Case-Based Reasoning (CBR)

1 Introduction

Suppose you ask a system to solve a problem that is *identical* to one that it has solved previously. If the system solves the problem again 'from scratch' and therefore takes as long to re-solve as it took to solve it the first time, you would not describe the system as intelligent.

Equally, suppose you asked a system to solve a problem that is *similar* to one that it has solved previously. You might also expect it to avoid solving 'from scratch' as much as possible.

The idea that problems can often be solved by *recalling* the way previous problems were solved, rather than by *reasoning 'from scratch'* is what underlies *Case-Based Reasoning* (CBR).

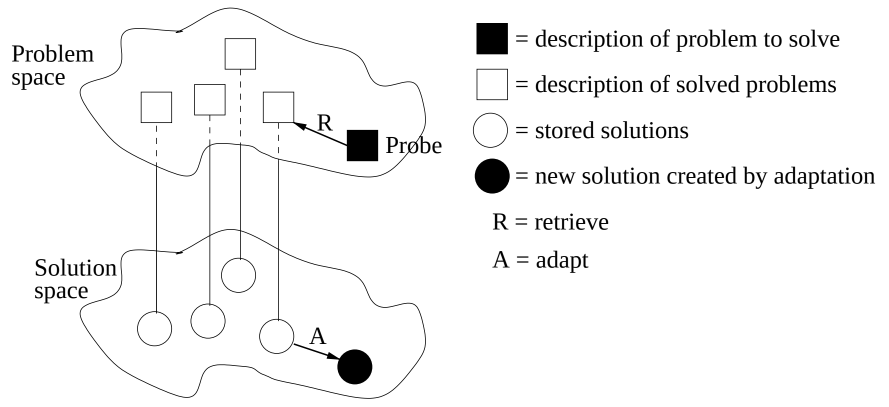
This can prove to be a highly effective problem-solving strategy in domains where the following 'slogan' holds true:

Similar problems have similar solutions.

2 A Pictorial Presentation of CBR

A system that uses CBR will have a *case base* containing a number of *cases*. Each case describes a problem that has already been solved. Each case will typically comprise at least two parts: a description of the problem, and a description of its solution.

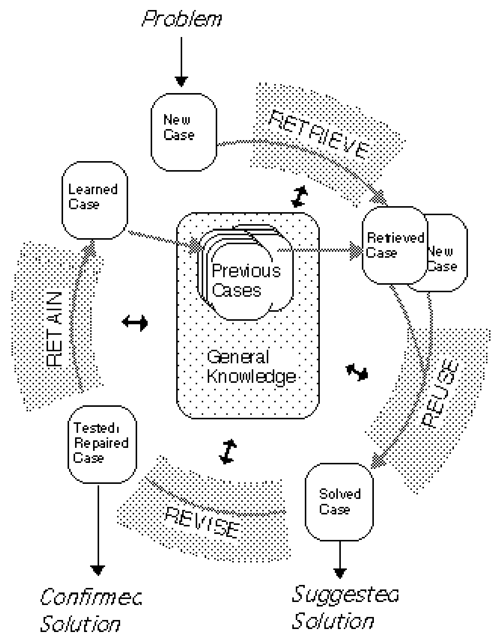
Now consider what happens when a new problem is described to the system. (The new problem is sometimes called the *probe*).


Problem space
R
Probe
Solution space
A
■ = description of problem to solve
□ = description of solved problems
○ = stored solutions
● = new solution created by adaptation
R = retrieve
A = adapt

(from Leake, D.: CBR in Context: The Present and Future, in Leake. D. (ed.), *Case-Based Reasoning: Experiences, Lessons, & Future Directions*, MIT Press, pp.3–30, 1996)

A similar problem is retrieved, along with its solution. But since this solution may not be directly applicable in the new circumstances, it is then adapted.

These two processes are situated in a broader context, referred to as the CBR-cycle:



(from Aamodt, A. & Plaza, P.: 'Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches', *Artificial Intelligence Communications*, IOS Press, vol. 7(1), pp.39-59, 1994).

The probe case is used to Retrieve a case from the case base. The retrieved case(s) are Reused to come up with a solution to the probe. (This is the process of adaptation.) The new solution is tested for success, e.g. by using it in the real world. If we're lucky, there'll be a 'teacher' (e.g. a human expert) around who can Revise it if it wasn't quite right. The probe with its revised solution can then be Retained, by being added to the case base.

Note how learning is a by-product of problem-solving. Each new problem the system tackles can be used to augment the base of experience that it will use to tackle subsequent problems.

There is much more to be said about CBR. We need to discuss at least the following: the way cases are represented, how similarity is computed, and how the adaptation is done.

Rather than looking at these issues in general terms now, we will instead look at a particular example of CBR. We will see how it has been used to build a route planning system. In this particular system, the advantages of using CBR include not only the improved problem-solving speed, but also the way it gives users 'personalised' routes.

A lot of CBR systems have been built and used in practice. Many of them leave the Reuse and Revise steps entirely to the human user. For the Retrieve step, they use the techniques we previously discussed in our lecture of k-Nearest Neighbours. (The only difference is that there we were talking about *distance* and here we are talking about *similarity*: but these are just inverses of each other.)