ and $\mathcal{A}_2^{10}$ shows persistent excitation activity already at the first stimulation.

14.4 Oscillating Localisations

E_2-excitation of $\mathcal{A}_1^{01}$ leads to formation of excitation wave-fragments trapped in a structurally defined domains.

E_2-excitation of $\mathcal{A}_1^{01}$ leads to formation of sparsely distributed localised oscillating excitations, or oscillators (Fig. 14.7). An oscillating localisation (oscillator) usually consist of one or two mobile localizations which shuffle inside a small compact domain of the automaton array. This micro-wave is updating states of links and thus influencing its own behaviour. $\mathcal{A}_1^{01}$, after E_2-stimulation and formation of oscillating localizations, is characterised by a smooth balanced distribution of cells in-degrees.

Examples of two most commonly found oscillators—O_1 and O_2—are shown in Figs. 14.8 and 14.9, and their characteristics in Fig. 14.10. Both oscillators have



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$ (d) $t = 4$ (e) $t = 5$

Fig. 14.5 Snapshots of excitation and links dynamics in automaton $\mathcal{A}_2^{10}$, L_0-start. In initial configuration two cells (minimum size of non-dying excitation) are excited others are resting. *Arrow* from cell x to cell y means $l_{xy} = 1$

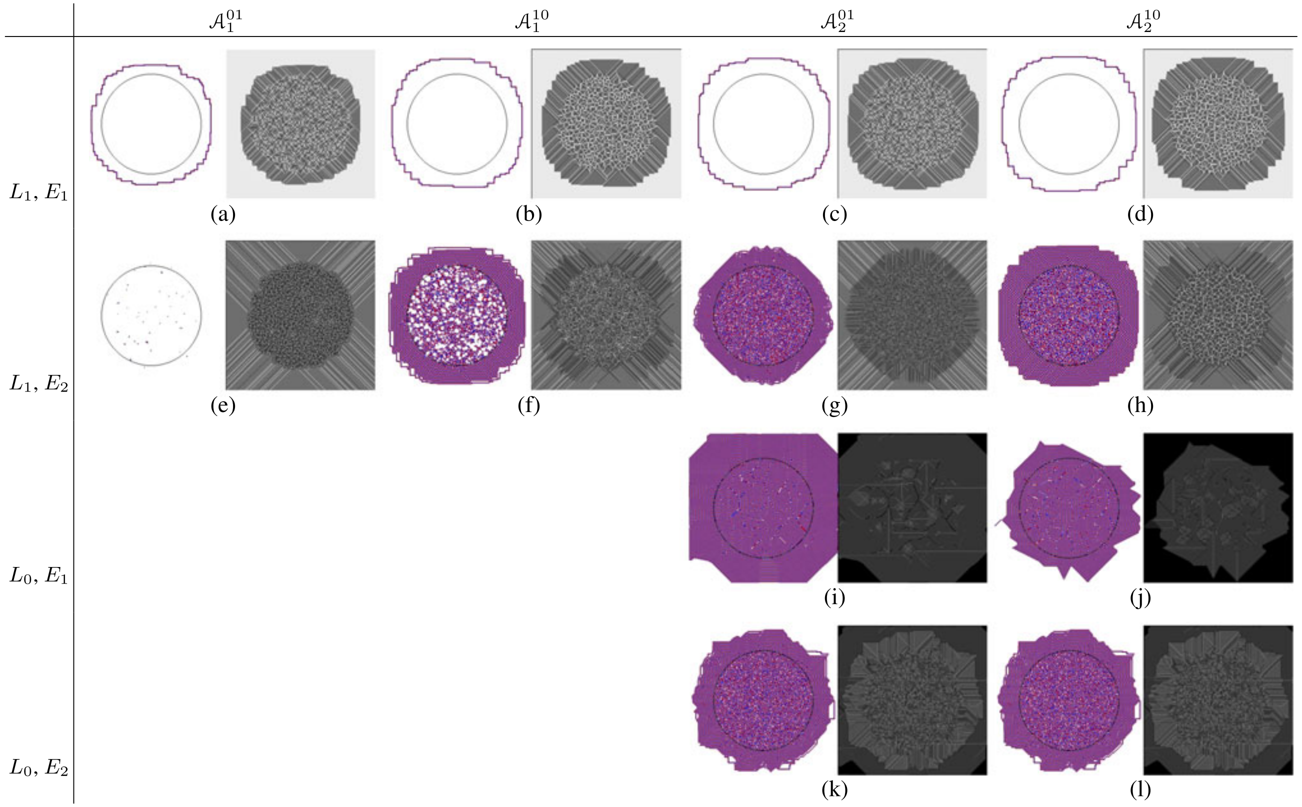


Fig. 14.6 Example of automaton behaviour in response to D-stimulation. Boundaries of disc D are shown as *circles*. Topology of links is represented by grey-levels of cells' in-degrees

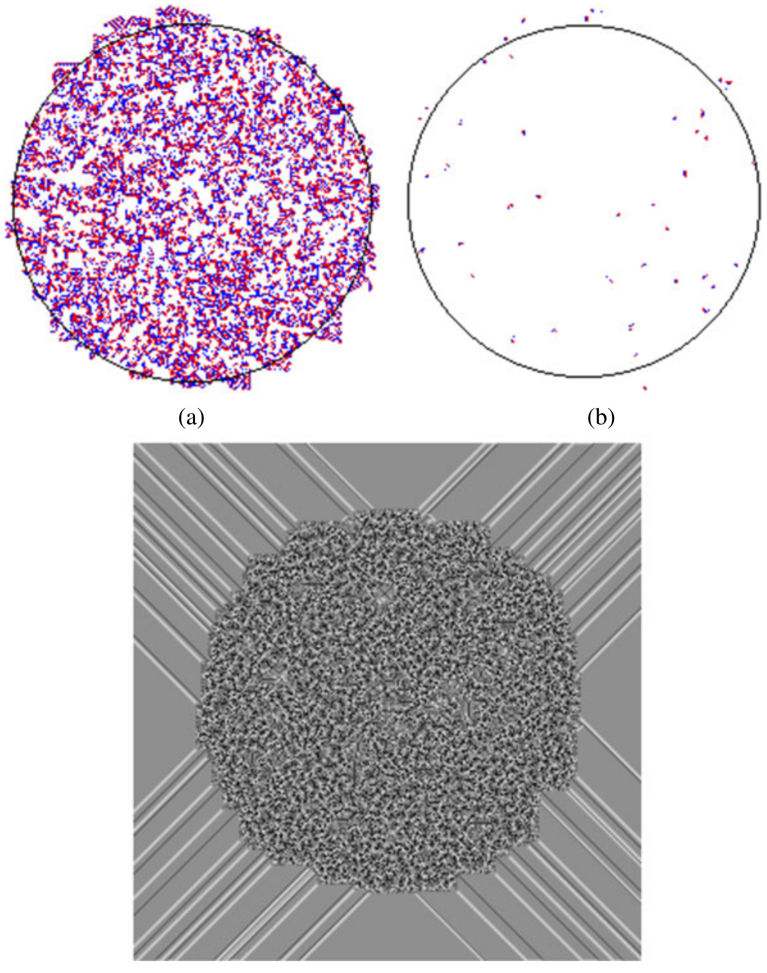


Fig. 14.7 Formation of localised excitations in $\mathcal{A}_1^{01}$, L_1-start. A localised excitation occupies a fixed size domain, the excitation may change its size periodically but never expands more then the certain fixed size. (**a**) configuration of automaton after E_2-excitation, boundary of D is shown by *black circle*; (**b**) the same automaton after transient period, only few oscillating localizations sustain; (**c**) links' conductivity presented via grey-values of cells' in-degrees

exactly the same minimum and maximum masses (measured as a sum of cells in excited and refractory states). Oscillator O_2 has much longer period than O_1 and larger maximum density. Oscillator O_1 spans large space during its transformation cycle, it occupies a sub-array of 6×6 cells when in its largest form.

In its minimal form O_1 consists of two cells: one cell is in state $+$ another in state $-$ (Fig. 14.8, $t = 0$). At next two steps of O_1's transformations a small localised excitation is formed. It propagates north (Fig. 14.8, $t = 1, 2, 3$). At fourth

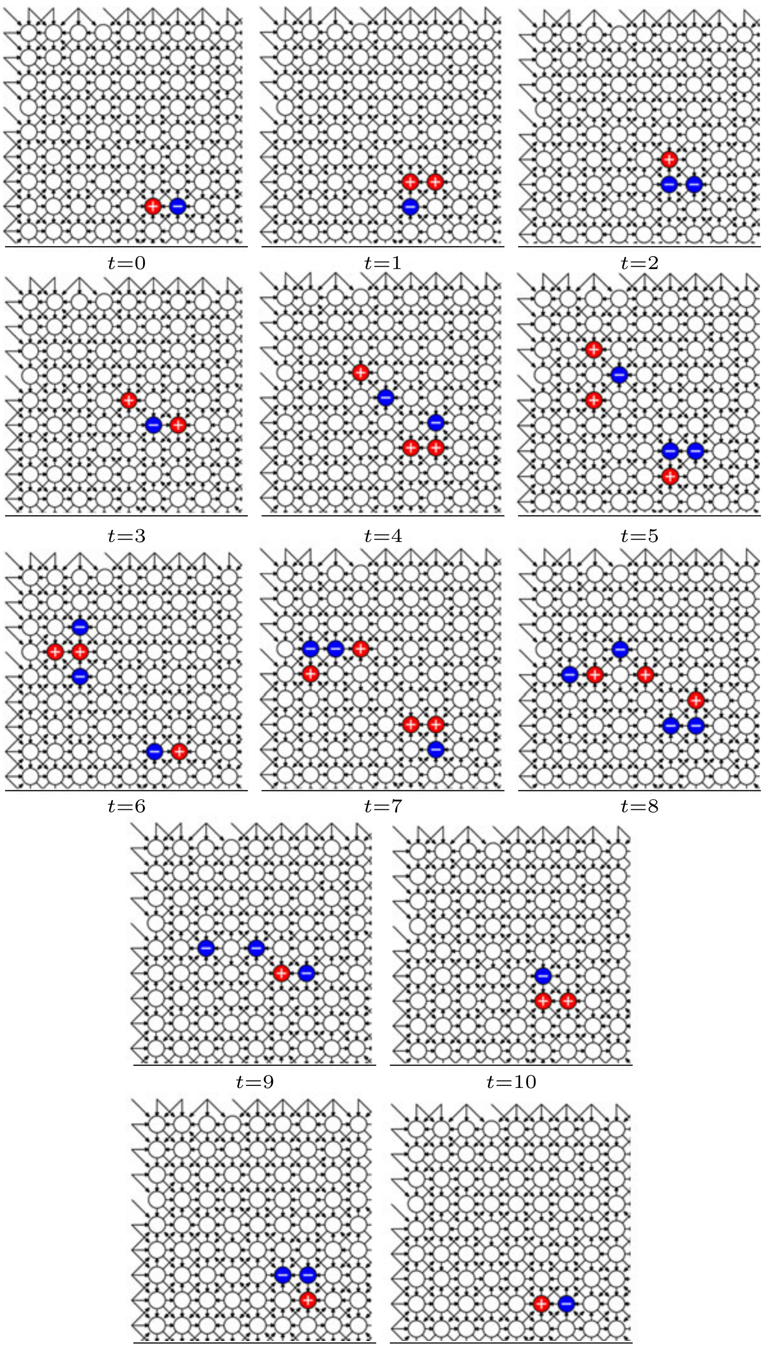


Fig. 14.8 Example of oscillator O_1 in $\mathcal{A}_1^{01}$

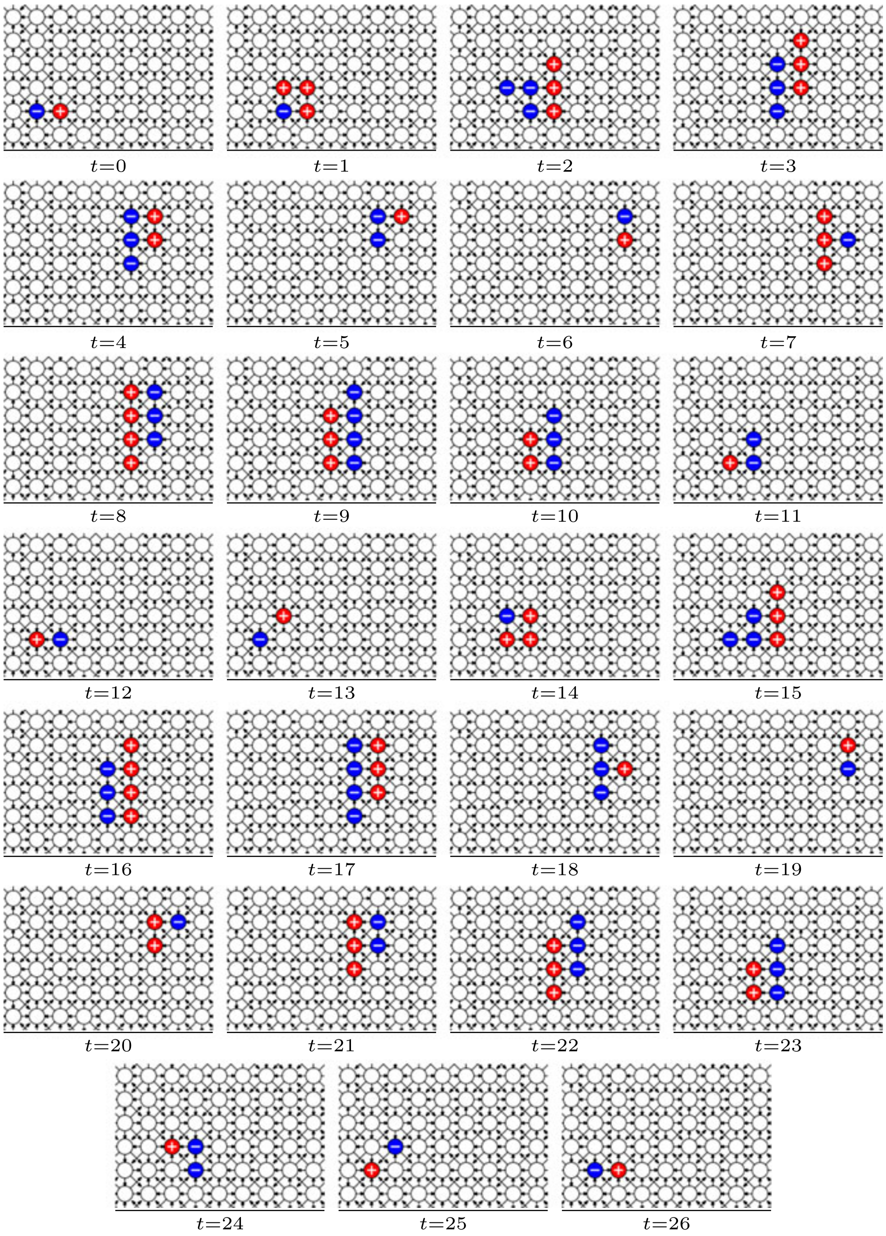


Fig. 14.9 Example of oscillator O_2 in $\mathcal{A}_1^{01}$

step of oscillator O_1's transformation the travelling localised excitation splits into two excitations: one travels north-west, another south (Fig. 14.8, $t = 4, 5$). The localisation travelling south returns to exact position of the first step excitation, just

Fig. 14.10 Characteristics of commonly found oscillators O_1 and O_2 in $\mathcal{A}_1^{01}$

Characteristic	O_1	O_2
Period	12	26
Minimum mass	2	2
Maximum mass	7	7
Minimum size	2	2
Maximum size	36	9
Minimum density	1	1
Maximum density	$\frac{1}{6}$	$\frac{1}{2}$

with swapped excited and refractory states (Fig. 14.8, compare $t = 6$ with $t = 0$) and then repeats a cycle of transformations $t = 1, 2$ (compare $t = 1$ with $t = 7$ an $t = 2$ with $t = 8$). The excitation travelling north-west become extinguished ($t = 8, 9$).

A couple $(-+)$ is a minimal configuration of oscillator O_2 (Fig. 14.9, $t = 0$). When the localisation starts it is development in configuration $(-+)$, and configuration of links' conductivity as shown in Fig. 14.9, it is transformed into excitation wave-fragment propagating east and north-east (Fig. 14.9, $t = 1, 2, 3$). At fourth step of oscillator's transformation the wave-fragment shrinks and by step $t = 6$ the oscillator repeats its original state $(-+)$ yet rotated by 90° clockwise. New excitation wave-fragment emerges and propagates west and south-west (Fig. 14.9, $t = 7 \dots 10$). The fragment contracts to configuration $(+-)$ by step $t = 12$. Then development and transformation of excitation wave-fragments is repeated (Fig. 14.9, $t = 13, \dots, 23$). The localisation returns to its original configuration $(-+)$ at $t = 26$.

Most oscillating localizations observed in experiments with $\mathcal{A}_1^{01}$ have very long periods. This is because the oscillators' behaviour is determined not only by cell-state transitions rules, as in classical cellular automata, but also by topology of links modified by repeated random stimulation and dynamics of the links affected by oscillators themselves.

14.5 Building Conductive Pathways

By exploring collisions between excitation wave-fragments travelling in $\mathcal{A}_2$ one can built information transmission pathways in an initially non-conductive medium. Routing primitives realised include signal splitting, signal echoing and signal turning.

Localizations travelling in $\mathcal{A}_2$, L_0-condition, can form pathways conductive for low-strength excitations. For example, a localisation of two excited and two refractory states propagates in $\mathcal{A}_2$ in the direction of its excited 'head'. The localisation forms a chain of conductive links oriented opposite to the localisation's velocity vector in case of $\mathcal{A}_2^{01}$, and in the direction of the localisation's propagation in case of $\mathcal{A}_2^{10}$ (Fig. 14.11).

A layout of conductive pathways, or 'wires', is determined by outcomes of collisions between path-laying particles. A detailed example of such collision-determined pathway building is shown in Fig. 14.12. Two travelling localiza-

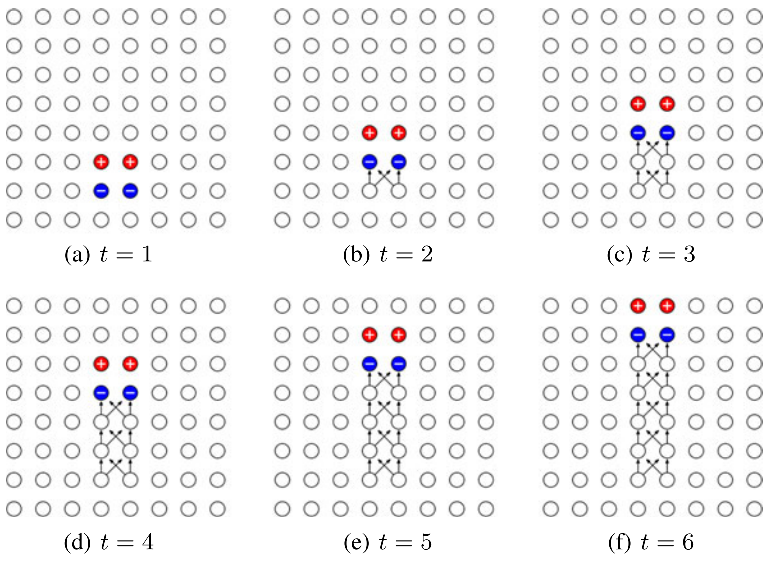


Fig. 14.11 Snapshots of excitation and links dynamics in automaton $\mathcal{A}_2^{10}$, L_0-start. In initial configuration two cells are excited, two cells are in refractory state, others are resting. *Arrow* from cell x to cell y means $l_{xy} = 1$

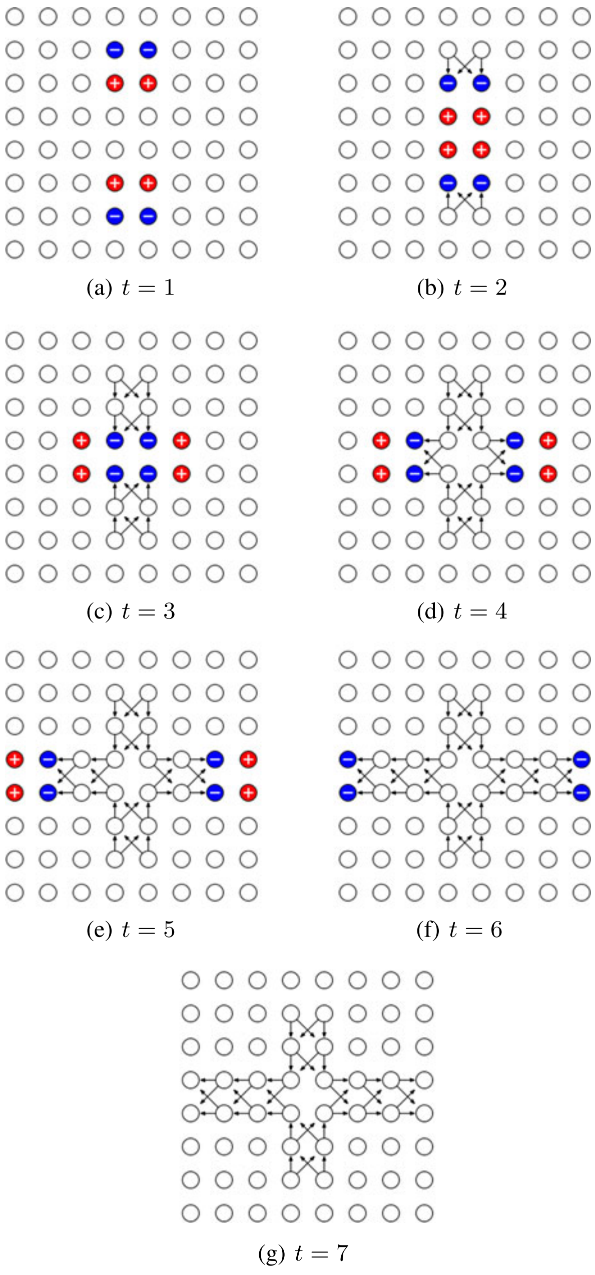
tions, 2^+-particles,[1] are initiated in $\mathcal{A}_2^{10}$ facing each other with their excited heads (Fig. 14.12a). One particle propagates south another north (Fig. 14.12a). When the particles collide they undergo elastic-like collision, in the result of which two 2^+-particles are formed: one travels east another west (Fig. 14.12c–f). When a weak excitation rule—a cell is excited if at least one neighbour is excited and no links are updated—is imposed on the automaton the pathways formed by these two colliding particles becomes selective. If an excitation is initiated in the southern or northern channel the excitation propagates till cross-junction and then branches into eastern and northern channels.

Pathways built by two 2^+-particles undergoing head-on collision being in different phases (odd and even distance between their start positions) and lateral offsets are shown in Fig. 14.13. Most T-bone collisions between 2^+-particles lead to formation of omnidirectionally growing excitation patterns. Only in situations when one particle hits a tail of another particle no uncontrollable growth occurs. The particle hitting a tail of another particle extinguishes and the other particle continues its journey undisturbed.

There are primitives of information routing implementable in collisions between 2^+-particles (Fig. 14.14). T-branching, or signal splitting, (Fig. 14.14a) is built by particles colliding with zero lateral shift and even number of cells between their initial positions (e.g. Fig. 14.13, entry (even, 0)). When signal travelling north reaches

[1] 2^+-particles are localizations consisting of two excited and two refractory states, which move along rows or columns of an excitable cellular array (Adamatzky 2011).

Fig. 14.12 Snapshots of excitation and link dynamics in automaton $\mathcal{A}_2^{10}$, L_0-start. In initial configuration two 2^+-particles are introduced in the medium. Four cells are excited, four cells are in refractory state, others are resting. *Arrow* from cell x to cell y means $l_{xy} = 1$



cross-junctions it splits into two signals—one travel west and another travels east; no signal continues straight propagation across junction.

Echo primitive (Fig. 14.14b) is constructed in a head-on collision between 2^+-particles with lateral shift two or three cells (see e.g. Fig. 14.13, entries (odd, 2),

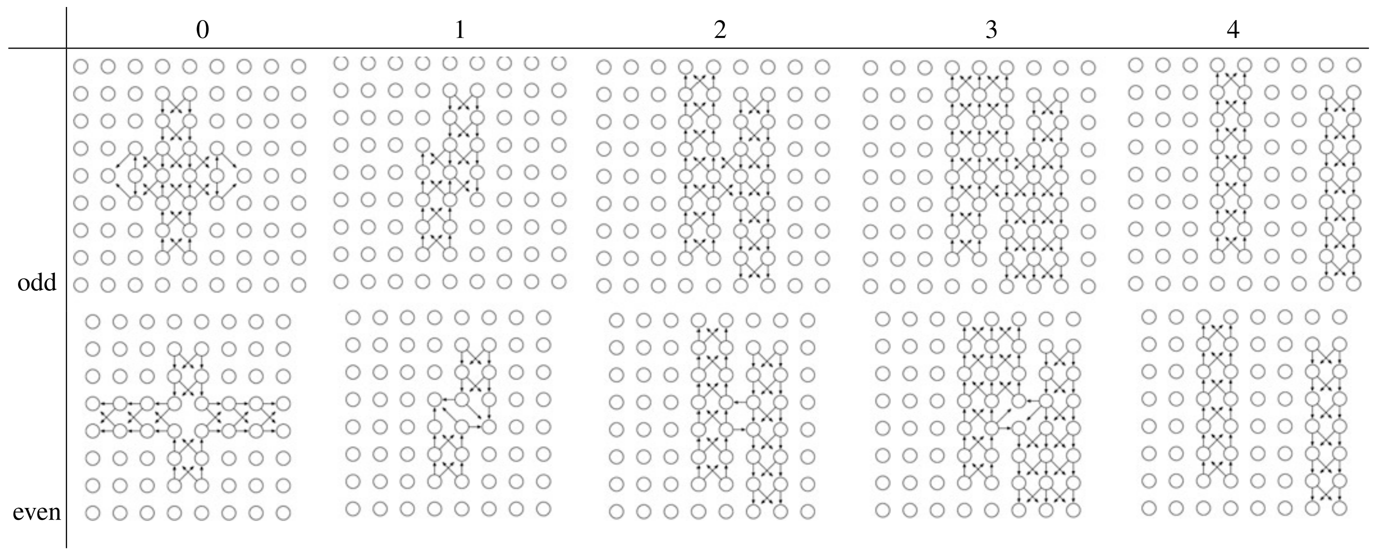


Fig. 14.13 Intersection of 'wires' build by propagating and colliding 2^+-particles. Automaton $\mathcal{A}_2^{10}$

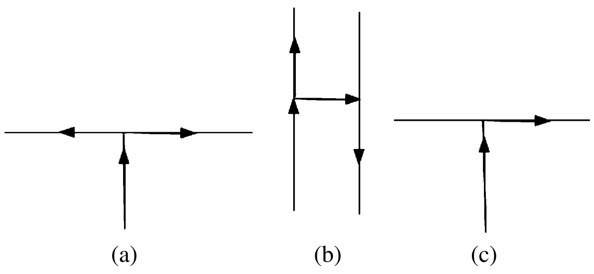


Fig. 14.14 Types of information transfer pathways laid out by interacting travelling localizations: (**a**) T-branching, (**b**) echo, (**c**) turning

(odd, 3), (even, 2), (even, 3)). The echo primitive consists of two anti-aligned (e.g. one propagates information northward and another southward) parallel information pathways. There is a bridge between the pathways. When signal propagating along one pathway reaches the bridge, it splits, daughter signal enters second pathway and propagates in the direction of mother signal's origination. In Fig. 14.14b mother-signal propagates north and daughter-signal south.

Turn primitive (Fig. 14.14c) is implemented in T-bone collision between two 2^+-particles. A signal generated at either of the pathways propagates in the direction of 2^+-particle which trajectory was undisturbed during collision.

14.6 Discussion

We designed a minimalistic model of a two-dimensional discrete memristive medium. Every site of such medium takes triple states, and a binary conductivity of links is updated depending on states of sites the links connect. The model is a hybrid between classical excitable cellular automata (Greenberg and Hastings 1978) and classical structurally-dynamic cellular automata (Ilachinsky and Halpern 1987). A memristive automaton with binary cell-states would give us even more elegant model however by using binary cell-states we could not easily detect source and sink of simulated 'currents'. Excitable cellular automata provide us with all necessary tools to imitate current polarity and to control local conductivity. From topology of excitation wave-fronts and wave-fragments we can even reconstruct relative location of a source of initiated current.

We defined two type of memristive cellular automata and characterised their space-time dynamics in response to point-wise and spatially extended perturbations. We classified several regimes of automata excitation activity, and provided detailed accounts of most common types of oscillating localizations. We did not undertake any systematic search for minimal oscillators though but just exemplified two most commonly found after random spatially-extended stimulation. Exhaustive search for all possible localised oscillations could be a topic of further studies.

With regards to formation of conductive pathways just few possible versions amongst many implementable were discussed in the papers. Opportunities to grow 'wires' in memristive automata are virtually unlimited. For example, in $\mathcal{A}_2^{01}$, L_0-start, after first D-stimulation (Fig. 14.15) generators of spiral and target waves are

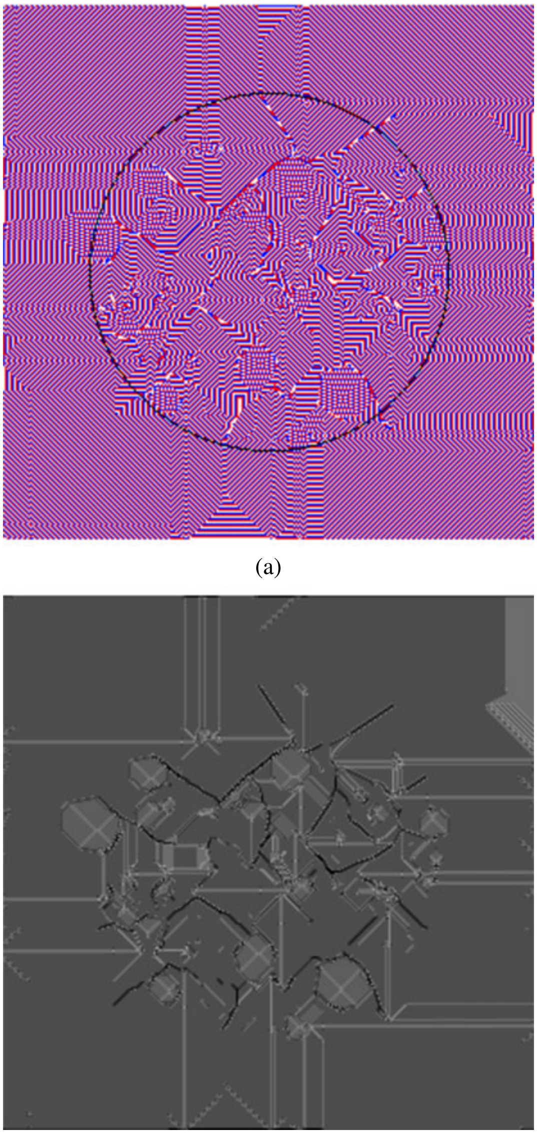


Fig. 14.15 Configuration of target waves (**a**) and configuration of cell in-degrees (**b**) in $\mathcal{A}_2^{01}$, L_0-start, D-stimulation, E_1-excitation. Boundary of D is shown by circle in (**a**)

(a)

(b)

formed inside D. Boundaries between the generators (they provide a partial approximation of a discrete Voronoi diagram over centres of the generators) are comprised of cells with high in-degrees. Such chains of high in-degree cells can play a role of conductive pathways even if we increase excitation threshold of the medium.

References

Adamatzky, A. (2011). *Computing in Nonlinear Media and Automata Collectives*. Bristol: IoP.

Alonso-Sanz, R. (2006). A structurally dynamic cellular automaton with memory. *Chaos, Solitons and Fractals*, *32*, 1285–1295.

Chua, L. O. (1971). Memristor—the missing circuit element. *IEEE Transactions on Circuit Theory*, *18*, 507–519.

Chua, L. O. (1980). Device modeling via non-linear circuit elements. *IEEE Transactions on Circuits and Systems*, *27*, 1014–1044.

Chua, L. O., & Kang, S. M. (1976). Memristive devices and systems. *Proceedings of the IEEE*, *64*, 209–223.

Erokhin, V., & Fontana, M. T. (2008). Electrochemically controlled polymeric device: a memristors (and more) found two years ago. arXiv:0807.0333v1 [cond-mat.soft].

Halpern, P., & Caltagirone, G. (1990). Behavior of topological cellular automata. *Complex Systems*, *4*, 623–651.

Hasslacher, B., & Meyer, D. A. (1998). Modelling dynamical geometry with lattice gas automata. *International Journal of Modern Physics C*, *9*, 1597–1605.

Hillman, D. (1998). *Combinatorial spacetimes*. Ph.D. dissertation. 234 pp. arXiv:hep-th/9805066v1.

Ilachinsky, A. (2009). Structurally dynamic cellular automata. In *Encyclopedia of complexity and systems science* (Vol. 19, pp. 8815–8850).

Ilachinsky, A., & Halpern, P. (1987). Structurally dynamic cellular automata. *Complex Systems*, *1*, 503–527.

Itoh, M., & Chua, L. (2009). Memristor cellular automata and memristor discrete-time cellular neural networks. *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, *19*, 3605–3656.

Greenberg, J. M., & Hastings, S. P. (1978). Spatial patterns for discrete models of diffusion in excitable media. *SIAM Journal on Applied Mathematics*, *34*, 515–523.

Requardt, M. (2003). A geometric renormalization group in discrete quantum space-time. *Journal of Mathematical Physics*, *44*, 5588.

Rosé, H., Hempel, H., & Schimansky-Geier, L. (1994). Stochastic dynamics of catalytic CO oxidation on Pt(100). *Physica. A*, *206*, 421–440.

Strukov, D. B., Snider, G. S., Stewart, D. R., & Williams, R. S. (2008). The missing memristor found. *Nature*, *453*, 80–83.

Tomita, K., Kurokawa, H., & Murata, S. (2009). Graph-rewriting automata as a natural extension of cellular automata. In *Understanding Complex Systems* (pp. 291–309). Berlin: Springer.

Williams, R. S. (2008). How we found the missing memristor. *IEEE Spectrum*, 2008-12-18.

Yang, J. J., Pickett, M. D., Li, X., Ohlberg, D. A. A., Stewart, D. R., & Williams, R. S. (2008). Memristive switching mechanism for metal-oxide-metal nanodevices. *Nature Nano*, *3*(7).

Part IV
Discussion

Chapter 15
A Turing Test for Emergence

Fabio Boschetti and Randall Gray

15.1 Introduction

Dealing with *complex* systems present a particular challenge to many traditional engineering approaches. The pertinent assumption inherent in these approaches is that component parts of a system can be neatly partitioned and that their interactions have limited, predictable effects. This assumption is not always tenable, and has an impact both on degree of overall control attainable and on the robustness of the resulting systems. A traffic controller does not need to give exact instructions to each vehicle on the road, and a Treasurer does not need to control each single business in a country; rather they both provide general guidelines which aim at a desire global outcome; they both rely on the local organization inherent in road traffic and business interactions to account for local details. Similarly we would like a designer to specify broad guidelines in order for a complex system to act according to a general requirement. Since the inherent organization we wish to exploit is often a dynamical and stable configuration, a system designed to capitalize on this may also display a robustness and adaptivity which is currently beyond our engineering abilities.

In the parlance of complex systems science, the global outcomes arising from broad guidelines on a system's components, including robustness and adaptivity, are often defined as emergent features. Because design inevitably requires a trial and error process, it is natural to expect that our community will need to develop methods to:

F. Boschetti (✉)
Marine and Atmospheric Research, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Private Bag 5, Wembley, WA 6913, Australia
e-mail: fabio.boschetti@csiro.au

R. Gray
Marine and Atmospheric Research, Commonwealth Scientific and Industrial Research Organisation (CSIRO), GPO Box 1538, Hobart, TAS 7001, Australia
e-mail: randall.gray@csiro.au

M. Prokopenko (ed.), *Advances in Applied Self-Organizing Systems*, Advanced Information and Knowledge Processing, DOI 10.1007/978-1-4471-5113-5_15,
© Springer-Verlag London 2013

- detect emergent features when they arise;
- categorize them in order to understand what classes of processes arise as a result of different initial conditions;
- experiment with various configurations in order to optimize the emergent processes.

Experimentation is something which is often carried out via computer simulation; however, computers are a perfect example of a 'traditional' engineering apparatus, and consequently display the very same features (lack of robustness and adaptivity, and a requirement for detailed instructions) which we are trying to circumvent. In this chapter we explore this apparent paradox and ask what kind of emergent features can be generated (and thus modelled) in a computational framework. We will show that this question directly relates to the other two items listed above, that is to the experimental detection of emergence and its classification.

15.2 Background

The concept of emergence evolved to capture our intuition that when a large number of entities interact, the resulting system can display features and behaviours which are not displayed by the individual constituents. The human body possesses behaviours and functions which are not expressed by our individual cells; metals show properties not displayed by individual atoms; societies undergo dynamics which transcend individuals. Basic examples can also be modelled very easily on a computer; famously, Conway's Game of Life (Gardner 1970) shows how very simple local rules generate features whose dynamics is not explicitly coded in the algorithm. Examples are so ubiquitous in Nature that some scientists suspect that all structures we see 'emerge' from underlying simpler levels (Laughlin and Pines 2000).

Nevertheless, emergence raises considerable intellectual and scientific challenge. Despite a vast literature, going back several decades (Corning 2005), no agreement can be found on a definition, nor on a framework for its study, nor on whether emergence is a 'real' natural phenomenon or merely a by-product of our perception, or a convenient way to make sense of processes otherwise too hard to comprehend (Crutchfield 1994b; Rabinowitz 2005).

Why should a process which appears so obvious and easy to model prove so hard to define and conceptualize? The fundamental reason is that in an emergent process it is very hard to discriminate 'who does what'. When I decide to listen to music, is it my 'emergent' self which takes the decision or my cells? My body depends on cellular activity for its functioning, so cells must be the controlling entities. However, no cell decides to listen to music since listening to music is not something cells 'do'. This leads straight into old and unsolved philosophical problems of causality, determinism and freewill.

Crucially, this is also a technological problem. Today, probably for the first time in history, technological developments in many applications depend on the understanding of emergent phenomena. Advances in Information Technology, Epidemiology, Ecosystem Management, Health Science, just to name a few, depend on approaches which go beyond traditional reductionism and address the understanding of how emergent properties arise, what they 'do' and how they can be controlled.

It thus seems natural that when we ask whether emergence is 'real' or merely lies 'in the eyes of the observer', or whether emergence is a distinct process of its own or encompasses different processes among which we are not yet able to discriminate, the answer needs to account for what these processes 'do'. In other words, we need to account for causal relationships and causal power. It may appear that we are trying to address a slippery problem (emergence) via one which is even more slippery (causality). This does not need to be so if we constrain what we mean by causality and we adopt an 'operative' definition. Following Pattee (1997) and Pearl (2000) we associate causal power with control: a process has causal power if, by acting upon it, we can change the effects it produces. Pattee (1997) describes this very simply: "*Useful causation requires control. … Clearly it is valuable to know that malaria is not a disease produced by "bad air" but results from Plasmodium parasites that are transmitted by Anopheles mosquitoes. What more do we gain in these examples by saying that malaria is caused by a parasite ..? I believe the common, everyday meaning of the concept of causation is entirely pragmatic. In other words, we use the word cause for events that might be controllable. In the philosophical literature controllable is the equivalent of the idea of power. In other words, the value of the concept of causation lies in its identification of where our power and control can be effective. For example, while it is true that bacteria and mosquitos follow the laws of physics, we do not usually say that malaria is caused by the laws of physics (the universal cause). That is because we can hope to control bacteria and mosquitos, but not the laws of physics.*"

Building from this observation and from the work of Shalizi (2001), Crutchfield (1994a, 1994b), Rabinowitz (2005), among others (Bickhard 2000; Bedau 1997; Campbell 1974; Goldstein 2002; Andersen et al. 2000; Kauffman 2000; Wiedermann and van Leeuwen 2002; Stannett 2003; Atay and Josty 2003; Darley 1994; Emmeche et al. 2000), we propose to discriminate between three types of emergence, depending on increasing level of 'causal' power: pattern formation, intrinsic emergence and causal emergence.

Despite the philosophical halo of the above discussion, our aim is utterly practical. In a scientific culture in which understanding is increasingly synonymous with computer modelling, we ask what forms of emergence can be studied by simulation and what we can gain from doing so. We will see that computational and 'causal' barriers are strongly related. This may lead to new insights into the limitations and future of the computer modelling of complex processes.

15.3 Formal Logic and Computation

There exists an equivalence between the workings of formal grammars, logical systems and computation (Chaitin 1997; Turing 1936; Ord 2002; Penrose 1989). All these start from some fundamental set of strings (starting symbols, axioms or input data), a set of rewriting rules (production rules, rules of inference, computer instructions), and they generate outputs (strings, theorems and computational results) which are obtained by transforming the a priori set via the rewriting rules.

In a formal system, true statements are almost always either theorems or tautologies (Kurt Gödel demonstrated that there are true statements which are not accessible from the axioms and rules of logic. These true statements are not theorems since they are not derived from the axioms). This is so because, given a set of axioms and inference rules, these statements are necessarily true (they are true for all possible scenarios and cannot be otherwise). Given a set of axioms and inference rules, these statements necessarily follow and are true for all possible scenarios and cannot be otherwise. Consequently, these statements do not provide any information about the real world (any information such a string may seem convey is a result of correspondences we see (or think we see) between the real world and the fundamental system, and is wholly dependent on our perception of these correspondences). An example clarifies the concept: the statement '*it's raining*' may be true today and may or may not be true tomorrow; it depends on its agreement with the vagaries of the real world. Assessing whether the statement is true or not provides information about the real world. Pythagoras' theorem, in contrast, is true independently of Nature's vagaries, it must be true and always will. The fact that Pythagoras' theorem is useful to us and matches our perception of reality is due to the clever choices of the basic axioms of geometry. It is because of the appropriateness of axioms collected in Euclid's work that the properties of triangles match our perception of reality.

Given Euclid's axioms and our rules of mathematical reasoning, Pythagoras' theorem is an inevitable consequence. It helps us to understand Nature better by simplifying geometrical considerations, by putting place holders in our geometrical thinking so we do not always need refer back to the axioms, and it helps us to communicate this understanding, but it does not provide any information which is not already implicit in the our axioms and rewriting rules. Theorems are transformations of information, not new information. In some sense, all the theorems of Euclidean geometry could be compressed, with no loss of information, into the basic axioms and inference rules (this is formally proved in Chaitin (1997) and is the base for Kolmogorov/Chaitin's complexity measure). It could reasonably be argued, though, that any decompressor which could reproduce the theorems of Euclidean geometry through its decompression would need to be at least as complex as a mathematician and, like a mathematician, would stand a reasonable chance of passing the Turing Test (which we discuss later).

The PCs on our desks are equivalent to a finite tape Turing Machine (TM), an abstract and general computational device commonly employed in theoretical computer science. Because the execution of a TM is equivalent to the application of production rules in a formal grammar, and to proof in a formal system (Turing 1936), it follows that the result of running a TM is equivalent to a theorem or a valid string: the results are independent of reality.

It thus also follows that the outputs of any of our computer models are similarly dictated by their initial state and the rewriting rules embodied by the program (technically, this is correct provided the PC does not allow for interaction with the outside world, see Wiedermann and van Leeuwen (2002) and van Leeuwen and Wiedermann (2001a)); a computer model transforms the information contained in its input via its coded algorithm, but does not generate information. Clearly, a model's output helps our finite mental capability to see consequences of what we coded (which at times we cannot envisage), but its truth status and relevance to the real world is limited to the truth and relevance of the user code and the input fed to the computation. No actual information about the real world is produced by a simulation. Information is generated solely by the writing of the code and the choice of the input. In this way, our choices about how we model a system are much like Euclid's choices and the comparison of the results of our simulations to what we observe in Nature tells us about the appropriateness of the rules we implement and the input we choose.

15.4 Algorithms and Physical Laws

In our perception of reality, causality manifests itself as physical laws (conversely, a physical law can represent both causal relations and mere correlations, from which it arises the philosophical dilemma behind causality. For the purpose of our discussion it is important to stress that causality can be represented only as a physical law, such as "for every action there is a corresponding equal and opposite reaction"). Our computational representation of physical laws involves algorithms which are essentially transformation rules (sequences of instructions). Since we have seen that transformation rules of this sort are constrained to produce results which are members of a set which is totally determined by these rules and the initial conditions, we need to conclude that the running of algorithms which represent physical laws can only produce similarly deterministic results. Any physical law (rule) which an algorithm can generate must already be implicit in the physical laws (rules) represented in the coded algorithm. No new physical law (or representation of it) can be generated by modelling.

When faced with the question "can genuinely novel causal laws emerge from lower level causal laws?" or "can causal laws which transcend the causal power of their constituents exist in Nature?" we can envisage two possible answers:

(i) either emergent, genuinely novel, causal laws can not exist and are only apparent and perceived as such because of the limitations in the representation we use;
(ii) or emergent causal laws must arise via natural processes which are non-algorithmic, fundamentally different from the workings of a formal logic system and consequently not computable in classical sense.

15.5 Three Levels of Emergence

In this section we examine three levels of emergence, often discussed in the literature. Our analysis focuses on the relative causal power of the emergence features they can generate.

15.5.1 Pattern Formation and Detection

Pattern formation captures the most intuitive view of emergence. The interaction of low level simple entities, leading to symmetry breaking, generates a coordinated behaviour; this is expressed by patterns which are novel and identifiable as such by an external observer. "The patterns do not appear to have specific meaning within the system, but obtain a special meaning to the observer once (and if) he/she is able to detect them. When this happens, the patterns become part of the tool-box the observer can employ to describe and study the process" (Crutchfield 1994b). Examples include the Game of Life discussed above, spiral waves in oscillating chemical reactions, convective cells in fluid flow and fractal structures in fractured media.

For the purpose of our discussion, pattern formation does not, in itself, imply causal power. Let's consider the Game of Life and the emergent gliders. Detecting their presence is useful for an observer to comprehend the effect of the local rules, to highlight the potentially universal computational capability of the system and possibly to devise a language able to compress their description (Shalizi 2001; Rabinowitz 2005). The question relevant to our discussion is whether the gliders can 'do' something or are simply 'passive' expressions of internal dynamics; can we exert any causal control on the gliders? What should we do to affect the behaviour of the gliders?

The obvious answer is that we could manipulate the Cellular Automata (CA) local rules. This however acts at the lower level (the CA cells) not at the level of the gliders. By doing so, gliders are still merely a representation of our manipulation of the local rules. Can we act on the gliders themselves? We believe that this could happen only via re-writing the CA code, that is via an external intervention and a complete redesign of the system. We will discuss this more extensively in Sect. 15.6.3. For now we suggest that pattern formation, per se, does not imply causal power.

15.5.2 Intrinsic Emergence

Intrinsic emergence refers to features which are important within the system because they confer additional functionality on the system itself. These emergent features may support global coordination-computation-behaviour like the motion of a flock of birds or stock market pricing (Crutchfield 1994b). Examples with immediate relevance to modelling are Minority Game models (Arthur 1994, 1998): agents must take local decisions on actions which result in an economic outcome but they are not able to communicate, so they have no information about other agents' behaviour. If they identify an emergent feature, providing information about the global dynamics of the population's economy, then they can use this measure to decide what actions to take (Boschetti 2005). This feature now acts as an avenue for global information processing and provides to the system the possibility for coordinated behaviour. Clearly, the agents' behaviour influences the global measure, but now the global measure affects the behaviour of the agents by determining their future actions. Self-referentiality becomes a fundamental ingredient for complex dynamics and intrinsic emergence.

Discriminating whether intrinsic emergence implies causal control is more challenging and is surely not as clear cut as for pattern formation. In a real world we could externally affect the stock market (with some sort of governmental intervention, for example) thereby changing indirectly the dynamics of the agents, who would respond to the sudden external change by altering their future behaviour. This intervention is not possible in the case of pattern formation described above, since we cannot intervene on a convective cell (for example) without acting directly on the molecules' motion. In the case of a simulation, we could affect the future behaviour of the model by changing the values of the emergent feature (market), without having to re-program the code. However, this is not fully satisfactory since, in a classic Turing Machine, no interaction with the computation is allowed and, consequently, the distinction between algorithm and input data is blurred.

15.5.3 Causal Emergence

The relation between emergence and causality has been studied under the term 'downward causation' or 'strong emergence' (Goldstein 2002; Bickhard 2000; Heylighen 1991). Roughly, 'a feature is emergent if it has some sort of causal power on lower level entities'. Like all topics involving causality, this is a subject open to considerable controversy (see Rabinowitz 2005). Here we refer to it as 'causal emergence' to highlight the fact that we employ the weaker definition of causality involving control and consequently our conclusions do not necessarily generalize to the global problem of downward causation. Another suitable name could be emergence of control.

With causal emergence we define the arising of structures on which we can exert direct control without manipulating, nor concerning ourselves with, the lower level constituents. As an example, we assume again that the ultimate cause of human behaviour lies in the biochemical process arising at a molecular and cellular level. Suppose I want to ask my friend Jim to play some music for me. I can do so by addressing him directly, for example by speaking or writing a message. Once a message is received, my friend will employ his biological machinery to accept the invitation, but I do not need to concern myself with it. I do not need to re-program complicated instructions into Jim's cellular sub-stratum. For all practical concerns, my friend acts as an entity with emergent causal power.

15.6 Modelling Causal Emergence

In the previous section we proposed to subdivide emergence into three classes depending on the causal power of the features they can generate, ranging from pattern formation, which generates features with no causal power, to intrinsic emergence, displaying limited, indirect causal power to causal emergence, empowered with full causal power.

In Sect. 15.4 we claimed that the generation of causal power cannot be modelled, since an algorithm cannot produce novel rules. If this statement is correct, then we deduce that while we can model pattern formation, and we may or may not be able to model intrinsic emergence (depending on whether we allow for interaction with data rather than instructions), we should not be able to model causal emergence. If true, this is quite a bad piece of news, since a large component of research on Complex Systems is today carried out via computer modelling and emergence is considered to be a crucial ingredient of complex systems.

This is a potentially important claim. For a claim to be meaningful, however, it needs to be relevant and falsifiable. In this section we discuss why this claim is relevant to current scientific investigation by addressing applications to biological and ecological modelling, Artificial Intelligence, Artificial Life and the mining of large scientific data sets. This will lead us along the difficult path of falsification via a variant of the Turing test, applicable to emergence processes. A full discussion of the falsification of this claim requires addressing much subtler issues of the philosophy of science and meta-mathematics which are beyond the scope of this paper, but which we touch upon briefly in the final Discussion.

15.6.1 Is This Relevant?

Pattern formation is usually considered the most trivial form of emergence. Nevertheless, its relevance to our scientific enquiry is beyond doubt. An inspiring exposition on the relevance of intrinsic emergence to the understanding of Nature can be found in Crutchfield (1994b), to which we refer the reader. Here we discuss the possible relevance of the concept of causal emergence. As mentioned above we distinguish causal emergence from Downward Causation in this work. Ample discussion of the related concept of Downward Causation can be found in Andersen et al. (2000).

Following our discussion in Sect. 15.4, the relevance of causal emergence depends essentially on whether we believe uncomputability can be found in Nature. On this topic, the scientific community is broadly divided into two groups. The first group, by far the largest, believes that uncomputability exists only in the abstract world of formal logic and pure mathematics, not in the natural world. A common justification of this view is that no example of uncomputability has so far been detected in Nature nor is there a specific need to include it in our descriptive tools. A smaller community believes that uncomputability can be found in Nature. Among these we can cite Penrose's famous claims about the super computability of the human brain (Penrose 1994, 1989; Stannett 2003; Kellett 2006). According to Penrose we can easily envisage real implementations of the abstract concept of a Turing Machine, so there is no reason to believe that uncomputability cannot be generated in

Nature. For a further discussion on this topic see also (Cooper and Odifreddi 2003; Calude et al. 1995). A natural observation for supporters of the latter view is that, if all tools enabling us to study Nature are based on computation (i.e. algorithms), then it follows that no uncomputable process can be detected. This observation leads to a possible third view of the problem, according to which Nature may or may not include uncomputable processes, but we will never be able to detect or access them because of the inherent limitation in the language we use to interpret it. We will come back to this possibility later.

15.6.2 Biological/Ecological Modelling

The idea underlying any computer modelling is to create a virtual laboratory where a researcher can perform experiments and scenario testing which would be impossible, impractical or too costly to carry out in the real world. The relevance of these experiments depends on how well the virtual laboratory resembles the real world. Nineteenth century physics has taught us that perfect accuracy is beyond our reach (Heisenberg Uncertainty Principle, for example), and this teaching is today well accepted. Nineteenth century mathematics has taught more fundamental concerns (Gödel's Theorem, for example), which, curiously, are more easily dismissed.

A considerable experience in engineering, physics and chemistry has shown immense practical benefits and, when the general limitations are carefully accounted for, has proved how useful computer modelling can be. When porting the approach to biological and ecological modelling it becomes tempting to employ the same method for studying processes like evolution, adaptation, and creation of novelty and diversity. However, we believe that these processes involve the same causal emergence we discussed above and it thus becomes necessary to ask whether the virtual laboratory has a similar functional relation to the real world to that enjoyed by physical systems. The same question can be framed as follows: to what extent can a biological agent be modelled within the same framework used to model non-living objects and processes?

To give a practical example of where the challenge may reside, it is useful to remember that a crucial concept in biological and ecological studies is the existence of multiple levels of organization (cells $\rightarrow$ organs $\rightarrow$ individuals $\rightarrow$ communities $\rightarrow$ species $\rightarrow$ ecologies, etc.).

According to our current understanding, these structures self-organize (they do not follow explicit external direction templates (Kauffman 2000)) and are linked by two-way (upward and downward) interactions. Many real world problems (ecological and renewable resource management for example) depend on our understanding (i.e., modelling) of this supposedly spontaneous generation of organization and two-way interactions. Obviously, the more complex the questions we ask, the more complicated the models we need to develop, and the more levels of organization we may need to include in the model. For example, in a fishery management problem we may want to study how individual fish organize themselves in schools or how individual fishers organize a fishing fleet. This represents one level of organization. If the specifics (or the scale) of the problem requires it, we may also need to model how schools of different fish interact, or how a school of fish interacts with a fishing fleet; this represents a second level of organization. In our model we can design a set of rules (a module) which controls the behaviour of the individual fish and vessels and a set of modules for the behaviour of fish schools and the fleet. However, if our purpose is to understand how these multiple levels arise and interact, then we would like the dynamics of the different levels to be shared or at least related. In principle we may want to code a single module (of the lower level) and see how higher levels of organization arise as a result; after all, this is what we conjecture happens in Nature.

Here, in our opinion, a fundamental discontinuity is revealed. In order to model this nesting of organization, the schools and fleets need to be more than mere patterns arising form the lower level; they need to be able to 'do' something. In particular, they need to be able to causally interact with other entities. Following our discussion in Sect. 15.6, this equates to asking whether we can exert control on the system without needing to 'refer back' or manipulate the rules controlling the individual fish and vessels. In other words, as we are able to ask our friend to play some music (without needing to concern ourselves about his 'lower level', local, biochemical rules) by merely interacting with him at a higher level similarly, we would like to be able to exert control on a school or fleet without having to concern ourselves with the lower level rules governing them. If we cannot do that, then we must conclude that the school/fleet system is merely a pattern, which we can identify and analyse, but which does not have any causal power. Thus the question is, can we exert such control?

15.6.3 *Artificial Life and Artificial Intelligence—A Turing Test for Emergence*

We believe the answer to the previous question is negative. We also believe this is merely a conjecture and that it cannot be proved. We also believe this issue is highly debatable since it mostly depends on potentially different interpretations of causal control, as we discuss in this section.

In Sect. 15.5.1 we expressed our opinion that the gliders in the Game of Life are mere patterns with no causal power and we asked ourselves whether we can interact with the gliders without re-coding their local rules. Answering this is not trivial, mostly because it depends on how much 'purpose' is placed in the original local rules.

We explain what we mean by 'purpose' with an example. Let's take a flocking model (Reynolds 1987). Birds fly in flocks by ensuring they maintain certain constraints on the position between each other. Suppose we now place an obstacle on the route of the flock. The flock will circumvent the obstacle. It thus appears that we were able to exert control on the behaviour of the flock; the flock appears to have causal power. However, we ask ourselves whether the flock has actually done anything which was not explicitly coded in the lower level rules. After all, all the flock did was to maintain flight by following a lead bird which avoided the obstacle. Is there any emergent behaviour in this? Is there any causal power which was not purposely coded.

Can we causally control the flock in any different way? Reasonable arguments can be given in both the affirmative and negative in answering this question. Interestingly, this is not particularly important. Let's consider once again my friend Jim playing some music. It is beyond ours current understanding to discriminate to what extent our verbal instruction interacts with his underlying biochemistry. Similarly, it is beyond our current knowledge to grasp how our higher level invitation (spoken request) is processed at his lower (biochemical) level for the task to be carried out. For our discussion, what matters is only our perception of causal control on Jim's behaviour. By analogy, we are led to conclude that in the Game of Life or flocking example, what matters is the perception to which we believe we can exert causal control over the higher level emergent features. Does it look as if those features possess causal control? Does it look like they do more than the limited number of behaviours purposely encoded in the local rules? Do system entities behave as if they were autonomously interacting with external processes and respond accordingly?

These new questions have the flavour of a 'Turing test for emergence'. Famously, the Turing test (for related variations, see the series of papers contained in *Minds and Machines* (Saygin et al. 2000; Sterrett 2000)) was designed to circumvent the difficult question of defining what intelligence is and to detect when a computer can be said to have achieved it (one of the original purposes of Artificial Intelligence at its very conception). Turing suggested as testing whether a human (an intelligent agent) was able to discriminate blindly between another human (another intelligent agent) and a computer. Should he/she not be able to, then we should conclude than the computer and the human act as intelligently as each other, and therefore they are both similarly intelligent. Following an analogous reasoning, we conceive an 'emergent' version of the test and we ask whether a process empowered with autonomous causal emergent properties (a human) can discriminate between another causal emergent process and a computer program. Should he/she not be able to do so, then we should conclude that the computer displays causal emergence.

We are not actually suggesting that the test be carried out in earnest. Rather, we would like to refer to and build upon the vast body of work (both conceptual and practical) carried out on the Turing test over several decades and extend some of the conclusions which may be relevant to the study of emergent processes and computer modelling. In this regard, notice that intelligence is itself often considered an emergent feature of the processing occurring in a nervous system. If we accept this view, then the 'Turing test for emergence' can be seen as a generalization of the traditional Turing test. Consequently extending the discussion of the traditional Turing test to emergence becomes more than merely exploiting an imaginary analogy.

The traditional Turing test has been subjected to considerable theoretical discussion and criticism. Nevertheless, practical implementations of the Turing test are carried out annually in the form of the Loebner prize (Wikipedia 2007). So far, it is widely accepted that improvement in the test performance over the years has not been particularly significant and ‘passing the test’ does not seem to be a likely short term outcome. The entire artificial intelligence community has, therefore, revisited its own role, scope and measure of success. Far from being a proof, this observation does somehow reinforce our conjecture that modelling causal emergence via computer simulation should, at the very least, not be taken for granted.

On a more positive side, this suggests a reason for the Complex System Sciences (CSS) community building more closely on the extensive experience accumulated along the difficult path followed by artificial intelligence. After a few decades of pessimism, a new breeze of optimism can be felt in both the artificial life and artificial intelligence community. This renewed confidence is not based on the infrastructure of logical programming or the complications of expert systems (as in the past), nor on hopes of super computability brought to us by quantum computing. Rather it depends on more down-to-earth, often biologically inspired, approaches. As an example, in a series of papers (Wiedermann and van Leeuwen 2002; Verbaan et al. 2004; van Leeuwen and Wiedermann 2000, 2001a, 2001b, 2003; Wiedermann 2000), van Leeuwen and Wiedermann show formally that agents interacting with their environment have computational capabilities which supersede classic computation. There are a number of reasons why interacting agents can achieve these acrobatics: they run indefinitely (as long as the agent is alive), they continuously receive input from a (potentially infinite) environment and from other agents (unlike a classic machine for which the input is determined and fixed at the beginning of the calculations), they can use the local environment to store and retrieve data and they can adapt to the environment. In particular, the agents’ adaptation to their environment means that the ‘algorithm’ within the agents can be updated constantly and in van Leeuwen and Wiedermann (2003) it is shown how super computability can arise from the very evolution of the agents. Also, in an interactive machine, the traditional distinction between data, memory and algorithm does not apply, which results in more dynamical and less specifiable computational outcomes (Milner 1993). Other classes of relatively down-to-earth machines which seem to guarantee to break classic computation barriers include fuzzy Turing machines (Wiedermann 2000).

Today human-computer interactions are standard in a large number of applications. Usually, these are seen as enhancing human capabilities by providing the fast computation resources available to electronic machines. Should we see the interaction in the opposite direction, as humans enhance the computational capabilities of electronic machines? In van Leeuwen and Wiedermann (2000) it is speculated that today personal computers, connected via the web to thousands of machines world wide, receiving inputs via various sensors and on-line instructions from users, are already beyond classic computers. Today sensors monitor several aspects of the environment routinely and some have even been installed on animals in the wilderness (Simonite 2005). Can we envisage a network computing system, in which agents (computers) interact with the environment via analog sensors, receive data from living beings, and instructions from humans to deal with unexpected situations? Could this be the way forward to understand emergence?

More intriguingly, could these systems potentially already sit on our desks?

15.6.4 Data Explosion and Scientific Data Mining

In a recent issue of Nature (Butler 2006; Muggleton 2006; Szalay and Gray 2006), the picture was drawn of a near future when improved instrumentation and extensive sensing will provide us with exponentially increasing quantities of data for scientific enquiry. This implies more information but at a considerable cost. It promises more and better information about a vast range of things, from space to ocean depths, from ecologies to the human body, from genomes to social behaviour. However, the data explosion may go beyond our ability to process and analyse it. Unravelling new mysteries of Nature will then be jeopardized by something as mundane as lack of time and resources. It is hypothesized that this will be circumvented by clever software able to supervise the instrumentation, detection of new interesting patterns and possibly use rule extraction algorithms that uncover new processes and biophysical or social laws; a very difficult task, but (supposedly) merely a technological one.

This picture relies on 2 assumptions:

(i) that all natural processes we may wish to study or detect are algorithmic;
(ii) that the process which allows us to understand and study Nature is also algorithmic.

Neither of these assumptions has been proved and both are open to debate. The first statement has been discussed above. The second one requires some clarification. First, a computational system which scans a data set in order to find patterns of interest must be algorithmic, by definition. Similarly, a system which, upon detecting a pattern, performs some rule extraction in order to attract our attention and suggest an interpretation also needs to be algorithmic. It seems evident that any algorithm capable of sifting through a stream of data and picking out just those novel patterns which are of interest to a human being, is more than a few steps along the way to passing the Turing test. Similarly, the second of the two systems bears a remarkable resemblance to the Halting Problem. An algorithmic system cannot, by definition, process a non-recursive language, from which it follows that if Nature displays a non-algorithmic process, this will not be detected by a fully automated computational system.

It is interesting to note that the rigors of formal logic apply not only to computational systems, but to the broader scientific method as well. The scientific method requires experiments to be reproducible. This implies that an experiment needs to follow a quite detailed and rigorous procedure in order to be replicated by different observers under inevitably different experimental settings. Basically, an experiment is reduced to an algorithm (Stannett 2003, page 122), and consequently scientific experimentation suffers the very same limitation of formal logic and computer systems, and thus is, by itself, unable to detect truly emergent processes. Curiously, the same desire for a rigorous, quasi-algorithmic approach affects scientific communication, with scientific journals often requiring a quasi-algorithmic way of writing. However, it is often suspected that the large leaps in scientific understanding are fired by a brilliance which may be non-algorithmic. While further considerable work needs to be done to understand this creative process, it seems that over-relying on formal logic not only to model, but also to detect and analyse Nature may come with the risky consequence of preventing us from seeing the very processes we want to discover.

15.7 Conclusions

Our aim is by no means to suggest that computer modelling is a purposeless activity. Rather, that clarity is needed to discriminate the means (computer modelling as a tool) from the aim (acquiring knowledge about Nature). In this framework, confusing the means with the aim equates to carrying out a scientific program (including experimental and formal analysis) in the virtual world of a computer model as if this was the 'real world' and then extend the 'virtual' results to the 'real' natural world, under the assumption that the two are, to some degree, isomorphic. Here the old Chinese saying "if all you have is a hammer, everything looks like a nail" nicely highlights possible dangers and could be translated as 'if all you have is a computer, everything looks computational'.

We can thus summarize our proposed guidelines as follows:

(i) Care should be used to discriminate among: the processes which are 'naturally' amenable to computer modelling; the processes which are numerically or theoretically intractable (large combinatorial problems, NP-hard problems, chaotic problems) but for which useful approximations can be found (either in terms of non optimal solutions or large scale approximations); and processes which may be fundamentally intractable.

(ii) The widely accepted conjecture that intractable problems do not exist in Nature should (at least) be carefully studied, rather than accepted dogmatically.

(iii) The rigors of the algorithmic approach do not apply only to the world of formal systems and computer languages. Scientific investigation (the iterative testing of hypotheses) is also subject to these constraints due to its algorithmic nature. Recently, a new scientific tendency is to call for a more free and creative way of reporting and discussing science. Complex System Science, which naturally mixes experts ranging from pure mathematics to social science, seems to be in a particularly fortunate development for thorough exploration of the potential for reintroduction of artistic and other creative contributions to science.

Acknowledgements This research was carried out as a part of the CSIRO Emergence Interaction Task, http://www.per.marine.csiro.au/staff/Fabio.Boschetti/CSS_emergence.htm.

References

Andersen, P. B., Emmeche, C., Finnemann, N. O., & Christiansen, P. V. (2000). *Downward causation*. Aarhus: Aarhus University Press.

Arthur, W. (1994). Inductive behaviour and bounded rationality. *The American Economic Review*, *84*, 406–411.

Arthur, W. (1998). Modeling market mechanism with evolutionary games. *Europhysics News*, *29*, 51–54.

Atay, F., & Josty, J. (2003). *On the emergence of complex systems on the basis of the coordination of complex behaviors of their elements* (Santa Fe Institute Working Paper, 04-02-005).

Bedau, M. A. (1997). Weak emergence. In J. Tomberlin (Ed.), *Philosophical perspectives: mind, causation, and world* (Vol. 11, pp. 375–399). Oxford: Blackwell.

Bickhard, M. H. (2000). Emergence. In P. B. Andersen, C. Emmeche, N. O. Finnemann, & P. V. Christiansen (Eds.), *Downward causation* (pp. 322–348). Aarhus: University of Aarhus Press.

Boschetti, F. (2005). Improved resource exploitation by collective intelligence. In A. Zerger & R. M. Argent (Eds.), *MODSIM05: international congress on modelling and simulation: advances and applications for management and decision making* (pp. 518–523). Canberra: Modelling and Simulation Society of Australia and New Zealand.

Butler, D. (2006). 2020 computing: everything, everywhere. *Nature*, *440*(7083), 402–405.

Calude, C., Campbell, D. I., Svozil, K., & Stefanescu, D. (1995). Strong determinism vs. computability. In W. Depauli-Schimanovich, E. Koehler, & F. Stadler (Eds.), *Downward causation* (pp. 115–131). Dordrecht: Kluwer Academic.

Campbell, D. T. (1974). Downward causation in hierarchically organized biological systems. In F. Ayala & T. Dobzhansky (Eds.), *Studies in the philosophy of biology* (pp. 179–186). Berkeley: University of California Press.

Chaitin, G. (1997). *The limits of mathematics: a course on information theory & limits of formal reasoning*. New York: Springer.

Cooper, B., & Odifreddi, P. (2003). Incomputability in nature. In S. B. Cooper & S. S. Goncharov (Eds.), *Computability and models* (pp. 137–160). Dordrecht: Kluwer Academic.

Corning, P. (2005). The re-emergence of emergence: a venerable concept in search of a theory. In *Holistic Darwinism: synergy, cybernetics, and the bioeconomics of evolution*, Chicago: Univ. of Chicago Press.

Crutchfield, J. (1994a). Is anything ever new? Considering emergence. In G. Cowan, D. Pines, & D. Melzner (Eds.), *SFI series in the sciences of complexity: Vol. XIX. Complexity: metaphors, models, and reality* (pp. 479–497). Redwood City: Addison-Wesley.

Crutchfield, J. P. (1994b). The calculi of emergence: computation, dynamics, and induction. *Physica D*, *75*, 11–54.

Darley, V. (1994). Emergent phenomena and complexity. In R. Brooks & P. Maes (Eds.), *Proceedings of artificial life IV*. Cambridge: MIT Press.

Emmeche, C., Koppe, S., & Stjernfelt, F. (2000). Levels, emergence, and three versions of downward causation. In P. B. Andersen, C. Emmeche, N. O. Finnemann, & P. V. Christiansen (Eds.), *Downward causation* (pp. 13–34). Aarhus: University of Aarhus Press.

Gardner, M. (1970). Mathematical games: the fantastic combinations of John Conway's new solitaire game "Life". *Scientific American*, *223*, 120–123.

Goldstein, J. (2002). The singular nature of emergent levels: suggestions for a theory of emergence. *Nonlinear Dynamics, Psychology, and Life Sciences*, *6*(4).

Heylighen, F. (1991). Modelling emergence. *World Futures*, *31*, 89–104. Special Issue on Emergence, G. Kampis, editor.

Kauffman, S. (2000). *Investigations*. London: Oxford University Press.

Kellett, O. (2006). *A multi-faceted attack on the busy beaver problem*. Masters thesis, Rensselaer Polytechnic Institute, Troy, New York.

Laughlin, R., & Pines, D. (2000). The theory of everything. *Proceedings of the National Academy of Sciences*, *97*(1), 28–31.

Milner, R. (1993). Elements of interaction. *Communications of the ACM*, *36*(1), 78–89.

Muggleton, S. (2006). 2020 computing: exceeding human limits. *Nature*, *440*(7083), 409–410.

Ord, T. (2002). *Hypercomputation: computing more than the Turing machine* (Technical report). University of Melbourne.

Pattee, H. (1997). Causation, control, and the evolution of complexity. In P. B. Andersen, C. Emmeche, N. O. Finnemann, & P. V. Christiansen (Eds.), *Downward causation*. Aarhus: University of Aarhus Press.

Pearl, J. (2000). *Causality: models, reasoning and inference*. Cambridge: MIT Press.

Penrose, R. (1989). *The emperor's new mind: concerning computers, minds, and the laws of physics*. London: Vintage.

Penrose, R. (1994). *Shadows of the mind: a search for the missing science of consciousness*. Oxford: Oxford University Press.

Rabinowitz, N. (2005). *Emergence: an algorithmic formulation*. Ph.D. thesis, The University of Western Australia.

Reynolds, C. W. (1987). Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, *21*(4), 25–34.

Saygin, A., Cicekli, I., & Akman, V. (2000). Turing test: 50 years later. *Minds and Machines*, *10*(4), 463–518.

Shalizi, C. (2001). *Causal architecture, complexity and self-organization in time series and cellular automata*. Ph.D. thesis, University of Michigan.

Simonite, T. (2005). Seals net data from cold seas. *Nature*, *438*, 402–403.

Stannett, M. (2003). Computation and hypercomputation. *Minds and Machines*, *13*(1), 115–153.

Sterrett, S. (2000). Turing's two tests for intelligence. *Minds and Machines*, *10*(4), 541–559.

Szalay, A., & Gray, J. (2006). 2020 computing: science in an exponential world. *Nature*, *440*(7083), 409–410.

Turing, M. A. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, *42*, 230–265.

van Leeuwen, J., & Wiedermann, J. (2000). *The Turing machine paradigm in contemporary computing* (Technical Report UU-CS-2000-33). Institute of Information and Computing Sciences, Utrecht University.

van Leeuwen, J., & Wiedermann, J. (2001a). Beyond the Turing limit—evolving interactive systems. In L. Pacholski & P. Ruzicka (Eds.), *Theory and practice of informatics* (pp. 90–109). Berlin: Springer.

van Leeuwen, J., & Wiedermann, J. (2001b). *A computational model of interaction in embedded systems* (Technical Report UU-CS-2001-02). Institute of Information and Computing Sciences, Utrecht University.

van Leeuwen, J., & Wiedermann, J. (2003). The emergent computational potential of evolving artificial living systems. *AI Communications*, *15*, 205–215.

Verbaan, P., van Leeuwen, J., & Wiedermann, J. (2004). Lineages of automata—a model for evolving interactive systems. In J. Karhumaki, H. Maurer, G. Paun, & G. Rozenberg (Eds.), *Theory is forever* (pp. 268–281). Berlin: Springer.

Wiedermann, J. (2000). *Fuzzy computations are more powerful than crisp ones*. (Technical Report V-828). Prague University.

Wiedermann, J., & van Leeuwen, J. (2002). The emergent computational potential of evolving artificial living systems. *AI Communications*, *15*(4), 205–216.

Wikipedia (2007). Loebner prize—Wikipedia, the free encyclopedia, available at http://en.wikipedia.org/wiki/loebner_prize. Online; accessed 25 January 2007.

Index