Decision making algorithms for board or card games (with and without perfect information)

Piotr Beling

May 12, 2017

Decision making algorithm is an algorithm which tries to calculate the best move possible to make in a given position in a game. In other words, for given input position, it points the move which is possible the best one. For instance, in this tic-tac-toe position shown in Figure 1, it should chose the move in the bottom-right corner, because every other can result in its defeat. A good decision making algorithm is a crucial element of each game-playing program.

Games like tic-tac-toe, chess, or draughts are examples of games with perfect information. In such games the players always have complete knowledge of the current situation position in the game. Nothing is hidden from them – they can see the whole board, and know every detail of the position they are in. The player knows exactly what moves are available to their opponent, and hence the process of making a decision is conceptually easy – they just need to calculate the best response for every possible move their opponent can make. For instance, in our tic-tac-toe example (Figure 1), if we went to middle-left, our opponent would easily win. In our example we had to predict only two moves ahead, but usually deeper analyses are required. And a number of scenarios[1] grows exponentially

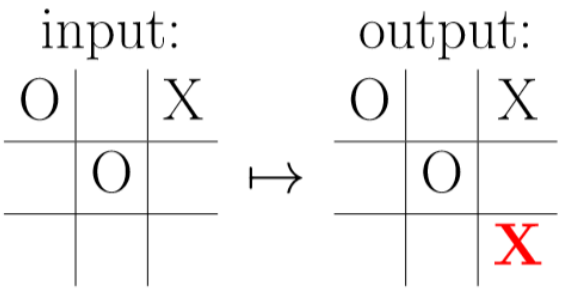

Figure 1: An example of an input position and an output move of decision making algorithm for tic-tac-toe.

[1] Figure 2 shows beginning of these scenarios in tic-tac-toe (with reflections and rotations omitted).

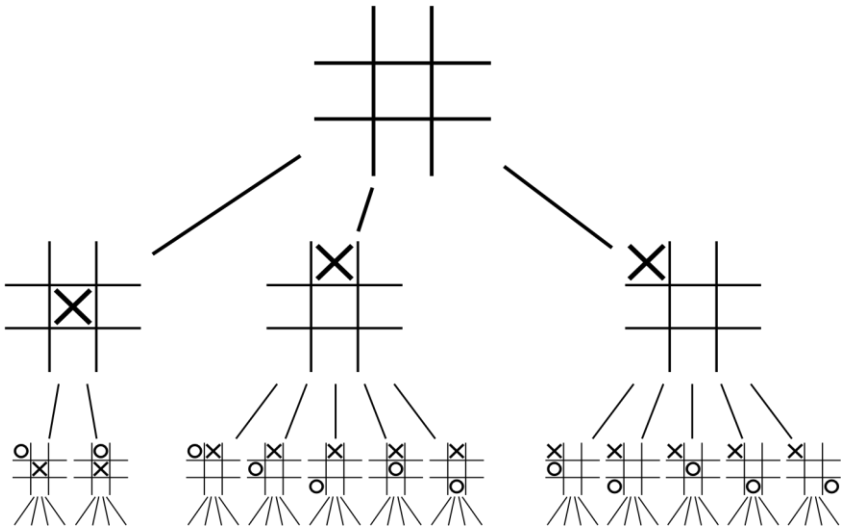

Figure 2: The first two plies of the game tree for tic-tac-toe. Since reflections and rotations are skipped, there are only three opening moves – a corner, a side, and the middle. Image has been retrieved from *Wikipedia* http://en.wikipedia.org/wiki/File:Tic-tac-toe-game-tree.svg

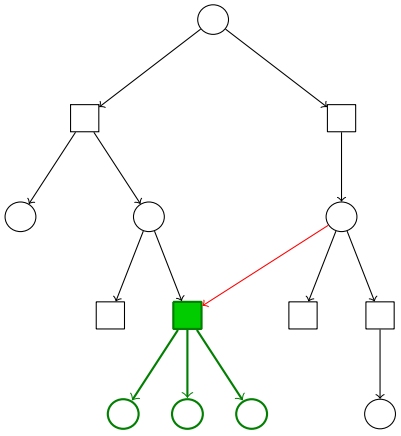

Figure 3: Often the same position can occur after several differing move sequences. This is called a transposition. Transposition table can be used to store and reuse the results obtained for positions analyzed so far. It enables the search algorithm to avoid multiple searches of the same position.

with the number of moves ahead we wish to consider. Also the calculation time grows exponentially. There might be much more possible courses of the game than atoms in the planet Earth. For this reason so called minimax algorithm, which naively considers all possible scenarios, is not suitable for most games. Luckily, there are many techniques which can significantly reduce the search space and speed up the calculations (but some of them give approximate results only). Some of the most important are: α-β punning, depth limited search (with heuristic evaluation), transposition tables (see Figure 3), and Monte Carlo Tree Search (MCTS). Note that the algorithms enumerated above are game-agnostic. It means that they can be applied to many different games. This stands in contrast to game-specific algorithms, for instance using rotational symmetries in tic-tac-toe to equate some moves (see Figure 2). On the whole, acceptable results (and strong playing programs) are obtained for most games with perfect information.

In games without perfect information some information has been hidden away from a player, like the cards of their opponents during a game of bridge. Under such circumstances, the player does not even know the precise position of the game, and, therefore, cannot be sure exactly what actions their adversaries can or cannot take. Basically, player cannot known which cards their opponent can play, because they even do not know which cards their opponent has. This makes choosing the best move nontrivial, even in theory.

Perfect Information Monte Carlo is one of the best-known and widely used methods for dealing with imperfect information games, especially card games. This method was put forward in 1989 by David Levy in the article “The million pound bridge program.” Figure 4 shows an example. In order to choose the best move possible to make in a given position, the method “imagines” a number of hypothetical, full pictures of this position. Each of these pictures are constructed by setting unseen information (like placement of hidden cards) at random, but with taking player’s previous observations under consideration, which is important since some facts about the current situation can be deduced from past decisions of other competitors. In each of these imagined scenarios, the result of taking each action (playing each card) is computed under the assumption that all participants have perfect information, that is, as if everybody put their cards on the table face up. In the end, the move that was the best on average is chosen.

The strength of the algorithm tends to increase with the numbers of scenarios which are considered, but this is a tradeoff between speed and predictive power. The more situations we consider, the more perfect information games we have to solve. Since time for decision is typically limited, we often need to compromise. However, if we speed up calculating the solution to prefect information games, we can consider more of them before the time for a decision runs out. That is why solving prefect information games as efficiently as possible is crucial.

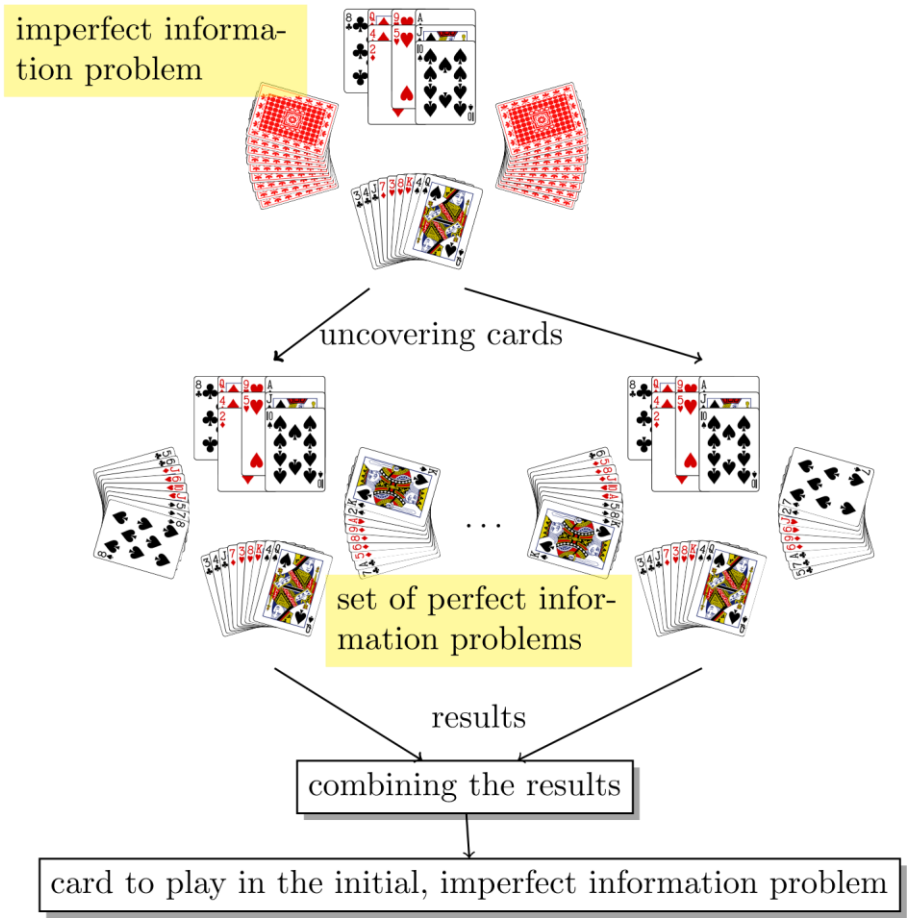

Figure 4: An application of Perfect Information Monte Carlo to the card game of bridge. In order to choose the card to play in imperfect information game, the algorithm generates a set of perfect information problems by (typically random) choice of defenders' cards and assumption that everybody can see one another's cards. Next, the number of tricks possible to win after playing each card is computed for all these problems. Finally, the card which gives the most tricks on average is chosen. (This idea was put forward in 1989 by David Levy in the article "The million pound bridge program.")