How computers will play bridge

There's been a lot of effort devoted to computer chess over the years, with impressive results. Far less effort has been devoted to computer bridge, and the results have been far more modest. In fact, they've been *so* modest that Zia has offered to play a match with a partner of his choosing against a pair of computers, with £1 million wagered on the outcome.

Zia is going to lose his bet. From a computational point of view, bridge is a slightly different problem than chess is – but not so different that recent advances in computation speed and search methodology won't lead to a serious automated bridge player in the next few years.

To answer the question posed in the title, computers are going to play bridge very differently than we do. I'll suggest an "architecture" for a computer bridge player in a few paragraphs; at this point, I'd like just to point out that computers already play *chess* very differently than we do.

A human playing chess will consider a few hundred (or perhaps only a few tens) of positions in selecting a move. The strength of a chess master lies in his ability to pick just the right handful of positions to examine. A computer, on the other hand, will consider many *millions* of positions in selecting a move. The computer's strength is in the brute force power of its search – it lacks the grandmaster's ability to select sensible moves for consideration, but compensates by examining far more possibilities than a human ever could.

This results in computers having a "style" of play that is very different from people's. Computers always seem to be hanging on to positions by the skin of their teeth; they analyze the tactical complexities without regard for the apparently dire nature of intermediate positions. They are absolutely ruthless about exploiting a technical error made by their opponents, however minor or obscure that error may be.

People have tried to make computer chess players that behave more like we do, carefully selecting relatively few positions for analysis. The performance of these programs is at best mediocre, and the performance of bridge programs that have been written from a similar perspective is likely to be similarly mediocre. The successful bridge programs will exploit the computer's speed (which is substantial) and not its common sense (which isn't).

How will these programs work? I can't say for sure, but I can make some educated guesses.

At the center of a computer bridge player will be a double-dummy analysis module. Let's say, just for the sake of argument, that I could analyze a double-dummy problem in about a second. Faced with a single dummy problem, how would I construct a line of play?

One approach would be to deal out the opponents' hands at random, consistent with their bidding, and select a play based on the double-dummy solution to the hand that was produced. Of course, this particular play may be bizarre, catering to the specific distribution that I generated. But if I deal the opponents' hands 100 times, and select the play that works best on average, I should find myself taking a fairly reasonable line.

Will this be viable approach? Probably. The machine will spend about 100 seconds thinking about the hand at the beginning, but it can hang onto all of the intermediate calculations so that it doesn't have to do a lot of new analysis as the hand proceeds. If the 100 hands that are generated for the opponents are representative, it should come up with a reasonable line of play – most of the time.

Before I talk about cases where this approach will cause the computer to make a bad play, there are a few things to be said about the method in general.

From a development point of view, the approach doesn't involve any "knowledge engineering." No one has to explain to the machine about finesses, elopements, guard squeezes, or what have you. It solves the double-dummy problems it generates *exactly*, by considering every possible line and every possible defense. If some fancy new squeeze is involved, it will be guaranteed to find it.

From a conceptual point of view, note that the machine is playing the hand one card at a time. It has no master plan such as drawing trumps and then running a side suit. (Or perhaps, as Neil Cohen suggests, it has 100 such plans and mediates among them.) At each point, it just plays the card that makes the most sense. Not the way people play the game at all!

There is a technical observation underlying this remark. It would be more effective, of course, to have the computer examine every possible master plan at each point, instead of only the next card to be laid on the table. But there are too many plans of play to enumerate this way; the number of cards that are legal to play is a relative handful.

Of course, what matters are not the conceptual or developmental issues. How good will the computer be?

From a technical point of view, it will be excellent. If one line is 53% and another 58%, it will take the 58% line whenever that line wins more frequently in its 100-hand sample. An insignificant edge (a 52.3% line as opposed to a 52.4% one) is not likely to show up in the sample, but any edge of practical value can be expected to.

The program's opening leads are likely to be devastating for similar reasons. Defending 3NT, should you set up your suit because the ♣K is likely to be an entry, or try to lead partner's best suit instead? The machine will know, and will lead accordingly.

The program will falsecard very well, treating as completely equivalent cards that are. It will be hard to *fool* with falsecards, since it will make analogous assumptions about its opponents' plays.

Given all these arguments, what's wrong with this approach? The short answer is that the machine will assume, as it plays each hand, that it will know the locations of the opponents' high cards. After all, it knows in each double-dummy instance where the high cards are!

As an example, the computer will assume that it can always play KJ10x opposite A98x for no losers; wherever the queen is, it can pick it up. What's more, it won't mind playing this suit by starting with the 9 (for example); you and I might play the ace first to cater to a singleton queen on our right but the machine will assume that it can cater to the singleton queen simply by playing the king and dropping it!

Worse still are the related mistakes that the machine will make on defense. In the above example, it won't have any qualms about unguarding the queen of the key suit – or even discarding it! After all, it will be assuming that its opponent is playing double dummy as well, and that the queen is essentially a useless card.

Before we write the program off, however, I should point out two things. First, how important these problems are is something that can only be determined at the table. The approach that I've outlined here has different strengths and different weaknesses than human players do. The relative merits of these strengths and weaknesses is something that can only be assessed in practice.

Second, the problems that I have described are hardly insurmountable. It is possible to use the card-by-card analysis at which the machine excels to identify lines of play that show promise, and to then require that a particular line of play not change until the opponents have played a revealing high card. Leading the 9 with KJ10x opposite A98x, you have to decide in advance whether or not you will play the king. If you decide not to, you can only change your mind if West plays the queen himself. The difficulty will be defining for the machine just what a "revealing high card" is.

Bidding Here is where we're really going to be in trouble.

Computers will bid using bidding systems that they invent. You and I will be able to follow their basic structure (it's going to be a lot more effective than BWS), but not the details.

For starters, we can take our second-per-hand double dummy analyzer and build a library of about a million hands. We'll use those to generate the basic bidding system. Is a 2♡ bid more effective if used constructively, or as a preempt? By considering the impact of each choice over a million hands, the machine will know. If 2♡ is a preempt, should it show hearts or some other hand type? What's the most effective way to disrupt a strong club auction? A computer can be expected to produce optimal or near-optimal answers to all of these questions. Is an ace really worth about as much as two queens? Just how much – exactly – are a QJ in one suit worth relative to a Q and J in separate suits?

It seems likely that the bidding systems we've devised over the years are tremendously inefficient; it's probably possible to use very carefully designed mechanisms to obtain comparable accuracy but using fewer bids. The bids that remain can be put to purely destructive uses – if Zia thought *he* bid a lot, you probably haven't seen anything yet.

In this area, at least, we can expect to keep up with the computers. A bidding system they design in advance will be something we can understand and use ourselves if we are so inclined. Yes, it will be highly artificial and tough to memorize. But we will still be able to model our own auctions on it if we so choose.

The tough part comes late in the auction, when the computers' million-hand sample is no longer statistically significant. Perhaps one hand has shown 19+ HCP with an unspecified 6-card major and the other has shown two aces and a queen with 3-3-2-5 exactly. We're at the level of 4♣. What next?

What's next for the computer is that it generates 100 hands meeting these patterns. It uses its double dummy analyzer to find the best contract in each. And using those results, it designs – on the fly – a bidding system to use from this point in the auction onward.

The reason this approach is viable is that the computer knows that its partner will generate the *same* sample of 100 hands from which the system is constructed. In order for the bidding systems used by the two machines to agree, all that is needed is for the random number generators used by the two computers to be synchronized. Note that the computer playing South does *not* generate either its sample hands or its system using information about the cards it actually holds; it only requires the sample hands to conform to the bidding to date. Should 4NT be Blackwood or key-card Blackwood? Not only will the machine make a sensible choice, but its partner will make the same choice – every time. Is a pass forcing, or not? Does a 5$\diamondsuit$ cue bid show first or second round control, and in what suit? It will be very discouraging playing against a partnership that invents its bidding system as the hand progresses, and never has a misunderstanding while doing so.

The machines will need to explain their conventions to their opponents, but this is unlikely to be a problem – provided that the length of the explanation is not a deterrent. A 5♣ call might well mean, "Either a singleton heart and the king of spades, or a heart void without the king of spades, or both minor suit queens, or the KJ of clubs." The language used for the explanation will be an English translation of the language generated internally by the machines.[1]

How long have we got left? For all of this to work, we need a program that can solve a single double-dummy problem in about a second. This is no small order – it seems that there are about 4 possible plays at each point (remember that a lot of the time you're just following suit), and there are 52 cards played during the course of the hand. So the number of possible play sequences that need to be analyzed is about $4^{52} \approx 10^{31}$. A personal computer can probably analyze about 10,000 of these each second, so that it should need 10^{27} seconds to analyze a complete hand – well over the age of the universe!

Using basic techniques from artificial intelligence, however, it's not too hard to reduce the "branching factor" (the number of legal moves at each point) from 4 to about 1.7. Bridge Baron, for example, can play a 5 or 6-card ending on a double-dummy basis in about 10 seconds, which does indeed correspond to a branching factor of 1.7. With this branching factor, a complete hand can be analyzed in a mere three years. Perhaps this is what the editors of *The Bridge World* meant when they remarked in the August, 1994 issue that, "A program that could ... do no more than tell whether a contract was laydown ... would be very far along the road to being able to play well. We don't expect to see this any time soon."

Unfortunately for us, artificial intelligence has more tricks up its sleeve than are used by the people who wrote Bridge Baron. It's not too hard to use published results to get the branching factor down to 1.4, so that a 5 or 6-card ending can be analyzed in about .1 second and a complete hand in about seven minutes – just four hundred times slower than is needed for effective play.

Results from my own research can bring the branching factor down to 1.27, and the time for a single hand down to about thirty seconds. Another slight improvement – to a branching

[1]Getting the machines to generate a system acceptable to the ACBL will, of course, be considerably more of a challenge.

factor of 1.2 instead of 1.27 – is all that is needed to analyze a hand in about a second. At the very worst, computers can be expected to be 30 times faster than they are now in about eight years. Either way, our time as world bridge champions is limited.