Heuristic:

The evaluation function is an important part of the AI ​​in the game of the game. It determines the decision-making steps of the AI. The main task is to evaluate the importance of the node. In generally terms, the evaluation function evaluates the current form of the game for the AI. It also simulates the form of the game after each step of decision for AI and makes the most favorable step for AI.

The most important step in the design of the evaluation function is also a very complicated part, which is a process of building a mathematical model. Designing the evaluation function requires some knowledge of the game of Double Card. In the process of Double Card’s game, the following situations will occur.

Assuming O means own, @ means the other side or boundary, X means empty point.

Connect Four: OOOOO. This situation indicates that one side has won the game. When this happens, we rate the current situation as infinity (100000).

Alive Three: XOOOX. One side has a three-segments connection and has two directions which are not been blocked. This situation is considered to be more likely to win. The score is 10000.

Dead Three: @OOOX, @OOXO, @OXOO, @XOOO and so on. One side has a three- segments connection and has one direction not been blocked. This situation is considered to have a big advantage, with a score of 1000.

Alive Two: XOOXX, XOXOX, XXOOX. One of the two sides have a two- segments connection and has not been blocked. In addition, another same segment can form an alive three situation. So, it means that which has a big advantage. The score is 1000(Same like Dead Three).

Dead Two: @XOOX, @XXOO, @XOXO, @OXOX and so on. One of the two sides have a two-segments connection and has one direction not been blocked. In addition, another same segment can form a dead three, which means has an advantage. Hence, the score is 500.

Not Available Three: @OOO@. Can't connect the same segments into four，The score is -100.

Not Available Two: @OO@. Can't connect the same segments into four，The score is -100.

Not Available One: @O@. Can't connect the same segments into four，The score is -100.

Special Situation:

Double Dead Three: One side has two or more dead three situation. This situation is considered to be more likely to win. The score is 10000.

Double Two: One side has two or more alive two situation. This situation is considered to be more likely to win. The score is 10000.

Dead Three and Alive Two: One side has one dead three and one alive two. This situation is considered to be more likely to win. The score is 10000.  
These are the weighting ratios of the evaluation factors we just mentioned. But only these weight ratios are not enough. We also need function to calculate the final result, in other word, which is a method that can give a score according to the current situation. According to these rules, all the pieces of the computer on the board are scored, and the sum is the unilateral score of the computer, which is named ScoreAI, and then the player's pieces are equally scored to obtain the ScoreHuman. The function is f(n)= ScoreAI – scoreHuman, which is the total score for the current situation. So in order to find out all the situation, the way is to first turn the two-dimensional board into N one-dimensional arrays in four directions (row, column, positive diagonal, negative diagonal), then we can perform a score calculation on all one-dimensional arrays.

Problem:

In the heuristic method, the more winning and failing conditions are considered. The more computing time and space it consumes, the lower the CPU’s utilization. The number of mini-max searches can be approximated to mn (m: branches in Second to last level, n: branches in last level). Although the number of searches is reduced by alpha-beta pruning, and the running speed is improved. But this is not enough. Because for the total number of searches(mn), pruning can only reduce the value of n and cannot reduce the value of m, the number of searches cannot be greatly reduced. The value of m can be greatly reduced by sorting the leaf nodes. However, due to limited time, this method is not implemented. The following solutions have been proposed to greatly improve the time efficiency of the algorithm and the utilization of the CPU:

Solution 1:

Discard isolated points. Each search only searches for the position which has other segments in the surrounding area. This can avoid searching for some obviously useless nodes and can greatly improve the overall search speed.

Solution 2:

When there is a losing or a winning game in the search process, it will return directly and no longer search. When there is a situation that must be lost or must win, it is not necessary to continue the search, and it can directly return the valuation value of the current game.

Solution3:

Use history table. Because in each iteration of the alpha-beta pruning function, the pieces in the board will only increase (decrease) one, and the scores in most areas of the board are unchanged, so the entire board is not required to have a full scan for each iteration. We can save the score of the board in the previous iteration. When each iteration, firstly scan the history table. If the one of the positions in history table is same as new position in current iteration. An history table gives a new score. This eliminates the need to rescan the entire board each time. Efficiency can be improved.

Optimization:

Randomized AI's position of card

The current algorithm will give a fixed response to human, which will result in the human will win all the time after human has wined AI once and then playing this way every time. In order to avoid this situation, we can increase the flexibility of the AI ​​by randomly selecting one of several locations with similar estimates when the AI ​​selects the position.