PROLOGUE

The adjective "intelligent" in the term "intelligent systems" is a misnomer. No one has ever claimed that an intelligent system in an engineering application possesses the kind of intelligence that allows it to *induce* new knowledge, (1) or "to contemplate its creator, or how it evolved to be the system that it is". (2) Åström and McAvoy (3) have suggested terms such as "knowledgeable" and "informed" to accentuate the fact that these software systems depend on large amounts of (possibly) fragmented and unstructured knowledge. For the purposes of this book, the term "intelligent system" always implies a computer program, and although the quotation marks around the adjective intelligent may be dropped occasionally, no one should perceive it as a computer program with attributes of human-like intelligence. Instead, the reader should interpret the adjective as characterizing a software artifact that possesses a computational procedure, an algorithm, which attempts to "model and emulate," and thus automate an engineering task that used to be carried out *informally* by a human. Whether or not this models the actual cognitive process in a human is beyond the scope of this book.

In the wide spectrum of engineering activities, collectively known as *process engineering* and encompassing tasks from product and process development through process design and optimization to process operations and control, so-called intelligent systems have played an important role. Ten years ago the broad introduction of knowledge-based expert systems created a pop culture that started affecting many facets of process engineering work. Expert systems were followed by their cousins, fuzzy systems, and the explosion in the use of neural networks. During the same period, the object-oriented programming (OOP) paradigm, one of the most successful "products" of artificial intelligence, has led to a revolutionary rethinking of programming practices, so that today OOP is the paradigm of choice in software engineering. After 10 years of work, 15 books/monographs/edited volumes, over 700 identified papers in archival research and professional journals, 65 reviews/tutorial/industrial survey papers, about 150 Ph.D. theses, and several thousand industrial applications worldwide, (4) the area of what is known as "intelligent systems" has turned from fringe to mainstream in a large number of process engineering activities. These include monitoring and analysis of process operations, fault diagnosis, supervisory control, feedback control, scheduling and planning of process operations, simulation, and process and product design. The early emphasis on tools and methodologies, originated by research in artificial intelligence, has given place to more integrative approaches, which focus more on the engineering problem and its characteristics. So, today, one does not encounter as frequently as 10 years ago conference sessions with titles including terms such as "expert systems," "knowledge-based systems," or "artificial

intelligence." Instead one sees many more mature contributions, from both the academic and industrial worlds, in mainstream engineering sessions, with significant components of what one would have earlier termed "intelligent systems." The evolving complementarity in the use of approaches from artificial intelligence, systems and control theory, mathematical programming, and statistics is a strong indication of the maturity that the area of intelligent systems is reaching.

A. The Current Setting

The explosive growth of academic research and industrial practice in the synthesis, analysis, development, and deployment of intelligent systems is a natural phase in the saga of the Second Industrial Revolution. If the First Industrial Revolution in 18th century England ushered the world into an era characterized by machines that extended, multiplied, and leveraged human *physical capabilities*, the Second, currently in progress, is based on machines that extend, multiply, and leverage human *mental abilities*. (5) The thinking man, *Homo sapiens*, has returned to its Platonic roots where "all virtue is one thing, knowledge." Using the power and versatility of modern computer science and technology, software systems are continuously developed to preserve knowledge for it is perishable, clone it for it is scarce, make it precise for it is often vague, centralize it for it is dispersed, and make it portable for it is difficult to distribute. The implications are staggering and have already manifested themselves, reaching the most remote corners of the earth and the inner sancta of our private lives. In this expanding pervasiveness of computers, intelligent systems can affect and are affecting the way we educate, entertain, and govern ourselves, communicate with each other, overcome physical and mental disabilities, and produce material wealth. Computer-based deployment of "knowledge" has been thrust by modern sociologists into the center of our culture as the force most effective in resolving inequities in the distribution of biological, historical and material inheritance. But what is the tangible evidence? Software systems have been composed to do the following: (5–7) (i) harmonize chorales in the style of Johann Sebastian Bach and automate musical compositions into new territories; (ii) write original stanzas and poems with thematic uniformity, which could pass as human creations for about half of the polled readers; (iii) compose original drawings and "photographs" of nonexistent worlds; (iv) "author" complete books.

Equally impressive are the results in engineering and science. Characterized as "knowledgeable," "informed," "expert," "intelligent," or any other denotation, software systems have expanded tremendously the scope of automation in scientific and engineering activities. (4–13)

B. The Theoretical Scope and Limitations of Intelligent Systems

So what? a skeptic may ask. Are the above examples manifestations of the computer's long-awaited, human-like intelligence? No one familiar with Gödel's theorem of incompleteness would ask such a question, (14,15) for this theorem states

that it is not possible to create a formal system that is both consistent and complete. As such, you cannot create a software system based on some sort of a formal system, i.e., a consistent set of axioms, which can reflect upon itself and discover (not invent) a new dimension of knowledge. (1)

Indeed, whenever you focus your attention on any of the so-called intelligent systems, and you take the time to learn the mechanisms they use to generate their marvelous and wondrous behavior, you come up with the anti-climactic realization that everything is quite ordinary and perfectly expectable with no surprises or mystical insights. Such reaction reminds us of how Sherlock Holmes reacted when a man questioned the brilliance of his deductive reasoning in solving one of his cases:

> Mr. Jabez Wilson laughed heavily. "Well, I never!" said he. "I thought at first that you had done something clever, but I see that there was nothing in it, after all." "Begin to think, Watson," said Holmes, "that I made a mistake in explaining. 'Omne ignotum promagnifico,' you know, and my poor little reputation, such as it is, will suffer shipwreck if I am so candid."

Similarly, Alan Turing, the father of the digital computer and creator of the Turing Test for checking the "intelligence" of a machine, put it this way:

> The extent to which we regard something as behaving in an intelligent manner is determined as much by our own state of mind and training as by the properties of the object under consideration. If we are able to explain and predict its behavior or if there seems to be little underlying plan, we have little temptation to imagine intelligence. With the same object, therefore, it is possible that one man would consider it as intelligent and another would not; the second man would have found out the rules of its behavior.

C. The Character of the Ten Paradigms

All the paradigms of intelligent systems in this volume have plans and assume extensive amounts of knowledge. As such they are ordinary computer programs and they emulate a precise computational procedure, which uses a predefined set of data. In the Aristotelian form, "all instruction given or received (by the intelligent systems) by way of argument, proceeds from preexistent knowledge." Consequently, one should see all cases put forward by the individual chapters as nothing more than paradigms for new uses of the computer. Every one of them carries out deduction from a predefined set of knowledge, using explicit reasoning strategies. The reader should not search for inductive generation of new knowledge, even when the terms "induction" and "inductive reasoning" have been loosely employed in some chapters. Instead, the reader should see each chapter as a computer-based paradigm in *capturing, articulating,* and *utilizing* various forms of knowledge. As a result, the reader will notice that the ten chapters of this volume serve as a paradigm of an integrative attitude to the modeling and processing of knowledge. Nowhere in this volume will the reader find artificial debates on the superiority of a numerical over a symbolic approach or vice versa. On the contrary, the engineering

methodologies advanced by the individual chapters indicate that *all available knowledge should be acquired, modeled, and used* within a framework that requires interaction and/or integration of processing methodologies from artificial intelligence, systems and control theory, operations research, statistics, and others.

It is this integrative attitude that today characterizes most of the work in the area of "intelligent systems for process engineering," as the editors of this volume have indicated in a recent review article. (4) It is this need for integrative approaches that has moved the applications of artificial intelligence into the mainstream of engineering activities. This is certainly the pivotal feature that characterizes the ten paradigms discussed in the subsequent chapters.

The ten chapters of this volume advance ten distinct paradigms for the use of ideas and methodologies from artificial intelligence in conjunction with techniques from various other areas. They represent the culmination of research efforts which started in 1986 at the *Laboratory for Intelligent Systems in Process Engineering* (LISPE) of the Chemical Engineering Department at MIT, and currently are spread over a half a dozen academic institutions. Each chapter, as the corresponding title indicates, is centered around two themes. The first theme (represented by the first part of a title) is drawn from the artificial intelligence techniques discussed in the specific chapter, while the second theme (represented by the second part of the title) focuses on a process engineering problem. It should be noted, though, that it is the process engineering problem, its formulation and characteristics, that sets the tone for every chapter. The various components of the corresponding intelligent systems serve specific needs. Nowhere will the reader find the "a technique in search for a problem to solve" attitude, which has led to the distortion of several engineering problems and the malignant proliferation of techniques. As a result, even if future developments suggest a change in the techniques used, the formulation of the engineering problems may retain the bulk of the essential features proposed by each of the ten chapters.

D. Themes Covered by the Ten Chapters

Let us now give a brief synopsis of the themes advanced by each of the ten chapters. The five chapters of Volume 21 (Part I) advance paradigms which are related to product and process design, while the five chapters of Volume 22 (Part II) focus on aspects of process operations.

Volume 21: Product and Process Design

Chapter 1. *MODELING LANGUAGES: Declarative and Imperative Descriptions of Chemical Reactions and Processing Systems*

To model is to represent reality, and modeling as an essential task of any engineering activity is always *contextual.* Within the scope of differing engineer-

ing contexts, the same physical entity, e.g. molecule, chemical reaction, or process flowsheet, is represented with a broad variety of models. An enormous amount of effort is expended in the development and maintenance of a *Babel of models*, sporting different languages and being at cross purposes with each other, although like their biblical counterpart they share a common progenitor—in this case, the fundamentals of chemistry/physics and the principles of chemical engineering science. Creating a language that supports the expeditious generation of consistent models has become the key to unlocking the power of computer-aided tools, and unleashing the explosive synergism between human and computer. However, a modeling language is of little use if it only creates representations of physical entities as "things unto themselves" without meaningful semantic designation to what it purports to represent. Furthermore, the model of an entity should contain all knowledge that has some bearing on the representation of that entity, be that *declarative* or *imperative* (procedural) in character. Chapter 1 describes two modeling languages; LCR (*L*anguage for *C*hemical *R*easoning) to represent molecules and chemically reactive systems, and MODEL.LA. (MODELing LAnguage) for the representation of processing systems. Both are based on the same principles and have, to a large extent, a common structure. Both have been based on ideas and techniques which originated in artificial intelligence, and both have been implemented in a similar object-oriented programming environment.

Chapter 2. *AUTOMATION IN DESIGN:*
The Conceptual Synthesis of Chemical Processing Schemes

If you really know how to carry out an engineering task, then you can instruct a computer to do it automatically. This self-evident truism can be used as the litmus test of whether a human "really" knows how to, say, design an engineering artifact. Experience has shown that engineers have been able to automate the process of design in very few instances, thus demonstrating the presence of serious flaws in (a) their understanding of how to do design and/or (b) their ability to clearly articulate the design methodology, both of which can be traced to the inherent difficulty of making the "best" design decisions. The pivotal element in automating the design process is *modeling the design process itself*, which includes the following modeling tasks: (1) modeling the *structure of design tasks* that can take you from the initial design specifications to the final engineering artifact; (2) representing the *design decisions* involved in each task, along with the assumptions, simplifications, and methodologies needed to frame and make the design decisions; (3) modeling the *state of the evolving design*, along with the underlying rationale. Chapter 2 shows how one can use ideas and techniques from artificial intelligence, e.g., symbolic modeling, knowledge-based systems, and logic, to construct a computer-implemented model of the design process itself. Using Douglas' hierarchical approach as the conceptual model of the design process, this chapter shows how to generate models of the design tasks' structure, design decisions, and the state of design, thus leading to automation of large

segments of the synthesis of chemical processing schemes. The result is a *human-aided, machine-based* design paradigm, with the computer "knowing" how the design is done, what the scope of design is, and how to provide explanations and the rationale for the design decisions and the resulting final design. Such a paradigm is in sharp contrast with the traditional *computer-aided, human-based* prototype, where the computer carries out numerical calculations and data fetching from files and databases, but has no notion of how the design is done, knowledge resting exclusively in the province of the individual human designer.

Chapter 3. *SYMBOLIC AND QUANTITATIVE REASONING: Design of Reaction Pathways through Recursive Satisfaction of Constraints*

Given a fixed, predetermined set of elementary reactions, to compose reaction pathways (mechanisms) which satisfy given specifications in the transformation of available raw materials to desired products is a problem encountered quite frequently during research and development of chemical and biochemical processes. As in the assembly of a puzzle, the pieces (available reaction steps) must fit with each other (i.e., satisfy a set of constraints imposed by the precursor and successor reactions) and conform with the size and shape of the board (i.e., the specifications on the overall transformation of raw materials to products). Chapter 3 draws from *symbolic and quantitative reasoning* ideas of AI which allow the systematic synthesis of artifacts through a *recursive satisfaction of constraints* imposed on the artifact as a whole and on its components. The artifacts in this chapter are mechanisms of catalytic reactions and pathways of biochemical transformations. The former require the construction of *direct* mechanisms, without cycles or redundancies, to determine the basic legitimate chemical transformations in a reacting system. The latter are the chemical engines of living cells, and they represent legitimate routes for the biochemical conversion of substrates to products either desired from a bioprocess or essential for cell survival. The algorithms discussed in this chapter could be used in one of the following two settings: (a) Synthesize alternative pathways of chemical/biochemical reactions as a means to interpret overall transformations which are experimentally observed. (b) Synthesize reaction pathways in the course of exploring new, alternative production route. This chapter discusses examples in both directions. Although it is concerned only with constraints on the directionality and stoichiometry of elementary reactions, the ideas can be extended to include other types of constraints arising, for example, from kinetics or thermodynamics.

Chapter 4. *INDUCTIVE AND DEDUCTIVE REASONING: The Case of Identifying Potential Hazards in Chemical Processes*

All reasoning carried out by computers is *deductive*; i.e., any software has all the necessary data, stored in various forms in a database, and possesses all the

necessary algorithms to operate on the set of data and *deduce* some results. Many researchers in the area of cognitive psychology make similar claims on the reasoning mechanisms of human beings. The fact remains, though, that both humans and machines can use very simple "algorithms" on small sets of data and produce results which could not have been visible to the "naked eye" of direct reasoning. In such cases, we tend to talk about the *inductive* capabilities of either of the two. These ideas are nowhere more prominent than in the area of *hazards identification and analysis.* One often hears, "if I knew that the conversion of A to B could be catalyzed by the presence of C then I would have foreseen the last disaster, and have done something about it," with the speaker converting a problem of *inductive* identification (i.e., induce the possibility of a hazard from the list of chemicals) into an issue of deductive reasoning. Chapter 4 demonstrates that the identification of hazards is essentially an interplay between inductive and deductive reasoning. Through inductive reasoning one attempts to generate all potential hazardous top-level events which can be justified by the presence of a set of chemicals. The reasoning is called inductive because it has the potential to generate specific knowledge that was not "visible" ahead of time. Once the potentially harmful top-level events have been identified, deductive reasoning attempts to "walk" through the processing scheme, its unit operations, and their design or operating characteristics (assumptions, or decisions), and generate the preconditions which would enable the occurrence of a specific top-level event. The inductive reasoning procedures operate on a set of chemicals and create in an *exhaustive, bottom-up* manner many alternative reaction pathways, some of which could lead to a hazard, e.g., release of large amounts of energy over a short period of time. On the other hand, the deductive reasoning procedures are *goal-directed* and operate in a *top-down* manner. Chapter 4 develops the detailed framework for the implementation of these ideas which, among other benefits, offers the following advantages: (a) Formalizing the hazards identification problem and unifying the methodological approaches at any stage of the design activities and (b) systematizing the generation and evaluation of mechanisms for the prevention of hazards, or containment of their effects.

Chapter 5. *SEARCHING SPACES OF DISCRETE SOLUTIONS: The Design of Molecules Possessing Desired Physical Properties*

Strings of letters make words. From words to verses and stanzas, a poet composes a work with its own dynamic behavior, e.g., emotional impact on the reader, which transcends the character of its components. In an analogous manner, atoms form functional groups and these in turn yield molecules with distinct behavior, e.g., physical properties. It takes a Homeric or Shakespearean genius to convert letters to an epic with a predefined desired impact. It suffices to efficiently search a space of combinatorial alternatives in order to identify the molecules which satisfy the desired constraints on a set of physical properties. Often the requisite scientific knowledge is fragmented, dispersed, and nonformalized, making the

deductive search for the desired molecules inefficient or impossible. The inductive "genius" of a scientist or engineer is needed to break the impasse in such cases. By evolution or revolution one needs to respond to tighter and shifting product specifications and identify new solvents, pharmaceuticals, imaging chemicals, herbicides and pesticides, refrigerants, polymeric materials, and many others. Chapter 5 sketches the characteristics of an intelligent, computer-aided tool to support the synthetic search for the desired molecules. With functional groups as the "letters" of an alphabet, automatic and interactive procedures compose and screen classes of potential molecules. The automatic synthesis algorithm defines and searches the space of discrete solutions (molecules) through a hierarchical sequence of the space's representations. However, one should never overestimate the effectiveness of search algorithms in locating the desired solutions. Quite frequently one needs to resort to human-driven, abductive jumps. Chapter 5 also describes how automatic search can become interwoven with effective man–machine interaction. Thus, the resulting computer-aided tool, the *Molecule Designer*, constitutes a paradigm of an intelligent system with two distinct but integrated and complementary capabilities. Examples of the synthesis of refrigerants, solvents, polymers, and pharmaceuticals illustrate the logic and features of the design procedures in the *Molecule Designer*.

Volume 22: Process Operations

Chapter 6: NONMONOTONIC REASONING: The Synthesis of Operating Procedures in Chemical Plants

The inherent difficulty of planning a sequence of actions to take you from one point to another usually increases as more obstacles are placed in your way. The number of these obstacles (constraints) that you must circumvent determines the complexity of the task, because any time you run into one of them you must *backtrack* and try an alternative step or path of steps. Such *serial* (or *linear* or *monotonic*) construction of a plan is fraught with pitfalls and repeated backtracking. The more the constraints, the more inefficient the monotonic planning. If, on the other hand, an action-step (a Clobberer) leads to the violation of a constraint, then *do not backtrack*. Take another action-step (a White Knight) which, when it precedes a Clobberer, negates the impact of the Clobberer, and you never need to backtrack. So the more constraints the more efficient your planning process. Such *nonserial* (or *nonlinear*, or *nonmonotonic*) reasoning has become the essence of all modern and efficient planners, whether they are *logic-based* and *explicit*, or *implicit* enumerators of alternative plans. The purpose of Chapter 6 is twofold: (i) To introduce the ideas of *nonmonotonic reasoning* in the planning of process operations. (ii) To demonstrate how nonmonotonic planning can be used to synthesize operating procedures for chemical processes, either off-line for standard tasks (e.g., routine start-up or shut-down), or on-line for real-time response to

large departures from desired conditions. It is shown that hierarchical modeling of process operations and operators is essential for the efficient deployment of nonmonotonic planning, and that the tractability of the resulting algorithms is strictly dependent on the form of the operators. In this regard, the modeling needs in this chapter draw heavily from the material of Chapter 1. Nonmonotonic planners handle with superb efficiency constraints on (a) the temporal ordering of operations, (b) avoidable mixtures of chemical species, and (c) bounding quantitative conditions on the state of a process. Consequently, they could be used to generate explicitly all feasible operating procedures, leaving a far smaller search space for the selection of the optimum procedure by a numerical optimizer.

Chapter 7. *INDUCTIVE AND ANALOGICAL LEARNING: Data-Driven Improvement of Process Operations*

Informed and systematic observation of naturally generated data can lead to the formulation of interesting and effective generalizations. While some statisticians believe that experimentation is the only safe and reliable way to "learn" and achieve operational improvements in a manufacturing system, other statisticians and all the empirical machine learning researchers contend that by looking at past historical records and sets of examples, it is possible to extract and generate important new knowledge. Chapter 7 draws from *inductive and analogical learning* ideas in an effort to develop systematic methodologies for the extraction of structured new knowledge from operational data of manufacturing systems. These methodologies do not require any a priori decisions/assumptions either on the character of the operating data (e.g., probability density distributions) or on the behavior of the manufacturing operations (e.g., linear or nonlinear structured quantitative models), and they make use of *instance-based learning* and *inductive symbolic learning* techniques developed in artificial intelligence. They are aimed to be complementary to the usual set of statistical tools that have been employed to solve analogous problems. Thus, one can see the material of Chapter 7 as an attempt to fuse statistics and machine learning in solving specific engineering problems. The framework developed in this chapter is quite generic and can be used to generate operational improvement opportunities for manufacturing systems (a) which are simple or complex (with internal structure), (b) whose performance is characterized by one or multiple objectives, and (c) whose performance metrics are categorical (qualitative) or continuous (real numbers). A series of industrial case studies illustrates the learning ideas and methodologies.

Chapter 8. *EMPIRICAL LEARNING THROUGH NEURAL NETWORKS: The Wave-Net Solution*

Empirical learning is an ever-lasting and ever-improving procedure. Although *neural networks* (NN) captured the imagination of many researchers as an outgrowth of activities in artificial intelligence, most of the progress was

accomplished when empirical learning through NNs was cast within the rigorous analytical framework of the *functional estimation problem*, or *regression*, or *model realization*. Independently of the name, it has been long recognized that, due to the inductive nature of the learning problem, to achieve the desired accuracy and generalization (with respect to the available data) in a dynamic sense (as more data become available) one needs to seek the unknown approximating function(s) in functional spaces of varying structure. Consequently, a recursive construction of the approximating functions at multiple resolutions emerges as a central requirement and leads to the utilization of wavelets as the basis functions for the recursively expanding functional spaces. Chapter 8 fuses the most attractive features of a NN, i.e., representational simplicity, capacity for universal approximation, and ease in dynamic adaptation, with the theoretical soundness of a recursive functional estimation problem, using wavelets as basis functions. The result is the *Wave-Net* (*Wave*let *Net*work), a multiresolution hierarchical NN with localized learning. Within the framework of a Wave-Net, where adaptation of the approximating function is allowed, we have explored the use of the L^{∞} error measure as the design criterion. One may cast any form of data-driven empirical learning within the framework of a Wave-Net to address a variety of modeling situations encountered in engineering problems, such as design of process controllers, diagnosis of process faults, and planning and scheduling of process operations. Chapter 8 discusses the properties of a Wave-Net and illustrates its use on a series of examples.

Chapter 9. *REASONING IN TIME: Modeling, Analysis, and Pattern Recognition of Temporal Process Trends*

The plain record of a variable's numerical values over time does not invoke appreciable levels of cognitive activity in a human. Although it can cause a fervor of numerical computations by a computer, the levels of cognitive appreciation of the variable's temporal behavior remain low. On the other hand, if one presents the human with a graphical depiction of the variable's temporal behavior, the level of cognition increases and a wave of reasoning activities is unleashed. Nevertheless, when the human is presented with scores of graphs depicting the temporal behavior of interacting variables, his/her reasoning abilities are severely tested. In such a case, the computer will happily continue crunching numbers without ever rising above the fray and thus developing a "mental" model, interpreting correctly the temporal interactions among the many variables. Reasoning in time is very demanding, because time introduces a new dimension with significant levels of additional freedom and complexity. While the real-valued representation of variables in time is completely satisfactory for many engineering tasks (e.g., control, dynamic simulation, planning and scheduling of operations), it is very unsatisfactory for all those tasks which require decision-making via logical reasoning (e.g., diagnosis of process faults, recovery of operations from large unso-

licited deviations, "supervised" execution of start-up or shut-down operating procedures). To improve the computer's ability to reason efficiently in time, we must first establish new forms for the representation of temporal behavior. It is the purpose of Chapter 9 to examine the engineering needs for temporal decision-making and to propose specific models which encapsulate the requisite temporal characteristics of individual variables and composite processes. Through a combination of analytical techniques, such as *scale-space filtering, wavelet-based, multiresolution decomposition of functions,* and modeling paradigms from artificial intelligence, Chapter 9 develops a concise framework that can be used to model, analyze, and synthesize the temporal trends of process operations. Within this framework, the modeling needs for logical reasoning in time can be fully satisfied, while maintaining consistency with the numerical tasks carried out at the same time. Thus, through the modeling paradigms of this chapter, one may put together intelligent systems which use consistent representations for their logical-reasoning and numerical tasks.

Chapter 10. *INTELLIGENCE IN NUMERICAL COMPUTING: Improving Batch Scheduling Algorithms through Explanation-Based Learning*

Learning comes from reflection upon accumulated experience and the identification of patterns found among the elements of past experience. All numerical algorithms used in scientific and engineering computing are based on the same paradigm: *execute a predetermined sequence of calculation tasks and produce a numerical answer.* The implementation of the specific numerical algorithm is oblivious to the experience gained during the solution of a specific problem and, in the next encounter, a different, or even the same problem is solved through the execution of exactly the same sequence of calculation steps. The numerical algorithm makes no attempt to reflect upon the structure and patterns of the results it produced, or to reason about the structure of the calculations it has performed. Chapter 10 shows that this need not be the case. By allowing an algorithm to reflect upon and reason with aspects of the problems it solves and its *own structure of computational tasks, the algorithm can learn* how to carry out its tasks more efficiently. Such *intelligent numerical computing* represents a new paradigm, which will dominate the future of scientific and engineering computing. But, in order to unlock the computer's potential for the implementation of truly intelligent numerical algorithms, the *procedural* depiction of a numerical algorithm must be replaced by a *declarative* representation of the algorithmic logic. Such a requirement upsets an established tradition and imposes new educational challenges, which most educators and educational curricula have not, as yet, even recognized. This chapter shows how one can take a branch and bound algorithm, used to identify optimal schedules of batch operations, and endow it with the ability to learn to improve its own effectiveness in locating the optimal scheduling policies for flowshop problems. Given that most batch scheduling problems are NP-hard,

it becomes clear how important it is to improve the effectiveness of algorithms for their solution. Using the Ibaraki framework, a branch and bound algorithm is declaratively modeled as a *discrete decision process.* Then explanation-based machine learning strategies can be employed to uncover patterns of generic value in the experience gained by the branch and bound algorithm from solving specific instances of scheduling problems. The logic of the uncovered patterns (i.e., new knowledge) can be incorporated into the control strategy of the branch and bound algorithm when the next problem is to be solved.

GEORGE STEPHANOPOULOS AND CHONGHUN HAN

REFERENCES

1. Stephanopoulos, G., Computers, Systems, Languages and Other Fragments. *CAST Newsletter*, Spring (1994).
2. Antsaklis, P. J. and Passino, K. M., eds., "An Introduction to Intelligent and Autonomous Control." Kluwer, Norwell, MA, 1993.
3. Åström, K. J. and McAvoy, T. J., Intelligent Control, *J. Proc. Cont.* **2**(3), 115 (1992).
4. Stephanopoulos, G. and Han, C., "Intelligent Systems in Process Engineering: A Review," *Proc. PSE*, Kyongju, Korea, 1994.
5. Kurtzweil, R., "The Age of Intelligent Machines." MIT Press, Cambridge, MA, 1990.
6. Mandelbrot, B. B., "The Fractal Geometry of Nature." W. H. Freeman, New York, NY, 1983.
7. Davis, P. J. and Hersh, R. "Descartes' Dream: The World According to Mathematics." Harcourt Brace Jovanovich, San Diego, CA, 1986.
8. Stephanopoulos, G. and Mavrovouniotis, M.L., eds., Artificial Intelligence in Chemical Engineering—Research and Development, *Comp. Chem. Eng.* **12**(9/10), (1988).
9. Stephanopoulos, G., Artificial Intelligence and Symbolic Computing in Process Engineering Design. *In* "Foundations of Computer-Aided Process Design" (J. J. Siirola, I. E. Grossmann, and G. Stephanopoulos, eds.), p. 21, Elsevier, New York, 1989.
10. Stephanopoulos, G., Artificial Intelligence: What Will Its Contributions Be to Process Control? *In* "The Second Shell Process Control Workshop" (D. M. Prett, C. E. Garcîa, and B. L. Ramaker, eds.), p. 591, Butterworths, Stoneham, MA, 1990.
11. Stephanopoulos, G., Brief Overview of AI and Its Role in Process Systems Engineering. *In* "Process Systems Engineering, Vol. I," CACHE, 1992.
12. Mavrovouniotis, M., ed., "Artificial Intelligence in Process Engineering." Academic Press, San Diego, CA, 1990.
13. Quantrille, T. E. and Liu, Y. A., "Artificial Intelligence in Chemical Engineering." Academic Press, San Diego, CA, 1991.
14. Hofstadter, D. R., "Gödel, Escher, Bach: An Eternal Golden Braid." Vintage Books, New York, 1980.
15. Penrose, R., "The Emperor's New Mind." Oxford University Press, Oxford, UK, 1989.