2. Problem Solving

Artificial Intelligence and Neural Network (AINN)

Part IV

Dr. Udaya Raj Dhungana
Assist. Professor
Pokhara University, Nepal
Guest Faculty
Hochschule Darmstadt University of Applied Sciences, Germany
E-mail: udaya@pu.edu.np and udayas.epost@gmail.com

Overview

- Planning- Linear and Non-linear planning
- Means-Ends Analysis
- MYCIN

Planning

- The purpose of planning is to find a sequence of actions that achieves a given goal when performed starting in a given state.

- In other words, given a set of operator instances (defining the possible primitive actions by the agent), an initial state description, and a goal state description or predicate, the planning agent computes a plan.

Simple Planning Agent

- Earlier we saw that problem-solving agents are able to plan ahead - to consider the consequences of sequences of actions - before acting.
- We also saw that a knowledge-based agents can select actions based on explicit, logical representations of the current state and the effects of actions.
- This allows the agent to succeed in complex, inaccessible environments that are too difficult for a problem-solving agent.

Problem Solving Agents + Knowledge-based Agents = Planning Agents

Simple Planning Agent

- The task of planning is the same as problem solving.
- Planning can be viewed as a type of problem solving in which the agent uses beliefs about actions and their consequences to search for a solution over the more abstract space of plans

What is a plan?

- A sequence of operator instances, such that "executing" them in the initial state will change the world to a state satisfying the goal state description.
- Goals are usually specified as a conjunction of goals to be achieved.
- Properties of planning algorithm:
  - **Soundness**: A planning algorithm is sound if all solutions found are legal plans
    - All preconditions and goals are satisfied.
    - No constraints are violated (temporal, variable binding)
  - **Completeness**:
    - A planning algorithm is complete if a solution can be found whenever one actually exists.
    - A planning algorithm is strictly complete if all solutions are included in the search space.
  - **Optimality**:
    - A planning algorithm is optimal if the order in which solution are found is consistent with some measure of plan quality.

Linear Planning

- **Basic idea:** work on one goal until completely solved before moving on to the next goal. Planning algorithm maintains goal stack.
- **Implications:**
  - No interleaving of goal achievement
  - Efficient search if goals do not interact (much)
- Advantages:
  - Reduced search space, since goals are solved one at a time
  - Advantageous if goals are (mainly) independent
  - Linear planning is sound
- Disadvantages:
  - Linear planning ma produce suboptimal solutions (based on the number of operators in the plan)
  - Linear planning is incomplete.

Linear Planning

- Sub optimal plans:
  - Result of linearity, goal interactions and poor goal ordering
  - Initial state: at(obj1, locA), at(obj2, locA), at(747, locA)
  - Goals: at(obj1, locB), at(obj2, locB)
  - Plan: [load(obj1, 747, locA); fly(747, locA, locB); unload(obj1, 747, locB); fly(747, locB, locA); Load(obj2, 747, locA); fly(747, locA, locB); unload(obj2, 747, locB)]

Non-linear Planning

- Concept of non-linear planning:
  - Use goal set instead of goal stack. Include in the search space all possible sub goal orderings.
  - Handles goal interactions by interleaving method.
- Advantages:
  - Non-linear planning is sound.
  - Non-linear planning is complete.
  - Non-linear planning may be optimal with respect to plan length (depending on search strategy employed)
- Disadvantages:
  - It takes larger search space, since all possible goal orderings may have to be considered.
  - Somewhat more complex algorithm

Non-linear Planning Algorithm

```
NLP (initial-state, goals) -
    1. State = initial state, plan=[]; goalset = goals, opstack = []
    2. Repeat until goalset is empty
        A. Choose a goal g from the goalset
        B. If g does not match state, then
            • Choose an operator o whose add-list matches goal g
            • Push o on the opstack
            • Add the preconditions of o to the goals
        C. While all preconditions of operator on top of opstack are met in
           state
            • Pop operator 0 from top of opstack
            • State = apply(0 , state)
            • Plan = [plan, 0]
```

Means-End Analysis

- A collection of search techniques that can reason either forward or backward are studied in previous chapters. In these techniques, a given problem, one direction or the other must be chosen.
- However a mixture of the two directions is appropriate.
- Such a mixed strategy would make it possible to solve the major parts of a problem first and then go back and solve the small problems that arise in “gluing” the big pieces together.
- A technique known as **Means-end analysis** allows us to do that.

Means-End Analysis

- Means–ends analysis (MEA) is a problem solving technique used commonly in AI for limiting search in AI programs.
- It is also a technique used at least since the 1950s as a creativity tool.
- MEA is also related to means–ends chain approach used commonly in consumer behavior analysis.

Means-End Analysis

- The MEA technique is a strategy to control search in problem-solving.
- Given a current state and a goal state, an action is chosen which will reduce the difference between the two.
- The action is performed on the current state to produce a new state, and the process is recursively applied to this new state and the goal state.
- Note that, in order for MEA to be effective, the goal-seeking system must have a means of associating to any kind of detectable difference those actions that are relevant to reducing that difference.
- It must also have means for detecting the progress it is making (the changes in the differences between the actual and the desired state), as some attempted sequences of actions may fail and, hence, some alternate sequences may be tried.

Means-End Analysis

- When knowledge is available concerning the importance of differences, the most important difference is selected first to further improve the average performance of MEA over other brute-force search strategies.
- However, even without the ordering of differences according to importance, MEA improves over other search heuristics by focusing the problem solving on the actual differences between the current state and that of the goal.

Means-End Analysis

Algorithm: Means-Ends Analysis

1. Compare CURRENT to GOAL. If there are no differences between them, then return.
2. Otherwise, select the most important difference and reduce it by doing the following until success or failure is signaled:
   A. Select a new operator O, which is applicable to the current difference. If there are no such operators, then return failure.
   B. Apply O to CURRENT. Generate descriptions of two states: O-START, a state in which O's preconditions are satisfied and O-RESULT, the state that would result if O were applied in O-START.
   C. If (FIRST-PART ← MEA (CURRENT, O-START)) and

   (LAST-PART ←MEA (O-RESULT, GOAL))

   are successful, then signal success and return the result of concatenating FIRST-PART, O and LAST-PART.

Means-End Analysis

- **General Problem Solver (GPS)**
  - The MEA was first introduced in 1961 by Allen Newell and Herbert A. Simon in their computer problem-solving program General Problem Solver (GPS)
- **STRIPS**
  - an automated planning computer program
- **Prodigy**
  - a problem solver developed in a larger learning-assisted automated planning project started at Carnegie Mellon University by Jaime Carbonell, Steven Minton and Craig Knoblock, is another system that used MEA.

MyCIN

- MYCIN was an early expert system designed to identify bacteria causing severe infections, such as bacteremia and meningitis, and to recommend antibiotics, with the dosage adjusted for patient's body weight — the name derived from the antibiotics themselves, as many antibiotics have the suffix "-mycin". The Mycin system was also used for the diagnosis of blood clotting diseases. MYCIN was developed over five or six years in the early 1970s at Stanford University in Lisp by Edward Shortliffe. MYCIN was never actually used in practice but research indicated that it proposed an acceptable therapy in about 69% of cases, which was better than the performance of infectious disease experts who were judged using the same criteria.

MyCIN

- MYCIN operated using a fairly simple inference engine, and a knowledge base of ~600 rules. It would query the physician running the program via a long series of simple yes/no or textual questions. At the end, it provided a list of possible culprit bacteria ranked from high to low based on the probability of each diagnosis, its confidence in each diagnosis' probability, the reasoning behind each diagnosis, and its recommended course of drug treatment.

MyCIN

- MYCIN was based on certainty factors rather than probabilities. These certainty factors CF are in the range [– 1,+1] where –1 means certainly false and +1 means certainly true. The system was based on rules of the form: IF: the patient has signs and symptoms $s1 \wedge s2 \wedge \ldots \wedge sn$, and certain background conditions $t1 \wedge t2 \wedge \ldots \wedge tm$ hold THEN: conclude that the patient had disease di with certainty CR
- The idea was to use production rules of this kind in an attempt to approximate the calculation of the conditional probabilities $p(di \mid s1 \wedge s2 \wedge \ldots \wedge sn)$, and provide a scheme for accumulating evidence that approximated the reasoning process of an expert.

MyCIN

- Practical use / Application: MYCIN was never actually used in practice. This wasn't because of any weakness in its performance. As mentioned, in tests it outperformed members of the Stanford medical school faculty. Some observers raised ethical and legal issues related to the use of computers in medicine — if a program gives the wrong diagnosis or recommends the wrong therapy, who should be held responsible? However, the greatest problem, and the reason that MYCIN was not used in routine practice, was the state of technologies for system integration, especially at the time it was developed. MYCIN was a stand-alone system that required a user to enter all relevant information about a patient by typing in response to questions that MYCIN would pose.

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

THANK YOU

End of Chapter