Lecture 4: Search

Dr. Roman V Belavkin

BIS3226

Contents

1 Introduction to Section Problems **1**

2 Types and Examples of Search Strategies **3**

3 Search in Rule-Based Systems **5**

1 Introduction to Section Problems

A Simple Search Problem

Problem 1 (Choice)**.** *Choose optimal element in the following set:*

$$\{£0, £1, £2, £3, £4, £5, £6, £7, £8, £9\}$$

Problem 2 (Search by Guessing)**.** *Find which of the following numbers I am thinking about:*

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Problem 3 (Search by Computing)**.** *Find the largest prime number in the following set:*

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Question 1. *Analyse the difference between these problems.*

Understanding the Search Problems

- The problem defines a set of possible states $b \in B$ (the problem space), on which there is a preference relation $(B, \lesssim)$, and the goal state is the top element $\top \in B$.
- There is a limited choice of actions $a \in A$ realising some states:

$$A \ni a \mapsto b \in B$$

- The goal state can usually be reached by a sequence of actions:

$$(a_1, \ldots, a_m) \in A \times \cdots \times A$$

- Action sequences that lead to the goal state 'faster' are preferred.

Complexity
For n actions, the number of all sequences of length m is

$$n^m$$

The 'Water Jugs' Problem

Problem 4 (Water Jugs)**.** • *One 8–litre jug full of water and two empty jugs: one for 3 and another for 5 litres.*

- *The goal is to share the water equally between two people:*

$$8 \mapsto 4 + 4$$

- Problem states can be represented as $b = (b_1, b_2, b_3)$, where b_1, b_2 and b_3 are the amounts of water in the jugs.
- The initial state is $(0, 0, 8)$, and the goal state is $(0, 4, 4)$.
- At each moment, there are 6 actions:

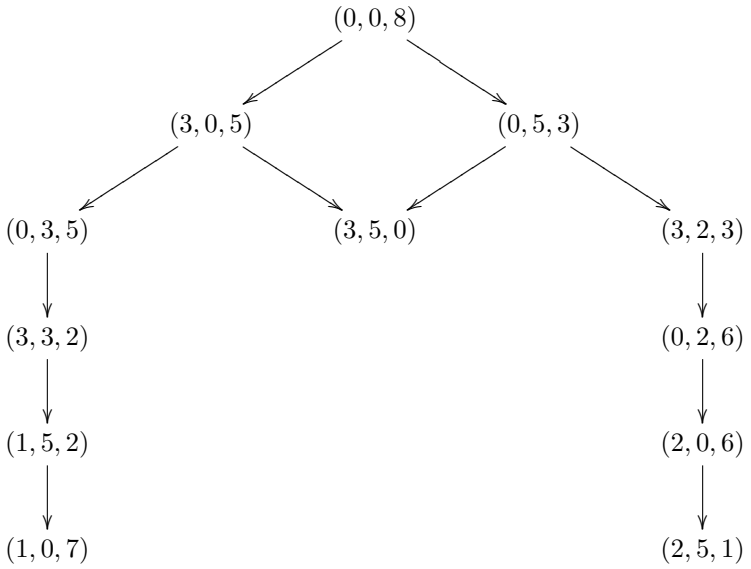
$$A = \{(b_1, b_2), (b_1, b_3)(b_2, b_1)(b_2, b_3)(b_3, b_1)(b_3, b_2)\}$$

where action $a = (b_i, b_j)$ means $b_i \mapsto b_j$.

The 'Water Jugs' Problem Space

$$(0, 8, 0)$$
$$(1, 7, 0)(0, 7, 1)$$
$$(2, 6, 0)(1, 6, 1)(0, 6, 2)$$
$$(3, 5, 0)(2, 5, 1)(1, 5, 2)(0, 5, 3)$$
$$(4, 4, 0)(3, 4, 1)(2, 4, 2)(1, 4, 3)(0, 4, 4)$$
$$(5, 3, 0)(4, 3, 1)(3, 3, 2)(2, 3, 3)(1, 3, 4)(0, 3, 5)$$
$$(6, 2, 0)(5, 2, 1)(4, 2, 2)(3, 2, 3)(2, 2, 4)(1, 2, 5)(0, 2, 6)$$
$$(7, 1, 0)(6, 1, 1)(5, 1, 2)(4, 1, 3)(3, 1, 4)(2, 1, 5)(1, 1, 6)(0, 1, 7)$$
$$(8, 0, 0)(7, 0, 1)(6, 0, 2)(5, 0, 3)(4, 0, 4)(3, 0, 5)(2, 0, 6)(1, 0, 7)(0, 0, 8)$$

Decision Tree



2 Types and Examples of Search Strategies

Simple List Search Strategies

Problem 5 (Find an Item)**.** *Find which of the following numbers I am thinking about:*

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

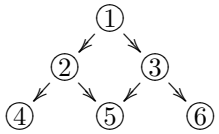
Linear search

- Check each item in the list (randomly or in order).
- For a list of n items, at most n tests are needed (complexity of order $O(n)$).

Binary (interval) search

- Check if the item is in one of the half-intervals (e.g. '*less than 5*'.
- At most $\log n$ tests are needed (complexity $O(\log n)$).

Simple Tree Search Strategies



Breadth-first search

- First, check all nodes on the same depth.
- Not good if a tree is too 'broad'.

Depth-first search

- First, check all nodes in the same branch.
- Not good if a tree is too 'deep'.

Best-first search

- First, check the most 'promising' (best) node.
- Requires the definition of a preference or utility for the nodes.

Uninformed and Informed Search

Uninformed search
Uses some fixed strategy that does not use any knowledge about a specific problem. Usually, can be applied to many problems, but the complexity of the problem can be too high.

1. Linear search;
2. Binary search;
3. Breadth-first search;
4. Depth-first search.

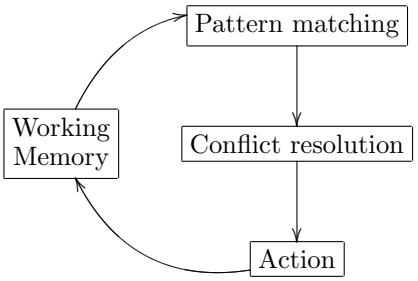
Informed search
Uses a *heuristic* based on the knowledge of a specific problem type. Heuristic can reduce the search space and complexity of the problem.

1. Best-first search;
2. A^* algorithm;
3. Dijkstra's algorithm.

3 Search in Rule-Based Systems

Recognise-Act Cycle

Reasoning in rule-based systems occurs in cycles. Each cycle consists of three stages:



Pattern Matching : (recognition) the contents of the working memory is compared with the rules. All rules that *match* the current problem state are selected into a *conflict set*.

Conflict Resolution : a single rule from the conflict set is selected.

Action : the action part of the selected rule is performed. It may result in changing the working memory.

Conflict Resolution

Sometimes, several rules can match the working memory, but only one has to be selected (hence the conflict). There are several strategies to deal with conflict resolution:

Refraction : once the rule has fired, it is not used again.

Recency : use the rule that has been used recently in such situation.

Specificity : use the rule with the more specific condition (more facts).

Priority : assign priority to rules (i.e. rank, utility, probability, cost, etc) and choose the one with the highest priority.

Parallel : fire all rules with separate lines of reasoning.

Reasoning Directions

Rule–based systems can use two directions for reasoning during problem solving:

Forward (data–driven): Start $\longrightarrow$ Goal

Backward (goal–driven): Start $\longleftarrow$ Goal

- The corresponding types of reasoning are called *forward* and *backward chaining* respectively.

Example: Weather Forecast ES

1	IF	*cyclone*	THEN	*clouds*
2	IF	*anticyclone*	THEN	*clear sky*
3	IF	*pressure is low*	THEN	*cyclone*
4	IF	*pressure is high*	THEN	*anticyclone*
5	IF	*arrow is down*	THEN	*pressure is low*
6	IF	*arrow is up*	THEN	*pressure is high*

Forward Chaining

Cycle	Working Memory	Conflict set	Rule fired
0	arrow is up	6	6
1	arrow is up, pressure is high	6, 4	4
2	arrow is up, pressure is high, anticyclone	6, 4, 2	2
3	arrow is up, pressure is high, anticyclone, clear sky	6, 4, 2	Halt

The reasoning starts from the available data (data-driven), and the working memory is matched against the left-hand-side of the rules.

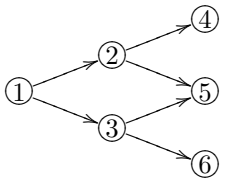
Backward Chaining

Cycle	Working Memory	Conflict set	Rule fired
0	clear sky	2	2
1	clear sky, anticyclone	2, 4	4
2	clear sky, anticyclone, pressure is high	2, 4, 6	6
3	clear sky, anticyclone, pressure is high, arrow is high	2, 4, 6	Halt

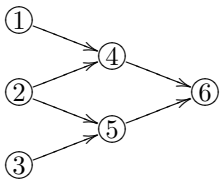
The reasoning starts from the goal state (goal-driven), and the working memory is matched against the right-hand-side of the rules.

Forward or Backward Chaining?

Forward reasoning is suitable for problems, when the number of accessible problem states *increases* (e.g. when there are more goal states than initial sates):



Backward reasoning is suitable for problems, when the number of accessible problem states *decreases* (e.g. when there are more initial states than the goal sates):



Summary

- Search problems are related to choice problems, decision-making and action selection strategies.
- The difficulty is unknown relation between actions (sequences of actions) and problem states (sequences of states):

$$A \ni a \mapsto b \in B$$

- Analysis of the search space (i.e. structure of the problem space and its relation to actions) allows us to find heuristic and make informed search.