Summary of :

Artificial Intelligence

A Modern Approach

Third Edition

1 Introduction

This chapter defines AI and establishes the cultural background against which it has developed. Some of the important points are as follows:

- Different people approach AI with different goals in mind. Two important questions to ask are: Are you concerned with thinking or behavior? Do you want to model humans or work from an ideal standard?
- In this book, we adopt the view that intelligence is concerned mainly with **rational action**. Ideally, an **intelligent agent** takes the best possible action in a situation. We study the problem of building agents that are intelligent in this sense.
- Philosophers (going back to 400 B.C.) made AI conceivable by considering the ideas that the mind is in some ways like a machine, that it operates on knowledge encoded in some internal language, and that thought can be used to choose what actions to take.
- Mathematicians provided the tools to manipulate statements of logical certainty as well as uncertain, probabilistic statements. They also set the groundwork for understanding computation and reasoning about algorithms.
- Economists formalized the problem of making decisions that maximize the expected outcome to the decision maker.
- Neuroscientists discovered some facts about how the brain works and the ways in which it is similar to and different from computers.
- Psychologists adopted the idea that humans and animals can be considered information-processing machines. Linguists showed that language use fits into this model.
- Computer engineers provided the ever-more-powerful machines that make AI applications possible.
- Control theory deals with designing devices that act optimally on the basis of feedback from the environment. Initially, the mathematical tools of control theory were quite different from AI, but the fields are coming closer together.
- The history of AI has had cycles of success, misplaced optimism, and resulting cutbacks in enthusiasm and funding. There have also been cycles of introducing new creative approaches and systematically refining the best ones.
- AI has advanced more rapidly in the past decade because of greater use of the scientific method in experimenting with and comparing approaches.
- Recent progress in understanding the theoretical basis for intelligence has gone hand in hand with improvements in the capabilities of real systems. The subfields of AI have become more integrated, and AI has found common ground with other disciplines.

2 Intelligent agents

This chapter has been something of a whirlwind tour of AI, which we have conceived of as the science of agent design. The major points to recall are as follows:

- An **agent** is something that perceives and acts in an environment. The **agent function** for an agent specifies the action taken by the agent in response to any percept sequence.
- The **performance measure** evaluates the behavior of the agent in an environment. A **rational agent** acts so as to maximize the expected value of the performance measure, given the percept sequence it has seen so far.
- A **task environment** specification includes the performance measure, the external environment, the actuators, and the sensors. In designing an agent, the first step must always be to specify the task environment as fully as possible.
- Task environments vary along several significant dimensions. They can be fully or partially observable, single-agent or multiagent, deterministic or stochastic, episodic or sequential, static or dynamic, discrete or continuous, and known or unknown.
- The **agent program** implements the agent function. There exists a variety of basic agent-program designs reflecting the kind of information made explicit and used in the decision process. The designs vary in efficiency, compactness, and flexibility. The appropriate design of the agent program depends on the nature of the environment.
- **Simple reflex agents** respond directly to percepts, whereas **model-based reflex agents** maintain internal state to track aspects of the world that are not evident in the current percept. **Goal-based agents** act to achieve their goals, and **utility-based agents** try to maximize their own expected "happiness."
- All agents can improve their performance through **learning**.

3 Solving problems by searching

This chapter has introduced methods that an agent can use to select actions in environments that are deterministic, observable, static, and completely known. In such cases, the agent can construct sequences of actions that achieve its goals; this process is called **search**.

- Before an agent can start searching for solutions, a **goal** must be identified and a well-defined **problem** must be formulated.
- A problem consists of five parts: the **initial state**, a set of **actions**, a **transition model** describing the results of those actions, a **goal test** function, and a **path cost** function. The environment of the problem is represented by a **state space**. A **path** through the state space from the initial state to a goal state is a **solution**.
- Search algorithms treat states and actions as **atomic** : they do not consider any internal structure they might possess.
- A general TREE-SEARCH algorithm considers all possible paths to find a solution, whereas a GRAPH-SEARCH algorithm avoids consideration of redundant paths.
- Search algorithms are judged on the basis of **completeness**, **optimality**, **time complexity**, and **space complexity**. Complexity depends on b, the branching factor in the state space, and d, the depth of the shallowest solution.
- **Uninformed search** methods have access only to the problem definition. The basic algorithms are as follows:
  - **Breadth-first search** expands the shallowest nodes first; it is complete, optimal for unit step costs, but has exponential space complexity.

  - **Uniform-cost search** expands the node with lowest path cost, g(n), and is optimal for general step costs.
  - **Depth-first search** expands the deepest unexpanded node first. It is neither complete nor optimal, but has linear space complexity. **Depth-limited search** adds a depth bound.
  - **Iterative deepening search** calls depth-first search with increasing depth limits until a goal is found. It is complete, optimal for unit step costs, has time complexity comparable to breadth-first search, and has linear space complexity.
  - **Bidirectional search** can enormously reduce time complexity, but it is not always applicable and may require too much space.
- **Informed search** methods may have access to a **heuristic** function h(n) that estimates the cost of a solution from n.
  - The generic **best-first search** algorithm selects a node for expansion according to an **evaluation function**.
  - **Greedy best-first search** expands nodes with minimal h(n). It is not optimal but is often efficient.
  - **A∗ search** expands nodes with minimal f(n) = g(n) + h(n). A∗ is complete and optimal, provided that h(n) is admissible (for TREE-SEARCH) or consistent (for GRAPH-SEARCH). The space complexity of A∗ is still prohibitive.

4 Beyond Classical Search

This chapter has examined search algorithms for problems beyond the "classical" case of finding the shortest path to a goal in an observable, deterministic, discrete environment.

- *Local search* methods such as **hill climbing** operate on complete-state formulations, keeping only a small number of nodes in memory. Several stochastic algorithms have been developed, including **simulated annealing**, which returns optimal solutions when given an appropriate cooling schedule.
- Many local search methods apply also to problems in continuous spaces. **Linear programming** and **convex optimization** problems obey certain restrictions on the shape of the state space and the nature of the objective function, and admit polynomial-time algorithms that are often extremely efficient in practice.
- A **genetic algorithm** is a stochastic hill-climbing search in which a large population of states is maintained. New states are generated by **mutation** and by **crossover**, which combines pairs of states from the population.

5 Constraint Satisfaction Problems

- **Constraint satisfaction problems** (CSPs) represent a state with a set of variable/value pairs and represent the conditions for a solution by a set of constraints on the variables. Many important real-world problems can be described as CSPs.
- A number of inference techniques use the constraints to infer which variable/value pairs are consistent and which are not. These include node, arc, path, and k-consistency.
- **Backtracking search**, a form of depth-first search, is commonly used for solving CSPs. Inference can be interwoven with search.

- The **minimum-remaining-values** and **degree** heuristics are domain-independent methods for deciding which variable to choose next in a backtracking search. The **least-constraining-value** heuristic helps in deciding which value to try first for a given variable. Backtracking occurs when no legal assignment can be found for a variable. **Conflict-directed backjumping** backtracks directly to the source of the problem.
- Local search using the **min-conflicts** heuristic has also been applied to constraint satisfaction problems with great success.