

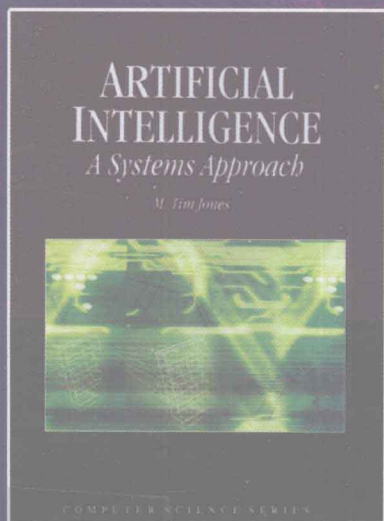
国外计算机科学教材系列

人工智能

Artificial Intelligence
A Systems Approach

英文版

[美] M. Tim Jones 著



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

国外计算机科学教材系列

人 工 智 能

(英文版)

Artificial Intelligence: A Systems Approach

[美] M. Tim Jones 著

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书包含当前人工智能 (AI) 研究的主要内容, 尤其强调实际应用, 涉及数据挖掘等许多最新应用领域。全书共 13 章, 分别讲述了 AI 的历史、不用知识的搜索、用知识的搜索、AI 与博弈、知识表示、机器学习、演化计算、神经网络 I、机器人学与 AI、智能 Agent、来自生物模型与混合模型以及 AI 语言。本书给出了算法的较详细实现, 与现有的以理论基础为核心的大多数经典人工智能著作相比, 本书有自身的鲜明特色, 且内容与国内人工智能课程的教学内容吻合, 尤其有利于培养学生解决人工智能实际问题的能力。

本书适合高等学校计算机、自动化等信息学科的本科生和研究生阅读, 也适合广大人工智能爱好者自学使用, 本书也能为人工智能研究人员了解各种算法的设计思路和具体实现框架提供参考。

Authorized reprint from the English language edition, entitled Artificial Intelligence: A Systems Approach, 978-0-7637-7337-3 by M. Tim Jones.

ORIGINAL ENGLISH LANGUAGE EDITION PUBLISHED BY

Jones and Bartlett Publishers, Inc.

40 Tall Pine Drive

Sudbury, MA 01776

COPYRIGHT 2009

ALL RIGHTS RESERVED

本书英文影印版由 Jones and Bartlett Publishers, Inc. 授予电子工业出版社。

未经出版者预先书面许可, 不得以任何方式复制或抄袭本书的任何部分。

版权贸易合同登记号 图字: 01-2009-1744

图书在版编目 (CIP) 数据

人工智能 = Artificial Intelligence: A Systems Approach: 英文 / (美) 琼斯 (Jones, M. T.) 著. - 北京: 电子工业出版社, 2009.6

(国外计算机科学教材系列)

ISBN 978-7-121-08656-4

I. 人… II. 琼… III. 人工智能 - 教材 - 英文 IV. TP18

中国版本图书馆 CIP 数据核字 (2009) 第 059254 号

责任编辑: 马 岚

印 刷: 北京市顺义兴华印刷厂

装 订: 三河市双峰印刷装订有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787 × 980 1/16 印张: 23.75 字数: 532 千字

印 次: 2009 年 6 月第 1 次印刷

定 价: 45.00 元

凡所购买电子工业出版社的图书有缺损问题, 请向购买书店调换; 若书店售缺, 请与本社发行部联系。联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

Contents^①

Chapter 1 The History of AI	1
What is Intelligence?	1
The Search for Mechanical Intelligence	2
The Very Early Days (the early 1950s)	2
Artificial Intelligence Emerges as a Field	4
AI's Winter	5
AI Re-emerges	7
AI Inter-Disciplinary R&D	9
Systems Approach	9
Overview of this Book	10
Chapter Summary	13
References	13
Resources	13
Exercises	13
Chapter 2 Uninformed Search	15
Search and AI	15
Classes of Search	15
General State Space Search	15
Trees, Graphs, and Representation	19
Uninformed Search	21
Improvements	34
Algorithm Advantages	34
Chapter Summary	34
Algorithms Summary	35
References	35
Exercises	35
Chapter 3 Informed Search	37
Informed Search	37
Best-First Search (Best-FS)	37
A* Search	42
Hill-Climbing Search	49
Simulated Annealing (SA)	50
Tabu Search	57
Constraint Satisfaction Problems (CSP)	62

① 本影印版未包含光盘。书中提及的CD-ROM上的所有内容, 均可直接从华信教育资源网(www.hxedu.com.cn) 下载。为保持与英文原书的一致性, 本影印版未改动书中关于CD-ROM的英文叙述。

Constraint Satisfaction Algorithms	64
Chapter Summary	65
Algorithms Summary	66
References	66
Resources	66
Exercises	66
Chapter 4 AI and Games.....	68
Two-Player Games	68
The Minimax Algorithm	70
Classical Game AI	80
Video Game AI	92
Chapter Summary	104
References	105
Resources	105
Exercises	106
Chapter 5 Knowledge Representation	108
Introduction	108
Types of Knowledge	108
The Role of Knowledge	108
Semantic Networks	109
Frames	110
Propositional Logic	112
First-Order Logic (Predicate Logic)	115
Semantic Web	123
Computational Knowledge Discovery	125
Ontology	126
Communication of Knowledge	126
Chapter Summary	127
References	128
Resources	128
Exercises	128
Chapter 6 Machine Learning	130
Machine-Learning Algorithms	130
Chapter Summary	147
Resources	147
Exercises	147
Chapter 7 Evolutionary Computation	149
Short History of Evolutionary Computation	149
Biological Motivation	152

Genetic Algorithms (GA)	153
Genetic Programming (GP)	162
Evolutionary Strategies (ES)	169
Differential Evolution (DE)	175
Particle Swarm Optimization (PSO)	181
Evolvable Hardware	188
Chapter Summary	188
References	189
Resources	189
Exercises	189
Chapter 8 Neural Networks I	191
Short History of Neural Networks	191
Biological Motivation	192
Fundamentals of Neural Networks	192
The Perceptron	197
Least-Mean-Square (LMS) Learning	201
Learning with Backpropagation	204
Probabilistic Neural Networks (PNN)	212
Other Neural Network Architectures	216
Tips for Building Neural Networks	218
Chapter Summary	220
References	220
Exercises	220
Chapter 9 Neural Networks II	222
Unsupervised Learning	222
Hebbian Learning	223
Simple Competitive Learning	227
K-Means Clustering	234
Adaptive Resonance Theory (ART)	241
Hopfield Auto-Associative Model	248
Chapter Summary	252
References	252
Exercises	253
Chapter 10 Robotics and AI	254
Introduction to Robotics	254
Braitenburg Vehicles	258
Natural Sensing and Control	259
Perception with Sensors	260
Actuation with Effectors	260
Robotic Control Systems	260

Simple Control Architectures	262
Movement Planning	264
Group or Distributed Robotics	266
Robot Programming Languages	267
Robot Simulators	267
Chapter Summary	267
References	267
Resources	267
Exercises	268
Chapter 11 Intelligent Agents	269
Anatomy of an Agent	269
Agent Properties and AI	271
Agent Environments	272
Agent Taxonomies	274
Agent Architectures	282
Agent Languages	294
Agent Communication	297
ACL (FIPA Agent Communication Language)	299
Chapter Summary	300
Resources	300
References	301
Exercises	301
Chapter 12 Biologically Inspired and Hybrid Models	302
Cellular Automata (CA)	302
Artificial Immune Systems	306
Artificial Life	309
Fuzzy Systems	316
Evolutionary Neural Networks	320
Ant Colony Optimization (ACO)	326
Affective Computing	332
Resources	333
Chapter 13 The Languages of AI	334
Language Taxonomy	334
Languages of AI	340
Other Languages	368
Chapter Summary	369
References	369
Resources	369
Exercises	369

Chapter

1

The History of AI

The history of AI is interesting all by itself. It's a modern-day drama, filled with excitement and anticipation, discovery, and disappointment. From over-promises of early (and later) AI research, to fears of the unknown from the general public, AI's history is worthy of study by itself. In this chapter, we'll explore AI's tumultuous history and also provide a summary introduction to each of the chapters of this book.

WHAT IS INTELLIGENCE?

To build software that is deemed intelligent, it's helpful to begin with a definition of intelligence. Intelligence can be simply defined as a set of properties of the mind. These properties include the ability to plan, solve problems, and in general, reason. A simpler definition could be that intelligence is the ability to make the right decision given a set of inputs and a variety of possible actions.

Using this simple definition of intelligence (making the right decision), we can apply this not only to humans, but also to animals that exhibit rational behavior. But the intelligence that is exhibited by human beings is much more complex than that of animals. For example, humans have the ability to communicate with language, but so do some animals. Humans can also solve problems, but the same can be said of some animals. One difference then is that humans embody many aspects of intelligence (the ability to communicate, solve problems, learn and adapt) where animals typically embody a small number of intelligent characteristics, and usually at a much lower level than humans.

We can use the same analogy on AI applied to computer systems. For example, it's possible to build an application that plays a world-class game of Chess, but this program knows nothing of the game of Checkers, nor how to make a good cup of tea. A data mining application can help identify fraud, but can't navigate a complex environment. From this perspective, the most complex and intelligent applications can be deemed intelligent from one perspective, but lack even the simplest intelligence that can be seen in the least intelligent of animals.



Famed author Isaac Asimov once wrote about his experience with aptitude tests in the army. In the army, he scored well above the norm. But what he realized was that he could score well on tests that were developed by others that shared his academic bents. He opined that if the tests were developed by people involved in auto repair, he would have scored very poorly. The issue being that tests are developed around a core of expertise, and scoring poorly on one doesn't necessarily indicate a lack of intelligence.

THE SEARCH FOR MECHANICAL INTELLIGENCE

History is filled with stories of the creation of intelligent machines. In the 800s BC, the *Iliad* described the winged Talos, a bronze automaton forged by Hephaestus to protect Crete. The inner workings of Talos weren't described, except that he was bronze, and filled with ichor (or a Greek god's blood). A more recent example is Mary Shelley's *Frankenstein*, in which the scientist recreates life from old. In 1921, Karel Capek's play "Rossum's Universal Robots" introduced the concept of cheap labor through robotics.

But one of the most interesting applications of artificial intelligence, in a non-robotic form, was that of the HAL 9000 introduced by Arthur C. Clark in his novel "2001: A Space Odyssey." HAL was a sentient artificial intelligence that occupied the Discovery spaceship (en route to Jupiter). HAL had no physical form, but instead managed the spaceship's systems, visually watched the human occupants through a network of cameras, and communicated with them in a normal human voice. The moral behind the story of HAL was one of modern-day programming. Software does exactly what one tells it to do, and can make incorrect decisions trying to focus on a single important goal. HAL obviously was not created with Isaac Asimov's three laws of robotics in mind.

THE VERY EARLY DAYS (THE EARLY 1950s)

While the term artificial intelligence had not yet been conceived, the 1950s were the very early days of AI. Early computer systems were being built, and the ideas of building intelligent machines were beginning to form.

Alan Turing

In 1950 it was Alan Turing who asked whether a machine could think. Turing not long before had introduced the concept of his universal abstract machine (called the *Turing Machine*) that was simple and could solve any mathematical problem (albeit with some complexity). Building on this idea, Turing wondered that if a computer's response were indistinguishable from a human, then the computer could be considered a thinking machine. The result of this experiment is called the *Turing Test*.

In the Turing test, if the machine could fool a human into thinking that it was also human, then it passed the intelligence test. One way to think of the Turing test is by communicating to the other agent through a keyboard. Questions are asked of the peer through written text, and responses are provided through the terminal. This test provides a way to determine if intelligence was created. Considering the task at hand, not only must the intelligent peer contain the necessary knowledge to have an intelligent conversation, it must be able to parse and understand natural language and generate natural language responses. The questions may involve reasoning skills (such as problem solving), so mimicking humans would be a feat!


An important realization of Turing during this period was the need to start small and grow intelligence, rather than expecting it to materialize. Turing proposed what he called the *Child Machine* in which a lesser intelligent agent would be created and then subjected to a course of education. Rather than assume that we could build an adult intelligence, we would build a child intelligence first and then inject it with knowledge. This idea of starting small and at lower levels corresponds with later ideas of

so-called “scruffy” thinkers. The human brain is complex and not fully understood, instead of striving to imitate this, why not start smaller at the child (or even smaller organism) and work our way up? Turing called this the *blank sheets* argument. A child is like a notebook that’s full of blank sheets, but is a mechanism by which knowledge is stored.

Alan Turing’s life ended at a young age, but he’s considered the founder of the field of AI (even though the moniker would not be applied for another six years).

AI, Problem Solving, and Games

Some of the earliest applications of AI focused on games and general problem solving. At this time, creating an intelligent machine was based on the belief that the machine would be intelligent if it could do something that people do (and perhaps find difficult).


 *In 1950, Claude Shannon proposed that the game of Chess was fundamentally a search problem. In fact, he was correct, but brute force search isn’t truly practical for the search space that exists with Chess. Search, heuristics, and a catalog of opening and ending moves provides a faster and more efficient way to play Chess. Shannon’s seminal paper on computer Chess produced what is called the Shannon number, or 10^{120} , which represents the lower bound of the game tree complexity of Chess. [Shannon 1950]*

The first AI program written for a computer was called “The Logic Theorist.” It was developed in 1956 by Allen Newell, Herbert Simon, and J. C. Shaw to find proofs for equations. [Newell 1956] What was most unique about this program is that it found a better proof than had existed before for a given equation. In 1957, Simon and Newell built on this work to develop the General Problem Solver (GPS). The GPS used means-end analysis to solve problems, but in general was restricted to toy problems.

Like complex math, early AI researchers believed that if a computer could solve problems that they thought were complex, then they could build intelligent machines. Similarly, games provided an interesting testbed for the development of algorithms and techniques for intelligent decision making.

In the UK at Oxford University in the early 1950s, researchers developed game-playing programs for two complex games. Christopher Strachey developed a Checkers playing program on the Ferranti Mark I. By 1952, his program could play a reasonable game of Checkers. Dietrich Prinz developed a program, again for the Ferranti Mark I, that could play Chess (mate-in-two variety). His program could search a thousand possible moves, but on this early computer, it required significant time and played very slowly.

In 1952, Arthur Samuel raised the bar for AI programs. His Checkers playing program, which ran on the IBM 701, included learning and generalization. What Samuel did with his learning Checkers program was unique in that he allowed two copies of his program to play one another, and therefore learn from each other. The result was a program that could defeat its creator. By 1962, Samuel’s Checkers program defeated the former Connecticut Checkers champion.

 *Samuel’s program, and his approach of playing copies against one another, is one of the first examples of computing survival of the fittest and the field which came to be called evolutionary computation.*

ARTIFICIAL INTELLIGENCE EMERGES AS A FIELD

By the mid 1950s, AI began to solidify as a field of study. At this point in AI's life, much of the focus was on what is called *Strong AI*. Strong AI is focused on building AI that mimics the mind. The result is a sapient entity with human-like intelligence, self-awareness, and consciousness.

The Dartmouth AI Summer Research Project

In 1956, the Dartmouth AI Conference brought about those involved in research in AI: John McCarthy (Dartmouth), Marvin Minsky (Harvard), Nathaniel Rochester (IBM), and Claude Shannon (Bell Telephone Laboratories) brought together researchers in computers, natural language processing, and neuron nets to Dartmouth College for a month-long session of AI discussions and research. The Summer research project on AI began:

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

Since then, many AI conferences have been held around the world, and on a variety of disciplines studied under the AI moniker. In 2006, Dartmouth held the "Dartmouth Artificial Intelligence Conference: The Next Fifty Years" (informally known as AI@50). The conference was well attended (even from a few that attended the first conference 50 years prior), and analyzed AI's progress and how its challenges relate to those of other fields of study.

Building Tools for AI

In addition to coining the term artificial intelligence, and bringing together major researchers in AI in his 1956 Dartmouth conference, John McCarthy designed the first AI programming language. LISP was first described by McCarthy in his paper titled "Recursive Functions of Symbolic Expressions and their Computation by Machine, Part I." The first LISP compiler was also implemented in LISP, by Tim Hart and Mike Levin at MIT in 1962 for the IBM 704.

This compiler introduced many advanced features, such as incremental compilation. [LISP 2007] McCarthy's LISP also pioneered many advanced concepts now familiar in computer science, such as trees (data structures), dynamic typing, object-oriented programming, and compiler self-hosting.

LISP was used in a number of early AI systems, demonstrating its usefulness as an AI language. One such program, called SHRDLU, provides a natural language interface to a table-top world of objects. The program can understand queries about the table-top "world," reason about the state of things in the world, plan actions, and perform some rudimentary learning. SHRDLU was designed and implemented by Terry Winograd at the MIT AI Lab on a PDP-6 computer.

LISP, and the many dialects that evolved from it, are still in wide use today. Chapter 13 provides an introduction to the languages of AI, including LISP.

The Focus on Strong AI

Recall that the focus of early AI was in Strong AI. Solving math or logic problems, or engaging in dialogue, was viewed as intelligent, while activities such as walking freely in unstable environments (which we do every day) were not.

In 1966, Joseph Weizenbaum of MIT developed a program that parodied a psychologist and could hold an interesting dialogue with a *patient*. The design of Eliza would be considered simple by today's standards, but its pattern-matching abilities, which provided reasonable responses to patient statements was real to many people. This quality of the program was troubling to Weizenbaum who later became a critic of AI because of its lack of compassion.

Constrained Applications


While much of early AI was Strong-focused, there were numerous applications that focused on solving practical problems. One such application was called the “Dendral Project,” emerging in 1965 at Stanford University. Dendral was developed to help organic chemists understand the organization of unknown organic molecules. It used as its inputs mass spectrometry graphs and a knowledge base of chemistry, making it the first known expert system.

Other constrained applications in this era include Macsyma, a computer algebra system developed at MIT by Carl Engelman, William Martin, and Joel Moses. Macsyma was written in MacLisp, a dialect of LISP developed at MIT. This early mathematical expert system demonstrated solving integration problems with symbolic reasoning. The ideas demonstrated in Macsyma eventually made their way into commercial math applications.

Bottom-Up Approaches Emerge

Early AI focused on a top-down approach to AI, attempting to simulate or mimic the higher level concepts of the brain (planning, reasoning, language understanding, etc.). But bottom-up approaches began to gain favor in the 1960s, primarily modeling lower-level concepts, such as neurons and learning at a much lower level. In 1949, Donald Hebb introduced his rule that describes how neurons can associate with one another if they are repeatedly active at the same time. The contribution of one cell's firing to enable another will increase over time with persistent firing, leading to a strong relationship between the two (a causal relationship).

But in 1957, the perceptron was created by Frank Rosenblatt at the Cornell Aeronautical Laboratory. The perceptron is a simple linear classifier that can classify data into two classes using an unsupervised learning algorithm. The perceptron created considerable interest in neural network architectures, but change was not far away.

 *Hebbian learning, perceptrons, and more advanced neural network architectures and learning algorithms are covered in the neural network Chapters 8 and 9.*

AI'S WINTER

Prior to the 1970s, AI had generated considerable interest, and also considerable hype from the research community. Many interesting systems had been developed, but these fell quite short of the predictions

made by some in the community. But new techniques such as neural networks breathed new life into this evolving field, providing additional ways for classification and learning. But the excitement of neural networks came to an end in 1969 with the publication of the monograph titled “Perceptrons.” This monograph was written by Marvin Minsky and Seymour Papert, strong advocates of Strong (or top-down) AI. The authors rightly demonstrated that single-layer perceptrons were limited, particularly when confronted with problems that are not linearly separable (such as the XOR problem). The result was a steep decline of funding into neural network research, and in general, research in AI as a field. Subsequent research would find that the multi-layer networks solved the linear separation problem, but too late for the damage done to AI.

Hardware built for AI, such as the LISP machines, also suffered a loss of interest. While the machines gave way to more general systems (not necessarily programmed in LISP), the functional languages like LISP continued to attract attention. Popular editors such as EMACS (developed during this period) still support a large user community with a scripting shell based on LISP.

Results-Oriented Applications

While there was a reduction in focus and spending in AI research in the 1970s, AI development continued but in a more focused arena. Applications that showed promise, such as expert systems, rose as one of the key developments in this era.

One of the first expert systems to demonstrate the power of rules-based architectures was called MYCIN, and was developed by Ted Shortliffe following his dissertation on the subject while at Stanford (1974). MYCIN operated in the field of medical diagnosis, and demonstrated knowledge representation and inference. Later in this decade, another dissertation at Stanford by Bill VanMelles built on the MYCIN architecture and serves as a model for the expert system shell (still in use today). In Chapter 5 we’ll provide an introduction to the representation of knowledge and inference with logic.

Other results-oriented applications included those focused on natural language understanding. The goal of systems in this era was in the development of intelligent question answering systems. To understand a question stated in natural language, the question must first be parsed into its fundamental parts. Bill Woods introduced the idea of the Augmented Transition Network (or ATN) that represents formal languages as augmented graphs. From Eliza in the 1960s to ATNs in the 1970s, Natural Language Processing (NLP) and Natural Language Understanding (NLU) continues today in the form of chatbots.

Additional AI Tools Emerge

John McCarthy introduced the idea of AI-focused tools in the 1950s with the development of the LISP language. Expert systems and their shells continued the trend with tools for AI, but another interesting development that in a way combined the two ideas resulted from the Prolog language. Prolog was a language built for AI, and was also a shell (for which expert systems could be developed). Prolog was created in 1972 by Alain Colmeraur and Phillippe Roussel based on the idea of Horn clauses. Prolog is a declarative high-level language based on formal logic. Programs written in Prolog consist of facts and rules that reason over those facts. You can find more information on Prolog in Chapter 5 Knowledge Representation and Chapter 13, The Languages of AI.

Neat vs Scruffy Approaches

A split in AI, its focus, and basic approaches was also seen during this period. Traditional, or top-down AI (also called Good-Old-Fashioned-AI, or GOFAI for short) continued during this period but new approaches began to emerge that looked at AI from the bottom-up. These approaches were also labeled *Neat* and *Scruffy* approaches segregating them into their representative camps. Those in the neat camp favored formal approaches to AI that were pure and provable. But those in the scruffy camp used methods less provable but still yielding useful and significant results. A number of scruffy approaches to AI that became popular during this period included genetic algorithms (modeling natural selection for optimization) and neural networks (modeling brain behavior from the neuron up).

Genetic algorithms became popularized in the 1970s due to the work of John Holland and his students at the University of Michigan. Holland's book on the topic continues to be a useful resource. Neural networks, while stagnant for a time after the publication of "Perceptrons," were revived with Paul John Werbos' creation of the backpropagation algorithm. This algorithm remains the most widely used supervised learning algorithm for training feedforward neural networks. You can learn more about genetic algorithms and evolutionary computation in Chapter 3 and neural networks in Chapters 8, and 9.

AI RE-EMERGES

Just as spring always follows the winter, AI's winter would eventually end and bring new life into the field (starting in the mid to late 1980s). The re-emergence of AI had significant differences from the early days. Firstly, the wild predictions of creating intelligent machines were for the most part over. Instead, researchers and AI practitioners focused on specific goals primarily in the *weak* aspects of AI (as opposed to *Strong* AI). Weak AI focused on solving specific problems, compared to Strong AI, whose goal was to emulate the full range of human cognitive capabilities. Secondly, the field of AI broadened to include many new types of approaches, for example, the biologically inspired approaches such as Ant Colony Optimization (ACO).

The Silent Return

An interesting aspect of AI's return was that it occurred silently. Instead of the typical claims of Strong AI, weak algorithms found use in a variety of settings. Fuzzy logic and fuzzy control systems were used in a number of settings, including camera auto-focus, antilock braking systems as well as playing a part in medical diagnosis. Collaborative filtering algorithms found their way into product recommendation at a popular online bookseller, and popular Internet search engines use AI algorithms to cluster search results to help make finding what you need easier.

The silent return follows what Rodney Brooks calls the "AI effect." AI algorithms and methods transition from being "AI" to standard algorithms and methods once they become practically useful. The methods described above are one example, another is speech recognition. The algorithms behind recognizing the sounds of speech and translating them into symbols were once described within the confines of AI. Now these algorithms are commonplace, and the AI moniker has long since passed. Therefore, the AI effect has a way of diminishing AI research, as the heritage of AI research becomes lost in the practical application of the methods.

Messy and Scruffy Approaches Take Hold

With AI's resurgence came different views and approaches to AI and problem solving with AI algorithms. In particular, the scruffy approaches became more widespread and the algorithms became more applicable to real-world problems. Neural networks continued to be researched and applied, and new algorithms and architectures resulted. Neural networks and genetic algorithms combined to provide new ways to create neural network architectures that not only solved problems, but did so in the most efficient ways. This is because the survival of the fittest features of the genetic algorithm drove neural network architectures to minimize for the smallest network to solve the given problem at hand. The use of genetic algorithms also grew in a number of other areas including optimization (symbolic and numerical), scheduling, modeling and many others. Genetic algorithms and neural networks (supervised and unsupervised) are covered in Chapters 7, 8, and 9.

Other bottom-up and biologically inspired approaches followed in the 1990s and beyond. In early 1992, for example, Marco Dorigo introduced the idea of using stigmergy (indirect communication in an environment, in this case, pheromones). Dorigo's use of stigmergy was applied to a variety of problems. Ant Colony Optimization (or ACO) is demonstrated with the traveling salesman problem in Chapter 12.

Also emerging out of the messy approaches to AI was a new field called Artificial Life. Artificial Life research studies the processes of life and systems related to life through a variety of simulations and models. In addition to modeling singular life, ALife also simulates populations of lifeforms to help understand not only evolution, but also the evolution of characteristics such as language. Swarm intelligence is another aspect of this that grew from ALife research. ALife is interesting in the context of AI because it can use a number of AI methods such as neural networks (as the neuro-controller of the individuals in the population) as well as the genetic algorithm to provide the basis for evolution. This book provides a number of demonstrations of ALife both in the context of genetic algorithms and neural networks.



One of the earliest simulation environments that demonstrated artificial life was the “game of life” created by John Conway. This was an example of a cellular automaton, and is explored later.

Another bottom-up approach that evolved during AI's re-emergence used the human immune system as inspiration. Artificial Immune Systems (or AIS) use principles of the immune system and the characteristics that it exhibits for problem solving in the domains of optimization, pattern recognition, and data mining. A very novel application of AIS is in computational security. The human body reacts to the presence of infections through the release of antibodies which destroy those infectious substances. Networks of computers can perform the same function, for example, in the domain of network security. If a software virus is found on a computer within a given network, other “antibody” programs can be dispatched to contain and destroy those viruses. Biology continues to be a major source of inspiration for solutions to many types of problems.

Agent Systems

Agents, which are also referred to as intelligent agents or software agents, are a very important element of modern-day AI. In many ways, agents are not an independent aspect of but instead a vehicle for AI

applications. Agents are applications that exhibit characteristics of intelligent behavior (such as learning or classification), but are not in themselves AI techniques. There also exists other agent-based methods such as agent-oriented computing and multi-agent systems. These apply the agent metaphor for solving a variety of problems.

One of the most popular forms of intelligent agents is “agency” applications. The word agency is used because the agent represents a user for some task that it performs for the user. An example includes a scheduling application. Agents representing users intelligently negotiate with one another to schedule activities given a set of constraints for each user.

The concept of agents has even been applied to the operation of a deepspace spacecraft. In 1999 NASA integrated what was called the “Remote Agent” into the Deep Space 1 spacecraft. Deep Space 1’s goal was to test a number of high-risk technologies, one of which was an agent that was used to provide autonomy to the spacecraft for limited durations of time. The Remote Agent employed planning techniques to autonomously schedule experiments based on goals defined by ground operators. Under constrained conditions, the Remote Agent succeeded in proving that an intelligent agent could be used to autonomously manage a complicated probe and satisfy predefined objectives.

Today you’ll find agents in a number of areas, including distributed systems. Mobile agents are independent agents that include autonomy and the ability to travel amongst nodes of a network in order to perform their processing. Instead of the agent communicating with another agent remotely, the mobile agent can travel to the other agent’s location and communicate with it directly. In disconnected network situations, this can be very beneficial. You can learn more about intelligent agents (including mobile agents) in Chapter 11.

AI INTER-DISCIPLINARY R&D

In many cases, AI research tends to be fringe research, particularly when it’s focused on Strong AI. But what’s notable about research in AI is that the algorithms tend to find uses in many other disciplines beyond that of AI. AI research is by no means pure research, but its applications grow well beyond the original intent of the research. Neural networks, data mining, fuzzy logic, and Artificial Life (for example) have found uses in many other fields. Artificial Life is an interesting example because the algorithms and techniques that have resulted from research and development have found their way into the entertainment industry (from the use of swarming in animated motion pictures to the use of AI in video games).

Rodney Brook’s has called this the AI effect, suggesting that another definition for AI is “almost implemented.” This is because once an AI algorithm finds a more common use, it’s no longer viewed as an AI algorithm but instead just an algorithm that’s useful in a given problem domain.

SYSTEMS APPROACH

In this book, the majority of the algorithms and techniques are studied from the perspective of the systems approach. This simply means that the algorithm is explored in the context of inputs and outputs. No algorithm is useful in isolation, but instead from the perspective of how it interacts with its environment (data sampling, filtering, and reduction) and also how it manipulates or alters its environment.

Therefore, the algorithm depends on an understanding of the environment and also a way to manipulate the environment. This systems approach illustrates the practical side of artificial intelligence algorithms and techniques and identifies how to ground the method in the real world (see Figure 1.1).

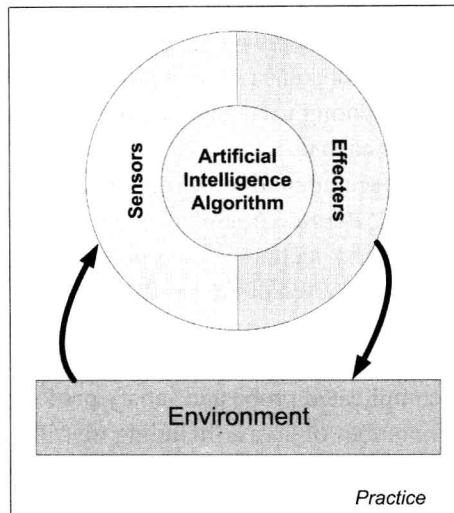


FIGURE 1.1 The systems approach to Artificial Intelligence.

As an example, one of the most interesting uses of AI today can be found in game systems. Strategy games, for example, commonly occupy a map with two or more opponents. Each opponent competes for resources in the environment in order to gain the upper hand over the other. While collecting resources, each opponent can schedule the development of assets to be used to defeat the other. When multiple assets exist for an opponent (such as a military unit), they can be applied in unison, or separately to lay siege on another opponent.

Where strategy games depend on a higher-level view of the environment (such as would be viewed from a general), first-person shooter games (FPS) take a lower-level view (from that of a soldier). An agent in an FPS depends most often on its view of the battlefield. The FPS agent's view of the environment is at a much lower level, understanding cover, objectives, and local enemy positions. The environment is manipulated by the FPS agent through its own movement, attacking or defending from enemies (through finding cover), and possibly communicating with other agents.

An obvious example of the systems approach is in the field of robotics. Mobile robots, for example, utilize an array of sensors and effects that make up the physical robot. At the core of the robot is one or more algorithms that yield rational behavior.

In each case, the AI algorithm that's chosen is the core of an agent's sensors (inputs) and effectors (outputs). For this reason, the algorithm can't truly be useful or understood unless it's considered from its place in the environment.

OVERVIEW OF THIS BOOK

This book covers a wide range of AI techniques, each segmented appropriately into their particular genre. The following chapter summaries present the ideas and methods that are explored.