Preface

Natural Computing is the field of research that investigates human-designed computing inspired by nature as well as computing taking place in nature, that is, it investigates models and computational techniques inspired by nature, and also it investigates, in terms of information processing, phenomena taking place in nature.

Examples of the first strand of research include neural computation inspired by the functioning of the brain; evolutionary computation inspired by Darwinian evolution of species; cellular automata inspired by intercellular communication; swarm intelligence inspired by the behavior of groups of organisms; artificial immune systems inspired by the natural immune system; artificial life systems inspired by the properties of natural life in general; membrane computing inspired by the compartmentalized ways in which cells process information; and amorphous computing inspired by morphogenesis. Other examples of natural-computing paradigms are quantum computing and molecular computing, where the goal is to replace traditional electronic hardware, by, for example, bioware in molecular computing. In quantum computing, one uses systems small enough to exploit quantum-mechanical phenomena to perform computations and to perform secure communications more efficiently than classical physics, and, hence, traditional hardware allows. In molecular computing, data are encoded as biomolecules and then tools of molecular biology are used to transform the data, thus performing computations.

The second strand of research, computation taking place in nature, is represented by investigations into, among others, the computational nature of self-assembly, which lies at the core of the nanosciences; the computational nature of developmental processes; the computational nature of biochemical reactions; the computational nature of bacterial communication; the computational nature of brain processes; and the systems biology approach to bionetworks where cellular processes are treated in terms of communication and interaction, and, hence, in terms of computation.

Research in natural computing is genuinely interdisciplinary and forms a bridge between the natural sciences and computer science. This bridge connects the two, both at the level of information technology and at the level of fundamental research. Because of its interdisciplinary character, research in natural computing covers a whole spectrum of research methodologies ranging from pure theoretical research, algorithms, and software applications to experimental laboratory research in biology, chemistry, and physics.

Computer Science and Natural Computing

A preponderance of research in natural computing is centered in computer science. The spectacular progress in Information and Communication Technology (ICT) is highly supported by the evolution of computer science, which designs and develops the instruments needed for this progress: computers, computer networks, software methodologies, etc. As ICT has such a tremendous impact on our everyday lives, so does computer science.

However, there is much more to computer science than ICT: it is the science of information processing and, as such, a fundamental science for other disciplines. On one hand, the only common denominator for research done in such diverse areas of computer science is investigating various aspects of information processing. On the other hand, the adoption of Information and Information Processing as central notions and thinking habit has been an important development in many disciplines, biology and physics being prime examples. For these scientific disciplines, computer science provides not only instruments but also a way of thinking.

We are now witnessing exciting interactions between computer science and the natural sciences. While the natural sciences are rapidly absorbing notions, techniques, and methodologies intrinsic to information processing, computer science is adapting and extending its traditional notion of computation, and computational techniques, to account for computation taking place in nature around us. Natural Computing is an important catalyst for this two-way interaction, and this handbook constitutes a significant record of this development.

The Structure of the Handbook

Natural Computing is both a well-established research field with a number of classical areas, and a very dynamic field with many more recent, novel research areas. The field is vast, and so it is quite usual that a researcher in a specific area does not have sufficient insight into other areas of Natural Computing. Also, because of its dynamic development and popularity, the field constantly attracts more and more scientists who either join the active research or actively follow research developments.

Therefore, the goal of this handbook is two-fold:

(i) to provide an authoritative reference for a significant and representative part of the research in Natural Computing, and
(ii) to provide a convenient gateway to Natural Computing for motivated newcomers to this field.

The implementation of this goal was a challenge because this field and its literature are vast — almost all of its research areas have an extensive scientific literature, including specialized journals, book series, and even handbooks. This implies that the coverage of the whole field in reasonable detail and within a reasonable number of pages/volumes is practically impossible.

Thus, we decided to divide the presented material into six areas. These areas are by no means disjoint, but this division is convenient for the purpose of providing a representative picture of the field — representative with respect to the covered research topics and with respect to a good balance between classical and emerging research trends.

Each area consists of individual chapters, each of which covers a specific research theme. They provide necessary technical details of the described research, however they are self-contained and of an expository character, which makes them accessible for a broader audience. They also provide a general perspective, which, together with given references, makes the chapters valuable entries into given research themes.

This handbook is a result of the joint effort of the handbook editors, area editors, chapter authors, and the Advisory Board. The choice of the six areas by the handbook editors in consultation with the Advisory Board, the expertise of the area editors in their respective

areas, the choice by the area editors of well-known researchers as chapter writers, and the peer-review for individual chapters were all important factors in producing a representative and reliable picture of the field. Moreover, the facts that the Advisory Board consists of 68 eminent scientists from 20 countries and that there are 105 contributing authors from 21 countries provide genuine assurance for the reader that this handbook is an authoritative and up-to-date reference, with a high level of significance and accuracy.

Handbook Areas

The material presented in the handbook is organized into six areas: Cellular Automata, Neural Computation, Evolutionary Computation, Molecular Computation, Quantum Computation, and Broader Perspective.

Cellular Automata

Cellular automata are among the oldest models of computation, dating back over half a century. The first cellular automata studies by John von Neumann in the late 1940s were biologically motivated, related to self-replication in universal systems. Since then, cellular automata gained popularity in physics as discrete models of physical systems, in computer science as models of massively parallel computation, and in mathematics as discrete-time dynamical systems. Cellular automata are a natural choice to model real-world phenomena since they possess several fundamental properties of the physical world: they are massively parallel, homogeneous, and all interactions are local. Other important physical constraints such as reversibility and conservation laws can be added as needed, by properly choosing the local update rule. Computational universality is common in cellular automata, and even starkly simple automata are capable of performing arbitrary computation tasks. Because cellular automata have the advantage of parallelism while obeying natural constraints such as locality and uniformity, they provide a framework for investigating realistic computation in massively parallel systems. Computational power and the limitations of such systems are most naturally investigated by time- and space-constrained computations in cellular automata. In mathematics — in terms of symbolic dynamics — cellular automata are viewed as endomorphisms of the full shift, that is, transformations that are translation invariant and continuous in the product topology. Interesting questions on chaotic dynamics have been studied in this context.

Neural Computation

Artificial neural networks are computer programs, loosely modeled after the functioning of the human nervous system. There are neural networks that aim to gain understanding of biological neural systems, and those that solve problems in artificial intelligence without necessarily creating a model of a real biological system. The more biologically oriented neural networks model the real nervous system in increasing detail at all relevant levels of information processing: from synapses to neurons to interactions between modules of interconnected neurons. One of the major challenges is to build artificial brains. By reverse-engineering the mammalian brain in silicon, the aim is to better understand the functioning of the (human)

brain through detailed simulations. Neural networks that are more application-oriented tend to drift further apart from real biological systems. They come in many different flavors, solving problems in regression analysis and time-series forecasting, classification, and pattern recognition, as well as clustering and compression. Good old multilayered perceptrons and self-organizing maps are still pertinent, but attention in research is shifting toward more recent developments, such as kernel methods (including support vector machines) and Bayesian techniques. Both approaches aim to incorporate domain knowledge in the learning process in order to improve prediction performance, e.g., through the construction of a proper kernel function or distance measure or the choice of an appropriate prior distribution over the parameters of the neural network. Considerable effort is devoted to making neural networks efficient so that large models can be learned from huge databases in a reasonable amount of time. Application areas include, among many others, system dynamics and control, finance, bioinformatics, and image analysis.

Evolutionary Computation

The field of evolutionary computation deals with algorithms gleaned from models of organic evolution. The general aim of evolutionary computation is to use the principles of nature's processes of natural selection and genotypic variation to derive computer algorithms for solving hard search and optimization tasks. A wide variety of instances of evolutionary algorithms have been derived during the past fifty years based on the initial algorithms, and we are now witnessing astounding successes in the application of these algorithms: their fundamental understanding in terms of theoretical results; understanding algorithmic principles of their construction; combination with other techniques; and understanding their working principles in terms of organic evolution. The key algorithmic variations (such as genetic algorithms, evolution strategies, evolutionary programming, and genetic programming) have undergone significant developments over recent decades, and have also resulted in very powerful variations and recombinations of these algorithms. Today, there is a sound understanding of how all of these algorithms are instances of the generic concept of an evolutionary search approach. Hence the generic term "evolutionary algorithm" is nowadays being used to describe the generic algorithm, and the term "evolutionary computation" is used for the field as a whole. Thus, we have observed over the past fifty years how the field has integrated the various independently developed initial algorithms into one common principle. Moreover, modern evolutionary algorithms benefit from their ability to adapt and self-adapt their strategy parameters (such as mutation rates, step sizes, and search distributions) to the needs of the task at hand. In this way, they are robust and flexible metaheuristics for problem-solving even without requiring too much special expertise from their users. The feature of self-adaptation illustrates the ability of evolutionary principles to work on different levels at the same time, and therefore provides a nice demonstration of the universality of the evolutionary principle for search and optimization tasks. The widespread use of evolutionary computation reflects these capabilities.

Molecular Computation

Molecular computing is an emergent interdisciplinary field concerned with programming molecules so that they perform a desired computation, or fabricate a desired object, or

control the functioning of a specific molecular system. The central idea behind molecular computing is that data can be encoded as (bio)molecules, e.g., DNA strands, and tools of molecular science can be used to transform these data. In a nutshell, a molecular program is just a collection of molecules which, when placed in a suitable substrate, will perform a specific function (execute the program that this collection represents). The birth of molecular computing is often associated with the 1994 breakthrough experiment by Leonard Adleman, who solved a small instance of a hard computational problem solely by manipulating DNA strands in test tubes. Although initially the main effort of the area was focused on trying to obtain a breakthrough in the complexity of solving hard computational problems, this field has evolved enormously since then. Among the most significant achievements of molecular computing have been contributions to understanding some of the fundamental issues of the nanosciences. One notable example among them is the contribution to the understanding of self-assembly, a central concept of the nanosciences. The techniques of molecular programming were successfully applied in experimentally constructing all kinds of molecular-scale objects or devices with prescribed functionalities. Well-known examples here are self-assembly of Sierpinski triangles, cubes, octahedra, DNA-based logic circuits, DNA "walkers" that move along a track, and autonomous molecular motors. A complementary approach to understanding bioinformation and computation is through studying the information-processing capabilities of cellular organisms. Indeed, cells and nature "compute" by "reading" and "rewriting" DNA through processes that modify DNA (or RNA) sequences. Research into the computational abilities of cellular organisms has the potential to uncover the laws governing biological information, and to enable us to harness the computational power of cells.

Quantum Computation

Quantum computing has been discussed for almost thirty years. The theory of quantum computing and quantum information processing is simply the theory of information processing with a classical notion of information replaced by its quantum counterpart. Research in quantum computing is concerned with understanding the fundamentals of information processing on the level of physical systems that realize/implement the information. In fact, quantum computing can be seen as a quest to understand the fundamental limits of information processing set by nature itself. The mathematical description of quantum information is more complicated than that of classical information — it involves the structure of Hilbert spaces. When describing the structure behind known quantum algorithms, this reduces to linear algebra over complex numbers. The history of quantum algorithms spans the last fifteen years, and some of these algorithms are extremely interesting, and even groundbreaking — the most remarkable are Shor's factorization in polynomial time and Grover's search algorithm. The nature of quantum information has also led to the invention of novel cryptosystems, whose security is not based on the complexity of computing functions, but rather on the physical properties of quantum information. Quantum computing is now a well-established discipline, however implementation of a large-scale quantum computer continues to be extremely challenging, even though quantum information processing primitives, including those allowing secure cryptography, have been demonstrated to be practically realizable.

Broader Perspective

In contrast to the first five areas focusing on more-established themes of natural computing, this area encompasses a perspective that is broader in several ways. First, the reader will find here treatments of certain well-established and specific techniques inspired by nature (e.g., simulated annealing) not covered in the other five areas. Second, the reader will also find application-centered chapters (such as natural computing in finance), each covering, in one chapter, a collection of natural computing methods, thus capturing the impact of natural computing as a whole in various fields of science or industry. Third, some chapters are full treatments of several established research fields (such as artificial life, computational systems biology, evolvable hardware, and artificial immune systems), presenting alternative perspectives and cutting across some of the other areas of the handbook, while introducing much new material. Other elements of this area are fresh, emerging, and novel techniques or perspectives (such as collision-based computing, nonclassical computation), representing the leading edge of theories and technologies that are shaping possible futures for both natural computing and computing in general. The contents of this area naturally cluster into two kinds (sections), determined by the essential nature of the techniques involved. These are "Nature-Inspired Algorithms" and "Alternative Models of Computation". In the first section, "Nature-Inspired Algorithms", the focus is on algorithms inspired by natural processes realized either through software or hardware or both, as additions to the armory of existing tools we have for dealing with well-known practical problems. In this section, we therefore find application-centered chapters, as well as chapters focusing on particular techniques, not otherwise dealt with in other areas of the handbook, which have clear and proven applicability. In the second section, "Alternative Models of Computation", the emphasis changes, moving away from specific applications or application areas, toward more far-reaching ideas. These range from developing computational approaches and "computational thinking" as fundamental tools for the new science of systems biology to ideas that take inspiration from nature as a platform for suggesting entirely novel possibilities of computing.

Handbook Chapters

In the remainder of this preface we will briefly describe the contents of the individual chapters. These chapter descriptions are grouped according to the handbook areas where they belong and given in the order that they appear in the handbook. This section provides the reader with a better insight into the contents, allowing one to design a personal roadmap for using this handbook.

Cellular Automata

This area is covered by nine chapters.

The first chapter, "Basic Concepts of Cellular Automata", by Jarkko J. Kari, reviews some classical results from the theory of cellular automata, relations between various concepts of injectivity and surjectivity, and some basic dynamical system concepts related to chaos in cellular automata. The classical results discussed include the celebrated Garden-of-Eden and Curtis–Hedlund–Lyndon theorems, as well as the balance property of surjective cellular

automata. All these theorems date back to the 1960s. The results are provided together with examples that illustrate proof ideas. Different variants of sensitivity to initial conditions and mixing properties are introduced and related to each other. Also undecidability results concerning cellular automata are briefly discussed.

A popular mathematical approach is to view cellular automata as dynamical systems in the context of symbolic dynamics. Several interesting results in this area were reported as early as 1969 in the seminal paper by G.A. Hedlund, and still today this research direction is among the most fruitful sources of theoretical problems and new results. The chapter "Cellular Automata Dynamical Systems", by Alberto Dennunzio, Enrico Formenti, and Petr Kůrka, reviews some recent developments in this field. Recent research directions considered here include subshifts attractors and signal subshifts, particle weight functions, and the slicing construction. The first two concern one-dimensional cellular automata and give precise descriptions of the limit behavior of large classes of automata. The third one allows one to view two-dimensional cellular automata as one-dimensional systems. In this way combinatorial complexity is decreased and new results can be proved.

Programming cellular automata for particular tasks requires special techniques. The chapter "Algorithmic Tools on Cellular Automata", by Marianne Delorme and Jacques Mazoyer, covers classical algorithmic tools based on signals. Linear signals as well as signals of nonlinear slope are discussed, and basic transformations of signals are addressed. The chapter provides results on using signals to construct functions in cellular automata and to implement arithmetic operations on segments. The methods of folding the space–time, freezing, and clipping are also introduced.

The time-complexity advantage gained from parallelism under the locality and uniformity constraints of cellular automata can be precisely analyzed in terms of language recognition. The chapter "Language Recognition by Cellular Automata", by Véronique Terrier, presents results and questions about cellular automata complexity classes and their relationships to other models of computations. Attention is mainly directed to real-time and linear-time complexity classes, because significant benefits over sequential computation may be obtained at these low time complexities. Both parallel and sequential input modes are considered. Separate complexity classes are given also for cellular automata with one-way communications and two-way communications.

The chapter "Computations on Cellular Automata", by Jacques Mazoyer and Jean-Baptiste Yunès, continues with the topic of algorithmic techniques in cellular automata. This chapter uses the basic tools, such as signals and grids, to build natural implementations of common algorithms in cellular automata. Examples of implementations include real-time multiplication of integers and the prime number sieve. Both parallel and sequential input and output modes are discussed, as well as composition of functions and recursion.

The chapter "Universalities in Cellular Automata", by Nicolas Ollinger, is concerned with computational universalities. Concepts of universality include Turing universality (the ability to compute any recursive function) and intrinsic universality (the ability to simulate any other cellular automaton). Simulations of Boolean circuits in the two-dimensional case are explained in detail in order to achieve both kinds of universality. The more difficult one-dimensional case is also discussed, and seminal universal cellular automata and encoding techniques are presented in both dimensions. A detailed chronology of important papers on universalities in cellular automata is also provided.

A cellular automaton is reversible if every configuration has only one previous configuration, and hence its evolution process can be traced backward uniquely. This naturally

corresponds to the fundamental time-reversibility of the microscopic laws of physics. The chapter "Reversible Cellular Automata", by Kenichi Morita, discusses how reversible cellular automata are defined, as well as their properties, how they are designed, and their computing abilities. After providing the definitions, the chapter surveys basic properties of injectivity and surjectivity. Three design methods of reversible cellular automata are provided: block rules, partitioned, and second-order cellular automata. Then the computational power of reversible cellular automata is discussed. In particular, simulation methods of irreversible cellular automata, reversible Turing machines, and some other universal systems are given to clarify universality of reversible cellular automata. In spite of the strong constraint of reversibility, it is shown that reversible cellular automata possess rich information processing capabilities, and even very simple ones are computationally universal.

A conservation law in a cellular automaton is a statement of the invariance of a local and additive energy-like quantity. The chapter "Conservation Laws in Cellular Automata", by Siamak Taati, reviews the basic theory of conservation laws. A general mathematical framework for formulating conservation laws in cellular automata is presented and several characterizations are summarized. Computational problems regarding conservation laws (verification and existence problems) are discussed. Microscopic explanations of the dynamics of the conserved quantities in terms of flows and particle flows are explored. The related concept of dissipating energy-like quantities is also discussed.

The chapter "Cellular Automata and Lattice Boltzmann Modeling of Physical Systems", by Bastien Chopard, considers the use of cellular automata and related lattice Boltzmann methods as a natural modeling framework to describe and study many physical systems composed of interacting components. The theoretical basis of the approach is introduced and its potential is illustrated for several applications in physics, biophysics, environmental science, traffic models, and multiscale modeling. The success of the technique can be explained by the close relationship between these methods and a mesoscopic abstraction of many natural phenomena.

Neural Computation

This area is covered by ten chapters.

Spiking neural networks are inspired by recent advances in neuroscience. In contrast to classical neural network models, they take into account not just the neuron's firing rate, but also the time moment of spike firing. The chapter "Computing with Spiking Neuron Networks", by Hélène Paugam-Moisy and Sander Bohte, gives an overview of existing approaches to modeling spiking neural neurons and synaptic plasticity, and discusses their computational power and the challenge of deriving efficient learning procedures.

Image quality assessment aims to provide computational models to predict the perceptual quality of images. The chapter "Image Quality Assessment — A Multiscale Geometric Analysis-Based Framework and Examples", by Xinbo Gao, Wen Lu, Dacheng Tao, and Xuelong Li, introduces the fundamentals and describes the state of the art in image quality assessment. It further proposes a new model, which mimics the human visual system by incorporating concepts such as multiscale analysis, contrast sensitivity, and just-noticeable differences. Empirical results clearly demonstrate that this model resembles subjective perception values and reflects the visual quality of images.

Neurofuzzy networks have the important advantage that they are easy to interpret. When applied to control problems, insight about the process characteristics at different operating regions can be easily obtained. Furthermore, nonlinear model predictive controllers can be developed as a nonlinear combination of several local linear model predictive controllers that have analytical solutions. Through several applications, the chapter "Nonlinear Process Modelling and Control Using Neurofuzzy Networks", by Jie Zhang, demonstrates that neurofuzzy networks are very effective in the modeling and control of nonlinear processes.

Similar to principal component and factor analysis, independent component analysis is a computational method for separating a multivariate signal into additive subcomponents. Independent component analysis is more powerful: the latent variables corresponding to the subcomponents need not be Gaussian and the basis vectors are typically nonorthogonal. The chapter "Independent Component Analysis", by Seungjin Choi, explains the theoretical foundations and describes various algorithms based on those principles.

Neural networks has become an important method for modeling and forecasting time series. The chapter "Neural Networks for Time-Series Forecasting", by G. Peter Zhang, reviews some recent developments (including seasonal time-series modeling, multiperiod forecasting, and ensemble methods), explains when and why they are to be preferred over traditional forecasting models, and also discusses several practical data and modeling issues.

Support vector machines have been extensively studied and applied in many domains within the last decade. Through the so-called kernel trick, support vector machines can efficiently learn nonlinear functions. By maximizing the margin, they implement the principle of structural risk minimization, which typically leads to high generalization performance. The chapter "SVM Tutorial — Classification, Regression and Ranking", by Hwanjo Yu and Sungchul Kim, describes these underlying principles and discusses support vector machines for different learning tasks: classification, regression, and ranking.

It is well known that single-hidden-layer feedforward networks can approximate any continuous target function. This still holds when the hidden nodes are automatically and randomly generated, independent of the training data. This observation opened up many possibilities for easy construction of a broad class of single-hidden-layer neural networks. The chapter "Fast Construction of Single-Hidden-Layer Feedforward Networks", by Kang Li, Guang-Bin Huang, and Shuzhi Sam Ge, discusses new ideas that yield a more compact network architecture and reduce the overall computational complexity.

Many recent experimental studies demonstrate the remarkable efficiency of biological neural systems to encode, process, and learn from information. To better understand the experimentally observed phenomena, theoreticians are developing new mathematical approaches and tools to model biological neural networks. The chapter "Modeling Biological Neural Networks", by Joaquin J. Torres and Pablo Varona, reviews some of the most popular models of neurons and neural networks. These not only help to understand how living systems perform information processing, but may also lead to novel bioinspired paradigms of artificial intelligence and robotics.

The size and complexity of biological data, such as DNA/RNA sequences and protein sequences and structures, makes them suitable for advanced computational tools, such as neural networks. Computational analysis of such databases aims at exposing hidden information that provides insights that help in understanding the underlying biological principles. The chapter "Neural Networks in Bioinformatics", by Ke Chen and Lukasz A. Kurgan, focuses on proteins. In particular it discusses prediction of protein secondary structure, solvent accessibility, and binding residues.

Self-organizing maps is a prime example of an artificial neural network model that both relates to the actual (topological) organization within the mammalian brain and at the same time has many practical applications. Self-organizing maps go back to the seminal work of Teuvo Kohonen. The chapter "Self-organizing Maps", by Marc M. Van Hulle, describes the state of the art with a special emphasis on learning algorithms that aim to optimize a predefined criterion.

Evolutionary Computation

This area is covered by thirteen chapters.

The first chapter, "Generalized Evolutionary Algorithms", by Kenneth De Jong, describes the general concept of evolutionary algorithms. As a generic introduction to the field, this chapter facilitates an understanding of specific instances of evolutionary algorithms as instantiations of a generic evolutionary algorithm. For the instantiations, certain choices need to be made, such as representation, variation operators, and the selection operator, which then yield particular instances of evolutionary algorithms, such as genetic algorithms and evolution strategies, to name just a few.

The chapter "Genetic Algorithms — A Survey of Models and Methods", by Darrell Whitley and Andrew M. Sutton, introduces and discusses (including criticism) the standard genetic algorithm based on the classical binary representation of solution candidates and a theoretical interpretation based on the so-called schema theorem. Variations of genetic algorithms with respect to solution representations, mutation operators, recombination operators, and selection mechanisms are also explained and discussed, as well as theoretical models of genetic algorithms based on infinite and finite population size assumptions and Markov chain theory concepts. The authors also critically investigate genetic algorithms from the perspective of identifying their limitations and the differences between theory and practice when working with genetic algorithms. To illustrate this further, the authors also give a practical example of the application of genetic algorithms to resource scheduling problems.

The chapter "Evolutionary Strategies", by Günter Rudolph, describes a class of evolutionary algorithms which have often been associated with numerical function optimization and continuous variables, but can also be applied to binary and integer domains. Variations of evolutionary strategies, such as the $(\mu+\lambda)$-strategy and the (μ,λ)-strategy, are introduced and discussed within a common algorithmic framework. The fundamental idea of self-adaptation of strategy parameters (variances and covariances of the multivariate normal distribution used for mutation) is introduced and explained in detail, since this is a key differentiating property of evolutionary strategies.

The chapter "Evolutionary Programming", by Gary B. Fogel, discusses a historical branch of evolutionary computation. It gives a historical perspective on evolutionary programming by describing some of the original experiments using evolutionary programming to evolve finite state machines to serve as sequence predictors. Starting from this canonical evolutionary programming approach, the chapter also presents extensions of evolutionary programming into continuous domains, where an attempt towards self-adaptation of mutation step sizes has been introduced which is similar to the one considered in evolutionary strategies. Finally, an overview of some recent applications of evolutionary programming is given.

The chapter "Genetic Programming — Introduction, Applications, Theory and Open Issues", by Leonardo Vanneschi and Riccardo Poli, describes a branch of evolutionary

algorithms derived by extending genetic algorithms to allow exploration of the space of computer programs. To make evolutionary search in the domain of computer programs possible, genetic programming is based on LISP S-expression represented by syntax trees, so that genetic programming extends evolutionary algorithms to tree-based representations. The chapter gives an overview of the corresponding representation, search operators, and technical details of genetic programming, as well as existing applications to real-world problems. In addition, it discusses theoretical approaches toward analyzing genetic programming, some of the open issues, as well as research trends in the field.

The subsequent three chapters are related to the theoretical analysis of evolutionary algorithms, giving a broad overview of the state of the art in our theoretical understanding. These chapters demonstrate that there is a sound theoretical understanding of capabilities and limitations of evolutionary algorithms. The approaches can be roughly split into convergence velocity or progress analysis, computational complexity investigations, and global convergence results.

The convergence velocity viewpoint is represented in the chapter "The Dynamical Systems Approach — Progress Measures and Convergence Properties", by Silja Meyer-Nieberg and Hans-Georg Beyer. It demonstrates how the dynamical systems approach can be used to analyze the behavior of evolutionary algorithms quantitatively with respect to their progress rate. It also provides a complete overview of results in the continuous domain, i.e., for all types of evolution strategies on certain objective functions (such as sphere, ridge, etc.). The chapter presents results for undisturbed as well as for noisy variants of these objective functions, and extends the approach to dynamical objective functions where the goal turns into optimum tracking. All results are presented by means of comparative tables, so the reader gets a complete overview of the key findings at a glance.

The chapter "Computational Complexity of Evolutionary Algorithms", by Thomas Jansen, deals with the question of optimization time (i.e., the first point in time during the run of an evolutionary algorithm when the global optimum is sampled) and an investigation of upper bounds, lower bounds, and the average time needed to hit the optimum. This chapter presents specific results for certain classes of objective functions, most of them defined over binary search spaces, as well as fundamental limitations of evolutionary search and related results on the "no free lunch" theorem and black box complexity. The chapter also discusses the corresponding techniques for analyses, such as drift analysis and the expected multiplicative distance decrease.

Concluding the set of theoretical chapters, the chapter "Stochastic Convergence", by Günter Rudolph, addresses theoretical results about the properties of evolutionary algorithms concerned with finding a globally optimal solution in the asymptotic limit. Such results exist for certain variants of evolutionary algorithms and under certain assumptions, and this chapter summarizes the existing results and integrates them into a common framework. This type of analysis is essential in qualifying evolutionary algorithms as global search algorithms and for understanding the algorithmic conditions for global convergence.

The remaining chapters in the area of evolutionary computation report some of the major current trends.

To start with, the chapter "Evolutionary Multiobjective Optimization", by Eckart Zitzler, focuses on the application of evolutionary algorithms to tasks that are characterized by multiple, conflicting objective functions. In this case, decision-making becomes a task of identifying good compromises between the conflicting criteria. This chapter introduces the concept and a variety of state-of-the-art algorithmic concepts to use evolutionary algorithms

for approximating the so-called Pareto front of solutions which cannot be improved in one objective without compromising another. This contribution presents all of the required formal concepts, examples, and the algorithmic variations introduced into evolutionary computation to handle such types of problems and to generate good approximations of the Pareto front.

The term "memetic algorithms" is used to characterize hybridizations between evolutionary algorithms and more classical, local search methods (and agent-based systems). This is a general concept of broad scope, and in order to illustrate and characterize all possible instantiations, the chapter "Memetic Algorithms", by Natalio Krasnogor, presents an algorithmic engineering approach which allows one to describe these algorithms as instances of generic patterns. In addition to explaining some of the application areas, the chapter presents some theoretical remarks, various different ways to define memetic algorithms, and also an outlook into the future.

The chapter "Genetics-Based Machine Learning", by Tim Kovacs, extends the idea of evolutionary optimization to algorithmic concepts in machine learning and data mining, involving applications such as learning classifier systems, evolving neural networks, and genetic fuzzy systems, to mention just a few. Here, the application task is typically a data classification, data prediction, or nonlinear regression task — and the quality of solution candidates is evaluated by means of some model quality measure. The chapter covers a wide range of techniques for applying evolutionary computation to machine learning tasks, by interpreting them as optimization problems.

The chapter "Coevolutionary Principles", by Elena Popovici, Anthony Bucci, R. Paul Wiegand, and Edwin D. de Jong, deals with a concept modeled after biological evolution in which an explicit fitness function is not available, but solutions are evaluated by running them against each other. A solution is evaluated in the context of the other solutions, in the actual population or in another. Therefore, these algorithms develop their own dynamics, because the point of comparison is not stable, but coevolving with the actual population. The chapter provides a fundamental understanding of coevolutionary principles and highlights theoretical concepts, algorithms, and applications.

Finally, the chapter "Niching in Evolutionary Algorithms", by Ofer M. Shir, describes the biological principle of niching in nature as a concept for using a single population to find, occupy, and keep multiple local minima in a population. The motivation for this approach is to find alternative solutions within a single population and run of evolutionary algorithms, and this chapter discusses approaches for niching, and the application in the context of genetic algorithms as well as evolutionary strategies.

Molecular Computation

This area is covered by eight chapters.

The chapter "DNA Computing — Foundations and Implications", by Lila Kari, Shinnosuke Seki, and Petr Sosík, has a dual purpose. The first part outlines basic molecular biology notions necessary for understanding DNA computing, recounts the first experimental demonstration of DNA computing by Leonard Adleman in 1994, and recaps the 2001 milestone wet laboratory experiment that solved a 20-variable instance of 3-SAT and thus first demonstrated the potential of DNA computing to outperform the computational ability of an unaided human. The second part describes how the properties of DNA-based information, and in particular the Watson–Crick complementarity of DNA single strands, have influenced

areas of theoretical computer science such as formal language theory, coding theory, automata theory, and combinatorics on words. More precisely, it explores several notions and results in formal language theory and coding theory that arose from the problem of the design of optimal encodings for DNA computing experiments (hairpin-free languages, bond-free languages), and more generally from the way information is encoded on DNA strands (sticker systems, Watson–Crick automata). Lastly, it describes the influence that properties of DNA-based information have had on research in combinatorics on words, by presenting several natural generalizations of classical concepts (pseudopalindromes, pseudoperiodicity, Watson–Crick conjugate and commutative words, involutively bordered words, pseudoknot bordered words), and outlining natural extensions in this context of two of the most fundamental results in combinatorics of words, namely the Fine and Wilf theorem and the Lyndon–Schützenberger result.

The chapter "Molecular Computing Machineries — Computing Models and Wet Implementations", by Masami Hagiya, Satoshi Kobayashi, Ken Komiya, Fumiaki Tanaka, and Takashi Yokomori, explores novel computing devices inspired by the biochemical properties of biomolecules. The theoretical results section describes a variety of molecular computing models for finite automata, as well as molecular computing models for Turing machines based on formal grammars, equality sets, Post systems, and logical formulae. It then presents molecular computing models that use structured molecules such as hairpins and tree structures. The section on wet implementations of molecular computing models, related issues, and applications includes: an enzyme-based DNA automaton and its applications to drug delivery, logic gates and circuits using DNAzymes and DNA tiles, reaction graphs for representing various dynamics of DNA assembly pathways, DNA whiplash machines implementing finite automata, and a hairpin-based implementation of a SAT engine for solving the 3-SAT problem.

The chapter "DNA Computing by Splicing and by Insertion−Deletion", by Gheorghe Păun, is devoted to two of the most developed computing models inspired by DNA biochemistry: computing by splicing, and computing by insertion and deletion. DNA computing by splicing was defined by Tom Head already in 1987 and is based on the so-called splicing operation. The splicing operation models the recombination of DNA molecules that results from cutting them with restriction enzymes and then pasting DNA molecules with compatible ends by ligase enzymes. This chapter explores the computational power of the splicing operation showing that, for example, extended splicing systems starting from a finite language and using finitely many splicing rules can generate only the family of regular languages, while extended splicing systems starting from a finite language and using a regular set of rules can generate all recursively enumerable languages. Ways in which to avoid the impractical notion of a regular infinite set of rules, while maintaining the maximum computational power, are presented. They include using multisets and adding restrictions on the use of rules such as permitting contexts, forbidding contexts, programmed splicing systems, target languages, and double splicing. The second model presented, the insertion−deletion system, is based on a finite set of axioms and a finite set of contextual insertion rules and contextual deletion rules. Computational power results described here include the fact that insertion−deletion systems with context-free insertion rules of words of length at most one and context-free deletion rules of words of unbounded length can generate only regular languages. In contrast, for example, the family of insertion−deletion systems where the insertion contexts, deletion contexts, and the words to be inserted/deleted are all of length at most one, equals the family of recursively enumerable languages.

The chapter "Bacterial Computing and Molecular Communication", by Yasubumi Sakakibara and Satoshi Hiyama, investigates attempts to create autonomous cell-based Turing machines, as well as novel communication paradigms that use molecules as communication media. The first part reports experimental research on constructing *in vivo* logic circuits as well as efforts towards building *in vitro* and *in vivo* automata in the framework of DNA computing. Also, a novel framework is presented to develop a programmable and autonomous *in vivo* computer in a bacterium. The first experiment in this direction uses DNA circular strands (plasmids) together with the cell's protein-synthesis mechanism to execute a finite state automaton in *E. coli.* Molecular communication is a new communication paradigm that proposes the use of molecules as the information medium, instead of the traditional electromagnetic waves. Other distinctive features of molecular communication include its stochastic nature, its low energy consumption, the use of an aqueous transmission medium, and its high compatibility with biological systems. A molecular communication system starts with a sender (e.g., a genetically modified or an artificial cell) that generates molecules, encodes information onto the molecules (called information molecules), and emits the information molecules into a propagation environment (e.g., aqueous solution within and between cells). A molecular propagation system (e.g., lipid bilayer vesicles encapsulating the information molecules) actively transports the information molecules to an appropriate receiver. A receiver (e.g., a genetically modified or an artificial cell) selectively receives the transported information molecules, and biochemically reacts to the received information molecules, thus "decoding" the information. The chapter describes detailed examples of molecular communication system designs, experimental results, and research trends.

The chapter "Computational Nature of Gene Assembly in Ciliates", by Robert Brijder, Mark Daley, Tero Harju, Nataša Jonoska, Ion Petre, and Grzegorz Rozenberg, reviews several approaches and results in the computational study of gene assembly in ciliates. Ciliated protozoa contain two functionally different types of nuclei, the macronucleus and the micronucleus. The macronucleus contains the functional genes, while the genes of the micronucleus are not functional due to the presence of many interspersing noncoding DNA segments. In addition, in some ciliates, the coding segments of the genes are present in a permuted order compared to their order in the functional macronuclear genes. During the sexual process of conjugation, when two ciliates exchange genetic micronuclear information and form two new micronuclei, each of the ciliates has to "decrypt" the information contained in its new micronucleus to form its new functional macronucleus. This process is called gene assembly and involves deleting the noncoding DNA segments, as well as rearranging the coding segments in the correct order. The chapter describes two models of gene assembly, the intermolecular model based on the operations of circular insertion and deletion, and the intramolecular model based on the three operations of "loop, direct-repeat excision", "hairpin, inverted-repeat excision", and "double-loop, alternating repeat excision". A discussion follows of the mathematical properties of these models, such as the Turing machine computational power of contextual circular insertions and deletions, and properties of the gene assembly process called invariants, which hold independently of the molecular model and assembling strategy. Finally, the template-based recombination model is described, offering a plausible hypothesis (supported already by some experimental data) about the "bioware" that implements the gene assembly.

The chapter "DNA Memory", by Masanori Arita, Masami Hagiya, Masahiro Takinoue, and Fumiaki Tanaka, summarizes the efforts that have been made towards realizing Eric Baum's dream of building a DNA memory with a storage capacity vastly larger than

the brain. The chapter first describes the research into strategies for DNA sequence design, i.e., for finding DNA sequences that satisfy DNA computing constraints such as uniform melting temperature, avoidance of undesirable Watson–Crick bonding between sequences, preventing secondary structures, avoidance of base repeats, and absence of forbidden sequences. Various implementations of memory operations, such as access, read, and write, are described. For example, the "access" to a memory word in Baum's associative memory model, where a memory word consists of a single-stranded portion representing the address and a double-stranded portion representing the data, can be implemented by using the Watson–Crick complement of the address fixed to a solid support. In the Nested Primer Molecular Memory, where the double-stranded data is flanked on both sides by address sequences, the data can be retrieved by Polymerase Chain Reaction (PCR) using the addresses as primer pairs. In the multiple hairpins DNA memory, the address is a catenation of hairpins and the data can be accessed only if the hairpins are opened in the correct order by a process called DNA branch migration. After describing implementations of writable and erasable hairpin memories either in solution or immobilized on surfaces, the topic of *in vivo* DNA memory is explored. As an example, the chapter describes how representing the digit 0 by regular codons, and the digit 1 by wobbled codons, was used to encode a word into an essential gene of *Bacillus subtilis*.

The chapter "Engineering Natural Computation by Autonomous DNA-Based Biomolecular Devices", by John H. Reif and Thomas H. LaBean, overviews DNA-based biomolecular devices that are autonomous (execute steps with no external control after starting) and programmable (the tasks executed can be modified without entirely redesigning the DNA nanostructures). Special attention is given to DNA tiles, roughly square-shaped DNA nanostructures that have four "sticky-ends" (DNA single strands) that can specifically bind them to other tiles via Watson–Crick complementarity, and thus lead to the self-assembly of larger and more complex structures. Such tiles have been used to execute sequential Boolean computation via linear DNA self-assembly or to obtain patterned 2D DNA lattices and Sierpinski triangles. Issues such as error correction and self-repair of DNA tiling are also addressed. Other described methods include the implementation of a DNA-based finite automaton via disassembly of a double-stranded DNA nanostructure effected by an enzyme, and the technique of whiplash PCR. Whiplash PCR is a method that can achieve state transitions by encoding both transitions and the current state of the computation on the same DNA single strand: The free end of the strand (encoding the current state) sticks to the appropriate transition rule on the strand forming a hairpin, is then extended by PCR to a new state, and finally is detached from the strand, this time with the new state encoded at its end. The technique of DNA origami is also described, whereby a scaffold strand (a long single DNA strand, such as from the sequence of a virus) together with many specially designed staple strands (short single DNA strands) self-assemble by folding the scaffold strand — with the aid of the staples — in a raster pattern that can create given arbitrary planar DNA nanostructures. DNA-based molecular machines are then described such as autonomous DNA walkers and programmable DNA nanobots (programmable autonomous DNA walker devices). A restriction-enzyme-based DNA walker consists of a DNA helix with two sticky-ends ("feet") that moves stepwise along a "road" (a DNA nanostructure with protruding "steps", i.e., single DNA strands).

The chapter "Membrane Computing", by Gheorghe Păun, describes theoretical results and applications of membrane computing, a branch of natural computing inspired by the architecture and functioning of living cells, as well as from the organization of cells in tissues, organs, or other higher-order structures. The cell is a hierarchical structure of compartments, defined by membranes, that selectively communicate with each other. The computing model

that abstracts this structure is a membrane system (or P system, from the name of its inventor, Gheorghe Păun) whose main components are: the membrane structure, the multisets of objects placed in the compartments enveloped by the membranes, and the rules for processing the objects and the membranes. The rules are used to modify the objects in the compartments, to transport objects from one compartment to another, to dissolve membranes, and to create new membranes. The rules in each region of a P system are used in a maximally parallel manner, nondeterministically choosing the applicable rules and the objects to which they apply. A computation consists in repeatedly applying rules to an initial configuration of the P system, until no rule can be applied anymore, in which case the objects in a priori specified regions are considered the output of the computation. Several variants of P systems are described, including P systems with symport/antiport rules, P systems with active membranes, splicing P systems, P systems with objects on membranes, tissue-like P systems, and spiking neural P systems. Many classes of P systems are able to simulate Turing machines, hence they are computationally universal. For example, P systems with symport/antiport rules using only three objects and three membranes are computationally universal. In addition, several types of P systems have been used to solve NP-complete problems in polynomial time, by a space–time trade-off. Applications of P systems include, among others, modeling in biology, computer graphics, linguistics, economics, and cryptography.

Quantum Computation

This area is covered by six chapters.

The chapter "Mathematics for Quantum Information Processing", by Mika Hirvensalo, contains the standard Hilbert space formulation of finite-level quantum systems. This is the language and notational system allowing us to speak, describe, and make predictions about the objects of quantum physics. The chapter introduces the notion of quantum states as unit-trace, self-adjoint, positive mappings, and the vector state formalism is presented as a special case. The physical observables are introduced as complete collections of mutually orthogonal projections, and then it is discussed how this leads to the traditional representation of observables as self-adjoint mappings. The minimal interpretation, which is the postulate connecting the mathematical objects to the physical world, is presented. The treatment of compound quantum systems is based mostly on operative grounds. To provide enough tools for considering the dynamics needed in quantum computing, the formalism of treating state transformations as completely positive mappings is also presented. The chapter concludes by explaining how quantum versions of finite automata, Turing machines, and Boolean circuits fit into the Hilbert space formalism.

The chapter "Bell's Inequalities — Foundations and Quantum Communication", by Časlav Brukner and Marek Żukowski, is concerned with the nature of quantum mechanics. It presents the evidence that excludes two types of hypothetical deterministic theories: neither a nonlocal nor a noncontextual theory can explain quantum mechanics. This helps to build a true picture of quantum mechanics, and is therefore essential from the philosophical point of view. The Bell inequalities show that nonlocal deterministic theories cannot explain the quantum mechanism, and the Kochen–Specker theorem shows that noncontextual theories are not possible as underlying theories either. The traditional Bell theorem and its variants, GHZ and CHSH among them, are presented, and the Kochen–Specker theorem is discussed. In this chapter, the communication complexity is also treated by showing how the violations

of classical locality and noncontextuality can be used as a resource for communication protocols. Stronger-than quantum violations of the CHSH inequality are also discussed. They are interesting, since it has been shown that if the violation of CHSH inequality is strong enough, then the communication complexity collapses into one bit (hence the communication complexity of the true physical world seems to settle somewhere between classical and stronger-than quantum).

The chapter "Algorithms for Quantum Computers", by Jamie Smith and Michele Mosca, introduces the most remarkable known methods that utilize the special features of quantum physics in order to gain advantage over classical computing. The importance of these methods is that they form the core of designing discrete quantum algorithms. The methods presented and discussed here are the quantum Fourier transform, amplitude amplification, and quantum walks. Then, as specific examples, Shor's factoring algorithm (quantum Fourier transform), Grover search (amplitude amplification), and element distinctness algorithms (quantum random walks) are presented. The chapter not only involves traditional methods, but it also contains discussion of continuous-time quantum random walks and, more importantly, an extensive presentation of an important recent development in quantum algorithms, viz., tensor network evaluation algorithms. Then, as an example, the approximate evaluation of Tutte polynomials is presented.

The chapter "Physical Implementation of Large-Scale Quantum Computation", by Kalle-Antti Suominen, discusses the potential ways of physically implementing quantum computers. First, the DiVincenzo criteria (requirements for building a successful quantum computer) are presented, and then quantum error correction is discussed. The history, physical properties, potentials, and obstacles of various possible physical implementations of quantum computers are covered. They involve: cavity QED, trapped ions, neutral atoms and single electrons, liquid-form molecular spin, nuclear and electron spins in silicon, nitrogen vacancies in diamond, solid-state qubits with quantum dots, superconducting charge, flux and phase quantum bits, and optical quantum computing.

The chapter "Quantum Cryptography", by Takeshi Koshiba, is concerned with quantum cryptography, which will most likely play an important role in future when quantum computers make the current public-key cryptosystems unreliable. It gives an overview of classical cryptosystems, discusses classical cryptographic protocols, and then introduces the quantum key distribution protocols BB84, B92, and BBM92. Also protocol OTU00, not known to be vulnerable under Shor's algorithm, is presented. In future, when quantum computers are available, cryptography will most probably be based on quantum protocols. The chapter presents candidates for such quantum protocols: KKNY05 and GC01 (for digital signatures). It concludes with a discussion of quantum commitment, oblivious transfer, and quantum zero-knowledge proofs.

The complexity class BQP is the quantum counterpart of the classical class BPP. Intuitively, BQP can be described as the class of problems solvable in "reasonable" time, and, hence, from the application-oriented point of view, it will likely become the most important complexity class in future, when quantum computers are available. The chapter "BQP-Complete Problems", by Shengyu Zhang, introduces the computational problems that capture the full hardness of BQP. In the very fundamental sense, no BQP-complete problems are known, but the promise problems (the probability distribution of outputs is restricted by promise) bring us as close as possible to the "hardest" problems in BQP, known as BQP-complete promise problems. The chapter discusses known BQP-complete promise problems. In particular, it is shown how to establish the BQP-completeness of the Local Hamiltonian Eigenvalue

Sampling problem and the Local Unitary Phase Sampling problem. The chapter concludes with an extensive study showing that the Jones Polynomial Approximation problem is a BQP-complete promise problem.

Broader Perspective

This area consists of two sections, "Nature-Inspired Algorithms" and "Alternative Models of Computation".

Nature-Inspired Algorithms

This section is covered by six chapters.

The chapter "An Introduction to Artificial Immune Systems", by Mark Read, Paul S. Andrews, and Jon Timmis, provides a general introduction to the field. It discusses the major research issues relating to the field of Artificial Immune Systems (AIS), exploring the underlying immunology that has led to the development of immune-inspired algorithms, and focuses on the four main algorithms that have been developed in recent years: clonal selection, immune network, negative selection, and dendritic cell algorithms; their use in terms of applications is highlighted. The chapter also covers evaluation of current AIS technology, and details some new frameworks and methodologies that are being developed towards more principled AIS research. As a counterpoint to the focus on applications, the chapter also gives a brief outline of how AIS research is being employed to help further the understanding of immunology.

The chapter on "Swarm Intelligence", by David W. Corne, Alan Reynolds, and Eric Bonabeau, attempts to demystify the term Swarm Intelligence (SI), outlining the particular collections of natural phenomena that SI most often refers to and the specific classes of computational algorithms that come under its definition. The early parts of the chapter focus on the natural inspiration side, with discussion of social insects and stigmergy, foraging behavior, and natural flocking behavior. Then the chapter moves on to outline the most successful of the computational algorithms that have emerged from these natural inspirations, namely ant colony optimization methods and particle swarm optimization, with also some discussion of different and emerging such algorithms. The chapter concludes with a brief account of current research trends in the field.

The chapter "Simulated Annealing", by Kathryn A. Dowsland and Jonathan M. Thompson, provides an overview of Simulated Annealing (SA), emphasizing its practical use. The chapter explains its inspiration from the field of statistical thermodynamics, and then overviews the theory, with an emphasis again on those aspects that are important for practical applications. The chapter then covers some of the main ways in which the basic SA algorithm has been modified by various researchers, leading to improved performance for a variety of problems. The chapter briefly surveys application areas, and ends with several useful pointers to associated resources, including freely available code.

The chapter "Evolvable Hardware", by Lukáš Sekanina, surveys this growing field. Starting with a brief overview of the reconfigurable devices used in this field, the elementary principles and open problems are introduced, and then the chapter considers, in turn, three main areas: extrinsic evolution (evolving hardware using simulators), intrinsic evolution (where the evolution is conducted within FPGAs, FPTAs, and so forth), and adaptive hardware

(in which real-world adaptive hardware systems are presented). The chapter finishes with an overview of major achievements in the field.

The first of two application-centered chapters, "Natural Computing in Finance — A Review", by Anthony Brabazon, Jing Dang, Ian Dempsey, Michael O'Neill, and David Edelman, provides a rather comprehensive account of natural computing applications in what is, at the time of writing (and undoubtedly beyond), one of the hottest topics of the day. This chapter introduces us to the wide range of different financial problems to which natural computing methods have been applied, including forecasting, trading, arbitrage, portfolio management, asset allocation, credit risk assessment, and more. The natural computing areas that feature in this chapter are largely evolutionary computing, neural computing, and also agent-based modeling, swarm intelligence, and immune-inspired methods. The chapter ends with a discussion of promising future directions.

Finally, the chapter "Selected Aspects of Natural Computing", by David W. Corne, Kalyanmoy Deb, Joshua Knowles, and Xin Yao, provides detailed accounts of a collection of example natural computing applications, each of which is remarkable or particularly interesting in some way. The thrust of this chapter is to provide, via such examples, an idea of both the significant impact that natural computing has already had, as well as its continuing significant promise for future applications in all areas of science and industry. While presenting this eclectic collection of marvels, the chapter also aims at clarity and demystification, providing much detail that helps see how the natural computing methods in question were applied to achieve the stated results. Applications covered include Blondie24 (the evolutionary neural network application that achieves master-level skill at the game of checkers), the design of novel antennas using evolutionary computation in conjunction with developmental computing, and the classic application of learning classifier systems that led to novel fighter-plane maneuvers for the USAF.

Alternative Models of Computation

This section is covered by seven chapters.

The chapter "Artificial Life", by Wolfgang Banzhaf and Barry McMullin, traces the roots, raises key questions, discusses the major methodological tools, and reviews the main applications of this exciting and maturing area of computing. The chapter starts with a historical overview, and presents the fundamental questions and issues that Artificial Life is concerned with. Thus the chapter surveys discussions and viewpoints about the very nature of the differences between living and nonliving systems, and goes on to consider issues such as hierarchical design, self-construction, and self-maintenance, and the emergence of complexity. This part of the chapter ends with a discussion of "Coreworld" experiments, in which a number of systems have been studied that allow spontaneous evolution of computer programs. The chapter moves on to survey the main theory and formalisms used in Artificial Life, including cellular automata and rewriting systems. The chapter concludes with a review and restatement of the main objectives of Artificial Life research, categorizing them respectively into questions about the origin and nature of life, the potential and limitations of living systems, and the relationships between life and intelligence, culture, and other human constructs.

The chapter "Algorithmic Systems Biology — Computer Science Propels Systems Biology", by Corrado Priami, takes the standpoint of computing as providing a philosophical foundation for systems biology, with at least the same importance as mathematics, chemistry,

or physics. The chapter highlights the value of algorithmic approaches in modeling, simulation, and analysis of biological systems. It starts with a high-level view of how models and experiments can be tightly integrated within an algorithmic systems biology vision, and then deals in turn with modeling languages, simulations of models, and finally the postprocessing of results from biological models and how these lead to new hypotheses that can then re-enter the modeling/simulation cycle.

The chapter "Process Calculi, Systems Biology and Artificial Chemistry", by Pierpaolo Degano and Andrea Bracciali, concentrates on the use of process calculi and related techniques for systems-level modeling of biological phenomena. This chapter echoes the broad viewpoint of the previous chapter, but its focus takes us towards a much deeper understanding of the potential mappings between formal systems in computer science and systems interpretation of biological processes. It starts by surveying the basics of process calculi, setting out their obvious credentials for modeling concurrent, distributed systems of interacting parts, and mapping these onto a "cells as computers" view. After a process calculi treatment of systems biology, the chapter goes on to examine process calculi as a route towards artificial chemistry. After considering the formal properties of the models discussed, the chapter ends with notes on some case studies showing the value of process calculi in modeling biological phenomena; these include investigating the concept of a "minimal gene set" prokaryote, modeling the nitric oxide-cGMP pathway (central to many signal transduction mechanisms), and modeling the calyx of Held (a large synapse structure in the mammalian auditory central nervous system).

The chapter on "Reaction–Diffusion Computing", by Andrew Adamatzky and Benjamin De Lacy Costello, introduces the reader to the concept of a reaction–diffusion computer. This is a spatially extended chemical system, which processes information via transforming an input profile of ingredients (in terms of different concentrations of constituent ingredients) into an output profile of ingredients. The chapter takes us through the elements of this field via case studies, and it shows how selected tasks in computational geometry, robotics, and logic can be addressed by chemical implementations of reaction–diffusion computers. After introducing the field and providing a treatment of its origins and main achievements, a classical view of reaction–diffusion computers is then described. The chapter moves on to discuss varieties of reaction–diffusion processors and their chemical constituents, covering applications to the aforementioned tasks. The chapter ends with the authors' thoughts on future developments in this field.

The chapter "Rough–Fuzzy Computing", by Andrzej Skowron, shifts our context towards addressing a persistent area of immense difficulty for classical computing, which is the fact that real-world reasoning is usually done in the face of inaccurate, incomplete, and often inconsistent evidence. In essence, concepts in the real world are vague, and computation needs ways to address this. We are hence treated, in this chapter, to an overarching view of rough set theory, fuzzy set theory, their hybridization, and applications. Rough and fuzzy computing are broadly complementary approaches to handling vagueness, focusing respectively on capturing the level of distinction between separate objects and the level of membership of an object in a set. After presenting the basic concepts of rough computing and fuzzy computing in turn, in each case going into some detail on the main theoretical results and practical considerations, the chapter goes on to discuss how they can be, and have been, fruitfully combined. The chapter ends with an overview of the emerging field of "Wisdom Technology" (Wistech) as a paradigm for developing modern intelligent systems.

The chapter "Collision-Based Computing", by Andrew Adamatzky and Jérôme Durand-Lose, presents and discusses the computations performed as a result of spatial localizations in

systems that exhibit dynamic spatial patterns over time. For example, a collision may be between two gliders in a cellular automaton, or two separate wave fragments within an excitable chemical system. This chapter introduces us to the basics of collision-based computing and overviews collision-based computing schemes in 1D and 2D cellular automata as well as continuous excitable media. Then, after some theoretical foundations relating to 1D cellular automata, the chapter presents a collision-based implementation for a 1D Turing machine and for cyclic tag systems. The chapter ends with discussion and presentation of "Abstract Geometrical Computation", which can be seen as collision-based computation in a medium that is the continuous counterpart of cellular automata.

The chapter "Nonclassical Computation — A Dynamical Systems Perspective", by Susan Stepney, takes a uniform view of computation, in which inspiration from a dynamical systems perspective provides a convenient way to consider, in one framework, both classical discrete systems and systems performing nonclassical computation. In particular, this viewpoint presents a way towards computational interpretation of physical embodied systems that exploit their natural dynamics. The chapter starts by discussing "closed" dynamical systems, those whose dynamics involve no inputs from an external environment, examining their computational abilities from a dynamical systems perspective. Then it discusses continuous dynamical systems and shows how these too can be interpreted computationally, indicating how material embodiment can give computation "for free", without the need to explicitly implement the dynamics. The outlook then broadens to consider open systems, where the dynamics are affected by external inputs. The chapter ends by looking at constructive, or developmental, dynamical systems, whose state spaces change during computation. These latter discussions approach the arena of biological and other natural systems, casting them as computational, open, developmental, dynamical systems.

Acknowledgements

This handbook resulted from a highly collaborative effort. The handbook and area editors are grateful to the chapter writers for their efforts in writing chapters and delivering them on time, and for their participation in the refereeing process.

We are indebted to the members of the Advisory Board for their valuable advice and fruitful interactions. Additionally, we want to acknowledge David Fogel, Pekka Lahti, Robert LaRue, Jason Lohn, Michael Main, David Prescott, Arto Salomaa, Kai Salomaa, Shinnosuke Seki, and Rob Smith, for their help and advice in various stages of production of this handbook. Last, but not least, we are thankful to Springer, especially to Ronan Nugent, for intense and constructive cooperation in bringing this project from its inception to its successful conclusion.

Leiden; Edinburgh; Nijmegen;
Turku; London, Ontario
October 2010

Grzegorz Rozenberg (Main Handbook Editor)
Thomas Bäck (Handbook Editor and Area Editor)
Joost N. Kok (Handbook Editor and Area Editor)
David W. Corne (Area Editor)
Tom Heskes (Area Editor)
Mika Hirvensalo (Area Editor)
Jarkko J. Kari (Area Editor)
Lila Kari (Area Editor)

http://www.springer.com/978-3-540-92909-3

Handbook of Natural Computing
Rozenberg, G.; Baeck, Th.; Kok, J.N. (Eds.)
2012, LII, 2052 p. 332 illus., 60 illus. in color. In 4 volumes, not available separately., Hardcover
ISBN: 978-3-540-92909-3