How to teach Deep Blue to play poker and deliver groceries

January 9 2014, by Graham Kendall

Beating computers is hard, and it's going to get harder. Credit: George Widman/AP

Deep Blue gained world-wide attention in 1997 when it defeated the then chess world champion Garry Kasparov. But playing chess was all that Deep Blue could do. Ask it to play another game, even a simpler one, such as checkers, and Deep Blue would not even know how to play at beginner level. The same is also true of many other programs that can beat humans. Computers that can play poker cannot play bridge.

This type of tailored software development is also apparent in systems that we rely on every day. A system that produces nurse rosters may not be able to cope with producing shift patterns for a factory, even though they are both personnel scheduling systems. Programs that plan delivery routes of an online supermarket cannot usually be used to schedule appointments for servicing home appliances, even though they are both examples of a Vehicle Routing Problem.

In recent years there has been a growing interest in a field called hyper-heuristics, which aims to develop more general computer systems. The idea is to build systems that are not tailored for just one type of problem, but which can be reused for a wide range of problems.

The figure below shows a typical hyper-heuristic framework. Let's assume that this framework is being used to tackle a nurse rostering problem, where we have to assign nurses to work a certain number of shifts over a certain time period, say a week.

If we start with a possible shift pattern (perhaps from the previous week), we can do certain things to improve it. For example, we could move a nurse from one shift to another, we could swap two nurses or we could remove all nurses from a certain shift (say the Wednesday evening shift) and replace them with nurses that do not meet their contractual arrangements, just to give a few examples. These changes to the shift pattern are usually called heuristics.

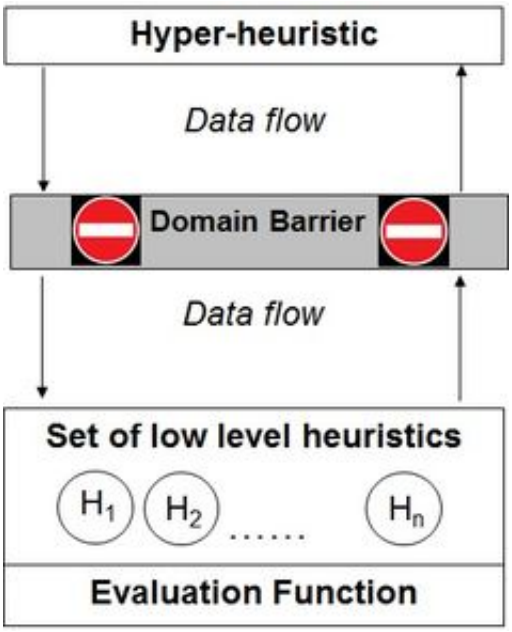
Royal Flush Credit: Images of Money

The important thing is that we have a number of these low-level heuristics that we can use to improve the current roster. All these heuristics are placed in the bottom of the framework. We now choose one of these heuristics and execute it (for instance, swap one nurse with another). We repeat the process of choosing and executing a heuristic over and over again, in the hope that we will gradually get a better roster. The quality of the roster is measured by the evaluation function, which checks the outcome.

The key to this approach is to decide in which order to execute the low-level heuristics. This is where the top part of the framework comes into play. The hyper-heuristic looks at the state of the system and decides which heuristic to execute. This is repeated until we decide to stop (maybe after a certain period of time, or after we have executed the low-level heuristics a certain number of times).

What makes a hyper-heuristic different, from other heuristic-selecting algorithms, is the "domain barrier". This stops the higher level hyper-heuristic knowing anything about the problem it is trying to solve. The hyper-heuristic only has access to data that is common to any problem. This includes how long each low-level heuristic took to execute, the

track record of each low-level heuristic (how well it has performed), how pairs of low-level heuristics work with each other, to give just a few examples.



Hyper-heuristic framework Credit: Kendall

The benefit of the domain barrier is that we can replace the low-level heuristics, and the evaluation function, with another type of problem. As the hyper-heuristic has no knowledge of the problem being tackled we would hope that we can use the same higher level algorithm to tackle this new problem. And, indeed, this has been shown to be the case in a large number of scientific problems.

The challenge in hyper-heuristics lies in developing a robust high-level strategy that is able to adapt to as many different problems as possible. We are still some way off having a hyper-heuristic that is able to produce nurse rosters, plan deliveries and play poker, but, given the pace

of progress in this field, we hope to achieve this goal in the not-too-distant future.

This story is published courtesy of The Conversation *(under Creative Commons-Attribution/No derivatives).*

Source: The Conversation