A CRITIQUE OF PROBLEM-SOLVING ABILITY

by László Lindner
Budapest

Having undertaken several tests on the problem-solving abilities of various computers and programs, and having compared them with human solving (e.g., ICCA Journal, Vol. 6, No. 3), I was naturally fascinated by Göran Grottling's paper on the same subject (ICCA Journal, Vol. 8, No. 2). First, I must pay homage to his effort in having 18 computers solve 16 problems each. The easiest of the 288 solutions took 2 seconds, but the hardest on the slowest computer over 50 hours; the average time spent for a problem must have been on the order of one hour, to which we have to add the time spent in setting up the problems and in establishing the timings which not all engines do themselves. In all, I am not aware of any such thorough and painstaking test in problem solving.

Still, I venture to question whether Grottling's method arrived at a correct ranking of solving abilities. Preliminarily it is necessary to stress that various distinct components enter into the notion of solving ability. While finite and definable, they are rather complex. The components combine, ideally, into a perfect problem-solving program. The requirements on such a program go well beyond finding one keymove in the shortest time. A realistic ranking of programs by program-solving ability must allow for all of these components, in theory no less than in practice.

This article does not set out to challenge Grottling's ranking scheme. I think it is perfectly adequate for ranking the computers' speed by the time taken to find the keymoves of Sam Loyd's 16 problems as submitted. Nor, when discussing computer chess, can we usefully introduce the 'value of a solution', as we definitely can for human solvers. Computers are unaware of human difficulties. But let us note that finding the keymove does not coincide with the full solution.

A full solution also includes the problem's variations: the threat, Black's defensive moves and White's subsequent continuation, finally leading to mate. By human convention, it is usual to record all variations up to a resultant mate-in-1. Hence, in a 2-mover the keymove suffices, but when solving three- or more-movers, the lack of variations forces us to think through a host of successive new problems, though these are shorter by one move.

TWO EXAMPLES

Two relatively simple examples from Grottling's article will serve to illustrate my point. The computer's keymove in Problem 6 correctly is 1. Bb7! However, this is a long way from the full story. The reader is required to rack his brain to see how and why this leads to mate, e.g., what is the threat embodied in this move? We have to move for Black in order for the threat to be revealed. It is only upon 2. Bb8! that the threat is apparent, because it is only then easy to understand that 3. Ba7 is unavoidable.

It is up to us to find that Black's only defense against the threat is 1. ... Bxb7! Only when we make this move, the computer will reply with 2. Nd6! with the double mating threat 2. Nxb7 and 2. Ne4 (2. ... Kd5 3. Nxb7

The solving time of Problem 8 shows similar differences. When asked for the keymove only, it took me 570 s (Grottling: 641 s), with variations time spent was 675 s. I am not sure whether Grottling had asked for the keymove only or the variations as well.

It should also be mentioned that Matebadix's solving times may vary with the sequence of setting up the pieces. I consistently followed the sequence of the pieces as shown in the article. Another interesting note: having modified the position by the moves 1. Qgl Bf3, and requesting mate in 3 moves, this took less than the difference above: 245 s with variations, 165 s without.

Quoting these variable figures may have upset the reader. Whatever the cause behind them, their very variability tends to show that the comparison of solving times is not a simple task. On the other hand, Mark V or Conchess (or rather its latest version, Mephisto B + P, which I had the opportunity to test) do not log any supplemenary time for variations. The reason for this might be that Matebadix's solutions are displayed on a screen, which takes some time, while letting the lights twinkle on the boards of Conchess or Mephisto hardly takes any.

FULL WIDTH OR FIRST HIT?

Another element to be considered when observing and evaluating comparative solving times is the difference in the programs' completeness of solution. Some will show all keymoves and all continuations available. Others, having found a keymove or some continuations to Black's defense, cease computation and so are unable to explore alternatives. It must be admitted that, even in this day and age, most computers are programmed to find one solution only. This makes the solving mode less interesting and less useful. It is essential that problems are checked for soundness, whether they have other solutions (are 'cooks') and/or whether they have other continuations ('duals'), contrary to the author's intention when composing a thematic problem.

Again, there is an essential lack of comparability between systems engaging in a full-width search and those which stop after having found a single solution. Grottling is right when stating that the time differences are due to processor rates and programming tactics. We must add a third element in computers which are content with one solution, which is the element of chance. In these engines, solving time also depends in quite an essential manner on the sequence of white moves analyzed. Even the first move investigated may well happen to be the keymove; if the computer then stops searching, the time scored will be negligible.

To be specific: assume a problem without Pawns and with castling excluded. If the algorithm searches square by square and if there are four mirror-equivalent positions, it may well happen that the four problems each have different solving times. The same may happen if the search is by pieces not by squares, in which case the sequence of setting up may again result in differences. These findings have grave consequences: when computers are incapable of full-width searching, i.e., when they do not find all keymoves and good continuations, they cannot, in principle, be realistically compared with those adopting other tactics, and not even fairly among themselves.

The 18 computers and programs tested by Grottling show three types of behaviour: