Preface

Software Science is a discipline that studies the theoretical framework of software as instructive and behavioral information, which can be embodied and executed by generic computers in order to create expected system behaviors and machine intelligence. *Intelligence science* is a discipline that studies the mechanisms and theories of abstract intelligence and its paradigms such as natural, artificial, machinable, and computational intelligence. The convergence of software and intelligent sciences forms the transdisciplinary field of *computational intelligence*, which provides a coherent set of fundamental theories, contemporary denotational mathematics, and engineering applications.

This book entitled *Advances in Abstract Intelligence and Soft Computing* is the third volume in the IGI Series of Advances in Software Science and Computational Intelligence. The book encompasses 25 chapters of expert contributions selected from the International Journal of Software Science and Computational Intelligence during 2011. The book is organized in four sections on:

1. Computational intelligence;
2. Cognitive computing;
3. Software science;
4. Applications in computational intelligence and cognitive computing.

Section 1. Computational Intelligence

Intelligence science studies theories and models of the brain, and the relationship between the concrete physiological brain and the abstract soft mind. Intelligence science is a new frontier with the fertilization of brain science, biology, psychology, neuroscience, cognitive science, cognitive informatics, philosophy, information science, computer science, anthropology, and linguistics. A fundamental view developed in software and intelligence sciences is known as abstract intelligence, which provides a unified foundation for the studies of all forms and paradigms of intelligence such as natural, artificial, machinable, and computational intelligence. *Abstract intelligence* (αI) is an enquiry of both natural and artificial intelligence at the neural, cognitive, functional, and logical levels from the bottom up. In the narrow sense, αI is a human or a system ability that transforms information into behaviors. However, in the broad sense, αI is any human or system ability that autonomously transfers the forms of abstract information between *data, information, knowledge,* and *behaviors* in the brain or intelligent systems.

Computational intelligence (CoI) is an embodying form of abstract intelligence (αI) that implements intelligent mechanisms and behaviors by computational methodologies and software systems, such as expert systems, fuzzy systems, cognitive computers, cognitive robots, software agent systems, genetic/

evolutionary systems, and autonomous learning systems. The theoretical foundations of computational intelligence root in cognitive informatics, software science, and denotational mathematics.

This section on computational intelligence encompasses the following seven chapters.

Chapter 1, *A Formal Knowledge Representation System (FKRS) for the Intelligent Knowledge Base of a Cognitive Learning Engine*, by Yousheng Tian, Yingxu Wang, Marina Gavrilova, and Guenther Ruhe, recognizes that the generic form of machine learning is a knowledge acquisition and manipulation process mimicking the brain. Therefore, knowledge representation as a dynamic concept network is centric in the design and implementation of the intelligent knowledge base of a Cognitive Learning Engine (CLE). This chapter presents a Formal Knowledge Representation System (FKRS) for autonomous concept formation and manipulation based on concept algebra. The Object-Attribute-Relation (OAR) model for knowledge representation is adopted in the design of FKRS. The conceptual model, architectural model, and behavioral models of the FKRS system is formally designed and specified in Real-Time Process Algebra (RTPA). The FKRS system is implemented in Java as a core component towards the development of the CLE and other knowledge-based systems in cognitive computing and computational intelligence.

Chapter 2, *Sparse Based Image Classification with Bag-of-Visual-Words Representations*, by Yuanyuan Zuo and Bo Zhang, presents a sparse representation based classification algorithm for human face recognition. The image database is restricted to human frontal faces with only slight illumination and expression changes. In this chapter, we apply the sparse representation based algorithm to the problem of generic image classification, with a certain degree of intra-class variations and background clutter. Experiments have been done with the sparse representation based algorithm and Support Vector Machine (SVM) classifiers on 25 object categories selected from the Caltech101 dataset. Experimental results show that without the time-consuming parameter optimization, the sparse representation based algorithm achieves comparable performance with SVM. The experiments also demonstrate that the algorithm is robust to a certain degree of background clutter and intra-class variations with the bag-of-visual-words representations. We argue that the sparse representation based algorithm can also be applied to generic image classification task when appropriate image feature is used.

Chapter 3, *Quotient Space-Based Boundary Condition for a Particle Swarm Optimization Algorithm*, by Yuhong Chi, Fuchun Sun, Langfan Jiang, Chunyang Yu, and Chunli Chen, proposes a novel quotient space-based boundary condition named QsaBC by using the properties of quotient space and homeomorphism for controling particles to fly inside the limited search space and dealing with the problems of slow search speed and premature convergence of particle swarm optimization algorithm. In QsaBC, Search space-zoomed factor and Attractor are introduced according to analyzing the dynamic behavior and stability of particles, which not only reduce the subjective interference and enforce the capability of global search, but also enhance the power of local search and escaping from an inferior local optimum. Four CEC'2008 benchmark functions were selected to evaluate the performance of QsaBC. Comparative experiments show that QsaBC can get the satisfactory optimization solution with fast convergence speed. Furthermore, QsaBC is more effective to do with errant particles, easier calculation, and better robustness than other experienced methods.

Chapter 4, *Medical Image Classification Using an Optimal Feature Extraction Algorithm and a Supervised Classifier Technique*, by Ahmed Kharrat, Karim Gasmi, Mohamed B. Messaoud, Nacéra Benamrane, and Mohamed Abid, presents a new approach for automated diagnosis and classification of Magnetic Resonance (MR) human brain images. The proposed method uses Wavelets Transform (WT) as input module to Genetic Algorithm (GA) and Support Vector Machine (SVM). It segregates MR brain images into normal and abnormal. Our contribution employs genetic algorithm for feature selection which requires much lighter computational burden in comparison with Sequential Floating Backward Selection (SFBS) and Sequential Floating Forward Selection (SFFS) methods. A percentage reduction rate of 88.63% is achieved. An excellent classification rate of 100% could be achieved using the support vector machine. The authors observe their results are significantly better than the results reported in a previous research work employing Wavelet Transform and Support Vector Machine.

Chapter 5, *EEG Feature Extraction and Pattern Classification based on Motor Imagery in Brain-Computer Interface*, by Ling Zou, Xinguan Wang, Guodong Shi, and Zhenghua Ma, identifies that accurate classification of EEG in left and right hand motor imagery is an important issue in brain-computer interface. Firstly, discrete wavelet transform method was used to decompose the average power of C3 electrode and C4 electrode in left-right hands imagery movement during some periods of time. The reconstructed signal of approximation coefficient A6 on the sixth level was selected to build up a feature signal. Secondly, the performances by Fisher Linear Discriminant Analysis with two different threshold calculation ways and Support Vector Machine methods were compared. The final classification results showed that false classification rate by Support Vector Machine was lower and gained an ideal classification results.

Chapter 6, *Inconsistency-Induced Learning for Perpetual Learners*, by Du Zhang and Meiliu Lu, identifies that one of the long-term research goals in machine learning is how to build never-ending learners. The state-of-the-practice in the field of machine learning thus far is still dominated by the one-time learner paradigm: some learning algorithm is utilized on data sets to produce certain model or target function, and then the learner is put away and the model or function is put to work. Such a learn-once-apply-next (or LOAN) approach may not be adequate in dealing with many real world problems and is in sharp contrast with the human's lifelong learning process. On the other hand, learning can often be brought on through overcoming some inconsistent circumstances. In this chapter, the authors propose a framework for perpetual learning agents that are capable of continuously refining or augmenting their knowledge through overcoming inconsistencies encountered during their problem-solving episodes. The never-ending nature of a perpetual learning agent is embodied in the framework as the agent's continuous inconsistency-induced belief revision process. The framework hinges on the agents recognizing inconsistency in data, information, knowledge, or meta-knowledge, identifying the cause of inconsistency, revising or augmenting beliefs to explain, resolve, or accommodate inconsistency. The authors believe that inconsistency can serve as one of the important learning stimuli toward building perpetual learning agents that incrementally improve their performance over time.

Chapter 7, *Toward Automatic Answers in an Interactive Question-Answer System*, by Tianyong Hao, Feifei Xu, Jingsheng Lei, and Wenyin Liu, proposes a strategy of automatic answer retrieval for repeated or similar questions in user-interactive systems by employing semantic question patterns. The used semantic question pattern is a generalized representation of a group of questions with both similar structure and relevant semantics. Specifically, it consists of semantic annotations (or constraints) for the variable components in the pattern and hence enhances the semantic representation and greatly reduces the ambiguity of a question instance when asked by a user using such pattern. The proposed method consists of four major steps: structure processing, similar pattern matching and filtering, automatic pattern generation, question similarity evaluation and answer retrieval. Preliminary experiments in a real question answering system show a precision of more than 90% of our method.

Section 2. Cognitive Computing

Computing systems and technologies can be classified into the categories of *imperative*, *autonomic,* and *cognitive* computing from the bottom up. The imperative computers are a passive system based on stored-program controlled mechanisms for data processing. The autonomic computers are goal-driven and self-decision-driven machines that do not rely on instructive and procedural information. Cognitive computers are more intelligent computers beyond the imperative and autonomic computers, which embody major natural intelligence behaviors of the brain such as thinking, inference, and learning. The increasing demand for non von Neumann computers for knowledge and intelligence processing in the high-tech industry and everyday lives require novel cognitive computers for providing autonomous computing power for various cognitive systems mimicking the natural intelligence of the brain.

Cognitive Computing (CC) is a novel paradigm of intelligent computing methodologies and systems based on *Cognitive Informatics* (CI), which implements computational intelligence by autonomous inferences and perceptions mimicking the mechanisms of the brain. CC is emerged and developed based on the transdisciplinary research in cognitive informatics, abstract intelligence, and *Denotational Mathematics* (DM). The latest advances in CI, CC, and DM enable a systematic solution for the future generation of intelligent computers known as *Cognitive Computers* (CogCs) that think, perceive, learn, and reason. A CogC is an intelligent computer for knowledge processing as that of a conventional von Neumann computer for data processing. CogCs are designed to embody *machinable intelligence* such as computational inferences, causal analyses, knowledge manipulations, machine learning, and autonomous problem solving.

This section on cognitive computing encompasses the following five chapters.

Chapter 8, *On Cognitive Models of Causal Inferences and Causation Networks*, by Yingxu Wang, reveals that human thought, perception, reasoning, and problem solving are highly dependent on causal inferences. The chapter presents a set of cognitive models for causation analyses and causal inferences. The taxonomy and mathematical models of causations are created. The framework and properties of causal inferences are elaborated. Methodologies for uncertain causal inferences are discussed. The theoretical foundation of humor and jokes as false causality is revealed. The formalization of causal inference methodologies enables machines to mimic complex human reasoning mechanisms in cognitive informatics, cognitive computing, and computational intelligence.

Chapter 9, *On Localities of Knowledge Inconsistency*, by Du Zhang, identifies that inconsistency is commonplace in the real world, in our long-term memory, and in knowledge based systems. Managing inconsistency is considered a hallmark of the plasticity of human intelligence. An important mental process that underpins human intelligence is belief revision. To facilitate belief revision, a necessary condition is that we need to know the localities and contexts of inconsistency, and how different types of inconsistency are clustered. In this chapter, we provide a formal definition of locality of inconsistency and describe how to use it to identify clusters of inconsistent circumstances in a knowledge base. The results in the chapter will help pave the way for a disciplined approach to managing knowledge inconsistency.

Chapter 10, *Adaptive Study Design through Semantic Association Rule Analysis*, by Ping Chen, Wei Ding, and Walter Garcia, aims to find valid correlations among data attributes which has been widely applied to many areas of data analysis. In this chapter we present a semantic network based association analysis model including three spreading activation methods, and apply this model to assess the quality of a dataset, and generate semantically valid new hypotheses for adaptive study design that is especially useful in medical studies. The authors evaluate their approach on a real public health dataset, the Heartfelt study, and the experiment shows promising results.

Chapter 11, *Qualitative Reasoning Approach to a Driver's Cognitive Mental Load*, by Shinichiro Sega, Hirotoshi Iwasaki, Hironori Hiraishi, and Fumio Mizoguchi, applies qualitative reasoning to a driver's mental state in real driving situations so as to develop a working load for intelligent transportation systems. The authors identify the cognitive state that determines whether a driver will be ready to operate a device in car navigation. In order to identify the driver's cognitive state, the authors measure eye movements during car-driving situations. The authors can acquire data for the various actions of a car driver, in particular braking, acceleration, and steering angles from our experiment car. They constructed a driver cognitive mental load using the framework of qualitative reasoning. They checked the response of our model by qualitative simulation. The authors also verified the model using real data collected by driving an actual car. The results indicated that our model could represent the change in the cognitive mental load based on measurable data. This means that the framework of this chapter will be useful for designing user interfaces for next-generation systems that actively employ user situations.

Chapter 12, *Intelligent Fault Recognition and Diagnosis for Rotating Machines using Neural Networks*, by Cyprian F. Ngolah, Ed Morden, and Yingxu Wang, examine the development of an intelligent fault recognition and monitoring system (Melvin I), which detects and diagnoses rotating machine conditions according to changes in fault frequency indicators. The signals and data are remotely collected from designated sections of machines via data acquisition cards. They are processed by a signal processor to extract characteristic vibration signals of ten key performance indicators (KPIs). A 3-layer neural network is designed to recognize and classify faults based on a pre-determined set of KPIs. The system implemented in the laboratory and applied in the field can also incorporate new experiences into the knowledge base without overwriting previous training. Results show that Melvin I is a smart tool for both system vibration analysts and industrial machine operators.

Section 3. Software Science

Software as instructive behavioral information has been recognized as an entire range of widely and frequently used objects and phenomena in human knowledge. Software science is a theoretical inquiry of software and its constraints on the basis of empirical studies on engineering methodologies and techniques for software development and software engineering organization. In the history of science and engineering, a matured discipline always gave birth to new disciplines. For instance, theoretical physics was emerged from general and applied physics, and theoretical computing was emerged from computer engineering. So does software science that emerges from and grows in the fields of software, computer, information, knowledge, and system engineering.

Software Science (SS) is a discipline of enquiries that studies the theoretical framework of software as instructive and behavioral information, which can be embodied and executed by generic computers in order to create expected system behaviors and machine intelligence. The discipline of software science studies the common objects in the abstract world such as software, information, data, concepts, knowledge, instructions, executable behaviors, and their processing by natural and artificial intelligence. From this view, software science is theoretical software engineering; while software engineering is applied software science in order to efficiently, economically, and reliably develop large-scale software systems. The phenomena that almost all the fundamental problems, which could not be solved in the last four decades in software engineering, simply stemmed from the lack of coherent theories in the form of software science. The vast cumulated empirical knowledge and industrial practice in software engineering have made this possible to enable the emergence of software science.

This section on software science encompasses the following six chapters.

Chapter 13, *Empirical Studies on the Functional Complexity of Software in Large-Scale Software Systems*, by Yingxu Wang and Vincent Chiew, recognizes that functional complexity is one of the most fundamental properties of software because almost all other software attributes and properties such as functional size, development effort, costs, quality, and project duration are highly dependent on it. The functional complexity of software is a macro-scope problem concerning the semantic properties of software and human cognitive complexity towards a given software system; while the computational complexity is a micro-scope problem concerning algorithmic analyses towards machine throughput and time/space efficiency. This chapter presents an empirical study on the functional complexity of software known as cognitive complexity based on large-scale samples using a Software Cognitive Complexity Analysis Tool (SCCAT). Empirical data are obtained with SCCAT on 7,531 programs and five formally specified software systems. The theoretical foundation of software functional complexity is introduced and the metric of software cognitive complexity is formally modeled. On the basis of cognitive complexity, the functional complexities of a large-scale software system, the air traffic control systems (ATCS), are rigorously analyzed. A novel approach to represent software functional complexities and their distributions in software systems is developed. The nature of functional complexity of software in software engineering is rigorously explained. The relationship between the symbolic and functional complexities of software is quantitatively analyzed.

Chapter 14, *The Formal Design Models of a File Management System (FMS)*, by Yingxu Wang, Cyprian F. Ngolah, Xinming Tan, Yousheng Tian, and Phillip C.-Y. Sheu, presents abstract files as a typical abstract data type for data objects and software modeling, which provides a standard encapsulation and access interface for manipulating large-volume information and persistent data. File management systems are an indispensable component of operating systems and real-time systems for file manipulations. This chapter develops a comprehensive design pattern of files and a File Management System (FMS). A rigorous denotational mathematics, Real-Time Process Algebra (RTPA), is adopted, which allows both architectural and behavioral models of files and FMS to be rigorously designed and implemented in a top-down approach. The conceptual model, architectural model, and the static/dynamic behavioral models of files and FMS are systematically presented. This work has been applied in the design and modeling of a real-time operating system (RTOS+).

Chapter 15, *The Formal Design Model of Doubly-Linked-Circular Lists (DLC-Lists)*, by Yingxu Wang, Cyprian F. Ngolah, Xinming Tan, and Phillip C.-Y. Sheu, formally models the Abstract Data Types (ADTs) as a set of highly generic and rigorously modeled data structures in type theory. Lists as a finite sequence of elements are one of the most fundamental and widely used ADTs in system modeling, which provide a standard encapsulation and access interface for manipulating large-volume information and persistent data. This chapter develops a comprehensive design pattern of formal lists using a doubly-linked-circular (DLC) list architecture. A rigorous denotational mathematics, Real-Time Process Algebra (RTPA), is adopted, which allows both architectural and behavioral models of lists to be rigorously designed and implemented in a top-down approach. The architectural models of DLC-Lists are created using RTPA architectural modeling methodologies known as the Unified Data Models (UDMs). The physical model of DLC-Lists is implemented using doubly linked nodes dynamically created in the memory. The behavioral models of DLC-Lists are specified and refined by a set of Unified Process Models (UPMs) in three categories namely the management operations, traversal operations, and node

I/O operations. This work has been applied in a number of real-time and nonreal-time system designs such as a real-time operating system (RTOS+), a file management system (FMS), and the ADT library for an RTPA-based automatic code generation tool.

Chapter 16, *Petri Nets and Discrete Event Systems*, by Juan L.G. Guirao and Fernando L. Pelayo, presents an overview over the relationship between Petri Nets and Discrete Event Systems as they have been proved as key factors in the cognitive processes of perception and memorization. In this sense, different aspects of encoding Petri Nets as Discrete Dynamical Systems that try to advance not only in the problem of reachability but also in the one of describing the periodicity of markings and their similarity, are revised. It is also provided a metric for the case of Non-bounded Petri Nets.

Chapter 17, *The Formal Design Models of the Universal Array (UA) and Its Implementation*, by Yingxu Wang, Jason Huang, and Jingsheng Lei, identifies that arrays are one of the most fundamental and widely applied data structures, which are useful for modeling both logical designs and physical implementations of multi-dimensional data objects sharing the same type of homogeneous elements. However, there is a lack of a formal model of the universal array based on it any array instance can be derived. This chapter studies the fundamental properties of Universal Array (UA) and presents a comprehensive design pattern of UA. A denotational mathematics, Real-Time Process Algebra (RTPA), is adopted, which allows both architectural and behavioral models of UA to be rigorously designed and refined in a top-down approach. The conceptual model of UA is rigorously described by tuple- and matrix-based mathematical models. The architectural models of UA are created using RTPA architectural modeling methodologies known as the Unified Data Models (UDMs). The physical model of UA is implemented using linear list that is indexed by an offset pointer of elements. The behavioral models of UA are specified and refined by a set of Unified Process Models (UPMs) for the creation, initialization, update, retrieve, and release operations of UA. As a case study, the formal UA models are implemented in Java, which demonstrate the seamless transformability from the formal specifications to code. This work has been applied in a number of real-time and nonreal-time systems such as compilers, a file management system, the real-time operating system (RTOS+), and the ADT library for an RTPA-based automatic code generation tool.

Chapter 18, *The Formal Design Models of Tree Architectures and Behaviors*, by Yingxu Wang and Xinming Tan, presents abstract trees as one of the most fundamental and widely used non-linear hierarchical structures of linked nodes. A binary tree (B-Tree) is a typical balanced tree where the fan-out of each node is at most two known as the left and right children. This chapter develops a comprehensive design pattern of formal trees using the B-Tree architecture. A rigorous denotational mathematics, Real-Time Process Algebra (RTPA), is adopted, which allows both architectural and behavioral models of B-Trees to be rigorously designed and implemented in a top-down approach. The architectural models of B-Trees are created using RTPA architectural modeling methodologies known as the Unified Data Models (UDMs). The physical model of B-Trees is implemented using the left and right child nodes dynamically created in memory. The behavioral models of B-Trees are specified and refined by a set of Unified Process Models (UPMs) in three categories namely the management operations, traversal operations, and node I/O operations. This work has been applied in a number of real-time and nonreal-time system designs such as a real-time operating system (RTOS+), a general system organization model, and the ADT library for an RTPA-based automatic code generator.

Section 4. Applications of Computational Intelligence and Cognitive Computing

A series of fundamental breakthroughs have been recognized and a wide range of applications has been developed in software science, abstract intelligence, cognitive computing, and computational intelligence in the last decade. Because software science and computational intelligence provide a common and general platform for the next generation of cognitive computing, some expected innovations in these fields will emerge such as cognitive computers, cognitive knowledge representation technologies, semantic searching engines, cognitive learning engines, cognitive Internet, cognitive robots, and autonomous inference machines for complex and long-series of inferences, problem solving, and decision making beyond traditional logic- and rule-based technologies.

This section on applications of computational intelligence and cognitive computing encompasses the following seven chapters.

Chapter 19, *Four-Channel Control Architectures for Bilateral and Multilateral Teleoperation*, by Yuji Wang, Fuchun Sun, and Huaping Liu, presets a four-channel architecture in teleoperation with force feedback in various existing literature. However, most of them focused on Lawrence architecture and did not research other cases. In this chapter we propose two other four-channel architectures: passive four-channel architecture and passive four-channel architecture with operator force. Furthermore, two types of multilateral shared control architecture based on passive four-channel architecture, which exists in space teleoperation are put forward, one is dual-master multilateral shared control architecture, and the other is dual-slave multilateral shared control architecture. Simulations show that these four architectures can maintain stability in the presence of large time delay.

Chapter 20, *Entropy Quad-Trees for High Complexity Regions Detection*, by Rosanne Vetro, Dan A. Simovici, and Wei Ding, introduces entropy quad-trees as structures derived from quad-trees by allowing nodes to split only when those correspond to sufficiently complex sub-domains of a data domain. Complexity is evaluated using an information-theoretic measure based on the analysis of the entropy associated to sets of objects designated by nodes. An alternative measure related to the concept of box-counting dimension is also explored. Experimental results demonstrate the efficiency of entropy quad-trees to mine complex regions. As an application, the authors used the proposed technique in the initial stage of a crater detection algorithm using digital images taken from Mars surface. Additional experimental results are provided that demonstrate the crater detection performance and analyze the effectiveness of entropy quad-trees for high-complexity regions detection in the pixel space with significant presence of noise. This work is focused on 2-dimensional image domains, but can be generalized to higher dimensional data.

Chapter 21, *Sitting Posture Recognition and Location Estimation for Human-Aware Environment*, by Yusuke Manabe and Kenji Sugawara, presents that the realization of human-computer symbiosis is a very important idea in the context of ubiquitous computing. Symbiotic Computing is a kind of concept to bridge the gap between situation in Real Space (RS) and data in Digital Space (DS). The purpose is mainly to develop an intelligent software application as well as to establish the next generation information platform in order to develop the symbiotic system. Therefore, the authors argue that it is necessary to build mutual cognition between human and system. Mutual cognition broadly consists of two functions: RS cognition and DS cognition. This chapter focuses on RS Cognition, which consists of many software functions for perceiving various situations such as events or human's activities in RS. In this study, the authors develop two perceptual functions, which are sitting posture recognition and human's location estimation for a person, as kinds of RS perception task. As the results of experiments, the authors found that their developed functions are quite competent to recognize a human's activities.

Chapter 22, *Generic Cabling of Intelligent Buildings Based on Ant Colony Algorithm*, by Yunlong Wang, and Guoming Luo, identifies that generic cabling is one of the basic foundations of intelligent buildings. Using operation flow in generic cabling, the index constraints affecting generic cabling have been evolved in this chapter. A mathematical model is built based on the ant colony algorithm with multiple constraints, and improvements were made on the original basis to extend the ant colony algorithm from the regular simple ant colony and structure to a multi-ant colony and structure. The equilibrium settlement of multiplex wiring is realized according to the introduction of the multi-ant colony model. The ant cycle model is combined to extend the optimization target from the local wiring path to the entire wiring path, and to solve the drawbacks existing in the regular ant colony algorithm and other search algorithms that take the local wiring path as the optimization target. The introduced retrospective algorithm make the ants avoid the path marked "invalid" in the subsequent search process and improves the search performance and convergence speed of the ant colony algorithm.

Chapter 23, *Potentials of Quadratic Neural Unit for Application***s**, by Ricardo Rodriguez, Ivo Bukovsky, and Noriyasu Homma, discusses the quadratic neural unit (QNU) and highlights its attractiveness for industrial applications such as for plant modeling, control, and time series prediction. Linear systems are still often preferred in industrial control applications for their solvable and single solution nature and for the clarity to the most application engineers. Artificial neural networks are powerful cognitive nonlinear tools, but their nonlinear strength is naturally repaid with the local minima problem, overfitting, and high demands for application-correct neural architecture and optimization technique that often require skilled users. The QNU is the important midpoint between linear systems and highly nonlinear neural networks because the QNU is relatively very strong in nonlinear approximation; however, its optimization and performance have fast and convex-like nature, and its mathematical structure and the derivation of the learning rules is very comprehensible and efficient for implementation. These advantages of QNU are demonstrated by using real and theoretical examples.

Chapter 24, *A Value-Based Framework for Software Evolutionary Testing*, by Du Zhang, presents that the fundamental objective in value-based software engineering is to integrate consistent stakeholder value propositions into the full extent of software engineering principles and practices so as to increase the value for software assets. In such a value-based setting, artifacts in software development such as requirement specifications, use cases, test cases, or defects, are not treated as equally important during the development process. Instead, they will be differentiated according to how much they are contributing, directly or indirectly, to the stakeholder value propositions. The higher the contributions, the more important the artifacts become. In turn, development activities involving more important artifacts should be given higher priorities and greater considerations in the development process. In this chapter, the authors propose a value-based framework for carrying out software evolutionary testing with a focus on test data generation through genetic algorithms. The proposed framework incorporates general principles in value-based software testing and makes it possible to prioritize testing decisions that are rooted in the stakeholder value propositions. It allows for a cost-effective way to fulfill most valuable testing objectives first and a graceful degradation when planned testing process has to be shortened.

Chapter 25, *Comparison of Promoter Sequences Based on Inter Motif Distance*, by A. Meera and Lalitha Rangarajan, considers that understanding how the regulation of gene networks is orchestrated is an important challenge for characterizing complex biological processes. The DNA sequences that comprise promoters do not provide much direct information about regulation. A substantial part of the regulation results from the interaction of transcription factors (TFs) with specific CIS regulatory DNA sequences. These regulatory sequences are organized in a modular fashion, with each module (enhancer) containing one or more binding sites for a specific combination of TFs. In the present work, the authors propose to investigate the inter motif distance between the important motifs in the promoter sequences of citrate synthase of different mammals. Also they use a new distance measure to compare the promoter sequences. Results reveal that there exists more similarity between organisms in the same chromosome.

This book is intended to the readership of researchers, engineers, graduate students, senior-level undergraduate students, and instructors as an informative reference book in the emerging fields of software science, cognitive intelligence, and computational intelligence. The editor expects that readers of *Advances in Abstract Intelligence and Soft Computing* will benefit from the 25 selected chapters of this book, which represents the latest advances in research in software science and computational intelligence and their engineering applications.

Yingxu Wang
University of Calgary, Canada